Προθεσμιακές Διακυμάνσεις Εκτιμημένες από την Αγορά Παραγώγων

Κωνσταντίνος Γκιώνης

Διατμηματικών Προγραμμάτων Μεταπτυχιακών Σπουδών

"Μαθηματική Προτυποποίηση σε Σύγχρονες Τεχνολογίες και την Οικονομία"

Εθνικό Μετσόβιο Πολυτεχνείο

Η εργασία κατατέθηκε για την απόκτηση του

Μεταπτυχιακού Διπλώματος Ειδικευσης ΕΜΠ

Αθήνα 2014

Contents

Introduction

Lower trading costs, ease of short-selling and creation of leverage are among the incentives for informed and experienced investors to prefer option markets. Moreover, option markets allows us to draw information about future states of an asset, as investors' views of potential future outcomes of the underlying assets are incorporated into market option prices. Therefore, one could argue, that a richer class of signals and better quality information can be mined from the option market in contrast to the stock market. In this thesis, we use this information to price forward variances, investigate their statistical properties and finally use them as a forecasting tool for real economic activity.

The essay's main purpose was for the student to engage contemporary research in the field of quantitative finance, by using the paper of Bakshi, Panayotov, and Skoulakis (2011) as the primary guide. Following them, we are using an option positioning to infer a term structure of forward variances contingent to the S&P 500 Index.

Realized variance of the returns on a positive underlying price S from time 0 to t, is defined to be the quadratic variation of $\log S$ at time t. In particular, if S is described by a process without jumps and has an instantaneous *volatility process* σ_t, then realized variance from time t to $t + \tau$ equals integrated variance, i.e.

$$\langle X \rangle_t^\tau = \int_t^{t+\tau} \sigma_u^2 du, \text{ where } X_t := \log\left(\frac{S_t}{S_0}\right)$$

and $\langle \cdot \rangle$ denotes the quadratic variation. Bakshi et al. (2011), for a constant risk-free rate r^* and the risk-neutral measure $\mathbb{Q}$, consider the exponential claim on integrated variance with price

$$H_t^{t,\tau} = e^{-r^*\tau} \mathbb{E}^{\mathbb{Q}} \left\{ e^{-\int_t^{t+\tau} \sigma_u^2 du} \middle| \mathcal{F}_t \right\},$$

to define the *forward variance* for times $t \in (t+\tau_1, t+\tau_2)$, with $\tau_1 < \tau_2$, as

$$f_t^{\tau_1,\tau_2} := \ln H_t^{t,\tau_1} - \ln H_t^{t,\tau_2}$$

The previous relationships should be viewed in contrast to the risk free discount bond at time t and the corresponding forward rate, given by

$$B_t^{t,\tau} = \mathbb{E}^{\mathbb{Q}} \left\{ e^{-\int_t^{t+\tau} r_u^2 du} \middle| \mathcal{F}_t \right\} \text{ and } g_t^{\tau_1,\tau_2} := \ln B_t^{t,\tau_1} - \ln B_t^{t,\tau_2}$$

for an instantaneous varying interest rate process r_t, which have been used traditionally to address questions in financial economics (among others see Fama and Bliss, 1987; Campbell and Shiller, 1991; Cochrane and Piazzesi, 2005). To the best of our knowledge, prior to Bakshi et al. (2011), there has been no work on addressing questions based on the term structure of forward variances.

To create the term structure we must first price the claim $H_t^{t,\tau}$. In the first chapter we present, in detail, the work of on pricing generic exponential claims on integrated variance and specialize their result as Bakshi et al. (2011) to find

$$H_t^{t,\tau} = e^{-r^*\tau} E^{\mathbb{Q}} \left\{ \sqrt{\frac{8}{7}} \sqrt{\frac{S_{t+\tau}}{S_t}} \cos\left(\arctan\left(\frac{1}{\sqrt{7}}\right) + \frac{\sqrt{7}}{2} \ln\left(\frac{S_{t+\tau}}{S_t}\right)\right) \middle| \mathcal{F}_t \right\}$$

Finally, we show how this payoff may be spanned to an investable portfolio of options as

$$H_t^{t,\tau} = e^{-r\tau} + \int_{K>S_t} \omega[K] C_t^\tau[K] dk + \int_{K<S_t} \omega[K] P_t^\tau[K] dK$$

where

$$\omega[K] = \frac{\frac{8}{\sqrt{14}} \cos\left(\arctan(1/\sqrt{7}) + (\sqrt{7}/2)\ln(K/S_t)\right)}{\sqrt{S_t} K^{3/2}}$$

and $C_t^\tau[K]$, $P_t^\tau[K]$ are the time t call and put prices of an option expiring at time $t+\tau$ with strike K.

In Chapter 2, which is the core of the thesis, we are using the formula above to create a time series of 121 observations, ranging from September 1998 to September 2008, for a term structure of forward variances consisting of maturities corresponding roughly to 19, 49, 79 and 109 days. Pricing the claim as the sum of two integrals upon a continuum of option prices raises several challenges. The first one is to gather data of option prices for each day of the time series and each maturity in the term structure. We used the OptionMetrics database to do it.

Secondly a statistical numerical estimation for the integrals must be employed, as a continuum of strikes is not available. Estimation robustness demands filtering the data for “unfair” prices and facilitating a method for extracting the risk neutral price curve (the curve of option prices across strikes) in order to integrate numerically. We present the corresponding literature in detail and also the theory on which the methods are based (i.e. splines as a statistical tool). Moreover as we misimplemented the methods proposed, yielding poor results, we tried a simpler one that to the best of our knowledge has not been tried before and seems to behave quite well.

In particular after filtering the data for zero prices, trading volume and open interest, we discarded prices creating arbitrage opportunities by violating monotonicity and/or convexity across strikes. For the remaining prices and for each date and strike we worked as follows. We fitted a cubic polynomial to the logarithm of prices using linear regression and used vega weighting to address noise encompassed in the far out-of-the-money and near in-the-money prices.

Having priced the claims $H_t^{t,\tau}$ for each date and maturity, we can easily proceed to calculate the forward variances' time series. In the final chapter we begin by investigating the statistical properties of forward variances. To familiarize the reader with the concepts driving our conclusions, we offer a brief summary of the theory of time-series and the most frequently used models. We find that there is no strong evidence against stationarity for the time series and thus we are able to use them as a forecasting tool. Moreover, after testing and comparing different models according to several information criteria, as Bakshi et al. (2011), we conclude that an ARMA(1,1)-GARCH(1,1) model is suitable for forward variances.

As an application, following Bakshi et al. (2011) we check whether forward variances predict real economic activity, which is proxied by non-farm payroll and industrial production using as predictors the constructed forward variances and the slope of Treasury yield curve measured by the difference between the ten-year and the three-month Treasury yields.

Programming in R (Core Team) (2013) played a central role in writing this thesis. Moreover, the teqniques used to manipulate data and conduct the statistical inference and analysis present an interest of their own and so commented code for every part is provided in the Appendix. Finally, there is a an extended summary in Greek accompanying the thesis.

Chapter 1

Pricing exponential claims on integrated variance

In this chapter we present the work of Carr and Lee (2009b), who in the word of Gatheral (2006) derived some of the most elegant and robust results in financial mathematics.

Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in\mathcal{T}}, \mathbb{P})$, where $\mathcal{T} = [0, +\infty] \subset \mathbb{R}$, be a filtered probability space satisfying the usual conditions, i.e.

- The probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is complete.
- The σ-algebras $\mathcal{F}_t$ contain all the sets in $\mathcal{F}$ of zero probability.
- The filtration $\mathcal{F}_t$ is right-continuous, i.e. $\forall\, t \in \mathcal{T}$ the σ-algebra $\mathbb{F}_{t+} \equiv \bigcap_{\epsilon>t} \mathcal{F}_\epsilon$ is equal to $\mathcal{F}_t$.

Let S_t denote the price at time t of the underlying asset. Assume that S_t follows a general diffusion process, i.e. S_t satisfies the following stochastic differential equation (SDE)

$$dS_t = a_t S_t dt + \sigma_t S_t d\tilde{W}t, \text{ and } S_0 > 0 \quad a.s. \tag{1.1}$$

for some $(\mathcal{F}_T, \mathbb{P})$-Brownian motion $\tilde{W}_t$, the $\mathcal{F}_t$-adapted processes a_t and the positive $\mathcal{F}_t$-adapted processes σ_t. Moreover, assume that for the instantaneous variance of the underlying share price $\sigma_t = \sigma(t, \omega)$ there exists a real bound m such that

$$\mathbb{E} \int_0^T \sigma_t^2 dt < m \in \mathbb{R} \tag{1.2}$$

This condition ensures that the Itô integral of σ_t is a martingale[1]. It is well known, that under the equivalent martingale (risk neutral pricing) measure Q, (1.1) becomes (e.g. see Shreve, 2008, Section 5.2.2)

$$dS_t = r_t S_t dt + \sigma_t S_t dW_t \text{ and } S_0 > 0 \quad a.s.^2 \tag{1.3}$$

for some $(\mathcal{F}_T, \mathbb{Q})$-Brownian motion W_t and the risk free interest rate process r_t. Note that (1.3) is an Itô process. Vital to our analysis is the logarithmic return process, denoted by

$$X_t := \log\left(\frac{S_t}{S_0}\right) = \log(S_t) - \log(S_0)$$

Applying Itô's lemma (see Øksendal, 2003, Theorem 4.1.2)[3] to $f(t,x) = \log\left(\frac{x}{S_0}\right)$ we easily compute that

$$dX_t = \left(r_t - \frac{1}{2}\sigma_t^2\right) dt + \sigma_t dW_t \tag{1.5}$$

Note that (1.5) is also an Itô process and thus the quadratic variation of X_t is proven to exist and to be equal to (Shreve, 2008, see Lemma 4.4.4)

$$\langle X \rangle_t = \int_0^t \sigma_u^2 du \tag{1.6}$$

Equation (1.6) is called the realized variance of the returns on the price S. Exponential claims on integrated variance, are claims contingent to this quantity. Carr and Lee (2009b) assume that the risk-free interest rate is zero or alternatively that

[1]Carr and Lee (2009b) assume that $\int_0^T \sigma_t^2 dt < m$ a.s., which is a weaker condition and suggests that the Itô integral is a local-martingale. Moreover, in Section 8 they drop this assumption. The condition we use is sufficient for our purposes.

[2]With the usual meaning, i.e. $\mathbb{P}(S_0 > 0) = 1 \Leftrightarrow S_0(\omega) > 0, \forall \omega \in \Omega \setminus N$, with $\mathbb{P}(N) = 0$. Note that since $\mathbb{P}$ and $\mathbb{Q}$ are equivalent probability measures an event is $\mathbb{P}$-a.s. if and only if it is $\mathbb{Q}$-a.s.

[3]If we apply the 4.1.8 rules to equation 4.1.7. we get another form of Itô's lemma

$$dY_t = \left(g_t(t, X_t) + u g_x(t, X_t) + \frac{1}{2} v^2 g_{xx}(t, X_t)\right) dt + v g_x(t, X_t) dB_t \tag{1.4}$$

for $g(t,x) \in C^2([0,\infty] \times \mathbb{R})$, $Y_t = g(t, X_t)$ and the Itô process $dX_t = udt + vdB_t$.

all prices are denominated relative to an asset that pays 1 at time T. In their setting, henceforth the Carr-Lee setting, (1.3) becomes

$$dS_t = \sigma_t S_t dW_t \text{ and } S_0 > 0 \quad a.s. \tag{1.7}$$

and (1.5) becomes

$$dX_t = -\frac{1}{2}\sigma_t^2 dt + \sigma_t dW_t \tag{1.8}$$

Finally, (1.6) remains unaltered.

1.1 Black Scholes and Correlation Immunity

We define a European payoff function, henceforth a payoff function F, corresponding to a contract expiring at time $T \in \mathcal{T}$, called the maturity, as any $\mathcal{F}_T$-measurable function. This simply means that a European payoff function is known at maturity T and onwards but not prior to it. For example the payoff function of a European call, with maturity $T \in \mathcal{T}$ and strike $K > 0$ is

$$F(S_T(\omega)) = (S_T(\omega) - K)^+ = \max(S_T(\omega) - K), 0)$$

For convenience we will suppress the ω notation. It is well known that under the risk neutral measure Q, the discounted payoff function is an $\mathcal{F}_t$-martingale and the fair value (arbitrage free) of a contract expiring at $T > t$ is

$$V(t) = \mathbb{E}^Q\left[e^{-\int_t^T r(u)du}F(S_T)\middle|\mathcal{F}_t\right] \tag{1.9}$$

Note, that for convenience, we will denote $\mathbb{E}_t^{\mathbb{Q}}[\cdot] := \mathbb{E}^{\mathbb{Q}}[\cdot|\mathcal{F}_t]$. The previous equation is also known as the risk-neutral pricing formula and in the Carr-Lee setting, where $r = 0$, has the form

$$V(t) = \mathbb{E}_t^Q\left[F(S_T)\right] \tag{1.10}$$

1.1.1 The Black-Scholes formula

Within the context of the model described in Black and Scholes (1973), known as the Black-Scholes model, the risk-free interest rate and the volatility are assumed constant, i.e. (1.5) becomes

$$dX_t = \left(r - \frac{1}{2}\sigma^2\right)dt + \sigma dW_t, \quad r, \sigma \in \mathbb{R} \tag{1.11}$$

Note that (1.11) describes a risky asset if and only if $\sigma > 0$. For what follows, we assume that it is. The risk-neutral pricing formula in this setting becomes

$$V(t) = e^{-r(T-t)}\mathbb{E}_t^Q\left[F(S_T)\right] \tag{1.12}$$

The integral equation of (1.11) is

$$X_T = X_0 + \left(r - \frac{1}{2}\sigma^2\right)T + \sigma W_T, \quad \text{where } X_0 = \log(S_0/S_0) = 0$$

Equivalently

$$X_t = \left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t$$

Subtract both sides to get

$$X_T - X_t = \left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma(W_T - W_t)$$

Given that W_t is a Brownian motion is easy to see that

$$\Big(X_T - X_t - r(T-t)\Big)\Big|\mathcal{F}_t \sim N\left(-\frac{1}{2}\sigma^2(T-t), \sigma^2(T-t)\right)$$

Notice that X_T is a random variable and that conditioned on $\mathcal{F}_t$, X_t becomes known and thus $X_t - r(T-t)$ is a constant. The variance of X_T and thus of

$$z := X_T - X_t - r(T-t)$$

is $v^2 := \sigma^2(T-t)$. In notation

$$z\big|\mathcal{F}_t \sim N\left(-\frac{1}{2}v^2, v^2\right)$$

The risk-neutral pricing formula contains the expected value of $F(S_T)$. Expressing $F(S_T)$ in terms of z will allow us to easily calculate this expectation. Note that

$$X_T - X_t = \log\left(\frac{S_T}{S_t}\right) \Rightarrow S_T = S_t e^{(X_T - X_t)}$$

and so using z observe that

$$S_T = S_t e^{r(T-t)} e^z$$

Notice that the previous equation suggests that the price at time T is the current price S_t "grown" according to the the risk-free interest rate r and randomized by the exponential of a normal variable whose parameters are a function of S_T's variance. We can easily observe now that that (1.12) becomes

$$V(t) = e^{-r(T-t)} \int_{-\infty}^{\infty} F\left(S_t e^{r(T-t)} e^z\right) \frac{1}{\sqrt{2\pi}v} e^{-\left(z+v^2/2\right)^2/(2v^2)} dz \tag{1.13}$$

Set $y = e^z$ and observe that $z = \ln(y) \Rightarrow dz = (1/y)dy$, that as $z \to -\infty \Rightarrow y \to 0$ and that as $z \to \infty \Rightarrow y \to \infty$, to get

$$V(t) = e^{-r(T-t)} \int_{0}^{\infty} F(S_t e^{r(T-t)} y) \frac{1}{\sqrt{2\pi}vy} e^{-\left(\ln(y)+v^2/2\right)^2)/(2v^2)} dy$$

Carr and Lee (2009b) define the above equation as to be the *Black-Scholes formula* and to be consistent they include the valuation of a riskless assets[4]. Note, that in their setting $r = 0$ and thus $S_T = S_t e^z = S_t y$. In particular, they define

$$F^{BS}(s, 0) := F(s), \quad \text{for } v = 0 \tag{1.14}$$

for a riskless asset and

$$F^{BS}(s, v) := \int_{0}^{\infty} F(sy, \omega) \frac{1}{\sqrt{2\pi}vy} e^{-(\ln(y)+v^2/2)^2/(2v^2)} dy, \quad \text{for } v > 0 \tag{1.15}$$

for a risky one. The kernel of the integrand is a lognormal density with parameters $\mu = -v^2/2$ and $\sigma = v$. It should be clear from the above that s in F^{BS} denotes the current price of the underlying asset and that s in F denotes the price of the underlying asset at expiration, in the sense that for each $y \in [0, \infty]$ the term sy is one of the possible prices of the underlying at expiration. The dual role of s might be off-putting, but is very convenient for the calculations to follow. The reader should simply keep in mind that the current price of a contract, i.e. F^{BS}, is a function of the current price of the underlying and that the payoff function, i.e. F, is a function of the price of the underlying at expiration.

Moreover, the argument v in (1.15) may be suppressed. Remember, that v is connected with the volatility of the underlying asset, which in the Black-Scholes

[4]Note that in the May 31, 2009 version they have a typo. Particularly instead of $\ln(y)$ they write y in the exponent of e. They notice though that the kernel of the integrand is a lognormal density.

context is considered constant. The Black-Scholes formula maps a payoff function to the fair time-t price of the corresponding contract. For example the time-t price of a European call with strike price K expiring at $T > t$ is

$$F^{BS}_{C,K}(S_t) = \int_0^\infty (S_t y - K)^+ \phi_y(v) dy$$

where $\phi_y(v)$ denotes the lognormal kernel with parameters $\mu = -v^2/2$ and $\sigma = v$.

1.1.2 The Mixing Formula

The standard correlated volatility models are of the following form (for some examples see Fouque, 2010)

$$\begin{aligned} dS_t &= \sigma_t S_t W_{0t} \\ d\sigma_t &= a(\sigma_t)dt + b(\sigma_t)dW_{2t} \end{aligned} \tag{1.16}$$

where

$$W_0 = \sqrt{1-\rho^2}W_1 + \rho W_2$$

for some W_1 and W_2 uncorrelated Brownian motions, i.e. $dW_1 dW_2 = 0$, and $|\rho| \leq 1$. Calculate $dW_0 dW_2 = \rho dt$ to notice that the correlation of W_0 and W_2 is ρ. Moreover, calculate

$$\begin{aligned} dS_t d\sigma_t &= \sigma_t S_t b(\sigma_t) dW_{0t} dW_{2t} \\ &= \sigma_t S_t [b(\sigma_t)\rho] dt \end{aligned}$$

to show that the correlation of S_t and σ_t is $b(\sigma_t)\rho$. Carr and Lee (2009b) do not specify the dynamics of the volatility process and just assume that σ and W are correlated with correlation $|\rho| \leq 1$. Hence for some W_1 and W_2 independent $\mathcal{F}_t$-Brownian motions, the SDE describing the price is

$$dS_t = \sqrt{1-\rho^2}\sigma_t S_t dW_{1t} + \rho \sigma_t S_t dW_{2t} \tag{1.17}$$

Elaborating what we mean by S_t and σ being dependent, assume that σ and W_2 are adapted to some filtration $\mathcal{H}_t \subseteq \mathcal{F}_t$, where $\mathcal{H}_t$ and $\mathcal{F}^{W_1}_T$ are independent. The following proposition describes the pricing formula for this setting, henceforth the Carr-Lee setting without the independence assumption. The proof applies the conditioning argument presented by Hull and White (1987) and is presented in Carr and Lee (2009b). Keep in mind that the proof is similar to the construction we used to define the Black-Scholes formula above.

Proposition 1 (Mixing Formula)**.** *In the Carr-Lee setting without the independence assumption, i.e. the dynamics of S_t and σ_t are described by* (1.17) *the pricing formula* (1.10) *for a payoff function F is*

$$\mathbb{E}_t^Q[F(S_T)] = \mathbb{E}_t^Q\left[F^{BS}\left(S_t M_{t,T}(\rho),\, \bar{\sigma}_{t,T}\sqrt{1-\rho^2}\right)\right] \tag{1.18}$$

where $F^{BS}(\cdot,\cdot)$ is described by (1.14, 1.15),

$$M_{t,T}(\rho) := \exp\left(-\frac{\rho^2}{2}\int_t^T \sigma_u^2 du + \rho\int_t^T \sigma_u dW_{2u}\right)$$

and

$$\bar{\sigma}_{t,T} := \left(\int_t^T \sigma_u^2 du\right)^{1/2}$$

Proof. Apply Itô's lemma for $f(x) = \ln(x/S_0)$ to each of the terms in the sum at the right hand side part of (1.17) to get

$$dX_t = \left(-\frac{1}{2}\left(\sqrt{1-\rho^2}\sigma_t\right)^2\right)dt + \sqrt{1-\rho^2}\sigma_t dW_{1t} - \frac{1}{2}(\rho\sigma_t)^2 dt + \rho\sigma_t dW_{2t}$$

The integral equation of the above is

$$\begin{aligned} X_T - X_t = &-\frac{1}{2}\left(\sqrt{1-\rho^2}\right)^2\int_t^T \sigma_t^2 dt + \sqrt{1-\rho^2}\int_t^T \sigma_t dW_{1t} \\ &-\frac{\rho^2}{2}\int_t^T \sigma_t^2 dt + \rho\int_t^T \sigma_t dW_{2t} \end{aligned}$$

We are using now the conditioning argument. We are conditioning on $\mathcal{F}_t \vee \mathcal{H}_T$ and thus constitute X_t, σ_t and W_{2t} known in $[0,T]$. Hence, the second term is the only random term and has mean 0 and variance $\left(\sqrt{1-\rho^2}\right)^2\bar{\sigma}_{t,T}^2$ (see Shreve, 2008, Theorem 4.4.9)[5]. Finally, observe that the last two terms are equal to $\ln M_{t,T}(\rho)$ to conclude that

$$\left(X_T - X_t - \ln M_{t,T}(\rho)\right)\Big|(\mathcal{F}_t \vee \mathcal{H}_T) \sim N\left(-\frac{1}{2}\left(\sqrt{1-\rho^2}\right)^2\bar{\sigma}_{t,T}^2,\, \left(\sqrt{1-\rho^2}\right)^2\bar{\sigma}_{t,T}^2\right)$$

[5]Theorem 4.4.2 simply states that the Itô integral of a deterministic function $I(t) = \int_0^t \Delta(s)dB_s$ is normally distributed with expected value zero and variance equal to the quadratic variation of the deterministic function, i.e. $\int_0^t \Delta^2(s)ds$.

Use the tower property of conditional expectation to get

$$\mathbb{E}_t^Q[F(S_T)] = E_t^Q[\mathbb{E}^Q[F(S_T)|\mathcal{F}_t \vee \mathcal{H}_T)]] = \mathbb{E}_t^Q\left[F^{BS}\left(S_t M_{t,T}(\rho),\, \bar{\sigma}_{t,T}\sqrt{1-\rho^2}\right)\right]$$

$\blacksquare$

1.1.3 Correlation Immunity

We are closing this section by introducing the concept of correlation immunity described in Carr and Lee (2009a, Definition 4.4). We denote the set of all $\mathcal{F}_t$-measurable functions as $m\mathcal{F}_t$.

Definition 1. *We say that a payoff function F is first-order ρ-neutral or ρ-immune or correlation-neutral or correlation-immune at time $t < T \in \mathcal{T}$, if there exists function $c \in m\mathcal{F}_t$, such that*

$$\frac{\partial F^{BS}}{\partial s}(S_t, \sigma) = c, \quad \forall\, \sigma \geq 0$$

almost surely.

The partial derivative of the Black-Scholes formula with respect to the price s computed at the current price S_t is called the contract's Black-Scholes delta. The delta measures the rate of change of the contract's price with respect to the changes in the underlying asset's price. In other words, it measures how sensitive is the contract to the change's of the underlying.

As we shall see in (1.19), if the contract's Black-Scholes delta is constant across all the parameters σ, which are connected to the volatility of the process S_T, then the contract's price does not depend on the correlation parameter ρ but to a meaningless order of $O(\rho^2)$.

To do this first observe that, ceteris paribus, function F^{BS} given by (1.18) may be considered a function of ρ. We are going to expand the function in a Taylor series about $\rho = 0$. Remember, that according to Taylor's theorem, a function $f(x)$, given that the first derivative exists, may be expanded over 0 as follows

$$f(x) = f(0) + xf'(0) + O(x^2)$$

where the last term $O(x^2)$ contains factors that are of order x^2. F^{BS} is defined by an integral and thus its first derivative exists. Moreover observe that for $\rho = 0$

$$F^{BS}(S_t M_{t,T}(0),\, \bar{\sigma}_{t,T}) = F^{BS}(S_t,\, \bar{\sigma}_{t,T})$$

and that

$$\begin{aligned}\left.\frac{\partial F^{BS}(S_t M_{t,T}(\rho),\, \bar{\sigma}_{t,T}\sqrt{1-\rho^2})}{\partial \rho}\right|_{\rho=0} &= \left.\frac{\partial F^{BS}(S_t M_{t,T}(\rho),\, \bar{\sigma}_{t,T}\sqrt{1-\rho^2})}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial \rho}\right|_{\rho=0} \\ &= S_t \int_t^T \sigma_u dW_{2u} \frac{\partial F^{BS}(S_t,\, \bar{\sigma}_{t,T})}{\partial \sigma}\end{aligned}$$

since

$$\begin{aligned}\left.\frac{\partial \sigma}{\partial \rho}\right|_{\rho=0} = \left.\frac{\partial S_t M_{t,T}(\rho)}{\partial \rho}\right|_{\rho=0} &= S_t M_{t,T}(\rho) \left.\left(-\rho \int_t^T \sigma_u^2 du + \int_t^T \sigma_u dW_{2u}\right)\right|_{\rho=0} \\ &= S_t \int_t^T \sigma_u dW_{2u}\end{aligned}$$

to find that

$$\begin{aligned}F^{BS}(S_t M_{t,T}(\rho),\, \bar{\sigma}_{t,T}\sqrt{1-\rho^2}) = F^{BS}(S_t,\, \bar{\sigma}_{t,T}) + \\ \rho S_t \int_t^T \sigma_u dW_{2u} \frac{\partial F^{BS}(S_t,\, \bar{\sigma}_{t,T})}{\partial \sigma} + O(\rho^2)\end{aligned}$$

So by (1.18)

$$\begin{aligned}\mathbb{E}_t^Q[F(S_T)] &= \mathbb{E}_t^Q\left[F^{BS}\left(S_t M_{t,T}(\rho),\, \bar{\sigma}_{t,T}\sqrt{1-\rho^2}\right)\right] \\ &= \mathbb{E}_t^Q\left[F^{BS}(S_t,\, \bar{\sigma}_{t,T})\right] + \rho S_t \mathbb{E}_t^Q\left[\int_t^T \sigma_u dW_{2u} \frac{\partial F^{BS}(S_t,\, \bar{\sigma}_{t,T})}{\partial \sigma}\right] + O(\rho^2)\end{aligned}$$

If F is correlation-immune and since the mean of an Itô integral is zero we get that

$$\mathbb{E}_t^Q[F(S_T)] = \mathbb{E}_t^Q\left[F^{BS}(S_t,\, \bar{\sigma}_{t,T})\right] + O(\rho^2) \tag{1.19}$$

As we mentioned before (1.19) suggests that the price of the contract described by the payoff function F, given that F is correlation-immune, does not depend on the correlation of S_t and σ_t but to a small amount of order ρ^2.

Correlation-immunity is an important concept for our purposes. In the following section we will price a payoff function h on the realized variance, i.e. we will price $h(\langle X\rangle_T)$. We will do this by finding another payoff function G of S_T. In other words, we will find a result of the following form

$$\mathbb{E}_t^Q[h(\langle X\rangle_T)] = \mathbb{E}_t^Q[G(S_T)] \tag{1.20}$$

In fact once we have found one G that satisfies the previous equation, one may describe an infinite family of such functions. This stems from Carr and Lee (2009a) who have proved that if W_t and σ_t are independent then the following general form of put-call symmetry applies

$$\mathbb{E}_t^Q\left[f\left(\frac{S_T}{S_t}\right)\right] = \mathbb{E}_t^Q\left[\frac{S_T}{S_t}f\left(\frac{S_t}{S_T}\right)\right] \tag{1.21}$$

Particularly, at first we will find a payoff function G satisfying (1.20) without being concerned about independence. Afterwards we will specify a G that is correlated-immune, in order to conclude the more realistic case where S_t and σ_t are correlated. Remember that in the setting of Carr and Lee (2009b) nothing is assumed about the dynamics of σ_t and thus applies to a variety of settings.

1.2 Pricing of Exponentials

In the following proposition presented by Carr and Lee (2009b, Proposition 5.1) we identify two payoff functions G satisfying (1.20) for a generic exponential payoff function $h(\langle X\rangle_T) = \exp(\lambda\langle X\rangle_T)$, where λ is a complex number.

Proposition 2. *For each complex number $\lambda \in \mathbb{C}$ and $t \in [0,T] \subset \mathcal{T}$,*

$$\mathbb{E}_t^{\mathbb{Q}}\left[e^{\lambda\langle X\rangle_T}\right] = e^{\lambda\langle X\rangle_t}\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{1/2\pm\sqrt{1/4+2\lambda}}\right] \tag{1.22}$$

Proof. An equivalent and more mathematically rigorous form of (1.8), known as the integral equation, is

$$X_T - X_t = -\frac{1}{2}\int_t^T \sigma_u^2 du + \int_t^T \sigma_u dW_u, \quad \forall\, [t,T] \subset \mathcal{T}$$

Using the quadratic variation notation we may rewrite the previous one as

$$X_T - X_t = -\frac{1}{2}\left(\langle X\rangle_T - \langle X\rangle_t\right) + \int_t^T \sigma_u dW_u, \quad \forall\, [t,T] \subset \mathcal{T}$$

We are using the conditioning argument on the σ-algebra $\mathcal{F}_T \vee \mathcal{F}_T^\sigma$ and observe that $\langle X\rangle_T - \langle X\rangle_t$ becomes a known constant and $\sigma_u = \sigma(u,\omega)$ a deterministic function of u in $[0,T]$. So the right hand side contains a constant and an Itô integral of a deterministic function. We conclude that

$$X_T - X_t\big| \left(\mathcal{F}_T \vee \mathcal{F}_T^\sigma\right) \sim N\left(-\frac{1}{2}\left(\langle X\rangle_T - \langle X\rangle_t\right), \langle X\rangle_T - \langle X\rangle_t\right)$$

Let $p \in \mathbb{C}$ and use the tower property and the characteristic function of a normal random variable to compute

$$\begin{aligned}
\mathbb{E}_t^{\mathbb{Q}}\left[e^{p(X_T - X_t)}\right] &= \mathbb{E}_t^{\mathbb{Q}}\left[\mathbb{E}^{Q}\left[e^{p(X_T-X_t)}\big| \left(\mathcal{F}_T \vee \mathcal{F}_T^\sigma\right)\right]\right] \\
&= \mathbb{E}_t^{\mathbb{Q}}\left[e^{-p\frac{1}{2}\left(\langle X\rangle_T - \langle X\rangle_t\right)+\frac{1}{2}p^2\left(\langle X\rangle_T - \langle X\rangle_t\right)}\right] \\
&= \mathbb{E}_t^{\mathbb{Q}}\left[e^{\left(\frac{1}{2}p^2 - \frac{1}{2}p\right)\left(\langle X\rangle_T - \langle X\rangle_t\right)}\right]
\end{aligned}$$

Set $\lambda = \frac{1}{2}p^2 - \frac{1}{2}p$ and solve $p^2 - p - 2\lambda = 0$, for p to obtain

$$p = \frac{1}{2} \pm \sqrt{\frac{1}{4} + 2\lambda}$$

Observe now that e^{pX_t} is an $\mathcal{F}_t$-measurable random variable and conclude that

$$\begin{aligned}
\mathbb{E}_t^{\mathbb{Q}}\left[e^{\lambda\langle X\rangle_T}\right] &= e^{\lambda\langle X\rangle_t}\mathbb{E}_t^{\mathbb{Q}}\left[e^{p(X_T-X_t)}\right] \\
&= e^{\lambda\langle X\rangle_t}\mathbb{E}_t^{\mathbb{Q}}\left[e^{p\log(S_T/S_t)}\right] \\
&= e^{\lambda\langle X\rangle_t}\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^p\right] \\
&= e^{\lambda\langle X\rangle_t}\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{\frac{1}{2}\pm\sqrt{\frac{1}{4}+2\lambda}}\right]
\end{aligned}$$

■

The previous proposition specifies two payoff function G, in the sense of (1.20). Observe that if we choose one, namely

$$\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{\frac{1}{2}-\sqrt{\frac{1}{4}+2\lambda}}\right]$$

we get the other one by put-call symmetry (1.21). Indeed

$$\begin{aligned}
\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{\frac{1}{2}-\sqrt{\frac{1}{4}+2\lambda}}\right] &= \mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)(S_t/S_T)^{\frac{1}{2}-\sqrt{\frac{1}{4}+2\lambda}}\right] \\
&= \mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)(S_T/S_t)^{-\frac{1}{2}+\sqrt{\frac{1}{4}+2\lambda}}\right] \\
&= \mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{\frac{1}{2}+\sqrt{\frac{1}{4}+2\lambda}}\right]
\end{aligned}$$

In Proposition 3, which is Proposition 5.9 of (Carr and Lee, 2009b), we will employ put-call symmetry to specify a suitable family of functions G and consequently we will chose one which is immune-correlated. To do this we use the following lemma.

Lemma 1. *Let $\lambda \in \mathbb{C}$ and denote by $\phi_y(v)$ the lognormal density function with parameters $-v^2/2$ and v,i.e.*

$$\phi_y(v) = e^{-\left(\ln(y)+v^2/2\right)^2)/(2v^2)}dy$$

Then for $p_\pm(\lambda) := \frac{1}{2} \pm \frac{1}{2\sqrt{1+8\lambda}}$

$$\int_0^\infty y^{p_+}\phi_y(v) = \int_0^\infty y^{p_-}\phi_y(v)$$

Proposition 3. *For any complex number $\lambda \in \mathbb{C}$ and $t \in [0,T] \subset \mathcal{T}$,*

$$\mathbb{E}_t^{\mathbb{Q}}\left[e^{\lambda\langle X\rangle_T}\right] = \mathbb{E}_t^{\mathbb{Q}}[G_{exp}(S_T, S_t, \langle X\rangle_t\,; \lambda)] \tag{1.23}$$

where

$$G_{exp}(S, u, q; \lambda) := e^{\lambda q}[\theta_+(S/u)^{p_+} + \theta_-(S/u)^{p_-}]$$

and

$$\theta_\pm(\lambda) := \frac{1}{2} \mp \frac{1}{2\sqrt{1+8\lambda}}, \qquad p_\pm(\lambda) := \frac{1}{2} \pm \frac{1}{2\sqrt{1+8\lambda}}$$

the payoff function $G(S_T) := G_{exp}(S_T, S_t, \langle X\rangle_t\,; \lambda)$ is correlation-immune.

Proof. From Proposition 2 chose

$$\mathbb{E}_t^{\mathbb{Q}}\left[e^{\lambda\langle X\rangle_T}\right] = e^{\lambda\langle X\rangle_t}\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{1/2+\sqrt{1/4+2\lambda}}\right] \tag{1.24}$$

Employ put-call symmetry (1.21) for some $f \in m\mathcal{F}_t$ to rewrite the expectation of the right hand side of (1.24) as

$$\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{1/2+\sqrt{1/4+2\lambda}} + f(S_T/S_t) - (S_T/S_t)f(S_T/S_t)\right]$$

For an arbitrary $\theta \in m\mathcal{F}_t$, chose

$$f(x) := \theta x^{1/2-\sqrt{1/4+2\lambda}}$$

to get

$$\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{1/2+\sqrt{1/4+2\lambda}}+\theta(S_T/S_t)^{1/2-\sqrt{1/4+2\lambda}}-(S_T/S_t)\theta(S_t/S_T)^{1/2-\sqrt{1/4+2\lambda}}\right]=$$
$$\mathbb{E}_t^{\mathbb{Q}}\left[(S_T/S_t)^{1/2+\sqrt{1/4+2\lambda}}+\theta(S_T/S_t)^{1/2-\sqrt{1/4+2\lambda}}-\theta(S_T/S_t)^{1/2+\sqrt{1/4+2\lambda}}\right]=$$
$$\mathbb{E}_t^{\mathbb{Q}}\left[(1-\theta)(S_T/S_t)^{1/2+\sqrt{1/4+2\lambda}}+\theta(S_T/S_t)^{1/2-\sqrt{1/4+2\lambda}}\right]$$

Observe that for the θ's and p's denoted in the proposition $\theta_+ + \theta_- = 1$ and $p_+ + p_- = 1$ are valid. Also, observe that due to symmetry $\theta_+ p_- = \theta_- p_+$ to conclude that $\theta_+ p_+ + \theta_- p_- = 1$ is also valid. Using this notation and the previous observations to rewrite (1.24) as

$$\mathbb{E}_t^{\mathbb{Q}}\left[e^{\lambda\langle X\rangle_T}\right]=e^{\lambda\langle X\rangle_t}\mathbb{E}_t^{\mathbb{Q}}\left[\theta_+(S/u)^{p_+}+\theta_-(S/u)^{p_-}\right]=\mathbb{E}_t^{\mathbb{Q}}[G_{exp}(S_T,S_t,\langle X\rangle_t\,;\lambda)]$$

We now show that the payoff function $G(S_T) := G_{exp}(S_T, S_t, \langle X\rangle_t\,; \lambda)$ is correlation-immune. Indeed,

$$\begin{aligned}\left.\frac{\partial F^{BS}(s)}{\partial s}\right|_{s=S_t} &= \left.\frac{\partial}{\partial s}\int_0^\infty G_{exp}(xy,S_t,\langle X\rangle_t\,;\lambda)\phi_y(v)dy\right|_{x=S_t}\\ &= \int_0^\infty \left.\frac{\partial}{\partial s}G_{exp}(xy,S_t,\langle X\rangle_t\,;\lambda)\right|_{x=S_t}\phi_y(v)dy\end{aligned}$$

where $\phi_y(v)$ is the lognormal density with parameters $-v^2/2$ and v. The last equality is valid by Leibniz's integral rule because $G_{exp}(S,u,q;\lambda)$ and $\partial G_{exp}(S,u,q;\lambda)/\partial S$ are continuous functions. So

$$\begin{aligned}\left.\frac{\partial F^{BS}(s)}{\partial s}\right|_{s=S_t} &= \int_0^\infty \left.\theta_+p_+x^{p_+-1}\left(\frac{y}{S_t}\right)^{p_+}+\theta_-p_-x^{p_--1}\left(\frac{y}{S_t}\right)^{p_-}\right|_{x=S_t}\phi_y(v)dy\\ &= \int_0^\infty \frac{\theta_+p_+}{S_t}y^{p_+}+\frac{\theta_-p_-}{S_t}y^{p_-}\phi_y(v)dy\\ &= \frac{\theta_+p_+}{S_t}\int_0^\infty y^{p_+}\phi_y(v)dy+\frac{\theta_-p_-}{S_t}\int_0^\infty y^{p_-}\phi_y(v)dy\end{aligned}$$

Now use Lemma 1 to get

$$\begin{aligned}\left.\frac{\partial F^{BS}(s)}{\partial s}\right|_{s=S_t} &= \frac{\theta_+p_+}{S_t}\int_0^\infty y^{p_+}\phi_y(v)dy+\frac{\theta_-p_-}{S_t}\int_0^\infty y^{p_+}\phi_y(v)dy\\ &= \frac{\theta_+p_+ + \theta_-p_-}{S_t}\int_0^\infty y^{p_+}\phi_y(v)dy = 0\end{aligned}$$

■

Bakshi, Panayotov, and Skoulakis (2011) employ a special case of exponential claims. Particularly they consider the claim contingent to $e^{-\langle X\rangle_T}$, whose correlation immune pricing formula we may derive from Proposition 3 by setting $\lambda = -1$. We provide the proof of the following statement based on a previous draft of their paper.

Corollary 1. *The correlation-immune payoff function of S_T corresponding to the contract with payoff $e^{-\langle X\rangle_T}$ at time $t < T \in \mathcal{T}$ is*

$$G_t(S_T) = \sqrt{\frac{8}{7}}\sqrt{\frac{S_T}{S_t}}\cos\left(\arctan\left(\frac{1}{\sqrt{7}}\right) + \frac{\sqrt{7}}{2}\ln\left(\frac{S_T}{S_t}\right)\right) \tag{1.25}$$

Proof. We calculate

$$G(S_T) = G_{exp}(S_T, S_t, \langle X\rangle_t\,; -1)$$

Observe that for $\lambda = -1$

$$\theta_\pm(-1) = \frac{1}{2} \mp \frac{1}{2\sqrt{7}i} = \frac{1}{2} \pm \frac{1}{2\sqrt{7}}i$$

It's absolute value is $\sqrt{\frac{1}{4} + \frac{1}{28}} = \sqrt{\frac{2}{7}}$ and its argument is

$$\arctan\left(\pm\frac{1/(2\sqrt{7})}{1/2}\right) = \arctan\left(\pm\frac{1}{\sqrt{7}}\right)$$

Thus its polar form is

$$\theta_\pm(-1) = \sqrt{2/7}e^{i\arctan(\pm 1/\sqrt{7})}$$

Finally simply observe that

$$p_\pm(-1) = \frac{1}{2} \pm \frac{\sqrt{7}}{2}i$$

So

$$G(S_T) = e^{-\langle X\rangle_t}[\theta_+(-1)(S/u)^{p_+(-1)} + \theta_-(-1)(S/u)^{p_-(-1)}]$$

and thus

$$\begin{aligned} e^{\langle X\rangle_t}G(S_T) &= \sqrt{\frac{2}{7}}e^{i\arctan(1/\sqrt{7})}\left(\frac{S_T}{S_t}\right)^{1/2+i\sqrt{7}/2} + \sqrt{\frac{2}{7}}e^{i\arctan(-1/\sqrt{7})}\left(\frac{S_T}{S_t}\right)^{1/2-i\sqrt{7}/2} \\ &= \sqrt{\frac{2}{7}}\sqrt{\frac{S_T}{S_t}}\left(e^{i\arctan(1/\sqrt{7})}\left(\frac{S_T}{S_t}\right)^{i\sqrt{7}/2} + e^{i\arctan(-1/\sqrt{7})}\left(\frac{S_T}{S_t}\right)^{-i\sqrt{7}/2}\right) \\ &= \sqrt{\frac{2}{7}}\sqrt{\frac{S_T}{S_t}}\left(e^{(\arctan(1/\sqrt{7})+\sqrt{7}/2\ln(S_T/S_t))i} + e^{(\arctan(-1/\sqrt{7})-\sqrt{7}/2\ln(S_T/S_t))i}\right) \end{aligned}$$

Arctan is an odd function and so the sum inside the parenthesis is of the form $e^{\phi i} + e^{-\phi i}$ which is easy to see, using Euler's identity, that it is equal to $2\cos(\phi)$. So

$$\begin{aligned} G(S_T) &= e^{-\langle X \rangle_t} \sqrt{\frac{2}{7}} \sqrt{\frac{S_T}{S_t}} 2\cos\left(\arctan\left(\frac{1}{\sqrt{7}}\right) + \frac{\sqrt{7}}{2} \ln\left(\frac{S_T}{S_t}\right) \right) \\ &= e^{-\langle X \rangle_t} \sqrt{\frac{8}{7}} \sqrt{\frac{S_T}{S_t}} \cos\left(\arctan\left(\frac{1}{\sqrt{7}}\right) + \frac{\sqrt{7}}{2} \ln\left(\frac{S_T}{S_t}\right) \right) \end{aligned}$$

■

1.3 Spanning Contingent claims

Pricing financial derivatives is a general practice of quantitative finance. In the context of martingale (arbitrage-free) pricing methodology this is done by a self-financing trading strategy, which is a *continuous* trading strategy where no infusions or withdrawals of money occur. The price of the derivative is defined to be the wealth of a self-financing trading strategy which acts upon primary assets, such as cash and the underlying asset, and completely replicates the cash flows of the derivative to be priced. Market completeness ensures than any derivative may be priced uniquely using this methodology. It is known that common stochastic volatility models, described by (1.16), are incomplete if stocks and bonds are the only trade primary assets (Rutkowski, 2010). Our framework is even more general, does not assume anything about the volatility process, and thus we cannot use the market completeness assumption.

We consider though that markets for cash, for stocks *and* for out-of-the-money European puts and calls exist for all strikes. The continuity of strikes is in accordance with the continuity of the trading strategy. It has been shown (Breeden and Litzenberger, 1978; Green and Jarrow, 1987; Nachman, 1988) that this market structure is complete for any derivative which is a smooth function of the underlying stock price. Moreover, a constructive approach is presented by Carr and Madan (2001), by Proposition-4. To prove the proposition we are going to use the following lemma.

Lemma 2. *Let $f : \mathbb{R} \to \mathbb{R}$ be a twice differentiatiable function and let $a \in \mathbb{R}$ be a fixed number. Then*

$$f(x) = f(a) + f'(a)(x-a) + \int_a^\infty f''(v)(x-v)^+ dv + \int_0^a f''(v)(v-x)^+ dv \quad (1.26)$$

Proof. Apply the *Fundamental Theorem of Calculus* for f, f' and the fixed number a to get the following equations

$$\begin{aligned} f(x) - f(a) &= \int_a^x f'(u)du \\ f'(u) - f'(a) &= \int_a^u f''(v)dv \end{aligned}$$

Combine them to get

$$\begin{aligned} f(x) - f(a) &= \int_a^x \left[f'(a) + \int_a^u f''(v)dv \right] du \\ &= \int_a^x f'(a)du + \int_a^x \int_a^u f''(v)dvdu \\ &= f'(a)(x-a) + \int_a^x \int_a^u f''(v)dvdu \end{aligned}$$

The double integral suggest integration over the domain

$$\begin{aligned} D &= \left\{ (v,u) \in \mathbb{R}^2 :\ a < v < u \text{ and } a < u < x \right\} \\ &= \left\{ (v,u) \in \mathbb{R}^2 :\ a < v < x \text{ and } v < u < x \right\} \end{aligned}$$

So by Fubini's theorem

$$\begin{aligned} f(x) - f(a) &= f'(a)(x-a) + \int_a^x \int_v^x f''(v)dudv \\ &= f'(a)(x-a) + \int_a^x f''(v)(v-x)dv \end{aligned}$$

Finally, using the notation of the characteristic function observe that

$$\begin{aligned} \int_a^x f''(v)(v-x)dv &= \mathbf{1}_{x>a} \int_a^x f''(v)(v-x)dv + \mathbf{1}_{x<a} \int_a^x f''(v)(v-x)dv \\ &= \mathbf{1}_{x>a} \int_a^x f''(v)(v-x)dv + \mathbf{1}_{x<a} \int_x^a f''(v)(x-v)dv \\ &= \int_a^\infty f''(v)(v-x)^+ dv + \int_0^a f''(v)(x-v)^+ dv \end{aligned}$$

Combine the above to get the desired result.

∎

Proposition 4. *Let $F \in C^2(\mathbb{R})$, i.e. $F : \mathbb{R} \to \mathbb{R}$ and its second derivative exist and it is continuous. Let also $t, T \in \mathcal{T}$ such that $t < T$. Then*

$$\mathbb{E}_t^Q[e^{-r\tau}F(S_T)] = e^{-r\tau}F(S_t) + F'(S_t)[C_t(S_t) + P_t(S_t)] + \int_{S_t}^{\infty} F''(K)C_t(K)dK + \int_0^{S_t} F''(K)P_t(K)dK \tag{1.27}$$

where $\tau = T - t$ and $C_t(x), P_t(x)$ are the time-t prices of a call and a put respectively with maturity T and strike x.

Proof. Apply Lemma 2 for $f = F$, $x = S_T$, $a = S_t$ and $v = K$ to get

$$F(S_T) = F(S_t) + F'(S_t)(S_T - a) + \int_a^{\infty} F''(K)(S_T - K)^+ dK + \int_0^a F''(K)(S_T - K)^+ dK$$

Note that $F(S_T)$ is a composition of a continuous (thus Borel) function and an $\mathcal{F}_T$-measurable function. So $F(S_T)$ is $\mathcal{F}_T$-measurable and thus is a European payoff function. Observe that

$$(S_T - a) = (S_T - a)^+ + (a - S_T)^+$$

and that $F'(S_t)$, $F''(K)$ are constants conditioned on $\mathcal{F}_t$. In order to get the result, discount by $e^{-r(T-t)}$ and take the conditional expectation $\mathbb{E}_t^Q[\cdot]$ of both sides. Note that the expectation goes inside the integrals by Fubini's theorem.

∎

We are concluding this section by spanning the payoff function used in Bakshi, Panayotov, and Skoulakis (2011).

Corollary 2. *The spanning of the payoff function* (1.25) *is*

$$\mathbb{E}_t^Q[e^{-r\tau}G_t(S_T)] = e^{-r\tau} + \int_{K>S_t} \omega[K]C_t[K]dk + \int_{K<S_t} \omega[K]P_t[K]dK \tag{1.28}$$

where

$$\omega[K] = \frac{\frac{8}{\sqrt{14}}\cos\left(\arctan(1/\sqrt{7}) + (\sqrt{7}/2)\ln(K/S_t)\right)}{\sqrt{S_t}K^{3/2}} \tag{1.29}$$

Proof. Recall that S_t is known and that

$$G_t(x) = \sqrt{\frac{8}{7}}\sqrt{\frac{x}{S_t}}\cos\left(\arctan\left(\frac{1}{\sqrt{7}}\right) + \frac{\sqrt{7}}{2}\ln\left(\frac{x}{S_t}\right)\right)$$

Observe that

$$G_t(S_t) = \sqrt{\frac{8}{7}}\cos\left(\arctan\left(\frac{1}{\sqrt{7}}\right)\right) = \sqrt{\frac{8}{7}}\sqrt{\frac{7}{8}} = 1$$

The first derivative is

$$G_t'(x) = \frac{1}{2x}G_t(x) - \sqrt{\frac{2}{xS_t}}\sin\left(\arctan\left(\frac{1}{\sqrt{7}}\right) + \frac{\sqrt{7}}{2}\ln\left(\frac{x}{S_t}\right)\right)$$

and so

$$G_t'(S_t) = \frac{1}{2S_t} - \frac{\sqrt{2}}{S_t}\sin\left(\arctan\left(\frac{1}{\sqrt{7}}\right)\right) = \frac{1}{2S_t} - \frac{\sqrt{2}}{S_t}\frac{1}{2\sqrt{2}} = 0$$

To find the second derivative, rewrite the equation containing the first derivative as

$$\sin\left(\arctan\left(\frac{1}{\sqrt{7}}\right) + \frac{\sqrt{7}}{2}\ln\left(\frac{x}{S_t}\right)\right) = \sqrt{\frac{xS_t}{2}}\left(\frac{1}{2x}G_t(x) - G_t'(x)\right)$$

Take the derivative of both sides to find that

$$\begin{aligned}\frac{7}{2}\sqrt{\frac{S_t}{8x^3}}G_t &= \sqrt{\frac{S_t}{8x}}\left(\frac{1}{2x}G_t - G_t'\right) + \sqrt{\frac{xS_t}{2}}\left(-\frac{1}{2x^2}G_t + \frac{1}{2x}G_t' - G_t''\right)\\ &= -\frac{1}{2}\sqrt{\frac{S_t}{8x^3}}G_t - \sqrt{\frac{xS_t}{2}}G_t''\end{aligned}$$

Solve for $G''(x)$

$$G_t''(x) = -\frac{2}{x^2}G_t(x) = \frac{\frac{8}{\sqrt{14}}\cos\left(\arctan(1/\sqrt{7}) + (\sqrt{7}/2)\ln(x/S_t)\right)}{\sqrt{S_t}x^{3/2}}$$

Apply the previous relations to determine the factors of Proposition 4.

■

Chapter 2

Extracting Forward Variances

In the previous chapter we presented a model free formula for pricing exponential claims on integrated variance. In particular, following the notation of Bakshi et al. (2011), we showed that the time-t value of the claim

$$H_t^{(t,n)} = \int_t^{t+\tau_n} \sigma_u^2 du \tag{2.1}$$

is

$$H_t^{(t,n)} = e^{-r\tau_n} + \int_{K>S_t} \omega[K] C_t^{(n)}[K] dK + \int_{K<S_t} \omega[K] P_t^{(n)}[K] dK \tag{2.2}$$

where r is the risk-free rate, $C_t^{(n)}[K]$ and $P_t^{(n)}[K]$ are the time-t prices of a call and a put with maturity $t + \tau_n$ and strike K, and

$$\omega[K] = -\frac{8}{\sqrt{14}} \frac{\cos\left(\arctan\left(\frac{1}{\sqrt{7}}\right) + \frac{\sqrt{7}}{2}\ln\left(\frac{K}{S_t}\right)\right)}{\sqrt{S_t} K^{3/2}} \tag{2.3}$$

Assume, by definition, that for $\tau_n = 0$, $H_t^{(t,0)} = 1$. In this chapter we calculate the prices of these claims for the S&P 500 index. Specifically, we approximate (2.2) for the last trading day of each month from September 1998 to September 2008, a total of 121 months, with τ_n being roughly equal to 19, 49, 79 and 109 days. The time series

$$H_t^{(t,n)}, \quad \text{for } t = 1, \ldots, T = 121, \text{ and } n = 1, 2, 3, 4 \tag{2.4}$$

will then be the basis for building the time series of forward variances

$$\mathbf{f}_t \equiv \begin{pmatrix} y_t^{(1)} \\ f_t^{(2)} \\ f_t^{(3)} \\ f_t^{(4)} \end{pmatrix} \equiv \begin{pmatrix} \ln H_t^{(t,0)} - \ln H_t^{(t,1)} \\ \ln H_t^{(t,1)} - \ln H_t^{(t,2)} \\ \ln H_t^{(t,2)} - \ln H_t^{(t,3)} \\ \ln H_t^{(t,3)} - \ln H_t^{(t,4)} \end{pmatrix} \tag{2.5}$$

Note that since $H_t^{(t,0)} = 1$, $y_t^{(1)} = -\ln H_t^{t,1}$. The calculation of the integrals in (2.2), require a continuum of option prices across strikes. Moreover the time series require roughly constant maturities across different trading days. Along with a presentation of the data we use, these issues are discussed in this chapter.

In particular, the chapter is constructed as follows. In the first section we describe the data, how we obtained them and any manipulation acted on them. In the second section, following the literature, we filter the data for unreliable observations and for those who form arbitrage opportunities, by checking prices' monotonicity and convexity across strikes. In the third section we compute the integrals. Specifically, after a small introduction on how splines are used to fit noisy data, we discuss how we interpolated across maturities to create new observations where needed. We continue by presenting a brief review of the literature of non parametric curve fitting techniques used for extracting risk-neutral probability density functions and conclude by presenting the method we employed. In the final section, some last comments are noted about the computation of the integrals and plots for the time-series of forward variances are provided.

The analysis is conducted using the statistical programming language R (Core Team) (2013) and the scripts may be found in Section-A. All the files and/or scripts mentioned in the text are available upon request.

2.1 Obtaining the Data

We obtained the S&P 500 index option data from OptionMetrics through the Wharton Research Data Services (WRDS) web interface (web queries). We downloaded data ranging from 01/09/1998 to 31/09/2008 for the S&P 500 Index SPX, with SECID 108105, for European options and for all maturities. Our data contained information about the Highest Closing Bid, the Lowest Closing Ask, the

Volume, the Open Interest and the Strike Price times 1000. Moreover, it contained the implied volatility and the sensitivity quantities Delta, Gamma, Theta and Vega/Kappa. We selected the date format DATE9 (e.g. 25JUL1984). The output format was a comma-delimited text file (.csv). The data contained $1,568,099$ observations.

We loaded the data in R as a Data Frame object and converted dates to R Date objects. We used the RQuantLib library (Eddelbuettel and Nguyen, 2014) to subset the data to the last trading (last business) day for each month using the United States NYSE calendar (see Part 1 in Script-A.2). The sample now contained $74,592$ observations.

The S&P 500 index monthly prices were acquired from the Center for Research in Security Prices (CRSP) through the WRDS web interface. We used the CRSP Stock Market Indexes (NYSE/AMEX/NASDAQ/ARCA) database and queried for the 'Level on S&P Composite index' ranging from September 1998 to September 2008. The output was written on a csv file, which we named SnP500Prices.csv. Note, that the dates of these data coincide with the last trading dates that we got by RQuantLib.

2.1.1 Risk-free rate

Hull (2012) mentions (see p. 76-77) that financial institutions have traditionally used the London Interbank Offered Rate (LIBOR) as risk-free rates. After the 2007 credit crisis though, use of LIBOR was criticized by many derivative dealers who gradually started using the overnight indexed swap (OIS) rate as a proxy for the risk-free rate. Since our data are prior to 2008 we chose LIBOR as the proxy for the risk free rate. For each day there are available LIBORs for a number of currencies maturing in 1 day (overnight), 1 week, 2 weeks and in 1 to 12 months. We obtained US Dollar (USD) LIBORs for all maturities.

LIBORs were acquired from two sources. The first one is EconStats[1] and the second one is the Federal Reserve Bank of St. Louis[2]. We compared the values

[1] http://www.econstats.com/r/rlib__d1.htm

[2] Data Source: FRED, Federal Reserve Economic Data, Federal Reserve Bank of St. Louis: London Interbank Offered Rate (LIBOR), based on U.S. Dollar; ICE Benchmark Administration; http://research.stlouisfed.org/fred2/categories/33003/downloaddata; accessed

contained to both databases. They were equal. Then, we started with EconStat and fill missing data from the FRED database. Both databases did not have data for 31/12/1999. We observed that rates were constant for the two previous days and so we used those. Finally, both databases did not have data for 2-Week LIBORs prior to 2000. We calculated those by interpolating[3] between the 1-Week LIBOR and the 1-Month LIBOR. The output was written on the LIBORs.csv file (see Script-A.1).

LIBOR is administrated, since 31 January 2014, by the Intercontinental Exchange (ICE) Benchmark Administration Ltd. Prior to that, it was administrated by the British Bankers' Association (BBA), whose website[4] we consulted to find all the required information and guidelines for practical applications. We found that LIBORs are annualized, simply compounded interest rates and thus the interest due for a payment of \$1 in time-t in the future is

$$r = 1 \times \left(\frac{\substack{\text{bbalibor} \\ \text{rate}}}{100} \right) \times \left(\frac{\substack{\text{Number of days} \\ \text{until t}}}{360} \right)$$

where bbalibor is substituted by linearly interpolating the LIBORs whose maturities surround t. It is then trivial to get the discount factors

$$\text{discount} = \frac{1}{1+r}$$

and the corresponding continuously compounded rates r_c by solving the following equation

$$\frac{1}{\text{discount}} = e^{r_c t} \Rightarrow r_c = \frac{1}{t} \ln \left(\frac{1}{\text{discount}} \right)$$

Note, that t is measured in years and by convention a year contains 360 days. Solve the previous equation for r_c to find that

$$r_c = \left(\frac{360}{\substack{\text{Number of days} \\ \text{until expiration}}} \right) \ln \left(\frac{1}{\text{discount}} \right)$$

June 9, 2014.

[3]By convention, a year contains 360 days. Driven by this assumption, we assume that each month has 30 days a week 7 days. Moreover, when we interpolate between the 2-Week LIBOR the 1-Month LIBOR we assume that two weeks contain 15 days (half of one month).

[4]http://www.bbalibor.com/technical-aspects

2.2 Filtering Data

Following the literature, see for example Neumann and Skiadopoulos (2013); Jiang and Tian (2005); Panigirtzoglou and Skiadopoulos (2004); Bliss and Panigirtzoglou (2002), option prices are defined as the mid bid-ask quote and several filters, enumerated in the following list, are applied to the sample. Our initial sample had 74,592. From these we omitted 76 options maturing in less than a week, as pricing anomalies might occur close to expiration. The new minimum expiration became 16 days. After applying the following filters, the sample was left with 17,421 observations. The filters were applied as they are being enumerated below. The number of observations excluded due to a filter is written in parenthesis.

F1: Options with bid price equal to zero (4,448).

F2: Options with zero open interest (13,052).

F3: Options with zero trading volume (33,342).

F4: With implied volatility greater than 100% or not available (NA) (1,043).

F5: In-the-money (ITM) options as defined below (3,038).

F6: Options whose price usual arbitrage bounds (0).

F7: Options whose price violates monotonicity (246).

F8: Options whose price violates convexity (1,137).

F9: Options whose price is less than 3/8 (789).

Filters 1, 2 and 3 are imposed to discard potentially wrong prices due to lack of liquidity. Prices with implied volatilities greater than 1 may correspond to extreme market conditions (outliers) and that is why they are not considered. Moreover, we discard observations where the implied volatility is not available. Filter 5 is applied because in-the-money (ITM) options are usually more expensive and less liquid than at-the-money (ATM) and out-of-the money (OTM) options. Existence of ATM options is extremely rare, since prices and strikes are quoted differently. In fact we did not encounter such an instance. So, following Jiang and Tian (2005), we define in-the-money options as call options with strike prices less than 97% of the asset price and put options with strike prices more than 103% of the asset price, in order to retain some information about the ATM region. The following filters (6, 7 and 8) are used to exclude option prices that form arbitrage opportunities. Finally, we apply Filter 9 to exclude prices that may be inaccurate due to errors arising from quotes being too close to the tick. Note, that in the comments of Script-A.2 we keep track of observations discarded by each filter.

We apply filters F1 to F4 in Part 2 of Script-A.2. In this part, we also exclude information that we are not going to need anymore. For each observation we keep the date, the type of the option (call or put), the expiration (in days), the strike divided by 1000, the price, the volume and the implied volatility. In Part 3 we load the underlying price for each observation, apply F5 and discard option maturing in less than 16 days. Finally, in Part 4 we load the discount factors, by linearly interpolating LIBORs maturing in the two closest months for each expiration. Subsequently we apply F6, to find that none of the observations violated the usual arbitrage bounds

$$
\begin{aligned}
&S_t - Ke^{-r(T-t)} \leq C_t \leq S_t \\
&Ke^{-r(T-t)} - S_t \leq P_t \leq Ke^{-r(T-t)}
\end{aligned}
$$

where C_t and P_t are the price of a call and put option respectively, K is the strike price and $T - t$ is the time until expiration expressed in years. Note, that in the script we are working with discount factors which are simply $e^{-r(T-t)}$, where $T - t$ are days until expiration divided by 360.

2.2.1 Checking strict Monotonicity

Violation of monotonicity leads to arbitrage opportunities (violation of no arbitrage conditions in vertical spreads). Prices' of call options must be strictly decreasing across strike prices, i.e.

$$K_1 < K_2 \Rightarrow C_1 > C_2$$

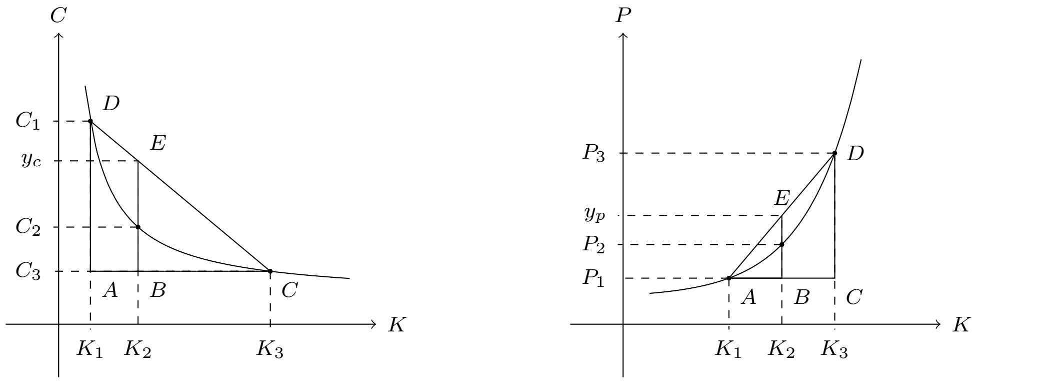
otherwise one could easily form an arbitrage strategy by selling C_1 and buying C_2. On the other hand, put prices must be strictly increasing across strike prices, i.e.

$$K_1 < K_2 \Rightarrow P_1 > P_2$$

Moreover, if $C_1 = C_2$ we discard the observation corresponding to C_1, since this is the one creating the arbitrage. If C_1 was the fair prices then C_2 should have never existed in the first place. Accordingly, if $P_1 = P_2$ we discard P_2. We do that in Part 5 of Script-A.2. We found 246 violations in total, 82 for calls and 164 for puts.

2.2.2 Checking Convexity

Prices of options should be convex across strike prices, otherwise arbitrage opportunities are created using strategies formed by generalized butterfly spreads. Prior to explaining the butterfly strategy, we derive, using Figure-2.1, convexity condition that must be satisfied by option prices. Our data are sorted increasingly for strike price. Moreover, due to the previous section, call prices are strictly decreasing and put prices are strictly increasing with respect the strike price.



(a) For any triplet of (K_1, C_1), (K_2, C_2) and (K_3, C_3) convexity for call options suggests that $C_2 \leq y_c$.

(b) For any triplet of (K_1, P_1), (K_2, P_2) and (K_3, P_3) convexity for put options suggests that $P_2 \leq y_p$.

Figure 2.1: Option prices must be convex with respect the strike. Here, we assume that we have already checked for monotonicity of option prices across strikes and thus prices of call options are strictly decreasing as strike increases and put options are strictly decreasing as strike increases.

In Figure-2.1a triangles (ACD) and (BCE) are similar and thus corresponding sides have lengths in the same ratio.

$$\frac{(BE)}{(AD)} = \frac{(BC)}{(AC)} \Rightarrow (BE) = \frac{(BC)}{(AC)}(AD)$$

Substitute with the corresponding coordinates to get

$$y_c - C_3 = \frac{K_3 - K_2}{K_3 - K_1}(C_1 - C_3)$$

So, the convexity condition for call options and for any triplet (K_1, C_1), (K_2, C_2) and (K_3, C_3) is

$$C_2 \leq C_3 + \frac{K_3 - K_2}{K_3 - K_1}(C_1 - C_3) \tag{2.6}$$

Note, since we have cleared the sample for strict monotonicity the equality is never achieved. Similarly, to derive the convexity condition for put options observe that triangles (ABE) and (ACD) in Figure-2.1b are similar to get that

$$\frac{(BE)}{(CD)} = \frac{(AB)}{(AC)} \Rightarrow (BE) = \frac{(AB)}{(AC)}(CD)$$

Substitute the corresponding lengths to get

$$y_p - P_1 = \frac{K_2 - K_1}{K_3 - K_1}(P_3 - P_1)$$

So, any triplet (K_1, P_1), (K_2, P_2) and (K_3, P_3) satisfies convexity if and only if

$$P_2 \leq P_1 + \frac{K_2 - K_1}{K_3 - K_1}(P_3 - P_1) \tag{2.7}$$

We derived conditions covering a greater class of arbitrage opportunities than the ones exploited by butterfly spreads. A butterfly spread involving calls, employs a triplet (K_1, C_1), (K_2, C_2) and (K_3, C_3), where K_2 is equidistant to K_1 and K_3, i.e $K_3 - K_2 = K_2 - K_1$. In this setting the violation of condition (2.6) is

$$C_2 > C_3 + \frac{1}{2}(C_1 - C_3) \Leftrightarrow -C_3 - C_1 + 2C_2 > 0$$

This is the position of a butterfly spread, which suggests to go long on the highest and lowest strike and buy two calls in the intermediary strike. It's easy to verify the well known result that the payoff of this position is always non-negative. Since the initial position is strictly positive, this is an arbitrage. Similarly one may verify that condition (2.7) is a general case of arbitrage opportunities created by butterfly spreads involving put options.

With Filter 7, we remove prices that form arbitrages. Specifically, for any consecutive triplet we check the appropriate condition and in case of violation we remove the middle price. Note that checking consecutive triplets is equivalent to check any triplet. The filter is implemented in Part 6 of Script-A.2. We found 1,137 arbitrage opportunities, 375 for calls and 762 for puts. The final filter F9 is implemented in Part 7 of the script, where we moreover export the "clean" data to a new csv file, called Clean.

2.3 Computing the series

Calculating the time series in (2.2) poses two challenges. The first one is that a constant horizon of maturities is needed. For each trading day, we need prices of options expiring in 19, 49, 79 and 109 days. We decided to interpolate as little as possible, resulting on taking approximate maturities. Essentially, for each day we have options maturing in the first, the second, the third and the forth month.

Approximate maturities, for the first three months, exist for almost all the trading days. There is only one date, that has no prices for the third month. In contrast, for many trading days, there are no available maturities for the forth month. To acquire them, as discussed below, we interpolate using the existing information.

The second challenge is that we need a continuum of strikes to calculate the integrals. To do this, a curve must be fitted to option prices across strikes. Subsequently, a numerical integration technique will approximate the value of the integral.

Following the literature mentioned in the beginning of Section-2.2 and the references therein, we experiment by interpolating prices and implied volatilities across strikes and call deltas using Merton's (1973) model. The model's functions are used as one-to-one mappings between implied volatilities and prices, maturities/strikes and call deltas. Their use does not impose any assumption about the dynamics of the underlying price. We are going to use the following transformations.

Denote the time-t price of the underlying by S_t, the strike price by K, the volatility by σ and the time to expiration by τ. Moreover, assume that the risk-free rate is zero and that S_t, K and τ are constants. Then, the one-to-one mappings between options' prices and implied volatilities are given by

$$C(\sigma; S_t, K, \tau) = \Phi(d_1)S_t - \Phi(d_2)K \tag{2.8}$$

$$P(\sigma; S_t, K, \tau) = \Phi(-d_2)K - \Phi(-d_1)S_t \tag{2.9}$$

for call and put prices respectively, with

$$d_{1,2} = \frac{1}{\sigma\sqrt{\tau}}\left[\ln\left(\frac{S_t}{K}\right) + \frac{\sigma^2}{2}\tau\right]$$

and $\Phi(\cdot)$ the normal cumulative probability function. Note, that in the scripts, we use the $d_2 = d_1 - \sigma\sqrt{\tau}$ relationship.

Now denote by σ^* the ATM implied volatility and assume that S_t and τ are constants. The one-to-one mapping of deltas and strikes is given by the following formulas

$$\delta_K = \Phi\left(\frac{1}{\sigma^*\sqrt{\tau}}\left[\ln\left(\frac{S_t}{K}\right) + \frac{\sigma^{*2}\tau}{2}\right]\right) \tag{2.10}$$

and

$$K_\delta = S_t \exp\left(\frac{\sigma^{*2}\tau}{2} - \Phi^{-1}(\delta)\sigma^*\sqrt{\tau}\right) \tag{2.11}$$

Following Panigirtzoglou and Skiadopoulos (2004), we use the ATM implied volatility for the transformations in order to retain the order of strikes (even if steep volatility skew are observed). In particular, we use the implied volatility corresponding to the strike closest to the underlying, since the exact ATM implied volatility is not available as prices are quoted in decimals and strikes are not.

Finally, we will use the option's vega which for zero interest rate and dividend yield is given by the following formula

$$V(\sigma;\, S_t,\, K\,, \tau) = S_t\phi(d_1)\sqrt{\tau} \tag{2.12}$$

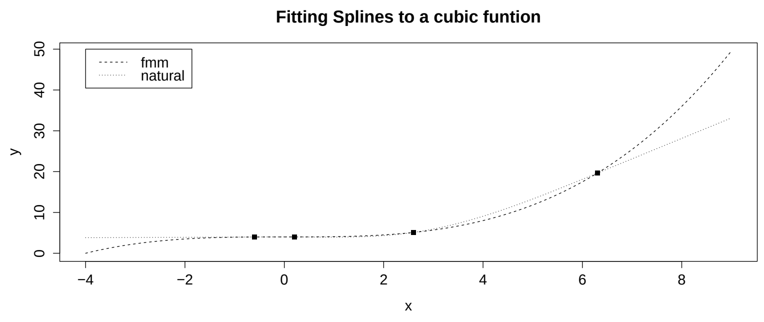
where $\phi(\cdot)$ is the probability density function of the normal distribution. The implementation of the formulas above are in Script-A.3.

2.3.1 Fitting uncertainty with splines

Interpolation for maturities and for prices/implied volatilities across strikes may be done using a cubic spline, a smooth spline or a weighted smooth spline. In this section we briefly discuss them, without presenting their numerical implementations (references are provided). We start with cubic splines.

Assume n-points $(x_1,\, y_1),\, (x_2,\, y_2),\, \ldots,\, (x_n,\, y_n)$ lie in the real two dimensional plane, with $a < x_1 < x_n < b$. A cubic spline, fitted to those points, is a continuous third order polynomial $S : [a, b] \to \mathbb{R}$, whose first and second derivatives are continuous too (a variant is Hermite splines, where S is only once differentiable continuously). Denote the class of such polynomials as $S_2[a, b]$. S is constructed as

Figure 2.2: We fit a cubic spline to the function $f(x) = (1/16)x^3 + 4$, in R using both boundary conditions. In R, the Forsythe, Moler, and Malcolm (1977) method is denoted by fmm.



a piecewise function consisting of $n-1$ third degree polynomials $S_k : [x_k,\, x_{k+1}] \to \mathbb{R}$ such that $S_k(x_k) = y_k$ and $S_k(x_{k+1}) = y_{k+1}$ for all positive integers $k < n$.

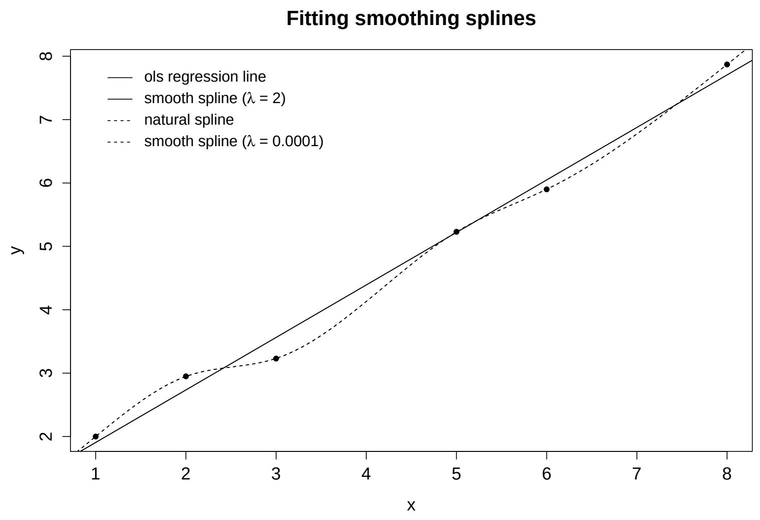
The continuity of S and it's first two derivatives fully determine the polynomials that start and end at an internal knot, i.e. S_k for $k = 2, 3, \ldots, n-2$. For S_1 and S_{n-1} boundary conditions must be given[5]. The boundary conditions for a natural cubic spline are such that $S_1''(x_1) = S_1'''(x_1) = S_{n-1}''(x_n) = S_{n-1}'''(x_n) = 0$, meaning that extrapolation is linear using the last two points at each end. In this case, cubic splines will be identical to any at most third degree polynomial only at the region defined by the internal knots. Another way, described by Forsythe et al. (1977) in Section-4.4, is to define boundary conditions using the last four points at each end. In this case the cubic spline will be identical to any at most third degree polynomial for all regions. Extrapolation for higher order polynomials will be based on information stemmed from S_1 and S_{n-1}.

Natural cubic splines have a very important statistical property. They are the ordinary least square (ols) regression lines of $y_i \sim x_i,\, x_i^2,\, x_i^3$. This is highly connected with the fact[6] that the average curvature of an natural spline S, i.e. $\int S''(x)dx$, is minimal for the curvatures of any other element of $S[a,b]$. You may observe this result in Figure-2.2. Between the boundary knots, the natural spline is less convex (has a smaller average curvature) than the Forsythe, Moler, and Malcolm (1977) one.

[5]For example see `http://mathworld.wolfram.com/CubicSpline.html`

[6]Green and Silverman (1993, Section 2.2.2.) provide formal derivations for all the observations we make about splines and for their numerical implementation.

Figure 2.3: We fit an ordinary least square (ols) regression line, a natural spline and two smooth splines for artificial data depicted by the plotted points. The ols line is identical to the natural spline for $\lambda = 2$ and the natural spline is identical to the smooth spline for $\lambda = 0$. This is a trivial example, where for each x_i there is only one y_i. In a more statistical context, where multiple y_i exist, the comparison between cubic splines and smooth splines makes no sense, as a natural spline is a function and cannot go through all the points. In this case though, the "statistical" comparison between an ols regression line and a smooth spline is, of course, still valid.



An extension of a natural cubic spline is the smooth spline, which is the function $g \in S_2[a, b]$ that minimizes the following quantity

$$\mathcal{L}(g, \lambda) = \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int_a^b (g''(x))^2 dx \tag{2.13}$$

for a given λ, which is called the smoothing parameter. The first term is the well-known Mean of Square Errors (MSE) and the second one is a penalty for the curvature of the curve, which we will denote as PC. The minimization problem suggests that we are willing to accept one unit increase in the MSE, as long as there is a decrease in $\int_a^b (g''(x))^2 dx$ greater or equal to λ,

Letting $\lambda \to \infty$ means that we do not care about the MSE and that we want PC as small as possible. In fact, by fitting a linear function, PC becomes equal to 0. So, in this case g will be the ols regression line. Note, that for practical applications a finite and low λ is needed to approximate the ols regression line. In fact, the more linear the data, a smaller λ is needed for the approximation to take place. In contrast, letting $\lambda \to 0$, suggests that we do not care about the curvature and that we want MSE to be as small as possible. We can achieve $MSE = 0$ by ensuring that $g(x_i) = y_i$. Moreover, since $\lambda \to 0$ and is not 0, we want g to have the smallest possible curvature. As we noted before, the cubic spline that goes through all the points and has the smallest possible curvatures, is the natural one. The limiting cases are shown in Figure-2.3.

To get a smooth spline, we must chose the smoothing parameter λ. As in many statistical problems, the researcher may subjectively choose λ based on her experience, or she may use some automatic method for its determination. The most common procedure is called cross-validation (cv). In general cv, a subset of the data are not used for the fitting but for assessing the generalization (predictability) of the fitting.

In smooth spline cv, for every observation x_i the predictability error is measured due to this observation given that the fitting was done using the sample excluding x_i. Then, λ is specified as the parameter minimizing this cumulative error across all observations. A slight variation is called generalized cross-validation (gcv), which uses the same technique as cv but with a different error metric. An important early reference of these is the paper of Craven and Wahba (1978). Another popular method, recommended by Hastie et al. (2009, see sec. 5.4.1)[7], is to choose λ by specifying the degrees of freedom willing to spend for the fitting. This method is considered to present a more intuitive choice due to its correspondence with the notion of degrees of freedom from the ols regression model.

Finally, we discuss a generalization called weighted smoothing spline, where g is the function minimizing the following quantity

$$\mathcal{L}(g, \lambda) = \sum_{i=1}^{n} w_i (y_i - g(x_i))^2 + \lambda \int_a^b (g''(x))^2 dx \tag{2.14}$$

The only difference with a smooth spline is that instead of the regular MSE, a weighted MSE is considered, in accordance with the weighted ols regression. The weights give the flexibility to penalize data differently. The automatic techniques

[7]This is a very good book, which may be downloaded for free from their website (`http://statweb.stanford.edu/~tibs/ElemStatLearn/`). In addition, there is an R package by Halvorsen (2012) implementing the examples of the book.

discussed for smooth splines are used for these too. The obvious motivation for the weights, emerge from data having y_i who are distributed with the same mean but different variances (heteroscedastic data). In this case the weights should be proportional to the reciprocal variances of the data. Their functionality though, far exceeds this apparent application.

We haven't discuss the numerical implementation of the methods above. We note, that they are very efficient as all the problems result in solving linear systems of sparse matrices or other easy linear algebra problems (a good reference is the book by Green and Silverman (1993)). All the above methods are implemented in R. In the build-in library, spline function fits a cubic spline and smooth.spline fits a smoothing spline (although a different parameter than λ is used). There is no built-in implementation of the weighted smooth spline. Packages by Nychka et al. (2014) and Ramsey and Ripley (2013) implement it with the sreg and smooth.Pspline functions respectively (and for the usual λ). The above discussion should render their documentations apprehensible.

2.3.2 Maturities

For the last trading day of each month we need prices of options expiring in the four following months. Bakshi et al. (2011) consider that these maturities correspond roughly to

$$\tau_i = 19,\ 49,\ 79 \text{ and } 109 \text{ days.}$$

Based on our sample we define

$$\tau_1 \in [16,\ 23], \quad \tau_2 \in [43,\ 53], \quad \tau_3 \in [78,\ 85], \quad \tau_4 \in [107,\ 114]$$

All dates but one have option prices expiring uniquely in the above first three intervals. In 30-06-1999 there is no expiration higher than 17 days and lower than 80 days. To create the new maturity we work as follows.

At first we discard any strike that does not have at least two maturities or it's range of maturities does not contain the desired one, in order to secure interpolation (extrapolation behaved very badly for natural, fmm and smooth splines). For each strike that survives, we translate option prices to implied volatilities. If there are less than four maturities we fit a natural cubic spline to implied volatilities as a function of expirations. If there are more than three maturities we fit a smoothing spline, with the smoothing parameter being automatically chosen by R using general cross validation. From the fitted line we extract the implied volatility for the desired maturity, which in this case equals 49. Finally, the new volatility is transformed back to a price.

After we do this for all strikes, we check the new prices for arbitrages as explained in Section-2.2 and discard those that are less than the minimum tick, which is equal to 0.05[8] (note that we did not check whether this filter was enabled), using Script-A.5. The whole procedure is done in Script-A.4 and is the same that we use to create new prices for options expiring in 109 days. The example at the end of the script runs the corresponding function for creating new prices for options at 30-06-1999 maturing in 49 days. The function has the option to plot the fitting for each strike.

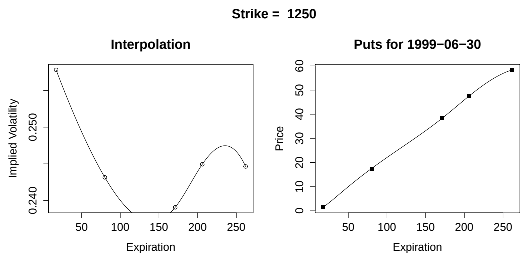
Prior to concluding to the previous method, others were tried. At first, we tried to interpolate linearly using maturities less than a year. This method produced a small amount of new prices, as strikes satisfying this condition were not as frequent as we hoped. There were even instances that no strike satisfied the condition. Secondly, we included every available maturity and extrapolated as well. Extrapolation did not behave reasonably and produced low-quality results. This is why we did not included it in our final method, even though scarcity of produced values was observed.

Subsequently, to include information from other maturities as well as the ones surrounding the desired one, we fitted cubic splines. In Figures 2.4a, c two instances of this method are shown. Concerns about overfitting depicted in Figure-2.4a led us to chose smoothing splines for samples with more than three maturities (to interpolate using smooth splines, at least four observations are needed). Overfitting was addressed as shown in Figure-2.4b.

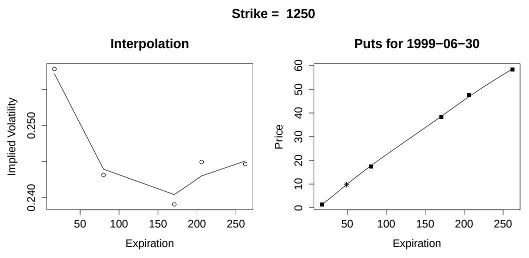
We do not provide the code for the initial methods, as there were slightly different from the one in Script-A.4. Finally, we note that the resulting curve took four different forms. It was either concave, convex or had a mild horizontal (like Figure-2.4d) or vertical s shape.

[8]The product description for options on the S&P 500 index is given at `http://www.cboe.com/products/indexopts/spx_spec.aspx`

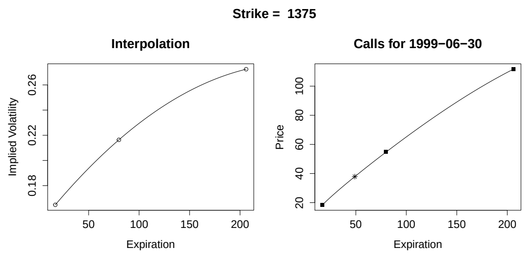
Figure 2.4: These graphs were produced while creating new prices at 1999-06-30 for an option expiring in 49 days. In each graph the original points are depicted along with the fitted lines. The star denotes the interpolated volatility. Graphs (a) and (b) are for a put option and a strike equal to 1250. There are four maturities greater than 16 days and less than a year. In particular, there are prices for options expiring in 17, 80, 171, 206 and 262 days. Graph (c) is for a call option and a strike equal to 1375 with expirations in 17, 80 and 206 days. Graph (d) is for a put option and a strike 1175 with two expirations in 17 and 80 days.



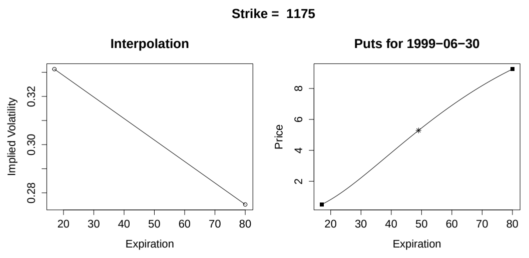
(a) The fitting was done using a cubic spline.



(b) The fitting was done using a smoothing spline with the smoothing parameter automatically chosen by R.



(c) This one an average case where three expirations exist.



(d) A slight s shape.

2.3.3 Extracting the risk-neutral price curve

The next step is to calculate the integral in (2.2). Recall from Proposition-4 that these integrals were derived by taking a risk neutral expectation restricted in a region for ATM and OTM strikes. Prior explaining the method we used, we provide a small review of the literature for estimating risk neutral probability density functions.

As Kang and Kim (2006) mention, estimating the risk neutral distribution can be done by parametric or non-parametric methods[9]. Our model free approach lies in the non-paramteric space, which is divided into kernel methods (e.g. Aït-Sahalia and Lo (1998)), maximum entropy methods (e.g. Buchen and Kelly (1996) and

[9]Some parametric methods and Shimko's (1993) method are implemented in R by Hamidieh (2014).

Neri and Schneider (2013)), and curve fitting methods. Following the recent literature cited at the beginning of the section we choose the last one. The chronology of curve fitting methods is presented by Jackwerth's (1999) and (2004) reviews which are partially presented below.

Curves may be fitted to prices or implied volatilites across strikes or deltas using the non-linear transformations (2.8)-(2.11). Shimko (1993) introduced this method by fitting a quadratic polynomial to implied volatilities with respect the strike prices. Brown and Toft (1999) extended it by using seventh-order polynomials (splines). Malz (1997) did the same, but used deltas instead of strikes in order to decrease non-linearity, as imlpied-volatility is better behaved across deltas than strikes. Campa et al. (1998) used cubic splines to fit the volatility smile and Rosenberg and Engle (2002) fit polynomials to logs of volatilites, in order to prevent negative implied volatilites. The reviews are concluded with the work of the author, Jackwerth (2000), where in his own words maximizes the smoothness of the smile and controls the trade-off between option price fit and smoothness explicitly. He uses a smooth spline but multiplies with λ the MSE and with $1 - \lambda$ the curvature error.

Bliss and Panigirtzoglou (2002), (2004) use the method developed by Panigirtzoglou in previous unpublished work at the Bank of England. They fit a weighted smooth spline to implied volatilites across deltas. They choose as weights the option vega ($v \equiv \partial C / \partial \sigma$) to account for presumed homoskedastic pricing errors in the underlying raw price data, such as those resulting from discrete tick size. Moreover, vega weighting places less weight on-away-from-the-money strikes, which have generally low liquidity. In the first paper they use smoothing parameters ranging from 0.01 to 0.0001 and in the second from 0.99 to 0.9999. They note that forecast was insensitive to the choice of λ in these regions and thus they report results using lambdas equal to 0.01 and 0.99 respectively. In the first paper they extrapolate linearly (as the natural spline) and in the second one horizontally.

A slight variation of the above is implemented by Panigirtzoglou and Skiadopoulos (2004), and Neumann and Skiadopoulos (2013). Instead of using vega weighting, they discard altogether observations with deltas greater than 0.99 and less 0.01. Outside these margins, Neumann and Skiadopoulos (2013) extrapolate horizontally. Moreover, to ensure an appropriate sample, they only create an implied volatility curve if there are observations with deltas less or equal to 0.25 and greater than or equal to 0.75.

Jiang and Tian (2005) following Bates (1991) and Campa et al. (1998) use natural cubic splines in the curve-fitting of implied volatilities. They say that cubic splines has the advantage that the obtained volatility function is smooth everywhere and provide an exact fit to known implied volatilities. A similar choice was made by Kostakis et al. (2011) for American-style S&P 500 future options.

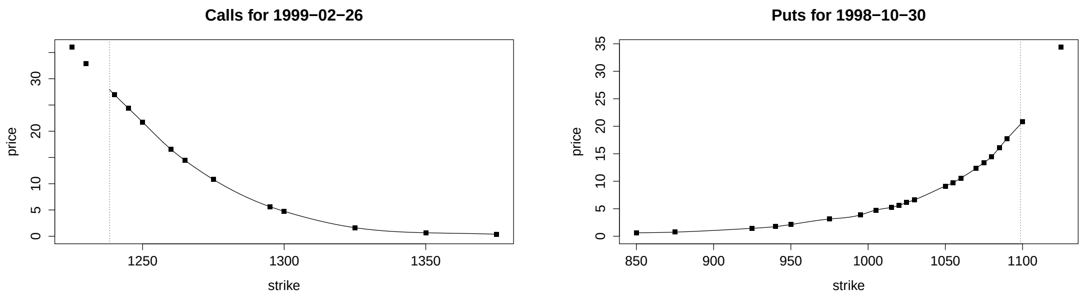
Kang and Kim (2006) use a slightly different method from the one proposed by Panigirtzoglou, which is very similar to the one proposed by Jackwerth. They multiply weights by the smoothing parameter and get the following variation of the minimization problem described by (2.14)

$$\mathcal{L}(g,\lambda) = \lambda \sum_{i=1}^{n} w_i(y_i - g(x_i))^2 + (1-\lambda)\int_a^b (g''(x))^2 dx \tag{2.15}$$

Similarly to Bliss and Panigirtzoglou (2004), they report insensitivity of empirical results to the choice of λ in the region $(0.990, 0.999)$. Moreover they note that finally they chose equal weights across the observations, as they found that the results were robust to the choice of alternative weighting schemes between observations, such as vega weighting. In a more recent work of theirs, Kang et al. (2014) use ordinary least squares to estimate the coefficients of Shimko's (1993) quadratic polynomial. Note, that this is similar with the smooth spline for a large enough lambda (in the case of implied volatilities a λ around 1 is sufficient), but not for a cubic polynomial but for a quadratic one.

Inspired by the literature above, we experimented with various methods. Recall, that following Jiang and Tian (2005) we have kept call option prices with strikes greater than the underlying multiplied with 0.97 and put prices with strikes less than 1.03 of the underlying. The first method we present is when we interpolated prices across strikes using a cubic spline, with fmm boundary conditions for a more horizontal extrapolation. In Figure-2.5 two instances of this method are available. As expected, overfitting problems occurred, which would lead to low quality predictions.

Figure 2.5: The simplest method we used was to interpolate prices across strikes, using cubic splines with the fmm boundary conditions. Over-fitting problems are apparent, especially when a lot of observations are available. The vertical dotted line at each graph is the price of the underlying.



(a) For calls in 26-02-1999.

(b) For Puts in 30-10-1998.

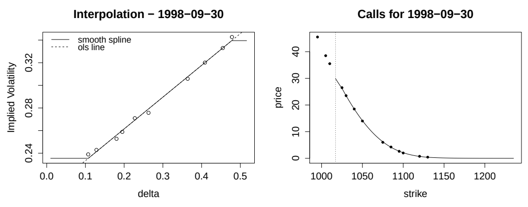
The second method we present is the one used by Neumann and Skiadopoulos (2013), as it is one of the most recent ones based on the fitting of smooth splines in the implied volatility - delta space. To implement it we worked as follows. We discard any ITM option left in our sample. We convert prices to implied volatilities and strikes to deltas, using the implied volatility of the strike closest to the price of the underlying. We discard observations for deltas less than 0.01 and greater than 0.99 for calls and puts accordingly[10]. For the remaining observations we use a smooth spline with $\lambda = 0.99$ to interpolate volatilities across deltas[11]. Subsequently we create a delta grid with 500 points. For calls the grid starts at 0.01 and ends to the delta corresponding to the ATM option. For puts it starts at the ATM option and ends at 0.99. For unknown deltas in the grid, implied volatilities are extrapolated horizontally using the closest known value.

This method addresses overfitting problems. In our opinion, this method is based on an underlying assumption. It assumes that a volatility smile exists and moreover that across deltas it consists of two "noisy" linear parts, the left and the

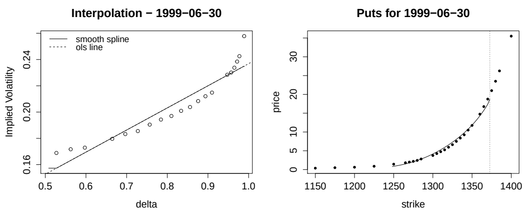
[10]For call options, deltas start around 0.5 for strikes close to the underlying and decreases as we move out-of-the money, i.e. as the strike increases. For put options, deltas start around 0.5 for ATM options and increases as we move out-of-the money, i.e. the strike decreases.

[11]The implementation we used was the sreg function by Nychka et al. (2014). Better results were obtained when we used the built-in smooth.spline with the spar parameter equal to 0.99. Using the spar parameter fitting was better for non linear data. Information for the spar parameter is given in the function's documentation. Green and Silverman (1993) may be found useful for its understanding.

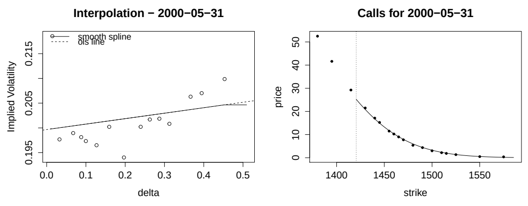
Figure 2.6: Some instances provided by the implementation of the method used by Neumann and Skiadopoulos (2013). In each figure, the left plot shows the interpolation of implied volatilities across deltas. For comparison, the ols regression line is provided. At the right plot the produced fitting of prices across strikes is shown. The vertical dotted line is the price of the underlying.



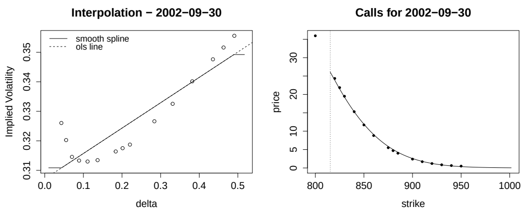
(a) Horizontal extrapolation of implied volatilities near the underlying leads to concavity (violation of arbitrage pricing).



(b) The fitted implied volatility curve is very close to the linear regression line, even though the data are not linear. Moreover many far out-of-the money observations are left out due to the range of the delta grid.



(c) Very noisy data of implied volatilities acrros delta. The produced fitting for prices seems very good.



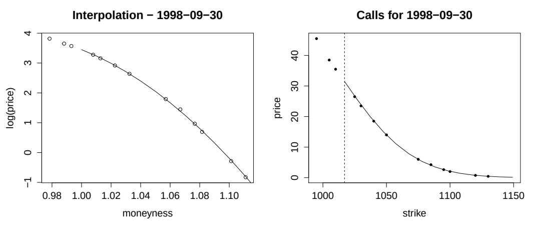
(d) Same problem with graph 2.6c. Due to the non-linear transformations the produced fitting behaves quite well.

right half smiles. The two halfs are connected with a horizontal line corresponding to the ATM region. Since implied volatility across deltas is a very smooth function and in addition takes it's global minimum at this region, the horizontal extrapolation makes sense. To emphasize these observations, we provide the ols regression line in the plots depicted in Figure-2.6.

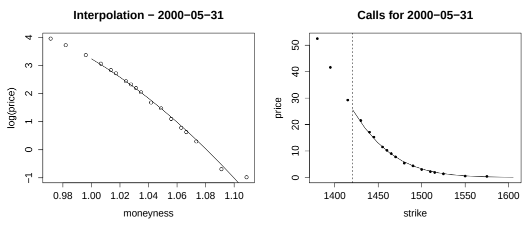
In our sample some problems occurred. In some instances, the horizontal extrapolation produces prices that form arbitrage opportunities. More importantly, the underlying linearity assumption is violated quite often in the sample which makes the predictability of our fitting questionable. Note that these problems occurred in our implementation and moreover that our implementation is only the second half of the method used by Neumann and Skiadopoulos (2013). They also interpolate to create constant maturities. The method is in Script-A.6.

Finally, inspired by the literature, we propose and try another method. Our

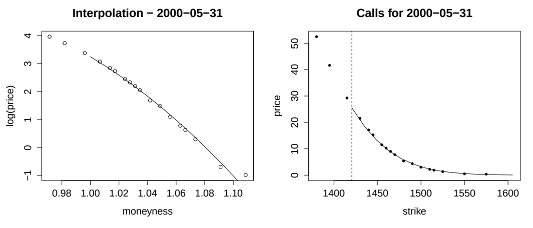
Figure 2.7: The same instances with Figure-2.6 created by interpolating in the price/moneyness space. Moneyness is defined as strike divided by the underlying.



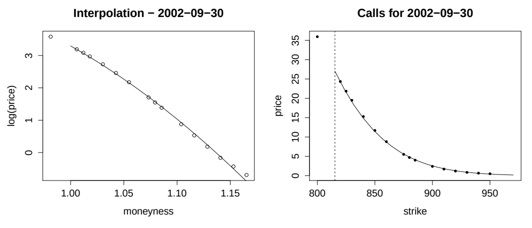
(a)



(b)



(c)



(d)

underlying assumption is driven by the fundamental results of free arbitrage pricing discussed in Section-2.2. We know that the price curve across strikes must be strictly monotonic and convex. That is why we started, by using ordinary least squares to estimate the coefficients of a quadratic polynomial in the price/strike space. This did not work, since the convexity of the price curve is not constant across strikes and in particular it changes quite rapidly as we move away from the money.

To address this, we tried to fit a cubic polynomial but overfitting problems occurred. Another way to lower convexity's rate velocity is to fit the logarithm of prices. In our sample we observed that convexity of log prices is constant across strikes and in some cases close to zero. The latter could suggest overfitting problems. Moreover, to address numerical approximation errors due to the exponential transformation and to have homogeneity across different maturities in the price grid, we interpolate across a form of moneyness (strikes divided by the underlying) instead of strikes. Finally, to address issues concerning the far OTM and ITM used for the fitting, we use vega weighting. ITM observations are weighted by the symmetrical far OTM weights. Tails are approximated by multiplying with 1.02 the maximum observed call moneyness and with 0.97 the minimum put moneyness. The accuracy of the grid used is 0.001, which translates roughly to \$1 in the strike space.

We would like to note that apart from the fitting advantages of this method, prediction quality stems from the theory of linear models. Moreover there is no need for specifying a smoothing parameter. Finally we emphasize, that we do not assume anything about the underlying process. We only assume that the price curve across strikes is convex, which is a fundamental arbitrage free hypothesis. The ols regression line is used to produce a second degree natural "spline". It's implementation is in Script-A.7 and for comparison Figure-2.7 is provided.

While we fit the curves, we compute the integral (2.2) as well, using the omega function (2.3) implemented in Script-A.3. In particular, once we have created the strike grid and the corresponding grid of the fitted prices, we multiply prices by $\omega(K)$, where K is the strike. Then, to integrate, we use the trapezoidal method implemented by function trapz of the pracma package by Borchers (2014).

2.4 Calculation of Forward Variances

In the final section of this chapter, we discuss the creation of the time series (2.5), done by Script-A.8. We first load the filtered data created in Section-2.2, which contain 17,421 observations. Some statistics about the number of observations for each maturity are show in Table-2.1.

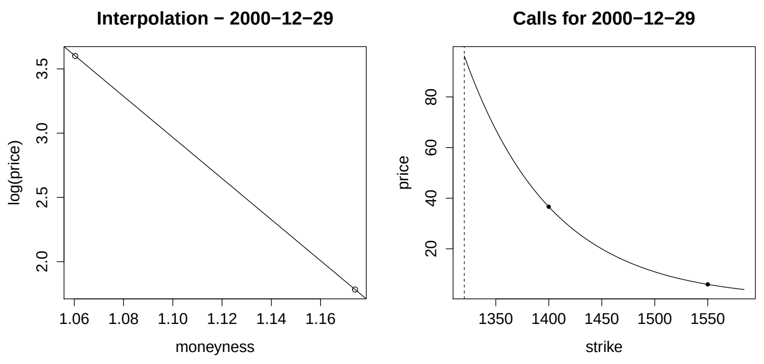
Table 2.1: Basic statistical measures about the observations in each maturity.

	Original				By Interpolation			
	$H_t^{(t,1)}$	$H_t^{(t,2)}$	$H_t^{(t,3)}$	$H_t^{(t,4)}$	$H_t^{(t,1)}$	$H_t^{(t,2)}$	$H_t^{(t,3)}$	$H_t^{(t,4)}$
1	242	240.0	242.0	96.0	242	242.0	242.0	242.0
Min.	6	6.0	1.0	1.0	6	6.0	4.0	2.0
1st Qu.	14	10.0	6.0	7.0	14	10.0	7.0	6.2
Median	17	15.0	8.0	9.0	17	15.0	9.0	8.0
Mean	19	16.4	9.8	9.4	19	16.3	10.1	8.7
3rd Qu.	22	21.2	12.0	12.0	22	21.0	12.0	10.0
Max.	48	41.0	28.0	20.0	48	41.0	28.0	20.0

For the first maturity ranging from 16 to 23 days, each date had at least 6 observations per call and put options and so integrals corresponding to time series $H_t^{(t,1)}$ were immediately calculated. For the second maturity ranging from 44 to 53 days, there were no observations for 30-06-1999. As discussed in Section-2.3.2 we interpolate implied volatilities across maturities to create observations for this date and then integrate.

For the third maturity ranging from 75 to 86 days there were observations for each date, but there were ten days who had less than four observations for either calls or puts. Again, we interpolated maturities to get more observations. In particular, for strikes that do not correspond to known values, we interpolated as discussed in Section-2.3.2. The created observations were combined with the known ones, and all the prices were once again checked for arbitrage and that they are above the minimum tick.

Figure 2.8: Plots corresponding to the fitting of the curve for calls in 29-12-2000, where only two observations were generated by interpolating implied volatilities across known maturities.



For the fourth maturity ranging from 107 days to 114 days, there were 73 days missing. For those we interpolate implied volatilities across maturities. This method produced less than four observations for 4 out of the 73 dates. In particular, in 29-12-2000 only two observations were created, see Figure-2.8. For this date, predictability concerns have to be taken into account. Moreover, note that for integration methods that are based on the fitting of smooth spline at least for observations are needed, which means that we had to extrapolate in order to acquire the required information. For the remaining days that had less than four observations we worked as in the third maturity.

Finally, once we calculated the series in (2.2) we calculated the forward variances (2.5), which are shown in Figure-2.9. For comparison we provide the plot by Bakshi et al. (2011) in Figure-2.10.

Figure 2.9: Time-series for the forward variances defined in (2.5). The time-series corresponds to last trading days of each month from September 1998 to September 2008. The underlying asset is the S&P 500 index. European calls and puts were used to calculate the integrals of (2.2). LIBORs were used as a proxy for the risk-free rate.

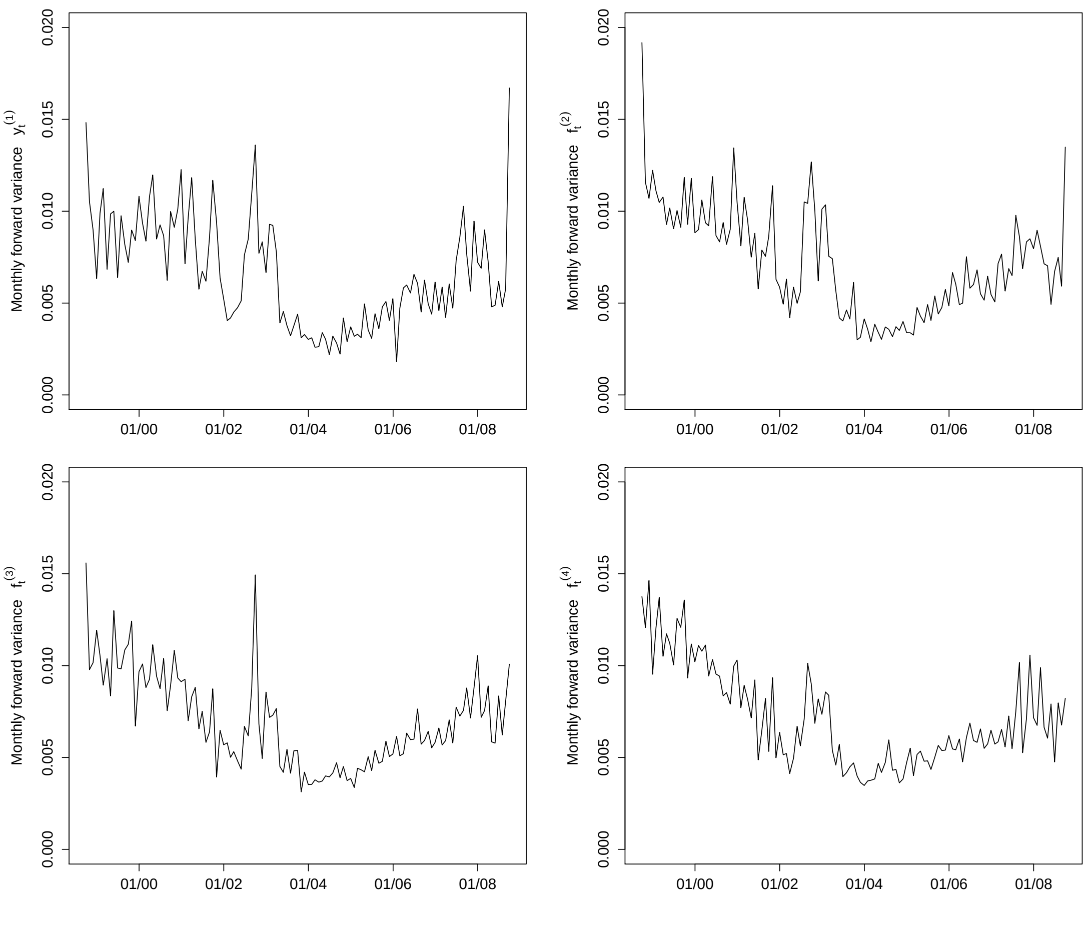
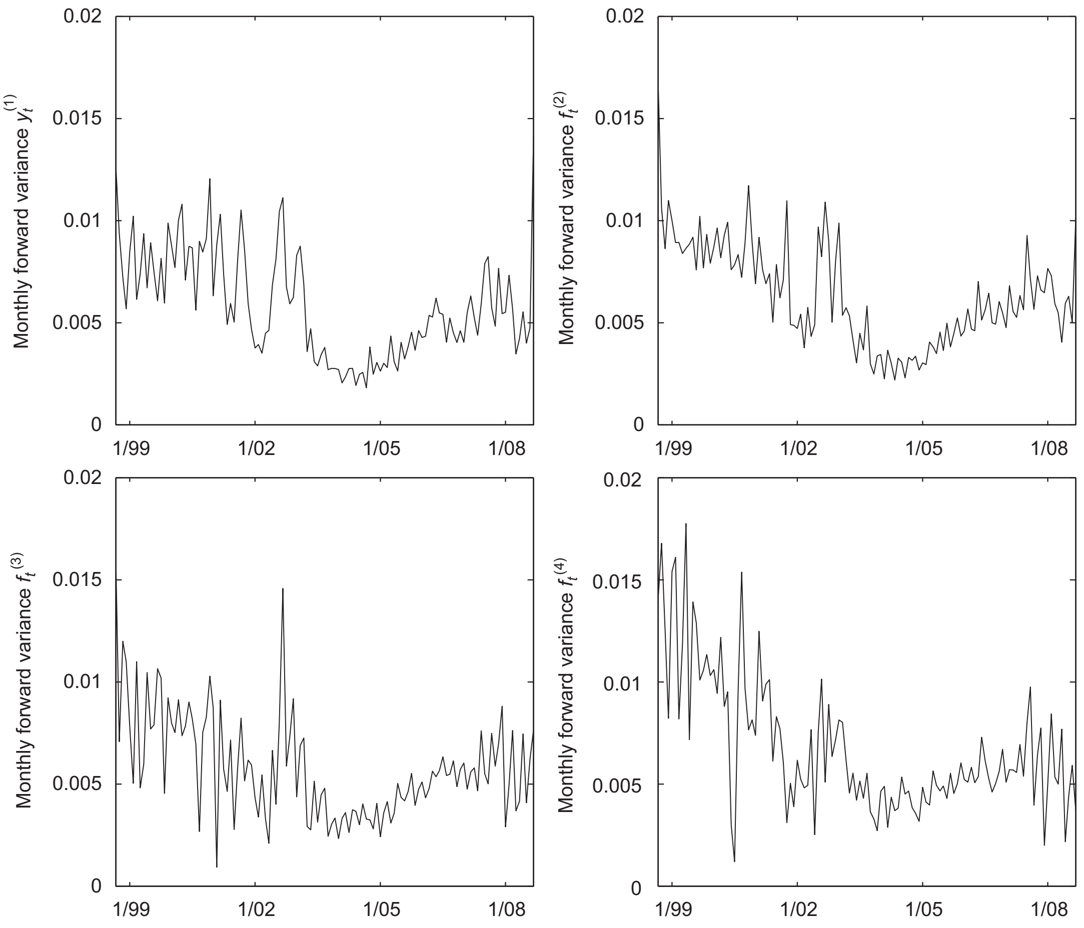


Figure 2.10: Time-series behavior of forward variances as depicted in Figure-1 by Bakshi et al. (2011). We provide these graphs for comparison with the ones we produced, as shown in Figure-2.9.



Chapter 3

Predictability of Forward Variances

In this chapter we study the forward variances produced in Chapter 2. Their statistical properties are investigated in the first section and their predictability for real economic activity is examined in the following one. The analysis is based on the paper of Bakshi, Panayotov, and Skoulakis (2011) and the results provided follow their format. It should be take into consideration that our results are highly biased toward the construction method we used in Cahpter 2 and the filtering of the initial data. In Section-3.1.1 we also provide the definitions of the most popular time series models and some of their basic theory. The analysis was done using R. We also use other packages referred to in the text. Finally, the time series analysis is done in Script-A.9 and the regressions in Script-A.10.

3.1 Statistical Properties of Forward Variances

The forward variances (fvars) created in the previous chapter are shown in Figure-3.1. Table-3.1 contains some basic statistical measures on the constructed series. We observe that on average fvars start around 0.4% and stabilizes close to 0.7% for maturities further in time. In Section-3.1.2 the statistical properties of the time series are discussed. In the following section, prior to the analysis of fvars, we provide some basic definitions on time series based mostly on Cowpertwait and Metcalfe (2009).

Figure 3.1: Time-series behavior of forward variances in the same graph. Note that $y_t^{(1)}$ has been converted to a 30-day rate.

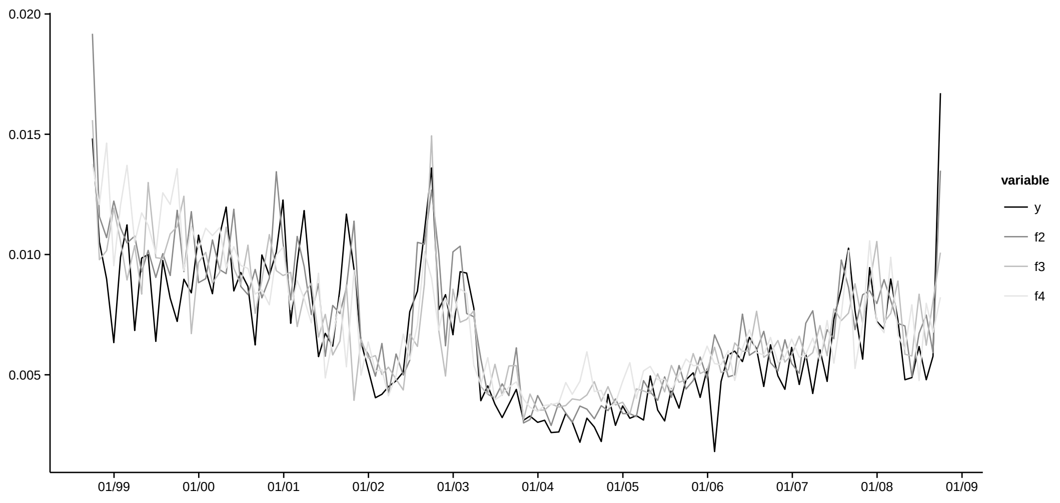


Table 3.1: Some basic statistics about the forward variances produced in Chapter 2. The sample period is 09/1988 to 09/2008 (121 observations) and values are percentages.

Statistic	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
y_t	0.4182	0.1860	0.1145	0.2780	0.5481	1.0549
$f_t^{(2)}$	0.7214	0.2908	0.2891	0.4915	0.9206	1.9178
$f_t^{(3)}$	0.6979	0.2544	0.3131	0.5041	0.8785	1.5588
$f_t^{(4))}$	0.7160	0.2661	0.3478	0.5149	0.9222	1.4632

3.1.1 Time Series basics

A stochastic process $\{x_t\}_{t \in \mathcal{T}}$ is said to be stationary if for for all $k \in \mathbb{R}$ and $\tau, t_1, \ldots, t_k \in \mathcal{T}$

$$F_X(x_{t_1+\tau}, \ldots, x_{t_{k+\tau}}) = F_X(x_{t_1}, \ldots, x_{t_k})$$

where F_X denotes the joint distribution. A weaker form, known as second-order stationarity, requires that only the first two moments be constant across time, i.e. $E[x_t] = \mu$ and $V[x_t] = \sigma^2$ for all $t \in \mathcal{T}$. The second-order term is usually omitted and such processes are referred as stationary.

A very simple process, which is used as a building block for more complex models, is called white noise and is a discrete stationary stochastic process w_t such that $E(w_t) = 0$ and $V(w_t) = \sigma_w^2 < \infty$, for all $t \in \mathcal{T}$. White noise is used for example to create random walks $x_t = x_{t-1} + w_t$. Using backward substitution we may rewrite a random walk as a function of consecutive white noises

$$x_t = x_0 + \sum_{k=1}^{t} w_k,$$

where x_0 is the known starting point of the process. From the above form we may easily observe that a random walk is not stationary, since $V[x_t] = k\sigma^2$.

Non stationarity is usually detected in correlograms. Correlograms are statistical plots created under the null hypothesis that the first two moments of the series are constant. Under the null hypothesis, the two moments are estimated using the usual unbiased estimators on the series' sample terms, i.e. the constant mean is $\hat{\mu}_x = 1/n \sum_{t=1}^{n} x_t$ and the constant variance is $\hat{\sigma}_x^2 = 1/(n-1) \sum_{t=1}^{n} (x_t - \mu_x)^2$. In particular, correlograms plot sample autocorrelations $r_k = c_k / c_0$, where

$$c_k = \frac{1}{n} \sum_{t=1}^{n-k} (x_t - \hat{x})(x_{t+k} - \hat{x}),$$

across different lags k. Under the null hypotheses c_k values should be near to zero. If there is a trend or a seasonal effect, correlograms will picture it. For example, a correlogram for a random walk will rapidly decay for initial lags, as x_{t+k} becomes less and less correlated to x_t as k increases. If an event takes place at every k steps of the series, then a spike will be depicted on lag k with somewhat elevated autocorrealtions around it.

An autoregressive model of order p, denoted with AR(p), is a process such that

$$x_t = a_1 x_{t-1} + \ldots + a_p x_{t-p} + w_t, \quad \text{with } a_p \neq 0 \tag{3.1}$$

Using the so called *backward shift operator* $Bx_t = x_{t-1}$, (3.1) may be rewritten as

$$\Theta_p(B)x_t = (1 - a_1 B - a_2 B^2 - \ldots - a_p B^p)x_t = w_t \tag{3.2}$$

It is proven that an AR(p) process is stationary if and only if the roots of $\Theta_p(B) = 0$ lie inside the unit circle, i.e. are less than 1. A moving average process, denoted as MA(q), is a stationary process x_t such that

$$x_t = w_t + b_1 w_{t-1} + \ldots + b_q w_{t-q} \tag{3.3}$$

Using the B operator (3.3) may be rewritten as

$$x_t = \Phi_q(B)w_t = (1 + b_1 B + b_2 B^2 + \cdots + b_1 B^q)w_t = \Phi_q(B)w_t \tag{3.4}$$

Combining linearly the two previous models, an Autoregressive Moving Average process is created, denoted as ARMA(p,q), such that

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \ldots + a_p x_{t-p} + w_t + b_1 w_{t-1} + \ldots + b_q w_{t-q} \tag{3.5}$$

or in the equivalent form

$$\Theta_p(B)x_t = \Phi_q(B)w_t \tag{3.6}$$

You may think of an ARMA(p,q) model as a generalization of a random walk. The difference is that the new step does not depend only on the current step x_t and the incoming noise w_t, but also on the previous p steps and q noises. Such models may be used for homoskedastic (constant volatility) processes with memory and trends on the mean.

The correlogram corresponding to the $\{x_t - \hat{\mu}_x\}_{t\geq}$ series of a heteroskedastic process x_t will show trends and spikes. To account for such processes we have to model volatility too. In particular, a Generalised Autoregressive Heteroskedastic, denoted as GARCH(p,q), model may be employed for the error term. A process ϵ_t is said to be a GARCH(p,q) model if

$$\epsilon_t = w_t \sqrt{h_t}, \quad \text{where } h_t = c + d_1 h_{t-1} + \ldots + d_q h_{t-q} + c_1 \epsilon_{t-1}^2 + \ldots + c_p \epsilon_{t-p}^2 \tag{3.7}$$

A GARCH(p_g, q_g) model is simply an ARMA(p_g, q_g) model for the error variance of the series. The right equation in 3.7 is an ARMA(q_g, p_g) model for the conditional variance of the series as ARMA(p, q) is the model for the conditional mean of the series. Also, note that the corresponding autoregressive GARCH$(p_g, 0)$ model is denoted as ARCH(p_g) and the moving average GARCH$(0, q_g)$ as GARCH(q_g).

A combination of the last two models will allow for trends in the mean and the volatility. An ARMA(p, q) - GARCH(p_g, q_g) model is a process defined by the following equations

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \ldots + a_p x_{t-p} + \epsilon_t + b_1 \epsilon_{t-1} + \ldots + b_q \epsilon_{t-q} \tag{3.8}$$

$$\epsilon_t = w_t \sqrt{h_t} \tag{3.9}$$

$$h_t = c_1 \epsilon^2_{t-1} + \ldots + c_{p_g} \epsilon^2_{t-p_g} + d_1 h_{t-1} + \ldots + d_{q_g} h_{t-q_g} \tag{3.10}$$

As an example, we provide an ARMA(1, 1) - GARCH(1, 1) model. Since each of the contributing models, i.e. AR, MA, ARCH, GARCH, have only one parameter, we will indicate the coefficients by their corresponding models' name.

$$x_t = \mathrm{AR} x_{t-1} + \epsilon_t + \mathrm{MA} \epsilon_{t-1} \tag{3.11}$$

$$\epsilon_t = w_t \sqrt{h_t} \tag{3.12}$$

$$h_t = g_0 + \mathrm{ARCH} \epsilon^2_{t-1} + \mathrm{GARCH} h_{t-1} \tag{3.13}$$

For this model it is very easy to test stationarity, since $\Theta(B) = (1 - \mathrm{AR})$. In particular, the series will be stationary if $-1 < \mathrm{AR} < 1$. An alternative to test the null hypothesis that the series is non-stationary ($H_0 : \mathrm{AR} = 1$) is to use the Phillips-Perron (Phillips and Perron, 1988; Perron, 1988) test, which is more robust (as it uses a non parametric correction to the test's statistic) than the augmented Dickey-Fuller statistic (Said and Dickey, 1984) for series with unspecified autocorrelation and heteroskedasticity in the disturbance process.

3.1.2 Statistics of Forward Variances

Stationarity hypothesis of the series bears fundamentally to the predictive quality of regressions employing them as predictors. Theory suggests that the forward variance process might be a stationary process. Recall (see equations (2.1) and

(2.5)) that forward variance rates are defined as the log returns of claims on the exponential of integrated variance. Now, if we assume that instantaneous variance σ_t is a mean-reverting (Ornstein-Uhlenbeck) process then stationarity for forward variance rates is inherited[1]. Bakshi et al. (2011) provide references supporting this hypothesis and, by Figure-3.1, we observe that there is no strong evidence against stationarity, as the mean and the variance of the processes seem to be constant across time.

Table 3.2: Values for several criteria for two ARMA-GRACH models. The ARMA(1,1) GARCH(1,1) is superior according to all criteria.

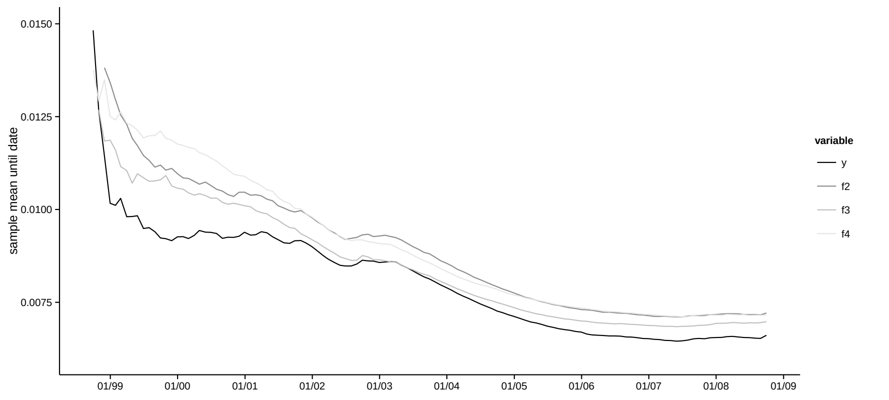
	$y_t^{(1)}$	$f_t^{(2)}$	$f_t^{(3)}$	$f_t^{(4)}$
ARMA(1,1) - GARCH(1,1) model				
Akaike	-10.088	-9.373	-9.767	-9.988
Bayes	-9.973	-9.257	-9.652	-9.872
Shibata	-10.091	-9.376	-9.771	-9.991
Hannan-Quinn	-10.041	-9.326	-9.721	-9.941
ARMA(2,2) - GARCH(2,2) model				
Akaike	-9.963	-9.258	-9.681	-9.815
Bayes	-9.755	-9.050	-9.473	-9.607
Shibata	-9.974	-9.268	-9.691	-9.825
Hannan-Quinn	-9.879	-9.174	-9.596	-9.730

[a] Fittings and calculations by the Ghalanos (2014) package.

Stationarity may also be graphically checked by an ergodic mean plot, where cumulative sample means are plotted against dates. Figure-3.2 suggests that the stationarity assumption is reasonable, since the mean seems to converge, even though the sample is small. To further investigate the series we fit a model. Correlograms in Figures 3.3 and 3.4 depict trends both in the mean and the variance of the series. Following Bakshi et al. (2011), ARMA(p,q) - GARCH(p,q) models are considered. We tested different models and compared them according to several information criteria. The results for two such models appear in Table-3.2. We concluded that an ARMA(1,1) - GARCH(1,1) is suitable for modeling forward variances.

[1]To elucidate this concept one may think of Brownian Motion (BM), which is a special case of mean-reverting processes. It is very popular to model a stock price as a Geometric Brownian Motion (GBM), which is an exponential on a BM, i.e. $S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}t\right) + \sigma W_t\right)$. It is well known that even though the stock price has a drift μ which makes it non-stationary the log returns of a stock price are stationary. Indeed, observe that $\log\left(\frac{S_{t+\tau}}{S_t}\right) = \left(\mu - \frac{\sigma^2}{2}t\right)\tau + \sigma(W_{t+\tau} - W_t)$.

Figure 3.2: An ergodic mean plot of the time series. Note that $y_t^{(1)}$ has been converted to a 30-day rate and that the sample contain a small number of observations (121).



The fitted parameters are in Panel D of Table-3.1. The autoregressive fitted coefficients are near the unit. This causes concern for non-stationarity as explained in Section-3.1.1. In Panel E we provide the standard errors for the AR coefficient estimations as proxy for the near-unit-root behavior of the model. Finally, taking into consideration the Phillip-Perron test of Panel C and the previous analysis we conclude that there is strong evidence to not reject the stationarity hypothesis.

3.2 Predictability of Forward Variances for real Economic Activity

In this section we use the constructed forward variances to predict real economic activity following Bakshi et al. (2011). The proxies for real economic activity are monthly growth rates of non-farm payroll and industrial production, as these are considered measures for employment and real output respectively. We also include the slope of the Treasury yield curve as a predictor, proxied as the difference between the ten-year and the three-month Treasury yields, in order to compare the predictability power of forward variances.

We remind that our results are highly biased towards the method we used to construct the series. The industrial production and the Treasury yields were downloaded from the Board of Governors of the Federal Reserve System (http:

Figure 3.3: Corelograms for the mean of forward variances. The slow decay suggests that a simple autoregrssive model, such the random walks, may not be suitable and that an ARMA model should be employed.

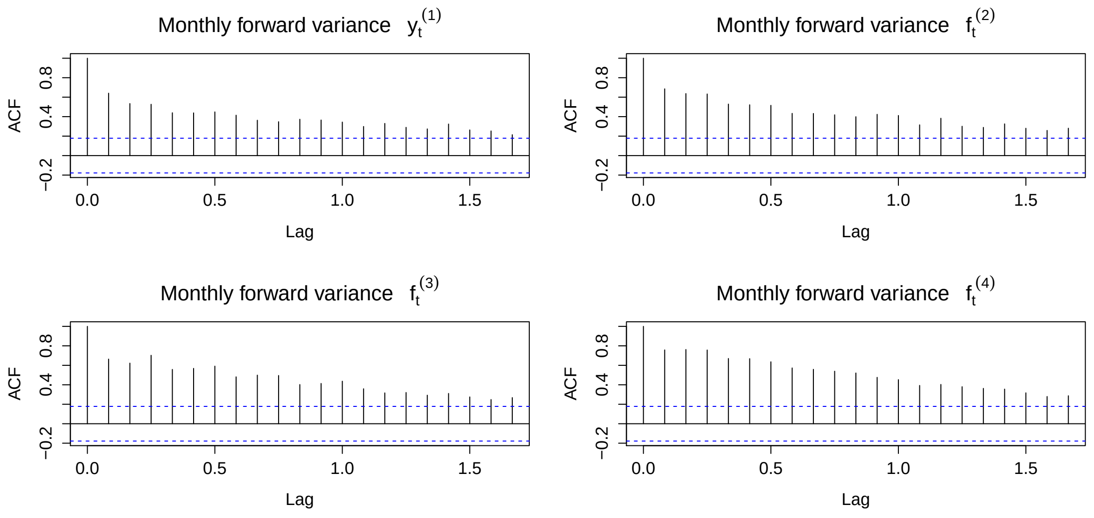


Figure 3.4: Corelogram for the variance of forward variances. Similar patterns as for the mean are depicted. They are most noticeable for $f_t^{(4)}$. Consistency suggests that an GRACH model should be employed for the variance error of the series.

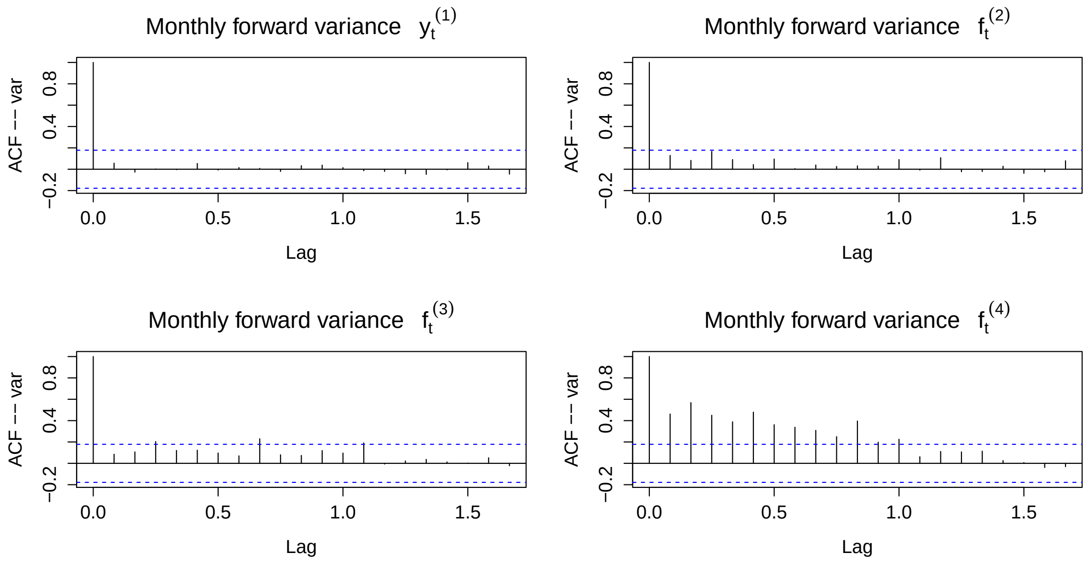


Table 3.3: Statistical features of forward variances. This table was created following Table 1 of Bakshi et al. (2011). Cross-correlations in Panel A suggest that forward variances across the term structure may have a low level of distinct information. Autocorrelations in Panel B give evidence that a low level ARMA model should be suitable to model the series. Based on the Dickey-Fuller test in Panel C the null hypothesis that the series are non-stationary may be rejected. In Panel D the estimated coefficients for the ARMA(1,1) - GARCH(1,1) model are provided. In Panel E we provide the standard error of the AR coefficients to further study the near unit root behavior of the series.

	$y_t^{(1)}$	$f_t^{(2)}$	$f_t^{(3)}$	$f_t^{(4)}$
Panel A: *Cross-correlations*				
$y_t^{(1)}$	1.000			
$f_t^{(2)}$	0.833	1.000		
$f_t^{(3)}$	0.794	0.815	1.000	
$f_t^{(4)}$	0.756	0.833	0.761	1.000
Panel B: *Autocorrelations*				
ACF(1)	0.641	0.686	0.664	0.758
ACF(2)	0.534	0.637	0.622	0.762
ACF(3)	0.527	0.633	0.703	0.757
ACF(4)	0.440	0.529	0.558	0.670
ACF(5)	0.439	0.522	0.568	0.668
ACF(6)	0.449	0.516	0.591	0.636
Panel C: *Unit root test – Phillips-Perron: null is I(1)*				
Dickey-Fuller	−4.17	−5.02	−5.49	−4.63
p-vlaue	0.01	0.01	0.01	0.01
Panel D: *ARMA(1,1)-GARCH(1,1) model parameter estimates*				
ARMA(1,1)				
AR	0.994	0.993	0.995	0.989
MA	−0.426	−0.453	−0.626	−0.609
GARCH(1,1)				
Const.	0.000	0.000	0.000	0.000
ARCH	0.003	0.000	0.377	0.521
GARCH	0.924	0.000	0.607	0.377
Panel E: *Near-unit-root behavior under ARMA(1,1)-GARCH(1,1)*				
AR st error	0.018	0.071	0.000	0.218

[a] The Phillip-Perron test was calculated using the R basic package. The ARMA(1,1)-GARCH(1,1) was fitted with the Ghalanos (2014) package. Standard errors of AR were acquired by the same package.

//www.federalreserve.gov/) under the G.17 and H.15 data codes. The non-farm payroll data were downloaded from the Bureau of Labor Statistics (http://www.bls.gov/data/). The models fitted are

$$g_{t+1}^{\text{payroll}} = a + b'\mathbf{f}_t + c \text{ yslope}_t + \epsilon_{t+1} \tag{3.14}$$

$$g_{t+1}^{\text{indus prod}} = a + b'\mathbf{f}_t + c \text{ yslope}_t + \epsilon_{t+1} \tag{3.15}$$

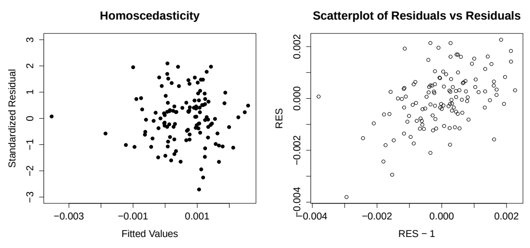
for the non-farm payroll and the industrial production respectively, where $\mathbf{f}_t$ is the vector containing the four forward variances and yslope_t is the Treasury yield slope.

Linear models using time series raise potential concerns regarding residuals behavior. If there are evidence for homoscedasticity and autocorrelation, statistical inference demands a different, than the usual, covariance matrix estimator. Following Bakshi et al. (2011) the Newey and West (NW) estimator is employed. In particular we use the Newey and West (1987) estimator with the lag parameter automatically chosen by Zeileis (2004), who implements the Newey and West (1994) method in R.

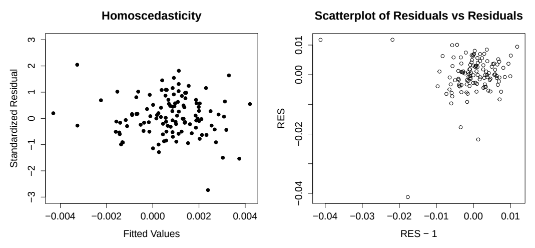
We check homskedasticity and autocorrelation of lag one with the usual plots depicted in Figure-3.5. The homoscedasticity assumption seems to be valid for both models. There is a clear trend for consecutive residuals for Model-3.14, but the residuals' independence assumption appears to hold for Model-3.15. We also provide the Durbin-Watson (DW) test for the null hypothesis that the errors of the model are serially uncorrelated against the alternative that they follow a first order autoregressive process (3.1). The test's p-value is in Table-3.4, where the regression results are collected. The test suggests similar conclusions, as the null hypothesis is rejected for the non-farm payroll model and not for the industrial production one. We conclude that the usual covariance matrix should be used for Model-3.15 and the NW one for Model-3.14. For consistency reasons, we provide statistics based on both estimates for the industrial production regression.

Based on the results of Table-3.4, we observe that the predictability of forward variances seems to be statistically insignificant for the industrial production. The p-values for the statistic testing the null hypothesis, that the coefficient is equal to zero, are large enough to accept H_0 for all coefficients. Moreover, the adjusted

Figure 3.5: Usual diagnostic plots to check residuals for homoscedasticity and autocorrelation (for lag = 1). Both linear fittings include the yield slope. In the first graph standard residuals are plotted against the fitted values in order to check homoscedasticity. In the second one, autocorrelation is tested, by plotting standard residuals e_t against e_{t-1}.



(a) Non-farm payroll



(b) Industrial production

coefficient determination $\bar{R}^2$ is very low, almost zero. On the other hand forward variances, as the treasury yield, seem to better explain non-farm payrolls.In particular, we find that when we do not include the treasury yield every forward variance except $f_t^{(2)}$ is significant in an alpha level of 5%. When the treasury yield is included $f_t^{(3)}$ is not significant anymore for the same alpha level.

Bakshi et al. (2011) have similar results but with important differences. They find that $f_t^{(2)}$ is not statistically important for predicting non-farm payroll. In contrast though, they find that $f_t^{(3)}$ is still statistically significant when the treasury yield is included. For the industrial production they find that $y_t^{(1)}$ and $f_t^{(4)}$ are statistically significant whether the treasury yield is included or not. Finally, their adjusted coefficients of determination are lower than ours in non-farm payroll and slightly higher for industrial production.

In our sample the coefficients that survive suggest that an increase on $y_t^{(1)}$ is translated to a consequently decrease in non-farm payroll. Alternatively, an increase on $f_t^{(3)}$ or $f_t^{(4)}$ is associated with a subsequent increase in non-farm payroll. Bakshi et al. (2011) have similar results for these coefficients.

Table 3.4: Can forward variances predict forthcoming real economic activity? Panel A shows that there is some predictability power for the non-farm payroll but the answer is negative for industrial production as Panel B shows. For each regression, the ordinary least squares (ols) estimators are provided. Their statistical significance is tested with the Newey and West (NW) estimator. Next to NW, the lag parameter, which has been automatically chosen by R, is provided. For the industrial production the ols p-values are given as well. We denote with $\bar{R}^2$ the adjusted coefficient of determination and with $\lfloor \text{DW} \rfloor$ the p-value of the Durbin-Watson test.

Dependent variable		Const.	$y_t^{(1)}$	$f_t^{(2)}$	$f_t^{(3)}$	$f_t^{(4)}$	yslope$_t$	$\bar{R}^2$ $\lfloor \text{DW} \rfloor$
Panel A: *Non-farm payroll*								
g_{t+1}^{payroll}	Coef.	0.001	−0.556	−0.093	0.157	0.279	−0.037	37.32%
	NW-8, p-val	0.049	0.000	0.143	0.077	0.000	0.006	$\lfloor 0.000 \rfloor$
g_{t+1}^{payroll}	Coef.	0.000	−0.627	−0.102	0.226	0.344		27.75%
	NW-8, p-val	0.862	0.000	0.093	0.009	0.000		$\lfloor 0.000 \rfloor$
Panel B: *Ind. Production*								
$g_{t+1}^{\text{indus prod}}$	Coef.	0.002	−0.819	0.527	−0.598	0.432	−0.033	0.63%
	NW-6, p-val	0.378	0.378	0.262	0.342	0.325	0.603	$\lfloor 0.112 \rfloor$
	usual p-val	0.429	0.202	0.269	0.192	0.322	0.524	
$g_{t+1}^{\text{indus prod}}$	Coef.	0.001	−0.882	0.518	−0.537	0.490		1.14%
	NW-7, p-val	0.570	0.291	0.259	0.332	0.171		$\lfloor 0.124 \rfloor$
	usual p-val	0.617	0.164	0.275	0.230	0.250		

[a] The Heteroskedasticity and autocorrelation consistent (HAC) covariance matrix estimator was calculated using the **sandwich** package by Zeileis (2004). The p-values and the DW test by the **lmtest** package of Zeileis and Hothorn (2002).

Conclusions

The main purpose of this thesis was for the student to engage contemporary research in the field of quantitative finance. In the first Chapter, which is the mathematical part, we rigorously presented part of the work conducted by Carr and Lee (2009b). Particularly, we showed that a generic exponential claim paying $e^{\lambda\langle X\rangle_T}$ may be expressed in terms of the underlying in the sense that there exists a payoff function G such that under the risk-neutral measure $\mathbb{Q}$,

$$\mathbb{E}^{\mathbb{Q}}\left[e^{\lambda\langle X\rangle_T}\right] = \mathbb{E}^{\mathbb{Q}}[G(S_T)]$$

Following Bakshi et al. (2011) we specialized this result for $\lambda = -1$ in order to define forward variances and spanned the RHS term in an investable portfolio of options. Note that this method builds on a non-parametric setting for an underlying process S_t with no jumps and pricing is first-order immune against the correlation of the underlying process and the instantaneous volatility process σ_t.

In the second chapter, which is the programming part and the core of the thesis, we created a term structure of forward variances contingent to the S&P 500 index, consisting of four maturities and ranging from 1998 to 2008. Filtering prices and extracting the risk neutral price curve across strike comprised the crucial challenges of the chapter. Prices were filtered by ensuring they satisfy monotonicity and convexity across strikes. We checked convexity by assuming that the price corresponding to the minimum strike was a fair one. Even though this was an arbitrary assumption and a different method should have been used, trading volumes corresponding to these prices were, in most cases, high enough to suggest their “fairness”. To extract the risk neutral price we experimented with various methods proposed in the literature. Production of poor results due to implementation errors, drove us to introduce a different method that behaved quite well, was based on fundamental non-arbitrage concepts, may be implemented easily and finally its quality stems from the theory of linear regression.

In the third chapter, which is the statistical aspect of the thesis, following Bakshi et al. (2011) we studied the statistical properties of forward variances and their predictability power for real economic activity. Our results are quite similar to theirs but with important differences. This is due to the lack of an academic or industry standard for cleaning option prices and extracting risk neutral price curves. Many methods are employed in the literature and so different results are expected.

Finally, we would have wanted to further enhance the method we used to calculate forward variances and experiment with it for different time frames and different underlings, study as Bakshi et al. (2011) their predictability power for asset returns, use them to address other question in finance and bolster the documented results with an appropriate theoretical framework.

References

Aït-Sahalia, Y., Lo, A. W., 1998. Nonparametric estimation of state-price densities implicit in financial asset prices. The Journal of Finance 53, 499–547.

Bakshi, G., Panayotov, G., Skoulakis, G., 2011. Improving the predictability of real economic activity and asset returns with forward variances inferred from option portfolios. Journal of Financial Economics 100, 475 – 495.

Bates, D. S., 1991. The crash of '87: Was it expected? the evidence from options markets. The Journal of Finance 46, pp. 1009–1044.

Black, F., Scholes, M., 1973. The pricing of options and corporate liabilities. The journal of political economy pp. 637–654.

Bliss, R. R., Panigirtzoglou, N., 2002. Testing the stability of implied probability density functions. Journal of Banking & Finance 26, 381 – 422.

Bliss, R. R., Panigirtzoglou, N., 2004. Option-implied risk aversion estimates. The journal of finance 59, 407–446.

Borchers, H. W., 2014. pracma: Practical Numerical Math Functions. R package version 1.6.4.

Breeden, D. T., Litzenberger, R. H., 1978. Prices of State-contingent Claims Implicit in Option Prices. The Journal of Business 51, 621–51.

Brown, G., Toft, K. B., 1999. Constructing binomial trees from multiple implied probability distributions. The Journal of Derivatives 7, 83–100.

Buchen, P. W., Kelly, M., 1996. The Maximum Entropy Distribution of an Asset Inferred from Option Prices. Journal of Financial and Quantitative Analysis 31, 143–159.

Campa, J. M., Chang, P., Reider, R. L., 1998. Implied exchange rate distributions: evidence from {OTC} option markets. Journal of International Money and Finance 17, 117 – 160.

Campbell, J. Y., Shiller, R. J., 1991. Yield spreads and interest rate movements: A bird's eye view. The Review of Economic Studies 58, 495–514.

Carr, P., Lee, R., 2009a. Put-call symmetry: Extensions and applications. Mathematical Finance 19, 523–560.

Carr, P., Lee, R., 2009b. Robust replication of volatility derivatives working paper. Unbublished working paper, New York University and University of Chicago .

Carr, P., Madan, D., 2001. Optimal positioning in derivative securities. Quantitative Finance 1, 19–37.

Cochrane, J. H., Piazzesi, M., 2005. Bond risk premia. American Economic Review 95, 138–160.

Cowpertwait, P. S., Metcalfe, A. V., 2009. Introductory time series with R. Springer.

Craven, P., Wahba, G., 1978. Smoothing noisy data with spline functions. Numerische Mathematik 31, 377–403.

Eddelbuettel, D., Nguyen, K., 2014. RQuantLib: R interface to the QuantLib library. R package version 0.3.12.

Fama, E. F., Bliss, R. R., 1987. The information in long-maturity forward rates. The American Economic Review pp. 680–692.

Forsythe, G. E., Moler, C. B., Malcolm, M. A., 1977. Computer methods for mathematical computations. Prentice-Hall.

Fouque, J.-P., 2010. Stochastic Volatility Models, John Wiley & Sons, Ltd.

Gatheral, J., 2006. The volatility surface: a practitioner's guide, vol. 357. John Wiley & Sons.

Ghalanos, A., 2014. rugarch: Univariate GARCH models. R package version 1.3-3.

Green, P. J., Silverman, B. W., 1993. Nonparametric regression and generalized linear models: a roughness penalty approach. CRC Press.

Green, R. C., Jarrow, R. A., 1987. Spanning and completeness in markets with contingent claims. Journal of Economic Theory 41, 202 – 210.

Halvorsen, K., 2012. ElemStatLearn: Data sets, functions and examples from the book: "The Elements of Statistical Learning, Data Mining, Inference, and Prediction" by Trevor Hastie, Robert Tibshirani and Jerome Friedman. R package version 2012.04-0.

Hamidieh, K., 2014. RND: Risk Neutral Density Extraction Package. R package version 1.1.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, second ed.

Hull, J., White, A., 1987. The pricing of options on assets with stochastic volatilities. The journal of finance 42, 281–300.

Hull, J. C., 2012. Options, Futures, and Other Derivatives. Pearson Education Limited, 8th ed.

Jackwerth, J. C., 1999. Option-implied risk-neutral distributions and implied binomial trees: A literature review. The Journal of Derivatives 7, 66–82.

Jackwerth, J. C., 2000. Recovering risk aversion from option prices and realized returns. Review of Financial Studies 13, 433–451.

Jackwerth, J. C., 2004. Option-implied risk-neutral distributions and risk aversion. CFA Institute Research Foundation of AIMR Publications 13, 1–86.

Jiang, G. J., Tian, Y. S., 2005. The Model-Free Implied Volatility and Its Information Content. Review of Financial Studies 18, 1305–1342.

Kang, B. J., Kim, T. S., 2006. Option-implied risk preferences: An extension to wider classes of utility functions. Journal of Financial Markets 9, 180 – 198.

Kang, B. J., Kim, T. S., Lee, H. S., 2014. Option-implied preference with model uncertainty. Journal of Futures Markets 34, 498–515.

Kostakis, A., Panigirtzoglou, N., Skiadopoulos, G., 2011. Market timing with option-implied distributions: A forward-looking approach. Management Science 57, 1231–1249.

Malz, A. M., 1997. Estimating the probability distribution of the future exchange rate from option prices. The Journal of Derivatives 5, 18–36.

Merton, R. C., 1973. Theory of Rational Option Pricing. Bell Journal of Economics 4, 141–183.

Nachman, D. C., 1988. Spanning and completeness with options. Review of Financial Studies pp. 311–328.

Neri, Schneider, 2013. A Family of Maximum Entropy Densities Matching Call Option Prices. Applied Mathematical Finance 20, 548–577.

Neumann, M., Skiadopoulos, G., 2013. Predictable dynamics in higher-order risk-neutral moments: Evidence from the s&p 500 options. Journal of Financial and Quantitative Analysis 48, 947–977.

Newey, W., West, K. D., 1987. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. Econometrica 55, 703–08.

Newey, W. K., West, K. D., 1994. Automatic lag selection in covariance matrix estimation. The Review of Economic Studies 61, 631–653.

Nychka, D., Furrer, R., Sain, S., 2014. fields: Tools for spatial data. R package version 7.1.

Øksendal, B., 2003. Stochastic Differential Equations - An Introduction with Applications. Universitext, Springer, 6th ed.

Panigirtzoglou, N., Skiadopoulos, G., 2004. A new approach to modeling the dynamics of implied distributions: Theory and evidence from the S&P 500 options. Journal of Banking & Finance 28, 1499–1520.

Perron, P., 1988. Trends and random walks in macroeconomic time series: Further evidence from a new approach. Journal of economic dynamics and control 12, 297–332.

Phillips, P. C., Perron, P., 1988. Testing for a unit root in time series regression. Biometrika 75, 335–346.

R (Core Team), 2013. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Ramsey, J., Ripley, B., 2013. pspline: Penalized Smoothing Splines. R package version 1.0-16.

Rosenberg, J. V., Engle, R. F., 2002. Empirical pricing kernels. Journal of Financial Economics 64, 341–372.

Rutkowski, M., 2010. Complete Markets, John Wiley & Sons, Ltd.

Said, S. E., Dickey, D. A., 1984. Testing for unit roots in autoregressive-moving average models of unknown order. Biometrika 71, 599–607.

Shimko, C. D., 1993. Bounds of probability. Risk Magazine 6, 33–37.

Shreve, S., 2008. Stochastic Calculus for Finance II: Continuous-Time Models. Springer, New York; London, first ed.

Zeileis, A., 2004. Econometric computing with hc and hac covariance matrix estimators. Journal of Statistical Software 11, 1–17.

Zeileis, A., Hothorn, T., 2002. Diagnostic checking in regression relationships. R News 2, 7–10.

Appendix A

Scripts

Script A.1: Obtaining LIBORs

```r
require(RQuantLib)
# LIBOR data were obtained by Federal Reserve Bank of St. Louis
# [1] and EconStats[2]. Download data from:
# [1] http://research.stlouisfed.org/fred2/categories/33003/
   downloaddata
# [2] http://www.econstats.com/r/rlib__d1.htm
# Assume that the files from [1] (LIBOR_csv_2.zip/data) and from
# [2] rlib__d1.csv renamed to rlib.csv are in the working
# directory. Morover note that we deleted unnecesary rows from
# rilb and kept only the values and their headers.

# dates contains the desired dates for which we want to
# extract data
dates = unname(
  getEndOfMonth(
    calendar = "UnitedStates/NYSE",
    dates = seq.Date(from = as.Date('1998/9/1'),
                     to = as.Date('2008/9/1'), by = "month")))

#################################################################
# Section 1: Get LIBORs from [2]
#################################################################
dat = read.csv(file = "rlib.csv")
# create a vector for filenames for the 1-4 months LIBOR
names = character(16)
names[1] = "Date"
names[2] = "Day"
names[3] = "1WK"
names[4] = "2WK"
for(i in 5:16){
  names[i] = paste(i - 4, "MN", sep = "")
}

names(dat) = names

# Subset to dates
dat$Date = as.Date(dat$Date, format = "%m/%d/%Y")
df = subset(dat, dat$Date %in% dates)
df = df[,-2]
rm(dates, i, names)

#################################################################
# Section 2: Get missing values from [1] for dates "2007-03-30"
# and "2006-01-31".
#################################################################
filename = character(14)
filename[1] = "USD1WKD156N.csv"
filename[2] = "USD2WKD156N.csv"
for(i in 1:12){
  if( i >= 10 )
    filename[i+2] = paste("USD", i, "MD156N.csv", sep = "")
  else
    filename[i+2] = paste("USD", i, "MTD156N.csv", sep = "")
}

for(date in c("2007-03-30", "2006-01-31")){
  # Read files for specific date and store LIBORs to value
  value = numeric(14)
  dateObs = as.Date(date)
  for(i in 1:14){
    if(i != 2){ # skip 2WK libor (default value is 0)
      # read file
      dat2 = read.csv(filename[i], stringsAsFactors=FALSE)
      # convert dates to Dates objects
      dat2$DATE = as.Date(dat2$DATE)
      value[i] =
        as.numeric(subset(dat2, dat2$DATE == dateObs)$VALUE)
    }
  }
  print(value)

  # Replace to df
  df[which(df$Date == dateObs), 2:15] = value
}
rm(dat2, dateObs, filename, i, value, date)

#################################################################
# Section 3: Get values from [2] for "1999-12-30" and use them
# for "1999-12-31"
#################################################################
```

```
vals = subset(dat, Date == as.Date("1999-12-30"))[3:16]
unname(vals)

df[which(df$Date == as.Date("1999-12-31")), 2:15] = vals
rm(dat, vals)

#################################################################
# Section 4: Interpolate for missing values for 2-Libor prior
# to 2000
#################################################################
rownames(df) = 1:121 # re-enumerate rows from 1-121

# Get 1-week LIBOR and 1-month LIBOR for missing values of
# 2-week LIBOR
r1W = df[94:121, 2]
r1M = df[94:121, 4]

# Interpolate
r2W = r1W + (r1M - r1W)/(30 - 7)*(14-7)

# Replace
df[94:121, 3] = r2W

rm(r1W, r1M, r2W)

#################################################################
# Section 5: Export Data
#################################################################
write.csv(df, file = "LIBORs.csv", row.names = F)
```

Script A.2: Extracting Data

```
require(RQuantLib)
# This script uses the following data files
# Sep1998Sep2008.csv, SnP500Prices.csv, LIBORs.csv
# The script runs in about 100 seconds.

#################################################################
# Section 1: Load data downloaded from OptionMetrics (file:
#         Sep1998Sep2008.csv) and filter for dates.
#
#       Observations Prior: 1,568,099
#       Observations After: 74,592
#       Observations Difference: 1,493,507
#################################################################
data = read.csv(file="Sep1998Sep2008.csv")

# Convert dates to R Dates objects
data$date = as.Date(as.character(data$date), format='%d%b%Y')
```

```

# Get last trading day for each month
dates = unique(
  data$date[isEndOfMonth(calendar="UnitedStates/NYSE",
                              data$date)])

# Subset data to the last trading day for each month
data = data[data$date %in% dates, ] #74592

# Get dates
dates = unique(data$date)

##################################################################
# Section 2: Apply initial filters and discard useless
#       information. Finally sort data according to
#       date, type, expiration and strike.
#
#       Observations Prior: 74,592
#       Observations After: 22,707
#       Observations Difference: 51,885
##################################################################

# Apply Filter F1 - obs thrown = 4,448
data = subset(data, best_bid != 0)

# Apply Filter F2 - obs thrown = 13,052
data = subset(data, open_interest != 0)

# Apply Filter F3 - obs thrown = 33,342
data = subset(data, volume != 0)

# Apply Filter F4 - obs thrown = 1,043
# Note that obs with implied volatility > 1 were 8.
# The same result if we test for volatility >= 1.
data = subset(data, impl_volatility >= 1 |
                !is.na(impl_volatility))

# Convert exdate to R Dates objects
data$exdate = as.Date(as.character(data$exdate),
                      format = '%d%b%Y')

data = data.frame(data$date, data$cp_flag,
                as.integer(data$exdate - data$date),
                data$strike_price/1000,
                0.5*(data$best_bid + data$best_offer),
                data$volume, data$impl_volatility)

names(data) = c("date", "type", "expiration", "strike", "price",
              "volume", "implVol")
```

```

# Short data
data = data[order(data$date, data$type, data$expiration,
                  data$strike), ]

##################################################################
# Section 2: Load underlying, throw ITM as defined in the
#            text and discard expiration less than a week.
#
#      Observations Prior: 22,707
#      Observations After: 19,593
#      Observations Difference: 3,114
##################################################################

# Load the S&P prices -- price of the underlying
snp500 = read.csv(file="SnP500Prices.csv")
colnames(snp500) = c("Date", "Price")

# Create appropriate s&p vector. For example, if for the 1st date
# we have 100 observations, then we repeat the 1st underlying
# price 100 times. [Remember that data are sorted for date]

# Get number of observations per date
numdates = as.vector(table(data$date))
# Create vector
snp500rep = rep(snp500$Price, numdates)

# Add underlying to data.
data = data.frame(data, snp500rep)
colnames(data)[8] = "underlying"

# Remove useless variables
rm(snp500, snp500rep, numdates)

# Apply Filter F5 - obs thrown = 3,038
data = data[(data$type == 'C' & data$strike >= 0.97*data$
   underlying) |
         (data$type == 'P' & data$strike <= 1.03*data$underlying
            ), ]

# Throw expirations prior to a week (7 days)
data = subset(data, expiration > 7)
# 76 obs were thrown. The new minimum expiration is
min(data$expiration) # => 16 days

# # Here Back-Up 1 was created
# write.csv(data, "BackUp.csv", row.names = FALSE)
##################################################################
# Section 3: Load LIBORs, create discount factors and check
#       arbitrage bounds
#
#       Observations Prior: 19,593
#       Observations After: 19,593
#       Observations Difference: 0
#################################################################

#################################################################
# Section 3.1: Create discount factors and add them to data
#################################################################
# Load LIBORs
libors = read.csv(file = "LIBORs.csv", header = T)
libors$Date = as.Date(libors$Date) # convert dates

# Convert to numbers instead of percentages
libors[2:ncol(libors)] = apply(libors[2:ncol(libors)], 2,
                               function(x) x/100)

# Throw 1-week LIBORs and
# sort them for quicker search
libors = libors[-2]
libors = libors[order(libors$Date), ]

# Variable rates will contain the appropriate (interpolated)
# rate for each observation.
rates = numeric(nrow(data))

# Repeat for each observation
for(i in 1:length(rates)){
  # Get the date
  date = data[i, ]$date

  # Variable idx is the position of date in vector libors.
  idx = which(libors$Date == date)

  # Find the last month prior to expiration.
  # Month 0 corresponds to the 2-Week LIBOR
  month = floor(data[i, ]$expiration/30)

  # The first column of libors is the date.
  # The 2-Week LIBOR is on column 2, the 1-Month LIBOR to
  # column 3, the 2-Month LIBOR to column 3 etc.
  if(month == 0){
    xs = c(15, 30)
    ys = c(libors[idx, month + 2], libors[idx, month + 3])
  }else if (month < 12){
    xs = c(30*month, 30*(month + 1))
    ys = c(libors[idx, month + 2], libors[idx, month + 3])
  }else{
```

```r
    xs = c(30*11:12)
    ys = c(libors[idx, 13:14])
  }
  rates[i] = splinefun(xs, ys)(data[i, ]$expiration)
}

# For every rate calculate the interest due for 1$
interest = rates*(data$expiration/360)

# Calculate the discount factor for 1$
discount = 1/(1+interest)

# Add discount factor to data
data = data.frame(data, discount)

# Remove useless variables
rm(libors, date, discount, i, idx, interest, month, rates, xs, ys
   )

# # Here Back-Up 2 was created
# write.csv(data, "BackUp2.csv", row.names = FALSE)
#################################################################
# Section 3.2: check arbitrage bounds
#################################################################
# calls-down => no arbitrage found
which(data$type=='C' &
        data$underlying - data$strike*data$discount > data$price)
# calls-up => no arbitrage found
which(data$type=='C' & data$price > data$underlying)
# puts-up => no arbitrage found
which(data$type=='P' &
        data$strike*data$discount - data$underlying > data$price)
# puts-down => no arbitrage found
which(data$type=='P' & data$price > data$strike*data$discount)

#################################################################
# Section 4: Check strict monotonicity
#
#       Observations Prior: 19,593
#       Observations After: 19,347
#       Observations Difference: 246
#################################################################

# We check monotonicity by checking two required and sufficient
# conditions. We check that a vector x is strictly  monotonically
# increasing (decreasing) by checking whether it is not
# decreasing (increasing) [1] and by whether it has duplicates
# [2]. We check them by the following commands:
#
```

```
# [1a] all(x == cummax(x)), for increasing
# [1b] all(x == cummin(x)), for decreasing
# [2] anyDuplicated(x) == 0

# Round prices to for I/O approximations
data$price = round(data$price, 7)


#################################################################
# Section 4.1.1b: For Calls (should be strictly decreasing)
#                 Found 39 arbitrages -- after: 19,554 obs
#################################################################

calls = subset(data, type == 'C')
# Step 1: Split across dates and expirations
# Step 2: For each element of the list check [1b]
# Step 3: Return those dissatisfy [1b] to c
c =
  which(
    lapply(
      split(calls, list(calls$date, calls$expiration)),
      function(x) (all(x$price == cummin(x$price)))
    ) == F
  )

# Get dates and strikes from c
cn = matrix(unlist(strsplit(names(c), "[.]")), ncol = 2,
            byrow = T)

# Remove prices that defy [1b]
# For each date and expiration in cn do
# Step 1: Subset data to date and expiration
# Step 2: Find values that spoil [1b]
# Step 3: Get the row names of those.
# Step 4: Remove them from data
for(i in 1:nrow(cn)){
  temp = subset(data, date == cn[i,1] & type == 'C' &
                  expiration == cn[i,2])
  rs = which((temp$price == cummin(temp$price)) == F)
  rs.names = rownames(temp)[rs]
  data = data[!(rownames(data) %in% rs.names), ]
}

# Remove useless variables
rm(calls, cn, temp, c, i, rs, rs.names)

#################################################################
# Section 4.1.2: Duplicates
#                Found 43 arbitrages -- after: 19,511 obs
```

```r
################################################################
calls = subset(data, type == 'C')
# Step 1: Split accross dates and expirations
# Step 2: For each element of the list check [2]
# Step 3: Return those desatisfy [2] to c
c =
  which(
    lapply(
      split(calls, list(calls$date, calls$expiration)),
      function(x) (anyDuplicated(x$price) == 0)
    ) == F
  )

# Get dates and strikes from c
cn = matrix(unlist(strsplit(names(c), "[.]")), ncol = 2,
            byrow = T)

# Remove prices that defy [2]
# For each date and expiration in cn do
# Step 1: Subset data to date and expiration
# Step 3: Get the row names of those spoil [2].
#         This implementation ensures that the last duplicate
#         is kept.
# Step 4: Remove them from data
for(i in 1:nrow(cn)){
  temp = subset(data, date == cn[i,1] & type == 'C' &
                  expiration == cn[i,2])
  rs.names = rownames(temp)[which(duplicated(temp$price) == T) -
      1]
   data = data[!(rownames(data) %in% rs.names), ]
}

# Remove useless variables
rm(calls, cn, temp, c, i, rs.names)

################################################################
# Section 4.2.1a: For Puts (should be strictly increasing)
#                 Found 72 arbitrages -- after 19,439 obs
################################################################
puts = subset(data, type == 'P')
# Step 1: Split accross dates and expirations
# Step 2: For each element of the list check [1b]
# Step 3: Return those desatisfy [1a] to c
c =
  which(
    lapply(
      split(puts, list(puts$date, puts$expiration)),
      function(x) (all(x$price == cummax(x$price)))
    ) == F
  )

# Get dates and strikes from c
cn = matrix(unlist(strsplit(names(c), "[.]")), ncol = 2,
            byrow = T)

# Remove prices that defy [1b]
# For each date and expiration in cn do
# Step 1: Subset data to date and expiration
# Step 2: Find values that spoil [1b]
# Step 3: Get the row names of those.
# Step 4: Remove them from data
for(i in 1:nrow(cn)){
  temp = subset(data, date == cn[i,1] & type == 'P' &
                  expiration == cn[i,2])
  rs = which((temp$price == cummax(temp$price)) == F)
  rs.names = rownames(temp)[rs]
  data = data[!(rownames(data) %in% rs.names), ]
}

# Remove useless variables
rm(puts, cn, temp, c, i, rs, rs.names)

#################################################################
# Section 4.2.2: Duplicates
#                 Found 92 arbitrages -- after 19,347 obs
#################################################################
puts = subset(data, type == 'P')
# Step 1: Split accross dates and strikes
# Step 2: For each element of the list check [2]
# Step 3: Return those desatisfy [2] to cdm
c =
  which(
    lapply(
      split(puts, list(puts$date, puts$expiration)),
      function(x) (anyDuplicated(x$price) == 0)
    ) == F
  )

# Get dates and strikes from c
cn = matrix(unlist(strsplit(names(c), "[.]")), ncol = 2,
            byrow = T)

# Remove prices that defy [2]
# For each date and expiration in cn do
# Step 1: Subset data to date and expiration
# Step 3: Get the row names of those spoil [2].
#         This implimentation ensures that the first duplicate
#         is kept.
```

```r
# Step 4: Remove them from data
for(i in 1:nrow(cn)){
  temp = subset(data, date == cn[i,1] & type == 'P' &
                  expiration == cn[i,2])
  rs.names = rownames(temp)[duplicated(temp$price)]
  data = data[!(rownames(data) %in% rs.names), ]
}

# Remove useless variables
rm(puts, cn, temp, c, i, rs.names)

################################################################
# Section 5: Check strict convexity
#
#      Observations Prior: 19,347
#      Observations After: 18,210
#      Observations Difference: 1,137
################################################################

################################################################
# Section 5.1: Check strict Convexity - Calls
#              Found 375 arbitrages -- after 18,972 obs
################################################################
# In vector toBeRemoved we will stor the names of the rows
# tha we will remove from data
toBeRemoved = character();

for(i in 1:length(dates)){
  dt = dates[i]
  for(x in unique(subset(data, date == dt)$expiration)){
    # Subset data for date dt and expiration x and store it to
    # temp
    temp = subset(data, type == 'C' & date == dt &
                    expiration == x)

    # Get the price and strike vectors of temp
    p = temp$price
    k = temp$strike

    # Store Violations to logical vector violations.
    # First and last connot be violations.
    # Defualt value of logical() is false.
    violations = logical(length(p))

    # At least two prices are needed to have convexity. Check it.
    if(length(p) > 2){
      # Check the condition for each triplet and store the
      # result in violations.
      for(j in 2:(length(p)-1)){
        threshold =
          (k[j+1] - k[j])/(k[j+1]-k[j-1])*(p[j-1]-p[j+1])+p[j+1]
        violations[j] = (p[j] > threshold)
      }
    }
    # If violations have at least one TRUE add rowname of temp
    # to vector to toBeRemoved
    if(any(violations)){
      toBeRemoved = c(toBeRemoved, rownames(temp[violations, ]))
    }
  }
}

# Remove from data
length(toBeRemoved) #=> # of arbitrages 377
data = data[!(rownames(data) %in% toBeRemoved), ]

# Remove useless variables
rm(dt, j, k, p, threshold, toBeRemoved, violations, x, i, temp)

#################################################################
# Section 5.3: Check strict Convexity - Puts
#              Found 762 arbitrages -- after 18,210 obs
#################################################################
# Same as 4.1 -- only change type and condition
toBeRemoved = character();

for(i in 1:length(dates)){
  dt = dates[i]
  for(x in unique(subset(data, date == dt)$expiration)){
    # Change type to 'P'
    temp = subset(data, type == 'P' & date == dt &
                  expiration == x)
    p = temp$price
    k = temp$strike
    violations = logical(length(p))
    if(length(p) > 2){
      for(j in 2:(length(p)-1)){
        # Changed the condition
        threshold =
          (k[j] - k[j-1])/(k[j+1]-k[j-1])*(p[j+1]-p[j-1])+p[j-1]
        violations[j] = (p[j] > threshold)
      }
    }
    if(any(violations)){
      toBeRemoved = c(toBeRemoved, rownames(temp[violations, ]))
    }
  }
}
```

```

# Remove from data
length(toBeRemoved) #=> # of arbitrages 802
data = data[!(rownames(data) %in% toBeRemoved), ]
rm(dt, j, k, p, threshold, toBeRemoved, violations, x,
   i, temp, dates)

#################################################################
# Section 6: Apply final filter and export data.
#
#       Observations Prior: 18,210
#       Observations After: 17,421
#       Observations Difference: 789
#################################################################
data = data[data$price >= 3/8, ]

write.csv(data, file = "cleanData.csv", row.names = F)
```

Script A.3: Functions used to calculate the Integrals.

```
require(RQuantLib); require(rootSolve);
#################################################################
# Functions ebs and iv are used for data frames which may
# contain observations for a given day and type of options.
# Functions ebs.vec and iv.vec are used for vectors.
#
# These functions are used for the one-to-one mapping between
# prices and implied volatilities as explained
# in the text. To caluclate implied volatilities, we employ the
# implied volatility function of RQuantLib.

ebs <- function(data){
  t = data$expiration/360;
  type = data$type[1]

  d1 = 1/(data$implVol*sqrt(t))*(log(data$underlying/data$strike)
      +
                                   data$implVol^2/2*t)
  d2 = d1 - data$implVol*sqrt(t)

  if(type == 'C')
    prices = pnorm(d1)*data$underlying - pnorm(d2)*data$strike
  if(type == 'P')
    prices = pnorm(-d2)*data$strike - pnorm(-d1)*data$underlying

  return(round(prices, 4))
}

ebs.vec <- function(ivs, strikes, s0, mat, type){
```

```r
  t = mat/360;

  d1 = 1/(ivs*sqrt(t))*(log(s0/strikes) + ivs^2/2*t)
  d2 = d1 - ivs*sqrt(t)

  if(type == 'C')
    prices = pnorm(d1)*s0 - pnorm(d2)*strikes
  if(type == 'P')
    prices = pnorm(-d2)*strikes - pnorm(-d1)*s0

  return(round(prices, 4))
}

iv <- function(data){
  if(nrow(data) == 0)
    print("error iv function")
  t = numeric(nrow(data))
  for(i in 1:nrow(data)){
    x = data[i, ]

    if(x$type == 'C')
      type = "call"
    if(x$type == 'P')
      type = "put"

    # We use the implied volatility provided by OptionMetrics
    # for initial guess.
    t[i] = EuropeanOptionImpliedVolatility(
      type, x$price, x$underlying,
      x$strike, 0, 0, x$expiration/360,
      x$implVol
      )$impliedVol
  }
  return(t)
}

iv.vec <- function(prices, strikes, s0, mat, type, guesses){
  if(type == 'C'){
    type = "call"
  }else type = "put"

  l = split(cbind(prices, strikes, guesses), 1:length(prices))
  ivs = as.vector(
    sapply(l, function(x){
      iv = EuropeanOptionImpliedVolatility(type, x[1], s0, x[2],
         0, 0, mat/360, x[3])$impliedVol
      return(iv)
      }))
  return(ivs)
```

```r
}

callDeltas <- function(atmIV, S, strikes, expiration){
  # This function calculates the call deltas for certain strikes,
  # or for certain maturities.

  t = expiration/360
  d1 = 1/(atmIV*sqrt(t))*(log(S/strikes) + atmIV^2*t/2)
  return(pnorm(d1))
}

inverseDeltas <- function(atmIV, S, deltas, expiration){
  # This function is the inverse of callDeltas for strikes

  t = expiration/360
  a = atmIV^2*t/2 - qnorm(deltas)*atmIV*sqrt(t)
  return(round(S*exp(a), 4))
}

vega <- function(ivs, strikes, s0, expiration){
  # This function is used to calculate option vegas.
  t = expiration/360

  d1 = 1/(ivs*sqrt(t))*(log(s0/strikes) + ivs^2/2*t)

  vegas = s0*dnorm(d1)*sqrt(t)
  return(vegas)
}

# # Example 1: Implied Volatilites - Prices
# ex = subset(read.csv("cleanData.csv"), date == "1999-06-30" &
    type == 'P' & expiration == 17)
# ebs(ex) == ebs.vec(ex$implVol, ex$strike, ex$underlying[1], 17,
     'P')
# iv(ex) == iv.vec(ex$price, ex$strike, ex$underlying[1], 17, 'P
    ', ex$implVol)
# rm(ex)

################################################################
# Omega function is used by integration functions.

omega <- function(strikes, s0){
  od = sqrt(s0)*(strikes)^(3/2)
  ou = 8/sqrt(14)*cos(
    atan(1/sqrt(7)) + sqrt(7)/2*log(strikes/s0)
  )
  o = -ou/od
  return(o)
}
```

Script A.4: Creating new Maturities.

```
#################################################################
# Function new Maturity takes a data frame of observations for a
# given day and returns a new data frame with new expirations for
# that day. It uses functionsDF.R for the one-to-one mappings
# between prices and volatilities. Interpolation is done by a
# cubic spline. Prices are checked for arbitrages and minimum
# tick for 0.05 with the functions in checkPrices.R. New discount
# factors are calculated for the new expiration.

source("functionsDF.R") # Load functions
source("checkPrices.R") # Load functions

plotMaturites <- function(x, mat){
  par(mfrow = c(1,2), oma = c(0,0,2,0))

  # Get type
  tp = x$type[1]
  if(tp == 'C'){
    tp = "Calls"
  }else tp = "Puts"

  # Create expirations sequence
  ms = seq(min(x$expiration, floor(0.8*mat)),
           max(x$expiration, ceiling(1.2*mat)), 1)

  # plot spline fitting
  if(nrow(x) < 4){
    l = splinefun(x$expiration, x$implVol, method = "natural");
    plot(x$expiration, x$implVol, xlab = "Expiration",
         ylab = "Implied Volatility",
         main = "Interpolation")
    lines(ms, l(ms))
  }else{
    l = smooth.spline(x$expiration, x$implVol);
    newIV = predict(l, mat)$y
    plot(x$expiration, x$implVol, xlab = "Expiration",
         ylab = "Implied Volatility",
         main = "Interpolation")
    lines(l)
  }

  # Plot prices fitting
  if(nrow(x) < 4){ newIVs = l(ms)
  }else newIVs = predict(l, ms)$y

```

```r
  newPrices = ebs.vec(newIVs, x$strike[1], x$underlying[1], ms, x
     $type[1])
  plot(ms, newPrices, xlab = "Expiration",
       ylab = "Price", type = 'l',
       main = paste(tp, " for ", as.character(x$date[1]), sep = "
          "))
  points(x$expiration, x$price, pch = 15)
  points(mat, newPrices[which(ms == mat)], pch = 8)
  title(paste("Strike = ", x$strike[1]), outer = T)
  # Prompt for next plot
  readline(prompt = "Pause. Press <Enter> to continue to next
     date...")
}

newMaturity.core <- function(df, mat, plot){
  # Splite per strike
  l = split(df, df$strike)
  # Throw strikes with less than 2 maturities
  l[sapply(l, nrow) < 2] = NULL

  # Throw strikes that their range do not
  # contain the desired maturity
  l[sapply(l, function(x){
    mat < min(x$expiration) | mat > max(x$expiration)
  })] = NULL

  l = lapply(l, function(x){
    # Replace OptionMetrics' impled volatilites
    x$implVol = iv(x)
    if(nrow(x) < 4){
      newIV = splinefun(x$expiration, x$implVol, method = "
         natural")(mat)
    }else{
      ss = smooth.spline(x$expiration, x$implVol);
      newIV = predict(ss, mat)$y
    }
    # Plot?
    if(plot == T) plotMaturites(x, mat)
    # Change information of first row and export
    x[1, ]$expiration = mat
    x[1, ]$implVol = newIV
    x[1, ]$price = ebs(x[1, ])
    x[1, ]$volume = NA
    return(x[1, ])
  })


  newDf = unsplit(l, as.numeric(names(l)))
  newDf = checkPrices(newDf)
  return(newDf)
}

newMaturity <- function(df, mat, plot = F){
  df.c = subset(df, type == 'C')
  if(nrow(df.c) > 0)
    df.c = newMaturity.core(df.c, mat, plot)
  df.p = subset(df, type == 'P')
  if(nrow(df.p) > 0)
   df.p = newMaturity.core(df.p, mat, plot)

  # combine results
  newDf = rbind(df.c, df.p)

  # Add corrent discount factor
  date = as.character(df$date[1])
  libors = subset(read.csv("LIBORS.csv"), Date == date)
  libor = splinefun(c(30,60), libors[(floor(mat/30) + 3):(ceiling
     (mat/30) + 3)])(mat)
  rate = (libor/100)*(mat/360)
  discount = 1/(1+rate)

  newDf$discount = discount
  return(newDf)
}

# # Example
# ex = subset(read.csv("cleanData.csv"), date == "1999-06-30")
# newMaturity(ex, 49, plot = T)
#
# # See all the data together
# d = rbind(ex, newMaturity(ex, 49))
# d[order(d$type, d$strike, d$expiration),]
# rm(ex, d)
```

Script A.5: Checking new prices for arbitrages and for being above the minimum tick.

```r
##################################################################
# Function checkPrices uses checkPrices.calls and checkPrices.put
# to check prices in a data frame containing observations for
# a given day, type and expiration. Prices are filtered as in
# extract data. The minimum allowd tick for an option price is
# 0.05.


checkPrices <- function(df){
  type = df$type[1]

  if(type == 'C'){
    newdf = checkPrices.calls(df)
  }else newdf = checkPrices.puts(df)
  return(newdf)
}

checkPrices.calls <- function(df){
  # Merton's Bounds
  df = subset(df,
              df$underlying - df$strike*df$discount <= df$price &
                df$price <= df$underlying)
  # Monotonicity
  df = df[which(df$price == cummin(df$price)), ]
  tbrm = which(duplicated(df$price) == T) - 1
  if(length(tbrm) > 0) df = df[-tbrm, ]
  # Convexity
  p = df$price
  k = df$strike

  violations = logical(length(p))
  if(length(p) > 2){
    for(j in 2:(length(p)-1)){
      threshold =
        (k[j+1] - k[j])/(k[j+1]-k[j-1])*(p[j-1]-p[j+1])+p[j+1]
      violations[j] = (p[j] > threshold)
    }
  }

  # Remove violations
  df = df[!violations, ]

  # Minimum Tick
  df = df[df$price > 0.05, ]
  # return
  return(df)
}

checkPrices.puts <- function(df){
  df = subset(df,
              df$strike*df$discount - df$underlying <= df$price &
              df$price <= df$strike*df$discount)
  # Monotonicity
  df = df[which(df$price == cummax(df$price)), ]
  tbrm = which(duplicated(df$price) == T)
  if(length(tbrm) > 0) df = df[-tbrm, ]

  # Convexity
  p = df$price
  k = df$strike

  violations = logical(length(p))
  if(length(p) > 2){
    for(j in 2:(length(p)-1)){
      threshold =
        (k[j] - k[j-1])/(k[j+1]-k[j-1])*(p[j+1]-p[j-1])+p[j-1]
      violations[j] = (p[j] > threshold)
    }
  }

  # Remove violations
  df = df[!violations, ]

  # Minimum Tick
  df = df[df$price > 0.05, ]
  # return
  return(df)
}
```

Script A.6: Calculating the integrals in the implied volatility delta space.

```
source('functionsDF.R')
require(fields)
require(pracma)
#################################################################
# Function calcIntegral takes a data frame containing obs for
# a single observation and calculates the integral as it is
# explained in the text. It uses plot.integral for plotting the
# fitting and the area to be integrated. Function omega is used
# to calculate omega.

plot.integral <- function(oldDeltas, oldIVs, newDeltas, newIVs,
                          initStrikes, initPrices, newStrikes,
                          newPrices, s0, tp, date){
  par(mfrow = c(1,2))

  # Get type
  if(tp == 'C'){
    tp = "Calls"
  }else tp = "Puts"

  plot(oldDeltas, oldIVs, ylab = "Implied Volatility", xlab = "
     delta",
       xlim = range(c(oldDeltas, newDeltas)), ylim = range(c(
          oldIVs, newIVs)),
       main = paste("Interpolation - ", date, sep = ""))
  lines(x = newDeltas, y = newIVs)
  # Plot linear regression line
  abline(lm(oldIVs ~ oldDeltas), lty = 2)
```

```r
  legend(x = "topleft", legend = c("smooth spline", "ols line"),
      lty = c(1, 2), bty = "n", cex = 0.9)

  plot(newStrikes, newPrices, type = 'l',
       main = paste(tp, " for ", as.character(date), sep = ""),
       xlim = range(c(initStrikes, newStrikes)),
       ylim = range(c(initPrices, newPrices)),
       xlab = "strike", ylab = "price")
  points(initStrikes, initPrices, pch = 20)
  abline(v = s0, lty = 3)
}

calcIntegral <- function(data, plot = F){
  dates = unique(data$date)
  hvals = numeric(length(dates))

  # For-loop to compute integral for each date
  for(i in 1:length(dates)){
    # For each date subset to date
    h = subset(data, date == dates[i])

    s0 = h$underlying[1]
    mat = h$expiration[1]

    # calculate first term of integral (the bond)
    hvals[i] = h$discount[1]

    #
       ###############################################################

    # Section 2: For Calls
    # Subset to calls
    h.c = subset(h, type == 'C')

    # Create deltas
    atmIV = h.c$implVol[which.min(abs(h.c$strike - s0))]

    # Throw ITM and get implied volatilites and oldDeltas
    tempc = subset(h.c, strike >= s0)
    oldIVs = iv(tempc)
    oldDeltas = callDeltas(atmIV, s0, tempc$strike, mat)

    # Discard Deltas less than <0.01
    idx = which(oldDeltas < 0.01)
    if(length(idx) != 0){
      oldIVs = oldIVs[-idx]
      oldDeltas = oldDeltas[-idx]
    }

    # Interpolate implied volatility wrt deltas
    # using a smoothing spline with smoothing 0.99
    l = sreg(y = oldIVs, x = oldDeltas, lambda = 0.99)

    # Find delta wich gives the ATM strike and create new Deltas
    deltaATM = uniroot(
      function(x) inverseDeltas(atmIV, s0, x, mat) - s0, c(0,1)
    )$root
    newDeltas = seq(0.01, deltaATM, length.out = 500)

    # Predict new implied volatilites
    newIVs = predict(l, newDeltas)

    # Extrapolate linearly outside known deltas
    # get the exact previous
    idxmin = which.min(abs(newDeltas - min(oldDeltas)))
    if(newDeltas[idxmin] > min(oldDeltas)) idxmin = idxmin - 1

    # get the exact next
    idxmax = which.min(abs(newDeltas - max(oldDeltas)))
    if(newDeltas[idxmax] < max(oldDeltas) ) idxmax = idxmax +1

    # Extrapolate linearly if needed
    if(idxmin > 0) newIVs[1:idxmin] = newIVs[idxmin + 1]
    if(idxmax <= length(newIVs)) newIVs[idxmax:length(newIVs)] =
       newIVs[idxmax - 1]

    # Get new strikes and prices
    newStrikes = inverseDeltas(atmIV, s0, newDeltas, mat)
    newPrices = ebs.vec(newIVs, newStrikes, s0, mat, 'C')

    # plot?
    if(plot == T){
      plot.integral(oldDeltas, oldIVs, newDeltas, newIVs, h.c$
         strike,
                   h.c$price, newStrikes, newPrices, s0, 'C',
                        dates[i])
      readline(prompt = "Pause. Press <Enter> to continue to puts
         ...")
    }

    # Add term
    hvals[i] = hvals[i] + trapz(newStrikes, newPrices*omega(
       newStrikes, s0))

    #
       ##############################################################

    # Section 2: Fur Puts
    # Subset to puts
    h.p = subset(h, type == 'P')

    # Create deltas
    atmIV = h.p$implVol[which.min(abs(h.p$strike - s0))]


    # Throw ITM and get implied volatilities
    tempp = subset(h.p, strike <= s0)
    oldIVs = iv(tempp)
    oldDeltas = callDeltas(atmIV, s0, tempp$strike, mat)

    # Discard Deltas > 0.99 or less than <0.01
    idx = which(oldDeltas > 0.99)
    if(length(idx) != 0){
      oldIVs = oldIVs[-idx]
      oldDeltas = oldDeltas[-idx]
    }

    # Interpolate implied volatility wrt deltas
    # using a smoothing spline with smoothing 0.99
    l = sreg(y = oldIVs, x = oldDeltas, lambda = 0.99)

    # Find delta wich gives the ATM strike
    deltaATM = uniroot(
      function(x) inverseDeltas(atmIV, s0, x, mat) - s0, c(0,1)
    )$root
    newDeltas = seq(deltaATM, 0.99, length.out = 500)

    # Predict new implied volatilites
    newIVs = predict(l, newDeltas)

    # Extrapolate linearly outside known deltas
    # get the exact previous
    idxmin = which.min(abs(newDeltas - min(oldDeltas)))
    if(newDeltas[idxmin] > min(oldDeltas)) idxmin = idxmin - 1

    # get the exact next
    idxmax = which.min(abs(newDeltas - max(oldDeltas)))
    if(newDeltas[idxmax] < max(oldDeltas) ) idxmax = idxmax +1

    # Extrapolate linearly if needed
    if(idxmin > 0) newIVs[1:idxmin] = newIVs[idxmin + 1]
    if(idxmax <= length(newIVs)) newIVs[idxmax:length(newIVs)] =
       newIVs[idxmax - 1]

    newStrikes = inverseDeltas(atmIV, s0, newDeltas, mat)
    newPrices = ebs.vec(newIVs, newStrikes, s0, mat, 'P')

```

```r
    # plot?
    if(plot == T){
      plot.integral(oldDeltas, oldIVs, newDeltas, newIVs, h.p$
          strike,
                            h.p$price, newStrikes, newPrices, s0, 'P',
                              dates[i])
      readline(prompt = "Pause. Press <Enter> to continue to next
          date...")
    }

    # Add term to integral
    hvals[i] = hvals[i] + trapz(newStrikes, newPrices*omega(
        newStrikes, s0))
  }
  return(hvals)
}


# Example
# ex = subset(read.csv("cleanData.csv"), expiration %in% 16:23)
# calcIntegral(ex, plot = T)
```

Script A.7: Calculating the integrals.

```r
require(pracma)
source("functionsDF.R")
#################################################################
# Function calcIntegral takes a data frame containing obs for
# a single observation and calculates the integral as it is
# explained in the text. It uses plot.integral for plotting the
# fitting and the area to be integrated. Function omega is used
# to calculate omega.

calcIntegral <- function(data, plot = F, step = 0.001){
  dates = unique(data$date)
  hvals = numeric(length(dates))

  # plot parameters
  if(plot == T) par(mfrow = c(1,2))

  # For-loop to compute integral for each date
  for(i in 1:length(dates)){
    # For each date subset to date
    curdate = dates[i] # current date
    h = subset(data, date == curdate)
    s0 = h$underlying[1]
    mat = h$expiration[1]

    # calculate first term of integral
    bond = h$discount[1]

    #
       ##############################################################
    # Section 2: For Calls

    h.c = subset(h, type == 'C')

#     # Keep only the last ITM observation
#     idx = which(h.c$strike > s0)[1] - 1
#     if(!is.na(idx) & idx >= 0) h.c = h.c[idx:nrow(h.c), ]

    x = h.c$strike/s0
    y = log(h.c$price)

    if(plot == T)
      plot(x, y, xlab = "moneyness", ylab = "log(price)",
           main = paste("Interpolation - ", curdate, sep = ""))

    # Calculate vega weights
    ivs = iv.vec(h.c$price, h.c$strike, s0, mat, 'C', h.c$implVol
       )
    vs = vega(ivs, h.c$strike, s0, mat)
    # For ITM use the weights of far-OTM

    l = lm(y ~ poly(x, 2, raw = T), weights = vs)
    strikes.c = seq(1, round(1.02*max(x), 2), step)
    prices.c = as.numeric(predict(l, data.frame(x = strikes.c)))

    if(plot == T) lines(strikes.c, prices.c)

    strikes.c = strikes.c*s0
    prices.c = exp(prices.c)

    if(plot == T){
      plot(strikes.c, prices.c,
           main = paste('Calls', " for ", curdate, sep = ""),
           xlim = range(c(strikes.c, h.c$strike)),
           ylim = range(c(prices.c, h.c$price)),
           type = 'l', xlab = "strike", ylab = "price")
      points(h.c$strike, h.c$price, pch = 20)
      abline(v = s0, lty = 2)
      readline(prompt = "Press <Enter> to proceed to puts.")
    }

    #
       ##############################################################
    # Section 2: Fur Puts
    h.p = subset(h, type == 'P')

#     # Keep only the last ITM observation
#     idx = tail(which(h.p$strike < s0), 1) + 1
#     if(!is.na(idx) & nrow(h.p) >= idx) h.p = h.p[1:idx, ]

    x = h.p$strike/s0
    y = log(h.p$price)

    if(plot == T)
      plot(x, y, xlab = "moneyness", ylab = "log(price)",
           main = paste("Interpolation - ", curdate, sep = ""))

    # Calculate vega weights
    ivs = iv.vec(h.p$price, h.p$strike, s0, mat, 'P', h.p$implVol
      )
    vs = vega(ivs, h.p$strike, s0, mat)

    l = lm(y ~ poly(x, 2), weights = vs)
    strikes.p = seq(round(0.96*min(x), 2), 1, step)
    prices.p = predict(l, data.frame(x = strikes.p))

    if(plot == T) lines(strikes.p, prices.p)

    strikes.p = strikes.p*s0
    prices.p = exp(prices.p)

    if(plot == T){
      plot(strikes.p, prices.p,
           main = paste('Puts', " for ", curdate, sep = ""),
           xlim = range(c(strikes.p, h.p$strike)),
           ylim = range(c(prices.p, h.p$price)),
           type = 'l', xlab = "strike", ylab = "price")
      points(h.p$strike, h.p$price, pch = 20)
      abline(v = s0, lty = 2)
      readline(prompt = "Press <Enter> to proceed to next day.")
    }

    #
      ##############################################################

    # Compute the integral

    hvals[i] = bond +
      trapz(strikes.c, prices.c*omega(strikes.c, s0)) +
      trapz(strikes.p, prices.p*omega(strikes.p, s0))
  }

```

```r
  # plot defaults
  par(mfrow = c(1,1))
  return(hvals)
}

# # Example
# ex = subset(read.csv("cleanData.csv"), expiration %in% 16:23)
# calcIntegral(ex, plot = T, step = 0.01)
```

Script A.8: Calculating the forward variances.

```r
source("interpolatingMaturities.R") # Load functions
source("integration.R")

nobsPerDateAndType <- function(data){
# This function counts the observations per type and price for
# a data frame containing observations for a singular expiration.

  l2 =
    lapply(split(data, list(data$type, data$date)), function(x){
      return(nrow(x))
    })

  cn = matrix(unlist(strsplit(names(l2), "[.]")), ncol = 2,
              byrow = T)
  cn = as.data.frame(cn)
  colnames(cn) = c('type', 'date')
  cn$date = as.Date(cn$date)
  cn$nobs = as.numeric(l2)


  return(cn)
}

#################################################################
# Load clean data and create the data frame for the series

data = read.csv(file= "cleanData.csv") # Load 17421 obs
data$date = as.Date(data$date) # Dates
hts = data.frame(unique(data$date))
colnames(hts) = "date"



#################################################################
# Create series

###############################################################
# First maturity, days in [16,23]
```

```r
h1 = subset(data, expiration %in% 16:23)

# No dates are missing and each date has at least 4 observations
numobs = nobsPerDateAndType(h1)
summary(numobs)
dts2 = numobs[numobs$nobs < 4, ]$date
dts2

hts$H1 = calcIntegral(h1)

##################################################################
# Second maturity, days in [44,53]
# missing for "1999-06-30"
h2 = subset(data, expiration %in% 44:53)

# Interpolate for missing value
temp = newMaturity(subset(data, date == "1999-06-30"), 49)

# bind new observations with2 old and sort
h2 = rbind(h2, temp)
h2 = h2[order(h2$date, h2$type, h2$expiration, h2$strike), ]

#Every date h2as at least 4 obsrevations
numobs = nobsPerDateAndType(h2)
summary(numobs)

# Calculate Integral
hts$H2 = calcIntegral(h2)

##################################################################
# Third maturity, days in [75,86]
# No missing dates
h3 = subset(data, expiration %in% 75:86)

# There are dates with less than 4 obs for calls or puts
numobs = nobsPerDateAndType(h3)
summary(numobs)
missing = numobs[numobs$nobs < 4, ]

# Interpolate maturities for those
for(i in 1:nrow(missing)){
  # subset data to the date and the type of the missing
      observations
  temp = subset(data, date == missing[i, ]$date & type == missing
      [i, ]$type)

  # Throw and backup known strikes
  oldStrikes = subset(temp, expiration %in% 75:86)
  temp = subset(temp, !(expiration %in% 75:86) & !(strike %in%
      oldStrikes$strike))

  # Create new obs for remaining strikes for the
  # expiration of the known observations
  temp = newMaturity(temp, oldStrikes$expiration[1])
  temp = rbind(temp, oldStrikes)
  temp = temp[order(temp$strike), ]

  # Recheck arbitrage
  if(temp$type[1] == 'C'){ temp = checkPrices.calls(temp)
  }else temp = checkPrices.puts(temp)

  # Bind new strikes with with initial dataframe
  h3 = rbind(h3, subset(temp, !(strike %in% oldStrikes$strike)))
}
# sort
h3 = h3[order(h3$date, h3$type, h3$expiration, h3$strike), ]

# Now there are at least 4 maturities for each day and type
numobs = nobsPerDateAndType(h3)
summary(numobs)

hts$H3 = calcIntegral(h3)

###############################################################
# Forth maturity, days in [107,114]
h4 = subset(data, expiration %in% 107:114)

# Find missing
notidx = which(!(hts$date %in% unique(h4$date)))
dts = hts$date[notidx]
length(dts)#=> missing 73 days

# Fill for missing with progress bar
pb <- txtProgressBar(min = 0, max = length(dts), style = 3)
for(i in 1:length(dts)){
  temp = newMaturity(subset(data, date == dts[i]), 109)
  h4 = rbind(h4, temp)
  setTxtProgressBar(pb, i)
}
# sort
h4 = h4[order(h4$date, h4$type, h4$expiration, h4$strike), ]

# There are dates and types with less than 4 obs
numobs = nobsPerDateAndType(h4)
summary(numobs)
missing = numobs[numobs$nobs < 4, ]

# there were obs that had less than 4 obs and they were
# created by interpolating volatilities:
missing$date %in% dts
#=> 2000-06-30.P-3obs, 2000-07-31.P-3obs,
#=> 2000-09-29.C-3obs, 2000-12-29.C-2obs
missing = subset(missing, !(date %in% dts))

for(i in 1:nrow(missing)){
  # subset data to the date and the type of the missing
      observations
  temp = subset(data, date == missing[i, ]$date & type == missing
      [i, ]$type)

  # Throw and backup known strikes
  oldStrikes = subset(temp, expiration %in% 107:114)
  temp = subset(temp, !(expiration %in% 107:114) & !(strike %in%
      oldStrikes$strike))

  # Create new obs for remaining strikes for the
  # expiration of the known observations
  temp = newMaturity(temp, oldStrikes$expiration[1])
  temp = rbind(temp, oldStrikes)
  temp = temp[order(temp$strike), ]

  # Recheck arbitrage
  if(temp$type[1] == 'C'){ temp = checkPrices.calls(temp)
  }else temp = checkPrices.puts(temp)

  # Bindwith initial dataframe
  h4 = rbind(h4, subset(temp, !(strike %in% oldStrikes$strike)))
}
# sort
h4 = h4[order(h4$date, h4$type, h4$expiration, h4$strike), ]
# Calculate integral
hts$H4 = calcIntegral(h4)

################################################################
# Export
# write.csv(hts, file = "Hseries.csv", row.names = F)

################################################################
# Load and plot time-series

# data = read.csv(file = "Hseries.csv")
# data$date = as.Date(data$date)
data = hts
rm(list=setdiff(ls(), "data"))

# Create forward variances
fvs = data.frame(data$date)
```

```
colnames(fvs) = "date"
fvs$y = -log(data$H1)
fvs$f2 = log(data$H1) - log(data$H2)
fvs$f3 = log(data$H2) - log(data$H3)
fvs$f4 = log(data$H3) - log(data$H4)



#plot
ylabs =
c(expression(paste("Monthly forward variance ", ~~y[t]^{~(1)})),
  expression(paste("Monthly forward variance ", ~~f[t]^{~(2)})),
  expression(paste("Monthly forward variance ", ~~f[t]^{~(3)})),
  expression(paste("Monthly forward variance ", ~~f[t]^{~(4)})))

# plot pars
oldmar = par()$mar
par(mfrow = c(2, 2), mar = c(2.1, 5.1, 2.1, 2.1))

for(i in 1:4){
  plot(fvs$date, fvs[, i+1], type = 'l', ylim = c(0, 0.02),
       xlab = "", ylab = ylabs[i], xaxt="n")
  axis.Date(1, x = fvs$date, format="%m/%y")
}
par(mfrow = c(1,1), mar = oldmar)
```

Script A.9: Forward Variances' time series analysis

```
################################################################
# Function new Maturity takes a data frame of observations for a

# Create the time series object
fvs = read.csv("forVars.csv", stringsAsFactors = F)
fvs$date = as.Date(fvs$date)
fvs.ts = ts(fvs[, 2:5], start = c(1998, 9), end = c(2008, 9),
            frequency = 12)

# Get some statistics
library(stargazer);
summary(fvs.ts)
summary(fvs.ts*100)
stargazer(fvs[, 2:5]*100, out.header = FALSE, digits = 4,
          title = "Some basic statistics about the forward
          variances produced in Chapter 2. The sample period is
          09/1988 to 09/2008 (121 observations) and values are
          percentages.",
          font.size = "footnotesize",iqr = T, nobs = F,
          label = "tab:forVarBasic",
          table.placement = "!ht")
```

```

# cross-correaltions table
cor(fvs.ts)

# Autocorrelations
titles =
 c(expression(paste("Monthly forward variance ", ~~y[t]^{~(1)})),
   expression(paste("Monthly forward variance ", ~~f[t]^{~(2)})),
   expression(paste("Monthly forward variance ", ~~f[t]^{~(3)})),
    expression(paste("Monthly forward variance ", ~~f[t]^{~(4)}))
)

# ACFs for means
par(mfrow = c(2, 2))
y.acf = acf(fvs.ts[, 1], main = titles[1])
f1.acf = acf(fvs.ts[, 2], main = titles[2])
f2.acf = acf(fvs.ts[, 3], main = titles[3])
f3.acf = acf(fvs.ts[, 4], main = titles[4])
par(mfrow = c(1, 1))

# ACFs for vars
par(mfrow = c(2, 2))
acf((fvs.ts[, 1] - mean(fvs.ts[, 1]))^2, main = titles[1],
    ylab = "ACF -- var")
acf((fvs.ts[, 2] - mean(fvs.ts[, 2]))^2, main = titles[2],
    ylab = "ACF -- var")
acf((fvs.ts[, 3] - mean(fvs.ts[, 3]))^2, main = titles[3],
    ylab = "ACF -- var")
acf((fvs.ts[, 4] - mean(fvs.ts[, 4]))^2, main = titles[4],
    ylab = "ACF -- var")
par(mfrow = c(1, 1))

acf.matrix = data.frame(cbind(y.acf$acf[2:7], f1.acf$acf[2:7],
                              f2.acf$acf[2:7], f3.acf$acf[2:7]))
for(i in 1:6) row.names(acf.matrix)[i] = paste("ACF(", i, ")",
                                               sep = "")
names(acf.matrix) = names(fvs[2:5])
acf.matrix


# Unit root test
# Truncation lag parameter = 4
lapply(fvs.ts, PP.test)
fvs.ts.pptest = data.frame(cbind(
  sapply(fvs.ts, function(x){
    temp = PP.test(x)
    return(c(temp$statistic, temp$p.value))
  })))
names(fvs.ts.pptest) = names(fvs[2:5])
```

```r
row.names(fvs.ts.pptest) = c("Dickey-Fuller", "p-vlaue")
fvs.ts.pptest

# Fit an ARMA(1, 1) - GARCH(1, 1) model to each series
# change parchOrder and armaOrder in rugarch for different
# ARMA(p,q)-GARCH(p,q) models
require(rugarch)
fvs.model =
  ugarchspec(variance.model =
               list(model = 'sGARCH', garchOrder = c(1, 1)),
             mean.model =
               list(armaOrder = c(1, 1), include.mean = F),
             distribution.model = "norm")

# Rugarch return many parameters..
# to understund -- str(y.argagrach)
y.armagrach = ugarchfit(spec = fvs.model, data = fvs.ts[, 1])
f2.armagrach = ugarchfit(spec = fvs.model, data = fvs.ts[, 2])
f3.armagrach = ugarchfit(spec = fvs.model, data = fvs.ts[, 3])
f4.armagrach = ugarchfit(spec = fvs.model, data = fvs.ts[, 4])

# Information criteria
cbind(infocriteria(y.armagrach), infocriteria(f2.armagrach),
      infocriteria(f3.armagrach), infocriteria(f4.armagrach))

# Create latex table
require(Hmisc)
# cross-correlations
latex(cor(fvs.ts), file = "", digits = 3, ctable = T,
      caption = "", size = "small")
# autocorrelations
latex(acf.matrix, file = "", digits = 3, ctable = T, caption = ""
    )
# phillips-perron test
latex(fvs.ts.pptest, file ="", digits = 3, ctable = T)
# model parameters
ag.pars = cbind(coef(y.armagrach), coef(f2.armagrach),
                coef(f3.armagrach), coef(f4.armagrach))
colnames(ag.pars) = names(fvs[2:5])
latex(ag.pars, file = "", dec = 3, ctable = T)
# AR standard error coefficients
latex(t(c(y.armagrach@fit$se.coef[1],
        f2.armagrach@fit$se.coef[2], f3.armagrach@fit$se.coef[3],
  f4.armagrach@fit$se.coef[4])), dec = 3, ctable = T, file = "")

################################################################
# Plots
require(ggplot2)
require(reshape2) # for melt
```

```r
library(scales) # for data breaks
meltfvs = fvs
meltfvs$y = (1+fvs$y)^(30/19) - 1
meltfvs1 = melt(meltfvs, id = "date")
meltfvs[, 2:5] =
  sapply(meltfvs[, 2:5],
         function(x) cumsum(x)/cumsum(rep(1, length(x))))
meltfvs = melt(meltfvs, id = "date")

ggplot(meltfvs1, aes(x = date, y = value, color = variable, group
    = variable)) +
  geom_line() +  scale_x_date(labels = date_format("%m/%y"),
      breaks = date_breaks("year")) + xlab("") +
  theme_classic() + scale_colour_grey(start = 0, end = .9) + ylab
      ("")


ggplot(meltfvs, aes(x = date, y = value,
                    color = variable, group = variable)) +
  geom_line() +
  scale_x_date(labels = date_format("%m/%y"),
               breaks = date_breaks("year")) + xlab("") +
  theme_classic() + scale_colour_grey(start = 0, end = .9) +
  ylab("sample mean until date") +
  ylim(c(0.006, 0.015))
```

Script A.10: Predicting real economic activity

```r
# Create the time series object
fvs = read.csv("forVars.csv", stringsAsFactors = F)
fvs$date = as.Date(fvs$date)

# nonfarm: Non-farm payroll
nonfarm = read.csv(file = "nonfarm.txt", header = T, skip = 9)
nonfarm = nonfarm[, 2:4]
names(nonfarm) = c("date", "month", "nf")
# keep relevant months
nonfarm = nonfarm[9:130, ]
nonfarmGrowth =
  diff(nonfarm[, 3], lag = 1)/nonfarm[1:(nrow(nonfarm) - 1), 3]
rm(nonfarm)

# IP: industrial production
ip = read.csv("FRB_G17.csv", header = T)
ipGrowth = diff(ip[, 2], lag = 1)/ip[1:(nrow(ip) - 1), 2]
rm(ip)

yield = read.csv("FRB_H15.csv", header = T, na.strings = "ND")
names(yield) = c("date", "threeMonth", "tenYear")
```

```
yield$date = as.Date(yield$date, format = "%m/%d/%Y")

# subset to last day for each month
yield.last = tail(yield, 1) # 31/10/2008
yield = subset(yield, date %in% unique(fvs$date))
yield = rbind(yield, yield.last)
rm(yield.last)

# Yield slope as 10year - 3month
yieldSlope = (yield$tenYear - yield$threeMonth)/100
rm(yield)

require(sandwich)
require(lmtest)
require(Hmisc)
reg.Data = cbind(fvs[, 2:5], ipGrowth, yieldSlope, nonfarmGrowth)
rm(fvs, ipGrowth, nonfarmGrowth, yieldSlope)
# Newey-West regression for IP with yield
summary(fitIP <- lm(ipGrowth ~ y + f2 + f3 + f4 + yieldSlope,
                        data = reg.Data))
ip.cf = coeftest(fitIP, df = Inf,
                  vcov = NeweyWest(fitIP, prewhite = F))

floor(bwNeweyWest(fitIP, kernel = "Bartlett", prewhite = F))
#=> 6
dwtest(fitIP)

# latex
temp = rbind(coef(fitIP), ip.cf[, 4]) # coefs and pvals
temp = cbind(temp, c(summary(fitIP)$adj.r.squared,
                         dwtest(fitIP)$p.value)) # add R^2 and DW
latex(temp, file ="", dec = 3, ctable = T)
# usual pvals
latex(t(summary(fitIP)$coefficients[, 4]), dec = 3, ctable = T,
    file = "")

# Repete the above for for IP without yield
summary(fitIP <- lm(ipGrowth ~ y + f2 + f3 + f4,
                        data = reg.Data))

# For nonfarm with yield
summary(fitIP <- lm(nonfarmGrowth ~ y + f2 + f3 + f4 + yieldSlope
    ,
                        data = reg.Data))

# For nonfarm without yield
summary(fitIP <- lm(nonfarmGrowth ~ y + f2 + f3 + f4,
                        data = reg.Data))

```

```r
# Diagnostic Plots for LM
par(mfrow = c(1,2))
plot(fitIP$fitted.values, rstandard(fitIP),
     ylab = "Standardized Residual", ylim = c(-3, 3),
     xlab = "Fitted Values",
     main = "Homoscedasticity", pch = 19)
# Autocorrelation of residuals
n = length(residuals(fitIP))
plot(residuals(fitIP)[1:(n-1)], residuals(fitIP)[2:n],
     xlab = "RES - 1", ylab = "RES",
     main = "Scatterplot of Residuals vs Residuals")
par(mfrow = c(1,1))
```