Spring 5-1-2014

High Frequency Data: Modeling Durations via the ACD and Log ACD Models

Lilian Cheung
University of Connecticut - Storrs, lilian.c.cheung@gmail.com

High Frequency Data: Modeling Durations via the ACD and Log ACD Models

Lilian Cheung*

Supervised by Professor Nalini Ravishanker
Department of Statistics
University of Connecticut, Storrs

A thesis submitted in partial fulfillment of the
requirements to graduate as an
Honors Scholar in Mathematics-Statistics

* Departments of Mathematics and Statistics, Department of Economics
Contact: lilian.c.cheung@gmail.com

Acknowledgements

Thank you to my thesis advisor, Professor Nalini Ravishanker, for her patience and support. Over the past few years, Professor Ravishanker has pushed me to attend conferences, explore unfamiliar topics, and learn on my own. She has never hesitated to share her expertise in every class, and her love of time series has been contagious. I am grateful for her encouragement and advice.

I would also like to thank the faculty of the Department of Mathematics, the Department of Statistics, and the Department of Economics for their roles in shaping my interests. In particular, Professor Nishith Prakash has provided guidance, encouragement, and moral support over the past year. He and Professor Richard Suen have encouraged me to develop my interest in economics. In addition, thank you to Professor Evarist Giné for supporting my pursuit of graduate school, to Daniel Kelleher for sharing his knowledge about graduate studies in my freshman year, and to the members of Professor Ravishanker's independent study group for their help in my junior year.

Finally, thank you to my friends and family for their support. My family's hard work, compassion, and love have always been a source of inspiration to me. To my parents: Thank you for your unconditional love, for proofreading my thesis, and for supporting my goals.

Abstract

This thesis proposes a method of finding initial parameter estimates in the Log ACD_1 model for use in recursive estimation. The recursive estimating equations method is applied to the Log ACD_1 model to find recursive estimates for the unknown parameters in the model. A literature review is provided on the ACD and Log ACD models, and on the theory of estimating equations. Monte Carlo simulations indicate that the proposed method of finding initial parameter estimates is viable. The parameter estimation process is demonstrated by fitting an ACD model and a Log ACD model to a set of IBM stock duration data.

Keywords: Time Series, High Frequency Duration Data, ACD and Log ACD Duration Models, Initial Parameter Estimation, Recursive Estimating Equations, Monte Carlo Simulations

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

1.1.1 High Frequency Data

High frequency time series data are data collected at high frequencies over time. Such data have become relevant across multiple fields due to technological advancements in data collection and data storage. For instance, in the medical field, electrocardiogram (ECG) data are sampled in real time at frequencies exceeding 100 cycles per second (Hejjel and Roth , 2004). In recent years, programmers have developed software to extract real time information on user preferences from websites. The classification of data as "high frequency data" depends upon the application. In medicine, some medical devices provide high frequency data sampled every second. In transportation, data on traffic volume and congestion can be collected every 90 seconds (Vlahogianni, Karlaftis, and Kepaptsoglou , 2010). Our focus is on high frequency financial data, which is readily available for analysis. Yan and Zivot (2003) define high frequency financial data as financial observations taken at a time scale finer than once per day.

Financial databases once recorded only daily data on the opening or closing characteristics markets. Now, an increasing number of databases contain real-time information on the time and associated characteristics (e.g., price and volume) of every transaction. These transaction-by-transaction financial data are collected at a time scale much finer than once per day. The unique nature of high frequency financial data prompted the development of

new methods of statistical modeling. Typically, high frequency financial data arrive over irregularly spaced time intervals. In other words, the duration of time between consecutive data points is not uniform; data may arrive rapidly, separated by short durations, or arrive slowly, with long durations between each arrival time (Pacurar , 2008).

Traditional discrete time series methods divided data points into fixed time intervals prior to analysis. These fixed-interval models were the first models used to analyze transaction-by-transaction financial data. However, the selection of a suitable time interval for structuring data presents certain problems. In addition to arriving over irregularly spaced time intervals, high frequency financial data tend to display varying patterns of intra-day and intra-week behavior. For instance, stock market activity tends to peak around opening and closing times of the market (Yan and Zivot , 2003). Dividing observations into short time intervals is suitable for the periods of high market activity; but in the periods of low market activity, could result in heteroskedasticity due to the presence of intervals that contain no additional information. Conversely, dividing observations into long time intervals would risk the loss of information contained in rapidly arriving data (Engle and Russell , 1998).

The problem with using fixed-interval models to analyze irregularly spaced time series data spurred Engle and Russell (1998) to propose a nonlinear model for the time intervals, or durations. This model, called the Autoregressive Conditional Duration (ACD) model, is a special case in the class of generalized duration models proposed by Thavaneswaran, Ravishanker, and Liang (2014).

1.1.2 The Interest in Durations

In financial applications, the interest in durations stems from the idea that the time between events in high frequency data contain valuable information. An event may be defined as a single transaction, as a price change exceeding a certain amount, or as some other characteristic of interest. Engle and Russell (1998) argue that clusters of high activity versus low activity, indicated by short durations versus long durations between events, can reveal information about the market microstructure. For instance, Easley and O'Hara (1992) suggest that the release of information into the stock market increases the number of informed traders in the market. Thus, a cluster of short durations between transactions could indicate the response of traders to new information in the market. The tendency for both short durations and long durations to be followed by durations of similar length is known as "clustering" (Pacurar , 2008).

Models for high frequency duration data are important across multiple fields. A considerable body of literature exists on the study of time intervals between heartbeats and neural spikes (Paninski 2010). In transportation, studying the duration of traffic congestion can aid officials in determining the time costs of traffic delays (Vlahogianni, Karlaftis, and Kepaptsoglou , 2010). Vlahogianni, Karlaftis, and Kepaptsoglou (2010) used the ACD model to model traffic congestion durations.

1.2 Durations

1.2.1 Definition

Formally, durations are defined as the time between two consecutive events. Let t_i be the time of occurrence for the i^{th} event. Then, the i^{th} duration is defined as the time interval between the events occurring at times t_i and t_{i-1}:

$$x_i = t_i - t_{i-1} \tag{1}$$

Since x_i are waiting times, $\{x_i\}$ must be a non–negative sequence of random variables.

1.2.2 Point Processes

A time series of durations generated from high frequency events arriving over time can be viewed as one realization of a temporal point process (i.e., a random process that generates points on the time axis). Point processes can be represented in one of several ways. In particular, point processes can be thought of as a time series of cumulative event counts, as a binary time series, or as a duration series. Each of these representations contains enough information to derive the others. Thus, given a sequence of durations, we can derive information about the number of events that have occurred and the arrival time of each event.

Mathematically, consider a sequence of events arriving over time, where the arrival times of each event are given by the sequence $\{t_0, t_1, ..., t_n\}$, and $t_0 \leq t_1 \leq ... \leq t_n$. Multiple events may occur at the same time. The time series of durations $\{x_i\}_{i=1}^{n}$ is one way to represent the process. We can also define $N(s)$ to be the number of events that have occurred by time

s, where $s \in [0, S]$ and S is the time of the last observation. $N(s)$ is the series of cumulative event counts over continuous time (Pacurar , 2008). At any given $s \in [0, S]$, an event either occurs if $s = t_i \in \{t_1, ..., t_n\}$, or does not occur. Define the binary time series indicating event occurrence as $event_s = \{1 \text{ if } s = t_i \in \{t_1, ..., t_n\} \text{ and } 0 \text{ otherwise}\}$. The value of $N(s)$ at any given time $s \in [0, T]$ is equivalent to the number of event occurrences between 0 and s. The sequence of durations is equivalent to the time between each event occurrence. Thus, there is a direct relationship between the time series of counts $N(s)$, the sequence of arrival times $\{t_i\}_{i=0}^n$, and the sequence of durations $\{x_i\}_{i=1}^n$ (Brillinger and Guttorp , 2002).

In many cases, additional information is stored along with the arrival time of each event. These cases are known as marked point processes, and each additional piece of information is called a mark. Define $\{M_0, M_1, ..., M_n\}$ to be the sequence of marks associated with the arrival times $\{t_0, t_1, ..., t_n\}$. Then, the marked point process may be represented via the time series of durations and marks $\{(x_i, M_i) \text{ where } i = 1, 2, ..., n\}$. Though we focus on the analysis of durations data $\{x_i, i = 1, 2, ..., n\}$ alone, we note that it is possible to jointly model durations and their associated marks.

Consider a point process denoted by the sequence of durations $\{x_i\}_{i=1}^n$. Let $\mathcal{F}_{i-1}^x$ denote the information associated with the sequence of durations up to time t_{i-1}, including durations $\{x_1, x_2, ..., x_{i-1}\}$ as well as any additional information associated with the durations (Pacurar , 2008). Point processes may evolve "*without* after-effects," in which case the value of the process at any time t_i is independent of its value at any previous point. Alternatively, point processes may evolve "*with* after-effects," in which case there is a non-zero

time dependence between the value of the point process at time t_i and the value of the point process at prior times. The ACD model developed by Engle and Russell (1998) considers point processes expressed as time series of durations (and marks) that evolve with after-effects. Duration modeling is one way to understand not only the sequence of durations over time, but also the sequence of arrival times of events and information associated with the arrival with those events.

1.3 Organization

This thesis reviews the literature on parameter estimation in duration models, focusing on parameter estimation in the class of Log ACD models introduced by Bauwens and Giot (2000). Specifically, the Estimation Equations approach is studied (see section 2.2.2). The thesis adds to the existing literature by deriving a method to find initial values for recursive parameter estimation in the Log ACD_1 model using the estimating equations approach presented by Thavaneswaran, Ravishanker, and Liang (2014). Additionally, a Monte Carlo simulation study is conducted to test the viability of the method. We use a set of IBM stock duration data to demonstrate the combined use of the initial values method and the recursive estimating equations method in fitting two duration models to the data.

The remainder of the thesis is organized as follows: Section 2 presents a brief overview of duration models and parameter estimation in duration models. Section 3 applies the recursive estimating equations method to Log ACD models and derives a method to find initial values for the Log ACD_1 model. Section 4 presents the results of a Monte Carlo simulation study of parameter estimation in the Log ACD models via estimating equations.

Section 5 applies the estimating equations approach to fit duration models to a set of IBM stock duration data. Section 6 concludes.

2 Background

2.1 A Brief Overview of Duration Models

2.1.1 The ACD Model

The ACD model is a non-linear model designed to capture the clustering effect often seen in high frequency financial durations as well as the time dependence between durations in point processes that evolve with after-effects (Engle and Russell , 1998).

Given a time series of durations $x_i = t_i - t_{i-1}$, where $i = 1, 2, ..$, the framework of the ACD model assumes that the time dependence between the i^{th} duration and prior durations is completely explained by ψ_i, where ψ_i is the expectation of the duration x_i given past information associated with the durations, viz., $\mathcal{F}_{i-1}^x$. The basic ACD(1,1) model proposed by Engle and Russell (1998) has the following form:

$$\begin{aligned} x_i &= \psi_i \varepsilon_i \\ \psi_i &= E[x_i|\mathcal{F}_{i-1}^x] = \omega + \alpha x_{i-1} + \beta \psi_{i-1} \end{aligned} \tag{2}$$

where the errors ε_i are assumed to be independent and identically distributed non-negative random variables with $E(\varepsilon_i) = 1$ and density function f_ε; the unknown parameters are $\boldsymbol{\theta} = \{\omega, \alpha, \beta\}$; and the conditions $\omega > 0$, $\alpha \geq 0$, $\beta \geq 0$, $\alpha + \beta < 1$ are required to ensure the non-negativity and weak stationarity of durations x_i. In addition, $\mathcal{F}_{i-1}^x$ is assumed to be independent from ε_i.

Engle and Russell (1998) also proposed a more general ACD(p,q) model with the form

$$\begin{aligned} x_i &= \psi_i \varepsilon_i \\ \psi_i &= \omega + \sum_{j=1}^{p} \alpha_j x_{i-j} + \sum_{j=1}^{q} \beta_j \psi_{i-j} \end{aligned} \tag{3}$$

where the unknown parameters are $\boldsymbol{\theta} = (\omega, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q)$. The conditions $\omega > 0$, $\alpha_j \geq 0$ for $j = 1, \cdots, p$, $\beta_j \geq 0$ for $j = 1, \cdots, q$ and $\sum_{j=1}^{p} \alpha_j + \sum_{j=1}^{q} \beta_j < 1$, are again required to ensure that the time series x_i is non-negative and stationary.

2.1.2 The Log ACD Models

One drawback of the standard ACD(p,q) model is the requirement of non-negativity constraints on the model parameters to ensure positive-valued durations. Bauwens and Giot (2000) proposed the Log ACD_1 and Log ACD_2 models to introduce a duration model with more flexibility than the ACD model. Unlike the ACD model, the Log ACD models have no positivity constraints on the model parameters, which facilitates the inclusion of exogenous explanatory variables in the model. We give the form of the Log ACD models, following the notation in Pacurar (2008).

The general Log ACD_1(p,q) model has the following form:

$$\begin{aligned} x_i &= \exp(\psi_i) \varepsilon_i \\ \psi_i &= \omega + \sum_{j=1}^{p} \alpha_j \log x_{i-j} + \sum_{j=1}^{q} \beta_j \psi_{i-j}, \end{aligned} \tag{4}$$

where $\sum_{j=1}^{\max(p,\ q)}(\alpha_j + \beta_j) < 1$ is required to ensure weak stationarity.

The general Log ACD_2(p,q) model has the following form:

$$\begin{aligned} x_i &= \exp(\psi_i)\varepsilon_i \\ \psi_i &= \omega + \sum_{j=1}^{p} \alpha_j \frac{x_{i-j}}{\exp(\psi_{i-j})} + \sum_{j=1}^{q} \beta_j \psi_{i-j}, \end{aligned} \tag{5}$$

where $\sum_{j=1}^{q} \beta_j < 1$ is required to ensure weak stationarity. In both Log ACD models, ε_i are assumed to be independent and identically distributed non-negative random variables.

Variations of the ACD(p,q) and Log ACD(p,q) models may be derived by specifying different density functions for ε_i and by varying the number of lags (p,q) to include in the model. For instance, Engle and Russell (1998) considered exponential ACD (EACD) and Weibull ACD (WACD) models.

2.1.3 Other Duration Models

Many other duration models evolved from the basic framework of the ACD model, including the Stochastic Conditional Durations (SCD) model (Bauwens and Veredas , 2004) and the fractionally integrated ACD model (Jasiak , 1998). For a more detailed survey on existing duration models, see Pacurar (2008).

Thavaneswaran, Ravishanker, and Liang (2014) introduced a class of generalized duration models that generalized the form of existing duration models in the literature. The

generalized duration model has the following form:

$$x_i = \widetilde{h}(\mathcal{F}_{i-1}^x, \psi_i, z_i)\varepsilon_i, \tag{6}$$

where $\widetilde{h}(\mathcal{F}_{i-1}^x, \psi_i, z_i)$ is some function of past information $\mathcal{F}_{i-1}^x$, ψ_i, and z_i. $\{\psi_i\} = E[x_i|\mathcal{F}_{i-1}^x]$ is the conditional mean of data $\{x_i\}_{i=1}^n$ given its associated information, and $\{z_i\}$ is a random process independent of the information $\mathcal{F}_{i-1}^x$.

The ACD(p,q) model is a member of the class of generalized duration models with $\widetilde{h}(\mathcal{F}_{i-1}^x, \psi_i, z_i) = \psi_i$. Likewise, the Log ACD(p,q) models are members of the class of generalized duration models with $\widetilde{h}(\mathcal{F}_{i-1}^x, \psi_i, z_i) = \exp(\psi_i)$.

2.2 Parameter Estimation in Duration Models

2.2.1 Maximum Likelihood Estimation in the ACD Model

Much of the literature on parameter estimation in the ACD(p,q) model focuses on maximum likelihood methods. Given data $\{x_i\}_{i=1}^n$ derived from the ACD(p,q) model with unknown parameters $\boldsymbol{\theta} = (\omega, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q)$, the likelihood of $\boldsymbol{\theta}$, equals the joint density function of the data, which has the following form given in Tsay (2009):

$$f(\boldsymbol{x_n}|\boldsymbol{\theta}) \;=\; f(\boldsymbol{x_g}|\boldsymbol{\theta}) \times \prod_{i=g+1}^{n} f(x_i|\boldsymbol{x_{i-1}}, \boldsymbol{\theta}),$$

where the conditional likelihood function is

$$\mathcal{L}(\theta|\boldsymbol{x_n}) = \prod_{i=g+1}^{n} f(x_i|\boldsymbol{x_{i-1}}, \boldsymbol{\theta}),$$

where $g = max(p, q) = t_0$, and the conditional log-likelihood function $\ell(\theta|\boldsymbol{x_n})$ is

$$\ell(\theta|\boldsymbol{x_n}) = \sum_{i=t_0+1}^{n} \log(f(x_i|\boldsymbol{x_{i-1}}, \boldsymbol{\theta})).$$

If the correct specification of ε_i, (and hence, $f(x_i|\boldsymbol{x_{i-1}}, \boldsymbol{\theta})$) is known, then the conditional maximum likelihood (ML) estimates can be derived by maximizing the conditional log-likelihood function. Otherwise, quasi maximum likelihood (QML) estimates may be obtained by choosing some distribution for ε_i and maximizing the conditional maximum likelihood function using the chosen distribution of ε_i. For instance, Engle and Russell (1998) use the conditional likelihood function of an ACD model with exponential ε_i to derive quasi maximum likelihood estimates for the ACD model.

2.2.2 Estimating Equations (EE) Method

Although the likelihood function for the standard ACD(p,q) model can be derived if the true distribution of ε_i is known, the distribution of ε_i is rarely known in reality. In the ACD(p,q) case, the QML method provides a viable way to estimate model parameters, but there are some duration models for which the QML and ML parameter estimation methods may not be feasible (Thavaneswaran, Ravishanker, and Liang , 2014). Thavaneswaran, Ravishanker, and Liang (2014) recently proposed a recursive method for parameter estima-

tion in the class of generalized duration models based on combined martingale estimating equations. This provides an alternative approach to likelihood based methods for fitting duration models.

Godambe (1985) first developed the martingale estimating equations approach to estimate parameters given data from some stochastic process. Based on this framework, Thavaneswaran and Abraham (1988) developed linear estimating equations to estimate parameters in nonlinear time series. Thavaneswaran, Ravishanker, and Liang (2014) expanded upon these ideas by deriving optimal combined estimating equations based on both linear and generalized martingale differences, and by proposing recursive estimating equations based on the combined estimating equations.

As discussed in Thavaneswaran, Ravishanker, and Liang (2014), given data $\{x_i\}_{i=1}^n$ that is one realization of a stochastic process dependent upon some parameter $\boldsymbol{\theta} \in \mathbb{R}^P$, and the information $\mathcal{F}_i^x$ associated with $\{x_1, \ldots, x_i, 1 \leq i \leq n\}$, the P-dimensional martingale estimating function $\mathbf{g}_n(\boldsymbol{\theta})$ is defined as

$$\mathbf{g}_n(\boldsymbol{\theta}) = \sum_{i=1}^{n} \mathbf{a}_{i-1}(\boldsymbol{\theta}) \mathbf{h}_i(\boldsymbol{\theta}),$$

where $\mathbf{h}_i(\boldsymbol{\theta}) = \mathbf{h}_i(x_1, \ldots, x_i, \boldsymbol{\theta}), 1 \leq i \leq n$ are Q-dimensional martingale differences with $P \leq Q$, and $\mathbf{a}_{i-1}(\boldsymbol{\theta}) = \mathbf{a}_{i-1}(x_1, \ldots, x_{i-1}, \boldsymbol{\theta})$ are $P \times Q$ matrices. It is assumed that $\mathbf{g}_n(\boldsymbol{\theta})$ are differentiable with respect to the components of $\boldsymbol{\theta}$, and that $\mathrm{E}\left[\frac{\partial \mathbf{g}_n(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \middle| \mathcal{F}_{n-1}^x \right]$ and $\mathrm{E}[\mathbf{g}_n(\boldsymbol{\theta}) \mathbf{g}_n(\boldsymbol{\theta})' | \mathcal{F}_{n-1}^x]$ are nonsingular for all $\boldsymbol{\theta}$ and for each $n \geq 1$. $\mathrm{E}[\mathbf{g}_n(\boldsymbol{\theta}) \mathbf{g}_n(\boldsymbol{\theta})' | \mathcal{F}_{n-1}^x]$

is further assumed to be positive definite for all $\boldsymbol{\theta}$. The optimal estimating function $\mathbf{g}_n^*(\boldsymbol{\theta})$ in the set of all $\mathbf{g}_n(\boldsymbol{\theta})$ maximizes the Godambe information matrix $\mathbf{I}_{\mathbf{g}_n}(\boldsymbol{\theta})$, where $\mathbf{I}_{\mathbf{g}_n}(\boldsymbol{\theta})$ is given by

$$\mathbf{I}_{\mathbf{g}_n}(\boldsymbol{\theta}) = \left(\sum_{i=1}^{n} \mathbf{a}_{i-1}(\boldsymbol{\theta}) E\left[\frac{\partial \mathbf{h}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \middle| \mathcal{F}_{i-1}^x\right]\right)' \times$$
$$\left(\sum_{i=1}^{n} E[(\mathbf{a}_{i-1}(\boldsymbol{\theta})\mathbf{h}_i(\boldsymbol{\theta}))(\mathbf{a}_{i-1}(\boldsymbol{\theta})\mathbf{h}_i(\boldsymbol{\theta}))' | \mathcal{F}_{i-1}^x]\right)^{-1} \left(\sum_{i=1}^{n} \mathbf{a}_{i-1}(\boldsymbol{\theta}) E\left[\frac{\partial \mathbf{h}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \middle| \mathcal{F}_{i-1}^x\right]\right),$$

and has the form

$$\mathbf{g}_n^*(\boldsymbol{\theta}) = \sum_{i=1}^{n} \mathbf{a}_{i-1}^*(\boldsymbol{\theta})\mathbf{h}_i(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(E\left[\frac{\partial \mathbf{h}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \middle| \mathcal{F}_{i-1}^x\right]\right)' (E[\mathbf{h}_i(\boldsymbol{\theta})\mathbf{h}_i(\boldsymbol{\theta})' | \mathcal{F}_{i-1}^x])^{-1}\mathbf{h}_i.$$

Solving the estimating equation $\mathbf{g}_n^*(\boldsymbol{\theta}) = 0$ yields the optimal estimate for the parameter $\boldsymbol{\theta}$ (Godambe , 1985).

Thavaneswaran, Ravishanker, and Liang (2014) defined the combined estimating function $g_C(\boldsymbol{\theta})$ to be

$$g_C(\boldsymbol{\theta}) = \sum_{i=1}^{n} (\mathbf{a}_{i-1}(\boldsymbol{\theta})m_i(\boldsymbol{\theta}) + \mathbf{b}_{i-1}(\boldsymbol{\theta})M_i(\boldsymbol{\theta}))\},$$

where $\mathbf{a}_{i-1}(\boldsymbol{\theta}) = \mathbf{a}_{i-1}(x_1, \ldots, x_{i-1}, \boldsymbol{\theta})$, $\mathbf{b}_{i-1}(\boldsymbol{\theta}) = \mathbf{b}_{i-1}(x_1, \ldots, x_{i-1}, \boldsymbol{\theta})$, $m_i(\boldsymbol{\theta}) = x_i - \mu_i(\boldsymbol{\theta})$ are martingale differences, and $M_i(\boldsymbol{\theta}) = \widetilde{q}(m_i(\boldsymbol{\theta})) - E[\widetilde{q}(m_i(\boldsymbol{\theta}))|\mathcal{F}_{i-1}^x]$ are generalized martingale differences for $i = 1, \cdots, n$. Various specifications for $\widetilde{q}(m_i(\boldsymbol{\theta}))$ may be chosen to yield different generalized martingale differences. The first two conditional moments of $\{x_i, i =$

$1, 2, ...\}$ are given by

$$\mu_i(\boldsymbol{\theta}) = E\left[x_i|\mathcal{F}_{i-1}^x\right] \tag{7}$$

$$\sigma_i^2(\boldsymbol{\theta}) = Var\left(x_i|\mathcal{F}_{i-1}^x\right) \tag{8}$$

and the quadratic variation of $m_i(\boldsymbol{\theta})$, quadratic variation of $M_i(\boldsymbol{\theta})$, and quadratic covariation of $m_i(\boldsymbol{\theta})$ and $M_i(\boldsymbol{\theta})$ are respectively defined as

$$\langle m\rangle_i = E[m_i^2(\boldsymbol{\theta})|\mathcal{F}_{t-1}^x] = \sigma_i^2(\boldsymbol{\theta}) \tag{9}$$

$$\langle M\rangle_i = E[\widetilde{q}^2(m_i(\boldsymbol{\theta})|\mathcal{F}_{i-1}^x] - \left(E[\widetilde{q}(m_i(\boldsymbol{\theta}))|\mathcal{F}_{i-1}^x]\right)^2 \tag{10}$$

$$\langle m, M\rangle_i = E[m_i(\boldsymbol{\theta})\widetilde{q}(m_i(\boldsymbol{\theta}))|\mathcal{F}_{i-1}^x]. \tag{11}$$

Using these definitions, Thavaneswaran, Ravishanker, and Liang (2014) derived the optimal combined estimating function

$$g_C^*(\boldsymbol{\theta}) = \sum_{i=1}^{n}\left(\mathbf{a}_{i-1}^*(\boldsymbol{\theta})m_i(\boldsymbol{\theta}) + \mathbf{b}_{i-1}^*(\boldsymbol{\theta})M_i(\boldsymbol{\theta})\right),$$

where

$$\mathbf{a}_{i-1}^*(\boldsymbol{\theta}) = \rho_i^2\left(-\frac{\partial\mu_i(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\frac{1}{\langle m\rangle_i} - \left(E\left[\frac{\partial\widetilde{q}_i}{\partial\boldsymbol{\theta}}\middle|\mathcal{F}_{i-1}^x\right] - \frac{\partial E\left[\widetilde{q}_i|\mathcal{F}_{i-1}^x\right]}{\partial\boldsymbol{\theta}}\right)\eta_i\right) \tag{12}$$

and

$$\mathbf{b}_{i-1}^*(\boldsymbol{\theta}) = \rho_i^2\left(\frac{\partial\mu_i(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\eta_i + \left(E\left[\frac{\partial\widetilde{q}_i}{\partial\boldsymbol{\theta}}\middle|\mathcal{F}_{i-1}^x\right] - \frac{\partial E\left[\widetilde{q}_i|\mathcal{F}_{i-1}^x\right]}{\partial\boldsymbol{\theta}}\right)\frac{1}{\langle M\rangle_i}\right), \tag{13}$$

with

$$\rho_i^2 = \left(1 - \frac{\langle m, M\rangle_i^2}{\langle m\rangle_i\langle M\rangle_i}\right)^{-1} \textit{and} \quad \eta_i = \frac{\langle m, M\rangle_i}{\langle m\rangle_i\langle M\rangle_i} \tag{14}$$

and the optimal recursive estimate for $\boldsymbol{\theta}$ based on the combined estimating function:

$$\begin{aligned}
\widehat{\boldsymbol{\theta}}_i &= \widehat{\boldsymbol{\theta}}_{i-1} + \mathbf{K}_i \left(\mathbf{a}^*_{i-1}(\widehat{\boldsymbol{\theta}}_{i-1}) m_i(\widehat{\boldsymbol{\theta}}_{i-1}) + \mathbf{b}^*_{i-1}(\widehat{\boldsymbol{\theta}}_{i-1}) M_i(\widehat{\boldsymbol{\theta}}_{i-1}) \right), \qquad (15) \\
\mathbf{K}_i &= \mathbf{K}_{i-1} \left(\mathbf{I}_p - \left(\mathbf{a}^*_{i-1}(\widehat{\boldsymbol{\theta}}_{i-1}) \frac{\partial m_i(\widehat{\boldsymbol{\theta}}_{i-1})}{\partial \boldsymbol{\theta}'} + \frac{\partial \mathbf{a}^*_{i-1}(\widehat{\boldsymbol{\theta}}_{i-1})}{\partial \boldsymbol{\theta}} m_i(\widehat{\boldsymbol{\theta}}_{i-1}) \right. \right. \\
&+ \left. \left. \mathbf{b}^*_{i-1}(\widehat{\boldsymbol{\theta}}_{i-1}) \frac{\partial M_i(\widehat{\boldsymbol{\theta}}_{i-1})}{\partial \boldsymbol{\theta}'} + \frac{\partial \mathbf{b}^*_{i-1}(\widehat{\boldsymbol{\theta}}_{i-1})}{\partial \boldsymbol{\theta}} M_i(\widehat{\boldsymbol{\theta}}_{i-1}) \right) \mathbf{K}_{i-1} \right)^{-1}, \qquad (16)
\end{aligned}$$

where $\mathbf{I}_P$ is the P-dimensional identity matrix, and the forms of $\mathbf{a}^*_{i-1}$ and $\mathbf{b}^*_{i-1}$ are given in (12) and (13).

The recursive estimating equations method uses information on the first four conditional moments of the observed process without requiring that the distribution of ε_i be specified, and can be applied to the entire class of generalized duration models. Particularly, the method can be used to estimate $\boldsymbol{\theta} = \omega, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q$ in the ACD(p,q) and Log ACD(p,q) models. Since the i^{th} recursive estimate of $\boldsymbol{\theta}$, $\widehat{\boldsymbol{\theta}}_i$, depends upon its previous estimate $\widehat{\boldsymbol{\theta}}_{i-1}$, the results of the method are affected by the initial values chosen for $\widehat{\boldsymbol{\theta}}_1$. Thus, the first step to using the recursive estimating equations method is to find a viable method to determine initial parameter values given data $\{x_i\}_{i=1}^n$. In the next section, we derive a method to determine initial parameter values in the Log ACD_1 model.

3 Parameter Estimation and Initial Values in the Log ACD Models

3.1 Recursive Estimation in the Log ACD Models

Recall the Log ACD_1(p,q) and Log ACD_2(p,q) models defined in (4) and (5). Given the mean, variance, third central moment, and fourth central moment of ε_i denoted respectively by μ_ε, σ_ε^2, γ_ε, and κ_ε, the conditional mean, variance, third central moment, and fourth central moment of $\{x_i\}$ have the following forms given in Thavaneswaran, Ravishanker, and Liang (2014):

$$\begin{aligned}
\mu_i(\boldsymbol{\theta}) &= E\left[x_i|\mathcal{F}_{i-1}^x\right] = \mu_\varepsilon \exp(\psi_i) \\
\sigma_i^2(\boldsymbol{\theta}) &= Var\left(x_i|\mathcal{F}_{i-1}^x\right) = \sigma_\varepsilon^2 \exp(2\psi_i) \\
\gamma_i(\boldsymbol{\theta}) &= E\left[(x_i - \mu_i(\boldsymbol{\theta}))^3|\mathcal{F}_{i-1}^x\right] = \gamma_\varepsilon \exp(3\psi_i) \\
\kappa_i(\boldsymbol{\theta}) &= E\left[(x_i - \mu_i(\boldsymbol{\theta}))^4|\mathcal{F}_{i-1}^x\right] = \kappa_\varepsilon \exp(4\psi_i).
\end{aligned}$$

Consider the quadratic generalized martingale difference M_i with $\widetilde{q}(m_i(\boldsymbol{\theta})) = m_i(\boldsymbol{\theta})^2$. The martingale difference $m_i(\boldsymbol{\theta})$ and the quadratic martingale difference $M_i(\boldsymbol{\theta})$ are given by

$$\begin{aligned}
m_i(\boldsymbol{\theta}) &= x_i - \mu_i(\boldsymbol{\theta}) = x_i - \mu_\varepsilon \exp(\psi_i) \\
M_i(\boldsymbol{\theta}) &= m_i(\boldsymbol{\theta})^2 - \sigma_i(\boldsymbol{\theta})^2 = (x_i - \mu_\varepsilon \exp(\psi_i))^2 - \sigma_\varepsilon^2 \exp(2\psi_i)
\end{aligned}$$

with the corresponding quadratic variations and quadratic covariation (see (9)-(11))

$$\begin{aligned}
\langle m \rangle_i &= \sigma_\varepsilon^2 \exp(2\psi_i) \\
\langle M \rangle_i &= (\kappa_\varepsilon - \sigma_\varepsilon^4) \exp(4\psi_i) \\
\langle m, M \rangle_i &= \gamma_\varepsilon \exp(3\psi_i).
\end{aligned}$$

Using these results, it is possible to show (see (12)-(14)) that

$$\begin{aligned}
\rho_i^2 &= \frac{\sigma_\varepsilon^2(\kappa_\varepsilon - \sigma_\varepsilon^4)}{\sigma_\varepsilon^2(\kappa_\varepsilon - \sigma_\varepsilon^4) - \gamma_\varepsilon^2} \\
\eta_i &= \frac{\gamma_\varepsilon}{\sigma_\varepsilon^2(\kappa_\varepsilon - \sigma_\varepsilon^4)\exp(3\psi_i)}
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{a}_{i-1}^*(\boldsymbol{\theta}) &= \rho_i^2 \left(-\frac{\partial \mu_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \frac{1}{\langle m \rangle_i} + \frac{\partial \sigma_i^2 \boldsymbol{\theta}}{\partial \boldsymbol{\theta}} \eta_i \right) \\
\mathbf{b}_{i-1}^*(\boldsymbol{\theta}) &= \rho_i^2 \left(\frac{\partial \mu_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \eta_i - \frac{\partial \sigma_i^2 \boldsymbol{\theta}}{\partial \boldsymbol{\theta}} \frac{1}{\langle M \rangle_i} \right).
\end{aligned}$$

Recursive estimates for the parameters $\boldsymbol{\theta} = (\omega, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q)$ in the Log ACD_1 and Log ACD_2 model can then be derived using equations (15)-(16), where ψ_i is defined according to the Log ACD_1 or Log ACD_2 model, respectively. [1]

[1] For computational ease, all partial derivatives in the recursive estimation procedure may be written in terms of $\frac{\partial \psi_i}{\partial \boldsymbol{\theta}}$, where $\frac{\partial \psi_i}{\partial \boldsymbol{\theta}}$ equals $(1, \log x_{i-1}, \ldots, \log x_{i-p}, \psi_{i-1}, \ldots, \psi_{i-q})$ or $(1, \frac{x_{i-1}}{\exp(\psi_{i-1})}, \ldots \frac{x_{i-p}}{\exp(\psi_{i-p})}, \psi_{i-1}, \ldots, \psi_{i-q})$ for the Log ACD_1(p,q) model or Log ACD_2(p,q) model, respectively.

3.2 Initial Values for Parameter Estimation in the Log ACD_1 Model

This section concerns the problem of finding initial values for the recursive estimating equation method given data $\{x_i\}_{i=1}^n$ in the Log ACD_1 context. The basic approach consists of rewriting the Log ACD_1 model as an autoregressive (AR) process, then fitting the AR model to the log-transformed data $\{\log x_i, i = 1, \ldots, n\}$. The AR(m) model has the following form:

$$x_i = \phi_0 + \phi_1 x_{i-1} + \phi_2 x_{i-2} + \ldots + \phi_m x_{i-m} + w_i, \tag{17}$$

where $\phi_0 = \mu(1 - \phi_1 - ... - \phi_m)$ can be considered the intercept of the process. For the purposes of model fitting, w_i is assumed to be independent and identically distributed (i.i.d.) Gaussian white noise.

3.2.1 The Log ACD_1(1,1) Case

Given data $\{x_i\}_{i=1}^n$, we will show that the Log ACD_1(1,1) model can be approximated using an AR(m) process. The Log ACD_1(1,1) model has the following form as per (4):

$$\begin{aligned} x_i &= \exp(\psi_i)\varepsilon_i \\ \psi_i &= \omega + \alpha \log x_{i-1} + \beta\psi_{i-1}, \end{aligned} \tag{18}$$

where $|\alpha+\beta|<1$ and ε_i are i.i.d. and follow some distribution with positive support. The unknown parameters are $\boldsymbol{\theta}=\{\omega,\alpha,\beta\}$. Taking the log transformation of x_i, we have:

$$\log x_i = \psi_i + \log \varepsilon_i. \tag{19}$$

Since the observed data $\{x_i, i=1,\ldots,n\}$ are known, the values of x_i and $\log x_i$ are known. Consider ψ_i:

Theorem 1. *In the Log ACD$_1$(1,1) model with ψ_i given by (18), for $i \geq 2$,*

$$\psi_i = \omega\left(\sum_{t=0}^{i-1}\beta^t\right) + \alpha\left(\sum_{t=0}^{i-2}\beta^t \log x_{i-1-t}\right). \tag{20}$$

Proof. For $i=1$, set $\psi_0=0$ and $x_0=1$. Then,

$$\psi_1 = \omega + \alpha\log x_0 + \beta\psi_0 = \omega.$$

For $i=2$,

$$\begin{aligned}
\psi_2 &= \omega + \alpha\log x_1 + \beta\psi_1 \\
&= \omega + \alpha\log x_1 + \beta\omega \\
&= \omega(1+\beta) + \alpha\beta^0\log x_1 \\
&= \omega\left(\sum_{t=0}^{1}\beta^t\right) + \alpha\left(\sum_{t=0}^{0}\beta^t\log x_{i-1-t}\right),
\end{aligned}$$

and (20) holds. Suppose that for $i = k$, where $k \geq 2$, the following is true:

$$\begin{aligned}\psi_k &= \omega + \alpha \log x_{k-1} + \beta \psi_{k-1} \\ &= \omega \left(\sum_{t=0}^{k-1} \beta^t \right) + \alpha \left(\sum_{t=0}^{k-2} \beta^t \log x_{k-1-t} \right).\end{aligned}$$

Consider $i = k + 1$. Then,

$$\begin{aligned}\psi_{k+1} &= \omega + \alpha \log x_k + \beta \psi_k \\ &= \omega + \alpha \log x_k + \beta \left(\omega \left(\sum_{t=0}^{k-1} \beta^t \right) + \alpha \left(\sum_{t=0}^{k-2} \beta^t \log x_{k-1-t} \right) \right) \\ &= \omega \left(\beta^0 + \sum_{t=0}^{k-1} \beta^{t+1} \right) + \alpha \left(\log x_k + \sum_{t=0}^{k-2} \beta^{t+1} \log x_{k-1-t} \right) \\ &= \omega \left(\beta^0 + \sum_{t=1}^{k} \beta^t \right) + \alpha \left(\beta^0 \log x_{(k+1)-1-0} + \sum_{t=1}^{k-1} \beta^t \log x_{(k+1)-1-t} \right) \\ &= \omega \left(\sum_{t=0}^{k} \beta^t \right) + \alpha \left(\sum_{t=0}^{k-1} \beta^t \log x_{(k+1)-1-t} \right), \textit{and} \\ \psi_i &= \omega \left(\sum_{t=0}^{i-1} \beta^t \right) + \alpha \left(\sum_{t=0}^{i-2} \beta^t \log x_{i-1-t} \right),\end{aligned}$$

which proves Theorem 1. □

Theorem 1 implies that as per (19), given $\{x_i, i = 1, \ldots, n\}$,

$$\begin{aligned}\log x_i &= \omega \left(\sum_{t=0}^{i-1} \beta^t \right) + \alpha \left(\sum_{t=0}^{i-2} \beta^t \log x_{i-1-t} \right) + \log \varepsilon_i \\ &= \omega(1 + \beta + \beta^2 + \ldots + \beta^{i-1}) + \alpha \log x_{i-1} + \alpha\beta \log x_{i-2} + \alpha\beta^2 \log x_{i-3} + \\ &\quad \ldots + \alpha\beta^{i-2} \log x_1 + \log \varepsilon_i.\end{aligned}$$

Recalling the form of the AR model in (17), we define

$$\phi_0 = \omega(1 + \beta + \beta^2 + \ldots + \beta^{i-1}) \tag{21}$$

$$\phi_1 = \alpha, \quad \phi_2 = \alpha\beta, \tag{22}$$

$$\phi_3 = \alpha\beta^2, \quad \ldots, \quad \phi_{i-1} = \alpha\beta^{i-2}$$

$$w_i \approx \log \varepsilon_i$$

so that for $i \geq 2$, the log-transformed data $\log x_i$ can be closely approximated by an AR(i-1) model: [2]

$$\log x_i = \phi_0 + \phi_1 \log x_{i-1} + \phi_2 \log x_{i-2} + \ldots + \phi_{i-1} \log x_1 + w_i.$$

Fitting the full AR(i-1) model to i observations is impractical for large i. However, if the Log $\text{ACD}_1(1,1)$ process is weakly stationary and $|\beta| < 1$, then

$$\lim_{i \to \infty} \beta^{i-1} = 0,$$

so that the coefficients in the AR(i-1) model approach zero as i approaches ∞, and the effect of higher order lags on $\log x_i$ decays geometrically to zero. Since the significance of higher order lags approaches zero, we may approximate the full AR(i-1) model using an

[2] Note that although $\log \varepsilon_i$ is not necessarily Gaussian white noise, $\log \varepsilon_i$ are i.i.d. random variables because ε_i are assumed to be i.i.d. random variables.

AR(m) model with m equal to some 'high enough' order of lags:

$$\log x_i = \phi_0 + \phi_1 \log x_{i-1} + \phi_2 \log x_{i-2} + \ldots + \phi_m \log x_{i-m} + w_i. \tag{23}$$

The coefficients in (23) provide approximations to the equations given in (21)-(22). The first AR coefficient ϕ_1 provides an estimate for α. Looking at the remaining $m-1$ AR coefficients, the j^{th} coefficient provides an approximation of $\alpha\beta^j$, which can be solved to obtain an estimate for β using the estimated value of α. Using the estimate for β, an estimate for ω may be extracted using ϕ_0. Thus, the initial parameter estimate for α is given by:

$$\widehat{\alpha} = \phi_1. \tag{24}$$

In principle, we may derive $m-1$ estimates of β using the coefficients $\phi_2, \ldots, \phi_m$ from (23), and (22). In practice, it suffices for simplicity to construct just one estimate of β using ϕ_2 from (23), and $\widehat{\alpha}$ from (24):

$$\widehat{\beta} = \frac{\phi_2}{\widehat{\alpha}} = \frac{\phi_2}{\phi_1}. \tag{25}$$

Finally, the intercept ϕ_0 of the AR(m) process in (23) approximately equals $\omega(1+\beta+\beta^2+\ldots+\beta^m)$. Since we assume that $|\beta| < 1$, $\sum_{i=0}^{\infty} \beta^i = \frac{1}{1-\beta}$. We use this as an approximation of $(1+\beta+\beta^2+\ldots+\beta^m)$, so that the estimate for ω using $\widehat{\beta}$ from (25) is given by [3]

$$\widehat{\omega} = \phi_0(1-\widehat{\beta}). \tag{26}$$

In the Log ACD_1(1,1) case, there are three unknown parameters requiring three equations to estimate. These three equations are given by the intercept ϕ_0 and first two AR coefficients ϕ_1 and ϕ_2 in (23). Thus, two is the minimum number of lags required to use the AR method for estimating initial parameter values in Log ACD_1(1,1) model.

If $|\beta|$ is close to 1, the coefficients in the full AR(i-1) model will decay more slowly than if $|\beta|$ is close to 0. In this case, a larger number of higher order lags will have a significant effect on $\log x_i$. In order to increase the accuracy of initial parameter estimates, a sufficient number of higher order lags should be included in the AR(m) model. Thus, although $m = 2$ is the minimum number of lags required to estimate $\boldsymbol{\theta} = \{\omega, \alpha, \beta\}$ in the Log ACD_1(1,1) model, including a larger number of lags is desirable to improve the initial parameter estimates. On the other hand, including too many lags in the AR(m) model tends to increase the amount of time required to obtain initial parameter estimates. Heuristic evidence suggests that including $m = 10$ to $m = 20$ lags is enough for finding adequate initial parameter estimates.

3.2.2 The Log ACD_1(2,1) Case

The method of initial parameter estimation derived in the previous section can be applied to the Log ACD_1(2,1) case. Initial parameter estimation in the Log ACD_1(2,1) case proceeds similarly to estimation in the Log ACD_1(1,1) case. Given data $\{x_i\}_{i=1}^n$, the Log ACD_1(2,1)

[3] Although the property $\sum_{i=0}^{N-1} \beta^i = \frac{1-\beta^N}{1-\beta}$ suggests that another viable estimate for ω may be $\phi_0 \frac{1-\widehat{\beta}^{m+1}}{1-\widehat{\beta}}$, heuristic evidence suggests that $\phi_0(1-\widehat{\beta})$ provides a better approximation for ω.

model has the following form as per (4):

$$\begin{aligned} x_i &= \exp(\psi_i)\varepsilon_i \\ \psi_i &= \omega + \alpha_1 \log x_{i-1} + \alpha_2 \log x_{i-2} + \beta\psi_{i-1}, \end{aligned} \tag{27}$$

with assumptions for weak stationarity and non-negativity given in (4). The unknown parameters are $\boldsymbol{\theta} = \{\omega, \alpha_1, \alpha_2, \beta\}$. After taking the log transformation of x_i (see (19)), it can be shown that:

Theorem 2. *In the Log ACD$_1$(2,1) model with ψ_i given by (27), for $i \geq 3$,*

$$\psi_i = \omega \left(\sum_{t=0}^{i-1} \beta^t \right) + \alpha_1 \log x_{i-1} + \left(\sum_{t=0}^{i-3} (\alpha_1 \beta^{t+1} + \alpha_2 \beta^t) \log x_{i-2-t} \right). \tag{28}$$

Proof. Set $\psi_0 = 0$ and $x_{-1} = x_0 = 1$. Then, for $i = 1$,

$$\psi_1 = \omega.$$

For $i = 2$,

$$\psi_2 = \omega(1 + \beta) + \alpha_1 \log x_1$$

For $i = 3$,

$$\begin{aligned} \psi_3 &= \omega + \alpha_1 \log x_2 + \alpha_2 \log x_1 + \beta\psi_2 \\ &= \omega + \alpha_1 \log x_2 + \alpha_2 \log x_1 + \beta(\omega(1 + \beta) + \alpha_1 \log x_1) \end{aligned}$$

$$
\begin{aligned}
&= \omega(1+\beta+\beta^2) + \alpha_1 \log x_2 + (\alpha_1\beta + \alpha_2)\log x_1 \\
&= \omega\left(\sum_{t=0}^{2}\beta^t\right) + \alpha_1 \log x_{i-1} + \left(\sum_{t=0}^{0}(\alpha_1\beta^{t+1} + \alpha_2\beta^t)\log x_{i-2-t}\right),
\end{aligned}
$$

and (28) holds. Suppose that for $i = k$, where $k \geq 3$, the following is true:

$$
\psi_k = \omega\left(\sum_{t=0}^{k-1}\beta^t\right) + \alpha_1 \log x_{k-1} + \left(\sum_{t=0}^{k-3}(\alpha_1\beta^{t+1} + \alpha_2\beta^t)\log x_{k-2-t}\right).
$$

Consider $i = k+1$. Then,

$$
\begin{aligned}
\psi_{k+1} &= \omega + \alpha_1 \log x_k + \alpha_2 \log x_{k-1} + \beta\psi_k \\
&= \omega + \alpha_1 \log x_k + \alpha_2 \log x_{k-1} + \\
&\quad \beta\left(\omega\left(\sum_{t=0}^{k-1}\beta^t\right) + \alpha_1 \log x_{k-1} + \left(\sum_{t=0}^{k-3}(\alpha_1\beta^{t+1} + \alpha_2\beta^t)\log x_{k-2-t}\right)\right) \\
&= \omega\left(1 + \sum_{t=0}^{k-1}\beta^{t+1}\right) + \alpha_1 \log x_k + \\
&\quad \left((\alpha_1\beta + \alpha_2)\log x_{k-1} + \left(\sum_{t=0}^{k-3}(\alpha_1\beta^{t+2} + \alpha_2\beta^{t+1})\log x_{k-2-t}\right)\right) \\
&= \omega\left(\beta^0 + \sum_{t=1}^{k}\beta^t\right) + \alpha_1 \log x_k + \\
&\quad \left((\alpha_1\beta^1 + \alpha_2\beta^0)\log x_{k-1} + \left(\sum_{t=1}^{k-2}(\alpha_1\beta^{t+1} + \alpha_2\beta^t)\log x_{k-1-t}\right)\right) \\
&= \omega\left(\sum_{t=0}^{(k+1)-1}\beta^t\right) + \alpha_1 \log x_{i-1} + \left(\sum_{t=0}^{(k+1)-3}(\alpha_1\beta^{t+1} + \alpha_2\beta^t)\log x_{(k+1)-2-t}\right), \textit{and} \\
\psi_i &= \omega\left(\sum_{t=0}^{i-1}\beta^t\right) + \alpha_1 \log x_{i-1} + \left(\sum_{t=0}^{i-3}(\alpha_1\beta^{t+1} + \alpha_2\beta^t)\log x_{i-2-t}\right),
\end{aligned}
$$

which proves Theorem 2. □

Recall the definition of the AR model in (17). Theorem 2 implies that given durations $\{x_i, i = 1, 2, ...\}$ from the Log ACD$_2$(2,1) model, for $i \geq 3$, $\log x_i$ may be approximated by an AR(i-1) model with the intercept and first three AR coefficients defined as

$$\phi_0 = \omega(1 + \beta + \beta^2 + \ldots + \beta^{i-1}) \tag{29}$$

$$\phi_1 = \alpha_1, \quad \phi_2 = \alpha_1\beta + \alpha_2, \quad \phi_3 = \alpha_1\beta^2 + \alpha_2\beta \tag{30}$$

$$w_i \approx \log \varepsilon_i.$$

Again, higher order lags depend upon β. Assuming that $|\beta| < 1$, the coefficients in the AR(i-1) representation of $\log x_i$ approach zero as i approaches ∞, so we may approximate the full AR(i-1) model using an AR(m) model with an intercept term and a minimum of three lags. The form of the AR(m) model is given in (23). Writing the intercept and first three lags of the AR(m) model as functions of the four unknown parameters (see (29)-(30)) provides the four equations that are sufficient for finding $\boldsymbol{\theta}$.

Fitting an AR(m), $m \geq 3$ model to the data and using the AR coefficients to solve for

the unknown parameters $\boldsymbol{\theta} = \{\omega, \alpha_1, \alpha_2, \beta\}$ yields the following results:[4]

$$\widehat{\alpha_1} = \phi_1 \tag{31}$$

$$\widehat{\alpha_2} = \phi_2 - \phi_1 \frac{\phi_3}{\phi_2} \tag{32}$$

$$\widehat{\beta} = \frac{\phi_3}{\phi_2} \tag{33}$$

$$\widehat{\omega} = \phi_0(1 - \widehat{\beta}) \tag{34}$$

3.2.3 Higher Order Cases

Note that in the Log ACD$_1$(1,1) case with $p = 1$ and $q = 1$, initial parameter estimates may be derived by fitting an AR(m) model with $m \geq 2$ lags to the log-transformed duration data. Similarly, in the Log ACD$_1$(2,1) case with $p = 2$ and $q = 1$, initial parameter estimates are derived via an AR(m) model with $m \geq 3$ lags. We postulate that initial parameter estimates may be derived for the general Log ACD$_1$(p,q) case by fitting an AR(m) model to log-transformed duration data, where $m \geq p + q$.

For the general Log ACD$_1$(p,q) case, there are $1 + p + q$ unknown parameters denoted by $\boldsymbol{\theta} = \{\omega, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q\}$. Thus, $1 + p + q$ equations are required to solve for the unknown parameters. We propose that these $1 + p + q$ equations can be obtained by writing the intercept and the first $p + q$ lags of the AR(m) model in terms of the unknown parameters $\boldsymbol{\theta}$. The formulas for higher order lags need not be specified, though including

[4] The value of $\widehat{\alpha_1}$ follows directly from (30). To find $\widehat{\alpha_2}$ and $\widehat{\beta}$, solve the equations $\phi_2 = \alpha_1\beta + \alpha_2$ and $\phi_3 = \alpha_1\beta^2 + \alpha_2\beta$ for α_2 and β using the value obtained for $\widehat{\alpha_1}$. Once the value for $\widehat{\beta}$ is found, ω may be approximated using $\phi_0(1 - \widehat{\beta})$ by reasoning analogous to the Log ACD$_1$(1,1) case.

a sufficient number of higher order lags may improve initial parameter estimates when $|\beta|$ is close to 1.

3.3 Forecasting

It is often desirable to assess the predictive capabilities of models fitted to time series data. A duration model fitted to a set of duration data can be used to predict l-step-ahead future durations, $l = 1, ..., L$, given data up to the h. Then, h is called the forecast origin. The calibration or fitting portion of the data included data up to the origin hth duration. Observations on the actual values of durations are recorded for $h+l = \{h+1, h+2, ..., h+L\}$, and are referred to as hold-out observations; they are useful for predictive cross-validation. The predicted future durations in the hold-out period are then compared to the observed future durations to determine assess the predictive capability of the model.

One way to compare predicted future durations to observed future durations is to use the mean absolute percentage error (MAPE) formula. Before defining the MAPE, we first introduce some notation. Denote the l-step-ahead forecast with forecast origin h by $x_h(l)$, where $l = 1, ..., L$. The fitted model will yield L forecasts from forecast origin h: $\{x_h(1), x_h(2), \ldots, x_h(L)\}$. Denote the corresponding observed values by $\{x_{h+1}, x_{h+2}, \ldots, x_{h+L}\}$. The MAPE is defined as:

$$MAPE = 100\frac{1}{L}\sum_{l=1}^{L}\left|\frac{x_{h+l} - x_h(l)}{x_{h+l}}\right|. \tag{35}$$

The MAPE is commonly used as a measure of the forecast adequacy of the fitted model.

It can be used as a method to compare multiple models fitted to the same data. The model with the lowest MAPE exhibits the lowest amount of errors in forecasts. To demonstrate how the MAPE is obtained, the process is shown below for the ACD(1,1) and Log ACD_1(1,1) models:

In the first case, suppose that h observations over time, $\{x_1, x_2, \ldots, x_h\}$ are given. An ACD(1,1) model is fitted to this set of data. In the ACD(1,1) model, the 1-step-ahead forecast is given by:

$$x_h(1) = E[x_{h+1}|\mathcal{F}_h^x] = \psi_{h+1}.$$

By definition, $\psi_i = E[x_i|\mathcal{F}_{i-1}^x]$. For $i = 1, ..., h$, x_i is a known constant. So, $\psi_i = E[x_i|\mathcal{F}_{i-1}^x] = x_i$ for $i = 1, ..., h$. This implies that ψ_{h+1} is known for the $h + 1^{th}$ observation:

$$\begin{aligned} \psi_{h+1} &= \omega + \alpha x_h + \beta\psi_h \\ &= \omega + \alpha x_h + \beta x_h \\ &= \omega + (\alpha + \beta)x_h. \end{aligned}$$

Using this information, multi-step-ahead forecasts from origin h may be derived by substituting $x_h(1)$ and ψ_{h+1} into the model as follows:

$$x_h(2) = \omega + \alpha x_h(1) + \beta\psi_{h+1}$$

$$
\begin{aligned}
x_h(3) &= \omega + \alpha x_h(2) + \beta\psi_{h+2} \\
&\dots \\
x_h(L) &= \omega + \alpha x_h(L-1) + \beta\psi_{L-1}.
\end{aligned}
$$

Given the values of the L forecasts $\{x_h(1), x_h(2), \dots, x_h(L)\}$ from forecast origin h and their corresponding observed values $\{x_{h+1}, x_{h+2}, \dots, x_{h+L}\}$, the MAPE is defined as by (35).

In the second case, suppose that h durations, $\{x_1, x_2, \dots, x_h\}$ are given. A Log ACD(1,1) model is fitted to this set of data. In the Log ACD_1 models, $\psi_i = \log E[x_i|\mathcal{F}_{i-1}^x]$ Bauwens and Giot (2000). Thus, the 1-step-ahead forecast is given by:

$$
x_h(1) = E[x_{h+1}|\mathcal{F}_h^x] = \exp(\psi_{h+1}).
$$

For $i = 1, ..., h$, x_i is a known constant. So, $\psi_i = \log E[x_i|\mathcal{F}_{i-1}^x] = \log x_i$ is known for $i = 1, ..., h$. This implies that ψ_{h+1} is known for the $h + 1^{th}$ observation:

$$
\begin{aligned}
\psi_{h+1} &= \omega + \alpha \log x_h + \beta\psi_h \\
&= \omega + \alpha \log x_h + \beta \log x_h \\
&= \omega + (\alpha + \beta) \log x_h.
\end{aligned}
$$

Using this information, multi-step-ahead forecasts from origin h may be derived by substi-

tuting $x_h(1)$ and ψ_{h+1} into the model as follows:

$$\begin{aligned} x_h(2) &= \exp\{\omega + \alpha \log x_h(1) + \beta\psi_{h+1}\} \\ x_h(3) &= \exp\{\omega + \alpha \log x_h(2) + \beta\psi_{h+2}\} \\ &\cdots \\ x_h(L) &= \exp\{\omega + \alpha \log x_h(L-1) + \beta\psi_{L-1}\}. \end{aligned}$$

Given the values of the L forecasts $\{x_h(1), x_h(2), \ldots, x_h(L)\}$ from forecast origin h and their corresponding observed values $\{x_{h+1}, x_{h+2}, \ldots, x_{h+L}\}$, the MAPE is defined as by (35).

4 Monte Carlo Simulation Study

4.1 Log $ACD_1(1,1)$ Model

Monte Carlo simulations are conducted to demonstrate the process of finding initial values for recursive EE estimation using observations generated using the Log $ACD_1(1,1)$ model in (18). We generate $L = 100$ sets of data, each of length $n = 4000$, from the Log $ACD_1(1,1)$ model. The $exponential(1)$ and $gamma(2, 0.5)$ error specifications are examined. Each set of data is then log-transformed, and initial parameter estimates are found by fitting an AR(m) model to the log-transformed data. We set $m = 10$. A sample plot of duration data simulated from the Log $ACD_1(1,1)$ model is shown in Figure 1.

Figure 1: One realization of the Log $ACD_1(1,1)$ process with $\boldsymbol{\theta} = \{1, 0.5, 0.3\}$ and $\varepsilon_i \sim exponential(1)$

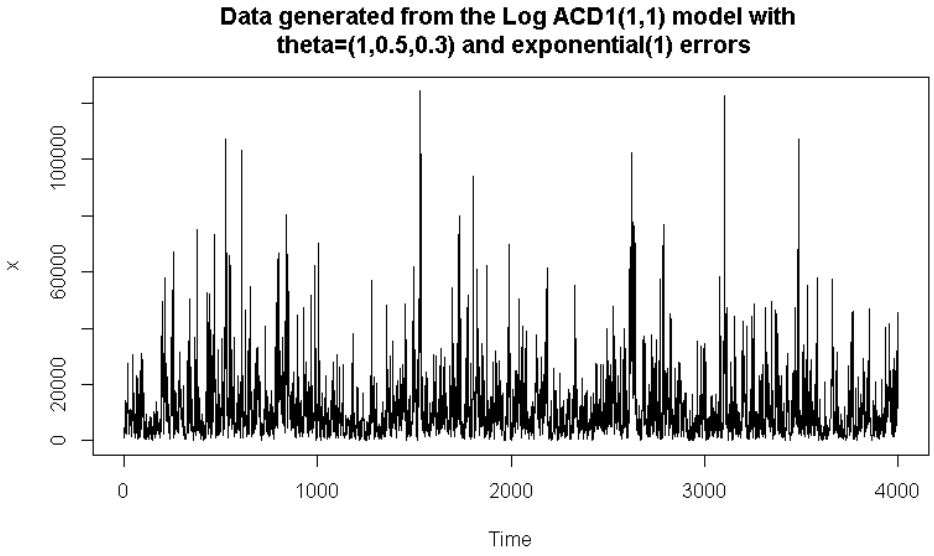


Table 1 shows the percentiles of initial parameter estimates for four different setups

of the Log $ACD_1(1,1)$ model. Table 2 shows the percentiles of the corresponding EE parameter estimates using the initial parameter estimates shown in Table 1. Table 3 shows recursive EE parameter estimates where the initial parameter estimates are not specified exactly, but are generated from a uniform distribution over the interval $(-5, 5)$ for ω and from within the interval $(-1, 1)$ for α and β. The true parameters and ε_i distributions corresponding to each setup are given in each table.

Table 1: Percentiles of the AR(m) initial parameter estimates for data generated from the Log $ACD_1(1,1)$ model: $n = 4000, L = 100$

True Parameters	Estimates				
	5th	25th	50th	75th	95th
$\omega = 1.0$	0.9314	1.0047	1.0464	1.0767	1.1297
$\alpha = 0.5$	0.4766	0.4864	0.4985	0.5082	0.5242
$\beta = 0.3$	0.2447	0.2771	0.3041	0.3291	0.3765
$\varepsilon \sim exponential(1)$					
$\omega = 3.0$	1.8200	1.9458	2.0575	2.1503	2.2572
$\alpha = 0.2$	0.1768	0.1863	0.1990	0.2081	0.2243
$\beta = -0.4$	-0.5414	-0.4482	-0.3984	-0.3482	-0.2575
$\varepsilon \sim exponential(1)$					
$\omega = 2.0$	1.7473	2.3495	2.6882	3.1356	3.5879
$\alpha = -0.1$	-0.1271	-0.1162	-0.1023	-0.0928	-0.0771
$\beta = 0.23$	0.0206	0.1390	0.2627	0.3544	0.5182
$\varepsilon \sim gamma(2, 0.5)$					
$\omega = -1.5$	-1.1350	-0.9579	-0.8184	-0.7227	-0.5216
$\alpha = -0.2$	-0.2272	-0.2161	-0.2022	-0.1927	-0.1771
$\beta = 0.65$	0.5320	0.6048	0.6573	0.7037	0.7864
$\varepsilon \sim gamma(2, 0.5)$					

Table 2: Percentiles of recursive EE estimates for data generated from the Log $ACD_1(1,1)$ model with initial values given in Table 1: $n = 4000, L = 100$

True Parameters	Estimates				
	5th	25th	50th	75th	95th
$\omega = 1$	0.9315	1.0047	1.0464	1.076	1.1298
$\alpha = 0.5$	0.4766	0.4864	0.4985	0.5082	0.5242
$\beta = 0.3$	0.2447	0.2771	0.3041	0.3291	0.3765
$\varepsilon \sim exponential(1)$					
$\omega = 3.0$	1.7968	1.9477	2.0542	2.1455	2.3128
$\alpha = 0.2$	0.1745	0.1845	0.1984	0.2084	0.2244
$\beta = -0.4$	-0.5565	-0.44277	-0.3965	-0.3467	-0.2515
$\varepsilon \sim exponential(1)$					
$\omega = 2.0$	1.7472	2.3495	2.6884	3.1358	3.5857
$\alpha = -0.1$	-0.1271	-0.1161	-0.1023	-0.0929	-0.0771
$\beta = 0.23$	0.02149	0.1389	0.2627	0.3544	0.5183
$\varepsilon \sim gamma(2, 0.5)$					
$\omega = -1.5$	-1.6290	-1.1735	-1.026	-0.7689	-0.4502
$\alpha = -0.2$	-0.5298	-0.2579	-0.1765	-0.1342	0.3892
$\beta = 0.65$	-0.1005	0.4939	0.5991	0.7621	0.9670
$\varepsilon \sim gamma(2, 0.5)$					

Table 3: Percentiles of recursive EE parameter estimates for data generated from the Log $ACD_1(1,1)$ model with initial values generated from a uniform distribution over some interval containing the true parameters: $n = 4000, L = 100$

True Parameters	Estimates				
	5th	25th	50th	75th	95th
$\omega = 1$	-4.4974	-2.6814	0.2595	3.0899	4.6259
$\alpha = 0.5$	-0.8006	-0.4146	-0.1634	0.46327	0.8672
$\beta = 0.3$	-0.6113	-0.3139	-0.1195	0.0718	0.4419
$\varepsilon \sim exponential(1)$					
$\omega = -3.0$	-4.1865	-2.3093	0.6420	2.9731	4.6200
$\alpha = 0.2$	-0.8116	-0.4301	0.0148	0.5162	0.8740
$\beta = -0.6$	-0.6929	-0.2454	-0.0549	0.1049	0.4227
$\varepsilon \sim exponential(1)$					
$\omega = 2.0$	-4.3328	-2.1014	0.0815	2.6086	4.5096
$\alpha = -0.1$	-0.8593	-0.4980	-0.0258	0.4552	0.9415
$\beta = 0.23$	-0.4448	-0.1879	0.0024	0.2660	0.6245
$\varepsilon \sim gamma(2, 0.5)$					
$\omega = -1.5$	-4.3314	-2.0190	0.1101	2.2028	4.5012
$\alpha = -0.2$	-0.9554	-0.5223	-0.0828	0.3056	0.8609
$\beta = 0.65$	-0.4474	-0.1355	0.0153	0.2531	0.6196
$\varepsilon \sim gamma(2, 0.5)$					

The results in Table 3 suggest that the recursive EE parameter estimation method is highly sensitive to initial values. The optimal recursive EE parameter estimates derived using initial values generated from uniform distributions do not always converge to the original true parameters. Table 1 suggests that the AR(m) method of finding initial values seems to provide a reasonable starting point for parameter estimation through the estimating equations method. The AR(m) initial parameter estimates are close to the true simulation parameters, and the optimal recursive EE parameter estimates remain close to the initial values. The recursive estimating equations procedure may be considered as a

way to refine the initial parameter estimates. This idea is explored further in the context of the Log ACD$_1$(2,1) model.

4.2 Log ACD$_1$(2,1) Model

We demonstrate the parameter estimation process in the Log ACD$_1$(2,1) model given by (27). Again, we generate $L = 100$ sets of data, each of length $n = 4000$, from the Log ACD$_1$(2,1) model. The $exponential(1)$ and $gamma(2, 0.5)$ error specifications are examined. Each set of data is log-transformed, and initial parameter estimates are found by fitting an AR(m) model to the log-transformed data, with $m = 15$. The estimating equations approach is then applied to the data. A sample plot of duration data simulated from the Log ACD$_1$(2,1) model is shown in Figure 2.

Table 4 shows the percentiles of initial parameter estimates for two different setups of the Log ACD$_1$(2,1) model. Table 5 shows the percentiles of the corresponding EE estimates using the initial parameter estimates in Table 4. Table 6 shows recursive EE estimates using initial values generated from a uniform distribution over some interval containing the true parameter values.[5] The true parameters and ε_i distributions corresponding to each setup are given in each table.

[5] In setup 1, initial parameter estimates for $\{\omega, \alpha_1, \alpha_2, \beta\}$ are generated from uniform distributions on the intervals (0,12), (0, 0.5), (-1,0), and (0,0.2), respectively. In setup 2, the intervals used are (4,8), (-0.3,0), (-0.9,-0.5), (-0.3,0), respectively.

Figure 2: One realization of the Log ACD$_1$(2,1) process with $\boldsymbol{\theta} = \{5, -0.5, -0.6, -0.2\}$ and $\varepsilon_i \sim gamma(2, 0.5)$

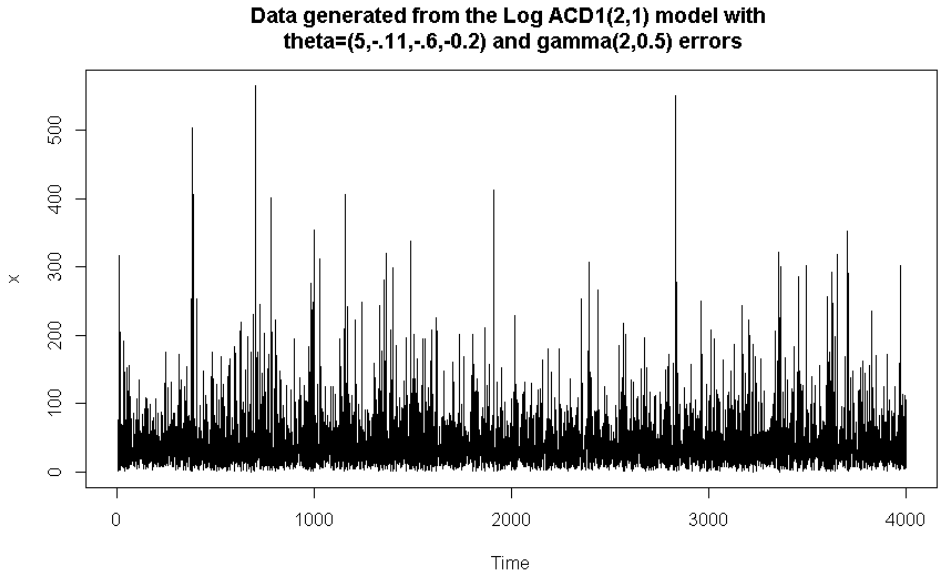


Table 4: Percentiles of the AR(m) initial parameter estimates for data generated from the Log ACD$_1$(2,1) model: $n = 4000, L = 100$

True Parameters	Estimates				
	5th	25th	50th	75th	95th
$\omega = 10.0$	8.5922	8.9632	9.1487	9.3766	9.7950
$\alpha_1 = 0.1$	0.0773	0.0862	0.0989	0.1080	0.1247
$\alpha_2 = -0.5$	-0.5199	-0.5084	-0.4993	-0.4906	-0.4700
$\beta = 0.06$	0.0103	0.0461	0.0675	0.0930	0.1258
$\varepsilon \sim exponential(1)$					
$\omega = 5$	6.4146	6.5580	6.6613	6.8095	6.9712
$\alpha_1 = -0.11$	-0.1371	-0.1260	-0.1124	-0.1031	-0.0873
$\alpha_2 = -0.6$	-0.6309	-0.6149	-0.6017	-0.5909	-0.5772
$\beta = -0.2$	-0.2493	-0.2104	-0.1771	-0.1641	-0.1368
$\varepsilon \sim gamma(2, 0.5)$					

Table 5: Percentiles of recursive EE parameter estimates for data generated from the Log $\text{ACD}_1(2,1)$ model with initial values given in Table 4: $n = 4000, L = 100$

True Parameters	Estimates				
	5th	25th	50th	75th	95th
$\omega = 10.0$	8.5145	8.9398	9.1336	9.2678	9.5829
$\alpha_1 = 0.1$	0.0767	0.0862	0.1099	0.1088	0.1264
$\alpha_2 = -0.5$	-0.5176	-0.5058	-0.4980	-0.4921	-0.4736
$\beta = 0.06$	0.0226	0.0578	0.0709	0.0941	0.1277
$\varepsilon \sim exponential(1)$					
$\omega = 5$	6.4146	6.5580	6.6613	6.8095	6.9713
$\alpha_1 = -0.11$	-0.1371	-0.1260	-0.1124	-0.1031	-0.0873
$\alpha_2 = -0.6$	-0.6308	-0.6149	-0.6017	-0.5909	-0.5772
$\beta = -0.2$	-0.2493	-0.2104	-0.1771	-0.1641	-0.1368
$\varepsilon \sim gamma(2, 0.5)$					

Table 6: Percentiles of recursive EE parameter estimates for data generated from the Log $\text{ACD}_1(2,1)$ model with initial values generated from a uniform distribution on some interval containing the true parameters: $n = 4000, L = 100$

True Parameters	Estimates				
	5th	25th	50th	75th	95th
$\omega = 10.0$	8.5823	8.9797	9.1493	9.3908	9.7978
$\alpha_1 = 0.1$	0.0771	0.0867	0.0995	0.1079	0.1246
$\alpha_2 = -0.5$	-0.5191	-0.5086	-0.4994	-0.4906	-0.4701
$\beta = 0.06$	0.0108	0.0463	0.0657	0.0932	0.1262
$\varepsilon \sim exponential(1)$					
$\omega = 5$	4.2240	5.0039	5.8379	6.9474	7.7230
$\alpha_1 = -0.11$	-0.2827	-0.2076	-0.1316	-0.0573	-0.0154
$\alpha_2 = -0.6$	-0.8928	-0.7882	-0.6549	-0.5773	-0.5062
$\beta = -0.2$	-0.2845	-0.2108	-0.1562	-0.0783	-0.0116
$\varepsilon \sim gamma(2, 0.5)$					

As in the Log $ACD_1(1,1)$ case, Table 3 and Table 4 show that the optimal recursive EE parameter estimates for $\boldsymbol{\theta} = \{\omega, \alpha_1, \alpha_2, \beta\}$ remain close to the initial values derived from the AR(m) method. In both setups, the initial parameter estimates and recursive EE estimates with AR(m) initial values are fairly close to the true simulation parameters.

Figure 3 and Figure 4 show histograms of the AR(m) initial parameter estimates and the recursive EE estimates generated using AR(m) initial values across $L = 100$ simulations. The histograms emphasize the similarities between parameter estimates generated by the two methods. For comparison, Figure 5 and Figure 6 show histograms of the recursive EE estimates generated using initial values drawn from uniform distributions are shown for setup 1 and setup 2.

Figure 5 and Figure 6 emphasize the importance of finding suitable initial values in the recursive estimating equations method. In setup 1, most initial values were drawn from uniform distributions over fairly wide intervals. Figure 5 shows that the resulting parameter estimates have a fairly wide spread, and do not appear to be centered around the true simulation parameters. By contrast, in setup 2, initial values were drawn from uniform distributions over smaller intervals containing the true parameters. Figure 6 shows that the resulting parameter estimates have a much smaller spread, and are centered around the true simulation parameters. These results suggest that the recursive EE method works well with suitable initial values, but may not provide good parameter estimates if the initial values are off range.

Figure 3: Histograms of AR(m) initial parameter estimates vs. recursive EE parameter estimates for setup 1 with $\boldsymbol{\theta} = \{10, 0.1, -0.5, 0.06\}$ and $\varepsilon_i \sim exponential(1)$

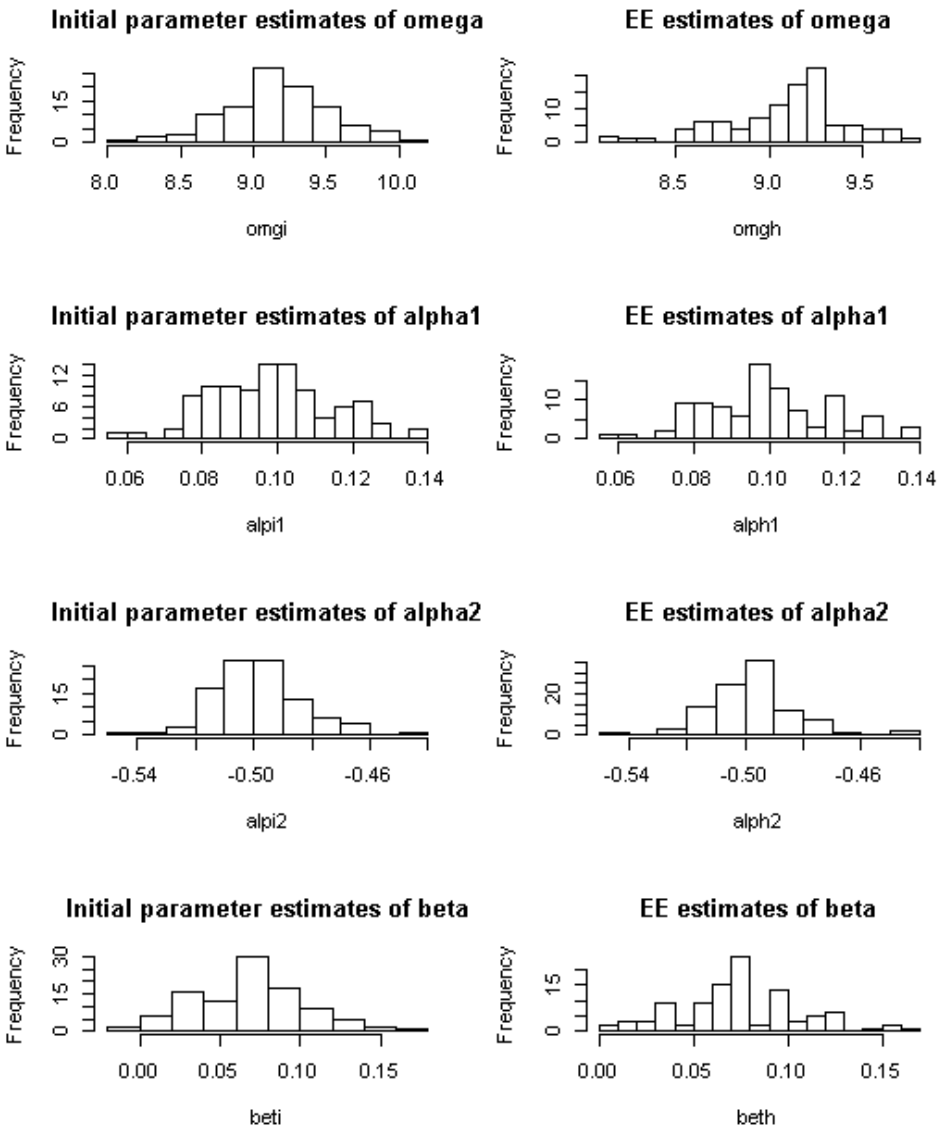


Figure 4: Histograms of AR(m) initial parameter estimates vs. recursive EE parameter estimates for setup 2 with $\boldsymbol{\theta} = \{5, -0.5, -0.6, -0.2\}$ and $\varepsilon_i \sim gamma(2, 0.5)$

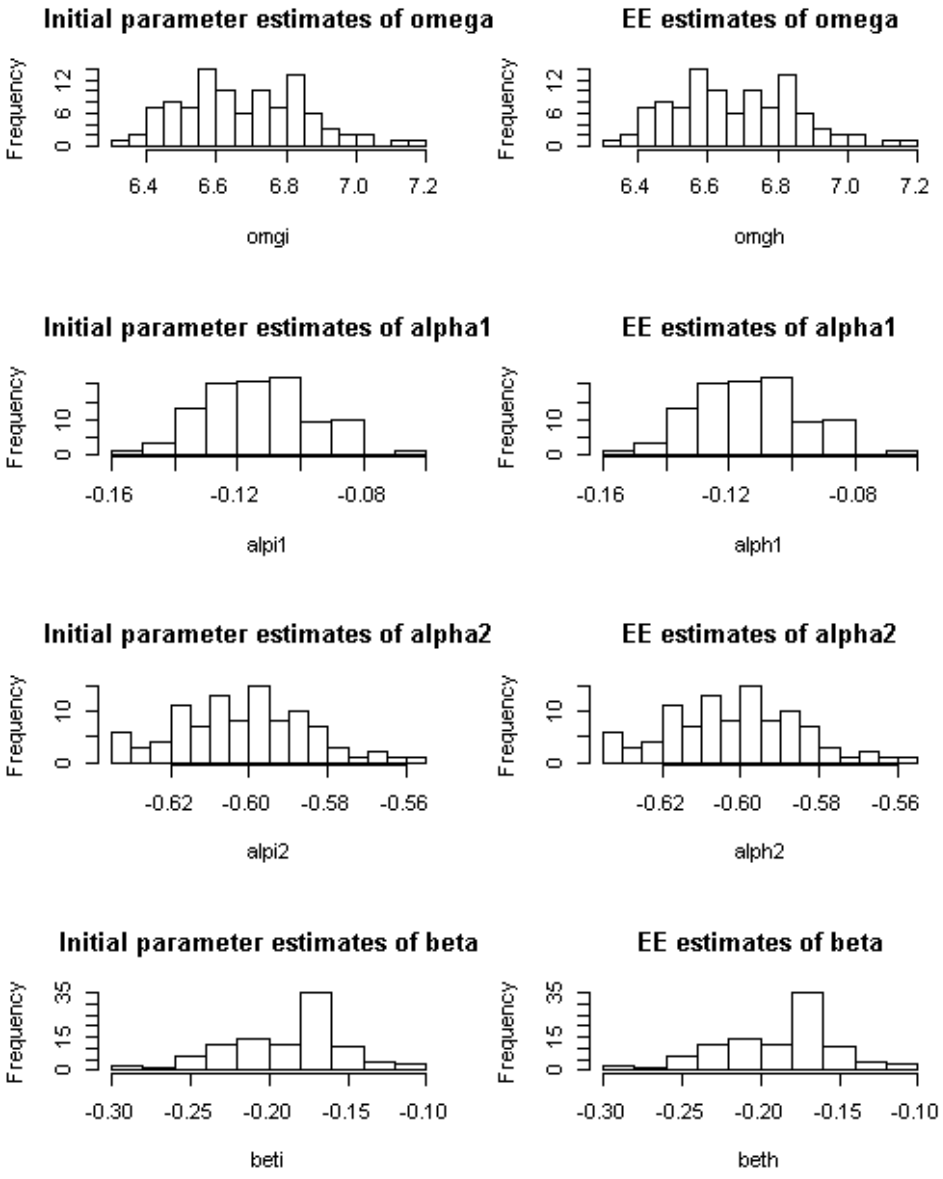


Figure 5: Histograms of recursive EE parameter estimates with initial values drawn from a uniform distribution for setup 1 with $\boldsymbol{\theta} = \{10, 0.1, -0.5, 0.06\}$ and $\varepsilon_i \sim exponential(1)$

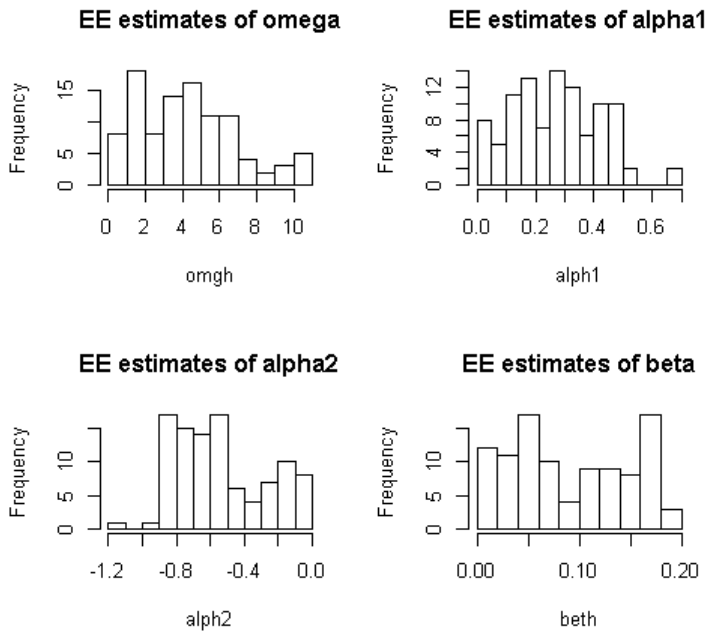
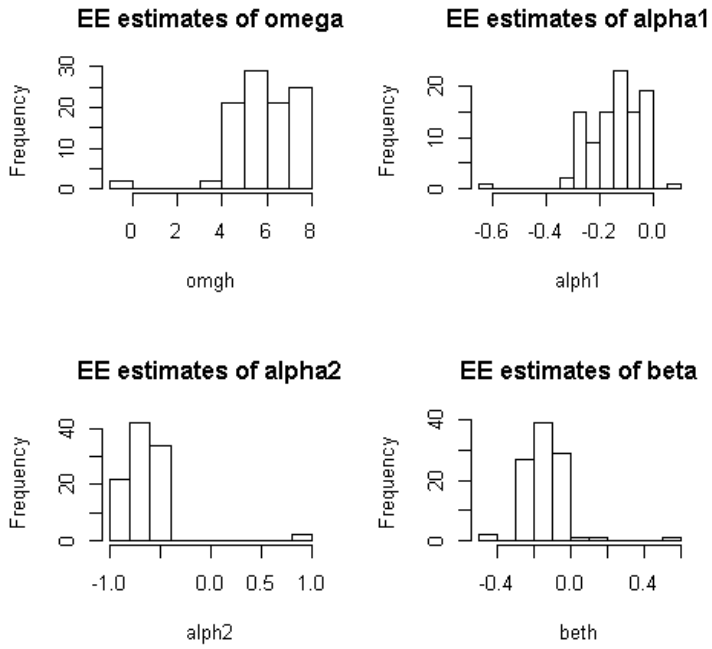


Figure 6: Histograms of recursive EE parameter estimates with initial values drawn from a uniform distribution for setup 2 with $\boldsymbol{\theta} = \{5, -0.5, -0.6, -0.2\}$ and $\varepsilon_i \sim gamma(2, 0.5)$



5 An Application to IBM Stock Data

5.1 Data Description

High frequency transaction-by-transaction IBM stock data was collected from the NASDAQ stock exchange over the course of the day on September 13, 2012. The raw data contained information on the time, price, and volume of each trade from 9:30:00am to 4:00:00pm. Given this data, we defined an event as a change in the IBM stock price of greater than or equal to \$0.0125. By recording the time at which each event occurred, we obtained a series of arrival times. Taking the difference of the arrival times and filtering out any zero-valued durations, we obtained a time series of 2786 observed durations.

Figure 7: Durations in raw IBM data. An event occurs when $\Delta price \geq \$0.0125$.

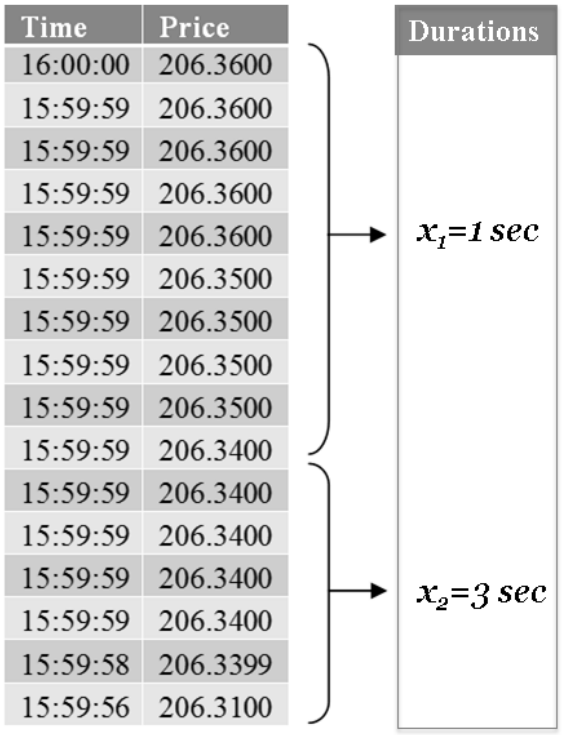
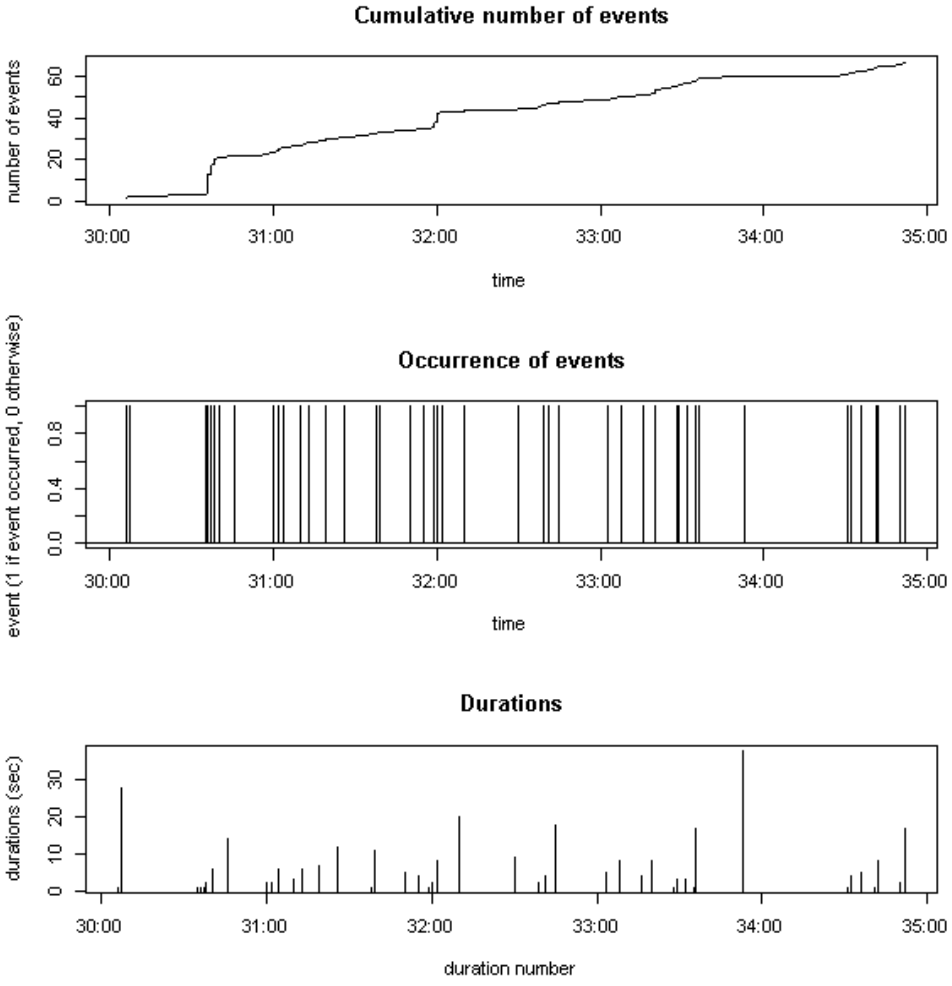


Figure 8 contains several plots of the IBM stock point process. Defining an event as $\Delta price \geq \$0.0125$, the cumulative number of events over the first five minutes of trading is shown on the topmost plot. The time at which each event occurs is indicated in the middle plot. The bottom plot shows the duration of time between every two event occurrences.

Figure 8: IBM stock data



The full time series of durations extracted from the raw IBM data is shown in Figure 9. The autocorrelation function (ACF) of the process shows a high degree of persistence. Note that durations near the start and close of the trading day tend to be shorter, while durations near midday tend to be longer. This is consistent with the literature on financial durations, and suggests that trading occurs more rapidly toward the start and end of each trading day (Engle and Russell , 1998). In addition, the series of durations shows a considerable amount of clustering, suggesting that periods of rapid trading are likely to be followed by rapid trading, while periods of slow trading are likely to be followed by slow trading.

Figure 9: Durations extracted from the raw IBM data

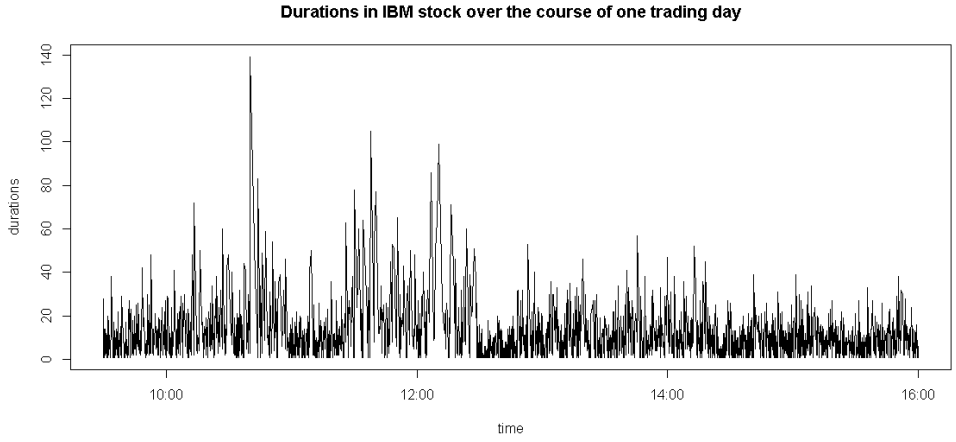
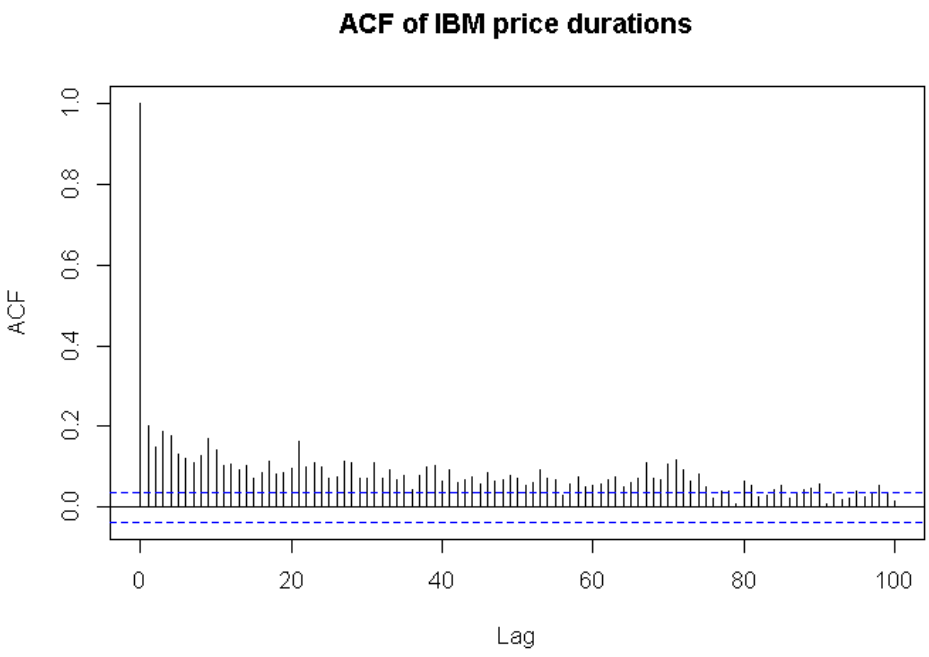


Figure 10: ACF of the duration data



5.2 Modeling IBM Price Durations

Each of the duration models mentioned in this paper seeks to account for the clustering and time dependence patterns seen in duration data. To demonstrate the use of parameter estimation techniques in duration models, we fit the ACD and Log ACD_1 duration models to the IBM duration data.

We fit an ACD(1,1) model to the data via the maximum likelihood method (see section 2.2.1), using R code posted by Ruey S. Tsay (2011). We assume that the errors are i.i.d. with an *exponential*(1) distribution, and we reserve 5 observations for later use to evaluate the MAPE of the model.

The results of the ML method are shown below:

```
Maximized log-likehood:  8507.623

Coefficient(s):
       Estimate  Std. Error  t value   Pr(>|t|)
omega 0.1246974   0.0422328  2.95262  0.0031509 **
alpha 0.0905070   0.0113037  8.00685 1.1102e-15 ***
beta  0.8958736   0.0134043 66.83471 < 2.22e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

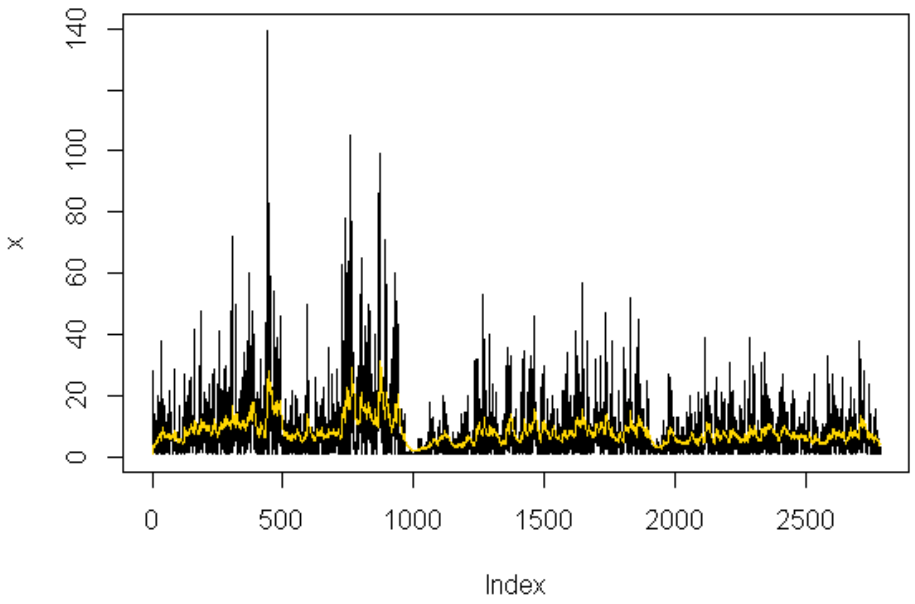
and the fitted model is:

$$\begin{aligned} \widehat{x_i} &= \widehat{\psi_i} \\ \widehat{\psi_i} &= 0.12469 + 0.09051 x_{i-1} + 0.89588 \psi_{i-1}. \end{aligned}$$

A plot of the fitted model against the IBM price durations is given in Figure 11. The

fitted model was initialized using the first observed duration. Engle and Russell (1998) also suggested using the value 1.0 or the average of the first few durations to initialize the process.

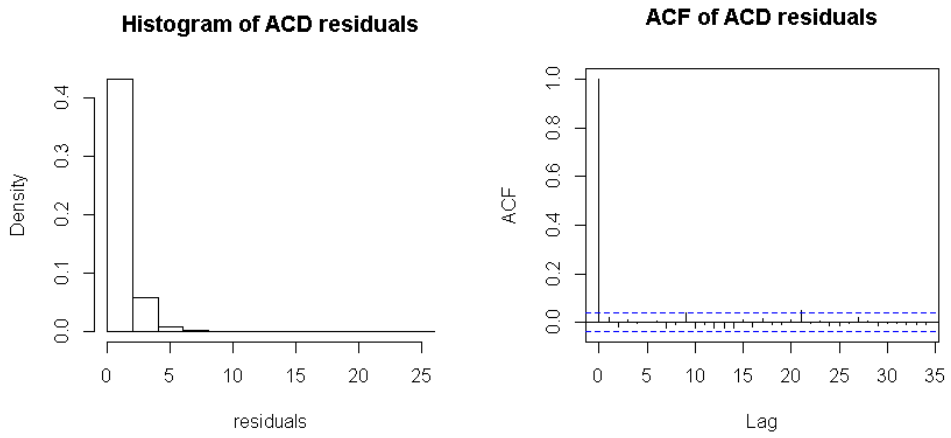
Figure 11: ACD model (gold) fitted to IBM duration data (black)



Given the fitted model, the residuals are given by $\widehat{\varepsilon_i} = x_i/\widehat{\psi_i}$ (Tsay 2007). The histogram of the residuals suggests that the residuals are exponentially distributed with a mean of approximately 1. The ACF of the residuals shows no significant higher order lags.

Using equation (35), the MAPE calculated using five forecasts from the fitted ACD(1,1) model and the last five observed durations is found to be: 88.49161.

Figure 12: Histogram and ACF of residuals from the ACD model



We also demonstrate the use of the recursive estimating equations method with AR(m) initial values by fitting a Log $\text{ACD}_1(1,1)$ model with moments from an *exponential*(1) distribution, again reserving 5 observations for use in calculating the MAPE.

The results of fitting a Log $\text{ACD}_1(1,1)$ model with moments from an *exponential*(1) distribution to the data using the recursive EE parameter estimation method with AR(m) initial values are:

```
The initial values from an AR(m) fit are: 0.6214453 0.1172845 0.5521869
The parameter estimates from the EE method are: 0.6130474 0.1175848 0.5638458
The mean absolute percentage error (MAPE) is: 189.602
[1] 189.602
```

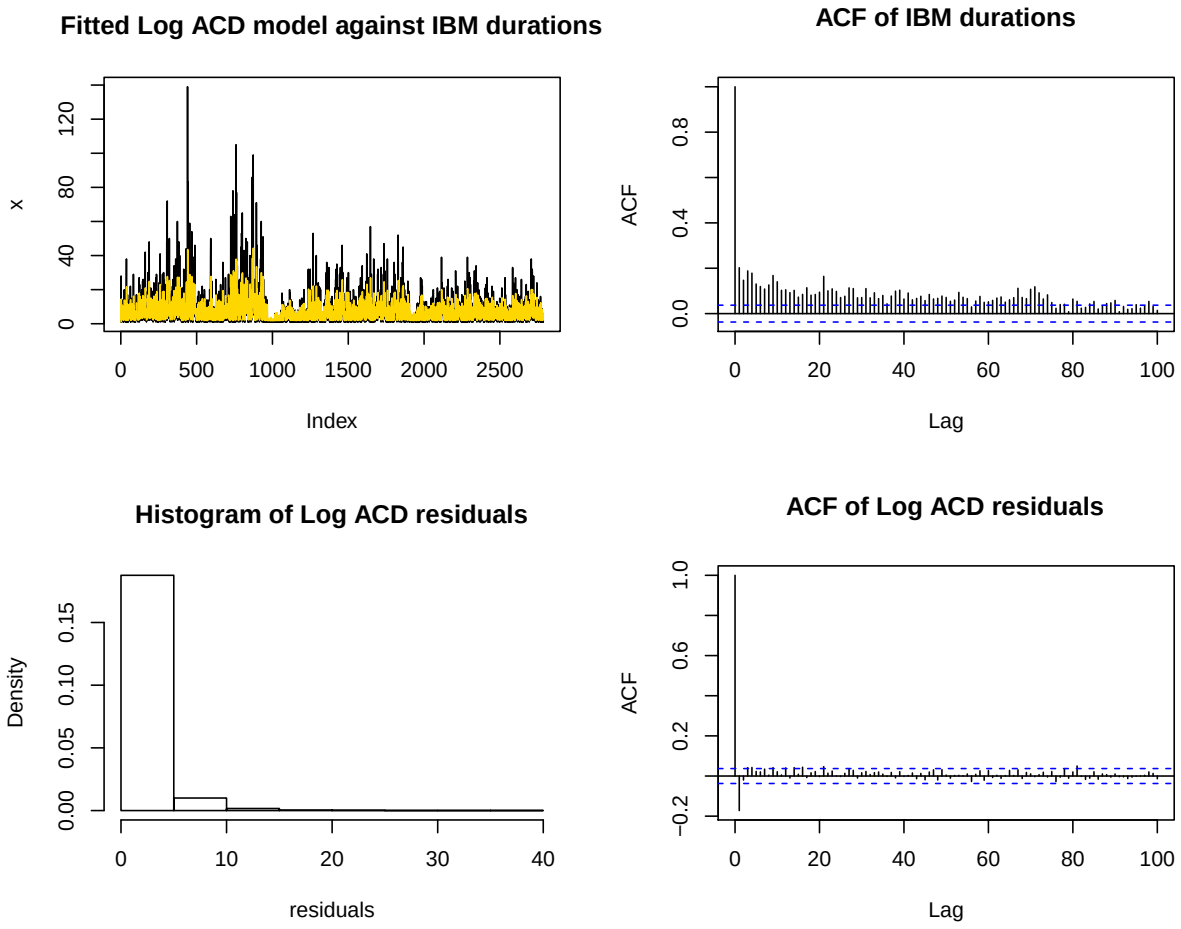
and the fitted model is:

$$\widehat{x_i} \;=\; \exp \widehat{\psi_i}$$

$$\psi_i \quad = \quad 0.6130 + 0.1175 \log x_{i-1} + 0.5638 \psi_{i-1}.$$

The MAPE for the fitted Log $\text{ACD}_1(1,1)$ model is higher than that for the ACD(1,1) model, suggesting that the ACD(1,1) model is a better fit for the IBM duration data. Figure 14 suggests that the Log $\text{ACD}_1(1,1)$ model overfits the data. The residuals in the Log ACD model are given by $\widehat{\varepsilon}_i = x_i / \exp(\widehat{\psi}_i)$ Bauwens and Giot (2000), and the histogram and ACF of the residuals are shown in Figure 14. Model fitting was initialized by setting $\psi_1 = \widehat{\omega}$. The residuals plot in Figure 14 shows a decrease in time dependence, suggesting that the Log $\text{ACD}_1(1,1)$ accounts for a substantial amount of the time dependence in the IBM duration data.

Figure 13: Log ACD model (gold) fitted to IBM duration data (black)



6 Conclusion

The recursive estimating equations approach derived by Thavaneswaran, Ravishanker, and Liang (2014) is applied to the Log ACD_1 model. An overview of duration models and maximum likelihood estimation in the ACD model are provided. In addition, this thesis derives the AR(m) method to find initial parameter estimates for parameter estimation. Initial parameter estimates for the Log ACD_1 model are used in the recursive estimating equations approach. The initial parameter estimation method is proven for the Log ACD_1(1,1) and Log ACD_1(2,1) cases. Future research can focus on generalizing the approach to the Log ACD_1(p,q) case. Monte Carlo simulations show that the AR(m) method successfully recovers parameter estimates close to the true simulation parameters. To demonstrate the process of parameter estimation, an ACD(1,1) model and a Log ACD_1(1,1) model are fitted to a series of IBM duration data. The predictive capability of each model is examined by calculating the MAPE for each model. In this case, the ACD(1,1) model had a lower MAPE, suggesting that the ACD(1,1) model is a better fit for the IBM data. This thesis shows that a viable method to find initial parameter values for the Log ACD_1 model given a set of data begins with writing the Log ACD_1 model as an AR(m) model. The AR(m) model is fitted to the data, and initial parameter estimates can be found from the intercept and coefficients of the fitted AR(m) model.

7 References

Bauwens, L. and Giot, P. (2000). The Logarithmic ACD model: an application to the Bid-ask quote process of three NYSE stocks. *Annales d' Économie et de Statistique* 60, 117-149.

Bauwens, L. and Veredas, D. (2004). The stochastic conditional duration model: a latent factor model for the analysis of financial durations. *Journal of Econometrics* 119, 381-412.

Brillinger, D. R., Guttorp, P., and Schoenberg, F.P. (2002). Point processes, temporal. *Encyclopedia of Environmetrics* 3, 1577-1581.

Easley, D. and O'Hara, M. (1992). Time and the Process of Security Price Adjustment. *Journal of Finance* Volume 47, No. 2. 577-605.

Engle, R. F. and Russell, J. R. (1998). Autoregressive conditional duration: a new model for irregularly spaced transaction data. *Econometrica* 66, 1127-1162.

Godambe, V. P. (1985). The foundations of finite sample estimation in stochastic process. *Biometrika* 72, 319-328.

Jasiak, J. (1998). Persistence in intratrade durations. *Finance* 19, 166-195.

Hejjel, L. and Roth, E. (2004). What is the adequate sampling interval of the ECG signal for heart rate variability analysis in the time domain? *Physiological Measurement* Volume 25, No. 6. 1405.

Liang, Y., Thavaneswaran, A. and Abraham, B. (2011). Joint estimation using quadratic estimating functions. *Journal of Probability and Statistics* Volume 2011 (2011), Article ID 372512, 14 pages.

Pacurar, M. (2008). Autoregressive conditional duration models in finance: a survey of the theoretical and empirical literature. *Journal of Economic Surveys* 22, 711-751.

Paninski, L. (2010) Chapter 2: Introduction to Point Processes. *Lecture Notes*. Available at http://www.stat.columbia.edu/ liam/teaching/neurostat-fall13/uri-eden-point-process-notes.pdf .

Sztajzel, J. (2004). Heart rate variability: a noninvasive electrocardiographic method to measure the autonomic nervous system. *Swiss Medical Weekly* 134, 514-522.

Thavaneswaran, A. and Abraham, B. (1988). Estimation of nonlinear time series models using estimating functions. *Journal of Time Series Analysis* 9, 99-108.

Thavaneswaran, A., Ravishanker, N., and Liang, Y. (2014). Generalized duration models and optimal estimation using estimating functions. *Annals of the Institute of Statistical Mathematics* Forthcoming, 29 pages.

Tsay, R. (2009). Autoregressive Conditional Duration Models. *Palgrave Handbook of Econometrics* 2, 1004-1024.

Tsay, R. (2011). *ACD.CMLE function* [R Code].

Vlahogianni, E., Karlaftis, M. G., and Kepaptsoglou, K. (2010). Nonlinear Autoregressive

Conditional Duration Models for Traffic Congestion Estimation. *Journal of Probability and Statistics* 2011, 13 pages.

Yan, B. and Zivot, E. (2003). Analysis of High-Frequency Financial Data with S-Plus.

8 Appendix of R Code

Custom Functions [6]

```
#############################################################################
##### SIMULATION FUNCTION
#############################################################################

# Log ACD1(m,q) Simulation function: Simulate from the Log ACD1(m,q) model
# n is the final sample size (after burn-in of nb time points)
# parameters are omega, alpha, beta
# alpha and beta can be vectors, depending on the number of lags to be
    included
# alpha = (alpha_1, alpha_2, ..., alpha_m )
# beta = (beta_1, beta_2, ..., beta_q )
# feps is the distribution of the errors epsilon
# par1 and par2 are parameters of feps
# par2 defaults to 1
# function is defined for the exponential, weibull, and gamma distributions
# in the exponential function, par1=lambda
# in the weibull function, par1=alpha and par2=beta
# in the gamma function, par1=k (shape) and par2=theta (scale)

fsim.logacd1 <- function (n, nb, omega, alpha, beta, feps, par1, par2=1) {
# Check for stationarity constraint
if (sum(alpha)+sum(beta)>=1) { cat("Error: The simulated process is not
    weakly stationary")
} else {
# total number of simulated datapoints
nt <- n + nb
# Initialize psis, xs, x, and psi
psis <- rep(1,nt)
xs <-rep(1,nt)
x <-rep(NA,n)   # we'll save the last n entries of xs into x
psi <-rep(NA,n)   # we'll save the last n entries of psis into psi
if (feps=="exponential") {
# randomly generate errors from exponential distribution
eps <- rexp(nt,par1)
# generate xs and psis
```

[6] Custom functions written by Lilian Cheung for use in R. The function used to generate recursive EE parameter estimates was modified from code provided by Dr. Nalini Ravishanker. These functions are used in the simulation studies as well as in the analysis of IBM data.

```
# Initial psis and xs values (depends on maximum number of lags)
for (t in (1:max(length(alpha), length(beta))) ) {
# define temporary lagged.logxs and lagged.psis vectors
# initialization
lagged.logxs <- rep(NA, length(alpha))
lagged.psis <- rep(NA, length(beta))
# calculation
for (i in 1:length(alpha)) {
# if the index t-i is less than 0, then define lagged.logxs[i]=0. Else,
    define lagged.logxs[i] = log(xs[t-i])
index <- t-i
if (index <= 0) {lagged.logxs[i] <- 0} else { lagged.logxs[i] <- log(xs[index
    ]) }
}
for (j in 1:length(beta)) {
# if the index t-j is less than 0, then define lagged.psis[j]=0. Else, define
     lagged.psis[j] = log(xs[t-j])
index <- t-j
if (index <= 0) {lagged.psis[j] <- 0} else { lagged.psis[j] <- psis[index] }
}
psis[t] = omega + alpha%*%lagged.logxs + beta%*%lagged.psis
xs[t] = exp(psis[t])*eps[t]
}
# For t = max(length(alpha), length(beta))+1 : nt
for (t in (max(length(alpha), length(beta))+1):nt) {
# define temporary lagged.logxs and lagged.psis vectors
# initialization
lagged.logxs <- rep(NA, length(alpha))
lagged.psis <- rep(NA, length(beta))
# calculation
for (i in 1:length(alpha)) { lagged.logxs[i] <- log(xs[t-i]) }
for (j in 1:length(beta)) { lagged.psis[j] <- psis[t-j] }
# calculate psis and xs
psis[t] = omega + alpha%*%lagged.logxs + beta%*%lagged.psis
xs[t] = exp(psis[t])*eps[t]
}
# save the simulation data: keep last n data and psi after burn-in save into
    x and psi
x <- ts(xs[(nb+1):nt])
psi <- ts(psis[(nb+1):nt])
# x is the simulated durations series (after the first nb points are deleted)
# output as dataframe
output <- as.data.frame(cbind(x,psi))
} else if (feps=="weibull") {
# randomly generate errors from weibull distribution
```

```
eps <- rweibull(nt, par1, par2)
# generate xs and psis
# Initial psis and xs value (depends on maximum number of lags)
for (t in (1:max(length(alpha), length(beta))) ) {
# define temporary lagged.logxs and lagged.psis vectors
# initialization
lagged.logxs <- rep(NA, length(alpha))
lagged.psis <- rep(NA, length(beta))
# calculation
for (i in 1:length(alpha)) {
# if the index t-i is less than 0, then define lagged.logxs[i]=0. Else,
    define lagged.logxs[i] = log(xs[t-i])
index <- t-i
if (index <= 0) {lagged.logxs[i] <- 0} else { lagged.logxs[i] <- log(xs[index
    ]) }
}
for (j in 1:length(beta)) {
# if the index t-j is less than 0, then define lagged.psis[j]=0. Else, define
     lagged.psis[j] = log(xs[t-j])
index <- t-j
if (index <= 0) {lagged.psis[j] <- 0} else { lagged.psis[j] <- psis[index] }
}
psis[t] = omega + alpha%*%lagged.logxs + beta%*%lagged.psis
xs[t] = exp(psis[t])*eps[t]
}
# For t = max(length(alpha), length(beta))+1 : nt
for (t in (max(length(alpha), length(beta))+1):nt) {
# define temporary lagged.logxs and lagged.psis vectors
# initialization
lagged.logxs <- rep(NA, length(alpha))
lagged.psis <- rep(NA, length(beta))
# calculation
for (i in 1:length(alpha)) { lagged.logxs[i] <- log(xs[t-i]) }
for (j in 1:length(beta)) { lagged.psis[j] <- psis[t-j] }
# calculate psis and xs
psis[t] = omega + alpha%*%lagged.logxs + beta%*%lagged.psis
xs[t] = exp(psis[t])*eps[t]
}
# save the simulation data: keep last n data and psi after burn-in save into
    x and psi
x <- ts(xs[(nb+1):nt])
psi <- ts(psis[(nb+1):nt])
# x is the simulated durations series (after the first nb points are deleted)
# output as dataframe
output <- as.data.frame(cbind(x,psi))
```

```
} else if (feps=="gamma") {
# randomly generate errors from gamma distribution
eps <- rgamma(nt,par1,par2)
# generate xs and psis
# Initial psis and xs value (depends on maximum number of lags)
for (t in (1:max(length(alpha), length(beta))) ) {
# define temporary lagged.logxs and lagged.psis vectors
# initialization
lagged.logxs <- rep(NA, length(alpha))
lagged.psis <- rep(NA, length(beta))
# calculation
for (i in 1:length(alpha)) {
# if the index t-i is less than 0, then define lagged.logxs[i]=0. Else,
    define lagged.logxs[i] = log(xs[t-i])
index <- t-i
if (index <= 0) {lagged.logxs[i] <- 0} else { lagged.logxs[i] <- log(xs[index
    ]) }
}
for (j in 1:length(beta)) {
# if the index t-j is less than 0, then define lagged.psis[j]=0. Else, define
     lagged.psis[j] = log(xs[t-j])
index <- t-j
if (index <= 0) {lagged.psis[j] <- 0} else { lagged.psis[j] <- psis[index] }
}
psis[t] = omega + alpha%*%lagged.logxs + beta%*%lagged.psis
xs[t] = exp(psis[t])*eps[t]
}
# For t = max(length(alpha), length(beta))+1 : nt
for (t in (max(length(alpha), length(beta))+1):nt) {
# define temporary lagged.logxs and lagged.psis vectors
# initialization
lagged.logxs <- rep(NA, length(alpha))
lagged.psis <- rep(NA, length(beta))
# calculation
for (i in 1:length(alpha)) { lagged.logxs[i] <- log(xs[t-i]) }
for (j in 1:length(beta)) { lagged.psis[j] <- psis[t-j] }
# calculate psis and xs
psis[t] = omega + alpha%*%lagged.logxs + beta%*%lagged.psis
xs[t] = exp(psis[t])*eps[t]
}
# save the simulation data: keep last n data and psi after burn-in save into
    x and psi
x <- ts(xs[(nb+1):nt])
psi <- ts(psis[(nb+1):nt])
# x is the simulated durations series (after the first nb points are deleted)
```

```
# output as dataframe
output <- as.data.frame(cbind(x,psi))
} else { cat("Error: The error distribution must be specified as exponential, gamma, or weibull") }
}
}

# EXAMPLE:
# test <- fsim.logacd1(100, 1000, 1, .5, .3, feps="exponential", 1); x <- test$x

##############################################################################
##### AR INITIAL VALUES FUNCTION
##############################################################################

# Log ACD1 initial parameters function: Calculate initial values for parameters in the log ACD(1,1) and log ACD(2,1) model by fitting an AR(m) model to the observed (simulated) data
# Given observed/simulated data x
# Order of the Log ACD1 model is given by (p,q)
# This function will only work for orders (1,1) and (2,1)
# Default is order (1,1)
# Order of the AR model used for initial values is given by m
# Default m is 10
# Minimum m required is 2 for the log ACD (1,1) case, but more lags (up to a certain point) will improve estimation
# Minimum m required is 3 for the log ACD (2,1) case, but more lags (up to a certain point) will improve estimation

finitval.logacd1 <- function (x, p=1, q=1, m=10) {
if (p==1 & q==1) {
if (m < 2) { cat("Error: For the Log ACD1 (1,1) case, m must exceed 2 to get valid initial parameter values", "\n")
} else {
# Take the log transform of the observed (simulated) data and define it as y
y <- log(x)
# Fit an AR(m) model to the log transformed data y, using a 'high enough' order m
ar.model <- arima(y, order=c(m,0,0), include.mean=TRUE)
# Untangle the coefficients
# ALPHA:
alpha.hat <- as.numeric( ar.model$coef[1] )
# BETA:
beta.hat <- as.numeric( ar.model$coef[2]/alpha.hat )
# OMEGA:
```

```
MEAN <- ar.model$coef[length(ar.model$coef)]
intercept <- MEAN*(1-sum(ar.model$coef[1:p]))
omega.hat <- as.numeric( intercept*(1-beta.hat) )
# SIGMA^2: variance of ln(eps)
sigma2.hat <- as.numeric( ar.model$sigma2 )

# Results
results <- data.frame(cbind(alpha.hat, beta.hat, omega.hat, sigma2.hat))
}
} else if (p==2 & q==1) {
if (m < 3) { cat("Error: For the Log ACD1 (2,1) case, m must exceed 3 to get
    valid initial parameter values", "\n") }
# Take the log transform of the observed (simulated) data and define it as y
y <- log(x)
# Fit an AR(m) model to the log transformed data y, using a 'high enough'
    order m
ar.model <- arima(y, order=c(m,0,0), include.mean=TRUE)
# Untangle the coefficients
# ALPHA.1:
alpha.1.hat <- as.numeric( ar.model$coef[1] )
# BETA:
beta.hat <- as.numeric( ar.model$coef[3]/ar.model$coef[2])
# ALPHA.2
# print alpha.2.hat and alpha.2
alpha.2.hat <- as.numeric ( (ar.model$coef[3] - ar.model$coef[1]*(ar.model$
    coef[3]/ar.model$coef[2])^2 ) / (ar.model$coef[3]/ar.model$coef[2]) )
# OMEGA:
MEAN <- ar.model$coef[length(ar.model$coef)]
intercept <- MEAN*(1-sum(ar.model$coef[1:p]))
omega.hat <- as.numeric( intercept*(1-beta.hat) )
# SIGMA^2: variance of ln(eps)
sigma2.hat <- as.numeric( ar.model$sigma2 )

# Results
results <- data.frame(cbind(alpha.1.hat, alpha.2.hat, beta.hat, omega.hat,
    sigma2.hat))
} else { cat("Error: This function is defined only for the Log ACD1 (1,1) and
     Log ACD1 (2,1) cases)", "\n") }
}

##############################################################################
##### ESTIMATING EQUATIONS FUNCTION
##############################################################################

# Log ACD1 estimating equations function: Calculate parameter estimates for
```

```
    the log ACD1(1,1) function using the estimating equations approach
# Given observed/simulated data x
# n is defined as the length of the observation vector
# Order of the Log ACD1 model is given by (p,q)
# This function will only work for orders (1,1) and (2,1)
# p is the length of alpha and q is the length of beta
# moments denotes the distribution from which we derive the first four
    central moments of epsilon
# For exponential moments, par1=lambda
# For weibull moments, par1=alpha and par2=beta
# For gamma moments, par1=k (shape) and par2=theta (scale)
# For none (no distribution specified, look at user input moments
# by default, the moments are drawn from an exponential(1) distribution
# par1 and par2 (optional) are parameters of the distribution of epsilon
# the default values are par1=1 and par2=1
# arinitval tells function whether or not to use AR(m) fit to get initial
    values
# by default, arinitval is TRUE and m is 20
# if false, function will look at user-input initial values in initval
# CHANGE_mod: change for different Duration Models

festeq.logacd1 <- function (x, p=1, q=1, momentsdist="exponential",
    usermoments, par1=1, par2=1, arinitval="TRUE", initval, m=20) {
n <- length(x)
if (p==1 & q==1) {
pdim <- 1+p+q        # number of parameters to be estimated
# MOMENTS
if (momentsdist=="exponential") {
lamda=par1
# central moments, exponential errors
mue=1/lamda
vare=1/lamda**2
skewe=2/lamda**3
kurte=9/lamda**4
} else if (momentsdist=="weibull") {
walpha=par1
wbeta=par2
#raw moments, weibull errors
fimom=wbeta*gamma(1+walpha**-1)
smom=(wbeta**2)*gamma(1+2*walpha**-1)
tmom=(wbeta**3)*gamma(1+3*walpha**-1)
fomom=(wbeta**4)*gamma(1+4*walpha**-1)
#central moments, weibull errors
mue=fimom
vare=smom-fimom**2
```

```
skewe=tmom-3*vare*fimom-fimom**3
kurte=fomom-4*skewe*fimom-6*vare*fimom**2-fimom**4
} else if (momentsdist=="gamma") {
k=par1
gtheta=par2
#raw moments, gamma errors
fimom=gtheta*gamma(1+k)/gamma(k)
smom=(gtheta**2)*gamma(2+k)/gamma(k)
tmom=(gtheta**3)*gamma(3+k)/gamma(k)
fomom=(gtheta**4)*gamma(4+k)/gamma(k)
#central moments, gamma errors
mue=fimom
vare=smom-fimom**2
skewe=tmom-3*vare*fimom-fimom**3
kurte=fomom-4*skewe*fimom-6*vare*fimom**2-fimom**4
} else if (momentsdist=="none") {
# check to see if user-input moments (usermoments) are valid
if (is.numeric(usermoments)==TRUE & is.vector(usermoments)==TRUE & length(
    usermoments)==4) {
# user-input central moments
mue=usermoments[1]
vare=usermoments[2]
skewe=usermoments[3]
kurte=usermoments[4]
} else {cat("Error: Unrecognized moments specification", "\n")}
} else { cat("Error: Unrecognized moments specification", "\n") }
# INITIALIZATION
# identity matrix
iden = diag(pdim)
# moments of x
mu <- rep(1,n)      # mu(i)
sigsq <- rep(1,n)  # sigsq(i)
gamma <- rep(1,n)  # third central moment of x (not skewness)
kappa <- rep(1,n)  # fourth central moment of x (not kurtosis)
# psi hat
psih <- rep(1,n)
# k matrix (variance-covariance) and k inverse (observed information)
kmat = array(NA, dim = c(pdim, pdim, n))
kinv = array(NA, dim = c(pdim, pdim, n))
# parameter estimates for each iteration
thehat = array(NA, dim = c(pdim, 1, n))
# derivative of psi and second derivative of psi
derpsi<-matrix(rep(0),pdim,1)
der2psi<-matrix(rep(0),pdim,pdim)
# derivative of mu, sigsq; second derivates of mu, sigsq
```

```
dermu<-matrix(rep(0),pdim,1)
dersigsq<-matrix(rep(0),pdim,1)
der2mu<-matrix(rep(0),pdim,pdim)
der2sigsq<-matrix(rep(0),pdim,pdim)
# derivative of m, M, quadratic variation, quadratic covariation, eta
derm<-matrix(rep(0),pdim,1)
derqm<-matrix(rep(0),pdim,1)
dervm<-matrix(rep(0),pdim,1)
dervqm<-matrix(rep(0),pdim,1)
dereta<-matrix(rep(0),pdim,1)
# optimal a and b
astr<-matrix(rep(0),pdim,1)
bstr<-matrix(rep(0),pdim,1)
# INITIAL VALUES
# CHANGE_mod
if (arinitval=="TRUE") {
initial<-finitval.logacd1(x, 1, 1, m)
initial<- as.numeric(c(initial[3], initial[1], initial[2]))
cat("The initial values from an AR(m) fit are:", initial, "\n")
# omega, alpha, beta initial values
} else if (arinitval=="FALSE") {
# check to see if user-input initial values (initval) are valid
if (is.numeric(initval)==TRUE & length(initval)==pdim) {
initial <- c(initval[1], initval[2], initval[3])
cat("The user-input initial values are:",initial, "\n")
} else { cat("Error: invalid initial values input", "\n")}
}
# ESTIMATING EQUATIONS
# Put initial values into initial positions of arrays
thehat[,,1]=initial
# Initial observed information and var-cov matrices
# CHANGE_mod
kinv[,,1]=diag(c(1/(initial[1]/2)**2,1/(initial[2]/2)**2,1/(initial[3]/2)**2)
    )
kmat[,,1]=solve(kinv[,,1])
# Recursive Estimation
# t=1
psih[1]=thehat[1,1,1]     # omega     #CHANGE_mod
# t=2:n
for (t in 2:n){
#CHANGE_mod (change psi, derivative of psi and second derivative of psi)
# Define psi
psih[t]=thehat[1,1,t-1]+thehat[2,1,t-1]*log(x[t-1])+thehat[3,1,t-1]*psih[t-1]
# Define derivatives of psih(t) wrt theta: pdim*1 vector
# First derivative
```

```
derpsi=matrix(c(1,log(x[t-1]),psih[t-1]),pdim,1)
# Second derivative:
der2psi=matrix(rep(0),pdim,pdim)
# mu_t, sigsq_t,gamma_t, kappa_t
mu=mue*exp(psih[t])
sigsq=vare*exp(2*psih[t])
gamma=skewe*exp(3*psih[t])  # recall skewe is third central moment
kappa=kurte*exp(4*psih[t])  # recall kurte is fourth central moment
# Compute m(t) and M(t)
m=x[t]-mu
qm=m**2-sigsq
# Compute Quadratic variations and covariance
vm=sigsq*exp(2*psih[t])
vqm=(kurte-vare**2)*exp(4*psih[t])
vmqm=skewe*exp(3*psih[t])
# Define rho^2(t) and eta(t)
#rho = vare*(kurte-vare**2)/(vare*(kurte-vare**2)-skewe**2)
termr= 1-(vmqm**2/(vm*vqm))
rho=1/termr
#eta = skewe/(vare*(kurte-vare**2)*exp(3*psih[t]))
eta=vmqm/(vm*vqm)
# First Derivatives of mu(t) and sigsq(t)
dermu=mue*derpsi*exp(psih[t])
dersigsq=2*vare*derpsi*exp(2*psih[t])
# Second Derivatives of mu(t) and sigsq(t)
der2mu=mue*exp(psih[t])*(der2psi+derpsi%*%t(derpsi))
der2sigsq=2*vare*exp(2*psih[t])*(der2psi+2*derpsi%*%t(derpsi))
# Define vectors astr and bstr
astr=rho*(-dermu/vm +dersigsq*eta)
bstr=rho*(dermu*eta - dersigsq/vqm)
# Define Derivatives of m(i) and M(i)
derm=-mue*exp(psih[t])*derpsi
derqm=2*m*derm - dersigsq
# Derivatives of <m>(i) and <M>(i)
dervm=2*vare*exp(2*psih[t])*derpsi
dervqm=4*(kurte-vare**2)*exp(4*psih[t])*derpsi
# Derivative of eta(i)
dereta=-3*skewe*derpsi/(vare*(kurte-vare**2)*exp(3*psih[t]))
# Derivatives of astr and bstr
# astr
terma1=(vm*der2mu -dermu%*%t(derm))/vm**2
terma2=der2sigsq*eta+dersigsq%*%t(dereta)
derastr=-rho*terma1 + rho*terma2
# bstr
termb1=der2mu*eta + dermu%*%t(dermu)
```

```
termb2=(vqm*der2sigsq - dersigsq%*%t(derqm))/vqm**2
derbstr=rho*termb1-rho*termb2
# Compute Kinv(t)
termk1=astr%*%t(derm) + m*derastr
termk2=bstr%*%t(derqm)+ qm*derbstr
kinv[,,t] = kinv[,,t-1] - (termk1+termk2)
# Invert to get K(t)
kmat[,,t]=solve(kinv[,,t])
# compute thehat[t]
termt=astr*m + bstr*qm
thehat[,,t]=thehat[,,t-1]+kmat[,,t]%*%termt
}
# ESTIMATES
# print(thehat[,,1:n])
cat("The parameter estimates from the EE method are:", thehat[,,n], "\n")
finalest <- thehat[,,n]
} else if (p==2 & q==1) {
pdim <- 1+p+q        # number of parameters to be estimated
# MOMENTS
if (momentsdist=="exponential") {
lamda=par1
# central moments, exponential errors
mue=1/lamda
vare=1/lamda**2
skewe=2/lamda**3
kurte=9/lamda**4
} else if (momentsdist=="weibull") {
walpha=par1
wbeta=par2
#raw moments, weibull errors
fimom=wbeta*gamma(1+walpha**-1)
smom=(wbeta**2)*gamma(1+2*walpha**-1)
tmom=(wbeta**3)*gamma(1+3*walpha**-1)
fomom=(wbeta**4)*gamma(1+4*walpha**-1)
#central moments, weibull errors
mue=fimom
vare=smom-fimom**2
skewe=tmom-3*vare*fimom-fimom**3
kurte=fomom-4*skewe*fimom-6*vare*fimom**2-fimom**4
} else if (momentsdist=="gamma") {
k=par1
gtheta=par2
#raw moments, gamma errors
fimom=gtheta*gamma(1+k)/gamma(k)
smom=(gtheta**2)*gamma(2+k)/gamma(k)
```

```
tmom=(gtheta**3)*gamma(3+k)/gamma(k)
fomom=(gtheta**4)*gamma(4+k)/gamma(k)
#central moments, gamma errors
mue=fimom
vare=smom-fimom**2
skewe=tmom-3*vare*fimom-fimom**3
kurte=fomom-4*skewe*fimom-6*vare*fimom**2-fimom**4
} else if (momentsdist=="none") {
# check to see if user-input moments (usermoments) are valid
if (is.numeric(usermoments)==TRUE & is.vector(usermoments)==TRUE & length(
    usermoments)==4) {
# user-input central moments
mue=usermoments[1]
vare=usermoments[2]
skewe=usermoments[3]
kurte=usermoments[4]
} else {cat("Error: Unrecognized moments specification", "\n")}
} else { cat("Error: Unrecognized moments specification", "\n") }
# INITIALIZATION
# identity matrix
iden = diag(pdim)
# moments of x
mu <- rep(1,n)      # mu(i)
sigsq <- rep(1,n)  # sigsq(i)
gamma <- rep(1,n)  # third central moment of x (not skewness)
kappa <- rep(1,n)  # fourth central moment of x (not kurtosis)
# psi hat
psih <- rep(1,n)
# k matrix (variance-covariance) and k inverse (observed information)
kmat = array(NA, dim = c(pdim, pdim, n))
kinv = array(NA, dim = c(pdim, pdim, n))
# parameter estimates for each iteration
thehat = array(NA, dim = c(pdim, 1, n))
# derivative of psi and second derivative of psi
derpsi<-matrix(rep(0),pdim,1)
der2psi<-matrix(rep(0),pdim,pdim)
# derivative of mu, sigsq; second derivates of mu, sigsq
dermu<-matrix(rep(0),pdim,1)
dersigsq<-matrix(rep(0),pdim,1)
der2mu<-matrix(rep(0),pdim,pdim)
der2sigsq<-matrix(rep(0),pdim,pdim)
# derivative of m, M, quadratic variation, quadratic covariation, eta
derm<-matrix(rep(0),pdim,1)
derqm<-matrix(rep(0),pdim,1)
dervm<-matrix(rep(0),pdim,1)
```

```
dervqm<-matrix(rep(0),pdim,1)
dereta<-matrix(rep(0),pdim,1)
# optimal a and b
astr<-matrix(rep(0),pdim,1)
bstr<-matrix(rep(0),pdim,1)
# INITIAL VALUES
# CHANGE_mod
if (arinitval=="TRUE") {
initial<-finitval.logacd1(x, 2, 1, m)
initial<- as.numeric(c(initial[4], initial[1], initial[2], initial[3]))
cat("The initial values from an AR(m) fit are:", initial, "\n")
# omega, alpha1, alpha2, beta initial values
} else if (arinitval=="FALSE") {
# check to see if user-input initial values (initval) are valid
if (is.numeric(initval)==TRUE & length(initval)==pdim) {
initial <- c(initval[1], initval[2], initval[3], initval[4])
cat("The user-input initial values are:",initial, "\n")
} else { cat("Error: invalid initial values input", "\n")}
}
# ESTIMATING EQUATIONS
# Put initial values into initial positions of arrays
thehat[,,1]=initial
thehat[,,2]=thehat[,,1]
# Initial observed information and var-cov matrices
# CHANGE_mod
kinv[,,1]=diag(c(1/(initial[1]/2)**2,1/(initial[2]/2)**2,1/(initial[3]/2)**
    2,1/(initial[4]/2)**2))
kmat[,,1]=solve(kinv[,,1])
kinv[,,2]=kinv[,,1]
kmat[,,2]=kmat[,,1]
# Recursive Estimation
# t=1 and t=2
#CHANGE_mod
psih[1]=thehat[1,1,1]     # omega
psih[2]=thehat[1,1,1]+thehat[2,1,1]*log(x[1])+thehat[4,1,1]*psih[1]    # omega
    + alpha1*logx1 + beta*psih1
# t=3:n
for (t in 3:n){
#CHANGE_mod (change psi, derivative of psi and second derivative of psi)
# Define psi
psih[t]=thehat[1,1,t-1]+thehat[2,1,t-1]*log(x[t-1])+thehat[3,1,t-1]*log(x[t
    -2])+thehat[4,1,t-1]*psih[t-1]
# Define derivatives of psih(t) wrt theta: pdim*1 vector
# First derivative
derpsi=matrix(c(1,log(x[t-1]),log(x[t-2]),psih[t-1]),pdim,1)
```

```
# Second derivative:
der2psi=matrix(rep(0),pdim,pdim)
# mu_t, sigsq_t,gamma_t, kappa_t
mu=mue*exp(psih[t])
sigsq=vare*exp(2*psih[t])
gamma=skewe*exp(3*psih[t])  # recall skewe is third central moment
kappa=kurte*exp(4*psih[t])  # recall kurte is fourth central moment
# Compute m(t) and M(t)
m=x[t]-mu
qm=m**2-sigsq
# Compute Quadratic variations and covariance
vm=sigsq*exp(2*psih[t])
vqm=(kurte-vare**2)*exp(4*psih[t])
vmqm=skewe*exp(3*psih[t])
# Define rho^2(t) and eta(t)
#rho = vare*(kurte-vare**2)/(vare*(kurte-vare**2)-skewe**2)
termr= 1-(vmqm**2/(vm*vqm))
rho=1/termr
#eta = skewe/(vare*(kurte-vare**2)*exp(3*psih[t]))
eta=vmqm/(vm*vqm)
# First Derivatives of mu(t) and sigsq(t)
dermu=mue*derpsi*exp(psih[t])
dersigsq=2*vare*derpsi*exp(2*psih[t])
# Second Derivatives of mu(t) and sigsq(t)
der2mu=mue*exp(psih[t])*(der2psi+derpsi%*%t(derpsi))
der2sigsq=2*vare*exp(2*psih[t])*(der2psi+2*derpsi%*%t(derpsi))
# Define vectors astr and bstr
astr=rho*(-dermu/vm +dersigsq*eta)
bstr=rho*(dermu*eta - dersigsq/vqm)
# Define Derivatives of m(i) and M(i)
derm=-mue*exp(psih[t])*derpsi
derqm=2*m*derm - dersigsq
# Derivatives of <m>(i) and <M>(i)
dervm=2*vare*exp(2*psih[t])*derpsi
dervqm=4*(kurte-vare**2)*exp(4*psih[t])*derpsi
# Derivative of eta(i)
dereta=-3*skewe*derpsi/(vare*(kurte-vare**2)*exp(3*psih[t]))
# Derivatives of astr and bstr
# astr
terma1=(vm*der2mu -dermu%*%t(derm))/vm**2
terma2=der2sigsq*eta+dersigsq%*%t(dereta)
derastr=-rho*terma1 + rho*terma2
# bstr
termb1=der2mu*eta + dermu%*%t(dermu)
termb2=(vqm*der2sigsq - dersigsq%*%t(derqm))/vqm**2
```

```
derbstr=rho*termb1-rho*termb2
# Compute Kinv(t)
termk1=astr%*%t(derm) + m*derastr
termk2=bstr%*%t(derqm)+ qm*derbstr
kinv[,,t] = kinv[,,t-1] - (termk1+termk2)
# Invert to get K(t)
kmat[,,t]=solve(kinv[,,t])
# compute thehat[t]
termt=astr*m + bstr*qm
thehat[,,t]=thehat[,,t-1]+kmat[,,t]%*%termt
}
# ESTIMATES
# print(thehat[,,1:n])
cat("The parameter estimates from the EE method are:", thehat[,,n], "\n")
finalest <- thehat[,,n]
} else { cat("Error: This function is only defined for the log ACD1 (1,1) and
    log ACD1 (2,1) cases") }
}

# EXAMPLES:
# test <- fsim.logacd1(100, 1000, 1, .5, .3, feps="exponential", 1); x <-
    test$x
# ee <- festeq.logacd1(x, 1,1, momentsdist="exponential", arinitval="TRUE");
    ee
# ee <- festeq.logacd1(x, 2,1, momentsdist="exponential", arinitval="TRUE");
    ee
# festeq.logacd1(durations, 1,1, momentsdist="exponential", arinitval="TRUE")
# The initial values from an AR(m) fit are: 0.6171753 0.1169692 0.5546814
# The parameter estimates from the EE method are: 0.607835 0.1172964
    0.5670814
# festeq.logacd1(durations[1:(length(durations)-4)], 1,1, momentsdist="
    exponential", arinitval="TRUE")
# The initial values from an AR(m) fit are: 0.6217596 0.1172356 0.5520966
# The parameter estimates from the EE method are: 0.6133423 0.1175344
    0.5637614

################################################################################
##### MEAN ABSOLUTE PERCENTAGE ERROR (MAPE) FUNCTION
################################################################################

# Calculates the MAPE of a Log ACD1(p,q) model using EE parameter estimates
    and AR-method initial values
# x is the full-length original observations vector
# n is the number of points used to fit the Log ACD model
# L is the number of steps to predict from forecast origin n
```

```
# This function only works for orders (1,1) and (2,1)
# momentsdist is the distribution used in the EE parameter estimation procedure
# For exponential moments, par1=lambda
# For weibull moments, par1=alpha and par2=beta
# For gamma moments, par1=k (shape) and par2=theta (scale)
# For none (no distribution specified, look at user input moments

fmape.logacd1 <- function(x, n, L, p, q, momentsdist, par1=1, par2=1) {
# make sure that n and L are valid
if (n < 0 | L < 0) { cat("Error: n and L must be positive") } else {
if (n > length(x)) { cat("Error: Please specify an n <= the length of the observed data") } else {
if (n+L > length(x)) { cat("Error: The sum of the number of points used to fit the model and the number of points used for prediction must not exceed the length of the observations vector") } else {
# define fit and prediction parts of x
fit <- x[1:n]
pred.actual <- x[(n+1):(n+L)]
# psi values associated with fit part of x
psi.fit <- log(x[1:n])
# Log ACD1(1,1) Case
if (p==1 & q==1) {
# EE Parameter estimation
if (momentsdist=="exponential") {
param <- festeq.logacd1(fit, 1,1, momentsdist="exponential", par1, par2, arinitval="TRUE")
} else if (momentsdist=="gamma") {
param <- festeq.logacd1(fit, 1,1, momentsdist="gamma", par1, par2, arinitval="TRUE")
} else if (momentsdist=="weibull") {
param <- festeq.logacd1(fit, 1,1, momentsdist="weibull", par1, par2, arinitval="TRUE")
} else { cat("Error: Unrecognized moments distribution") }
# define omega, alpha, beta estimates
omega <- param[3]; alpha <- param[1]; beta <- param[2]
# Prediction (j=1:L steps from forecast origin n) given n values of x
pred.psi <- rep(NA, L)
pred.xhat <- rep(NA, L)
# For j=1
pred.psi[1] <- omega + alpha*log(x[(n+1)-1]) + beta*psi.fit[(n+1)-1]
pred.xhat[1] <- exp(pred.psi[1])
# For j=2:L steps from n
for (j in 2:L) {
pred.psi[j] <- omega + alpha*log(x[(n+j)-1]) + beta*pred.psi[j-1]
```

```
pred.xhat[j] <- exp(pred.psi[j])
}
# MAPE Calculation
mape <- 100*(mean(abs((pred.actual-pred.xhat)/pred.actual)))
cat("The mean absolute percentage error (MAPE) is:", mape, "\n")
mape
} else if (p==2 & q==1) {
# EE Parameter estimation
if (momentsdist=="exponential") {
param <- festeq.logacd1(fit, 2,1, momentsdist="exponential", par1, par2,
    arinitval="TRUE")
} else if (momentsdist=="gamma") {
param <- festeq.logacd1(fit, 2,1, momentsdist="gamma", par1, par2, arinitval=
    "TRUE")
} else if (momentsdist=="weibull") {
param <- festeq.logacd1(fit, 2,1, momentsdist="weibull", par1, par2,
    arinitval="TRUE")
} else { cat("Error: Unrecognized moments distribution") }
# define omega, alpha1, alpha2, beta estimates
omega <- param[4]; alpha1 <- param[1]; alpha2 <- param[2]; beta <- param[3]
# Prediction (j=1:L steps from forecast origin n) given n values of x
pred.psi <- rep(NA, L)
pred.xhat <- rep(NA, L)
# For j=1
pred.psi[1] <- omega + alpha1*log(x[(n+1)-1]) + alpha2*log(x[(n+1)-2]) + beta
    *psi.fit[(n+1)-1]
pred.xhat[1] <- exp(pred.psi[1])
# For j=2:L steps from n
for (j in 2:L) {
pred.psi[j] <- omega + alpha1*log(x[(n+j)-1]) + alpha2*log(x[(n+j)-2]) + beta
    *pred.psi[j-1]
pred.xhat[j] <- exp(pred.psi[j])
}
# MAPE Calculation
mape <- 100*(mean(abs((pred.actual-pred.xhat)/pred.actual)))
cat("The mean absolute percentage error (MAPE) is:", mape, "\n")
mape
} else { cat("This function is only defined for the Log ACD1(1,1) and Log
    ACD1(2,1) cases") }
}
}
}
}
```

Code for Log_1(1,1) Monte Carlo Simulations

```
##############################################################################
##### (1) MONTE CARLO SIMULATION - LOG ACD1(1,1) , nsim=100
##############################################################################
# Prior to using this code: Load custom functions

# number of simulations
nsim = 100
# initialize simulation number
isim = 1
# initialize omgi, alphi, and beti to store initial values across nsim
    simulations:
omgi = rep(NA, nsim)
alpi = rep(NA, nsim)
beti = rep(NA, nsim)
# initialize omgh, alph, and beth to store parameter estimates across nsim
    simulations:
omgh = rep(NA, nsim)
alph = rep(NA, nsim)
beth = rep(NA, nsim)

# simulation setup
n <- 4000      # length of sample after burn-in
nb <- 1000       # burn-in
# parameters
# setup 1
omg <- 1
alp <- 0.5
bet <- 0.3
# setup 2
# omg <- 3
# alp <- 0.2
# bet <- -0.4
# setup 3
# omg <- 2
# alp <- -0.1
# bet <- .23
# setup 4
# omg <- -1.5
# alp <- -0.2
# bet <- 0.65
# distribution
```

```
# setups 1 and 2
distr <- "exponential"
par1 = 1
par2 = 1
# setups 3 and 4
# distr <- "gamma"
# par1 = 2
# par2 = 0.5

#####################################
## AR(m) initial parameter estimates
#####################################

set.seed(123457)   # random seed for generation

for (isim in 1:nsim) {
# simulation
sim <- fsim.logacd1(n, nb, omg, alp, bet, feps=distr, par1, par2)
x <- sim$x
# initial values
init <- finitval.logacd1(x, p=1, q=1, m=10)
# store initial parameter estimates into omgi, alphi, beti
omgi[isim] <- init[3]
alpi[isim] <- init[1]
beti[isim] <- init[2]
omgi <- as.numeric(omgi); alpi <- as.numeric(alpi); beti <- as.numeric(beti)
}
quantile(omgi,probs=c(0.05,.25,.5,.75,.95))
quantile(alpi,probs=c(0.05,.25,.5,.75,.95))
quantile(beti,probs=c(0.05,.25,.5,.75,.95))

# sample graphs
# ts.plot(x, main="Data generated from the Log ACD1(1,1) model with \n theta
    =(1,0.5,0.3) and exponential(1) errors")

#####################################
## EE with AR(m) initial parameter estimates
#####################################

set.seed(123457)   # random seed for generation

for (isim in 1:nsim) {
# simulation
sim <- fsim.logacd1(n, nb, omg, alp, bet, feps=distr, par1, par2)
x <- sim$x
```

```
# estimation with AR initial values
try(EE <- festeq.logacd1(x, 1,1, momentsdist=distr, par1, par2, arinitval="
    TRUE", m=10))
# store estimated parameters into omgh, alph, beth
omgh[isim] <- EE[1]
alph[isim] <- EE[2]
beth[isim] <- EE[3]
}

quantile(omgh,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(alph,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(beth,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)

#####################################
## EE without AR(m) initial parameter estimates
#####################################

set.seed(123457)   # random seed for generation

# re-initialize omgh, alph, and beth to store parameter estimates across nsim
    simulations:
omgh = rep(NA, nsim)
alph = rep(NA, nsim)
beth = rep(NA, nsim)

for (isim in 1:nsim) {
# simulation
sim <- fsim.logacd1(n, nb, omg, alp, bet, feps=distr, par1, par2)
x <- sim$x
# estimation without AR initial values
initval1 <- runif(1, -5, 5)
initval2 <- runif(1, -1, 1)
initval3 <- runif(1, -1+abs(initval2), 1-abs(initval2))
try(EE <- festeq.logacd1(x, 1,1, momentsdist=distr, par1, par2, arinitval="
    FALSE", initval=c(initval1, initval2, initval3)))
# store estimated parameters into omgh, alph, beth
omgh[isim] <- EE[1]
alph[isim] <- EE[2]
beth[isim] <- EE[3]
}

quantile(omgh,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(alph,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(beth,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
```

Code for Log_1(2,1) Monte Carlo Simulations

```
##############################################################################
##### (2) MONTE CARLO SIMULATION - LOG ACD1(2,1) , nsim=100
##############################################################################
# Prior to using this code: Load custom functions

# number of simulations
nsim = 100
# initialize simulation number
isim = 1
# initialize omgi, alphi, and beti to store initial values across nsim
    simulations:
omgi = rep(NA, nsim)
alpi1 = rep(NA, nsim)
alpi2 = rep(NA, nsim)
beti = rep(NA, nsim)
# initialize omgh, alph, and beth to store parameter estimates across nsim
    simulations:
omgh = rep(NA, nsim)
alph1 = rep(NA, nsim)
alph2 = rep(NA, nsim)
beth = rep(NA, nsim)

# simulation setup
n <- 4000      # length of sample after burn-in
nb <- 1000       # burn-in
# parameters
# setup 1
# omg <- 10
# alp1 <- 0.1
# alp2 <- -0.5
# bet <- 0.06
# setup 2
omg <- 5
alp1 <- -0.11
alp2 <- -0.6
bet <- -0.2
```

```
# distribution
# setup 1
# distr <- "exponential"
# par1 = 1
# par2 = 1
# setup 2
distr <- "gamma"
par1 = 2
par2 = 0.5

#####################################
## AR(m) initial parameter estimates and EE using user-input initial
    parameter estimates from the AR(m) estimation
#####################################

set.seed(123457)   # random seed for generation

for (isim in 1:nsim) {
# simulation
sim <- fsim.logacd1(n, nb, omg, c(alp1, alp2), bet, feps=distr, par1, par2)
x <- sim$x
# AR initial values
try(init <- finitval.logacd1(x, p=2, q=1, m=12))
# store initial parameter estimates into omgi, alphi, beti
omgi[isim] <- init[4]
alpi1[isim] <- init[1]
alpi2[isim] <- init[2]
beti[isim] <- init[3]
# estimation with user-input AR initial values
try(EE <- festeq.logacd1(x, p=2,q=1, momentsdist=distr, par1, par2, arinitval
    ="FALSE", initval=c(as.numeric(omgi[isim]), as.numeric(alpi1[isim]), as.
    numeric(alpi2[isim]), as.numeric(beti[isim]))))
# store EE estimated parameters into omgh, alph, beth
omgh[isim] <- EE[1]
alph1[isim] <- EE[2]
alph2[isim] <- EE[3]
beth[isim] <- EE[4]
}

omgi <- as.numeric(omgi)
alpi1 <- as.numeric(alpi1)
alpi2 <- as.numeric(alpi2)
beti <- as.numeric(beti)

quantile(omgi,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
```

```
quantile(alpi1,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(alpi2,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(beti,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)

quantile(omgh,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(alph1,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(alph2,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(beth,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)

# GRAPHS
windows()
par(mfrow=c(4,2))
hist(omgi, 12, main="Initial parameter estimates of omega"); hist(omgh, 12,
    main="EE estimates of omega")
hist(alpi1, 12, main="Initial parameter estimates of alpha1"); hist(alph1,
    12, main="EE estimates of alpha1")
hist(alpi2, 12,main="Initial parameter estimates of alpha2"); hist(alph2, 12,
    main="EE estimates of alpha2")
hist(beti, 12,main="Initial parameter estimates of beta"); hist(beth,12, main
    ="EE estimates of beta")

#####################################
## EE without AR(m) initial parameter estimates
#####################################

set.seed(123457)   # random seed for generation

# re-initialize parameter storage vectors:
omgh = rep(NA, nsim)
alph1 = rep(NA, nsim)
alph2 = rep(NA, nsim)
beth = rep(NA, nsim)

for (isim in 1:nsim) {
# simulation
sim <- fsim.logacd1(n, nb, omg, c(alp1, alp2), bet, feps=distr, par1, par2)
x <- sim$x
# estimation without AR initial values
initval1 <- runif(1, 4, 8)
initval2 <- runif(1, -0.3, 0)
initval3 <- runif(1, -0.9, -0.5)
initval4 <- runif(1, -0.3, 0)
try(EE <- festeq.logacd1(x, p=2,q=1, momentsdist=distr, par1, par2, arinitval
    ="FALSE", initval=c(initval1, initval2, initval3, initval4)))
# store estimated parameters into omgh, alph, beth
```

```
omgh[isim] <- EE[1]
alph1[isim] <- EE[2]
alph2[isim] <- EE[3]
beth[isim] <- EE[4]
}

quantile(omgh,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(alph1,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(alph2,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)
quantile(beth,probs=c(0.05,.25,.5,.75,.95), na.rm=TRUE)

# GRAPHS
windows()
par(mfrow=c(2,2))
hist(omgh, 12, main="EE estimates of omega")
hist(alph1, 12, main="EE estimates of alpha1")
hist(alph2, 12,main="EE estimates of alpha2")
hist(beth,12, main="EE estimates of beta")
```

Code for Processing and Analyzing IBM duration data[7]

```
##############################################################################
##### DESCRIPTION OF DATA
##############################################################################
# Note: in the interests of saving space, the raw data are not provided in
    the appendix.

####################
# READ IN RAW IBM DATA
####################
# Change path to location of the data
rawIBM <- read.table("C://Users//Lilian//Documents//2014 Spring//...",header=
    T)
# Change the class of $time from a factor variable to date/time (note: date
    will be the current date)
rawIBM$time <- strptime(rawIBM$time, format="%H:%M:%S")
# rawIBM is in reverse chronological order by time. So invert it.
```

[7] Raw transaction-by-transaction IBM stock data was collected from NASDAQ over the course of the trading day on 9/13/2012 by Chen Ge

```
rawIBM <- rawIBM[order(nrow(rawIBM):1),]
rownames(rawIBM) <- as.character(seq(1, length(rawIBM$time),1))

####################
# EXTRACT DURATIONS
####################
# An event occurs if the price difference between two consecutive trades exceeds some critical value
crit <- 0.0125
# keep only the time and price columns
IBMdata <- rawIBM[,1:2]
attach(IBMdata)
# INITIALIZATION
# initialize a new variable event={1 if an event occurred, 0 otherwise} that is indexed by i
event <- rep(NA, nrow(IBMdata))
# initialize t, which will store the position at which an event occurs
t <- 1
# initialize i, which will start at 2 since that is the time at which the first event can occur
i <- 2
# FOR LOOP
for (i in 2:nrow(IBMdata)) {
# if an event occurs (i.e., |price at time i - price at previous event time| >= critical value)
if (abs(price[i]-price[t])>=crit) {
# set event[i]=1
event[i] <- 1
# store the position at which the event occurred as t (so it can be compared to the next observed price in the next loop)
t <- i
# if an event does not occur
} else {
# set event[i]=0
event[i] <- 0
}
}
# bind event with IBMdata
IBMdata <- cbind(IBMdata, event)
# define arrival times, which are the times associated with an event occurring (times such that event=1)
arr.times <- time[event==1]
# define the number of cumulative events
cum.events <- rep(NA, nrow(IBMdata))
for (i in 2:nrow(IBMdata)) {
```

```
cum.events[i] <- sum(event[2:i])
}
# define durations, which are the non-zero times between events
durations <- as.numeric(diff(arr.times)[diff(arr.times)!=0] )
durations <- durations[is.na(durations)==FALSE]

####################
# PLOTS
####################
windows()
par(mfrow=c(3,1))
plot(time[1:150], cum.events[1:150], type="l", ylab="number of events", xlab="time", main="Cumulative number of events")
plot(time[1:150], event[1:150], type="h", ylab="event (1 if event occurred, 0 otherwise)", xlab="time", main="Occurrence of events") ; abline(h=0)
plot(arr.times[diff(arr.times)!=0][2:45], durations[1:44], type="h", ylab="durations (sec)", xlab="time", main="Durations")
# the ith duration is associated with the time from event_i-1 to event_i
# short durations are associated with quickly occurring events; long durations are associated with slowly occurring events

# DURATIONS SERIES
windows()
plot(arr.times[diff(arr.times)!=0][2:2787],durations, type="l", xlab="time", main="Durations in IBM stock over the course of one trading day")

# ACF
acf(durations)

detach(IBMdata)

#############################################################################
##### ACD MODEL FITTING
#############################################################################
# Note: This portion of the code uses the function ACD.CMLE, which was written by Ruey S. Tsay, and is not provided in this appendix. The code is cited in the references section.
# Prior to using this code: Load duration data and the ACD.CMLE function

x <- durations
n <- length(durations)-5
L <- 5

# define fit and prediction parts of x
fit <- x[1:n]
```

```
pred.actual <- x[(n+1):(n+L)]
# psi values associated with fit part of x
psi.fit <- x[1:n]

# define omega, alpha, beta estimates
acdparam <- ACD.CMLE(durations, order=c(1,1), cond.dist="exp", ini.est=NULL)
omega <- acdparam$estimates[1]; alpha <- acdparam$estimates[2]; beta <-
    acdparam$estimates[3]

# Prediction (j=1:L steps from forecast origin n) given n values of x
pred.psi <- rep(NA, L)
pred.xhat <- rep(NA, L)
# For j=1, time n+1
pred.psi[1] <- omega + alpha*x[n] + beta*psi.fit[n]
pred.xhat[1] <- pred.psi[1]
# For j=2:L steps from n, time n+j
for (j in 2:L) {
pred.psi[j] <- omega + alpha*(x[(n+j)-1]) + beta*pred.psi[j-1]
pred.xhat[j] <- pred.psi[j]
}
# MAPE Calculation
mape <- 100*(mean(abs((pred.actual-pred.xhat)/pred.actual)))
mape

# Fitting the model
# Redefine and initialize psi.fit and x.fit
psi.fit <- rep(1,length(x))
x.fit <-rep(1,length(x))

# For t=1
psi.fit[1] <- x[1]
x.fit[1] <- psi.fit[1]
# For t=2:n
for (t in 2:length(x)) {
psi.fit[t] = omega + alpha*x[t-1] + beta*psi.fit[t-1]
x.fit[t] = psi.fit[t]
}

# Graphs
windows()
plot(x, type="l", main="Fitted ACD model against IBM durations")
lines(x.fit, type="l", col="gold")

windows()
par(mfrow=c(1,2))
```

```
hist(x[1:n]/psi.fit[1:n], main="Histogram of ACD residuals")
acf(x[1:n]/psi.fit[1:n], main="ACF of ACD residuals")

################################################################################
##### LOG ACD MODEL FITTING
################################################################################
# Prior to using this code: Load duration data and custom functions

x <- durations
L <- 5              # observations withheld
n <- length(durations)-L  # observations used to fit model

# EE parameter estimation
# param <- festeq.logacd1(durations[1:n], 1,1, momentsdist="weibull", 9, 1,
    arinitval="TRUE")
  # The initial values from an AR(m) fit are: 0.6171753 0.1169692 0.5546814
  # The parameter estimates from the EE method are: 0.607835 0.1172964
    0.5670814

# MAPE
#1
fmape.logacd1(durations, length(durations)-L, L, 1, 1, momentsdist="
    exponential", 100, 1)
#2
fmape.logacd1(durations, length(durations)-L, L, 1, 1, momentsdist="gamma",
    0.5, 8)

######################
# Graphs
######################

param <- festeq.logacd1(durations[1:n], 1,1, momentsdist="exponential", 1,
    arinitval="TRUE")
omega <- param[3]
alpha <- param[1]
beta <- param[2]

# define fit and prediction parts of x
    fit <- x[1:n]
    pred.actual <- x[(n+1):(n+L)]
  # psi values associated with fit part of x (known values of x)
    psi.fit <- log(x[1:n])


# Prediction (j=1:L steps from forecast origin n) given n values of x
```

```
    pred.psi <- rep(NA, L)
    pred.xhat <- rep(NA, L)
    # For j=1
      pred.psi[1] <- omega + alpha*log(x[(n+1)-1]) + beta*psi.fit[(n+1)-1]
      pred.xhat[1] <- exp(pred.psi[1])
    # For j=2:L steps from n
      for (j in 2:L) {
        pred.psi[j] <- omega + alpha*log(x[(n+j)-1]) + beta*pred.psi[j-1]
        pred.xhat[j] <- exp(pred.psi[j])
      }
    # MAPE Calculation
    mape <- 100*(mean(abs((pred.actual-pred.xhat)/pred.actual)))
    mape

# Fit
  # Initialize psi.fit and x.fit
  psi.fit <- rep(1,length(x))
  x.fit <-rep(1,length(x))

  # For t=1
    psi.fit[1] <- log(x[1])
    x.fit[1] <- exp(psi.fit[1])
  # For t=2:n
    for (t in 2:length(x)) {
        psi.fit[t] = omega + alpha*log(x[t-1]) + beta*psi.fit[t-1]
        x.fit[t] = exp(psi.fit[t] )
    }

# Graphs
windows()
par(mfrow=c(2,2))
plot(x, type="l", main="Fitted Log ACD model against IBM durations")
lines(x.fit, type="l", col="gold")
acf(x, lag.max=100, main= "ACF of IBM durations")

hist(x[1:n]/exp(psi.fit[1:n]), freq=FALSE, xlab="residuals", main="Histogram
    of Log ACD residuals")
acf(x[1:n]/exp(psi.fit[1:n]), lag.max=100, main="ACF of Log ACD residuals")
```