MODELLING TRADE DURATIONS

MODELLING TRADE DURATIONS WITH THE BIRNBAUM-SAUNDERS AUTOREGRESSIVE MODEL

By

KIRILL MAYOROV, M.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Science

McMaster University

MASTER OF SCIENCE (2011)	McMaster University
(Statistics)	Hamilton, Ontario

TITLE:	Modelling trade durations with the Birnbaum-Saunders autoregressive model
AUTHOR:	Kirill Mayorov, M.Sc. (McMaster University)
SUPERVISOR:	Dr. N. Balakrishnan
NUMBER OF PAGES:	ix, 104

Abstract

In this thesis we study the Birnbaum-Saunders autoregressive conditional duration (BS-ACD) model. As opposed to the standard ACD model, formulated in terms of the conditional mean duration, the BS-ACD model specifies the time-varying model dynamics in terms of the conditional median duration. By means of Monte Carlo simulations, we examine the asymptotic behaviour of the maximum likelihood estimators. We then present a study of numerical efficacy of some optimization algorithms in relation to the BS-ACD model. On a practical side, we fit the BS-ACD model to samples for six securities listed on the New York Stock Exchange.

Acknowledgements

This work could never have been completed without ample guidance, assistance and encouragement. In this regard, I would like to thank the following persons:

My supervisor, Professor N. Balakrishnan, who knows virtually everything and finds joy in sharing. Thank you for sharing some of your knowledge and plenty of your time and wisdom. I am indeed fortunate to learn from a scholar of your calibre.

Professor Aaron Childs and Professor Maung Min-Oo for valuable advice and comments which led to important improvements of the thesis[1].

The Department of Mathematics and Statistics at McMaster University for providing financial support.

My friends in the Department, especially William Volterman, Debanjan Mitra, and Man Ho Ling, for helping me clarify my arguments.

A special word of thanks to my parents, Irina and Vyacheslav, for all the effort you put in to my education, both as a person and a student. Thank you for all the time you invested and the example you set.

My wife, Anna, who by now probably knows about R more than I do. Thank you for bearing with me through late nights and the many hours of being a computer widow. Thank you for your love, support, and companionship.

[1] All remaining errors are mine.

In Memory of Professor Alexander Abrosimov

Contents

List of Figures

List of Tables

Introduction

Since the seminal work of Engle and Russell [18], the modelling of financial data at transaction level has become an ongoing topic in the area of financial econometrics. The ultimate high-frequency data in finance are transaction-by-transaction or trade-by-trade data in security markets, so-called "ultra-high-frequency" data [17], where time is often measured in seconds. Transaction data possess a number of unique characteristics that do not appear in lower frequencies. The most salient feature, however, is that they are fundamentally irregularly spaced. This feature challenges researchers as standard econometric techniques refined over the years are no longer applicable. Moreover, recent models from the market microstructure literature[2] argue that time may convey information and, therefore, should also be modelled.

There are several approaches to tackling this feature of the data. One of them was originated by Engle and Russell in [18], where the durations between successive events (trades, quotes, price changes, etc.) are the quantities being modeled. These authors proposed a class of models called the Autoregressive Conditional Duration, or ACD, models, where conditional expected durations are modeled in a fashion similar to the way conditional variances are modeled using ARCH and GARCH models of Engle [16] and Bollerslev [10].

[2]See Section 1.1.

This thesis is devoted to a fairly new model introduced by Bhatti [8], referred to as the Birnbaum-Saunders autoregressive conditional duration model (BS-ACD). The model serves a competing alternative to the ACD class. As noted in [8], the BS-ACD model is not a true ACD model in the construction of [18], but rather a likelihood defined dynamic point process specified in terms of a time-varying conditional median duration.

The present work addresses numerical aspects of estimation of the parameter estimators of a simple case of the BS-ACD model, BS-ACD(1,1), and contains the following results:

- By performing numerical simulations, we obtain statistical evidence towards asymptotic normality of the maximum likelihood estimators.
- By comparing the numerical efficiency of the Nelder-Mead (NM) and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithms, we discuss the efficacy of the methods as applied to the BS-ACD(1,1) model estimation.
- We illustrate the BS-ACD(1,1) model by fitting it to samples of transaction-by-transaction data on six stocks traded on the New York Stock Exchange (NYSE).

The thesis is organized as follows. Chapter 1 reviews market microstructure theory and recalls the standard ACD model. In Chapter 2, we review important properties of the BS-ACD model and produce Monte Carlo simulations on the maximum likelihood estimation for various sample sizes. Chapter 4 is devoted to the efficacy studies of the NM and BFGS methods in the context of the BS-ACD model. In the last chapter we summarize results and discuss directions for future work.

Chapter 1

The basic ACD model

In this chapter, we review the essentials of the market microstructure theory, recall the standard ACD model and discuss its disadvantages.

1.1 Market microstructure theory

The theory of market microstructure is concerned with the frictions that cause the behaviour of asset prices to deviate from full-information (complete market) expectations. Microstructure literature provides theoretical arguments as to the possible nature of trading behaviour that could explain such deviations.

The benefit of a high-frequency data model is that factors that are identifiable at a transaction level might be lost in aggregations when sampling is fixed at regular intervals, such as analyzing daily or weekly prices. It is not our intention to provide a comprehensive review of market microstructure, but rather to highlight theories of direct concern. For a comprehensive review see [35] or [42].

Since the introduction of the asymmetric information model by Glosten

and Milgrom [23], models usually follow the assumption of two types of traders. The first type, called informed traders, have superior information unknown to the general public while the second group of traders, called liquidity traders, are motivated by non-information related concerns such as inventory control or portfolio rebalancing. The counterparty to a transaction is called a market maker. The market maker sets the price to compensate for the possibility that the trader has superior information. This compensation is incorporated in the bid-ask spread. The spread is thus, at least in part, a premium for the risk that the trader's information is superior to that of the market maker. When the market maker thinks it more likely that traders are of the informed kind, he increases the spread with the implication that volatility increases due to, amongst other things, bid-ask bounce at a wider spread. Over time the market maker infers the private information of the informed traders from the order flow and sets the bid-ask spread to "centre" on the new true value.

This provides a possible explanation for the presence of a bid-ask spread even in an efficient market for a risk neutral market maker with zero expected profit. Diamond and Verrecchia [13] argue that a proportion of informed traders might not already own the stock to which private information applies. If the news is good, this should have little effect, since one who expects the price to increase would purchase the stock. However, if the news is bad, and a trader does not own the stock, he would have to sell short. If a proportion of these traders are restricted from short selling, they will merely refrain from transacting in the share in question. Therefore, under the Diamond and Verrecchia [13] model, lower trade frequency (and hence longer trade duration) might be an indication of bad news, so that prices would tend to decrease in periods of relative quiet.

The model developed in [1] distinguishes between two types of liquidity

traders. "Discretionary" liquidity traders who have some control over the timing of their transactions within a certain time interval and "non-discretionary" liquidity traders who transact in a random fashion. They find that it is advantageous for "discretionary" traders to concentrate their trades together. As their model takes the arrival of private information to be exogenous and random, this implies that the proportion of informed traders should on average be higher when transactions occur less frequently, with the implication that higher average duration should imply higher volatility.

The Easley and O'Hara [14] model also assumes the arrival of liquidity traders in a random fashion, but does not distinguish between different kinds of liquidity traders. They argue that informed traders will want to trade while their information has value. This will cause informed trades to be clustered together. The market maker knows this and will be mindful of the order low to determine the likelihood of informed trading. They therefore suggest that periods of higher trade frequency would imply that informed trading, with the associated implications for spread and volatility, is then more likely.

Engle and Russell in [18] find evidence consistent with the model from [14], i.e. high trade frequency implies informed trading.

Thus, trade durations play an important role in market microstructure theory since they are used as proxies for the existence of information in the market, and therefore, are predictors for other market microstructure variables.

1.2 The basic ACD model

Recall that trade duration is the waiting time between two consecutive trades. Let us put this mathematically in the following

Definition 1.1 *Let T_i denote a sequence of random transaction times. Define the trade duration by $X_i = T_i - T_{i-1}$, $i = 1, 2, ..., N$.*

The most popular autoregressive duration approach is proposed by Engle [17] and Engle and Russell [18]. The basic idea of the autoregressive conditional duration (ACD) model is a dynamic parameterization of the conditional mean function

$$\Psi_i \equiv \Psi_i(\boldsymbol{\theta}) = \mathbb{E}[X_i|\mathcal{F}_{i-1};\boldsymbol{\theta}],$$

where $\boldsymbol{\theta}$ stands for an $M \times 1$ parameter vector, $M \in \mathbb{N}$, and $\mathcal{F}_{i-1}$ is the filtration of the processes generating the trades containing both the internal and external history of the processes up to time T_{i-1}.

Further, it is assumed that the standardized durations

$$\varepsilon_i = \frac{X_i}{\Psi_i}$$

follow an i.i.d. process defined on positive support with $\mathbb{E}[\varepsilon_i] = 1$ and independent of Ψ_i. Obviously, the ACD model can be regarded as a GARCH model for duration data.

The basic ACD specification is based on a linear parameterization of the conditional mean function

$$\Psi_i = \omega + \sum_{j=1}^{p} \alpha_j x_{i-j} + \sum_{j=1}^{q} \beta_j \Psi_{i-j}, \tag{1.1}$$

where $\omega > 0$, $\alpha \geqslant 0$, $\beta \geqslant 0$. Such a model is referred to as an $ACD(p,q)$ model.

When the distribution of ε_i is exponential, the resulting model is called an EACD(p,q) model. Similarly, if ε_i follows a Weibull or a (standardized) Generalized Gamma distribution, the model is a WACD(p,q) model and a GACD(p,q), respectively.

The expressions of the unconditional mean and variance of the ACD(p,q) model can be calculated and appear somewhat cumbersome. For simplicity, here only formulae for the EACD(1,1) model (e.g., [27]) are given:

$$\mathbb{E}\left[X_i\right] = \frac{\omega}{1-\alpha_1-\beta_1},$$

$$Var\left[X_i\right] = \left(\mathbb{E}\left[X_i\right]\right)^2 \left[\frac{1-(\alpha_1+\beta_1)^2+\alpha_1^2}{1-(\alpha_1+\beta_1)^2-\alpha_1^2}\right].$$

It is evident that $Var\left[X_i\right] > \left(\mathbb{E}\left[X_i\right]\right)^2$ thereby implying that the standard EACD model is featured by overdispersion. This property might be regarded as the counterpart to the "overkurtosis property" of the Gaussian GARCH model.

The weak stationarity[1] conditions of the EACD(1,1) model are similar to the weak stationarity condition of the GARCH model and are furnished by

$$\alpha_1+\beta_1 < 1,$$

$$(\alpha_1+\beta_1)^2+\alpha_1^2 < 1.$$

1.3 Disadvantages of the basic ACD model

There is considerable empirical evidence in the ACD literature against exponentially distributed errors. Paper [18] already considered the Weibull distribution, [26] suggested the Burr distribution, and [34] used the generalized gamma distribution. The ACD model is built by specifying a parametric representation of the conditional expected duration, and then relying on a quasi maximum likelihood argument (QML) for deriving the asymptotics of the estimated parameters. This means that the model potentially can be misspecified,

[1] That is, the first two moments of X_i are time invariant.

implying that the parameter estimators might be biased and inefficient. Only the EACD model provides consistent estimators under model misspecification by QML.

The reason is, as shown in [25], the QML parameter estimators of a correctly specified conditional mean model are consistent if, and only if, the quasi-maximum likelihood is based on a distribution belonging to the linear exponential family, regardless of what the true density is. The exponential distribution does belong to the linear exponential family, while the Weibull, Burr, and generalized gamma distributions do not (except for special cases). This is why the QML approach based on the exponential distribution will produce consistent estimators regardless of the true error distribution, while QML based on these other distributions will not unless the distribution used is the true density.

The degree of misspecification is relatively easily assessed by comparing the hazard function implied by the parametric specification of the estimated model with a nonparametric estimate of the same. Certain specifications will restrict the hazard function to shapes that may misrepresent the behaviour of the agents in the market.

Another serious drawback of transaction duration studies implemented so far is that the studies working with the WACD and EACD models are restricted to monotone hazard functions. Hence these models imply that the hazard function must either increase or decrease during a time-spell, or stay constant as for the exponential case. However, the hazard function is often found to be non-monotone and unimodal: the hazard function appears to be increasing for very small durations and decreasing for longer durations (see [34], [17], [26]). This feature is important to recognize since a misspecified hazard function can have severe consequences, in particular, in finite samples.

In [4], a comparison of several different ACD models is made. They conclude that a good model has to have a conditional distribution that is able to put a lot of probability mass on small durations but not too much on very small durations.

Thus, the combination of lack-of-fit and the absence of QML in existing models keeps the distributional specification of ACD models an open problem.

Chapter 2

The Birnbaum-Saunders autoregressive model

In this chapter, we discuss the BS-ACD model for trade durations and examine the results of Monte Carlo experiments.

2.1 The Birnbaum-Saunders distribution

The Birnbaum–Saunders (BS) distribution is an important lifetime distribution from a problem of material fatigue. It was first introduced in the context of failure caused by crack propagation [9], where the crack grows due to repeated stresses. In view of its importance in reliability, there is a large volume of literature on the BS distribution including problems of inference [15, 37], generalizations [32, 47, 53], and regression models [2, 33, 46].

The BS distribution is defined in terms of the standard normal distribution by means of the random variable

$$X = \frac{\beta}{4}\left[\alpha Z + \sqrt{(\alpha Z)^2 + 4}\right]^2, \tag{2.1}$$

where

$$Z = \frac{1}{\alpha}\left(\sqrt{\frac{X}{\beta}} - \sqrt{\frac{\beta}{X}}\right) \sim N(0,1), \tag{2.2}$$

$\alpha > 0$ is the shape parameter and $\beta > 0$ is the scale parameter. This is denoted by $X \sim BS(\alpha, \beta)$. Based on the (2.1)-(2.2) stochastic representations, the cumulative distribution function (cdf) of X is given by

$$F_X(x|\alpha,\beta) = \Phi\left[\frac{1}{\alpha}\left(\sqrt{\frac{x}{\beta}} - \sqrt{\frac{\beta}{x}}\right)\right], \; x > 0, \text{ and } \alpha, \beta > 0, \tag{2.3}$$

where $\Phi(\cdot)$ is the cdf of the standard normal distribution.

The probability density function (pdf) is then

$$f_X(x|\alpha,\beta) = \frac{1}{2\sqrt{2\pi}\alpha\beta}\left[\left(\frac{\beta}{x}\right)^{\frac{1}{2}} + \left(\frac{\beta}{x}\right)^{\frac{3}{2}}\right] \exp\left\{-\frac{1}{2\alpha^2}\left[\frac{x}{\beta} + \frac{\beta}{x} - 2\right]\right\}. \tag{2.4}$$

The BS distribution is the equilibrium mixture of the inverse Gaussian distribution and the convolution of this distribution with the chi-square distribution with one degree of freedom [53].

A property, which will turn out crucial for the purposes of the present thesis, was noted in [9]: the scale parameter β is the median as $F_X(\beta|\alpha,\beta) = \Phi(0) \equiv 0.5$. Further, the mean and variance are given, respectively, by

$$\mathbb{E}[X] = \beta\left(1 + \frac{\alpha^2}{2}\right) \tag{2.5}$$

and

$$Var[X] = (\alpha\beta)^2\left(1 + \frac{5\alpha^2}{4}\right).$$

As a matter of fact, as shown in [45] and as also can be inferred from [53], the BS distribution has finite moments of all orders.

As we discussed in Section 1.3, trade durations exhibit unimodal hazard functions. In this context, it is worth noting that the BS distribution has been

shown to have an unimodal hazard function for all values of α [29]. Hence, in accord with the conclusions of the previous section, the BS distribution can be used for an alternative ACD specification.

Example of pdf for different values of β and $\alpha = 1$ are located in Fig. 2.1, and their corresponding hazard functions are located in Fig. 2.2.

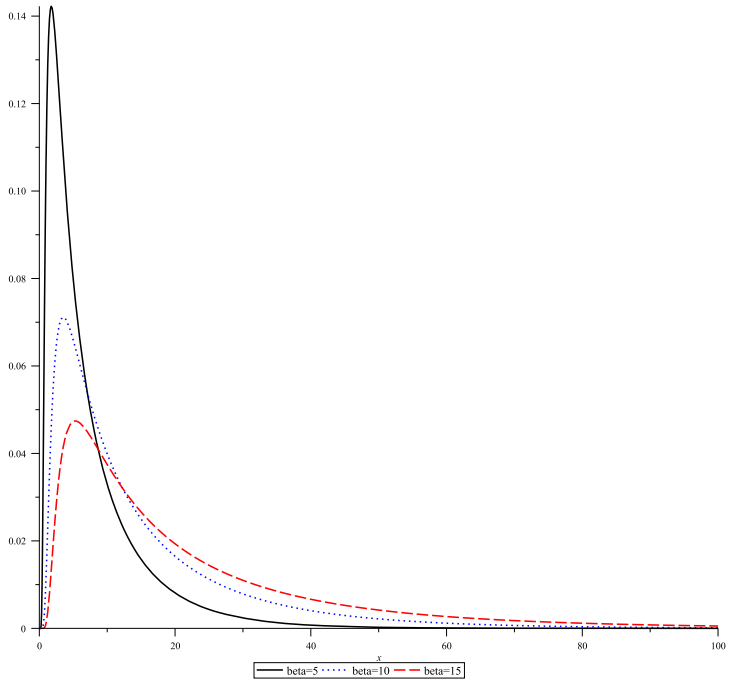

Figure 2.1: The BS pdf: $\alpha = 1$ and $\beta = 5, 10, 15$.

2.2 The BS-ACD model

Beginning from this section, we shall depart from the standard notation $BS(\alpha, \beta)$ for the Birnbaum-Saunders distribution toward $BS(\kappa, \sigma)$ as we shall reserve α and β to denote other parameters.

The BS-ACD model was introduced by Bhatti [8]. As noted in the

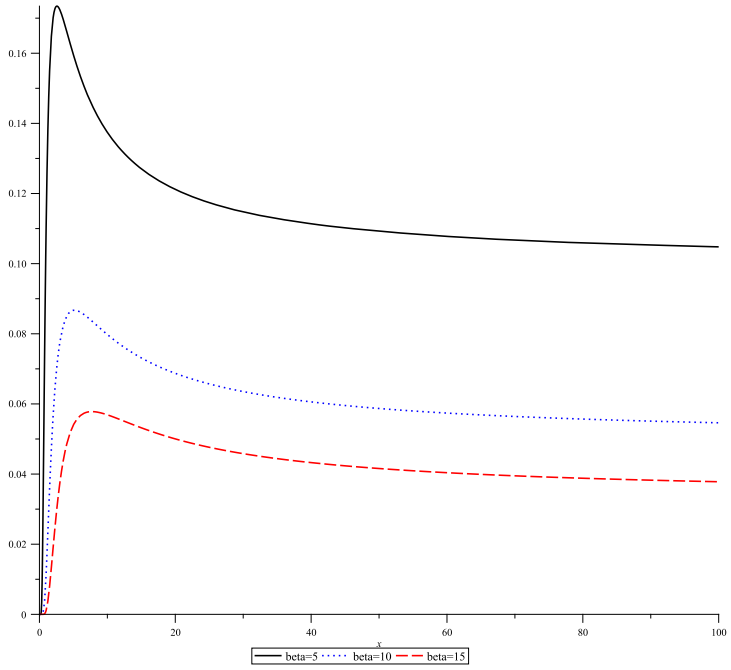

Figure 2.2: The BS hazard function: $\alpha = 1$ and $\beta = 5, 10, 15$.

paper, the model represents a dynamic point process model, but it is not however a true ACD model in the conventional sense. Moreover, quoting [8],

> *We have chosen to refer to the model as an ACD model because it is an "autoregressive conditional duration" model, and because we feel that it should be associated with traditional ACD models.*

As we saw in Section 2.1, σ is the median of the $BS(\kappa, \sigma)$ distribution. The BS-ACD model to be formulated below takes advantage of this property and is specified in terms of the conditional median duration $\sigma_i = F^{-1}(0.5|\mathcal{F}_{i-1})$, rather than the conditional mean duration. The conditional

pdf of $X|\sigma_i$ is given by

$$f_{X|\sigma}(X_i|\sigma_i;\kappa) = \frac{1}{2\sqrt{2\pi}\kappa\sigma_i}\left[\left(\frac{\sigma_i}{X_i}\right)^{\frac{1}{2}} + \left(\frac{\sigma_i}{X_i}\right)^{\frac{3}{2}}\right]\exp\left\{-\frac{1}{2\kappa^2}\left[\frac{X_i}{\sigma_i} + \frac{\sigma_i}{X_i} - 2\right]\right\}. \tag{2.6}$$

Temporal dynamics of the conditional median is governed by

$$\ln\sigma_i = \alpha + \sum_{j=1}^{p}\beta_j \ln\sigma_{i-j} + \sum_{j=1}^{q}\gamma_j\left[\frac{X_{i-j}}{\sigma_{i-j}}\right]. \tag{2.7}$$

Definition 2.1 *The model (2.6)-(2.7) is called the BS-ACD(p,q) model.*

By parameterizing of the BS distribution by its median we may hope to get nice statistical properties. For example, the median is a better measure of central tendency than the mean in highly skewed distributions. A detailed account of properties of conditional medians can be found in [51] and [22].

It should be noted that not every distribution admits a parameterization in terms of its mean. On the other hand, even if such a parameterization is possible, often this reparameterization can lead to estimation problems.

One can note from looking at (2.6)-(2.7) that the formulation does not assume either $X_i = \sigma_i\varepsilon_i$ or $X_i = \Psi_i\varepsilon_i$, where $\sigma_i = F^{-1}(0.5|\mathcal{F}_{i-1})$ and $\Psi_i = \mathbb{E}[X_i|\mathcal{F}_{i-1}]$. This is in part motivated by a desire to work with the distribution of X_i per se, conditional on a natural time varying parameter. In his work, Lunde [34] refers to the absence of the $X_i = \Psi_i\varepsilon_i$ assumption as the Nelson-form hinting at similarities with the EGARCH form proposed by Nelson [39]. On the other hand, Lunde calls the standard formulation (1.1) the Engle-Russell-form.

The temporal dynamics (2.7) is linear with respect to $\ln\sigma_i$. The logarithm automatically ensures the positivity of σ_i.

In this thesis, we are concerned with a simple situation of the BS-ACD(p,q) model, namely, the BS-ACD(1,1) model:

$$\ln \sigma_i = \alpha + \beta \ln \sigma_{i-1} + \gamma \left[\frac{X_{i-1}}{\sigma_{i-1}} \right]. \tag{2.8}$$

To capture a clustering effect well known in empirical finance, i.e. large (small) durations tend to be followed by large (small) durations, the log-linear model contains a lagged value of log-median process $\ln \sigma_{i-1}$. A large coefficient (near one) of β would indicate a strong persistence of clustering in the median.

Remark 2.2 *Empirical considerations suggest that the $BS-ACD(1,1)$ model be weakly stationary if $|\beta| < 1$.*

2.3 Monte Carlo simulations

As the requirement $X_i = \sigma_i \varepsilon_i$ was dispensed with, an estimation method of the BS-ACD model to be applied is maximum likelihood (ML) rather than QML. To estimate the parameters $(\alpha, \beta, \gamma, \kappa)$ of the BS-ACD(1,1) model, we maximize the logarithm of the ML function:

$$\begin{aligned} \ln \mathcal{L}(\alpha, \beta, \gamma, \kappa | x_0, x_1, ..., x_N) &\equiv \sum_{i=1}^{N} \ln f_{X|\sigma}(x_i | \sigma_i; \kappa) \\ &= \sum_{i=1}^{N} \left\{ -\ln \kappa - \ln \sigma_i + \ln \left[\left(\frac{\sigma_i}{x_i} \right)^{\frac{1}{2}} + \left(\frac{\sigma_i}{x_i} \right)^{\frac{3}{2}} \right] \right. \\ &\quad \left. - \frac{1}{2\kappa^2} \left[\frac{x_i}{\sigma_i} + \frac{\sigma_i}{x_i} - 2 \right] \right\}, \end{aligned} \tag{2.9}$$

where $\sigma_i = \sigma_i(\alpha, \beta, \gamma | x_0, x_1, ..., x_{i-1})$ is given by (2.8) and the additive constant has been ignored.

To study the ML estimators of the BS-ACD(1,1) model, we performed a simulation analysis. In financial high-frequency data, samples sizes usu-

ally span from 10000 to over 100000. We therefore chose to simulate BS-ACD(1,1) samples of sizes $N \in \{10000, 25000, 50000, 75000\}$ thus embracing small, medium, and large samples. The number of replications for each sample size is $n = 10000$.

The vector of true parameters $(\alpha, \beta, \gamma, \kappa) = (0.1, 0.9, 0.1, 1.1)$. The BS-ACD(1,1) samples were generated by transformation (2.1). All subsequent computations were done in the R statistical computing environment [43] with a number of dynamic link libraries written in the C programming language [44]. The R and C codes are listed in Appendix C.

Following [8], the BS-ACD(1,1) model was estimated through a two-stage procedure. At the beginning, we applied the Nelder–Mead (NM) method [38] to estimate the ACD parameters with the fixed initial value for κ fixed at

$$\sqrt{2\left(\frac{\overline{X}}{med(X)} - 1\right)}, \tag{2.10}$$

where $\overline{X}$ and $med(X)$ are, respectively, the sample mean and the (unconditional) sample median. This estimator is taken based on (2.5). Since the BS distribution is right skewed, with high probability (2.10) is well defined.

At the second stage, the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [11] was employed with the analytical gradient to estimate over the full parameter space. The gradient of the log-likelihood function is derived in elementary but technical Lemmas 2.3-2.4 and Proposition 2.5.

Lemma 2.3 *In the BS-ACD(1,1) model, the gradient of the conditional median σ_i (2.8) with respect to parameter vector $(\alpha, \beta, \gamma) \in \mathbb{R}^3$,*

$$\nabla\sigma_i = \left(\frac{\partial\sigma_i}{\partial\alpha}, \frac{\partial\sigma_i}{\partial\beta}, \frac{\partial\sigma_i}{\partial\gamma}\right),$$

is given recursively by

$$\begin{aligned}
\frac{\partial \sigma_i}{\partial \alpha} &= \sigma_i \left(1 + \frac{1}{\sigma_{i-1}} \frac{\partial \sigma_{i-1}}{\partial \alpha} \left(\beta - \gamma \frac{x_{i-1}}{\sigma_{i-1}}\right)\right), \\
\frac{\partial \sigma_i}{\partial \beta} &= \sigma_i \left(\ln \sigma_{i-1} + \frac{1}{\sigma_{i-1}} \frac{\partial \sigma_{i-1}}{\partial \beta} \left(\beta - \gamma \frac{x_{i-1}}{\sigma_{i-1}}\right)\right), \\
\frac{\partial \sigma_i}{\partial \gamma} &= \sigma_i \left(\frac{x_{i-1}}{\sigma_{i-1}} + \frac{1}{\sigma_{i-1}} \frac{\partial \sigma_{i-1}}{\partial \gamma} \left(\beta - \gamma \frac{x_{i-1}}{\sigma_{i-1}}\right)\right),
\end{aligned}$$

with $\nabla \sigma_0 = (0, 0, 0)$ *and* $i \geqslant 1$.

Proof Straightforward by applying the chain rule of differentiation. □

Lemma 2.4 *The expressions of Lemma 2.3 admit the following representations:*

$$\begin{aligned}
\frac{\partial \sigma_i}{\partial \alpha} &= \sigma_i \left(1 + \sum_{k=1}^{i-1} \prod_{j=k}^{i-1} \left(\beta - \gamma \frac{x_j}{\sigma_j}\right)\right), i \geqslant 1, \\
\frac{\partial \sigma_i}{\partial \beta} &= \sigma_i \left(\ln \sigma_{i-1} + \sum_{k=0}^{i-2} \ln \sigma_k \prod_{j=k+1}^{i-1} \left(\beta - \gamma \frac{x_j}{\sigma_j}\right)\right), i \geqslant 2, \\
\frac{\partial \sigma_i}{\partial \gamma} &= \sigma_i \left(\frac{x_{i-1}}{\sigma_{i-1}} + \sum_{k=0}^{i-2} \frac{x_k}{\sigma_k} \prod_{j=k+1}^{i-1} \left(\beta - \gamma \frac{x_j}{\sigma_j}\right)\right), i \geqslant 2,
\end{aligned}$$

where $\nabla \sigma_0 = (0, 0, 0)$ *and* $\nabla \sigma_1 = (\sigma_1, \sigma_1 \ln \sigma_0, \sigma_1 x_0 / \sigma_0)$.

Proof Follows directly from Lemma 2.3 by successive application of the recurrent formulae. □

Proposition 2.5 *In the BS-ACD(1,1) model, the gradient of the log-likelihood function (2.9) with respect to parameter vector* $\boldsymbol{\theta} = (\alpha, \beta, \gamma, \kappa) \in \mathbb{R}^4$,

$$\nabla \ln \mathcal{L} = \left(\frac{\partial \ln \mathcal{L}}{\partial \alpha}, \frac{\partial \ln \mathcal{L}}{\partial \beta}, \frac{\partial \ln \mathcal{L}}{\partial \gamma}, \frac{\partial \ln \mathcal{L}}{\partial \kappa}\right),$$

is given by

$$\begin{aligned}\frac{\partial \ln \mathcal{L}}{\partial \alpha} &= \sum_{i=1}^{N} G_i \frac{\partial \sigma_i}{\partial \alpha}, \frac{\partial \ln \mathcal{L}}{\partial \beta} = \sum_{i=1}^{N} G_i \frac{\partial \sigma_i}{\partial \beta}, \frac{\partial \ln \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{N} G_i \frac{\partial \sigma_i}{\partial \gamma}, \\ \frac{\partial \ln \mathcal{L}}{\partial \kappa} &= -\frac{N}{\kappa} + \frac{1}{\kappa^3} \sum_{i=1}^{N} \left(\frac{x_i}{\sigma_i} + \frac{\sigma_i}{x_i} - 2 \right),\end{aligned}$$

where

$$G_i = \frac{1}{2} \frac{(x_i - \sigma_i)}{(x_i + \sigma_i)} \frac{((x_i + \sigma_i)^2 - x_i \sigma_i \kappa^2)}{x_i \sigma_i^2 \kappa^2} \tag{2.11}$$

and the expressions of $\partial \sigma_i / \partial \alpha$, $\partial \sigma_i / \partial \beta$, *and* $\partial \sigma_i / \partial \gamma$ *are given by Lemma 2.3 or Lemma 2.4.*

Proof Straightforward by applying the chain rule of differentiation and elaborating elementary algebraic simplifications of the partial derivatives $\partial \ln \mathcal{L} / \partial \kappa$ and $\partial \ln f_{X|\sigma}(X_i|\sigma_i; \kappa) / \partial \sigma_i$, where we observe that

$$G_i \equiv \frac{\partial \ln f_{X|\sigma}(X_i|\sigma_i; \kappa)}{\partial \sigma_i}.$$

□

Corollary 2.6 *The ML estimator for* κ *has the following form:*

$$\widehat{\kappa} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\sqrt{\frac{X_i}{\sigma_i}} - \sqrt{\frac{\sigma_i}{X_i}} \right)^2}. \tag{2.12}$$

Proof Follows by equating $\partial \ln \mathcal{L} / \partial \kappa$ to 0 and recalling that $\kappa > 0$. □

Corollary 2.6 can be further used to simplify the likelihood equations for (α, β, γ):

$$\frac{\partial \ln \mathcal{L}}{\partial \alpha} = 0, \frac{\partial \ln \mathcal{L}}{\partial \beta} = 0, \frac{\partial \ln \mathcal{L}}{\partial \gamma} = 0. \tag{2.13}$$

By replacing κ^2 in (2.11) by (2.12), we can eliminate κ from equations (2.13). To any solution $\left(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}\right)$ to (2.13), then there corresponds a unique $\widehat{\kappa}$ given in (2.12).

For both stages of the ML estimation, the estimation of (α, β, γ) was initiated at $(0.01, 0.70, 0.01)$ and σ_0 was taken to be the unconditional median of the sample $x_1, ..., x_N$. The combination of the direct search (NM) and gradient based (BFGS) methods is justified by the fact that the two methods alone may face difficulties with convergence. For example, the NM algorithm alone may not converge or may converge to a nonstationary point [36]. On the other hand, the BFGS method need not converge unless the function has a quadratic Taylor expansion near an optimum.

By using the hybrid of the two methods (NM-BFGS), however, the convergence is quickly achieved. The issues with convergence and the efficacy of the NM, BFGS and NM-BFGS methods are addressed in Chapter 4.

We reported the ML estimation results in Table B.1. For each parameter and each sample size we report the following sample statistics for the MLE's: mean, skewness, kurtosis, bias, mean square error (MSE). In Table B.1 the sample skewness is defined as

$$\frac{\sqrt{n(n-1)}}{n-2} \frac{\left(\sum_{j=1}^{n} (x_j - \overline{x})^3\right)/n}{\left[\left(\sum_{j=1}^{n} (x_j - \overline{x})^2\right)/n\right]^{3/2}}$$

and the sample kurtosis as

$$\frac{\left(\sum_{j=1}^{n} (x_j - \overline{x})^4\right)/n}{\left[\left(\sum_{j=1}^{n} (x_j - \overline{x})^2\right)/n\right]^{2}},$$

where $\mathbf{x} = (x_1, ..., x_n)$ is a realization of a random sample.

The section is concluded by two technical remarks.

Remark 2.7 *The estimation of the BS-ACD(1,1) model was performed in the R statistical environment [43]. Due to the simulated samples size (ranging from*

10000 to 75000) and the recursive definition of the model, C functions were written for the R shared library [44]. While the simulated data sets would not be considered massive, they are quite large for time series data. The original package built in R alone was not able to perform the estimation. Computational software such as R and Matlab are designed for the efficient execution of vectorized and matrix computations. The BS-ACD model requires loops and executes much slower in R than in C. The difference between execution time in R and C is significant and determined the ability to estimate the model. In our case, estimation of a single sample of size 10000 was reduced from 57 seconds as an R function to 0.5 seconds as an R function with a C function. The increase in efficiency was paramount in performing the maximum likelihood estimation.

Remark 2.8 *A computer with processor Intel Core i5-2410M CPU @2.30 GHz and RAM 4 GB was used. The choice of the clock rate and the amount of random-access memory is essential. Older generation processors (Pentium, Core 2 Duo) lead to longer execution times. On the other hand, operating system (UNIX or Windows) hosting the simulations does not matter. However, author's limited experience shows that Windows x64 is strongly preferred over Windows x32.*

2.4 ML estimation results

A quick glance at the estimates in Table B.1 leads to the conclusion that $\widehat{\gamma}$ and $\widehat{\kappa}$ might be consistent and marginally asymptotically normally distributed. However, $\widehat{\alpha}$ and $\widehat{\beta}$ are persistently skewed and somewhat unstable, nonetheless, remaining "close" to a Gaussian distribution in terms of their skewness and

kurtosis values.

We once read the following argument [20]:

> *We usually apply normality tests to the results of processes that, under the null, generate random variables that are only asymptotically or nearly normal (with the "asymptotically" part dependent on some quantity which we cannot make large). In the era of cheap memory, big data, and fast processors, normality tests should always reject the null of normal distribution for large (though not insanely large) samples. And so, perversely, normality tests should only be used for small samples, when they presumably have lower power and less control over Type I rate.*

Hence, the question arises why we need to test for normality. First, nonlinearity and processes in empirical finance may lead to non-Gaussian distributions, and the generating mechanism of the processes can therefore be better understood by examining the distribution of selected variables. A second reason for implementing normality tests is that many statistical procedures require or are optimal under the assumption of normality, and it is therefore of interest to know whether or not this assumption is fulfilled. Of course it may also be of interest to test for the presence of other specific distributions, for example, extreme value distributions.

There are not many tests for normality which work in a robust way in the case of large samples. We chose to use the following tests: the Cramér–von Mises (CVM), Jarque–Bera (JB), and D'Agostino's (DA) omnimbis test (see [49] and references therein).

When N gets large, however, even the smallest deviation from perfect normality will lead to a significant result. A demonstration of this is in Table

2.1 where each normality test was applied to a simulated random sample from a genuine $N(0,1)$-distribution disturbed by adding points $\{1,0,2,0,1\}$ (which constitute only 0.05% of a sample of size 10000). The procedure was repeated 1000 times for each test and each sample size.

	Size of modelled samples						
	1000	**5000**	**10000**	**25000**	**50000**	**75000**	**100000**
CVM	0.154	0.686	0.965	1.000	1.000	0.966	0.491
JB	0.143	0.809	0.992	1.000	1.000	1.000	1.000
DA	0.178	0.836	0.988	1.000	1.000	1.000	1.000

Table 2.1: Proportion of p-values less than 0.05 in testing for normality (1000 runs).

In almost 100% of the cases, for large sample sizes the distribution is not seen any more as Gaussian when testing by the CVM, JB, and DA tests. Yet, by looking at the quantile-quantile (QQ) plots for the samples, one would probably never decide on a deviation from normality. With a very large sample size a normality test may detect statistically significant but unimportant deviations from normality. Hence standard normality tests are not practical for very large samples.

In this circumstances, the histogram and normal QQ-plot may be the single most valuable graphical aid in diagnosing how a population distribution appears to differ from a normal distribution. For values sampled from a normal distribution, the QQ-plot has the points all lying on or near the straight line drawn through the middle half of the points. Scattered points lying away from the line are suspected outliers. Since the histograms and QQ-plots are similar looking for parameters α, β, γ, and κ regardless of a sample size, for

conciseness only plots for $N = 10000$ were provided (see the figures in Section A.1 of Appendix A).

The histogram in Figure A.1 and the normal QQ-plot in Figure A.2 suggest that the ML estimators for α, β, γ, and κ should be marginally asymptotically distributed. However, as has been noted above, $\widehat{\alpha}$ and $\widehat{\beta}$ prove to be skewed (to a much higher degree than $\widehat{\gamma}$, and $\widehat{\kappa}$). In part, this might be caused by a residual effect of the estimation procedure (cf. Chapter 4) or/and the presence of outliers.

Given that the conventional measures of skewness and kurtosis are computed as an average and that averages are not robust [28] in the presence of outliers, we may want to apply a procedure commonly referred to as trimming, that is, to remove a percentage of the estimates on the both tails and recalculate the descriptive statistics. The trimming was performed separately on each estimate $\widehat{\alpha}$, $\widehat{\beta}$, $\widehat{\gamma}$ and $\widehat{\kappa}$ after sorting (in the ascending order) the sample thereof. We chose to trim 0.25% on each side. The trimmed ML estimates are shown in Table B.2. The quantiles of the trimmed samples were then plotted against the quantiles of the trimmed Gaussian distribution.

The reference line corresponding the truncated normal distribution is no longer a straight line but rather an S-shaped curve (see Figure A.3). The quantile plots for quantiles of $\widehat{\gamma}$ and $\widehat{\kappa}$ are close to their normal counterparts. However, a pronounced gap between the two curves is present in the case of $\widehat{\alpha}$ and $\widehat{\beta}$. Whatever the advantages are of such a quick, intuitively meaningful graphical method, there are pitfalls. A formal test is called for to assess what manner or extent of deviation from an S-shaped line fit might reasonably be expected to occur by chance, or how such reasonable variations might diminish as the sample size increases. Due to the difficulties faced by the standard normality tests mentioned earlier, a new test is needed tailored specifically for

the needs of the BS-ACD model.

To be able to judge whether the gap is statistically significant (for example, at the significance level 0.05 or 0.01), we devised a test for normality based upon quantiles (a discussion of a similar framework can be found in [21]). The idea of the test is analogous (and in a sense, which can be formalized, is equivalent) to that underlying the Kolmogorov-Smirnov test for discrepancy between two distribution functions.

Recall the following definition:

Definition 2.9 *Let X be a random variable with a cumulative distribution function (cdf) $F : D \to (0,1)$, where $D \subseteq \mathbb{R}$. The quantile function of X is defined as $Q(p) = \inf\{x \in D \mid p \leqslant F(x)\}$.*

It is clear that for a strictly monotone $F(\cdot)$ the quantile function $Q(p) = F^{-1}(p)$.

Sample quantiles provide nonparametric estimators of their population counterparts based on a set of independent observations $\{X_1, ..., X_n\}$ from the distribution F. Let $\{X_{(1)}, ..., X_{(n)}\}$ denote the order statistics of $\{X_1, ..., X_n\}$, and let $\widehat{Q}(p)$ denote the sample quantile definition. Then

$$\widehat{Q}(p) = (1-\lambda)X_{(j)} + \lambda X_{(j+1)},$$

where $(j-l)/n < p < (j-l+1)/n$ for some $l \in \mathbb{R}$ and $0 \leqslant \lambda \leqslant 1$. The value of λ is a function of $j = \lfloor pn + l \rfloor$ and $g = pn + l - j$ (here $\lfloor w \rfloor$ denotes the largest integer not greater than w). To numerically calculate sample quantiles, we used the default settings of the R function *quantile()*.

Then the proposed test statistic is

$$D_k^{QQ} = \sup_{p \in (0,1)} \left| \widehat{Q}_k(p) - \widehat{Q}_k^G(p) \right|, \tag{2.14}$$

where $k = 1, ..., K$ and $\widehat{Q}_k(\cdot)$ and $\widehat{Q}_k^G(\cdot)$ are, respectively, the sample quantile function of the trimmed MLE of a BS-ACD(1,1) model parameter (i.e. $\widehat{\alpha}$, $\widehat{\beta}$, $\widehat{\gamma}$ or $\widehat{\kappa}$) and the sample quantile function of the trimmed standard normal distribution, respectively. For computational convenience, before calculating (2.14), the MLE sample was standardized by applying the transformation:

$$s(X) = \frac{X - \overline{X}}{\sqrt{var(X)}},$$

and then the range of both the transformed MLE $Y = s(X)$ and generated $N(0, 1)$ samples was standardized to $[0, 1]$ by means of the following transformation:

$$\frac{Y - Y_{(1)}}{Y_{(N^*)} - Y_{(1)}},$$

where $N^* = N - 2\lfloor mN/100 \rfloor$ is the new sample size after removing $m\%$ of observations from each tail.

Critical values of D_k^{QQ} as $k \to \infty$ can be obtained by Monte Carlo simulations according to the following scheme:

1. Generate a sample of size N from BS-ACD(1,1) model with true values $\boldsymbol{\theta}^*$.
2. Obtain the MLE's $\widehat{\boldsymbol{\theta}} = \left(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}, \widehat{\kappa}\right)$.
3. Repeat steps 1-2 n times to obtain a sample of $\widehat{\boldsymbol{\theta}}$'s of size n.
4. Evaluate (2.14).
5. Repeat steps 1-4 K times to obtain a sample of D_k^{QQ} of size K.

Due to the extremely time consuming nature of the computations, we chose determine critical values for samples of size $N = 10000$ only (and $n =$

10000). We chose $m = 0.25\%$ and $K = 100$. Analogously, computations can be carried out for the other sample sizes. The critical values so obtained are shown in Table 2.2.

Also, this simulation can be viewed as a Monte-Carlo experiment with $N = 10000$ and $n = 1000000$. Plots corresponding to such an interpretation are depicted in Figure A.4. A distinct gap between the quantile curves of the trimmed MLE $\widehat{\alpha}$ and $\widehat{\beta}$ and trimmed $N(0,1)$ random variables is still in place whereas the quantile curves of $\widehat{\gamma}$ and $\widehat{\kappa}$ are virtually identical to the reference curves.

Quantile	α	β	γ	κ
90%	0.0745	0.0672	0.0291	0.0303
95%	0.0798	0.0712	0.0333	0.0322
99%	0.0882	0.0785	0.0403	0.0376

Table 2.2: Critical values of the test for normality designed for the BS-ACD(1,1) model.

The distances D^{QQ} between the sample quantiles of the trimmed ML estimates ($N = 10000$ and $n = 10000$) whose descriptive statistics are summarized in Table B.1 are given in Table 2.3.

	α	β	γ	κ
D^{QQ}	0.0658	0.0439	0.0111	0.0175

Table 2.3: Distances D^{QQ} for a particular sample of MLE's ($N = 10000$ and $n = 10000$).

Thus at the significance level of 0.01, we fail to reject the null hypothesis that the marginal distributions of the ML estimators $\widehat{\alpha}$, $\widehat{\beta}, \widehat{\gamma}$, and $\widehat{\kappa}$ are

Gaussian. This result contrasts with the conclusions reached in [8].

Remark 2.10 *Since an asymptotically normal estimator is consistent, the consistency of* $\widehat{\alpha}$, $\widehat{\beta}, \widehat{\gamma}$, *and* $\widehat{\kappa}$ *follows.*

The question remains whether the joint asymptotic distribution of $\widehat{\boldsymbol{\theta}} = \left(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}, \widehat{\kappa}\right)$ is normal. To date, as far as we are aware, theoretical regularity conditions ensuring joint asymptotic normality are only known for an ACD model with the $X_i = \Psi_i \varepsilon_i$ assumption [27]. We conjecture that in the absence of the assumption, limit theorems for martingales would have to be applied [6] with conditions, such as mixing and ergodicity [31], to be imposed on the ML function and its gradient.

Chapter 3

Application to real data

In this chapter, we fit the BS-ACD model to transaction-by-transaction data of six NYSE stocks. Coincidentally, the results of this chapter are almost identical to those obtained in Bhatti [8].

3.1 Description of data

We study tick-by-tick trade data[1] from the NYSE TAQ (Trade And Quote) database. Standard filtering rules were applied to the trade data: transactions recorded outside of normal trading hours (09:30-16:00 EST) were deleted, and simultaneous transactions were considered as a single transaction.

The following six stocks were chosen for the analysis: General Motors Corporation (GM), International Business Machines (IBM), Johnson and Johnson Company (JNJ), the McDonald's Corporation (MCD), the Proctor and Gamble Company (PG), and Schlumberger Limited (SLB). The period considered is from January 1, 2002 to February 28, 2002.

[1]The data sets were generously provided by Dr. Chad R. Bhatti (V.P., JPMorgan Chase).

Some descriptive statistics[2] are shown in Table 3.1 below.

	GM	IBM	JNJ	MCD	PG	SLB
Sample size	56,408	127,309	82,938	72,979	78,933	90,694
Median	9.000	5.000	7.000	7.000	7.000	5.000
60%-percentile	12.000	6.000	9.000	9.000	9.000	7.000
Mean	15.318	6.787	10.419	11.844	10.949	9.526
70%-percentile	17.000	8.000	11.000	13.000	12.000	10.000

Table 3.1: Descriptive statistics of the sample data: observations from 10:00 to 16:00.

It is known that trade durations are typically right skewed [18]. In the table, this is reflected by the fact that the sample means for all six samples lie between the 60%- and 70%-percentiles which are larger than the sample medians.

3.2 Removal of diurnal pattern

Usually one observes active trading during opening and closing hours and dormant trading around noon. This is reflected by short durations during active hours and longer durations around noon in a trading day. Figure 3.1 illustrates the diurnal effect for the GM sample: the trade durations were averaged over 30 minute time intervals (from 09:30 to 16:00) for all forty trading days and then smoothed by cubic splines. For computational convenience, calendar time was measured in seconds from midnight.

There are a number of ways to remove the diurnal pattern. For example,

[2]Technical details of the data representation are explained in Section D.2.

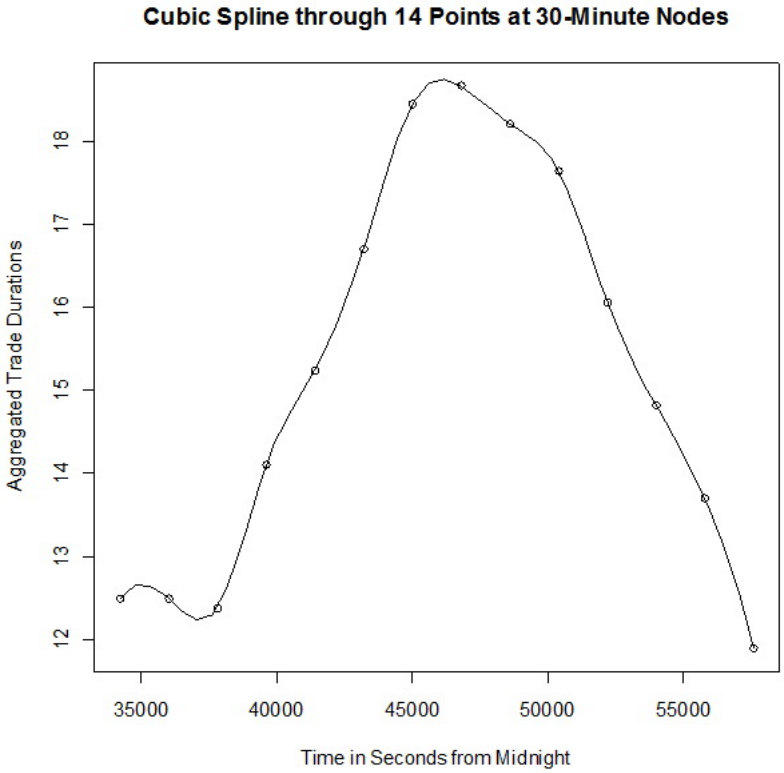

Figure 3.1: Presence of diurnal effect in GM trade durations.

using splines or Fourier expansions [52]. To remove the seasonal effects present in the duration process in a manner that is robust to outliers, we applied a semi-logarithmic version of the method suggested in [18] to get the diurnally adjusted durations.

Let as before $X_k = T_k - T_{k-1}$ be the trade duration, where T_k are calendar times of transactions and $\widehat{X}_k$ is the diurnally adjusted durations. Then we constructed

$$\widehat{X}_k = \exp\left[\ln X_k - \widehat{\phi}(T_k)\right],$$

where $\widehat{\phi}(T_k)$ is the fitted value from the regression

$$\ln X_k = \sum_{j=1}^{13} \beta_j Y_{j,k} + u_k, \; Y_{j,k} = \max(T_k - q_j, 0),$$

where q_j is the j-th partition point. Here q_1 is the first endpoint, 9:30, and q_{13} is the last end point, 15:30.

From now on, the term duration is understood as the diurnally adjusted durations and all of the subsequent data analysis is based on the diurnally adjusted durations unless stated otherwise.

3.3 Fitting the BS-ACD(1,1) model

Since the beginning of a trading days is usually very much different from the trading of the rest of the day, only observations in the period 10:00 to 16:00 are used. Further, for each day, we set the initial value σ_0 of the conditional median to be the median trade duration over the period [09:50,10:00). This is done to ensure that information is not carried over from a trading day to the subsequent one.

The estimation is performed by maximum likelihood. The parameter estimates are shown below.

	GM	**IBM**	**JNJ**	**MCD**	**PG**	**SLB**
$\widehat{\alpha}$	-0.0193	-0.0432	-0.0175	-0.0175	-0.0184	-0.0194
$\widehat{\beta}$	0.9878	0.9513	0.9759	0.9847	0.9864	0.9855
$\widehat{\gamma}$	0.0108	0.0309	0.0114	0.0101	0.0118	0.0117
$\widehat{\kappa}$	1.2136	0.8742	1.0431	1.2047	1.0636	1.1480

Table 3.2: BS-ACD(1,1) estimation results.

	GM	IBM	JNJ	MCD	PG	SLB
α	0.001	0.002	0.001	0.002	0.002
β	0.001	0.003	0.003	0.002	0.002
γ	0.001	0.001	0.001	0.001	0.001
κ	0.004	0.002	0.003	0.003	0.003

Table 3.3: Standard errors of the ML estimates.

In the figures, for the sake of brevity, we only display results for GM (the other stocks present very similar figures). Observe that $\widehat{\beta}$ is close to 1: the ACD parameters display strong autoregressive behavior. This effect is usually referred to as persistent, or tending to have long memory.

Standard errors for $\widehat{\theta} = \left(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}, \widehat{\kappa}\right)$ were calculated based on the White covariance matrix [18]:

$$WCov(\widehat{\theta}) = \left(\nabla^2 \ln \mathcal{L}\left(\widehat{\theta}\right)\right)^{-1} \left\{\left(\nabla \ln \mathcal{L}\left(\widehat{\theta}\right)\right)\left(\nabla \ln \mathcal{L}\left(\widehat{\theta}\right)\right)^t\right\} \left(\nabla^2 \ln \mathcal{L}\left(\widehat{\theta}\right)\right)^{-1},$$

where $\ln \mathcal{L}\left(\widehat{\theta}\right)$ is the value of the log-likelihood function at the MLE $\widehat{\theta}$.The standard errors are reported in Table 3.3.

Testing of $H_0 : \widehat{\alpha} = 0$, $H_0 : \widehat{\beta} = 1$, $H_0 : \widehat{\gamma} = 0$, $H_0 : \widehat{\kappa} = 1$ against $H_1 : \widehat{\alpha} \neq 0$, $H_1 : \widehat{\beta} \neq 1$, $H_1 : \widehat{\gamma} \neq 0$, $H_1 : \widehat{\kappa} \neq 1$, respectively, was done through the Wald statistic:

$$\frac{\widehat{\zeta} - \zeta_0}{se\left(\widehat{\zeta}\right)} \sim N(0,1),$$

where $\widehat{\zeta}$ and ζ_0 are a corresponding estimator and its proposed value, respectively. All estimates are statistically significant at the 1% level.

3.4 A diagnostic check of the BS-ACD(1,1) model

We now perform a diagnostic check of the $ACD(1,1)$ model based upon the Cox-Snell residuals [12]. As opposed to the standard $ACD(1,1)$ model, where $\widehat{X}_i/\widehat{\psi}_i$ is a valid residual, $\widehat{X}_i/\widehat{\sigma}_i$ is not a valid residual for the $BS-ACD(1,1)$ model. However, the Cox-Snell residual

$$-\ln\left(\widehat{S}\left(x_i \mid \mathcal{F}_{i-1}\right)\right),$$

where $\widehat{S}\left(x_i \mid \mathcal{F}_{i-1}\right)$ is the fitted conditional survival function, does constitute a valid residual for the model.

The Cox-Snell residuals will be $Exp(1)$-distributed if the model is correctly specified regardless of the model distributional assumption. We calculated the quantiles of the Cox-Snell residuals of the six fitted models and plotted them together with the quantiles of the $Exp(1)$ distribution (see Table 3.4 and Figure 3.2).

The residual analysis shows that the BS-ACD(1,1) model fits quantiles of $Exp(1)$ through the 99% quantile. In all six samples, the BS-ACD(1,1) model fits the sample data sufficiently well to provide valid maximum likelihood inference.

To test the absence of autocorrelations in the residuals, it is customary in the time series analysis to employ the Ljung-Box-Pierce test statistic

$$Q_N^m(z) = N(N+2)\sum_{k=1}^{m}\frac{\gamma_k^2}{N-k}$$

across m lags model, where $\gamma_k = corr\left(z_i, z_{i+k}\right)$ and N is the sample size.

For an $ACD(1,1)$ model, the statistic is $\chi^2(m-3)$-distributed. Following [8], we take $m = 15$. Then the critical value $\chi^2_{0.99}(12)$ equals 26.22. On

	Exp(1)	**GM**	**IBM**	**JNJ**	**MCD**	**PG**	**SLB**
1%	0.010	0.009	0.013	0.012	0.018	0.011	0.022
5%	0.051	0.038	0.030	0.031	0.040	0.033	0.045
10%	0.105	0.107	0.106	0.115	0.096	0.111	0.088
25%	0.288	0.304	0.315	0.291	0.279	0.304	0.272
50%	0.693	0.721	0.735	0.688	0.703	0.702	0.685
75%	1.386	1.423	1.377	1.382	1.406	1.382	1.404
90%	2.303	2.291	2.212	2.298	2.304	2.287	2.303
95%	2.996	2.926	2.886	2.971	2.960	2.962	2.984
99%	4.605	4.371	4.642	4.635	4.457	4.633	4.612

Table 3.4: Quantiles of the Cox-Snell residuals and Exp(1) distribution.

the other hand, calculations show that among the six samples the minimum of the empirical values for $Q_N^m(\cdot)$ is bigger than 63 for the Cox-Snell residuals. Hence we reject the null hypothesis for all six samples.

For a typical sample size of 50000 observations, as easy algebraic manipulations shows below, in order to fail to reject the null hypothesis the residual autocorrelation would have to not exceed 0.002 over 15 lags:

$$\begin{aligned} \sum_{k=1}^{15} \frac{\gamma_k^2}{N-k} &\leqslant 15\left(\max_k \gamma_k^2\right) \sum_{k=1}^{15} \frac{1}{50000-k} \approx 0.0045 \max_k \gamma_k^2, \\ \text{so if } Q_{50000}^{15}(z) &< 26.22, \text{ then } |\max_k \gamma_k| < \sqrt{\frac{26.22}{0.0045 \cdot 50000 \cdot (50000+2)}}, \end{aligned}$$

which is approximately 0.002. Thus, the Ljung-Box-Pierce test is not suitable for long time series.

Overall, the above practical results indicate that the BS-ACD model provides a good fit to trade durations and deserves further attention in future

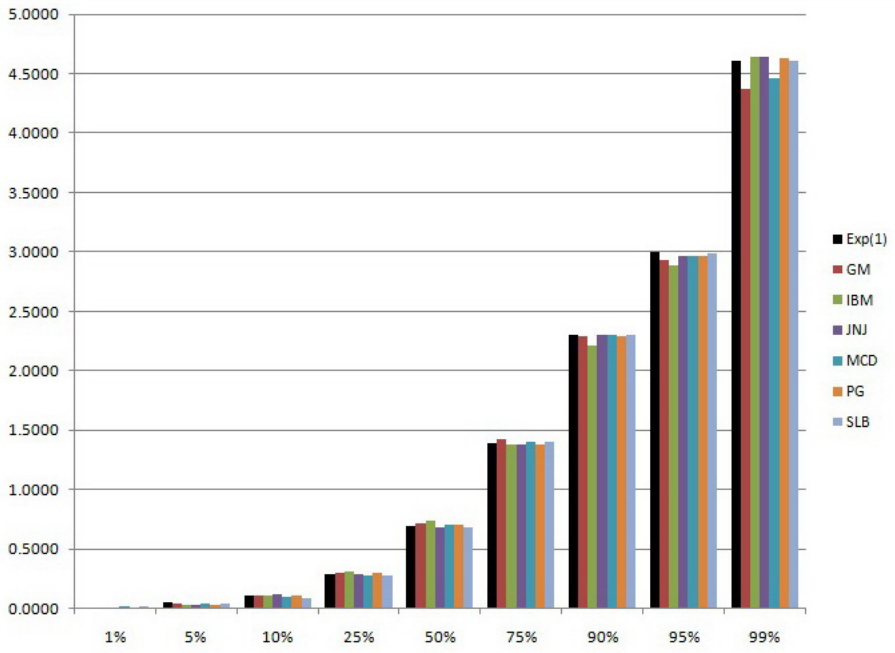

Figure 3.2: Quantiles of the Cox-Snell residuals and Exp(1) distribution.

considerations in the modelling of trade durations.

Chapter 4

Efficacy of optimization methods

In econometrics and time series analysis, there exist many optimization methods in use. The popular methods include the Berndt-Hall-Hall-Hausman (BHHH) algorithm [5], Nelder-Mead (NM), Broyden-Fletcher-Goldfarb-Shanno (BFGS) and biologically inspired genetic methods and simulated annealing.

In this chapter, in the context of the BS-ACD(1,1) model estimation, we discuss issues of numerical efficacy of the NM, BFGS algorithms, and a hybrid thereof. We shall try to point out weaknesses and strengths of the methods and make recommendations to researchers in empirical market microstructure as to which method to use depending on various criteria.

4.1 The NM method

The Nelder-Mead simplex method [38] belongs to the class of direct search algorithms. These algorithms are gradient-free and use values of the nonlinear

objective function $f : \mathbb{R}^n \to \mathbb{R}$ taken from a set of sample points and use that information to continue the sampling. In what follows, we shall give a brief description of the NM method as applied to a minimization problem. Obviously, maximization can be reduced to minimization by considering $-f$ as the new objective function.

The NM method maintains a simplex S of approximations to an optimal point. A simplex S in $\mathbb{R}^n$ is defined as the convex hull of $n+1$ vertices $\{x_j\}_{j=1}^{n+1}$, $(x_j \in \mathbb{R}^n)$. For example, a two-dimensional simplex is a triangle. The initial working simplex S has to be nondegenerate, i.e., the points $\{x_j\}_{j=1}^{n+1}$ must not lie in the same hyperplane.

In the algorithm the vertices are sorted according to the objective function values

$$f(x_1) \leqslant f(x_2) \leqslant \cdots \leqslant f(x_{n+1}),$$

where x_1 is called the best vertex and x_{n+1} the worst. If several vertices have the same objective value as x_1, the best vertex is not uniquely defined, but this ambiguity has little effect on the performance of the algorithm.

The method attempts to replace the worst vertex x_{n+1} with a new point of the form

$$x(\mu) = (1+\mu)\,\overline{x} - \mu x_{n+1},$$

where $\overline{x}$ is the centroid of the convex hull of $\{x_j\}_{j=1}^{n}$,

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

The value of μ is selected by rules that include reflection, expansion, outside and inside contraction, shrinking and sorting the vertices. The algorithm terminates if either $f(x_{n+1}) - f(x_1)$ is sufficiently small or a user-defined number of function evaluations has been expended.

The NM algorithm is not guaranteed to converge, even for smooth problems. However, the performance of the method in practice has proved to be generally good [30].

Despite the excellent reputation of the NM algorithm for robust high dimensional optimization without the use of derivative information, as applied to estimation of the BS-ACD(1,1) model, the R implementation of the algorithm from an initial value for β close to 1 (cf. [7]) repeatedly led to the termination of the algorithm due to a degenerate simplex or led to an estimate "far" from the true value. Although there are some documented instances where the NM algorithm has difficulties [3, 36, 50], this particular problem most likely arises from the behaviour of the likelihood function with respect to the sample data. When β is close to 1 or bigger than 1, the sequence of conditional medians σ_i tends to be increasing such that the log-likelihood function eventually explodes. Another reason for ill-behaving estimation by the NM method for particular initial conditions lies in that the log-likelihood function becomes discontinuous once a sample point X_i of large magnitude matches a conditional median σ_i of small magnitude.

Due to the above complications, convergence properties of the NM algorithm are difficult to study [30].

Remark 4.1 *The NM method is available through the R function optim().*

4.2 The BFGS method

Another popular optimization algorithm is due to Broyden, Fletcher, Goldfarb, and Shanno [11, 19, 24, 48]. The method belongs to the class of quasi-Newon methods. Quasi-Newton methods exploit user supplied gradient and update

an approximation of the Hessian matrix $\nabla^2 f(x^*)$ as the iteration progresses. In general, the transition from current approximations x_c and H_c of x^* and $\nabla^2 f(x^*)$ to new approximations x_+ and H_+ is given by the following steps:

1. Compute a search direction $d = -H_c^{-1}\nabla f(x_c)$.
2. Find $x_+ = x_c + \lambda d$ using a line search to insure sufficient decrease.
3. Use x_c, x_+, and H_c to update H_c and obtain H_+.

The way in which H_+ is computed determines the method. In particular, the BFGS method is a secant method because it satisfies the following (secant) equation:

$$H_+ s = y, \tag{4.1}$$

where

$$s = x_+ - x_c \text{ and } y = \nabla f(x_+) - \nabla f(x_c).$$

Specifically, the BFGS method's update is

$$H_+ = H_c + \frac{yy^T}{y^T s} - \frac{(H_c s)(H_c s)^T}{s^T H_c s}. \tag{4.2}$$

Given the displacement s and the change of gradients y, the secant equation requires that the symmetric positive definite matrix H_+ map s into y. This will be possible only if s and y satisfy the curvature condition:

$$s^T y > 0, \tag{4.3}$$

as can be easily seen by premultiplying (4.1) by s^T. When f is strongly convex, the inequality (4.3) will be satisfied for any two points x_+ and x_c. However, the condition will not always hold for nonconvex functions, and in this case (4.3) must be imposed explicitly.

It is reasonable to ask whether there are situations in which updating formula (4.2) can produce bad results. If at some iteration the matrix H_c becomes a very poor approximation, can one hope to correct it? A related question concerns the rounding errors that occur in finite-precision implementation of the BFGS algorithm: Can these errors grow to the point of erasing all useful information in the BFGS approximate matrix?

These and other questions have been studied analytically and experimentally, and it is known that the BFGS formula has very effective self-correcting properties. If the matrix H_c incorrectly estimates the curvature in the objective function, and if this bad estimate slows down the iteration, then the Hessian approximation will tend to correct itself within a few steps.

Remark 4.2 *The BFGS method is available through the R function optim().*

4.3 Numerical efficacy of the NM and BFGS methods

In Section 2.3, we introduced the two-stage estimation procedure. Recall that at the first step, we fix κ and estimate the triple (α, β, γ) by the NM algorithm. Having done this, the result is used as a starting value for the BFGS method which is applied to estimate $\boldsymbol{\theta} = (\alpha, \beta, \gamma, \kappa)$ over the whole parameter space Θ, where Θ is a compact subset of $\mathbb{R}^4$. We referred to the combination of the NM and BFGS algorithms as the NM-BFGS method.

For convenience, the problem of the BS-ACD(1,1) model estimation is restated below.

$$\sum_{i=1}^{N} \frac{1}{2\sqrt{2\pi}\kappa\sigma_i} \left[\left(\frac{\sigma_i}{X_i} \right)^{\frac{1}{2}} + \left(\frac{\sigma_i}{X_i} \right)^{\frac{3}{2}} \right] \exp \left\{ -\frac{1}{2\kappa^2} \left[\frac{X_i}{\sigma_i} + \frac{\sigma_i}{X_i} - 2 \right] \right\} \to \max_{\boldsymbol{\theta} \in \Theta},$$

where the conditional median is obtained recursively by

$$\ln \sigma_i = \alpha + \beta \ln \sigma_{i-1} + \gamma \left[\frac{X_{i-1}}{\sigma_{i-1}} \right].$$

We would like to compare and contrast the NM, NM-BFGS, and BFGS methods in their ability to estimate $\boldsymbol{\theta}$. Usually, the researcher needs to perform a large number of runs (ranging from 1000 to 1000000) in their simulations. Therefore, a time efficient method would be preferred. However, one could also ask about the accuracy of the estimation method. The accuracy can be characterized by the proximity of the estimates to the true value. Alternatively, the researcher may choose to work with the method which converges in the least amount of iterations/evaluations or/and gives the largest value of the maximum likelihood function.

Hence reasonable criteria for testing the performance of the three methods are the following:

1. CPU time (min),
2. Euclidean distance to the true value (min),
3. Number of evaluations (min),
4. Function value at the MLE (max),
5. Euclidean distance to the true value (min) and function value at the MLE (max),
6. Number of evaluations (min) and Euclidean distance to the true value (min) and function value at the MLE (max),

where in parentheses the desired objective is indicated.

Recall that throughout Section 2.3, the true value

$$\boldsymbol{\theta}^* \equiv (\alpha^*, \beta^*, \gamma^*, \kappa^*) = (0.1, 0.9, 0.1, 1.1) \tag{4.4}$$

was used. Based upon criteria 1-6, the efficacy of the NM, NM-BFGS, and BFGS methods is scrutinized by considering a 3-dimensional rectangular prism (also known as a cuboid) containing initial values $\widetilde{\boldsymbol{\theta}}_0 \equiv (\alpha_0, \beta_0, \gamma_0)$:

$$\begin{aligned}\mathcal{C}^\delta \equiv & \left(\alpha^*\left(1-\frac{\delta}{100}\right), \alpha^*\left(1+\frac{\delta}{100}\right)\right) \times \left(\beta^*\left(1-\frac{\delta}{100}\right), \beta^*\left(1+\frac{\delta}{100}\right)\right) \\ & \times \left(\gamma^*\left(1-\frac{\delta}{100}\right), \gamma^*\left(1+\frac{\delta}{100}\right)\right),\end{aligned}$$

where $\delta \in [0, 100]$. We will refer to $\delta_1 = \delta/100$ as the perturbation factor. Note that $\mathcal{C}$ is a Cartesian product of one-dimensional intervals. The prism can be viewed as a vicinity of $\widetilde{\boldsymbol{\theta}}^* \equiv (\alpha^*, \beta^*, \gamma^*)$. We chose δ to take up values 10, 50, and 90. Evidently, $\delta = 10$ and $\delta = 90$ represent extreme situations: $\mathcal{C}^{10}$ gives a prism of points located fairly close to $\widetilde{\boldsymbol{\theta}}^*$ whereas $\mathcal{C}^{90}$ is allowed to contain points at a bigger distance from $\widetilde{\boldsymbol{\theta}}^*$.

As noted in Remark 2.2, $|\beta| < 1$ should be imposed to maintain weak stationarity of the BS-ACD(1,1) model. Hence we modify the prism accordingly:

$$\begin{aligned}\mathcal{C}_M^\delta \equiv & \left(\alpha^*\left(1-\frac{\delta}{100}\right), \alpha^*\left(1+\frac{\delta}{100}\right)\right) \\ & \times \left(\max\left\{\beta^*\left(1-\frac{\delta}{100}\right), -1\right\}, \min\left\{\beta^*\left(1+\frac{\delta}{100}\right), 1\right\}\right) \\ & \times \left(\gamma^*\left(1-\frac{\delta}{100}\right), \gamma^*\left(1+\frac{\delta}{100}\right)\right).\end{aligned} \tag{4.5}$$

Note that preserving the logic of Section 2.3, the starting value for κ is constructed as

$$\sqrt{2\left(\frac{\overline{X}}{med(X)} - 1\right)}, \tag{4.6}$$

that is, independently of $\widetilde{\boldsymbol{\theta}}_0$.

Samples of sizes $N = 10000$ and 50000 were used for the efficacy study. The number of runs was $n = 1000$.

Then we test the performance of the three selected methods based upon the following algorithm:

1. Generate a sample of size N from BS-ACD(1,1) model with true values $\boldsymbol{\theta}^*$ from (4.4).

2. Draw an initial vector for (α, β, γ) at random from $\mathcal{C}_M^\delta$ for a fixed δ. For κ, take the initial value (4.6).

3. Obtain the MLE's $\left(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}, \widehat{\kappa}\right)$ by the NM, NM-BFGS, and the BFGS methods.

4. Repeat steps 1-3 $n = 1000$ times.

Then, in accordance with the criteria 1-6 suggested above, comparison of the three methods was performed. For brevity, let us denote $\overline{d}$, $\overline{\text{eval}}$, $\overline{time}$, respectively, the average Euclidean distance to the true value, average number of evaluations (of the function and its gradient altogether), and the average execution time (in seconds) per a single run. These characteristics are summarized in Table 4.1. The other tables were placed in Section B.2 of Appendix B.

One can observe that regardless of a sample size and the perturbation factor, the BFGS method is the fastest in terms of the cpu time.

			$\overline{d}$	$\overline{eval}$	$\overline{time}$
N=10000	$\delta = 10$	**NM**	0.0190	206	0.7019
		NM-BFGS	0.0190	163	0.5142
		BFGS	0.0189	97	0.2256
	$\delta = 50$	**NM**	0.0212	209	0.7220
		NM-BFGS	0.0187	181	0.5709
		BFGS	0.0187	114	0.2545
	$\delta = 90$	**NM**	0.0561	221	0.7974
		NM-BFGS	0.1333	210	0.6950
		BFGS	0.4048	137	0.3061
N=50000	$\delta = 10$	**NM**	0.0082	203	3.3685
		NM-BFGS	0.0082	190	2.7835
		BFGS	0.0082	105	1.1947
	$\delta = 50$	**NM**	0.0110	208	3.1173
		NM-BFGS	0.0083	205	2.7191
		BFGS	0.0083	122	1.2085
	$\delta = 90$	**NM**	0.0639	208	3.2499
		NM-BFGS	0.0092	219	3.0187
		BFGS	0.0728	152	1.4810

Table 4.1: Performance of the NM, NM-BFGS, and BFGS methods.

Remark 4.3 *It is interesting to note at this point that the speed of the BFGS method is in part due to the fact that the calculations are based upon the analytical gradient. Once a numerical approximation of the gradient is entertained, the situation is reversed: the NM method wins the first place whereas the BFGS algorithm goes last in the ranking. These calculations were not included in the thesis.*

As one could expect, the overall number of evaluations (of the log-likelihood function and its gradient) and the cpu time grow as δ_1 increases. On the other hand, although the same dynamics holds for the average Euclidean distance for $N = 10000$, the pattern breaks down for $N = 50000$. This is in part due to the fact the distances are positively skewed and a sample mean is not always an exhaustive measure of the central tendency. See Figure A.5 of Appendix A for an example of a histogram of distances.

If accuracy of the estimates is not of crucial concern for the researcher, then, as Table B.5 indicates, either of the NM, NM-BFGS or BFGS methods can be used. The table displays the number of times the MLE's produced by the methods coincide up to 2, 3, and 4 decimal places after the decimal point. However, only the BFGS and the NM-BFGS methods give estimates invariably close to each other up to the fourth decimal place.

It is not uncommon that the researcher may try various starting values and then select those which lead to estimates little different from each other. If this is the case and an initial value is close enough to the true value ($\delta_1 = 0.1$), then in terms of the distance the three methods are indistinguishable (see Figure A.5) and hence either method can be employed.

More interesting are the cases when the researcher does not possess a prior knowledge about the proximity of a starting value to the true value.

Consequently, with the lack of information, they might select a starting value from $\mathcal{C}_M^{50}$ or $\mathcal{C}_M^{90}$. To examine the stability of the methods in this situation, we resort to Tables B.3-B.4. The two tables count the number of times one method outperforms the other one in the pair in accordance with the six criteria on page 41.

It is easy to see that the NM algorithm loses in all six categories for samples of size $N = 10000$. For samples of size $N = 50000$, the NM method prevails over the NM-BFGS method in the number of evaluations ($\delta_1 \in \{0.5, 0.9\}$) and in the Euclidean distance ($\delta_1 = 0.5$). However, in terms the Euclidean distance to the true value an approximate parity remains among the estimates produced by all three methods.

Let us now turn to comparing the BFGS and the NM-BFGS algorithms. Out of the six criteria, the NM-BFGS method wins in only two: 4 and 5. However, the victory is convincing. In practice, criteria 4 and 5 serve the most informative indicators. However, if one's concern is the cpu time as well as the number of evaluations, the BFGS method is the indisputable leader. Between the BFGS and NM-BFGS methods, with respect to the ultimate criterion 6, the BFGS algorithm outperforms its competitor.

When the perturbation factor equals 0.5 or 0.9, it is worth pointing out that the NM-BFGS method behaves somewhat more reliably than the BFGS method alone. Once an initial value is drawn from either $\mathcal{C}_M^{50}$ or $\mathcal{C}_M^{90}$, the NM and the BFGS methods produce estimates which can be regarded as outliers. The number of outliers is higher than that produced by the NM-BFGS algorithm. In Figure A.6, histograms of the Euclidean distances to the true value $\boldsymbol{\theta}^*$ are depicted. The range for the distances is the least for the NM-BFGS method. As yet another evidence of the stable performance of the method, for example, consider the MLE's of κ. Figure A.7 displays the scatter

plot of the MLE's $\widehat{\kappa}$ produced by all three methods.

From the picture, we see that the degree of dispersion is the highest for the NM method. On the other hand, the NM-BFGS and BFGS methods produced only 2 and 1 outliers, respectively. That the BFGS algorithm gives only 1 outlier is misleading. From Figure A.8, it is clear that the true value is 6.

To summarize, in the context of the BS-ACD(1,1) model estimation, the following considerations are recommended to researchers in empirical market microstructure:

- The estimates provided by the three methods are virtually identical (up to 2 decimal places).
- With a prior information about a neighborhood around the true value $\boldsymbol{\theta}^*$, any of the three algorithms can be employed.
- In the lack of information, the BFGS method with the analytical gradient is preferred if one's concern is with execution time or accuracy of computations.
- In the lack of information, the NM-BFGS method with the analytical gradient is preferred if one wishes to achieve the combination of high accuracy in estimations and the largest value of the likelihood function.

Summary and open problems

Autoregressive conditional duration models play an important role in financial modelling. In this thesis, we have explored some properties of the BS-ACD(1,1) model.

Through extensive simulation experiments, we looked into the asymptotic behaviour of the ML estimators of the BS-ACD(1,1) model parameters. A dedicated test for normality was developed. With the numerical simulations and the test, we have demonstrated marginal asymptotic normality and consistency of the estimators.

We analyzed a real data example and demonstrated how the BS-ACD(1,1) model applies to data sets originating from intraday trading on the NYSE.

Finally, we studied the numerical efficacy of the Nelder-Mead and the BFGS methods and their combination in estimating the BS-ACD(1,1) model parameters. Conclusions have been made about the stability, speed, and accuracy of the algorithms.

As part of future research, it is important to derive regularity conditions ensuring joint asymptotic normality of the estimators in the BS-ACD(p,q) model. Also, weak and strong stationarity conditions must be established rigorously.

Out-of-sample forecasting ability of the model certainly needs a sepa-

rate study. Also, the forecasting power of the BS-ACD(p,q) model, possibly with $p > 1$ and $q > 1$, can be strengthened by including exogenous economic variables, such as intraday trading volume, prices, and bid-ask spread, into the equation governing the temporal dynamics of the conditional medians.

To generalize the BS-ACD(p,q) model, we suggest that a family of Generalized Birnbaum-Saunders (GBS) distributions [32] should be considered.

Definition 4.4 *Let a random variable Z follow a standard symmetrical distribution on $\mathbb{R}$ with the kernel $g(\cdot)$ of the pdf of Z, i.e. $Z \sim S(g)$. A random variable X is said to be $GBS(\kappa, \sigma; g)$-distributed if it admits the stochastic representation*

$$X = \frac{\sigma}{4}\left[\kappa Z + \sqrt{(\kappa Z)^2 + 4}\right]^2,$$

where

$$Z = \frac{1}{\kappa}\left(\sqrt{\frac{X}{\sigma}} - \sqrt{\frac{\sigma}{X}}\right) \sim S(g), \tag{4.7}$$

$\kappa > 0$ is the shape parameter and $\sigma > 0$ is the scale parameter. This is denoted by $X \sim GBS(\kappa, \sigma)$.

Standard symmetrical distributions on $\mathbb{R}$ include Gaussian, Cauchy, Laplace, Student t, Kotz-type, and others. The GBS distribution nests the classical BS distribution when $Z \sim N(0, 1)$ (also, cf. (2.2)). From the symmetry of the kernel and (4.7) it can be readily inferred that a GBS distribution can be parameterized in terms of the median. Hence we suggest that a $GBS - ACD(p, q; g)$ model should be developed to compare and contrast the performance of the new model with the BS-ACD(p,q) one.

Another direction for model generalizations can be developed by imposing the $X_i = \sigma_i \varrho_i$, where ϱ_i are positively supported i.i.d. random variables. We expect this assumption to simplify inference. As a particular case of such

extension, it is of interest to consider $\varrho_i = \exp(\varepsilon_i)$ with ε_i being i.i.d. random variables following the sinh-normal distribution $\mathrm{SN}(\kappa, 0, 2)$ [46], or, more generally, the sinh-spherical (SHS) distribution $SHS(\kappa, 0, 2; g)$ [32]. The $SN(\kappa, 0, 2)$ ($SHS(\kappa, 0, 2; g)$) distribution is symmetric with mean 0. It is related to the BS-distribution as follows: if $\varepsilon_i \sim SN(\kappa, 0, 2)$ ($SHS(\kappa, 0, 2; g)$) then $\exp(\varepsilon_i)$ $\sim BS(\kappa, 1)$ ($GBS(\kappa, 1; g)$) and hence $X_i \sim BS(\kappa, \sigma_i)$ ($GBS(\kappa, \sigma_i; g)$). However, in this case the QML estimation approach would have to be involved and all cautions discussed in Section 1.3 about the QML inference would have to be taken into account.

Apart from distributional assumptions, a possible generalization of the BS-ACD(p,q) model which we suggest is to employ an estimation procedure different from the ML or QML methods. To this end, estimating functions [41] and the method of modified moments [40] seem to be obvious candidates which have been demonstrated to enjoy a number of appealing statistical properties and ease of implementation.

Appendix A

Figures

A.1 Figures for Chapter 2

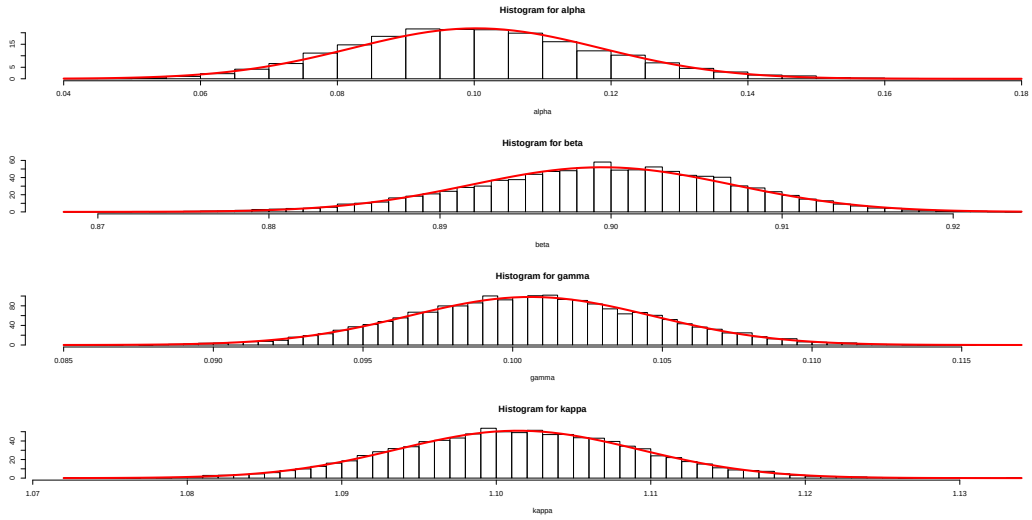

Figure A.1: Histograms of ML estimates (samples of size 10000, number of runs 10000) with the fitted density curve of a normal distribution.

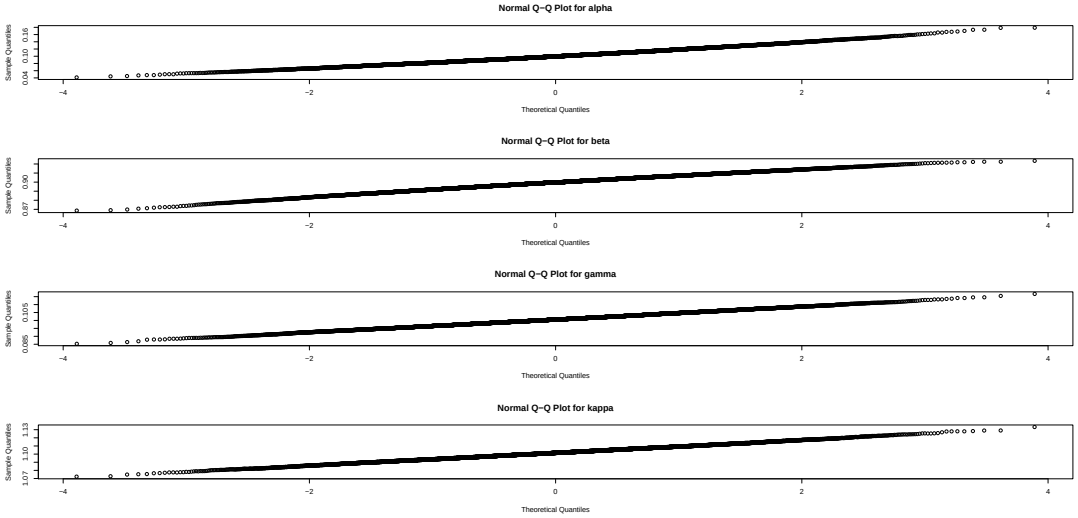

Figure A.2: Normal QQ plots of ML estimates (samples of size 10000, number of runs 10000).

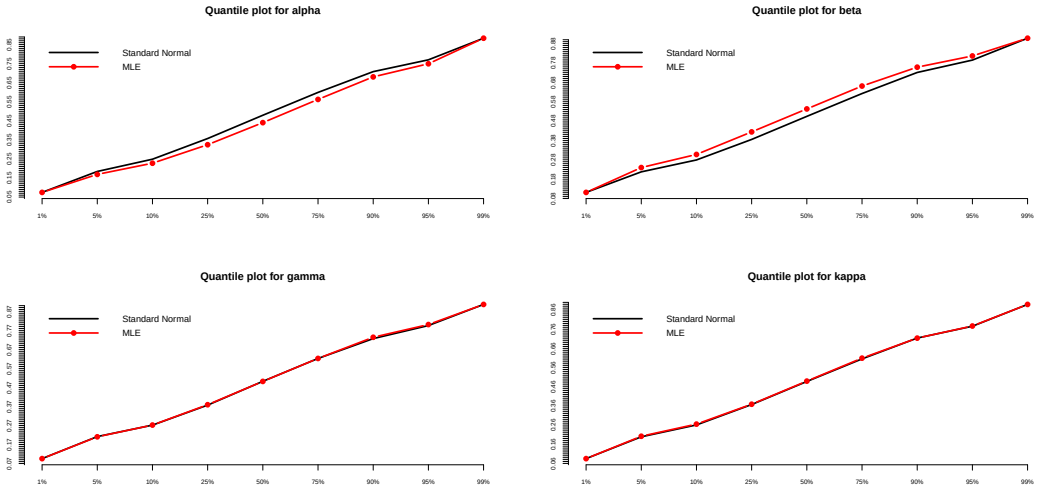

Figure A.3: Quantile plots of trimmed ML estimates (samples of size 10000, number of runs 10000). Trimming percentage 0.25%.

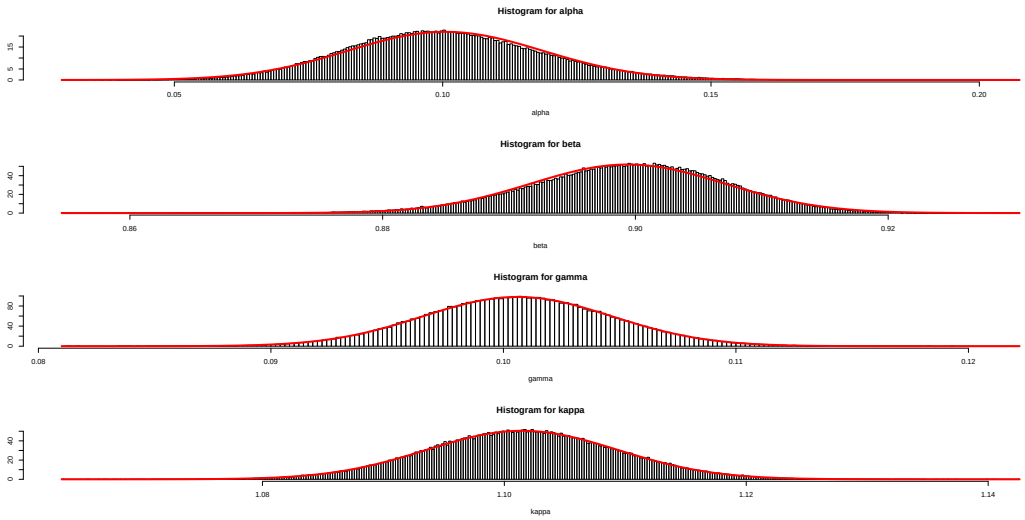

Figure A.4: Histograms of ML estimates (samples of size 10000, number of runs 1000000) with the fitted density curve of a normal distribution.

A.2 Figures for Chapter 4

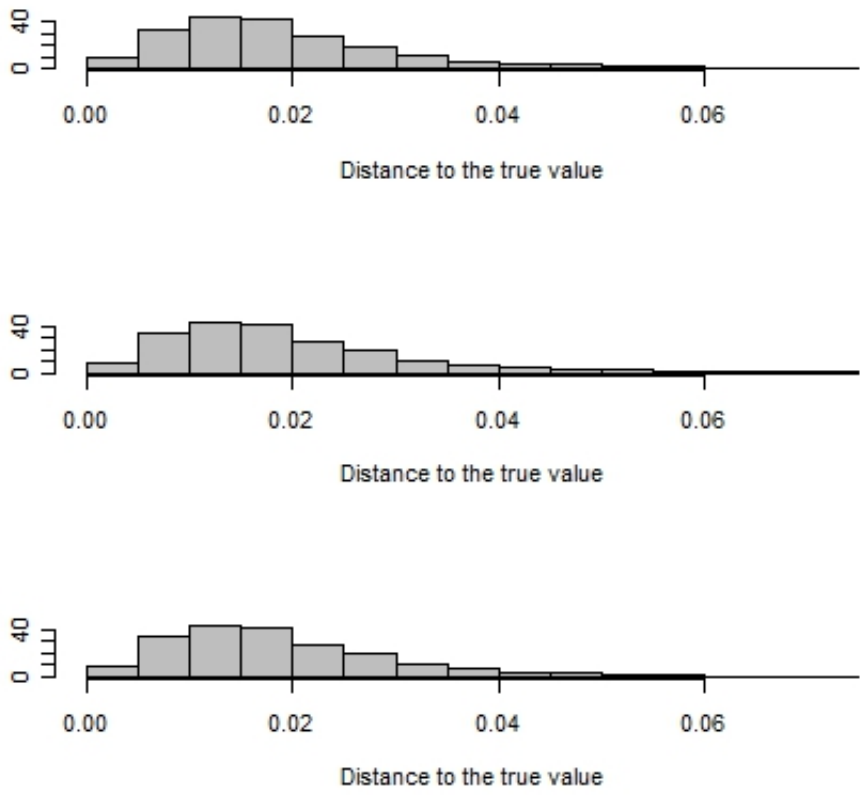

Figure A.5: Histograms of Euclidean distances to true values for samples of size 10000 and δ=10. From top to bottom: the NM, NM-BFGS, and BFGS methods.

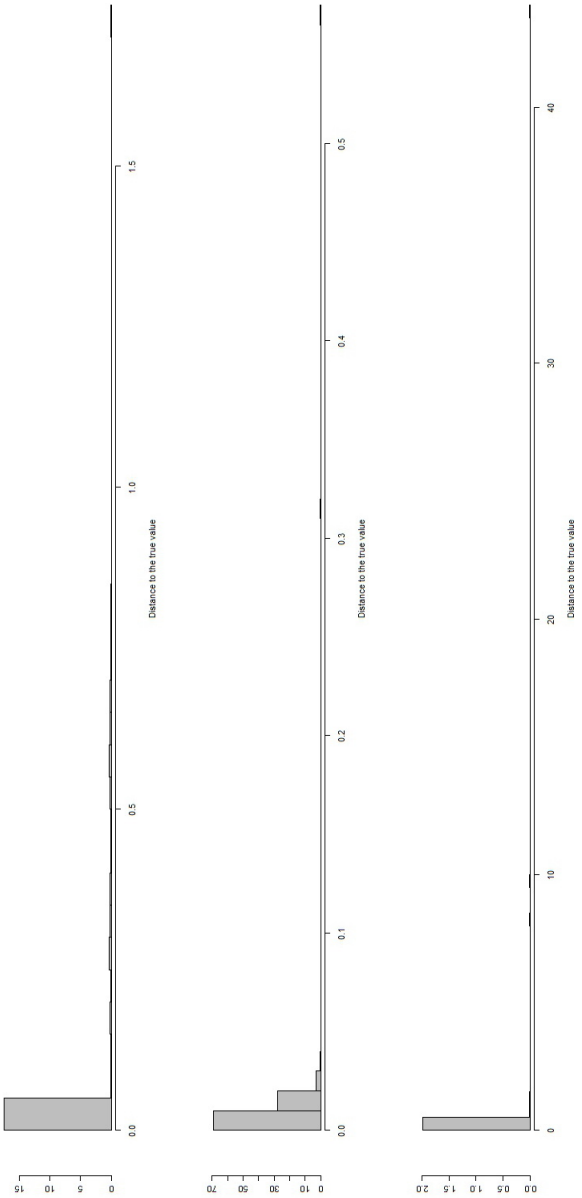

Figure A.6: Histograms of Euclidean distances to true values for samples of size 50000 and δ=90. From top to bottom: the NM, NM-BFGS, and BFGS methods.

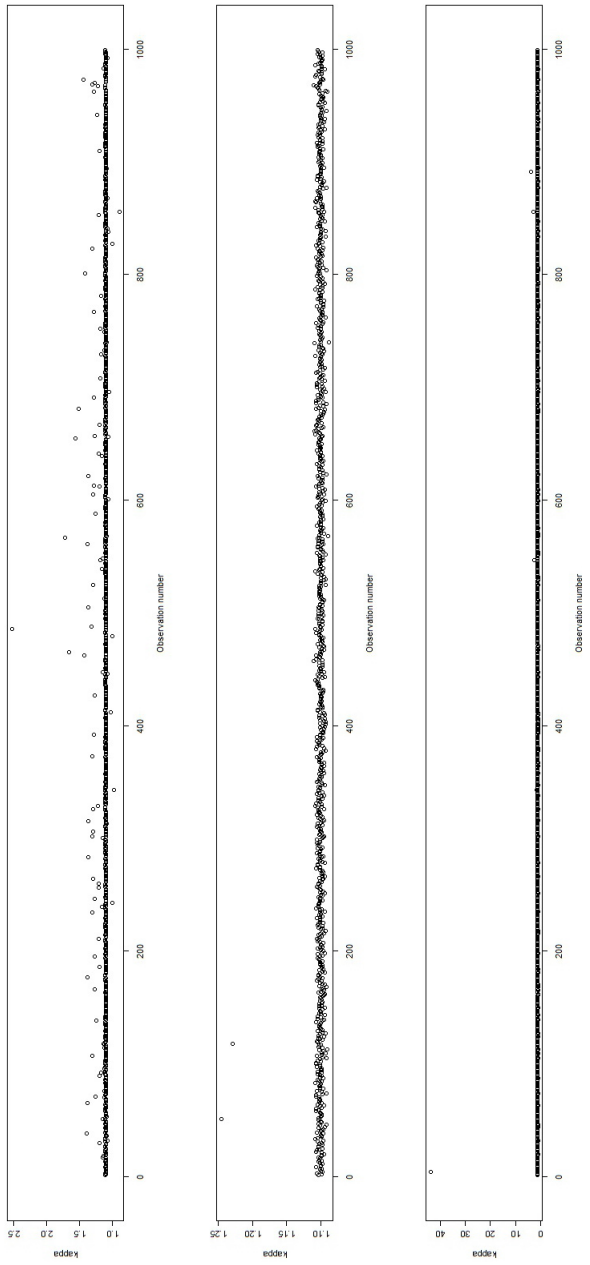

Figure A.7: Scatter plots of the kappa estimates for samples of size 50000 and δ=90. From top to bottom: the NM, NM-BFGS, and BFGS methods.

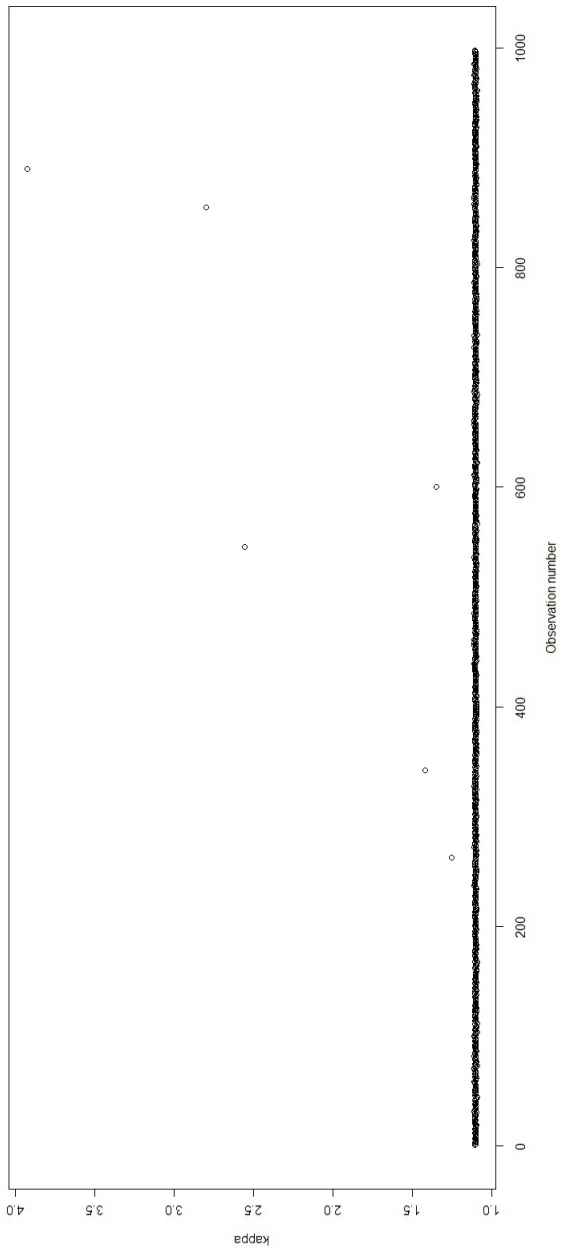

Figure A.8: Magnified scatter plot of the kappa estimates by the BFGS method for samples of size 50000 and δ=90.

Appendix B

Tables

B.1 Tables for Chapter 2

	n	10000				1000000
	N	10000	25000	50000	75000	10000
$\widehat{\alpha}$	True value	0.1000	0.1000	0.1000	0.1000	0.1000
	Mean	0.1003	0.1003	0.1002	0.1001	0.1005
	SD	0.0182	0.0114	0.0080	0.0066	0.0182
	Skewness	0.2600	0.1279	0.0936	0.1062	0.2565
	Kurtosis	3.1089	2.9553	3.0193	2.9516	3.1308
	Bias	0.0003	0.0003	0.0002	0.0001	0.0005
	MSE	0.0003	0.0001	0.0001	0.0000	0.0003
$\widehat{\beta}$	True value	0.9000	0.9000	0.9000	0.9000	0.9000
	Mean	0.8995	0.8997	0.8999	0.8999	0.8994
	SD	0.0077	0.0048	0.0034	0.0028	0.0077
	Skewness	-0.2164	-0.0862	-0.0817	-0.0765	-0.2195
	Kurtosis	3.0714	3.0224	3.0059	2.9371	3.1058
	Bias	-0.0005	-0.0003	-0.0001	-0.0001	-0.0006
	MSE	0.0001	0.0000	0.0000	0.0000	0.0001
$\widehat{\gamma}$	True value	0.1000	0.1000	0.1000	0.1000	0.1000
	Mean	0.1006	0.1002	0.1001	0.1001	0.1006
	SD	0.0041	0.0025	0.0018	0.0015	0.0041
	Skewness	0.0290	0.0326	0.0104	0.0417	0.0162
	Kurtosis	2.9679	2.9566	3.0047	2.9918	2.9950
	Bias	0.0006	0.0002	0.0001	0.0001	0.0006
	MSE	0.0000	0.0000	0.0000	0.0000	0.0000
$\widehat{\kappa}$	True value	1.1000	1.1000	1.1000	1.1000	1.1000
	Mean	1.1014	1.1006	1.1003	1.1002	1.1015
	SD	0.0078	0.0049	0.0035	0.0028	0.0079
	Skewness	0.0140	0.0714	0.0256	0.0194	0.0119
	Kurtosis	3.0270	2.9794	2.9850	3.0107	2.9750
	Bias	0.0014	0.0006	0.0003	0.0002	0.0015
	MSE	0.0001	0.0000	0.0000	0.0000	0.0001

Table B.1: ML estimates by Monte Carlo simulations.

	n	10000				1000000
	N	10000	25000	50000	75000	10000
$\widehat{\alpha}$	True value	0.1000	0.1000	0.1000	0.1000	0.1000
	Mean	0.1003	0.1003	0.1001	0.1000	0.1005
	SD	0.0178	0.0112	0.0078	0.0065	0.0178
	Skewness	0.2201	0.1100	0.0618	0.1015	0.2181
	Kurtosis	2.8246	2.7334	2.7380	2.7395	2.8320
	Bias	0.0003	0.0003	0.0001	0.0000	0.0005
	MSE	0.0003	0.0001	0.0001	0.0000	0.0003
$\widehat{\beta}$	True value	0.9000	0.9000	0.9000	0.9000	0.9000
	Mean	0.8995	0.8997	0.8999	0.8999	0.8994
	SD	0.0075	0.0047	0.0033	0.0027	0.0075
	Skewness	-0.1888	-0.0717	-0.0502	-0.0701	-0.1863
	Kurtosis	2.8201	2.7656	2.7311	2.7389	2.8172
	Bias	-0.0005	-0.0003	-0.0001	-0.0001	-0.0006
	MSE	0.0001	0.0000	0.0000	0.0000	0.0001
$\widehat{\gamma}$	True value	0.1000	0.1000	0.1000	0.1000	0.1000
	Mean	0.1006	0.1002	0.1001	0.1001	0.1006
	SD	0.0040	0.0025	0.0018	0.0014	0.0040
	Skewness	0.0241	0.0175	0.0073	0.0319	0.0154
	Kurtosis	2.7460	2.7349	2.7493	2.7829	2.7544
	Bias	0.0006	0.0002	0.0001	0.0001	0.0006
	MSE	0.0000	0.0000	0.0000	0.0000	0.0000
$\widehat{\kappa}$	True value	1.1000	1.1000	1.1000	1.1000	1.1000
	Mean	1.1014	1.1006	1.1003	1.1002	1.1015
	SD	0.0076	0.0048	0.0034	0.0028	0.0077
	Skewness	0.0086	0.0518	0.0183	0.0162	0.0119
	Kurtosis	2.7859	2.7415	2.7525	2.7716	2.7486
	Bias	0.0014	0.0006	0.0003	0.0002	0.0015
	MSE	0.0001	0.0000	0.0000	0.0000	0.0001

Table B.2: Trimmed ML estimates by Monte Carlo simulations. Trimming percentage 0.25%.

B.2 Tables for Chapter 4

Remark B.1 *In Tables B.3-B.4, columns are enumerated from 1 to 6 according to the following conventions:*

1. *CPU time (min),*
2. *Euclidean distance to the true value (min),*
3. *Number of evaluations (min),*
4. *Function value at the MLE (max),*
5. *Euclidean distance to the true value (min) and function value at the MLE (max),*
6. *Number of evaluations (min) and Euclidean distance to the true value (min) and function value at the MLE (max),*

where in parentheses the desired objective is indicated.

		1	**2**	**3**	**4**	**5**	**6**
$\delta = 10$	**NM**	172	484	247	35	17	8
	NM-BFGS	815	514	745	963	496	377
	NM	3	478	38	135	56	5
	BFGS	995	520	958	863	441	422
	NM-BFGS	19	447	74	733	329	23
	BFGS	979	551	919	261	144	125
$\delta = 50$	**NM**	249	477	335	37	16	9
	NM-BFGS	743	521	656	961	500	333
	NM	14	475	58	135	68	3
	BFGS	983	523	939	863	456	422
	NM-BFGS	16	489	59	712	363	24
	BFGS	980	509	938	285	159	149
$\delta = 90$	**NM**	352	446	437	35	16	14
	NM-BFGS	636	550	556	960	531	286
	NM	63	451	139	97	50	3
	BFGS	931	545	857	899	498	390
	NM-BFGS	43	503	115	674	338	30
	BFGS	952	493	876	316	154	126

Table B.3: Counts: Pairwise comparisons for samples of size 10000.

		1	**2**	**3**	**4**	**5**	**6**
$\delta = 10$	**NM**	289	465	436	110	58	42
	NM-BFGS	710	535	560	890	483	288
	NM	6	469	48	134	60	0
	BFGS	994	531	951	866	457	437
	NM-BFGS	16	496	61	587	283	20
	BFGS	984	504	936	412	200	187
$\delta = 50$	**NM**	368	504	518	122	67	47
	NM-BFGS	626	494	476	876	439	221
	NM	19	493	79	130	64	6
	BFGS	979	505	916	868	439	393
	NM-BFGS	15	502	55	560	281	20
	BFGS	983	496	942	437	217	201
$\delta = 90$	**NM**	443	403	550	92	45	31
	NM-BFGS	555	596	445	907	549	225
	NM	122	412	203	105	56	8
	BFGS	877	587	794	893	537	367
	NM-BFGS	74	518	178	574	307	35
	BFGS	925	481	821	424	214	164

Table B.4: Counts: Pairwise comparisons for samples of size 50000.

				m		
		Method 1	**Method 2**	**2**	**3**	**4**
N=10000	$\delta = 10$	**NM**	**NM-BFGS**	998	967	220
		NM	**BFGS**	998	942	188
		NM-BFGS	**BFGS**	998	973	716
	$\delta = 50$	**NM**	**NM-BFGS**	984	956	214
		NM	**BFGS**	984	921	183
		NM-BFGS	**BFGS**	998	968	716
	$\delta = 90$	**NM**	**NM-BFGS**	909	828	151
		NM	**BFGS**	909	811	132
		NM-BFGS	**BFGS**	989	960	703
N=50000	$\delta = 10$	**NM**	**NM-BFGS**	998	963	202
		NM	**BFGS**	998	940	173
		NM-BFGS	**BFGS**	1000	976	635
	$\delta = 50$	**NM**	**NM-BFGS**	982	944	177
		NM	**BFGS**	982	931	178
		NM-BFGS	**BFGS**	998	981	633
	$\delta = 90$	**NM**	**NM-BFGS**	864	780	122
		NM	**BFGS**	863	752	113
		NM-BFGS	**BFGS**	990	955	623

Table B.5: Counts: Coincidence of estimates up to 2, 3, and 4 decimal places.

Appendix C

C and R codes for Chapter 2

C.1 C codes for Section 2.3

```
/*************************************************************/
/*              Simulating BS-ACD(1,1) samples               */
/*************************************************************/
#include <R.h>
#include <Rmath.h>
void bssample(double *param, long *n, double *result) {
 long i; double z,zstar[*n+1],lns0,lns,bs[*n],a,b,g,k;
 a=param[0]; b=param[1]; g=param[2]; k=param[3];
 GetRNGstate();
 for(i=0;i<*n+1;i++){
    z=rnorm(0,1);
    zstar[i] =pow(k*z/2.0
                +sqrt(pow(k*z/2.0,2.0)+1.0),2.0);
 }
```

```
 PutRNGstate();
 lns0=0.0; result[0]=0.0; bs[0]=0.0;
 for(i=1;i<*n+1;i++){
    lns = a+b*lns0+g*zstar[i-1];
    bs[i-1] = exp(lns)*zstar[i];
    lns0=lns; result[i-1]=bs[i-1];
 }
}
/*************************************************************/
/*              Computing the conditional medians            */
/*************************************************************/
#include <R.h>
#include <Rmath.h>
void sigma_seq(double *param, double *ts, long *nts,
    double *sigmainit, double *result){
 long i; double lns_old, lns_new, s[*nts], a,b,g;
 s[0]=*sigmainit; lns_old=log(s[0]);
 result[0]=s[0];
 a=param[0]; b=param[1]; g=param[2];
 for(i=1; i < *nts; i++) {
    lns_new=a+b*lns_old+g*ts[i-1]/s[i-1];
    s[i]=exp(lns_new); lns_old=lns_new; result[i]=s[i];
 }
}
/*************************************************************/
/*              Computing the log-ML function                */
/*************************************************************/
```

```
#include <R.h>
#include <Rmath.h>
void MaxL(double *kk, double *ts, long *nts, double *s,
    double *result) {
 long i; double logML=0.0;
 for(i=0; i < *nts; i++) {
    logML=logML-log(*kk)-log(s[i])+log(sqrt(s[i]/ts[i])
            +pow(s[i]/ts[i],1.5))-0.5/((*kk)*(*kk))
            *(ts[i]/s[i]+s[i]/ts[i]-2.0);
 }
 *result=logML;
}
/*************************************************************/
/*     Computing the gradient of conditional medians       */
/*************************************************************/
#include<R.h>
#include<Rmath.h>
void gradSigma(double *param, double *ts, long *nts,
    double *s, double *dsdpar) {
 long i; double a,b,g;
 a=param[0]; b=param[1]; g=param[2];
 dsdpar[0]=0;dsdpar[1]=0;dsdpar[2]=0;
 dsdpar[3]=s[1];dsdpar[4]=s[1]*log(s[0]);
 dsdpar[5]=s[1]*ts[0]/s[0];
 for(i=2; i < *nts; i++) {
    dsdpar[(3*i)]=s[i]*(1.0+(1.0/s[i-1])
                    *dsdpar[(3*i-3)]*(b-g*ts[i-1]/s[i-1]));
```

```
    dsdpar[(3*i+1)]=s[i]*(log(s[i-1])+(1.0/s[i-1])
                    *dsdpar[(3*i-2)]*(b-g*ts[i-1]/s[i-1]));
    dsdpar[(3*i+2)]=(s[i]/s[i-1])*(ts[i-1]+dsdpar[(3*i-1)]
                    *(b-g*ts[i-1]/s[i-1]));
  }
}
/*************************************************************/
/*     Computing the gradient of the log-ML function       */
/*************************************************************/
#include <R.h>
#include <Rmath.h>
void gradML(double *k, double *ts, long *nts, double *s,
    double *dsdpar, double *gr) {
 long i; double dfds;
 gr[0]=0.0;gr[1]=0.0;gr[2]=0.0;gr[3]=0.0;
 for(i=0; i < *nts; i++) {
    dfds=-0.5*(ts[i]-s[i])*(-ts[i]*ts[i]-2*ts[i]*s[i]
        +s[i]*ts[i]*(*k)*(*k)-s[i]*s[i])/(s[i]*s[i]
        *ts[i]*(ts[i]+s[i])*(*k)*(*k));
    gr[0]=gr[0]+dfds*dsdpar[3*i];
    gr[1]=gr[1]+dfds*dsdpar[3*i+1];
    gr[2]=gr[2]+dfds*dsdpar[3*i+2];
    gr[3]=gr[3]-(s[i]*ts[i]*(*k)*(*k)-ts[i]*ts[i]
        -s[i]*s[i]+2*ts[i]*s[i])/((*k)*(*k)*(*k)
        *ts[i]*s[i]);
  }
}
```

```
/*********************************************************/
```

C.2 R codes for Section 2.3

```
#################################################################
#                    R wrappers for C codes                     #
#################################################################
# Load shared objects
dyn.load("BSnumGen.so")
dyn.load("Sigma.so")
dyn.load("MLC.so")
dyn.load("gradSigma.so")
dyn.load("gradML.so")
# Wrapper for generating BS-ACD(1,1) samples
BS.C <- function(param,nn){
    bs <- .C("bssample",as.double(param),as.integer(nn),
        result=double(nn))
    bs[["result"]]
}
# Wrapper for conditional medians
sig.C <- function(param,ts){
    nn <-length(ts)
    s_init<-init(ts)$sigma.init
    sig <- .C("sigma_seq",as.double(param),as.double(ts),
    as.integer(nn),as.double(s_init),result=double(length(ts)))
    sig[["result"]]
}
```

```
# Wrapper for the log-ML function to be used by
# the Nelder-Mead method
ML.C2 <- function(param3, kk,ts){
    s<-sig.C(param3[1:3],ts)
    nn<-length(ts)
MaxLC <- .C("MaxL",as.double(kk),as.double(ts),
            as.integer(nn),as.double(s),result=double(1))
    MaxLC[["result"]]
}
# Wrapper for the log-ML function to be used by
# the BFGS method
ML.C <- function(param4, ts){
    kk<-param4[4]
    s<-sig.C(param4[1:3],ts)
    nn<-length(ts)
MaxLC <- .C("MaxL",as.double(kk),as.double(ts),
            as.integer(nn),as.double(s),result=double(1))
    MaxLC[["result"]]
}
# Wrapper for the gradient of conditional medians
GradSig.C <- function(param,ts,s){
    nn <-length(ts)
    GradSig <- .C("gradSigma",as.double(param),
            as.double(ts),as.integer(nn),as.double(s),
            result=double(3*nn))
    GradSig[["result"]]
}
```

```
# Wrapper for the gradient of the log-ML function
GradML.C <- function(param,ts){
    nn <-length(ts)
    kk<-param[4]
    s<-sig.C(param[1:3],ts)
    dsdpar<-GradSig.C(param[1:3],ts,s)
    GradML <- .C("gradML",as.double(kk),as.double(ts),
            as.integer(nn),as.double(s),as.double(dsdpar),
            result=double(4))
    GradML[["result"]]
}
#################################################################
#        Initialization estimate for kappa and sigma            #
#################################################################
init = function(ts){
    k.init=sqrt(2*(mean(ts)/median(ts)-1))
    sigma.init=median(ts)
    return(list(k.init = k.init, sigma.init = sigma.init))
}
#################################################################
# Main Function for ML estimation of the BS-ACD(1,1) model      #
#################################################################
SimNMBF <- function(param,iniest,samsize,simnum,start){
    coef.NMBF<-matrix(rep(NA,19*simnum),ncol=19)
    for (j in start:simnum){
        # Generate a BS-ACD(1,1) sample
        sam<-BS.C(param,samsize)
```

```
        # Calculate descriptive statistics
        d.stat<-c(mean(sam),median(sam),var(sam))
        # Starting value of the estimate
        est0<-c(iniest,init(sam)$k.init)
        t0<-proc.time()
        coef.NMBF[j,1:15]<-Opt(1,param,est0,sam)
        coef.NMBF[j,16:18]<-d.stat
        coef.NMBF[j,19]<-proc.time()-t0
   }
    return(coef.NMBF)
}
#################################################################
#           Realization of three optimization methods       #
#################################################################
Opt<-function(method,param,est0,ts){
   # Values for method:
   # 0 for Nelder-Mead alone;
   # 1 for Nelder-Mead and BFGS;
   # 2 for BFGS alone;
    if (method==0){
        vec<-rep(NA,12)
        p<-optim(est0,fn=ML.C,gr=NULL,ts=ts,
                method ="Nelder-Mead",
                control = list(maxit=5000,fnscale=-1),
                hessian = FALSE)
        vec[1:8]<-c(p$par,est0)
        vec[9]<-p$value
```

```
    vec[10]<-sqrt(sum((param-p$par)^2))
    vec[11]<-p$counts[1];vec[12]<-p$convergence
}
if (method==1){
    vec<-rep(NA,15)
    p<-optim(est0[1:3],fn=ML.C2,gr=NULL,
            kk=est0[4],ts=ts,
            method ="Nelder-Mead",
            control = list(maxit=5000,fnscale=-1),
            hessian = FALSE)
    q<-optim(c(p$par,est0[4]),fn=ML.C,gr=GradML.C,
            ts=ts,method ="BFGS",
            control = list(maxit=5000,fnscale=-1),
            hessian = FALSE)
    vec[1:8]<-c(q$par,est0)
    vec[9]<-q$value
    vec[10]<-sqrt(sum((param-q$par)^2))
    vec[11]<-p$counts[1];vec[12]<-q$counts[1];
    vec[13]<-q$counts[2]
    vec[14]<-p$convergence;vec[15]<-q$convergence
}
if (method==2){
    vec<-rep(NA,13)
    q<-optim(est0,fn=ML.C,gr=GradML.C,ts=ts,
            method ="BFGS",
            control = list(maxit=5000,fnscale=-1),
            hessian = FALSE)
```

```
        vec[1:8]<-c(q$par,est0);vec[9]<-q$value
        vec[10]<-sqrt(sum((param-q$par)^2))
        vec[11]<-q$counts[1];vec[12]<-q$counts[2];
        vec[13]<-q$convergence
    }
    return(vec)
}
#################################################################
```

C.3 R codes for Section 2.4

```
#################################################################
#     Computation of skewness, kurtosis, bias, and MSE      #
#################################################################
skew<-function(ts){
    n<-length(ts)
    sk<-sqrt(n*(n-1))/(n-2)*mean((ts-mean(ts))^3)/(sd(ts))^3
    return(sk)
}
kurt<-function(ts){
    n<-length(ts)
    kur<-mean((ts-mean(ts))^4)/(sd(ts))^4
    return(kur)
}
bias<-function(trueparam,ts){
    b<-mean(ts)-trueparam
    return(b)
```

```
}
MSE<-function(trueparam,ts){
    ms<-mean((ts-trueparam)^2)
    return(ms)
}
#################################################################
#             Summary of the statistics of interest             #
#################################################################
DescrStats <- function(trueparam,ts,k){# ts is a matrix
    coef<-matrix(rep(NA,6*4),ncol=4)
    coef[1,]<-c(trueparam[1],trueparam[2],
            trueparam[3],trueparam[4]) #true parameters
    coef[2,]<-apply(ts,2,mean) #mean
    coef[3,]<-apply(ts,2,skew) #skewness
    coef[4,]<-apply(ts,2,kurt) #kurtosis
    coef[5,]<-c(bias(trueparam[1],ts[,1]),bias(trueparam[2],
            ts[,2]),bias(trueparam[3],ts[,3]),
            bias(trueparam[4],ts[,4])) #bias
    coef[6,]<-c(MSE(trueparam[1],ts[,1]),MSE(trueparam[2],
            ts[,2]),MSE(trueparam[3],ts[,3]),MSE(trueparam[4],
            ts[,4])) #MSE
    return(round(coef,k))
}
#################################################################
#       Quantile plots of trimmed MLE's and N(0,1) r.v.         #
#################################################################
QPlots<-function(param,simnum,trim,sort=TRUE, std=TRUE,var){
```

```
if (var==1){letter<-"alpha"};if (var==2){letter<-"beta"}
if (var==3){letter<-"gamma"};if (var==4){letter<-"kappa"}
z<-rnorm(simnum);z<-sort(z);
lb<-floor(simnum*trim/100);rb<-simnum-lb+1
z<-z[-union(1:lb,rb:simnum)]
z<-(z-min(z))/(max(z)-min(z))
if (sort==TRUE){p<-sort(param)} else {p<-param}
if (std==TRUE){p<-(p-mean(p))/sd(p)}
p<-p[-union(1:lb,rb:simnum)];
p<-(p-min(p))/(max(p)-min(p))
qn<-quantile(z,probs=c(0.01,0.05,0.10,0.25,
    0.50,0.75,0.90,0.95,0.99))
q<-quantile(p,probs=c(0.01,0.05,0.10,0.25,
    0.50,0.75,0.90,0.95,0.99))
yrange<-max(max(qn),max(q));vec<-c(1,5,10,25,
            50,75,90,95,99)
L<-paste(vec,"%",sep="")
plot(seq(1,9), qn,type="l",col="black",xlab="",
    ylab="",lwd=2.5,axes=F)
par(new=TRUE)
plot(seq(1,9), q,type="b",pch=19,col="red",xlab="",
    ylab="",axes=F,lwd=2.5,
    main=paste("Quantile plot for ",letter,sep=""))
legend(1.0,max(q),c("Standard Normal","MLE"),bty="n",
    lty=c(1,1),lwd=c(2.5,2.5),pch=c(-1,19),
    col=c("black","red"))
axis(1, at=seq(1,9), labels = L, cex.axis = 0.75)
```

```
    axis(2, at=seq(0,yrange,by=0.01),cex.axis = 0.75)
}
#################################################################
#      Distance between quantile curves of trimmed MLE's        #
#      and N(0,1) r.v.                                          #
#################################################################
QDist<-function(est,simnum,trim,std=FALSE){
    z<-rnorm(simnum);z<-sort(z);
    lb<-floor(simnum*trim/100);rb<-simnum-lb+1;
    z<-z[-union(1:lb,rb:simnum)]
    z<-(z-min(z))/(max(z)-min(z))
    est1<-apply(est[,1:4],2,sort)
    if (std==TRUE){est1<-apply(est1[,1:4],2,
        function(x) (x-mean(x))/sd(x))}
    est1<-est1[-union(1:lb,rb:simnum),]
    est1<-apply(est1,2,function(x)
        (x-min(x))/(max(x)-min(x)))
    qn<-quantile(z,probs=seq(0,1,by=0.01))
    q<-apply(est1,2,function(x)
        quantile(x,probs=seq(0,1,by=0.01)))
    dist<-apply(q,2,function(x) max(abs(x-qn)))
return(dist)
}
# Critical values of the test statistic.
QDistCrit<-function(dist){
    critdist<-apply(dist,2,function(x)
    quantile(x,probs=c(0.90,0.95,0.99)))
```

```
return(critdist)
}
#################################################################
#   1000 runs of the CVM, JB, and DA tests for normality    #
#################################################################
library(fBasics) # For the JB and DA tests
library(nortest) # For the CVM test
x <- replicate(1000,{ # generates 1000 different tests on
        each distribution
  c(cvm.test(rnorm(1000)+c(1,0,2,0,1))$p.value,
    cvm.test(rnorm(5000)+c(1,0,2,0,1))$p.value,
    cvm.test(rnorm(10000)+c(1,0,2,0,1))$p.value,
    cvm.test(rnorm(25000)+c(1,0,2,0,1))$p.value,
    cvm.test(rnorm(50000)+c(1,0,2,0,1))$p.value,
    cvm.test(rnorm(75000)+c(1,0,2,0,1))$p.value,
    cvm.test(rnorm(100000)+c(1,0,2,0,1))$p.value)
  }
)
rowMeans(x<0.05) # the proportion of significant deviations
y <- replicate(1000,{
  c(jarqueberaTest(rnorm(1000)+c(1,0,2,0,1))@test$p.value,
    jarqueberaTest(rnorm(5000)+c(1,0,2,0,1))@test$p.value,
    jarqueberaTest(rnorm(10000)+c(1,0,2,0,1))@test$p.value,
    jarqueberaTest(rnorm(25000)+c(1,0,2,0,1))@test$p.value,
    jarqueberaTest(rnorm(50000)+c(1,0,2,0,1))@test$p.value,
    jarqueberaTest(rnorm(75000)+c(1,0,2,0,1))@test$p.value,
    jarqueberaTest(rnorm(100000)+c(1,0,2,0,1))@test$p.value)
```

```
  }
)
rowMeans(y<0.05)
z <- replicate(1000,{
  c(dagoTest(rnorm(1000)+c(1,0,2,0,1))@test$p.value[1],
    dagoTest(rnorm(5000)+c(1,0,2,0,1))@test$p.value[1],
    dagoTest(rnorm(10000)+c(1,0,2,0,1))@test$p.value[1],
    dagoTest(rnorm(25000)+c(1,0,2,0,1))@test$p.value[1],
    dagoTest(rnorm(50000)+c(1,0,2,0,1))@test$p.value[1],
    dagoTest(rnorm(75000)+c(1,0,2,0,1))@test$p.value[1],
    dagoTest(rnorm(100000)+c(1,0,2,0,1))@test$p.value[1])
  }
)
rowMeans(z<0.05) # the proportion of significant deviations
#################################################################
```

Appendix D

C and R codes for Chapter 3

D.1 C codes for Section 3.2

```
/*************************************************************/
/*           Differences for diurnal pattern removal       */
/*************************************************************/
#include <R.h>
#include <Rmath.h>
void differ(long *node, long *times, long *ntimes,
    long *result){
 int j;long k,i=0;
 for (j=0;j<13;j++){
    for(k=1; k < *ntimes; k++){
        if (times[k]-node[j]>0){
            result[i]=times[k]-node[j];
        }
        else{
```

```
            result[i]=0;
        }
        i=i+1;
    }
 }
}
/*************************************************************/
/*    Removal of trades taken place between 09:30-09:59    */
/*************************************************************/
#include <R.h>
#include <Rmath.h>
void truncate(long *times, long *ntimes, double *duradj,
    double *result){
 long k;
 for(k=0; k < *ntimes; k++){
    if (times[k]>=36000){
        result[k]=duradj[k];
    }
 }
}
/*************************************************************/
```

D.2 R codes for Section 3.2

```
#################################################################
#                   R wrappers for C codes                      #
#################################################################
```

```
# Load shared objects
dyn.load("Season.so")
dyn.load("Trunc.so")
dyn.load("Sigma.so")
dyn.load("MLC.so")
dyn.load("gradSigma.so")
dyn.load("gradML.so")
# Wrapper for differences
diff.C <- function(nodes,times){
    n <-length(times)
    m <-(n-1)*13
    dif <- .C("differ",as.integer(nodes),as.integer(times),
            as.integer(n),result=integer(m))
    dif[["result"]]
}
# Wrapper for trimming trades
trunc.C <- function(data){
    dataset<-data[-1,]
    n <-dim(dataset)[1]
    tr <- .C("truncate",as.integer(dataset$TIME),
            as.integer(n),as.double(dataset$X_adj),
            result=double(n))
    tr[["result"]]
}
# Wrapper for the log-ML function to be used by
# the BFGS method
ML.C <- function(param, dataset){
```

```
    ts<-dataset$X_trunc[dataset$X_trunc!=0]
    s<-sigma.real(param[1:3],dataset)
    nn<-length(ts); kk<-param[4]
MaxLC <- .C("MaxL",as.double(kk),as.double(ts),
        as.integer(nn),as.double(s),
        result=double(length(kk)))
MaxLC[["result"]]
}
# Wrapper for the log-ML function to be used by
# the Nelder-Mead method
ML.C2<- function(param,kk,dataset){
    ts<-dataset$X_trunc[dataset$X_trunc!=0]
s<-sigma.real(param,dataset)
    nn<-length(ts)
    MaxLC <- .C("MaxL",as.double(kk),as.double(ts),
        as.integer(nn),as.double(s),
        result=double(length(kk)))
    MaxLC[["result"]]
}
# Wrapper for the gradient of conditional medians
GradSig.C <- function(param,dataset,s){
    ts<-dataset$X_trunc[dataset$X_trunc!=0]
    nn <-length(ts)
    GradSig <- .C("gradSigma",as.double(param),
            as.double(ts),as.integer(nn),as.double(s),
            result=double(3*nn))
    GradSig[["result"]]
```

```
}
# Wrapper for the gradient of the log-ML function
GradML.C <- function(param,dataset){
    ts<-dataset$X_trunc[dataset$X_trunc!=0]
    nn <-length(ts)
    kk<-param[4]
    s<-sigma.real(param[1:3],dataset)
    dsdpar<-GradSig.C(param[1:3],dataset,s)
    GradML <- .C("gradML",as.double(kk),as.double(ts),
            as.integer(nn),as.double(s),as.double(dsdpar),
            result=double(4))
    GradML[["result"]]
}
#################################################################
#                     Description of data                       #
#################################################################
#source("data_read.R")
#mydata[1,]
#   DATE    TIME  X  vol  total_valu ave_price hour half open
#2002-01-02 35420 13 1000    48620     48.62    9    9    1
# DATE = date
# TIME = time from midnight in seconds
# X = duration between trades in seconds
# total_valu = total value of trade (shares * price)
# ave_price = average price
# hour = trading hour
# half = h.1 for first half hour and h.2 for second
```

```
#       half hour for hours 10-15
# open = 0/1 opening trade of that trading day
#
# Note: total_valu is computed from compressed trades, i.e.
# multiple trades occurring at the same time are aggregated
# in volume, total value and used to compute an average price
# at that point in time
#################################################################
#                   Removal of diurnal pattern                  #
#################################################################
# Load useful libraries
# For importing datasets:
library(foreign)
# For accurate computation of Hessian matrix
library(numDeriv)

# Function to fit regression
fitting<-function(data){
    counter<-1
dat.lab <- paste("data$dat", 1:13, "[-1]",sep="")
fmla <- as.formula(paste("data$log_dur[-1] ~ ",
            paste(dat.lab, collapse= "+")))
fitt<-lm(fmla)
summ<-summary(fitt)$coefficients[-1,4]
num<-which(summ<0.05);v<-length(summ)
if ((v==length(num))||(length(num)==0)){
        return(list(fit=fitted(fitt),count=counter,
```

```
        summary=summary(fitt)))
    }
else{
        continue<-TRUE
        while(continue){
dat.lab <- paste("data$dat", num, "[-1]",sep="")
fmla <- as.formula(paste("data$log_dur[-1] ~ ",
            paste(dat.lab, collapse= "+")))
fit<-lm(fmla)
counter<-counter+1
summ<-summary(fit)$coefficients[-1,4]
            num<-which(summ<0.05)
            v<-length(summ)
if (v==length(num)){
 continue<-FALSE
 return(list(fit=fitted(fit),count=counter,
                   summary=summary(fit)))
}
  else if (length(num)==0){
   continue<-FALSE
   return(list(fit=fitted(fitt),count=counter,
              summary=summary(fitt)))
  }
}
}
}
# Main function to remove diurnal pattern
```

```
Deseason <-function(dataname){
    data<-read.dbf(paste(dataname,"_trades.dbf",sep=""))
n<-dim(data)[1]
data<-cbind(data,matrix(rep(NA,17*n),ncol=17))
colnames(data)[10:22]<-c("dat1","dat2","dat3","dat4",
            "dat5","dat6","dat7","dat8","dat9",
            "dat10","dat11","dat12","dat13")
colnames(data)[23:26]<-c("log_dur","log_dur_fit",
            "X_adj","X_trunc")
node<-rep(NA,13)
for (j in 1:13){
        if (j%%2==0){node[j]<-(9+(j%/%2))*3600}
            else {node[j]<-(9+(j-1)%/%2)*3600+30*60}
    }
dif<-diff.C(node,data[,2])
dif<-matrix(dif,ncol=length(node))
data[-1,10:22]<-dif
data[,23]<-log(data$X)
data[-1,24]<-(fitting(data))$fit
data[-1,25]<-exp(data[-1,23]-data[-1,24])
data[-1,26]<-trunc.C(data)
    return(data)
}
################################################################
```

D.3 R codes for Section 3.3

```
#################################################################
#     Initialization of conditional median and kappa            #
#################################################################
init.real<-function(data){
    dates<-unique(data$DATE)
    m<-length(dates)
    med<-numeric(m); kk<-numeric(m); k.init<-numeric(m)
    t0950<-9*3600+50*60; t1000<-10*3600
    for (i in 1:m){
        expr<-median(data$X_adj[(data$TIME>=t0950)
                &(data$TIME<t1000)&(data$DATE==dates[i])])
        med[i]<-expr
    }
    return(sigma.init = med)
}
k.init = function(dataset){
    ts<-dataset$X_trunc[dataset$X_trunc!=0]
    k.init=sqrt(2*(mean(ts)/median(ts)-1))
    return(k.init)
}
#################################################################
#                  Conditional median dynamics                  #
#################################################################
sigma.real <- function(param,dataset){
    s_init<-init.real(dataset); l<-length(s_init)
```

```
    dates<-unique(dataset$DATE)
    for (i in 1:l){
        ts <-dataset$X_trunc[(dataset$DATE==dates[i])
                &(dataset$X_trunc!=0)]
        nn <-length(ts); s0<-s_init[i]
        sig<-.C("sigma_seq",as.double(param),as.double(ts),
            as.integer(nn),as.double(s0),result=double(nn))
        sig<-sig[["result"]]
        if (i==1){sigma<-sig}
           else {sigma<-c(sigma,sig)}
    }
    return(sigma)
}
#################################################################
#           Estimation of the BS-ACD(1,1) parameters            #
#################################################################
DataEst <- function(iniest,dataset){
    t0<-proc.time()
    est0<-c(iniest,k.init(dataset))
    coef<-matrix(rep(NA,8),ncol=8)
    p<-optim(est0[1:3],fn=ML.C2,gr=NULL,est0[4],dataset,
        method ="Nelder-Mead",
        control = list(maxit=5000,fnscale=-1),
        hessian = FALSE)
    q<-optim(c(p$par,est0[4]),fn=ML.C,gr=GradML.C,dataset,
        method ="BFGS",
        control = list(maxit=5000,fnscale=-1),
```

```
        hessian = TRUE)
    coef[,1:4]<-q$par;
    coef[,5:8]<-(-1)*diag(solve(q$hessian))
    t<-proc.time()-t0
    return(list(estim=coef,time=t))
}
#################################################################
#                   Example: SLB dataset                        #
#################################################################
company<-"slb"
data<-Deseason(company)
data1<-data[-1,]
init.guess<-c(-0.03,0.8,0.01)
est<-DataEst(init.guess,data1)
MLE<-est$estim[1:4]
#################################################################
#                     Goodness-of-Fit                           #
#################################################################
# Cox-Snell residuals
CoxSnellres<-function(param,dataset){
    ts<-dataset$X_trunc[dataset$X_trunc!=0]
    s<-sigma.real(param[1:3],dataset)
    k<-param[4]; arg<-(1/k)*(sqrt(ts/s)-sqrt(s/ts))
    res<--log(1-pnorm(arg))
    q<-quantile(res,probs=c(0.01,0.05,0.10,0.25,0.50,
        0.75,0.90,0.95,0.99))
    return(list(res=res,quant=q))
```

```
}
# QQ-plots: Cox-Snell residuals and Exp(1) r.v.
QPlots<-function(MLE,dataset){
    qe<-qexp(c(0.01,0.05,0.10,0.25,0.50,0.75,0.90,0.95,0.99))
    q<-CoxSnellres(MLE,dataset)$quant
    vec<-c(1,5,10,25,50,75,90,95,99)
    L<-paste(vec,"%",sep="")
    plot(seq(1,9), qe,type="l",col="black",xlab="",
        ylab="",lwd=2.5,axes=F)
    par(new=TRUE)
    plot(seq(1,9), q,type="l",col="red",xlab="",
        ylab="",axes=F,lwd=2.5)
    legend(6.0,1.0,c("Exp(1)","BS-ACD(1,1) Residuals"),
        lty=c(1,1),lwd=c(2.5,2.5),col=c("black","red"))
    axis(1, at=seq(1,9), labels = L, cex.axis = 0.75)
    axis(2, at=seq(0,5,by=0.1),cex.axis = 0.75)
}
# Visual checks of goodness-of-fit
# QQ plot of Cox-Snell residuals
QPlots(MLE,data1)
# Histogram of Cox-Snell residuals
truehist(CoxSnellres(MLE,data1)$res,xlab="",xlim=c(0,8),
    ymax=1,col="lightblue")
lines(seq(0,8,by=0.1),dexp(seq(0,8,by=0.1)),
    lwd=2.5,col="black")
#################################################################
#                    Standard errors                            #
```

```
#############################################################
# Mean of Cox-Snell residuals
meanRes<-mean(CoxSnellres(MLE,data1)$res)
# Accurate Hessian matrix at MLE
hes<-hessian(ML.C,MLE,method="Richardson",
    method.args=list(),dataset=data1)
# Gradient at MLE
gr<-GradML.C(MLE,dataset=data1)
# Inverse of the robust (White) covariance matrix
rob.cov<-solve(hes)%*%gr%*%t(gr)%*%solve(hes)
# Standard errors
std.err<-sqrt(diag(rob.cov))
#############################################################
```

Appendix E

R codes for Chapter 4

Remark E.1 *Note that only new functions are listed. Functions from Appendix C are used without notice.*

E.1 Estimation

```
#############################################################
#   Draw initial values at random at a predefined distance  #
#   from the true value (0.1,0.9,0.1)                       #
#############################################################
UnifRN<-function(param,prop){
    n<-length(param); vec<-rep(NA,n)
    if (n==3){
        vec[1]<-runif(1, min=param[1]*(1-prop/100),
            max=param[1]*(1+prop/100))
        vec[2]<-runif(1, min=param[2]*(1-prop/100),
            max=min(param[2]*(1+prop/100),1-(1e-4)))
```

```
        vec[3]<-runif(1, min=param[3]*(1-prop/100),
            max=param[3]*(1+prop/100))
    }
    return(vec)
}
# Main function to do simulations
SimACD <- function(param,prop,samsize,simnum){
    coef.NM<-matrix(rep(NA,16*simnum),ncol=16)
    coef.NMBF<-matrix(rep(NA,19*simnum),ncol=19)
    coef.BF<-matrix(rep(NA,17*simnum),ncol=17)
    for (j in 1:simnum){
        iniest<-UnifRN(param[1:3],prop)
        sam<-BS.C(param,samsize)
        d.stat<-c(mean(sam),median(sam),var(sam))
        est0<-c(iniest,init(sam)$k.init)
        t0<-proc.time()
        coef.NM[j,1:12]<-Opt(0,param,est0,sam)
        coef.NM[j,13:15]<-d.stat
        coef.NM[j,16]<-(proc.time()-t0)[3]
        t0<-proc.time()
        coef.NMBF[j,1:15]<-Opt(1,param,est0,sam)
        coef.NMBF[j,16:18]<-d.stat
        coef.NMBF[j,19]<-(proc.time()-t0)[3]
        t0<-proc.time()
        coef.BF[j,1:13]<-Opt(2,param,est0,sam)
        coef.BF[j,14:16]<-d.stat
        coef.BF[j,17]<-(proc.time()-t0)[3]
```

```
    }
    return(list(NM=coef.NM,NMBF=coef.NMBF,BF=coef.BF))
}
#################################################################
```

E.2 Processing

```
#################################################################
# Evaluation of the performance of the NM, BF, and NM-BFGS  #
# methods by various criteria                                   #
#################################################################
# Pairwise comparison:
Compare2 <- function(data1,data2,ind1,ind2){
 #Assume all pathological cases have been removed.
 #Values for ind1, ind2:
 #0 for NM, 1 for NM-BFGS, 2 for BFGS.
 #Compare performance of two 2 methods based on
 # COUNTING how many times the following are attained:
 #1. Average time (min)
 #2. Average distance to true value (min)
 #3. Average number of evaluations (min)
 #4. Average function value at MLE (max)
 #5. Average distance (min) AND function value (max)
 #6. Average number of eval (min) AND Average distance (min)
 #      AND function value (MAX)
    perf<-matrix(rep(0,12),ncol=6)
    n<-dim(data1)[1]
```

```
#time
   perf[1,1]<-length(which(data1[,12]<data2[,12]))
   perf[2,1]<-length(which(data1[,12]>data2[,12]))
#distance
   perf[1,2]<-length(which(data1[,10]<data2[,10]))
   perf[2,2]<-length(which(data1[,10]>data2[,10]))
#Eval
   perf[1,3]<-length(which(data1[,11]<data2[,11]))
   perf[2,3]<-length(which(data1[,11]>data2[,11]))
#fval
   perf[1,4]<-length(which(data1[,9]>data2[,9]))
   perf[2,4]<-length(which(data1[,9]<data2[,9]))
#distance and fval
   perf[1,5]<-length(which((data1[,10]<data2[,10])
           &(data1[,9]>data2[,9])))
   perf[2,5]<-length(which((data1[,10]>data2[,10])
           &(data1[,9]<data2[,9])))
#distance and fval and eval number
   perf[1,6]<-length(which((data1[,10]<data2[,10])
           &(data1[,9]>data2[,9])&(data1[,11]<data2[,11])))
   perf[2,6]<-length(which((data1[,10]>data2[,10])
           &(data1[,9]<data2[,9])&(data1[,11]>data2[,11])))
   if (ind1==0){a<-"NM"}else if (ind1==1){a<-"NM-BFGS"}
   if (ind2==1){b<-"NM-BFGS"}else if (ind2==2){b<-"BFGS"}
   rownames(perf)<-c(a,b)
   colnames(perf)<-c("Min time","Min distance",
           "Min number of Eval","Max fval",
```

```
            "Min dist AND Max fval",
            "Min dist AND Max fval AND Min Numb of Eval")
    return(perf)
}
# Overall comparison:
Compare3 <- function(data1,data2,data3){
 #Assume data1 - NM, data2- NMBF, data3 - BF.
 #Compare overall performance of  3 methods based on
 #1. Average distance to true value
 #2. Average number of evaluations
 #3. Average time (sec)
 #4. Maximum distance
    perf<-matrix(rep(NA,15),ncol=5)
    perf[1,3:5]<-apply(data1[,10:12],2,mean)
    perf[2,3:5]<-apply(data2[,10:12],2,mean)
    perf[3,3:5]<-apply(data3[,10:12],2,mean)
    perf[1,1]<-min(data1[,10]);perf[2,1]<-min(data2[,10]);
    perf[3,1]<-min(data3[,10]);
    perf[1,2]<-max(data1[,10]);perf[2,2]<-max(data2[,10]);
    perf[3,2]<-max(data3[,10]);
    colnames(perf)<-c("Minimal distance","Maximal distance",
        "Average distance","Average number of evaluations",
        "Average time (sec)")
    rownames(perf)<-c("NM","NM-BFGS","BFGS")
    return(perf)
}
# Number of simultaneous coincidences of quadruples
```

```
# (a,b,g,k) up to k=2,3,4 digits after decimal point:
IdMatr<-function(data1,data2,data3){
    perf<-rep(NA,3); j<-1
    for (k in c(2,3,4)){
        perf<-cbind(perf,IdCol(data1,data2,data3,k))
        j<-j+1
    }
return(perf[,-1])
}
IdCol<-function(data1,data2,data3,k){
    perf<-rep(NA,3)
    perf[1]<-length(which((abs(data1[,1]-data2[,1])<=10^(-k))
        &(abs(data1[,2]-data2[,2])<=10^(-k))
        &(abs(data1[,3]-data2[,3])<=10^(-k))
        &(abs(data1[,4]-data2[,4])<=10^(-k))))
    perf[2]<-length(which((abs(data1[,1]-data3[,1])<=10^(-k))
        &(abs(data1[,2]-data3[,2])<=10^(-k))
        &(abs(data1[,3]-data3[,3])<=10^(-k))
        &(abs(data1[,4]-data3[,4])<=10^(-k))))
    perf[3]<-length(which((abs(data2[,1]-data3[,1])<=10^(-k))
        &(abs(data2[,2]-data3[,2])<=10^(-k))
        &(abs(data2[,3]-data3[,3])<=10^(-k))
        &(abs(data2[,4]-data3[,4])<=10^(-k))))
    return(perf)
}
#################################################################
```

Bibliography

[1] A.R. Admati and P. Pfleiderer. A theory of intraday patterns: volume and price variability. *Review of Financial Studies*, 1:3–40, 1988.

[2] N. Balakrishna, N. Balakrishnan, and T. Rahul. The Birnbaum–Saunders AR(1) model. Unpublished paper, 2011.

[3] R.R. Barton and J.S. Ivey. Nelder–Mead simplex modifications for simulations optimization. *Management Science*, 42:954–973, 1996.

[4] L. Bauwens, P. Giot, J. Grammig, and D. Veredas. A comparison of financial duration models via density forecasts. *International Journal of Forecasting*, 20:589–609, 2004.

[5] E. Berndt, B. Hall, R. Hall, and J. Hausman. Estimation and inference in nonlinear structural models. *Annals of Economic and Social Measurement*, 3:653–665, 1974.

[6] B.R. Bhat. On the method of maximum–likelihood for dependent observations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36:48–53, 1974.

[7] C.R. Bhatti. *Statistical models for intraday trading dynamics*. PhD thesis, Rice University, 2007.

[8] C.R. Bhatti. The Birnbaum–Saunders autoregressive conditional duration model. *Mathematics and Computers in Simulation*, 80:2062–2078, 2010.

[9] Z.W. Birnbaum and S.C. Saunders. A new family of life distributions. *Journal of Applied Probability*, 6:319–327, 1969.

[10] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31:307–327, 1986.

[11] C.G. Broyden. A new double–rank minimization algorithm. *AMS Notices*, 16:670, 1969.

[12] D.R. Cox and E.J. Snell. A general definition of residuals. *Journal of the Royal Statistical Society, Series B (Methodological)*, 30:248–275, 1968.

[13] D.W. Diamond and R.E. Verrecchia. Constraints on short–selling and asset price adjustment to private information. *Journal of Financial Economics*, 18:277–311, 1987.

[14] D. Easley and M. O'Hara. Time and the process of security price adjustment. *Journal of Finance*, 47:577–605, 1992.

[15] M. Engelhardt, L. Bain, and F. Wright. Inferences on the parameters of the Birnbaum–Saunders fatigue life distribution based on maximum likelihood estimation. *Technometrics*, 23:251–256, 1981.

[16] R.F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50:987–1008, 1982.

[17] R.F. Engle. The econometrics of ultra–high–frequency data. *Econometrica*, 68:1–22, 2000.

[18] R.F. Engle and J.R. Russell. Autoregressive conditional duration: a new model for irregularly spaced transaction data. *Econometrica*, 66:1127–1162, 1998.

[19] R. Fletcher. A new approach to variable metric methods. *Computer Journal*, 13:317–322, 1970.

[20] Internet forum StackExchange. Normality testing is essentially useless. `http://stats.stackexchange.com/questions/2492/`, 2010.

[21] F.F. Gan and K.J. Koehler. Goodness-of-fit tests based on P–P probability plots. *Technometrics*, 32:289–303, 1990.

[22] Y.N. Ghosh and B. Mukherjee. On probabilistic properties of conditional medians and quantiles. *Statistics & Probability Letters*, 76:1775–1780, 2006.

[23] L.R. Glosten and P.R. Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics*, 14:71–100, 1985.

[24] D. Goldfarb. A family of variable metric methods derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.

[25] C. Gouriéroux, A. Monfort, and A. Trognon. Pseudo maximum likelihood methods: theory. *Econometrica*, 52:681–700, 1984.

[26] J. Grammig and K.-O. Maurer. Non-monotonic hazard functions and the autoregressive conditional duration model. *Econometrics Journal*, 3:16–38, 2000.

[27] N. Hautsch. *Modelling irregularly spaced financial data - theory and practice of dynamic duration models*. Springer-Verlag, Berlin, 2004.

[28] T.-H. Kim and H. White. On more robust estimation of skewness and kurtosis: simulation and application to the S&P500 index. Working paper, University of California, San Diego, December 2003.

[29] D. Kundu, N. Kannan, and N. Balakrishnan. On the hazard function of Birnbaum–Saunders distribution and associated inference. *Computational Statistics and Data Analysis*, 52:2692–2702, 2008.

[30] J.C. Lagarias, J.A. Reeds, M.H. Wright, and P.E. Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

[31] E.L. Lehmann. *Theory of point estimation*. Wiley, New York, 1st edition, 1983.

[32] V. Leiva, Riquelme M., N. Balakrishnan, and A. Sanhueza. Lifetime analysis based on the generalized Birnbaum–Saunders distribution. *Computational Statistics and Data Analysis*, 52:2079–2097, 2008.

[33] A. Lemonte and G. Cordeiro. Birnbaum–Saunders nonlinear regression models. *Computational Statistics and Data Analysis*, 53:4441–4452, 2009.

[34] A. Lunde. A generalized gamma autoregressive conditional duration model. Technical report, Aalborg University, February 1999.

[35] A. Madhavan. Market microstructure: a survey. *Journal of Financial Markets*, 3:205–258, 2000.

[36] K.I.M. McKinnon. Convergence of the Nelder–Mead simplex method to a nonstationary point. *SIAM Journal of Optimization*, 9:148–158, 1998.

[37] S. Meintanis. Inference procedures for the Birnbaum–Saunders distribution and its generalizations. *Computational Statistics and Data Analysis*, 54:367–373, 2010.

[38] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[39] D.B. Nelson. Conditional heteroskedasticity in asset returns: a new approach. *Econometrica*, 59:347–370, 1991.

[40] H.K.T. Ng, D. Kundu, and N. Balakrishnan. Modified moment estimation for the two-parameter Birnbaum–Saunders distribution. *Computational Statistics & Data Analysis*, 43:283–298, 2003.

[41] K.H. Ng, D.E. Allen, and S. Peiris. Fitting Weibull ACD models to high frequency transactions data: a semi-parametric approach based on estimating functions. Working paper, Edith Cowan University, January 2009.

[42] M. O'Hara. *Market microstructure theory.* Blackwell Publishers Ltd, Cambridge, 1995.

[43] R Development Core Team. *R: a language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria, 2010. `http://www.R-project.org/`.

[44] R Development Core Team. *Writing R extensions, version 2.13.1.* R Foundation for Statistical Computing, Vienna, Austria, 2011. `http://www.R-project.org/`.

[45] J. Rieck. A moment–generating function with application to the Birnbaum–Saunders distribution. *Communications in Statistics - Theory and Methods*, 28:2213–2222, 2007.

[46] J. Rieck and J. Nedelman. A log-linear model for the Birnbaum–Saunders distribution. *Technometrics*, 33:51–60, 1991.

[47] A. Sanhueza, V. Leiva, and N. Balakrishnan. The generalized Birnbaum–Saunders distribution and its theory, methodology, and application. *Communications in Statistics - Theory and Methods*, 37:645–670, 2008.

[48] D.F. Shanno. Conditioning of quasi–Newton methods for function minimization. *Mathematics of Computation*, 24:647–657, 1970.

[49] H.C. Thode. *Testing for normality*. Marcel Dekker, New York, 2002.

[50] J.J. Tomick, S.F. Arnold, and R.R. Barton. Sample size selection for improved Nelder–Mead performance. In C. Alexopoulos, K. Kang, W.R. Lilegdon, and D. Goldsman, editors, *Proceedings of the 1995 winter conference*, pages 341–345, 1995.

[51] R.J. Tomkins. On conditional medians. *Annals of Probability*, 3:375–379, 1975.

[52] R.S. Tsay. *Analysis of financial time series*. Wiley, Hoboken, New Jersey, 3rd edition, 2010.

[53] I.N. Volodin and O.A. Dzhungurova. On limit distributions emerging in the generalized Birnbaum–Saunders model. *Journal of Mathematical Sciences*, 99:1348–1366, 2000.