UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

UNIVERSITY OF CAPE TOWN

MASTER'S DISSERTATION

High-frequency correlation dynamics: Is the Epps effect a bias?

Author:
Patrick CHANG

Supervisor:
Assoc. Prof. Tim GEBBIE
Dr. Etienne PIENAAR

A dissertation presented for the degree of
Master of Science in Mathematical Statistics

from the

Department of Statistical Sciences

February 23, 2021

Acknowledgements

I would like to extend my deepest gratitude to Tim Gebbie for introducing me to this area of research. His guidance throughout the project, intuition into the direction of the research problem, and many detailed technical conversations. I am grateful that Tim pushed me into using Julia for this dissertation. It significantly expedited the completion of this dissertation. I am also extremely grateful to Etienne Pienaar for the many detailed technical conversations and attention to detail in picking up my errors. Their assistance was instrumental in the completion of this dissertation.

I would like to thank the Statistical Finance Research Group in the department of Statistical Sciences for the various discussions relating to the work. In particular, I would like to thank Lionel Yelibi, Una Singo, Michael Gant, and Marcus Gawronsky. I would also like to thank Melusi Mavuso for the various discussions regarding the technical aspects behind this work, and Roger Bukuru for his assistance in my Honours project.

I would like to thank the anonymous reviewers from the SIAM Journal on Scientific Computing for helpful critique and suggestions with regards to ensuring the estimator indeed recovers the correct target estimate. I would also like to thank the anonymous reviewers from PLOS ONE for their feedback. I would also like to thank the reviewers who graded the dissertation for their comments and feedback.

I would like to acknowledge the financial assistance of the South African Statistical Association (SASA) towards this research. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to SASA.

Related Publications and Pre-prints

Much of the content in this dissertation has been submitted as a suite of pre-prints to the arXiv.org e-Print archive. The list of pre-prints include:

- Chang et al. (2020e): "Using the Epps effect to detect discrete data generating processes". This work has been submitted to Quantitative Finance (QF) for review.

Some of the content in this dissertation has been published in peer reviewed journals. The list of publications include:

- Chang et al. (2020d): "Malliavin-Mancino estimators implemented with non-uniform fast Fourier transforms". This work has been accepted for publication by SIAM Journal on Scientific Computing (SISC).
- Chang (2020): "Fourier instantaneous estimators and the Epps effect". This work has been accepted for publication by PLOS ONE.

Declaration of Authorship

I, Patrick CHANG, declare that this dissertation titled, "High-frequency correlation dynamics: Is the Epps effect a bias?" and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at this University.
- The contents of this dissertation has not been previously submitted for a degree or any other qualification at this University or any other institution.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

UNIVERSITY OF CAPE TOWN

Abstract

Department of Statistical Sciences

Master of Science

High-frequency correlation dynamics: Is the Epps effect a bias?

by Patrick CHANG

We tackle the question of whether Trade and Quote data from high-frequency finance are representative of discrete connected events, or whether these measurements can still be faithfully represented as random samples of some underlying Brownian diffusion in the context of modelling correlation dynamics. In particular, if the implicit notion of instantaneous correlation dynamics that are independent of the time-scale a reasonable assumption. To this end, we apply kernel averaging non-uniform fast Fourier transforms in the context of the Malliavin-Mancino integrated and instantaneous volatility estimators to speed up the estimators. We demonstrate the implicit time-scale investigated by the estimator by comparing it to the theoretical Epps effect arising from asynchrony. We compare the Malliavin-Mancino and Cuchiero-Teichmann Fourier instantaneous estimators and demonstrate the relationship between the instantaneous Epps effect and the cutting frequencies in the Fourier estimators. We find that using the previous tick interpolation in the Cuchiero-Teichmann estimator results in unstable estimates when dealing with asynchrony, while the ability to bypass the time domain with the Malliavin-Mancino estimator allows it to produce stable estimates and is therefore better suited for ultra high-frequency finance. We derive the Epps effect arising from asynchrony and provide a refined approach to correct the effect. We compare methods to correct for the Epps effect arising from asynchrony when the underlying process is a Brownian diffusion, and when the underlying process is from discrete connected events (proxied using a D-type Hawkes process). We design three experiments using the Epps effect to discriminate the underlying processes. These experiments demonstrate that using a Hawkes representation recovers the empiricism reported in the literature under simulation conditions that cannot be achieved when using a Brownian representation. The experiments are applied to Trade and Quote data from the Johannesburg Stock Exchange and the evidence suggests that the empirical measurements are from a system of discrete connected events where correlations are an emergent property of the time-scale rather than an instantaneous quantity that exists at all time-scales.

Contents

Chapter 1

Introduction

1.1 Overview

Co-variation is a key parameter in quantitative finance with a long-standing application in portfolio selection (Markowitz, 1952) and risk management (McNeil et al., 2005). More recent applications include unsupervised state discovery to discern changes in the financial markets in the high-frequency domain (Hendricks, 2017a; Hendricks et al., 2016a; Hendricks, 2016). The aforementioned applications often require the estimation of a large covariance matrices with say dimension D. In general, a covariance matrix of size DxD requires at least $\frac{1}{2}D(D+1)$ independent and identically distributed (IID) observations in order to estimate a non-singular invertible covariance matrix. The example given in López de Prado (2016) with a covariance matrix of size 50 means that at least 5 years of daily IID data is required. This presents a problem as it is known that inference based on long-periods of financial market data can be problematic because the process is not stationary (Dacorogna et al., 2001). In recent decades, high-frequency data availability has become less of a problem with data vendors offering improved data structures allowing people to collect market data with more ease. This at first glance may seem to ameliorate the issue of data scarcity when estimating covariance matrices. However, high-frequency tick-by-tick data comes with its own set of non-trivial issues.

The first issue with high-frequency tick-by-tick trade data is that the price discovery follows a different generating process compared to the low-frequency daily scale. At a day-to-day scale, the closing auctions follow a Walrasian auction where equilibrium is reached through a process of *tâtonnement*. This is achieved through a volume maximising auction algorithm used to determine the auction uncrossing price (JSE). On the other hand, intraday tick-by-tick price formation is the result of the flow of orders arriving into the continuous double auction market and the response to prices of individual orders (Bouchaud et al., 2008). The question of whether intraday correlation estimates are representative or can be reconciled with inter-day correlation estimates is beyond the scope of the dissertation.

Here the focus is on the *Epps effect* which is a bigger issue with high-frequency data in the context of covariance estimation. This effect is a key phenomenology relating to correlation dynamics in high-frequency finance where correlations decay as the time-scale of investigation decreases. This effect was first noticed by Epps (1979) when investigating the correlation between automakers. Understanding the effect is key to answering the question of whether or not we can use use high-frequency data to achieve better correlation estimates. Specifically, is the Epps effect a bias? The question, although subtle, questions the viability of using Brownian motions in the modelling of high-frequency finance when trying to recover the phenomenology of the Epps effect. In other words, are tick-by-tick trade data merely samples from an underlying continuous time stochastic process, or are they from an underlying web of inter-related discrete events?

In this dissertation, I will be arguing for the case that in fact Brownian motions are an insufficient modelling technique when it comes to recovering the entire phenomenology of the Epps effect.

1.2 Are Brownian motions good enough for high-frequency finance?

Following the seminal work of Bachelier (1900), Brownian motions have become a ubiquitous apparatus when it comes to modelling financial time series (Karatzas and Shreve, 1998). Much of the work has

been developed and thought for homogeneous (equally spaced in time) time series (Dacorogna et al., 2001). This was before the availability of high-frequency data when the methods did not need to deal with issues such as asynchrony. However, decades later and we are still using Brownian motions when modelling intraday financial time series with the assumption that the observed tick-by-tick trades are samples from an underlying continuous time stochastic process. Although Brownian motions have been successful when modelling intraday price dynamics (Bouchaud et al., 2008), the theory has been met with several issues when recovering the stylized facts around volatility and co-volatility.

1.2.1 Market microstructure effects

One of the first stylized facts that Brownian motions could not recover is the so called *signature plot*. This was one of the first indications that Brownian motions may not be suitable when it comes to modelling high-frequency finance, this however has since been reconciled with the argument of market microstructure noise. The signature plot is a plot of the Realised Volatility (RV) as a function of the sampling interval Δt. If P_t is the generic asset price at time t over the time period $[0,T]$, the common approach is to model $X_t = \log P_t$ as an Itô process:

$$dX_t = \mu_t dt + \sigma_t dB_t, \tag{1.1}$$

where B_t is a standard Brownian motion. Typically, the drift μ_t and the instantaneous variance σ_t^2 are continuous stochastic processes (Zhang et al., 2005). Now it is well understood in theory that the Realised Volatility estimator defined as:

$$\text{RV} = \sum_{h=0}^{\lfloor T/\Delta t \rfloor} \left(X_{(h+1)\Delta t} - X_{(h)\Delta t} \right)^2, \tag{1.2}$$

converges to the integrated volatility $\int_0^T \sigma_t^2 dt$ as the sampling interval decreases. This in theory means that the signature plot should be flat, with smaller Δt providing the best possible estimate. However, when plotting signature plots with empirical data, the RV estimates shoot up when Δt is small and flattens out as Δt increases. The literature has reconciled this stylized fact and Brownian motions using the market microstructure noise model (Aït-Sahalia et al., 2005, 2011; Zhang et al., 2005) also known as *latent models* to capture a vast array of issues known as market microstructure effects. This model operates under the assumption that the observed log-price Y_t is composed of a latent Brownian diffusion X_t plus some noise component ε_t, *i.e.*

$$Y_t = X_t + \varepsilon_t, \tag{1.3}$$

where the latent process X_t is the process of interest but is unobservable and ε_t captures the market microstructure effects. The market microstructure effects include but not limited to (Aït-Sahalia et al., 2011): bid-ask bounces, discreteness of price changes, rounding, measurement errors, *etc*. This model has proven to be successful and is the go to model when it comes to estimating the variance/co-variance in high-frequency finance (Ait-Sahalia et al., 2010; Zhang, 2010; Aït-Sahalia et al., 2005, 2011; Zhang et al., 2005; Griffin and Oomen, 2011).

1.2.2 The Epps effect

The second stylized fact that Brownian motions can not recover is the Epps effect. Under the latent model, there is an underlying diffusion component. Meaning that regardless of the size of the sampling interval, correlations exist between two assets. In other words, correlations do not depend on the sampling intervals Δt. Therefore, under these models the Epps effect is a statistical anomaly (apart from the effects of genuine lead-lags) and hence a bias.

Since Thomas Epps first noticed this phenomena in 1979, the Epps effect has been observed in stock markets and foreign exchange markets (See Mastromatteo et al. (2011) and references therein). In my previous work as part of my Honours project, we were also able to show the effect under volume time averaging (Chang et al., 2019a). In the early 2000s up to the mid 2010s, there was a surge in literature investigating the effect and its various sources. The main sources identified in contributing towards this effect are: *(i)* asynchrony, *(ii)* lead-lag, and *(iii)* tick-size. Concretely, modelling asynchrony using Poisson sampling, Renò (2003) explored the effect of asynchrony under the presence of lead-lag. Precup and Iori (2007) were able to demonstrate that different levels of asynchrony resulted in different behaviours of the Epps effect. Around the same time Tóth and Kertész (2007) derived an analytical expression characterising the Epps effect as a function of the rate from the Poisson sampling. Their work lead to the realisation that one can decompose the correlation at a certain scale Δt as a function of the correlation at smaller time scales Δt_0 (Tóth and Kertész, 2009). Subsequently, Münnix et al. (2010) investigated the direct impact of tick-size on the Epps effect. They were able to find a representation that combined the compensation of tick-size with asynchrony (Münnix et al., 2011). The analytical expression characterising the Epps effect in Tóth and Kertész (2007) was then further extended by Mastromatteo et al. (2011) to separate the effects from asynchrony to that of lead-lag.

Despite the collective effort of all the various researchers, these sources can only explain a fraction of the empirically observed Epps effect and compensating for these statistical causes cannot explain the entirety of the Epps effect (Mastromatteo et al., 2011; Münnix et al., 2011). Tóth and Kertész (2009) conjectured that this residual effect may be related to the time-scales of our human reaction and the investigation into the Epps effect has since dwindled off.

It was the work of Bacry et al. (2013a) that made me question the idea around whether or not Brownian motions are good enough for high-frequency finance. In their paper, they proposed a fine-to-coarse model to model intraday price dynamics using a mutually exciting Hawkes process. The model they proposed is able to recover the high-frequency stylized facts naturally, such as the signature plot and the Epps effect. Their model is built on an underlying web of inter-related discrete events. More importantly, in their model, correlation emerges for larger time-scales as a result of interactions between events which cannot be picked up at the finest of scales. This lead to our realisation that we might be looking at the Epps effect the wrong way, what if the Epps effect is not the decay of correlations from Brownian diffusions but rather the emergence of correlation from a complex and collective interaction of events. Under this emergence argument, the residual Epps effect is due to the fact that at the finest of scales the interaction between events cannot be detected. If this is the case, then this complex system of interacting events cannot be faithfully represented using Brownian diffusions even with clever sampling and additional sources of noise, which is why the various researchers could not account for the full effect. This means that we may need to re-think our modelling techniques when it comes to recovering the correlation dynamics in high-frequency finance.

1.3 Objective and structure of dissertation

This dissertation is an extension to my Honours project where we performed some preliminary Exploratory Data Analysis around the Epps effect. The goal behind this dissertation is to build high-speed machinery to further understand the volatility, co-volatility and correlation dynamics in order to gain further insight behind the Epps effect and why correlations break-down at the finest scales. After thoroughly understanding the impact of asynchrony in the Epps effect, I design some experiments in order to detect when the data is discrete so that we can test whether Empirical data can be faithfully represented using Brownian diffusions with clever sampling.

To this end, the dissertation is broken into three main chapters which are interconnected and each with its own novel aspect. Each chapter focuses on a specific idea with a motivation behind the work, introducing

the required tools, simulation experiments, empirical analysis and a small summary of what was achieved in each chapter.

The dissertation is organised as follows: Chapter 2 focuses on building high-speed machinery which will be used throughout the dissertation. Here I use non-uniform fast Fourier transforms to significantly improve the speed of estimation when using the Malliavin-Mancino Fourier estimator (Malliavin and Mancino, 2002, 2009). In this chapter, I further solidify the idea of how the number of Fourier coefficients of the price process N in the Malliavin-Mancino allows us to implicitly investigate different time-scales. This idea has been used in the literature by Renò (2003) and Precup and Iori (2007) but was never documented in detail. Here I clearly demonstrate this by comparing the implicit time-scales to the theoretical Epps effect arising from asynchrony derived by Tóth and Kertész (2007) and Mastromatteo et al. (2011). Chapter 3 I compare two Fourier instantaneous estimators to investigate the impact of the Epps effect arising from asynchrony on the spot estimates. I perform a range of simulation experiments to understand the impact of the various cutting frequencies in the Fourier estimators and the impact of the various time-scales under asynchrony. Moreover, I provide an *ad hoc* method to pick appropriate time-scales and cutting frequencies when dealing with the Epps effect. Chapter 4 I derive the Epps effect arising from asynchrony. I provide a refined method to correct for asynchrony and compare the method to alternative methods which correct for the Epps effect arising from asynchrony. I compare the correction methods on Brownian diffusions, Brownian diffusions with jumps, and the model proposed by Bacry et al. (2013a). From there I design experiments to detect using the Epps effect whether the underlying data is discrete or diffusion based. Chapter 5 summarises what was achieved in each chapter of this dissertation, provides a discussion for possible methods forward, and what I have learnt from this dissertation.

Chapter 2

Malliavin-Mancino estimators implemented with NUFFTs

This chapter is the extended version of Chang et al. (2020d) which has been accepted by SIAM Journal on Scientific Computing for publication. This chapter implements, benchmarks and tests kernel averaging Non-Uniform Fast-Fourier Transform (NUFFT) methods to improve the computational performance of the Malliavin-Mancino Fourier estimator for asynchronously sampled event-data. The implementation is performed for two versions of the Fourier estimator using the Dirichlet and Fejér Fourier basis kernels. Moreover, the NUFFT implementation is done with three common averaging kernels to convolve asynchronous data into a synchronous up-sampled grid for the Fast Fourier Transform (FFT): the Gaussian kernel, the Kaiser-Bessel kernel, and the exponential of semi-circle kernel. This chapter begins with Section 2.1 where I motivate the importance of this work. Then, in Section 2.2 I introduce the Malliavin-Mancino Fourier estimator, I proceed to discuss the various implementation methods and discuss the insights that NUFFTs provide in the estimator. In Section 2.3 I perform various benchmarks and test cases to demonstrate the effectiveness of NUFFTs in terms of speed and its ability to accurately recover the appropriate the integrated correlation and volatilities. Section 2.4 I demonstrate how to use the appropriate cutting frequency N in the Malliavin-Mancino estimator to investigate various time-scales. Finally, Section 2.5 ends this chapter with some closing remarks.

2.1 Motivation

One of the main features in ultra-high frequency finance is that trades arrive in an asynchronous fashion, meaning that observations arrive at random times with unequal spacing. This presents an issue for the majority of estimators which are developed for homogeneous observations. The Malliavin-Mancino estimator address the issue of asynchronous event data using an elegant Fourier approach. This has advantages over *ad hoc* averaging and interpolation methods built on the underlying assumptions of continuity, such as the approach taken in the well understood Hayashi-Yoshida estimator (Chang et al., 2019a). However, the Malliavin-Mancino estimator relies on the evaluation of Fourier coefficients which can be computationally expensive. Therefore, enhancing the speed of the estimator allows us to quickly extract realised covariance and correlation estimates. The novel contribution of this chapter is realising that we can use non-uniform fast Fourier transforms to evaluate the Fourier coefficients and therefore significantly improve the computation performance.

Performance is a key requirement in two related use cases. First, being able to carry large scale Monte-Carlo simulations over many features and for many different time-scales. This is particularly useful when it comes to investigating the Epps effect. Second, in a real-time environment where decisions are being made from streaming event-data.[1] The use of fast methods can reduce the time-scales of effective data-sampling. This can be important for learning algorithms that require many updates to identify an optimal relationship between actions and system states given an objective, such as Q-learning based implementations of reinforcement learning for trading (Hendricks et al., 2016b; Hendricks, 2017b; Hendricks and Wilcox, 2014).

[1] Although our implementation is computationally efficient, it is not fast enough to update each time a new trading event streams through. An online algorithm is better suited for that case. Where this work becomes useful is if one needs to recompute a correlation matrix say every minute.

2.2 Malliavin-Mancino estimators

Malliavin and Mancino (2002, 2009) proposed a nonparametric volatility/co-volatility estimator to deal with *asynchrony* (the arrival of unevenly spaced intraday trades) without the need to impute the data beforehand by means of synchronisation. Synchronisation is often used as a precursory step for classical estimators such as the Realised Volatility estimator to line the observations up on a homogeneous grid in the time domain. The Malliavin-Mancino (MM) estimator overcomes this by constructing the estimator in the frequency domain and representing the Fourier coefficients of the volatility process $\Sigma^{ij}(t)$ using the Fourier coefficients of the price process $X_t^i = \ln P_t^i$, where P_t^i is the generic asset price at time t.

The assumptions for the estimator are minimal, namely:

Assumption 2.2.1 *Let X_t^i be a continuous semi-martingale satisfying the stochastic differential equation:*

$$dX_t^i = \sum_{k=1}^{K} \sigma_k^i(t)\, dW_t^k + b^i(t)\, dt, \quad i = 1,...,n, \tag{A-I}$$

where $W = (W^1,...,W^K)$ are independent Brownian motions on a filtered probability space, and σ_*^* and b^* are adapted stochastic processes satisfying:

$$E\left[\int_0^T \left(b^i(t)\right)^2 dt\right] < \infty, \quad E\left[\int_0^T \left(\sigma_k^i(t)\right)^4 dt\right] < \infty, \quad k = 1,...,K;\ i = 1,...,D. \tag{A-II}$$

The estimator relies on a convolution in the frequency domain. To this end, Malliavin and Mancino (2009) use the *Bohr convolution product*. Given two functions Φ and Ψ on the integers $\mathbb{Z}$, the *Bohr convolution product* exists if the limit exists $\forall k$:

$$(\Phi *_B \Psi)(k) := \lim_{N\to\infty} \frac{1}{2N+1} \sum_{s=-N}^{N} \Phi(s)\Psi(k-s). \tag{2.1}$$

This leads to the main Theorem of Malliavin and Mancino (2009):

Theorem 2.2.1 *Theorem 2.1 of Malliavin and Mancino (2009):*
Consider a process X satisfying Assumption 2.2.1. We then obtain for $i, j = 1, 2$:

$$\frac{1}{2\pi}\mathscr{F}(\Sigma^{ij}) = \mathscr{F}(dX^i) *_B \mathscr{F}(dX^j). \tag{2.2}$$

The convergence of the convolution product in eq. (2.2) is attained in probability.

By re-scaling the trading times from $[0,T]$ to $[0,2\pi]$ using Algorithm 14 and Theorem 2.2.1, we have that for all $k \in \mathbb{Z}$:

$$\mathscr{F}(\Sigma^{ij})(k) = \lim_{N\to\infty} \frac{2\pi}{2N+1} \sum_{|s|\le N} \mathscr{F}(dX^i)(s)\mathscr{F}(dX^j)(k-s). \tag{2.3}$$

Here $\mathscr{F}(*)(\star)$ is the $\star^{\text{th}}$ Fourier coefficient of the $*$ process. Using the previous tick interpolation to avoid a downward bias in the estimator (Barucci and Renò, 2002) and a simple function approximation for the Fourier coefficients yields:

$$\begin{aligned}\mathscr{F}(dX^i)(k) &\approx \frac{1}{2\pi}\sum_{h=0}^{n_i-1}\exp\left(-\mathrm{i}kt_h^i\right)\delta_i(I_h),\\ \mathscr{F}(dX^j)(k) &\approx \frac{1}{2\pi}\sum_{\ell=0}^{n_j-1}\exp\left(-\mathrm{i}kt_\ell^j\right)\delta_j(I_\ell),\end{aligned} \tag{2.4}$$

where $(t_h^i)_{h=0,\dots,n_i}$ and $(t_\ell^j)_{\ell=0,\dots,n_j}$ are the observation times, and the price fluctuations are $\delta_i(I_h) = X^i_{t^i_{h+1}} - X^i_{t^i_h}$ and $\delta_j(I_\ell) = X^j_{t^j_{\ell+1}} - X^j_{t^j_\ell}$ for asset i and j respectively.

Remark 2.2.1 *Notice here that* $\mathrm{i} = \sqrt{-1} \in \mathbb{C}$ *is the imaginary unit in the exponential defining the Fourier transform. It should not be confused with integer indices i for the asset.*

Remark 2.2.2 *Notice that* n_i *and* n_j *a priori need not be the same. The Malliavin-Mancino estimator has an elegant method of dealing with asynchrony. Rather than lining up the observations in the time domain (what most estimators require by means of imputation), the Malliavin-Mancino estimator lines up the Fourier coefficients of the two processes in the Fourier domain (See eq. (2.3)) and hence bypassing the issue of asynchrony in the time domain.*

The integrated volatility/co-volatility can be obtained by setting $k = 0$ in eq. (2.3). This yields the Dirichlet representation as:

$$\hat{\Sigma}^{ij}_{n_i,n_j,N} = \frac{1}{2N+1}\sum_{\substack{|s|\le N\\ h=0,\ell=0}}^{n_i-1,n_j-1} e^{\mathrm{i}s(t_\ell^j - t_h^i)}\delta_i(I_h)\delta_j(I_\ell), \tag{2.5}$$

where $\hat{\Sigma}^{ij}_{n_i,n_j,N} = \int_0^T \Sigma^{ij}(t)dt$.

An alternate version of the Fourier estimator is the Fejér representation:

$$\hat{\Sigma}^{ij}_{n_i,n_j,N} = \frac{1}{N+1}\sum_{\substack{|s|\le N\\ h=0,\ell=0}}^{n_i-1,n_j-1}\left(1-\frac{|s|}{N}\right)e^{\mathrm{i}s(t_\ell^j - t_h^i)}\delta_i(I_h)\delta_j(I_\ell), \tag{2.6}$$

which is more stable under the presence of market microstructure noise (Malliavin and Mancino, 2009).

The various implementation methods follow the same general structure. First, re-scale the trading times from $[0,T]$ to $[0,2\pi]$ (See Algorithm 14) and compute the Nyquist frequency[2] (See Algorithm 15). Second, compute the non-normalised Fourier coefficients of $\mathscr{F}(dX^i)(k)$ $k \in \{-N,\dots,N\}$ for all assets. Finally, compute either the Dirichlet or Fejér representation of the estimator. The implementation methods only differ in the computation for the Fourier coefficients (step F.4). The general outline is given in Algorithm 1.

[2] Mancino et al. (2017) picks N such that MSE is minimised.

Require:

1. **P**: (n x D) matrix of sampled prices. Non-trade times are represented using *NaNs* or *NAs*.
2. **T**: (n x D) matrix of sampled times. Non-trade times are represented using *NaNs* or *NAs*.
3. N (Optional): cutoff frequency (Integer) used in the convolution. Default is set to be the Nyquist cutoff.
4. tol (Optional): error tolerance for NUFFTs. Determines the number of grid points to spread. Default is set to 10^{-12}.

Step I. Initialisation.

I.1. Re-scale the sampled times (**T**) (See Algorithm 14).

I.2. Compute the Nyquist cutoff (N)—unless specified otherwise through input parameter (See Algorithm 15).

Step F: Compute the Fourier coefficients, $k \in \{-N, ..., N\}$.

for i = 1 to D **do**

F.1. Extract the re-scaled sampled times for the i^{th} object: $\tilde{\mathbf{t}}^i = \mathbf{T}(i)$, excluding any *NaNs* or *NAs*.

F.2. Extract and compute the logarithm of the sampled prices for the i^{th} object: $\tilde{\mathbf{p}}_i = \ln\left(\mathbf{p}_i\left(\tilde{\mathbf{t}}^i\right)\right)$, excluding any *NaNs* or *NAs*.

F.3. Compute the returns: $\delta_i(I_h) = \tilde{p}_i(\tilde{t}^i_{h+1}) - \tilde{p}_i(\tilde{t}^i_h)$

F.4. Compute the Fourier coefficients:

$$c_k^+(i) = \sum_{h=1}^{n_i-1} e^{\mathrm{i}k\tilde{t}^i_h}\delta_i(I_h); \quad c_k^-(i) = \sum_{h=1}^{n_i-1} e^{-\mathrm{i}k\tilde{t}^i_h}\delta_i(I_h)$$

end for

Step C: Convolution.

C.1. The Dirichlet implementation:

$$\hat{\Sigma}^{ij}_{n_i,n_j,N} = \frac{1}{2N+1}\sum_{k=-N}^{N}[c_k^+(i)c_k^-(j)]$$

C.2. The Fejér implementation:

$$\hat{\Sigma}^{ij}_{n_i,n_j,N} = \frac{1}{N+1}\sum_{k=-N}^{N}\left(1-\frac{|k|}{N}\right)[c_k^+(i)c_k^-(j)]$$

Correlation: $R_{ij} = \dfrac{\hat{\Sigma}^{ij}_{n_i,n_j,N}}{\sqrt{\hat{\Sigma}^{ii}_{n_i,N}}\sqrt{\hat{\Sigma}^{jj}_{n_j,N}}}$

return (**Σ**, **R**)

Algorithm 1: The Malliavin-Mancino estimators computes the Dirichlet or Fejér implementation of the Malliavin-Mancino estimator using a complex exponential formulation of the Fourier transform (Malliavin and Mancino, 2009). The algorithm is a mere sketch provided by Hendricks et al. (2017) and is based on their MATLAB implementation (Malherbe et al., 2005).

2.2.1 Implementation methods

Taking a closer look at Algorithm 1, computing eq. (2.4) for $k \in \{-N,...,N\}$ and $i = 1,...,D$ features is the computationally intensive aspect of the algorithm. Note that n_i is the sample dimension for price P^i and n_j that of the price P^j, which can be different. I will outline various methods to evaluate the Fourier coefficients along with their use-case, benefits, pitfalls and general algorithm complexity. The complexity is given only for the synchronous case where $n = n_1 = ... = n_D$, $N = \frac{n}{2}$, and $\min_h\{t^i_{h+1} - t^i_h\} = \max_h\{t^i_{h+1} - t^i_h\}$, $\forall i = 1,...,D$.

Here the use case refers to the ability to evaluate synchronous or asynchronous time-series data. Furthermore, asynchrony is subdivided into two representations. First, the *missing data representation* where observations are observed at non-random times equispaced over $[0,T]$ and some grid points are missing. This for example can be 1-minute bar data where some intervals may contain no trades to construct the bar. Second, the *arrival time representation* where observations are observed at random times with unequal spacing. This for example is tick-by-tick trade data.

2.2.1.1 Benchmark for-loop implementation

The Mancino et al. (2017) implementation from their book uses a vanilla for-loop construction. The evaluation relies on looping through $\{-N,...,N\}$ to compute the k^{th} Fourier mode. The implementation does not rely on any techniques to improve performance and will act as a benchmark to compare against other methods. The method can be used for all synchronous and asynchronous cases and the complexity is the same as Discrete Fourier Transforms (DFTs) with a complexity of $\mathcal{O}(n^2)$.

Require:
1. $\delta_i = (\delta_i(I_h))_{h=0}^{n_i-1}$: vector of source strengths for asset i.
2. $\mathbf{t}^i = (t^i_h)_{h=0}^{n_i-1}$: vector of re-scaled sample times for asset i.
3. N: the cutoff frequency.

for s =1 to $2N+1$ **do**

$k = s - N - 1$

$$c_s^+ = \sum_{h=0}^{n_i-1} e^{\mathrm{i}kt^i_h}\delta_i(I_h)$$

$$c_s^- = \sum_{h=0}^{n_i-1} e^{-\mathrm{i}kt^i_h}\delta_i(I_h)$$

end for

return (c_k^+, c_k^-)

Algorithm 2: The for-loop implementation from Mancino et al. (2017) computes the Fourier coefficients using for-loops. The Julia implementation can be found in MScorrDK.jl or MScorrFK.jl on the GitHub resource (Chang et al., 2020c) and correspond to the Dirichlet and Fejér representation respectively. The implementation is based on the MATLAB implementation from Mancino et al. (2017).

2.2.1.2 Vectorised implementation

The legacy code implementation is based on a MATLAB implementation by Malherbe et al. (2005) and Hendricks et al. (2017). The difference compared to the vanilla for-loop implementation is that all the Fourier modes are evaluated in parallel by vectorising the computation. Here I further improve upon the legacy code by exploiting techniques found in Mancino et al. (2017). Concretely, I exploit the Hermitian symmetry $\mathscr{F}(dX^i)(k) = \overline{\mathscr{F}(dX^i)(-k)}$ where $\overline{\mathscr{F}}$ denotes the conjugate function of $\mathscr{F}$. This

is possible because the source strengths $\delta_i(I_h)$ are all real-valued. Therefore, we only need to evaluate $k \in \{1,...,N\}$ and obtain the conjugates for these Fourier modes. Finally, $\mathscr{F}(dX^i)(0) = \sum_{h=0}^{n_i-1} \delta_i(I_h)/2\pi$ must be computed to complete the range of Fourier modes required for the convolution. The method can be used for all synchronous and asynchronous cases with a complexity of $\mathscr{O}(n^2)$ which is the same as DFTs. The key concern with this method is the memory usage constraints that it can face. Inspecting Algorithm 3 we see that a large matrix of size (nxN) is required for the vectorisation which can adversely affect performance by either:

i.) pre-maturely ending the computation due to either insufficient memory or heap-size constraints, or

ii.) slow down performance due to an over-reliance on virtual-memory management.

Hence care with regards to memory management is crucial for effective performance enhancement for large data-sets.

Require:
1. $\delta_i = (\delta_i(I_h))_{h=0}^{n_i-1}$: vector of source strengths for asset i.
2. $\mathbf{t}^i = (t_h^i)_{h=0}^{n_i-1}$: vector of re-scaled sample times for asset i.
3. N: the cutoff frequency.

Set: $\mathbf{k} = (1,2,\ldots,N)^T$ a column vector 1 to N.
Compute: $\mathbf{c}_{1:N} = \delta_i^\intercal \exp\left(-\mathrm{i}\ \mathbf{t}^i\ \mathbf{k}^\intercal\right)$
Compute: $c_0 = \sum_{h=1}^{n_i-1} \delta_i(I_h)$
Piece $\mathbf{c}_{1:N}$, $\overline{\mathbf{c}_{1:N}}$ and c_0 together to obtain c_k^+ and c_k^-
return (c_k^+, c_k^-)

Algorithm 3: The vectorised code implementation (Hendricks et al., 2017; Malherbe et al., 2005) replaces the for-loops by vectorising the computation of the Fourier coefficients and exploits the Hermitian symmetry of the real source strengths. The Julia implementation can be found in CFTcorrDK.jl or CFTcorrFK.jl on the GitHub resource (Chang et al., 2020c) and correspond to the Dirichlet and Fejér representation respectively.

2.2.1.3 The fast Fourier transform

The fast Fourier transform (FFT) are algorithms that efficiently evaluate DFTs. While DFTs have a complexity of $\mathscr{O}(n^2)$, FFTs have a complexity of $\mathscr{O}(n\log n)$. The FFT implementation we use here is the current state-of-the-art FFTW package by Frigo and Johnson (2005) based on the Cooley and Tukey (1965) algorithm, also known as the radix-2 decimation in time algorithm. The algorithm works by exploiting the *Danielson-Lanczos Lemma* and bit-reversal.

Adopting a more general notation, recall that a DFT is defined as:

$$F(k) = \sum_{h=0}^{n-1} f_h e^{-\frac{2\pi \mathrm{i}}{n} hk}, \tag{2.7}$$

where k ranges from $0,...,n-1$. Here $f_h = \delta_i(I_h)$. For simplicity, we assume that n is a power of 2. The Danielson-Lanczos Lemma allows us to split the n-point DFT in eq. (2.7) into two $n/2$-point DFTs as

follows:

$$\begin{aligned} F(k) &= \sum_{h=0}^{n-1} f_h e^{-\frac{2\pi \mathrm{i}}{n} hk} \\ &= \sum_{m=0}^{n/2-1} f_{2m} e^{-\frac{2\pi \mathrm{i}}{n}(2m)k} + \sum_{m=0}^{n/2-1} f_{2m+1} e^{-\frac{2\pi \mathrm{i}}{n}(2m+1)k} \\ &= \underbrace{\sum_{m=0}^{n/2-1} f_{2m} e^{-\frac{2\pi \mathrm{i}}{n/2} mk}}_{\text{n/2 DFT for even source points}} + e^{-\frac{2\pi \mathrm{i}}{n} k} \underbrace{\sum_{m=0}^{n/2-1} f_{2m+1} e^{-\frac{2\pi \mathrm{i}}{n/2} mk}}_{\text{n/2 DFT for odd source points}} \\ &= F^e(k) + e^{-\frac{2\pi \mathrm{i}}{n} k} F^o(k). \end{aligned} \tag{2.8}$$

This allows us to compute the Fourier modes for $k \in 0, ..., n/2-1$ using an even and odd $n/2$-point DFT. To obtain the remaining Fourier modes we rely on the periodicity of the complex exponential, specifically:

$$\begin{aligned} F\left(k+\frac{n}{2}\right) &= \sum_{m=0}^{n/2-1} f_{2m} e^{-\frac{2\pi \mathrm{i}}{n/2} m\left(k+\frac{n}{2}\right)} + e^{-\frac{2\pi \mathrm{i}}{n}\left(k+\frac{n}{2}\right)} \sum_{m=0}^{n/2-1} f_{2m+1} e^{-\frac{2\pi \mathrm{i}}{n/2} m\left(k+\frac{n}{2}\right)} \\ &= \sum_{m=0}^{n/2-1} f_{2m} e^{-\frac{2\pi \mathrm{i}}{n/2} mk} e^{-2\pi m \mathrm{i}} + e^{-\frac{2\pi \mathrm{i}}{n} k} e^{-\pi \mathrm{i}} \sum_{m=0}^{n/2-1} f_{2m+1} e^{-\frac{2\pi \mathrm{i}}{n/2} mk} e^{-2\pi m \mathrm{i}} \\ &= \sum_{m=0}^{n/2-1} f_{2m} e^{-\frac{2\pi \mathrm{i}}{n/2} mk} - e^{-\frac{2\pi \mathrm{i}}{n} k} \sum_{m=0}^{n/2-1} f_{2m+1} e^{-\frac{2\pi \mathrm{i}}{n/2} mk} \\ &= F^e(k) - e^{-\frac{2\pi \mathrm{i}}{n} k} F^o(k). \end{aligned} \tag{2.9}$$

Figure 2.1 demonstrates how we can reconstruct the n-point DFT using two $n/2$-point DFTs by applying the Danielson-Lanczos Lemma. Due to the shape of the data flow, this operation is also known as the *butterfly*.

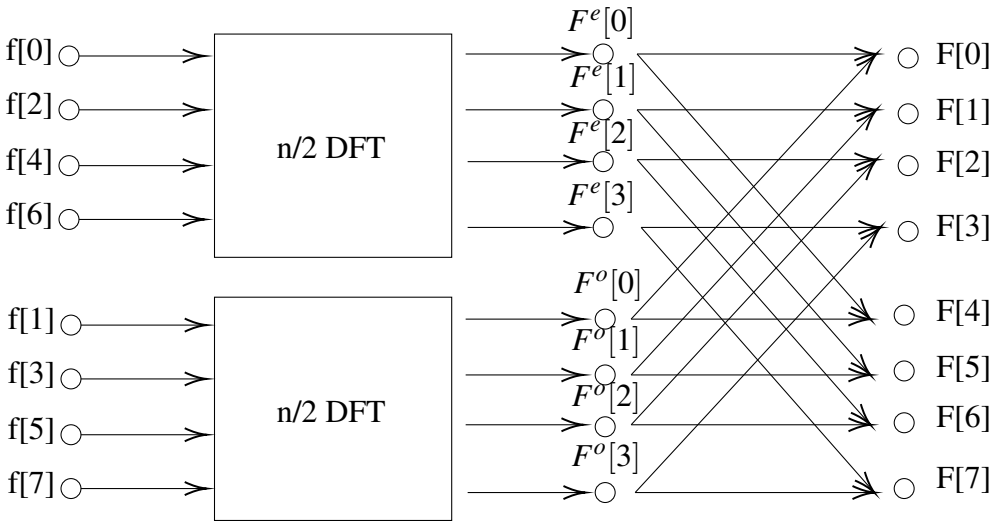


Figure 2.1: The figure demonstrates the reconstruction of the n-point DFT using two $n/2$-point DFTs by applying the Danielson-Lanczos Lemma.

The amazing aspect about this lemma is that it can be applied recursively until a one-point DFT is reached, which is just the source strength itself. Meaning

$$F^{eoo....e}(k) = f_h \quad \text{for some } h. \tag{2.10}$$

However, the lemma alone does not make the algorithm practical. We still need to assign each source strength to a one-point DFT from the sequences of even and odds during the recursion. This assignment is easily solved with bit-reversal. By taking the original vector and rearranging it into bit-revered order, we obtain the required vector for easy bookkeeping (Press et al., 1992). Now by combining adjacent one-point DFTs to get two-point DFTs, we can apply this recursively using the butterfly operation to obtain the required n-point DFT. Thus the decimation in time algorithm is complete.

Due to the radix-2 requirement, it is almost always recommended that one zero-pad the source points to the closest power of two. In the implementation, I let the FFTW package perform its various heuristics to optimize performance (Frigo and Johnson, 2005). Specifically, I do not *plan* the FFT in advance as that incurs an initialization cost and is only recommended for repeated transforms of the same size.

Finally, the implementation further exploits the Hermitian symmetry making this the fastest current implementation known to me. The key constraint relating to this method is its restriction to strictly synchronous data so that the evaluation becomes a simple discrete Fourier transform. Meaning that $\min_h\{t^i_{h+1} - t^i_h\} = \max_h\{t^i_{h+1} - t^i_h\} = 2\pi/n_i$, $\forall i = 1,...,D$. Therefore, eq. (2.4) can reduce to (ignoring the scaling factor):

$$\sum_{h=0}^{n_i-1} e^{-\mathrm{i}kt^i_h}\delta_i(I_h) = \sum_{h=0}^{n_i-1} e^{-\frac{2\pi\mathrm{i}}{n}hk}\delta_i(I_h).$$

2.2.1.4 The zero-padded fast Fourier transform

The first method to bring speed into the asynchronous case is the Zero-padded FFT (ZFFT). The implementation extends the FFT implementation by zero padding missing observations which allows for the naive computation for asynchrony using a missing data representation. By zero padding, we can recover the "uniform sampling" required for the FFT while ensuring that the contribution of the zero-padded $\delta_i(I_h)$ has no contribution to eq. (2.7). Thus allowing the correct recovery of eq. (2.4).

The implementation is achieved by computing the minimum sampling interval $\min_h\{t^i_{h+1} - t^i_h\} = \Delta t_0$ and creating a new over-sampled grid with intervals Δt_0. The observations are then placed at the nearest neighbour on the new over-sampled grid. The FFT algorithm is then applied to the new over-sampled grid. The implementation retains a complexity of $\mathcal{O}(n\log n)$ but is slower than the FFT implementation as it does not exploit the Hermitian symmetry and requires the additional step of creating an over-sampled grid. This is the benchmark asynchronous approach to the fast Fourier transform.

Figure 2.2 demonstrates two points:

i.) how the zero-padded implementation works, and

ii.) why the implementation does not work for the asynchronous case using an arrival time representation.

When asynchrony is induced using a missing data representation, the original grid has equal spacing Δt_0. Therefore the points on the over-sampled grid will be the same time points as the original grid, and the points will either have zero or the corresponding $\delta_i(I_h)$, *i.e.* there is no shifting of time points. However, when asynchrony is induced using an arrival time representation, the original grid does not have equal spacing Δt_0. Therefore the time points from the original grid will be shifted (Seen in the third arrow from the left in Figure 2.2).

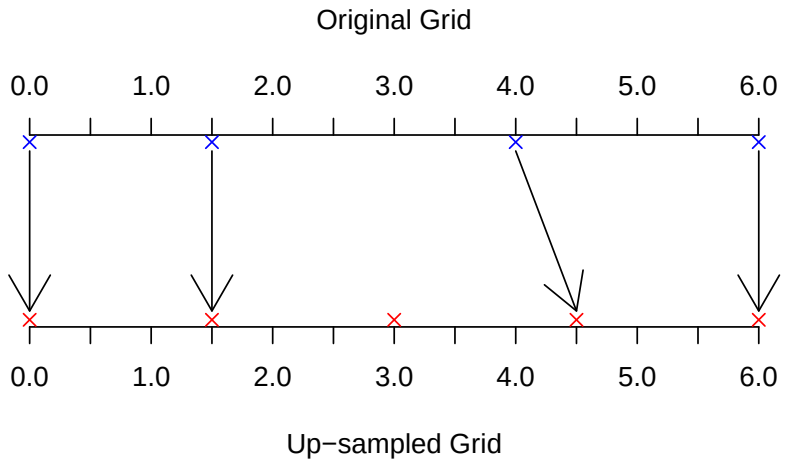


Figure 2.2: The figure is a toy example to show how the zero-padding works for the zero-padded FFT implementation. The minimum sampling interval $\Delta t_0 = 1.5$. A new uniform over-sampled grid is created and observations are placed on the nearest neighbouring point on the over-sampled grid.

Require:

1. $\delta_i = (\delta_i(I_h))_{h=0}^{n_i-1}$: vector of source strengths for asset i.
2. $\mathbf{t}^i = (t_h^i)_{h=0}^{n_i-1}$: vector of re-scaled sample times for asset i ($t_h^i \in [0, 2\pi]$).
3. $N^* = \lfloor \frac{2\pi}{\Delta t_0} \rceil$, where $\lfloor x \rceil$ denotes rounding x to the nearest Integer and Δt_0 is the minimum distance between sampled times from Algorithm 15.

Initialise: $(\tilde{f}_\ell)_{\ell=1}^{N^*} = \mathbf{0}$, a zero vector of length N^*.
for $h = 1$ to $n_i - 1$ **do**
  $\ell = \lfloor \frac{t_h^i N^*}{2\pi} \rceil + 1$
  $\tilde{f}_\ell = \delta_i(I_h)$
end for
return ($\tilde{f}_\ell$ for FFT computation)

Algorithm 4: The zero-padded FFT implementation creates a uniform grid and shifts the non-uniform source points to the nearest grid point on an up-sampled uniform grid. The Julia implementation can be found in FFTZPcorrDK.jl or FFTZPcorrFK.jl on the GitHub resource (Chang et al., 2020c) and correspond to the Dirichlet and Fejér representation respectively. Note that the index ℓ is set for languages with array indices starting from 1.

2.2.1.5 Non-uniform fast Fourier transform

The Non-Uniform fast Fourier transform (NUFFT) implementation of the Malliavin-Mancino estimator is the main contribution of this chapter. We want a fast algorithm to evaluate eq. (2.4) when $(t_h^i)_{h=1,\dots,n_i}$ are non-uniformly spaced in $[0, 2\pi]$. This can be achieved by using the 1-dimensional "type 1" NUFFT (Greengard and Lee, 2004; Barnett et al., 2018) (also known as the adjoint NUFFT (Potts and Steidl, 2003)). Recall that we are interested in the evaluation of:

$$\mathscr{F}(dX^i)(k) = \frac{1}{2\pi} \sum_{h=1}^{n_i-1} \delta_i(I_h) e^{-\mathrm{i}kt_h^i}, \tag{2.11}$$

for $k \in \{-N, \dots, N\}$, and $i = 1, \dots, D$ features. We use the more popular NUFFT approach which can be summarised in the following three steps (Greengard and Lee, 2004; Potts and Steidl, 2003; Barnett et al., 2018):

i.) convolve the non-uniform source points onto an over-sampled uniform grid with a kernel function $\varphi(\cdot)$,

ii.) perform FFT on the uniform up-sampled grid, and

iii.) deconvolve the effects of the convolution in the Fourier space.

Step 1 (spreading)

Equation (2.11) describes the exact Fourier coefficients of the function

$$f_{dX^i}(x) = \sum_{h=0}^{n_i-1} \delta_i(I_h)\delta(x-t_h^i), \tag{2.12}$$

viewed as a periodic function over the domain with periodicity p, and $\delta(x)$ denotes the delta function (unit impulse). This function is clearly not well-resolved by a uniform mesh in x.

Let $\varphi(\cdot)$ be the choice of kernel function with periodicity p and defined on the same domain as t_h^i. Then its periodisation is given as:

$$\tilde{\varphi}(x) = \sum_{r=-\infty}^{\infty} \varphi(x-rp). \tag{2.13}$$

If we define $f_{\varphi,dX^i}(x)$ to be the convolution

$$f_{\varphi,dX^i}(x) = f_{dX^i} * \tilde{\varphi}(x) = \int_{\mathscr{D}} f_{dX^i}(y)\tilde{\varphi}(x-y)dy, \tag{2.14}$$

where $\mathscr{D}$ is the domain in which t_h^i and $\varphi(\cdot)$ are defined. Then f_{φ,dX^i} is p-periodic C^∞ function and can be well-resolved by a uniform mesh in x.

Step 2 (FFT)

With eq. (2.14), we can obtain the Fourier coefficients of $f_{\varphi,dX^i}(x)$ as:

$$F_\varphi(dX^i)(k) = \frac{1}{2\pi}\int_0^{2\pi} f_{\varphi,dX^i}(x)e^{-\mathrm{i}kx}dx, \tag{2.15}$$

using a standard FFT on the over-sampled grid.

Step 3 (correction)

Once we have obtained $F_\varphi(dX^i)(k)$, we can compute $F(dX^i)(k)$:

$$F(dX^i)(k) = \frac{F_\varphi(dX^i)(k)}{\hat{\varphi}(k)}, \tag{2.16}$$

as a consequence from the convolution theorem.

To set the theoretical scene for the practical implementation, let $M_N = 2N+1$ be the number of Fourier modes we want returned, σ be the over-sampling ratio,[3] ξ_ℓ be the ℓ^{th} location on the over-sampled grid with $\ell \in \{0,...,M_r - 1 = \sigma M_N - 1\}$ and ω is the spreading width with M_{sp} as the spreading in each direction.

[3]The over-sampling ratio plays a role in controlling the accuracy and speed of NUFFT methods. The larger the ratio, the more accurate, but suffers in terms of speed due to having to interpolate observations onto a finer grid. Therefore, σ requires careful balance. Most studies have settled on $\sigma = 2$ (Barnett et al., 2018).

Let us consider the three most popular kernels: the Gaussian kernel using the fast Gaussian gridding implementation from Greengard and Lee (2004), the Kaiser-Bessel kernel using the implementation approach of Potts and Steidl (2003), and the exponential of semi-circle used by Barnett et al. (2018) in their the state-of-the-art FINUFFT package.

The Gaussian kernel and its Fourier transform is defined as:

$$\varphi_G(x) = e^{-x^2/4\tau}, \tag{2.17}$$

and

$$\hat{\varphi}_G(k) = \sqrt{2\pi} e^{-k^2\tau}, \tag{2.18}$$

where τ is defined as

$$\tau = \frac{1}{M^2} \frac{\pi}{\sigma(\sigma - 0.5)} M_{sp}. \tag{2.19}$$

The Kaiser-Bessel pair is defined as:

$$\varphi_{KB}(x) = \frac{1}{\pi} \begin{cases} \frac{\sinh\left(b\sqrt{M_{sp}^2 - M_r^2 x^2}\right)}{\sqrt{M_{sp}^2 - M_r^2 x^2}} & |x| \leq \frac{M_{sp}}{M_r}, \\ \frac{\sin\left(b\sqrt{M_r^2 x^2 - M_{sp}^2}\right)}{\sqrt{M_r^2 x^2 - M_{sp}^2}} & \text{otherwise}, \end{cases} \tag{2.20}$$

and

$$\hat{\varphi}_{KB}(k) = \frac{1}{M_r} I_0\left(m\sqrt{b^2 - (2\pi k/M_r)^2}\right), \tag{2.21}$$

where $b = \pi\left(2 - \frac{1}{\sigma}\right)$ and $I_0(\cdot)$ is the modified zero-order Bessel function (Potts and Steidl, 2003). Finally, the exponential of semi-circle pair is defined as:

$$\phi_{ES}(x) = \begin{cases} e^{\beta\left(\sqrt{1-x^2}-1\right)} & |x| \leq 1, \\ 0 & \text{otherwise}, \end{cases} \tag{2.22}$$

and

$$\hat{\phi}_{ES}(k) = \int_{-\infty}^{\infty} \phi_{ES}(x) e^{\mathrm{i}kx} dx, \tag{2.23}$$

where $\beta = 2.3\omega$. The kernel is re-scaled to have support between $[-\alpha, \alpha]$ with $\alpha = \pi\omega/M_r$. Thus the re-scaled kernel is:

$$\varphi_{ES}(x) = \phi_{ES}(x/\alpha), \tag{2.24}$$

and

$$\hat{\varphi}_{ES}(k) = \alpha\hat{\phi}_{ES}(\alpha k). \tag{2.25}$$

The exponential of semi-circle kernel has no known analytic Fourier transform, therefore numerical integration is used to obtain $\hat{\varphi}_{ES}(k)$. See Barnett et al. (2018) for their elegant approach in computing eq. (2.25).

We focus our attention on the implementation for the various Kernels which have different periodicity. The Gaussian and exponential of semi-circle are 2π-periodic with domain on $[0, 2\pi]$ (Greengard and Lee, 2004; Barnett et al., 2018), while the Kaiser-Bessel kernel is 1-periodic with domain on $[0, 1]$. Therefore ξ_ℓ, $t_h^i \in [0, 2\pi]$ for the Gaussian and exponential of semi-circle kernel, and ξ_ℓ, $t_h^i \in [0, 1]$ for

the Kaiser-Bessel kernel.[4]

Concretely, using eq. (2.13) we can compute the source strength on the over-sampled grid, given by the periodic discrete convolution

$$f_{\varphi,dX^i}(\xi_\ell) = \sum_{h=0}^{n_i-1} \delta_i(I_h)\tilde{\varphi}(\xi_\ell - t_h^i), \quad \text{for } \ell = 0,...,M_r-1. \tag{2.26}$$

Now the DFT of the over-sampled grid in eq. (2.26) can be evaluated using the standard FFT:

$$F_\varphi(dX^i)(k) = \frac{1}{M_r}\sum_{l=0}^{M_r-1} f_{\varphi,dX^i}(\xi_\ell)e^{-2\pi \mathrm{i}k\ell/M_r}, \quad \text{for } k = -M_r/2,...,M_r/2-1. \tag{2.27}$$

Finally, we correct for the convolution using eq. (2.16) and retain the M_N central frequencies (Barnett et al., 2018). Namely, $k = -N,...,N$.

The Fourier coefficient $F(dX^i)(k)$ in eq. (2.16) is the evaluation of $\mathscr{F}(dX^i)(k)$ in eq. (2.11) using non-uniform fast Fourier techniques. The level of numerical accuracy between eq. (2.16) and eq. (2.11) can be measured as the relative ℓ^2-norm in the output vector defined as:

$$\varepsilon = \frac{\|\mathbf{F}(dX^i) - \boldsymbol{\mathcal{F}}(dX^i)\|_2}{\|\boldsymbol{\mathcal{F}}(dX^i)\|_2}. \tag{2.28}$$

Moreover, the desired level of accuracy can be controlled by the amount of spreading in each direction M_{sp} (in terms of number of grid points). We found that setting $M_{sp} = \lfloor \frac{-\ln(\varepsilon)(\sigma-1/2)}{(\pi(\sigma-1))} + \frac{1}{2} \rfloor$ for the Gaussian kernel, $M_{sp} = \lfloor \frac{1}{2}(\lceil \log_{10}(\frac{1}{\varepsilon}) \rceil + 2) \rfloor$ for the Kaiser-Bessel kernel, and $M_{sp} = \lfloor \frac{1}{2}(\lceil \log_{10}(\frac{1}{\varepsilon}) \rceil + 2) \rfloor + 2$ for the exponential of semi-circle kernel allow us to achieve the desired relative error level.[5]

Remark 2.2.3 *Notice the notational difference between $F(dX^i)(k)$ in eq. (2.16) and $\mathscr{F}(dX^i)(k)$ in eq. (2.11). The previously discussed methods are direct evaluations of $\mathscr{F}(dX^i)(k)$ without manipulating the data beforehand (a main property of the Malliavin-Mancino estimator). Here $F(dX^i)(k)$ is the evaluation of $\mathscr{F}(dX^i)(k)$ using non-uniform fast Fourier techniques. Due to the spreading step, the NUFFT implementation is theoretically a "new" estimator. Even though it is an algorithm which combines the convolution theorem with FFTs.*

Remark 2.2.4 *Notice how the support of the kernels (τ for the Gaussian, the cut-off for the Kaiser-Bessel and α for the exponential semi-circle) all depend on M_r or M_N and σ. This is to ensure that the up-sampled grid can resolve the convoluted function in eq. (2.14).*

Remark 2.2.5 *The choice of kernel has a fascinating history and has a significant impact on the speed and accuracy of NUFFTs. The Gaussian kernel was the first kernel choice with rigorous analysis done by Dutt and Rokhlin (1993). Before that, it was realised by Jackson et al. (1991) that a good kernel function should have (Barnett et al., 2018):*

i.) small numerical support to ensure the spreading cost is low, and

ii.) be smooth to ensure that the DFT aliasing error is small.

However, these criteria are conflicting. It was known that the family of prolate spheroidal wavefunctions (PSWF) of order zero would optimise the above criteria, but the PSWF requires around 600 lines of code.

[4] Potts and Steidl (2003) have domain on $[-\frac{1}{2},\frac{1}{2}]$, but I change it to $[0,1]$ for easier implementation. The actual domain is not important provided the periodicity is correct, this is because all that matters is the distances between t_h^i and ξ_ℓ.

[5] I tuned the M_{sp} such that it always strictly achieves the desired error level. Note that my choice of M_{sp} is stricter than that in the literature. Specifically, Barnett et al. (2018) set $\omega = \lceil \log_{10}(\frac{1}{\varepsilon}) \rceil + 1$. NUFFT error.jl is the test script to check that the desired relative ℓ^2 error is strictly achieved and can be found in the GitHub resource (Chang et al., 2020c).

It was then realised that the Kaiser-Bessel of order zero is a good approximation of the PSWF but only requiring around 10 lines of code. This lead to the use of the Kaiser-Bessel kernel. The contribution of Barnett et al. (2018) is that they realised the exponential of semi-circle kernel is a good approximation of the Kaiser-Bessel kernel and requires even less code. Moreover, they overcome the additional cost of having to numerically evaluate eq. (2.25) by exploiting a Gauss-Legendre quadrature with "phase winding".

At first glance, eq. (2.26) seems a lot more expensive than it actually is. This is based on two observations: first, the kernels in equation eqs. (2.17), (2.20) and (2.22) are sharply peaked in a manner such that the contribution of $\delta_i(I_h)$ to grid points outside the kernel width is zero (the kernels have small numerical support). Second, the evaluation of $\tilde{\varphi}(\cdot)$ is unnecessary; we only need to evaluate $\varphi(\cdot)$ (See Figure 2.3). This is because the purpose of $\tilde{\varphi}(\cdot)$ is to account for the periodicity when spreading near the end points of the over-sampled grid. Using these observations, we can efficiently implement eq. (2.26) by looping through the source points. Find the nearest up-sampled grid point ξ_{ℓ^*} that is less than or equal to t_h^i. Spread to the $s \in \{-M_{sp}, ..., M_{sp}\}$ nearest grid points ξ_{ℓ^*-s} with $\delta_i(I_h)\varphi(t_h^i - \xi_{\ell^*} - \frac{sp}{M_r})$, subject to the condition that when $\ell^* - s < 0$ the index becomes $\ell^* - s + M_r$, and when $\ell^* - s \geq M_r$ the index becomes $\ell^* - s - M_r$ to account for the correct indices due to the periodicity.

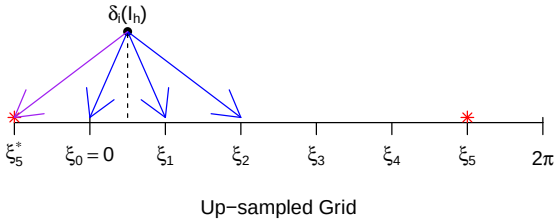


Figure 2.3: The figure is a toy example to show how the spreading works for a single source point t_h^i. The up-sampled grid is denoted by ξ_ℓ, and the source strength is spread to the nearest grid points as $\delta_i(I_h)\varphi(\xi_\ell - t_h^i)$. The figure aims to show that we only need to evaluate $\varphi(\xi_\ell - t_h^i)$ instead of $\tilde{\varphi}(\xi_\ell - t_h^i)$. The grid points ξ_5^* and ξ_5 (denoted by a red star) is the same point due to the periodicity, but the distance between $\xi_5 - t_h^i$ is large resulting in $\varphi(\xi_5 - t_h^i) \approx 0$. $\tilde{\varphi}(\xi_5 - t_h^i)$ fixes this by accounting for the periodicity. We can reduce the unnecessary computation of eq. (2.13) by contributing $\delta_i(I_h)$ to $f_{\varphi,dX^i}(\xi_5)$ with $\delta_i(I_h)\varphi(\xi_5^* - t_h^i)$.

The method can be used for all synchronous and asynchronous cases and has a complexity of $\mathcal{O}(M_r \log M_r + n|\log(\varepsilon)|)$ (Barnett et al., 2018).

Require:

1. $\delta_i = (\delta_i(I_h))_{h=0}^{n_i-1}$: vector of source strengths for asset i.
2. $\mathbf{t}^i = (t_h^i)_{h=0}^{n_i-1}$: vector of re-scaled sample times for asset i ($t_h^i \in [0, 2\pi]$).
3. $M_N = 2N+1$: the number of Fourier modes computed.
4. ε: error tolerance.

Step I. Initialisation:

I.1. Set: $\sigma = 2$.

I.2. Set: $M_r = \sigma M_N$; $M_{sp} = \lfloor \frac{1}{2}(\lceil \log_{10}(\frac{1}{\varepsilon}) \rceil + 2) \rfloor + 2$.

I.3. Initialise: $(\tilde{f}_\ell)_{\ell=1}^{M_r} = \mathbf{0}$, a zero vector of length M_r.

Step C: Convolution (See eq. (2.26)).

for $h = 0$ to $n_i - 1$ **do**

$b_0 = \lfloor t_h^i M_r \rfloor$, index of nearest up-sampled grid $\xi_{b_0} \leq t_h^i$.

$\mathrm{d} = t_h^i - \frac{b_0}{M_r}$.

$b_d = \min(M_{sp} - 1, b_0)$; $b_u = \min(M_{sp}, M_r - b_0 - 1)$.

for $k = -M_{sp}$ to $-b_d - 1$ **do**

$\tilde{f}_{b_0+k+M_r+1} = \tilde{f}_{b_0+k+M_r+1} + \delta_i(I_h)\varphi_{ES}(d - \frac{2\pi k}{M_r})$.

end for

for $k = -b_d$ to b_u **do**

$\tilde{f}_{b_0+k+1} = \tilde{f}_{b_0+k+1} + \delta_i(I_h)\varphi_{ES}(d - \frac{2\pi k}{M_r})$.

end for

for $k = b_u + 1$ to M_{sp} **do**

$\tilde{f}_{b_0+k-M_r+1} = \tilde{f}_{b_0+k-M_r+1} + \delta_i(I_h)\varphi_{ES}(d - \frac{2\pi k}{M_r})$.

end for

end for

Step F: Compute FFT on over-sampled grid (See eq. (2.27)).

F.1. Find Fourier coefficients $F_{ES}(dX^i)(k)$ via FFT on the grid $\tilde{f}_\ell$.

Step D: Deconvolution (See eq. (2.16)).

D.1. Compute: $\hat{\varphi}_{ES}(k) = \alpha \int_{-\infty}^{\infty} \phi_{ES}(x) e^{\mathrm{i}\alpha k x} dx$ using numerical integration.

D.2. Compute: $F(dX^i)(k) = \frac{2\pi}{M_r} F_{ES}(dX^i)(k) / \hat{\varphi}_{ES}(k)$.

return $(F(dX^i)(k), k \in \{-N, ..., N\})$

Algorithm 5: The Exponential of semi-circle NUFFT implementation uses the interpolate and deconvolve approach with the Exponential of semi-circle kernel. The algorithm is a naive implementation based on the steps provided in Barnett et al. (2018) and does not exploit the piecewise polynomial kernel approximation nor the Gauss-Legendre quadrature for implementation acceleration used in FINUFFT (Barnett et al., 2018). The implementation relies on the QuadGK package to perform the numerical integration using adaptive Gauss-Kronrod quadrature. The Julia implementation can be found in NUFFT-ES.jl on the GitHub resource (Chang et al., 2020c). The algorithm is set for languages with array indices starting from 1.

Require:

1. $\delta_i = (\delta_i(I_h))_{h=0}^{n_i-1}$: vector of source strengths for asset i.
2. $\mathbf{t}^i = (t_h^i)_{h=0}^{n_i-1}$: vector of re-scaled sample times for asset i ($t_h^i \in [0,1]$).
3. $M_N = 2N+1$: the number of Fourier modes computed.
4. ε: error tolerance.

Step I. Initialisation:

I.1. Set: $\sigma = 2$.

I.2. Set: $M_r = \sigma M_N$; $M_{sp} = \lfloor \frac{1}{2}(\lceil \log_{10}(\frac{1}{\varepsilon}) \rceil + 2) \rfloor$.

I.3. Initialise: $(\tilde{f}_\ell)_{\ell=1}^{M_r} = \mathbf{0}$, a zero vector of length M_r.

Step C: Convolution (See eq. (2.26)).

for $h = 0$ to $n_i - 1$ **do**

$b_0 = \lfloor t_h^i M_r \rfloor$, index of nearest up-sampled grid $\xi_{b_0} \leq t_h^i$.

$\mathrm{d} = t_h^i - \frac{b_0}{M_r}$.

$b_d = \min(M_{sp} - 1, b_0)$; $b_u = \min(M_{sp}, M_r - b_0 - 1)$.

for $k = -M_{sp}$ to $-b_d - 1$ **do**

$\tilde{f}_{b_0+k+M_r+1} = \tilde{f}_{b_0+k+M_r+1} + \delta_i(I_h)\varphi_{KB}(d - \frac{k}{M_r})$.

end for

for $k = -b_d$ to b_u **do**

$\tilde{f}_{b_0+k+1} = \tilde{f}_{b_0+k+1} + \delta_i(I_h)\varphi_{KB}(d - \frac{k}{M_r})$.

end for

for $k = b_u + 1$ to M_{sp} **do**

$\tilde{f}_{b_0+k-M_r+1} = \tilde{f}_{b_0+k-M_r+1} + \delta_i(I_h)\varphi_{KB}(d - \frac{k}{M_r})$.

end for

end for

Step F: Compute FFT on over-sampled grid (See eq. (2.27)).

F.1. Find Fourier coefficients $F_{KB}(dX^i)(k)$ via FFT on the grid $\tilde{f}_\ell$.

Step D: Deconvolution (See eq. (2.16)).

D.1. Compute: $F(dX^i)(k) = \frac{1}{M_r} F_{KB}(dX^i)(k) / \hat{\varphi}_{KB}(k)$.

return $(F(dp_i)(k), k \in \{-N, ..., N\})$

Algorithm 6: The Kaiser-Bessel NUFFT implementation uses the interpolate and deconvolve approach with the Kaiser-Bessel kernel. The algorithm can be applied to any kernel that is 1-periodic. The algorithm is optimised by removing the unnecessary computation of $\tilde{\varphi}(\cdot)$ as seen in Figure 2.3. However, the algorithm is structured without the precomputation step used in Potts and Steidl (2003) and evaluates the algorithm "on-the-fly". The Julia implementation can be found in NUFFT-KB.jl on the GitHub resource (Chang et al., 2020c). The algorithm is set for programming languages with array indices starting from 1.

Require:

1. $\delta_i = (\delta_i(I_h))_{h=0}^{n_i-1}$: vector of source strengths for asset i.
2. $\mathbf{t}^i = (t_h^i)_{h=0}^{n_i-1}$: vector of re-scaled sample times for asset i ($t_h^i \in [0,2\pi]$).
3. $M_N = 2N+1$: the number of Fourier modes computed.
4. ε: error tolerance.

Step I. Initialisation:

I.1. Set: $\sigma = 2$.

I.2. Set: $M_r = \sigma M_N$; $M_{sp} = \lfloor \frac{-\ln(\varepsilon)(\sigma-1/2)}{(\pi(\sigma-1))} + \frac{1}{2} \rfloor$.

I.3. Set: $\lambda = \frac{\sigma^2 M_{sp}}{\sigma(\sigma-0.5)}$; $h_x = \frac{2\pi}{M_r}$; $t_1 = \frac{\pi}{\lambda}$.

I.4. Set: $\tau = \frac{\pi\lambda}{M_r^2}$.

I.5. Initialise: $(\tilde{f}_\ell)_{\ell=1}^{M_r} = \mathbf{0}$, a zero vector of length M_r.

for k = 1 to M_{sp} **do**

$E_{3,k} = \exp\left(-t_1 k^2\right)$

end for

Step C: Convolution (See eq. (2.26)).

for h = 0 to $n_i - 1$ **do**

$b_0 = \lfloor \frac{t_h^i}{h_x} \rfloor$, index of nearest up-sampled grid $\xi_{b_0} \leq t_h^i$.

d = $\frac{t_h^i}{h_x} - b_0$.

$E_0 = \mathbf{0}$, a zero vector of length $2M_{sp}$.

$E_1 = e^{-t_1 \mathrm{d}^2}$; $E_{0,M_{sp}} = E_1$; $E_2 = e^{2t_1 \mathrm{d}}$.

for k = 1 to M_{sp} **do**

$E_{0,M_{sp}+k} = E_{3,k} E_1 E_2^k$.

end for

for k = 1 to $M_{sp} - 1$ **do**

$E_{0,M_{sp}-k} = E_{3,k} E_1 E_2^{-k}$.

end for

$b_d = \min(M_{sp} - 1, b_0)$; $b_u = \min(M_{sp}, M_r - b_0 - 1)$.

for $k = -M_{sp} + 1$ to $-b_d - 1$ **do**

$\tilde{f}_{b_0+k+M_r+1} = \tilde{f}_{b_0+k+M_r+1} + \delta_i(I_h) E_{0,M_{sp}+k}$.

end for

for $k = -b_d$ to b_u **do**

$\tilde{f}_{b_0+k+1} = \tilde{f}_{b_0+k+1} + \delta_i(I_h) E_{0,M_{sp}+k}$.

end for

for $k = b_u + 1$ to M_{sp} **do**

$\tilde{f}_{b_0+k-M_r+1} = \tilde{f}_{b_0+k-M_r+1} + \delta_i(I_h) E_{0,M_{sp}+k}$.

end for

end for

Step F: Compute FFT on over-sampled grid (See eq. (2.27)).

F.1. Find Fourier coefficients $F_G(dX^i)(k)$ via FFT on the grid $\tilde{f}_\ell$.

Step D: Deconvolution (See eq. (2.16)).

D.1. Compute: $F(dX^i)(k) = \sqrt{\frac{\pi}{\tau}} e^{k^2 \tau} F_G(dX^i)(k) \frac{1}{M_r}$.

return $(F(dX^i)(k), k \in \{-N, ..., N\})$

Algorithm 7: The Fast Gaussian Gridding NUFFT implementation uses the interpolate and deconvolve approach with the Gaussian kernel. The algorithm is specific for the Gaussian kernel as it uses the fast Gaussian gridding implementation by Greengard and Lee (2004) to reduce the number of exponential evaluations. The Julia implementation can be found in NUFFT-FGG.jl on the GitHub resource (Chang et al., 2020c). The algorithm is a replication of the FORTRAN source code from Greengard and Lee (2004), but adjusted for programming languages with array indices starting from 1.

2.2.2 Insights from NUFFTs

The non-uniform FFT methods present not only a speed advantage, but it also provides insight in:

i.) the averaging scale N investigated using the Malliavin-Mancino estimator. This was used by Renò (2003) and Precup and Iori (2007) where they used the choice cutting frequency N to investigate different time-scales, and

ii.) interpolation of financial data.

First, the Malliavin-Mancino estimator aims to represent the Fourier coefficients of the volatility process as a function of the Fourier coefficients of the price process. Therefore, investigation into different time-scales of the volatility process is limited to the sampling rate of the price process. The highest sampling rate present in the data is N_0, therefore the Nyquist frequency is $0.5N_0 = N$—the highest component frequency we can investigate without introducing aliasing. Meaning we are band-limited to frequencies $\leq N$.

To reconstruct the volatility process at the Nyquist frequency, we require at least $2N$ samples. This condition is satisfied by construction of the *Bohr convolution product*, with Fourier modes ranging from $\{-N,...,N\}$—resulting in a sampling frequency $M_N = 2N+1$ samples.

The relation between the number of Fourier modes and the sampling interval is simply $\frac{T}{M_N} = \Delta t$, this is a consequence of the implicit time-scale related to DFTs. Therefore, we can investigate different time-scales by investigating different frequency ranges. This is due to a consequence of the sampling theorem—which in essence states that to perfectly reconstruct a certain frequency, one needs at least twice the amount of samples. Meaning by reducing the number of Fourier modes (investigating larger time-scales), we are reducing the number of samples and thereby aliasing the larger frequencies. With this in mind, we are able to investigate different time-scales by perfectly reconstructing frequencies $< 0.5M_N$, through the cost of aliasing frequencies $\geq 0.5M_N$.

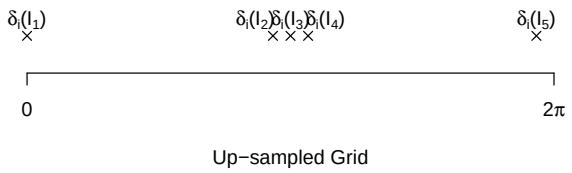


Figure 2.4: A toy example is used to show how the choice of N has an impact on averaging and the time-scale. For fixed $M_{sp} = 6$, when N is small (*e.g.* $N = 5$) M_r is also small ($M_r = 22$). Meaning the M_r grid points will be more spaced out and each M_r grid point will have contributions from most of the source strengths $\delta_t(I_h)$; therefore, investigating smaller time-scales by averaging. On the other hand, when N is large (*e.g.* $N = 40$) M_r is also large ($M_r = 162$). Meaning the M_r grid points will be more closely spaced and only some of the M_r grid points will have contributions from separate source points and majority of the M_r grid points will have no contributions. This illustrates the intuitive idea of how changing N allows one to investigate different time-scales using the ideas from NUFFTs.

The insight of NUFFT methods, is that the relation between N and the time-scale is demonstrated more intuitively (See Figure 2.4). For fixed M_{sp}, when N is small M_r is also small. Therefore, the M_r grid points will be more spread out and each grid point will have contributions from multiple source strengths averaged based on the choice of kernel $\varphi(\cdot)$. While for the case when N is large M_r will also be large. Meaning the M_r grid points are more tightly packed and fewer grid points will have contributions from separate source strengths, essentially there is less averaging.

Second, the interpolation (convolution) is explicit in NUFFT methods. This is interesting because interpolation of financial data can result in estimates being biased such as linear interpolation (Barucci and Renò, 2002) or interpolation based on underlying continuity assumptions such as the Hayashi-Yoshida estimator (Chang et al., 2019b). We argue these methods are flawed because they do not account for

the effect of interpolation; whereas NUFFT methods account for this by deconvolving the interpolation effects in the Fourier space.

Remark 2.2.6 *When I refer to bias in the Hayashi-Yoshida estimator, it is not in the traditional statistical sense because the Hayashi-Yoshida is indeed an unbiased estimator (Hayashi and Yoshida, 2005) in the traditional sense. What I mean here is that because it corrects for the Epps effect through its multiple contributions using underlying assumptions of continuity, it does not correctly recover the correlations from the observables which should demonstrate the Epps effect. The estimator is biased from the strictly data-informed perspective. As we shall see in Chapter 4, although the Epps effect is a bias in the perspective of Brownian diffusions (when trying to recover the underlying correlation of the synchronous process), the effect of asynchrony (when viewed from a data-informed perspective) can be quantified. Therefore, under a data-informed approach, this theoretical Epps effect is in fact the ground truth.*

Remark 2.2.7 *When I refer to NUFFT methods not presenting a bias, this is with respect to the estimates achieved with the direct evaluation of eq. (2.4) in the Malliavin-Mancino estimator as we shall see in Section 2.3.2. Testing the accuracy of NUFFT methods with respect to the direct evaluation is critical. This is because one of the main characteristics of the Malliavin-Mancino estimators is that it does not require the manipulation of the original data in the computation of eq. (2.4). This manipulation of data is explicit when computing eq. (2.16). Therefore, we need to perform numerous checks to ensure the NUFFT methods recover the correct estimates.*

2.3 Algorithm Performance and Benchmarking

Here the benchmarking is done using Monte Carlo simulations. All the seeds for replicating the work are in the respective script files in the GitHub resource (Chang et al., 2020c). Let us investigate the various factors influencing the speed and accuracy of the algorithms. To this end, let us use the Geometric Brownian Motion (GBM) which satisfies the following Stochastic Differential Equations (SDEs):

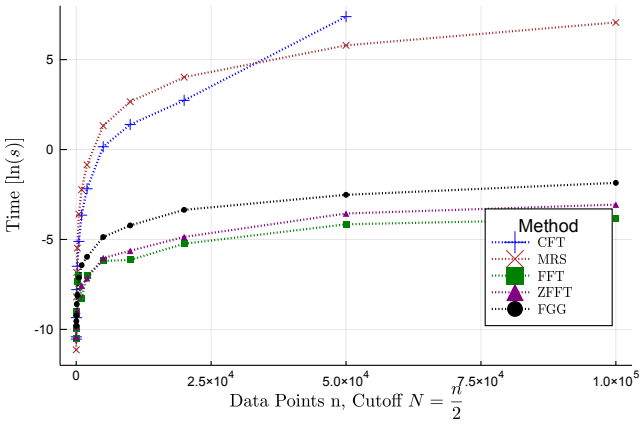
$$\frac{dP_t^i}{P_t^i} = \mu_i dt + \sigma_i dW_t^i, \quad i = 1,...,D, \tag{2.29}$$

with $\text{Corr}(dW^i, dW^j) = \rho^{ij}$. The GBM is simulated using the Euler–Maruyama scheme (which is strong order 0.5 in the sense of Kloeden and Platen (2013)) with equal spacing Δt between the observations (See Algorithm 19) and are at the same time across the features. This is the synchronous case. Asynchrony is then induced from the synchronous case using two approaches: *i.)* the missing data representation, and *ii.)* the arrival time representation. The missing data representation is obtained by removing a percentage of random observations. The arrival time representation is obtained by sampling each of the synchronous price paths using an exponential inter-arrival with rate λ_i (this is also known as Poissonian sampling).

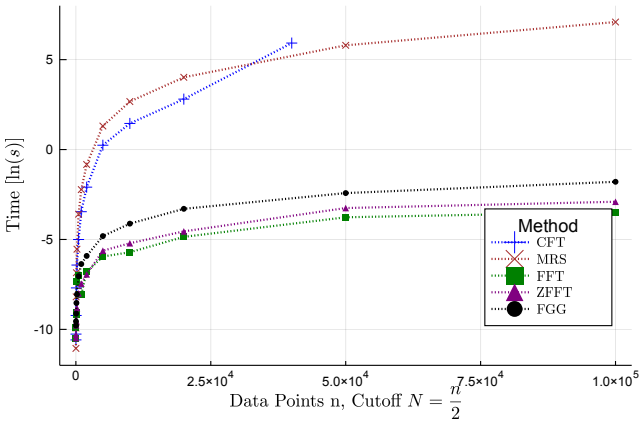
2.3.1 Benchmark Timing

The common factors affecting the computation time for all the algorithms are: *i.)* the number of data points n, *ii.)* the number of Fourier coefficients $M_N = 2N + 1$, and *iii.)* the number of features D. The parameter specific to the non-uniform FFT method is the tolerance ε which determines the spreading width ω for the averaging kernels.

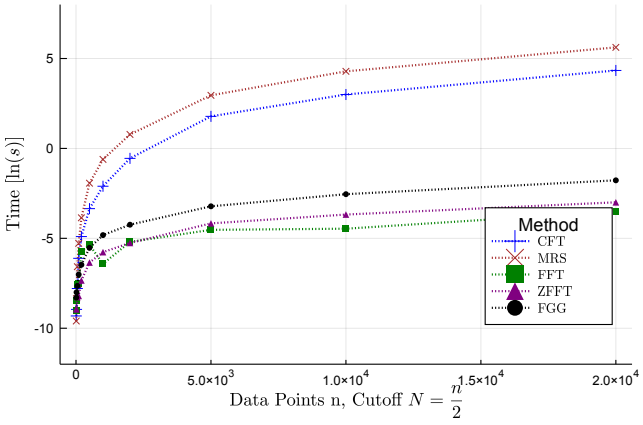
The benchmarking is done using a 2.5GHz base clock speed Quad-Core Intel i7-4870HQ with 16GB of 1600MHz DDR3L (CL=11) RAM on MacOS version 10.15.1 with JuliaPro version 1.2.0. GCC8 (this has since been changed to GCC9) is used as a requirement for the Julia interface to FINUFFT provided by af Klinteberg (2018).



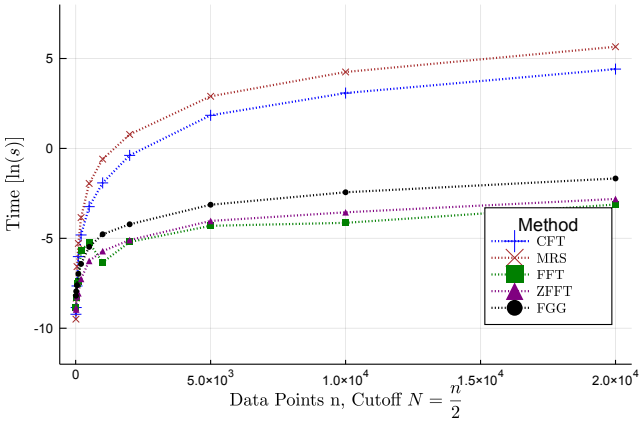
(a) Compute time and data-size (Dirichlet, 2-Assets)



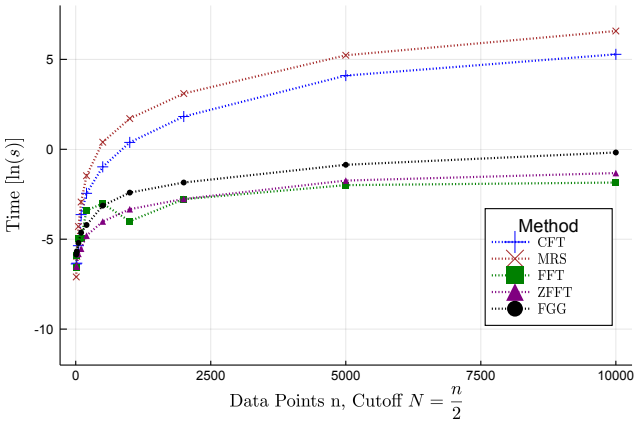
(b) Compute time and data-size (Fejér, 2-Assets)



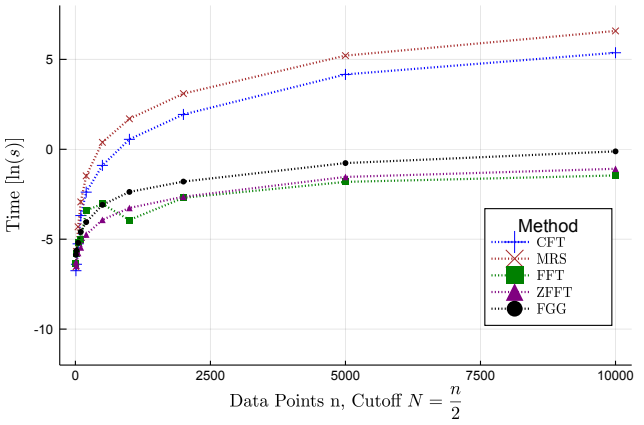
(c) Compute time and data-size (Dirichlet, 10-Assets)



(d) Compute time and data-size (Fejér, 10-Assets)



(e) Compute time and data-size (Dirichlet, 100-Assets)



(f) Compute time and data-size (Fejér, 100-Assets)

Figure 2.5: Here we investigate the algorithm complexity between traditional implementation methods against fast Fourier methods. The logarithm of compute time (measured in seconds) is plotted as a function of the number of data points n for the various methods. The first and second columns are the Dirichlet and Fejér representation respectively. The first to third rows are $D = 2, 10$ and 100 features respectively. The traditional methods investigated are: the vectorised implementation (CFT, $+$) and the 'for-loop" implementation (MRS, $\times$). The fast Fourier methods investigated are: the FFT (FFT, $\square$), the zero-padded FFT (ZFFT, $\triangle$), and the NUFFT using the fast Gaussian gridding (FGG, $\circ$) with the default $\varepsilon = 10^{-12}$. We see the efficacy of fast methods when it comes to reducing compute time. The figures can be recovered using the Julia script files Dirichlet Timing.jl and Fejer Timing.jl on the GitHub resource (Chang et al., 2020c).

Let us first investigate the common factors affecting the compute time of the algorithms. First, we investigate the computation time as a function of the number of data points for a synchronous GBM with $n_1 = ... = n_D = n$. I use the synchronous case because the Nyquist frequency in this case will be $N = \frac{n}{2}$ which scales linearly with the number of data points. Figure 2.5 compares traditional implementations against fast Fourier methods. The traditional implementations include: the for-loop implementation (MRS), the vectorised implementation (CFT). The fast Fourier methods include: the FFT (FFT), the zero-padded FFT (ZFFT), and the fast Gaussian gridding implementation of the NUFFT (FGG) with the default tolerance of $\varepsilon = 10^{-12}$. The complexity $\mathcal{O}(n)$ plots are plotted with the logarithm of compute time (measured in seconds) against the number of data points n. The compute time is the minimum estimate over 10 replications. This is because the minimum is a robust estimator for the location parameter of the time distribution (Chen and Revels, 2016). The first and second columns are the Dirichlet and Fejér representation respectively. The first to third rows are $D = 2, 10$ and 100 features respectively. Here the induced covariance matrix for $D = 10$ and 100 are created using a uniform random matrix and re-scaled appropriately (See Algorithms 20 and 21). The choice of a uniform random matrix is purely for convenience. Although it will only produce positive correlations, this will not influence the estimates of compute times.

There are several things to notice in Figure 2.5. First, the for-loop and vectorised implementation have the same general shape but the vectorised implementation is faster because it exploits the Hermitian symmetry. Notice however in Figures 2.5a and 2.5b the vectorised implementation becomes slower than the for-loop implementation when n becomes large. This is due to the memory constraints. The vectorised implementation requires a complex matrix of size $n \times N$ and each element in the matrix requires 16 bytes to store a complex 64-bit floating point number. Therefore, 20GB of memory is required for simply 5×10^4 data points and thus requiring the use of virtual memory resulting in the deterioration of performance.

Second, the difference in compute time between fast Fourier methods and the traditional implementations are significant. Table 2.1 I report the compute time (measured in seconds) for 2 features with $n = 10^5$ data points. Here the for-loop takes 1176 seconds while the fast Gaussian gridding takes 0.119 seconds, this is 10,000 times faster and scales even more as n increases. The logarithmic nature of Figure 2.5 underplays how significant this difference is.

Method	MRS	KB	ES	FGG	FINUFFT
Time (s)	1176s	2.161s	0.190s	0.119s	0.0331s

Table 2.1: The table reports the compute time measured in seconds for various algorithms using 2 features with 10^5 data points. The methods considered are: the "for-loop" implementation (MRS), the fast Gaussian gridding (FGG), Kaiser-Bessel (KB), exponential of semi-circle using our naive implementation (ES), and the implementation from FINUFFT (FINUFFT). The NUFFT methods are computed using the default $\varepsilon = 10^{-12}$. The times are extracted from Figures 2.5 and 2.6.

Third, between the fast Fourier methods we have: FFT, zero-padded FFT, and FGG from fastest to slowest. There are two reasons for this. First the number of Fourier modes computed are different. The FFT exploits the Hermitian symmetry so it only computes $N + 1$ Fourier modes, the zero-padded FFT computes M_N Fourier modes, and the FGG computes M_r Fourier modes. Second, the FFT requires no computation beforehand, the zero-padded FFT needs to zero-pad the missing data, and the FGG has the additional convolution and deconvolution step.

Lastly, the breadth of features D can impact the computation time depending on the choice of N. We can either pick N to be the same across the features, or a specific N for each pairwise entry. The case when N is the same across the features is simple. We only need to compute the M_N Fourier coefficients for the D features, this is the method I use in Figure 2.5. This presents two advantages: *i.)* the time-scale investigated will be the same for all the features, and *ii.)* if we use the Fejér basis kernel, we

can guarantee positive semi-definiteness in the covariance matrix (Mancino et al., 2017; Mancino and Sanfelici, 2011). The case when N changes for the different features becomes more nuanced. For example, the arrival time representation will have different Nyquist frequencies for each feature. Here when N changes, we need to compute $\frac{D(D-1)}{2}$ pairwise estimates for each $\hat{\Sigma}^{ij}_{n_i,n_j,N}$ entry. When this is the case, or when the Dirichlet basis kernel is used, we are not guaranteed a positive semi-definite matrix. This can present challenges. For example, when an invertible covariance matrix is required such as in the case for portfolio optimisation. Although this can be ameliorated by transforming the non-positive semi-definite matrix to the closest positive semi-definite matrix under some appropriate norm (Lindskog, 2001), or using extensions of these type of transformations (Matoti, 2009); doing so comes with an additional computational cost.

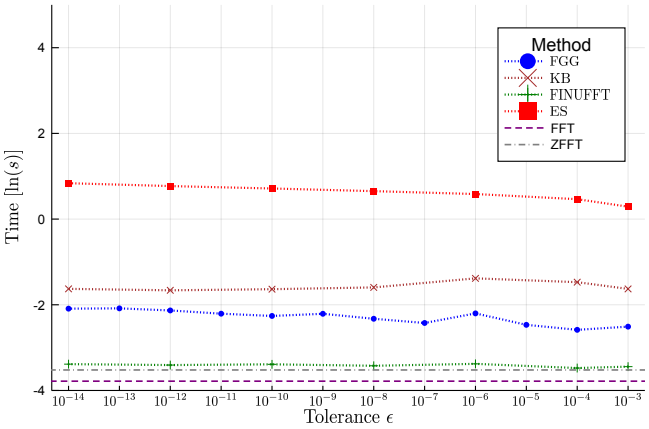
Let us investigate the degree of asynchrony as a variable of study. Specifically the impact on computation time under the arrival time representation. Here we only consider the case when $D = 2$. Let $1/\lambda_1$ be the mean inter-arrival time used to sample the first feature, and N_1 be the corresponding Nyquist frequency from the feature;[6] similarly $1/\lambda_2$ and N_2 for the second feature. Therefore, the N used in eqs. (2.3), (2.5) and (2.6) is $N = \min\{N_1, N_2\}$ to avoid aliasing.

				Dirichlet [sec]		Fejér [sec]	
n_1	n_2	N	$1/\lambda_2$	FGG	MRS	FGG	MRS
319	336	48855	30	0.047676	3.83602	0.050635	3.93206
319	272	36164	40	0.027140	2.51448	0.032651	2.64331
319	214	7128	50	0.004719	0.40825	0.005266	0.44622
319	169	38917	60	0.021548	2.19374	0.028445	2.40181
319	168	25006	70	0.013686	1.39089	0.016143	1.50772
319	106	2281	80	0.000995	0.09976	0.001036	0.10474
319	119	3325	90	0.001844	0.14592	0.001923	0.15903
319	98	2227	100	0.000765	0.09277	0.000805	0.09451

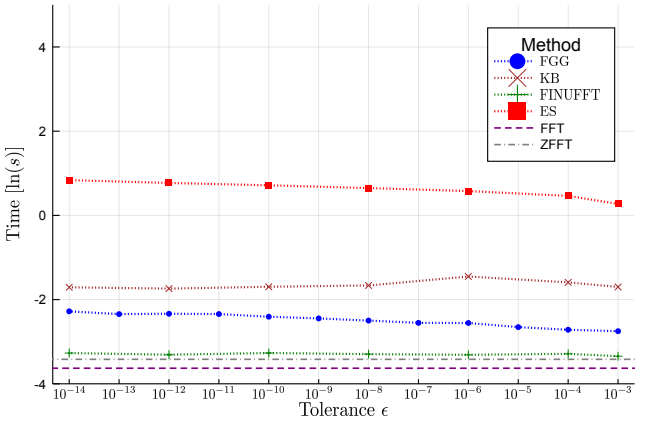
Table 2.2: The Dirichlet and Fejér computation times (measured in seconds) are given for varying degree of asynchrony. Here a synchronous grid with 10^4 data points is sampled with $1/\lambda_1 = 30$ for the first feature, while $1/\lambda_2$ ranges from 30 to 100 in increments of 10 seconds for the second feature. The table reports the exact n_i and N from the sampling. We see that the fast Gaussian gridding implementation of the NUFFT (FGG) outperforms the for-loop implementation (MRS) and as $1/\lambda_2$ increases, n_2 and N decrease, resulting in a faster compute time. However, increasing $1/\lambda_2$ does not guarantee that Δt_0 will be larger, which can lead to a larger N and therefore a longer compute time for some cases. The results can be recovered using the script file Asynchrony Timing.jl on the GitHub resource (Chang et al., 2020c).

Table 2.2 reports the Dirichlet and Fejér computation time (minimum estimate over 10 replications and measured in seconds) for the for-loop implementation (MRS) and the fast Gaussian gridding implementation of the NUFFT (FGG) using the default $\varepsilon = 10^{-12}$. We simulate a synchronous grid with $n = 10^4$ data points. This process is then sampled using an exponential inter-arrival with rate λ_i for the i^{th} feature. The first feature is sampled with an average inter-arrival $1/\lambda_1$ of 30 seconds, while the second feature is sampled with an average inter-arrival $1/\lambda_2$ ranging from 30 to 100 seconds in increments of 10. The exact number of observed data points $n_i \approx n/\lambda_i$ and Nyquist frequency N from the sampling is also reported. The table demonstrates two things: first, the FGG is significantly faster than the for-loop implementation. Second, as $1/\lambda_2$ increases we get faster compute times (in general) because n_2 and N decrease (due to larger inter-arrivals). This however, is not guaranteed because a larger $1/\lambda_2$ does not ensure Δt_0 (minimum Δt) for the second feature to also be larger. Thus, N does not always decrease which can therefore lead to longer compute times.

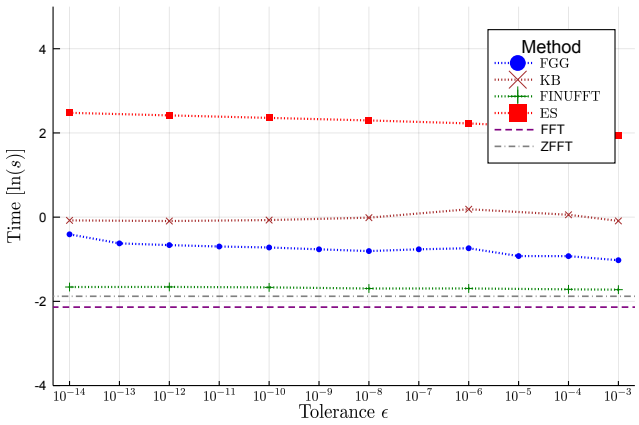
[6] The Nyquist frequency here is $\lfloor \frac{T}{2\Delta t_0} \rfloor$ where Δt_0 is the smallest distance between two consecutive prices (Mancino et al., 2017).



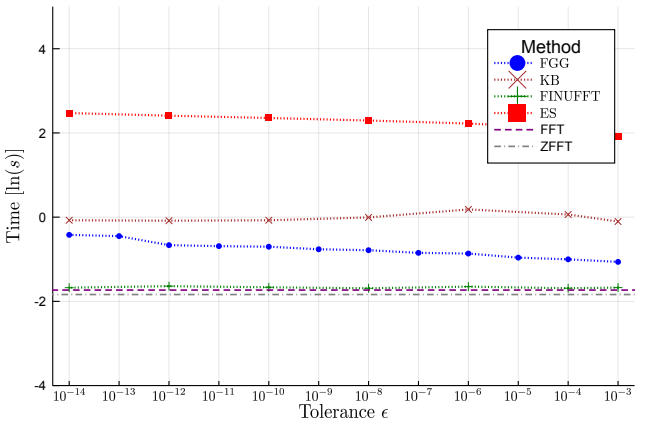
(a) Compute time and tolerance (Dirichlet, 2-Assets, $n = 10^5$)



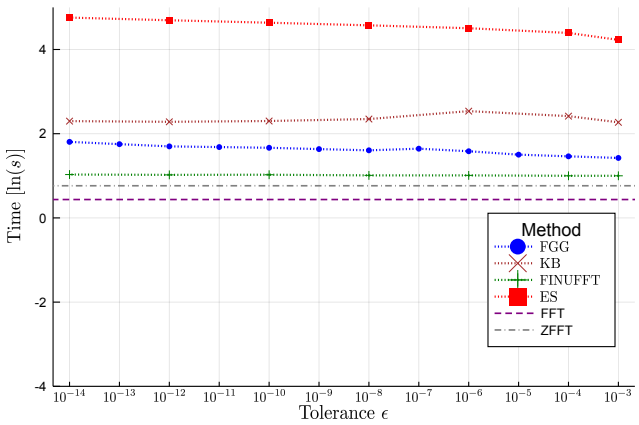
(b) Compute time and tolerance (Fejér, 2-Assets, $n = 10^5$)



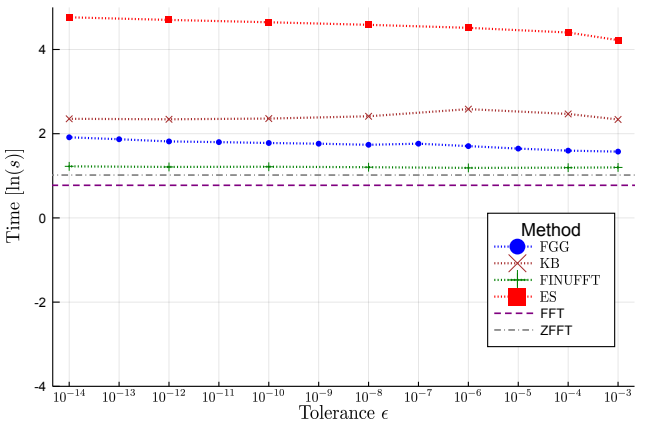
(c) Compute time and tolerance (Dirichlet, 10-Assets, $n = 10^5$)



(d) Compute time and tolerance (Fejér, 10-Assets, $n = 10^5$)



(e) Compute time and tolerance (Dirichlet, 100-Assets, $n = 10^5$)



(f) Compute time and tolerance (Fejér, 100-Assets, $n = 10^5$)

Figure 2.6: Here we investigate the algorithm complexity for the fast Fourier methods. The logarithm of compute time (measured in seconds) is plotted as a function of the error tolerance ε. The FFT (FFT, dashed lines) and the zero-padded FFT (ZFFT, dash dots) are plotted as a baseline for comparison. The first and second columns are the Dirichlet and Fejér representation respectively. The first to third rows are $D = 2, 10$ and 100 features respectively, each with $n = 10^5$ data points. The NUFFT methods compared include: the Gaussian kernel (FGG, $\circ$), the Kaiser-Bessel kernel (KB, $\times$), the exponential of semi-circle kernel with the naive implementation (ES, $\square$), and the implementation by Barnett et al. (2018) (FINUFFT, $+$). The results are consistent with the results from Barnett et al. (2018), where the FINUFFT implementation is faster than the fast Gaussian gridding which is faster than the Kaiser-Bessel. The figures can be recovered using the Julia script file Error Timing.jl on the GitHub resource (Chang et al., 2020c).

The last factor influencing compute time is the tolerance ε which is specific to NUFFT methods. I explore the algorithmic complexity here as a function of ε rather than n. Here I use a synchronous grid with $n = 10^5$ data points. Recall that the complexity of NUFFTs are given as: $\mathcal{O}(M_r \log M_r + n|\log(\varepsilon)|)$. It depends on ε because the tolerance determines the spreading width which controls the number grid points a single observations needs to be interpolated to as part of the first step in the algorithm.

Figure 2.6 compares the following algorithms. As a baseline for comparison, we have: the FFT (FFT) and the zero-padded FFT (ZFFT). The NUFFT methods include: the fast Gaussian gridding with the Gaussian kernel (FGG), the Kaiser-Bessel kernel (KB), the exponential of semi-circle using the naive implementation (ES), and the FINUFFT package by Barnett et al. (2018) (FINUFFT). The plot is the logarithm of compute time (minimum estimate over 10 replications and measured in seconds) as a function of ε. The first and second columns are the Dirichlet and Fejér representation respectively, while the first to third row is $D = 2, 10$ and 100 features respectively. Each feature has $n = 10^5$ data points.

The results in Figure 2.6 are consistent with that of Barnett et al. (2018). Between the NUFFT methods, we have: the FINUFFT implementation using the exponential of semi-circle, the fast Gaussian gridding, the Kaiser-Bessel kernel evaluated "on-the-fly",[7] and the exponential of semi-circle using the naive implementation listed in order from fastest to slowest. Finally, the FFT and zero-padded FFT outperform all NUFFT methods as expected.

To unpack these results, let us take a closer look at the inner workings behind each algorithm. Besides the differences in the number of Fourier modes computed in the FFT, ZFFT and NUFFT methods as previously discussed; there is also a difference in the computations before performing the FFT. The FFT does not need to assign any data points nor compute any quantities before performing the FFT. The ZFFT needs to assign n data points to the over-sampled grid but does not require any evaluations before performing the FFT. The NUFFT methods need to assign ωn points to the over-sampled grid and further requires ωn evaluations of $f_j \varphi(\cdot)$ before performing the FFT. This explains the difference in speed between the FFT, ZFFT and NUFFT methods.

Now the differences between the NUFFT methods is how they reduce the number of evaluations required. The fast Gaussian gridding reduces the number of exponential evaluations for $\varphi_G(\cdot)$ by separating the exponential into three components:

$$e^{-(x_j - 2\pi m/M_r)^2/4\tau} = \left[e^{-x_j^2/4\tau}\right] \left[e^{x_j \pi/M_r \tau}\right]^m \left[e^{-(\pi m/M_r)^2/\tau}\right]. \tag{2.30}$$

Now instead of ω exponential evaluations for each source point, we only need two exponential evaluations per source point and pre-compute $e^{-(\pi m/M_r)^2/\tau}$. This reduces the number of exponential evaluations from ωn to $\omega + 2n$.

The Kaiser-Bessel kernel exploits the fact that it is both smooth and has narrow support (Barnett et al., 2018). Therefore, it is able to reduce the number of kernel evaluations by reducing ω while achieving a comparable level of accuracy. For example, if we want roughly 12 digit accuracy, the Gaussian kernel requires $\omega = 24$ while the Kaiser-Bessel only needs $\omega = 13$ to achieve the same accuracy (Barnett et al., 2018).

Finally, the exponential of semi-circle has narrow support similar to the Kaiser-Bessel kernel but is simpler and faster to evaluate. The downfall is that there is no known analytic Fourier transform, therefore it incurs an additional cost of having to numerically evaluate the integral in eq. (2.25). My implementation of the exponential of semi-circle is naive compared to the implementation by Barnett et al. (2018). I do not use the two additional implementation techniques to improve the compute time. Specifically, I do not exploit the piecewise polynomial kernel approximation to accelerate the evaluation of eq. (2.24); and

[7] "on-the-fly" refers to the fact that I do not pre-compute the spreading quantities, preventing large RAM overhead.

I use a numerical integration package QuadGK to compute eq. (2.25) which uses the adaptive Gauss-Kronrod quadrature rather than the Gauss-Legendre quadrature with "phase winding" technique used by Barnett et al. (2018). This naive implementation of the exponential of semi-circle allows the like-for-like comparison between the various kernel based on my implementation; and it illustrates the importance of the implementation techniques used in Barnett et al. (2018). In Figure 2.6, without the appropriate implementation techniques, the exponential of semi-circle is significantly slower than the other kernels. However, with the appropriate implementation, Barnett et al. (2018) are able to reduce the compute time of the exponential of semi-circle to a similar time as the zero-padded FFT.

It must be highlighted that there is room for further optimisation on the compute times I have presented, specifically for the case of many features where the computation of the M_N Fourier modes for each feature can be computed in parallel.

2.3.2 Benchmark Accuracy

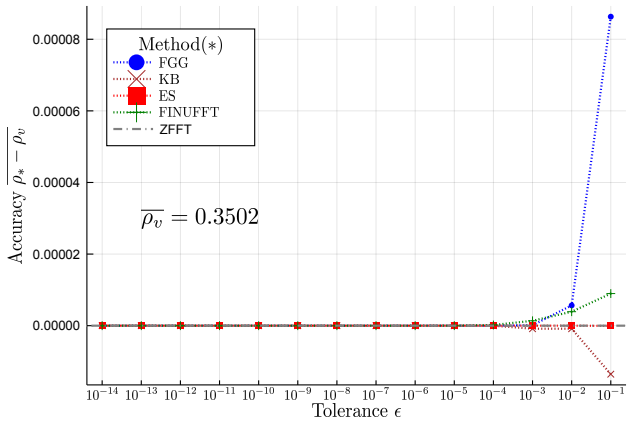
It is clear that fast Fourier techniques can significantly improve the computation speed, but we need understand under what conditions they fail to ensure the techniques are used correctly. Moreover, with NUFFT methods, we need to know what level of numerical accuracy is required to ensure that the Fourier coefficients evaluated using NUFFT techniques in eq. (2.16) can recover the same estimates of eqs. (2.5) and (2.6) using the direct evaluation of eq. (2.4). This is done by testing fast Fourier methods on the synchronous case, the missing data representation, and the arrival time representation. Next we need to ensure that the NUFFT methods can recover the correct estimates for various values of N. In other words, we need to ensure that there is no adverse interplay between the kernel averaging (the convolution step in the NUFFT methods) and the time-scale averaging (the choice of N and its implied time-scale). Lastly, I perform a bias-MSE analysis and sensitivity analysis to ensure that the NUFFT methods indeed correctly recover the target integrated volatility/co-volatility.

Integrated correlation

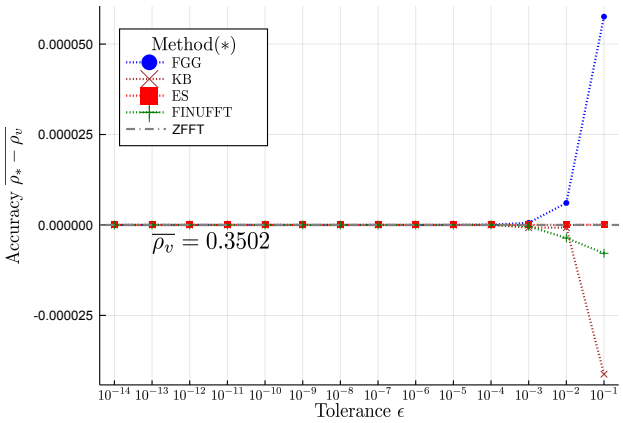
In the following two experiments, I simulate a bivariate Geometric Brownian Motion $n = 10^4$ synchronous data points. The daily parameters are: $\mu_1 = 0.01$, $\mu_2 = 0.01$, $\sigma_1^2 = 0.1$, $\sigma_2^2 = 0.2$, $\rho^{12} = 0.35$. Here the discretisation interval is $\Delta t = \frac{1}{86400}$. Each unit interval can be thought of as a second in a 24 hour trading day. From the synchronous case, the missing data representation is obtained by randomly removing 40% of the observations from each price path; and the arrival time representation is achieved by sampling the first price path with an exponential inter-arrival with mean 30 seconds and the second price path with a mean of 45 seconds. In these two experiments I measure accuracy as the difference between the estimates from the fast Fourier methods and the estimates from the vectorised implementation averaged over 100 replications. This is because we are interested whether fast Fourier methods allow us to achieve the same estimates of eqs. (2.5) and (2.6) using the most direct evaluation of eq. (2.4).

The first experiment in Figure 2.7 investigates the accuracy of the following fast Fourier methods: the fast Gaussian gridding (FGG), the Kaiser-Bessel kernel (KB), the exponential of semi-circle with the naive implementation (ES), and the FINUFFT implementation (FINUFFT), and the zero-padded FFT (ZFFT). The three simulation scenarios are: the synchronous case (first row), the missing data representation (second row), and the arrival time representation (third row). The first and second column are the Dirichlet and Fejér representation respectively. Note that the Nyquist frequency (Nyq.) is used for all the estimates here.

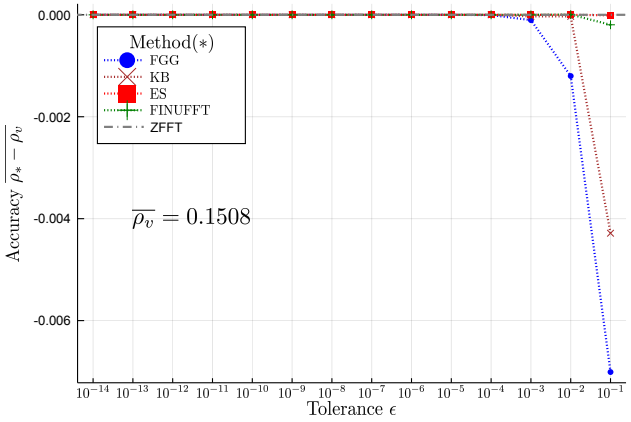
There are several things to notice in Figure 2.7. First, all the NUFFT methods can accurately recover the vectorised implementation with the tolerance level $\varepsilon < 10^{-4}$. Moreover, we see that the ES kernel can recover the correct estimates for $\varepsilon = 10^{-1}$. This is not a property of the ES kernel, rather it was due to



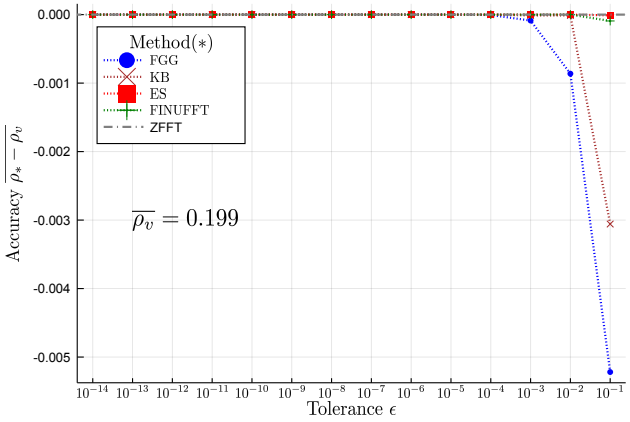
(a) Accuracy and tolerance (Dirichlet, Synchronous)



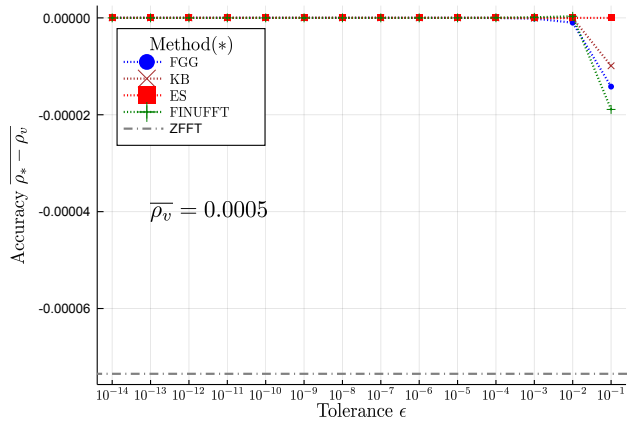
(b) Accuracy and tolerance (Fejér, Synchronous)



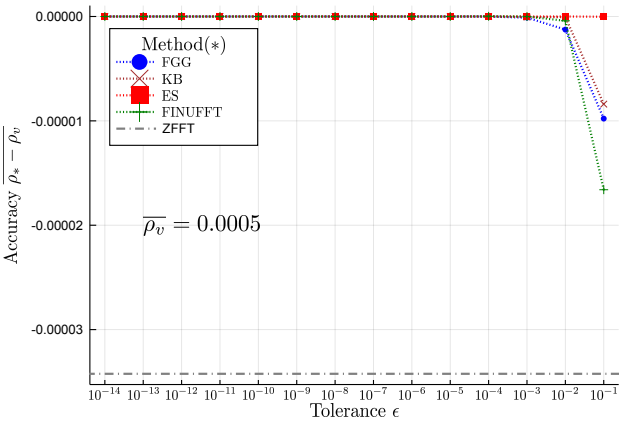
(c) Accuracy and tolerance (Dirichlet, Down-sampled 40%)



(d) Accuracy and tolerance (Fejér, Down-sampled 40%)



(e) Accuracy and tolerance (Dirichlet, Arrival time, Nyquist)



(f) Accuracy and tolerance (Fejér, Arrival time, Nyquist)

Figure 2.7: Here we investigate the accuracy of various fast Fourier methods as a function of the tolerance ε under various simulation conditions. Accuracy is measured as the difference between the estimates of the various fast Fourier methods ρ_* and the estimate using the vectorised implementation (CFT) ρ_v averaged over the 100 replications. The average correlation estimate from the vectorised implementation is provided as an inset in each figure. The baseline process is a synchronous GBM with 10^4 data points. The simulation settings are: the synchronous case (first row), the missing data representation (second row), and the arrival time representation (third row). The fast Fourier methods investigated are: the fast Gaussian gridding (FGG, $\circ$), the Kaiser-Bessel kernel (KB, $\times$), the exponential of semi-circle with the naive implementation (ES, $\square$) and the FINUFFT implementation (FINUFFT, $+$), and the zero-padded FFT (ZFFT, dash dots). The first and second column are the Dirichlet and Fejér representation respectively. The Nyquist frequency (Nyq.) is used for all the estimates here, therefore note that N changes for each replication under the arrival time representation. We see that as long as $\varepsilon < 10^{-4}$, the NUFFT methods can recover the estimates using the vectorised implementation. More importantly, the zero-padded FFT fails for the arrival time representation. The figures can be recovered using the Julia script file AccSynDS.jl and AccRE.jl on the GitHub resource (Chang et al., 2020c).

my choice of M_{sp} for the ES implementation to ensure that the requested tolerance is always strictly met. We see that the FINUFFT implementation diverges from $\varepsilon = 10^{-4}$ due to their more lenient choice of ω. This means each source point must be spread in each direction for a minimum of $M_{sp} = 4$ grid points for the Gaussian kernel, $M_{sp} = 3$ grid points for the Kaiser-Bessel kernel and $M_{sp} = 3$ grid points for the exponential of semi-circle to recover the vectorised estimate.[8] Second, when the tolerance level is insufficient, the divergence away for NUFFT methods seems to be an artefact from the lack of precision requested in $\mathbf{F}(dX^i)$ as there is no clear pattern in the divergence. Finally, the zero-padded FFT recovers the correct estimates for the synchronous and missing data representation, but it fails for the arrival time representation. The reason it fails is due to the shifting of time points without accounting for this through an appropriate convolution and deconvolution which therefore results in the incorrect estimates. The failure of the ZFFT for the truly asynchronous case is the main motivation behind why we need non-uniform FFT methods. NUFFT methods overcome the issue of smearing time points through a convolution and deconvolution step to correct the effects of shifting the points to a uniform grid by preserving the power spectrum.

In the next experiment, we must understand if there is any adverse relationship between the kernel averaging and the time-scale averaging. This is to ensure we can accurately use NUFFT methods to investigate the various time-scales. Here we investigate this using the arrival time representation (using the same parameters as before) but for three choices of N.

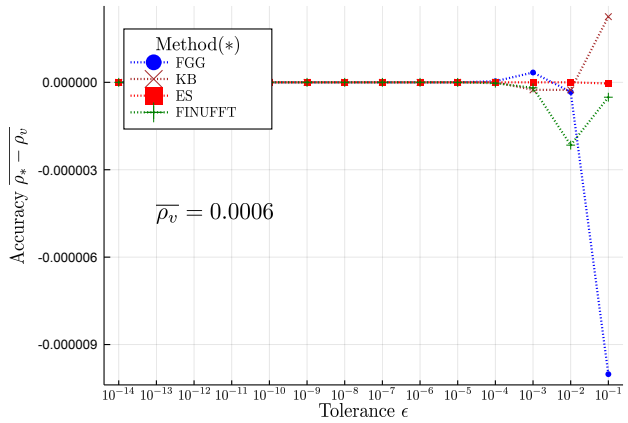
Previously in Figure 2.7, because the Nyquist frequency was used, N was changing for each replication under the arrival time representation. Figure 2.8 I fix three cases of N for the various replications. The first N is computed as the minimum Nyquist frequency across the 100 replications resulting in $N = 18,021$. The second N is computed based on the smallest *average* sampling interval resulting in $N = \lfloor \frac{10,000}{30\times 2} \rfloor = 166$. Finally, the last N is chosen to be arbitrarily small subject to the condition that the corresponding M_r is larger than ω for $\varepsilon = 10^{-14}$. This is to ensure the up-sampled grid is larger than the total spreading width. I pick $N = 15$ for the final case. The zero-padded FFT is excluded because the implementation only computes the case when N is the Nyquist frequency.

Figure 2.8 we see that there is no clear relationship between the two types of averaging. We can recover the vectorised estimates for any choice of N provided the tolerance $\varepsilon < 10^{-4}$. Once again, the divergence for the various kernels seem to be from the lack of precision requested in $\mathbf{F}(dX^i)$ as there is no clear pattern. Lastly, the average correlation estimates from the vectorised implementation are provided as insets. The results are consistent with the prior results of Renò (2003) and my Honours project (Chang et al., 2019a). As the sampling frequency increases (N increases), the sampling interval decreases which results in the Epps effect arising from asynchrony.

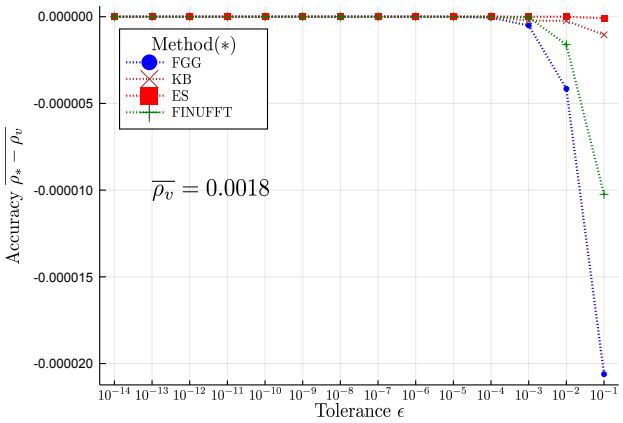
Integrated volatility and co-volatility

Although I have shown that the estimates correctly recover the integrated correlation (relative to the vectorised implementation), it is not exactly clear that the methods indeed recover the correct target which is the integrated volatility and co-volatility in eqs. (2.5) and (2.6). Therefore, I further perform a bias-MSE and sensitivity analysis for the peace of mind.

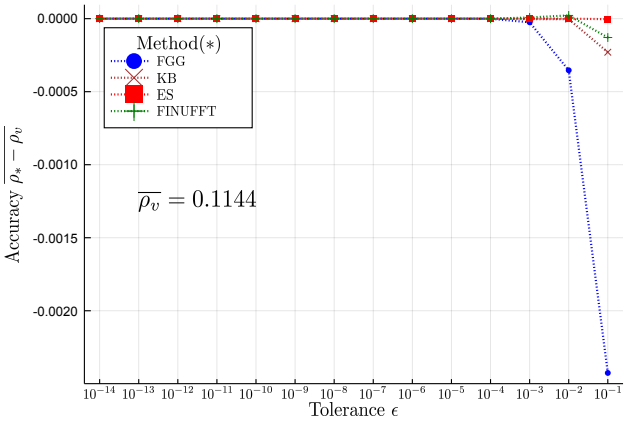
[8] The M_{sp} requirement is calculated based on $\varepsilon = 10^{-4}$ for the Gaussian and Kaiser-Bessel kernel and $\varepsilon = 10^{-1}$ for the ES kernel.



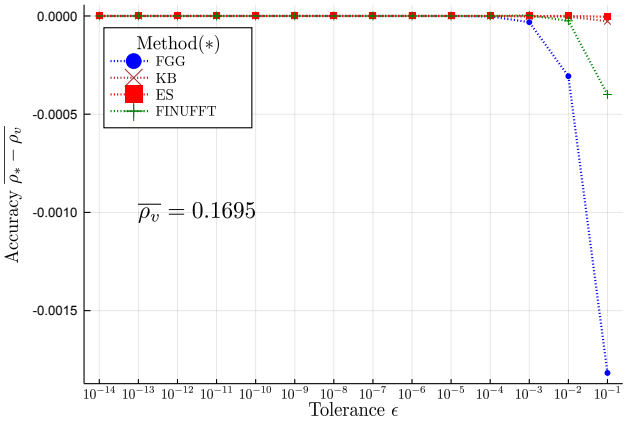
(a) Accuracy and tolerance (Dirichlet, Arrival time, $N = 18021$)



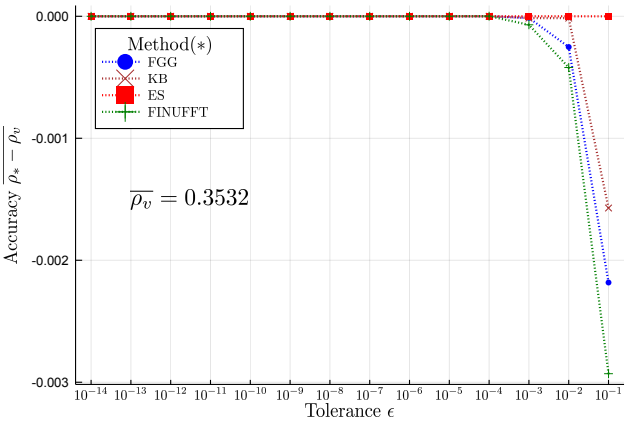
(b) Accuracy and tolerance (Fejér, Arrival time, $N = 18021$)



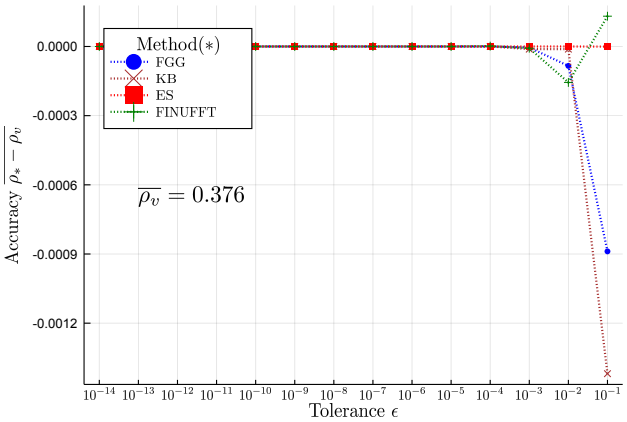
(c) Accuracy and tolerance (Dirichlet, Arrival time, $N = 166$)



(d) Accuracy and tolerance (Fejér, Arrival time, $N = 166$)

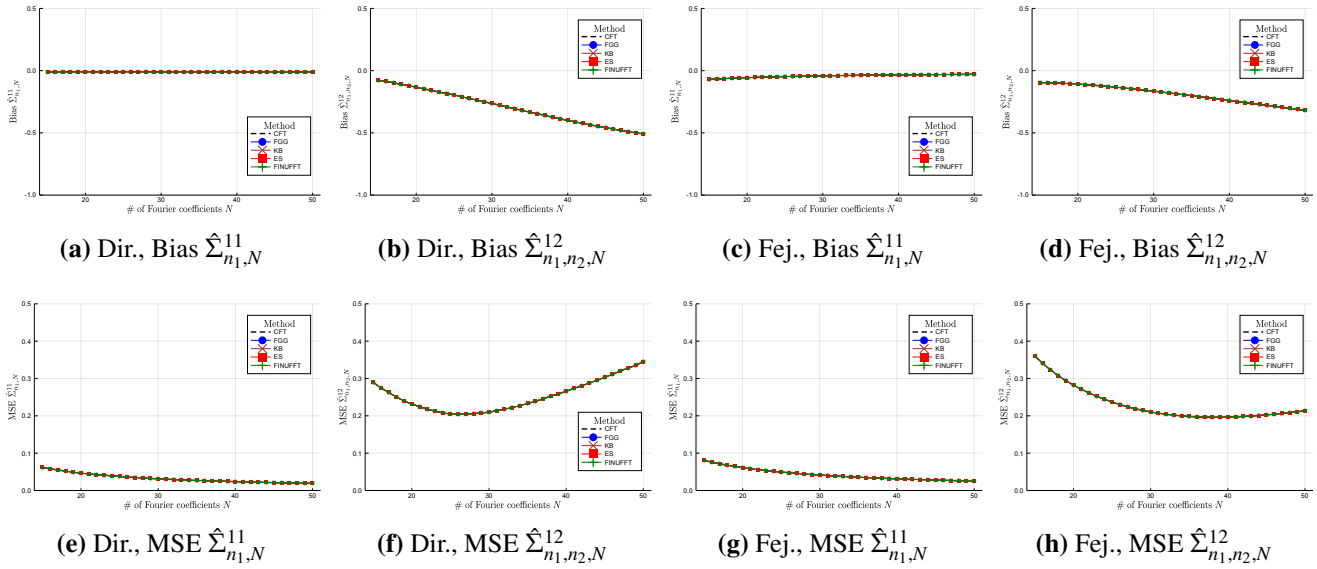


(e) Accuracy and tolerance (Dirichlet, Arrival time, $N = 15$)



(f) Accuracy and tolerance (Fejér, Arrival time, $N = 15$)

Figure 2.8: Here we investigate the inter-play between kernel averaging and time-scale averaging. The figures plot the accuracy of various fast Fourier methods as a function of the tolerance ε for three choices of N under the arrival time representation. Accuracy is measured as the difference between the estimates of the various fast Fourier methods ρ_* and the estimate using the vectorised implementation (CFT) ρ_v averaged over the 100 replications. The average correlation estimate from the vectorised implementation is provided as an inset in each figure. The fast Fourier methods investigated are: the fast Gaussian gridding (FGG, $\circ$), the Kaiser-Bessel kernel (KB, $\times$), the exponential of semi-circle with the naive implementation (ES, $\square$), and the FINUFFT implementation (FINUFFT, $+$). The first and second columns are the Dirichlet and Fejér representation respectively. We see that there is no clear relation between the two types of averaging. The divergence from higher tolerance levels seems to be an artefact of errors arising from the lack of precision requested. The figures can be recovered using the Julia script file AccRE.jl on the GitHub resource (Chang et al., 2020c).



(a) Dir., Bias $\hat{\Sigma}^{11}_{n_1,N}$ **(b)** Dir., Bias $\hat{\Sigma}^{12}_{n_1,n_2,N}$ **(c)** Fej., Bias $\hat{\Sigma}^{11}_{n_1,N}$ **(d)** Fej., Bias $\hat{\Sigma}^{12}_{n_1,n_2,N}$

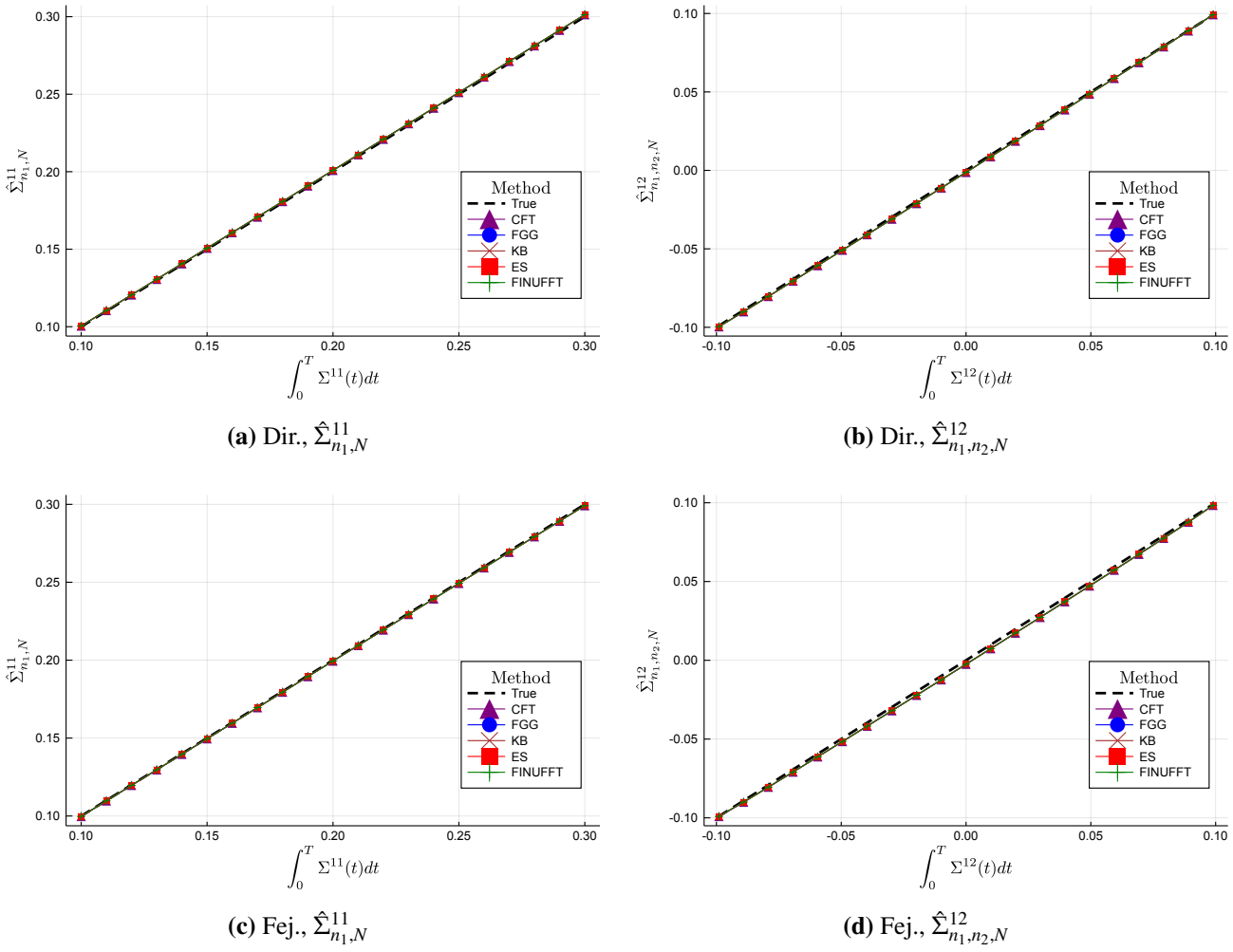
(e) Dir., MSE $\hat{\Sigma}^{11}_{n_1,N}$ **(f)** Dir., MSE $\hat{\Sigma}^{12}_{n_1,n_2,N}$ **(g)** Fej., MSE $\hat{\Sigma}^{11}_{n_1,N}$ **(h)** Fej., MSE $\hat{\Sigma}^{12}_{n_1,n_2,N}$

Figure 2.9: Here we investigate the bias (first row) and MSE (second row) of the integrated covariance as a function of the number of Fourier coefficients using the Dirichlet (first and second columns) and Fejér (third and fourth columns) representation. The results report 10,000 replications. The base-line price process is a synchronous GBM with $n = 100$ data points. The missing data representation is induced here using the regular non-synchronous trading, where the second asset is observed at every second trade of asset one. The methods investigated are: the vectorised implementation (CFT, black dashes), the fast Gaussian gridding (FGG, $\circ$), the Kaiser-Bessel kernel (KB, $\times$), the exponential of semi-circle with our naive implementation (ES, $\square$), and the FINUFFT implementation (FINUFFT, $+$). The NUFFT methods are computed using the default $\varepsilon = 10^{-12}$. The fast Fourier methods recover the same bias and MSE results as the vectorised implementation and are consistent with the results of Mancino et al. (2017). The figures can be recovered using the Julia script file MSEBias.jl on the GitHub resource Chang et al. (2020c).

To this end, the bias and MSE analysis in Figure 2.9 compares the integrated covariance as a function of the number of Fourier coefficients for the vectorised implementation and the NUFFT methods with a default of $\varepsilon = 10^{-12}$ for 10,000 replications. Here a synchronous bivariate GBM with $n = 100$ is simulated with the same parameters as before, with the exception that $\Delta t = 1/n$. Here I introduce a special case of the missing data representation known as the regular non-synchronous trading (Reg-NS) used by Mancino et al. (2017). With Reg-NS, the second asset is observed at every second trade of the first asset (both processes are observed on X^i_0 and $X^i_{2\pi}$, $i = 1,2$). The methods investigated are: the vectorised implementation (CFT), the fast Gaussian gridding (FGG), the Kaiser-Bessel kernel (KB), the exponential of semi-circle with our naive implementation (ES), and the FINUFFT implementation (FINUFFT). Figure 2.9 the first row is the bias, the second row is the MSE; the first and second column is the Dirichlet representation, and the third and fourth column is the Fejér representation. We see that the NUFFT methods recover the same bias ans MSE results as the vectorised implementation. The results are consistent with that of Mancino et al. (2017). Under asynchronous observations, the integrated volatility $\hat{\Sigma}^{11}_{n_1,N}$ does not present a bias for all values of N, but the MSE is large for small values of N due to the increased variability in the estimator (Mancino et al., 2017). The situation is different for the co-volatility $\hat{\Sigma}^{12}_{n_1,n_2,N}$. Here we have an increase in bias for larger N (smaller time-scales) as a result of the Epps effect. This of course also affects the MSE. Renò (2003) has shown that the simplest method to correct for the effect is to pick a smaller N. In other words, we simply investigate a larger time-scale.

Mancino et al. (2017) suggest picking N by minimizing the MSE for an optimal bias and variance tradeoff. There has been work in the literature regarding the speed of convergence with respect to the degree of asynchrony with the MSE criterion in mind using the Malliavin-Mancino estimator (Chen, 2019; Mancino et al., 2017; Park et al., 2016). My concern with picking N to minimise the MSE is two fold. First, this implicitly assumes that we have a latent model and we should be concerned with

the deviation away from the true correlation coefficient which does not depend on Δt; but if the true correlation depends on Δt then the method becomes problematic. This is because picking N to minimise MSE can result in us averaging away important sources of the Epps effect. Second, we do not know the true correlation level when using empirical data. Mancino et al. (2017) overcome this by obtaining estimates from a larger time-scale where microstructure effects are minimal, they use daily estimates. Here my concern is that the two data generating processes are different, and to the best of my knowledge, it remains unclear if there is a seamless mapping between the two generative processes. Even if we do not use daily estimates, but rather minimising MSE with respect to the estimates using say 5-minute intervals. Doing so is simply trying to match the estimates to an estimate at a particular time-scale, which again can average away important sources of the Epps effect. I argue that the better approach is to estimate the correlation at a particular time-scale and then disentangling the known statistical effects (I will demonstrate this in Chapter 4).



(a) Dir., $\hat{\Sigma}^{11}_{n_1,N}$ **(b)** Dir., $\hat{\Sigma}^{12}_{n_1,n_2,N}$

(c) Fej., $\hat{\Sigma}^{11}_{n_1,N}$ **(d)** Fej., $\hat{\Sigma}^{12}_{n_1,n_2,N}$

Figure 2.10: Here we investigate the integrated volatility/co-volatility estimates as a function of various values of the true simulation process. This is achieved by simulating a synchronous bivariate GBM with $n = 10^4$ data points with the true value of $\int_0^T \Sigma^{11}(t)dt$ ranging from 0.1 to 0.3 and the true value of $\int_0^T \Sigma^{12}(t)dt$ ranging from -0.1 to 0.1. The methods investigated are: the vectorised implementation (CFT, purple line), the fast Gaussian gridding (FGG, blue dash-dot-dot), the Kaiser-Bessel kernel (KB, orange dashes), the exponential of semi-circle with our naive implementation (ES, red dash-dots), and the FINUFFT implementation (FINUFFT, green dots). The NUFFT methods are computed using the default $\varepsilon = 10^{-12}$. The estimates recover a linear relationship between the estimated integrated covariance and the true integrated covariance, confirming that the NUFFT methods can correctly recover the target estimates. The figures can be recovered using the Julia script file Sensitivity.jl on the GitHub resource (Chang et al., 2020c).

Finally, to ensure that the results are not just achieved for the specific parameter choices from before, I perform a sensitivity analysis to ensure that the NUFFT implementations can correctly recover the target integrated covariance. Here a synchronous bivariate GBM is simulated with $n = n_1 = n_2 = 10^4$ data points with discretisation interval $\Delta t = 1/n$. The first and second row of Figure 2.10 is the Dirichlet and Fejér representation respectively. For the integrated variance $\int_0^T \Sigma^{11}(t)dt$ the true value ranges from 0.1 to 0.3, and the true value ranges from -0.1 to 0.1 for the integrated covariance $\int_0^T \Sigma^{12}(t)dt$.

Figure 2.10 plots the estimated integrated estimates as a function of the true values. The methods investigated are: the vectorised implementation (CFT), the fast Gaussian gridding (FGG), the Kaiser-Bessel kernel (KB), the exponential of semi-circle with the naive implementation (ES), and the FINUFFT implementation (FINUFFT). The NUFFT methods use the default tolerance level $\varepsilon = 10^{-12}$. We see that the plots recover a linear relationship between the estimates and the true value. This confirms that the NUFFT methods can correctly recover the target estimates regardless of the simulation parameter. Moreover, the estimates from the various methods are exactly the same which is demonstrated in Figures 2.7 and 2.8 but confirmed here for the integrated volatility and co-volatility.

The key take-away from this section is that the NUFFT methods can be used for all cases of synchronous and asynchronous observations provided we request a low enough tolerance so that the Fourier coefficients $\mathbf{F}(dX^i)$ can correctly recover the estimates of eqs. (2.5) and (2.6).

2.4 Correlations and time-scale averaging

2.4.1 Simulated data

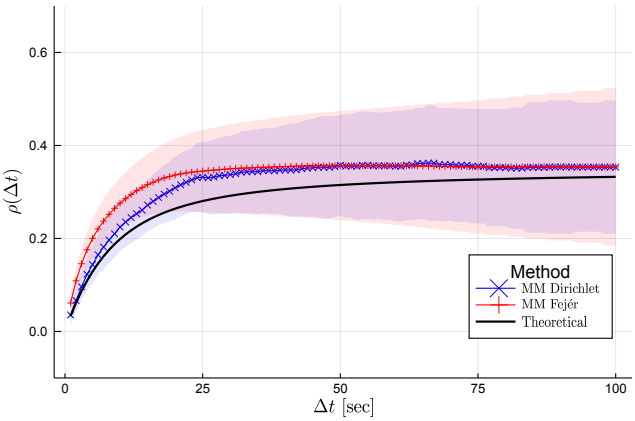
Let us very clearly demonstrate the link between N and the implied time-scale. Here let us compare the estimates to the theoretical Epps effect arising from asynchrony at a particular time-scale. Concretely, we can investigate different time-scales using the Malliavin-Mancino estimator through the choice of the cutting frequency N. Specifically by using the relation $\Delta t = \frac{T}{M_N} = \frac{T}{2N+1}$ as discussed in Section 2.2.2. This follows the insights introduced by Renò (2003) and Precup and Iori (2007) to investigate the Epps effect. Precup and Iori (2007) demonstrated that higher levels of asynchrony resulted in a larger drop in correlation. This is recovered in Figure 2.7 with the average correlation provided as insets for varying levels of asynchrony. Renò (2003) was able to demonstrate the Epps effect as a function of sampling frequency under the arrival time representation of asynchrony. This is recovered in Figure 2.8 with the average correlation provided as insets for different choices of N. Here I want to further solidify the idea.

Following the work of Renò (2003) and Precup and Iori (2007), Tóth and Kertész (2007) and Mastromatteo et al. (2011) were able to analytically derive the Epps effect arising from the arrival time representation as:

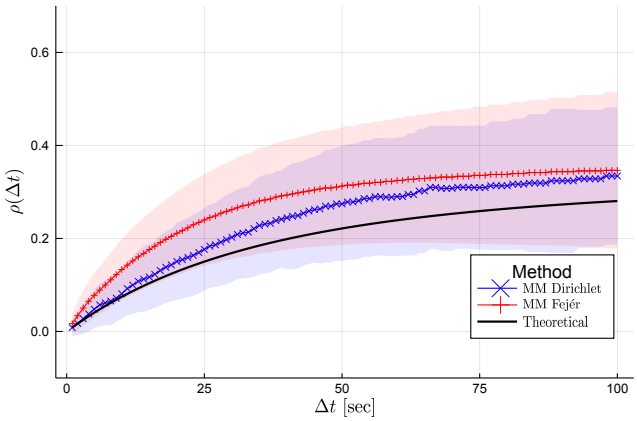
$$\tilde{\rho}^{ij}_{\Delta t} = c\left(1 + \frac{1}{\lambda \Delta t}\left(e^{-\lambda \Delta t} - 1\right)\right). \tag{2.31}$$

Here c is the induced correlation and the sampling rate is λ and is the same for the price paths. This will serve as the baseline theoretical Epps effect. I will derive eq. (2.31) later in Chapter 4.

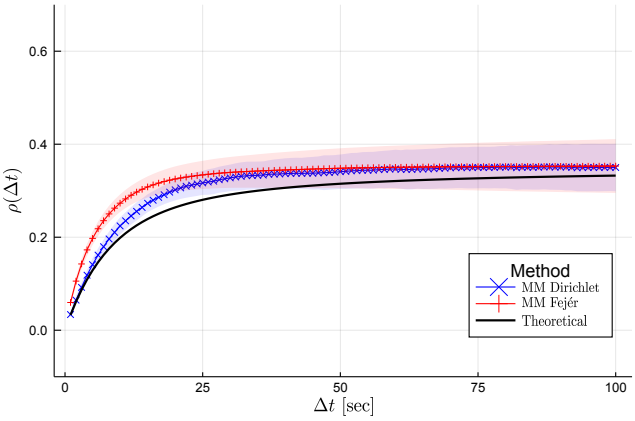
Now to compare the Malliavin-Mancino estimates to eq. (2.31), I simulate T data points from a synchronous bivariate GBM with the same parameters in Section 2.3.2. Let us consider one hour, one trading day, and one trading weeks' worth of simulated data. Assuming that each trading day is 8 hours in Calendar time, this translates to $T = 3600, 28800$ and 144000 synchronous data points for the various cases. This corresponds to the first to third row in Figure 2.11 respectively. The synchronous price paths are then each sampled using an exponential inter-arrival with the same rate λ to create the arrival time representation of the asynchronous price paths. Here I use two choices of λ: $\lambda = 1/5$ and $\lambda = 1/20$. This corresponds to the first and second column of Figure 2.11 respectively.



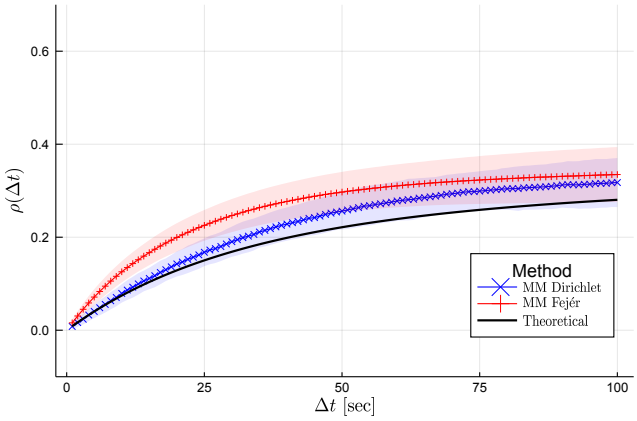
(a) Correlation and sampling interval ($\lambda = 1/5$, $T = 3600$)



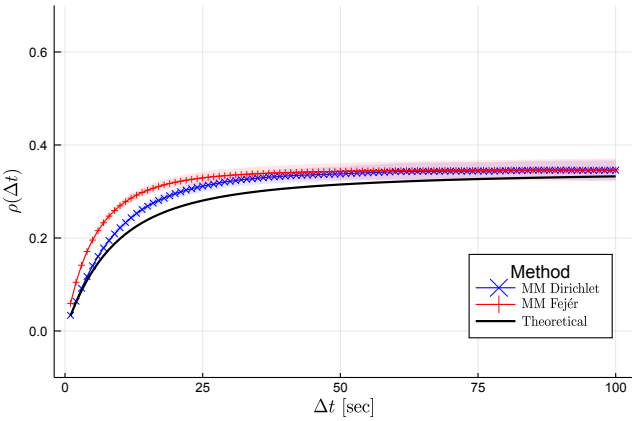
(b) Correlation and sampling interval ($\lambda = 1/20$, $T = 3600$)



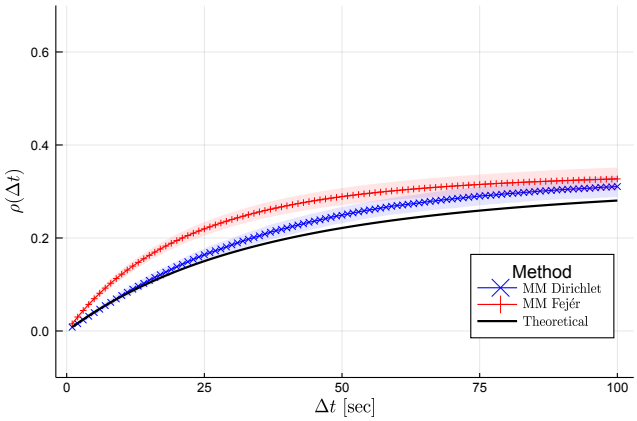
(c) Correlation and sampling interval ($\lambda = 1/5$, $T = 28800$)



(d) Correlation and sampling interval ($\lambda = 1/20$, $T = 28800$)



(e) Correlation and sampling interval ($\lambda = 1/5$, $T = 144000$)



(f) Correlation and sampling interval ($\lambda = 1/20$, $T = 144000$)

Figure 2.11: Here we investigate the link between time-scale averaging in the Malliavin-Mancino estimator compared to the theoretical Epps effect in eq. (2.31). The arrival time representation samples T synchronous data points (each points representing a second in a day) with an inter-arrival time with rates $\lambda = 1/5$ (first column) and $\lambda = 1/20$ (second column). The induced correlation across the replications is 0.35. Here the thick black line is the theoretical Epps effect in eq. (2.31). The Malliavin-Mancino estimates for each Δt are obtained using the Dirichlet (blue $\times$ with label "MM Dirichlet") and the Fejér (red $+$ with label "MM Fejér") representation. The process is repeated 100 times to obtain 100 estimates at each Δt. The average correlation estimate at each Δt is then plotted with error bars representing 68% of the variability at each Δt. We see that the Dirichlet kernel better recovers the theoretical Epps curve given in eq. (2.31) while the Fejér kernel is biased upwards with respect to the Dirichlet kernel (and the theoretical Epps effect) because of the weighting of frequencies. The figures can be recovered using the Julia script file MMZandMM.jl on the GitHub resource (Chang et al., 2020c).

Figure 2.11 plots eq. (2.31) as a function of Δt (the thick black line labelled "Theoretical") ranging from 1 to 100 seconds. This is compared to the Malliavin-Mancino estimators. The corresponding N for a given Δt in the Malliavin-Mancino estimator is given by:[9]

$$N = \left\lfloor \frac{1}{2}\left(\frac{T}{\Delta t} - 1\right)\right\rfloor. \tag{2.32}$$

Here the Malliavin-Mancino estimates are computed using the fast Gaussian gridding NUFFT method with a tolerance of $\varepsilon = 10^{-12}$. This is done for the Dirichlet (blue $\times$ with label "MM Dirichlet") and the Fejér (red $+$ with label "MM Fejér"). The process of simulating, sampling and estimating is repeated 100 times so that the variability between the estimates can be investigated for various n_i with $i = 1,2$ and N. Here, $n_i \approx T/\lambda$ on average from the Poissonian sampling. Figure 2.11 plots the average correlation estimates over the various replications with error bars representing 68% of the variability between the estimates at each Δt. Here the sampling standard deviation and a t-distribution is used to compute the error bars. Since the estimation is performed 100 times, we use a t-distribution with 99 degrees of freedom.

There are three things to notice in Figure 2.11. First, the precision of the estimates improves as n_i and N increase *i.e.* for decreasing time-scales. For a fixed T, we see the effect of larger N on the precision (ignoring the variability from n_i changing from the replications). However, the exact contributions of n_i and N leading to the increased precision for larger T is unclear, as larger T implies larger n_i and N. Second, the Dirichlet kernel better recovers the theoretical Epps curve while the Fejér is biased upwards with respect to the Dirichlet basis and the theoretical Epps effect. This is because the Fejér kernel places more weight on the lower frequencies which makes it more stable under microstructure noise (Mancino and Sanfelici, 2011).[10] Finally, due to the weighting of frequencies in the Fejér kernel we get smoother estimates compared to the Dirichlet kernel. Although it is not clear here due to averaging the replications, this is more clear in Figure 2.12 where I plot individual realisations.

Deciding which basis kernel to use depends on how one wants to treat the Epps effect. The Epps effect is well known and has many factors contributing to it (Münnix et al., 2010, 2011; Renò, 2003; Saichev and Sornette, 2014; Tóth and Kertész, 2009, 2007; Mastromatteo et al., 2011; Precup and Iori, 2007). The effects include statistical causes that require correction such as asynchrony (Renò, 2003; Precup and Iori, 2007; Münnix et al., 2011; Tóth and Kertész, 2007) and tick-size (Münnix et al., 2010, 2011), but also genuine effects such as lead-lag (Renò, 2003; Mastromatteo et al., 2011) and sampling interval dependent correlations (Bacry et al., 2013a). If one is purely interested in removing the Epps effect, then the Fejér kernel is more effective as more weight is placed on lower order frequencies to avoid market microstructure noise. This issue as mentioned before is that this can average away and conceal genuine effects causing the Epps effect. Therefore, I argue the Dirichlet kernel is more appropriate because it better recovers the theoretical Epps effect. Correcting for the statistical causes of the Epps effect should be done after estimating the empirical observables. This is to ensure we can recover the genuine causes resulting in a decay of correlations.

Remark 2.4.1 *The reason the Dirichlet kernel does not recover exactly the theoretical Epps effect is because the theoretical Epps effect is derived using the previous tick interpolation. This is fundamentally different to how the Malliavin-Mancino estimator deals with asynchrony. Previous tick interpolation lines up the observations in the time domain, while the Malliavin-Mancino estimator lines of the frequencies in the Fourier domain. It is unclear as to which method is the most representative of reality, but in Chapter 3 I demonstrate some issues with the previous tick interpolation which may point to the fact that the Malliavin-Mancino estimator is more appropriate.*

[9] Note that eq. (2.32) may not always be a perfect conversion due to the range of integer Fourier modes in eq. (2.3).

[10] A detailed investigation of the estimators under the presence of various types of microstructure noise has been investigated by Mancino and Sanfelici (2011).

2.4.2 Real-world data

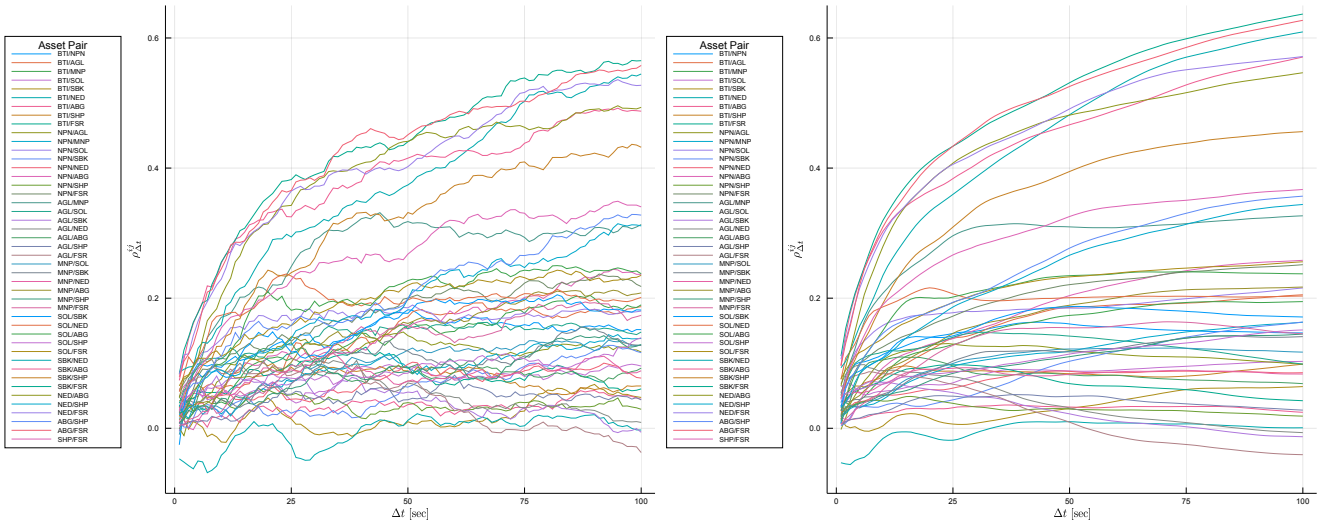
High-frequency correlation dynamics for 10 equities listed on the Johannesburg Stock Exchange (JSE) are given as a real-world example. To this end, Trade and Quote event data for the 10 equities are extracted from Bloomberg Pro. The trade data is then processed to remove any repeated time stamps by aggregating the trades with the same time stamp using a volume weighted average. I discuss this in more detail in Appendix C.2. The 10 equities considered are: FirstRand Limited (FSR), Shoprite Holdings Ltd (SHP), Absa Group Ltd (ABG), Nedbank Group Ltd (NED), Standard Bank Group Ltd (SBK), Sasol Ltd (SOL), Mondi Plc (MNP), Anglo American Plc (AGL), Naspers Ltd (NPN) and British American Tobacco Plc (BTI). The period considered is the week from 24/06/2019 to 28/06/2019. The data is for a five day period and the equities trade for seven hours and 50 minutes each day. This yields $T = 5 \times 28,200 = 141,000$ seconds in the period of consideration. The TAQ data is discrete and asynchronous with different rates of trading for different stocks.

Tickers	Vol. Traded	Unique Trades	$1/\hat{\lambda}$ [sec]
BTI	3,143,263	7,893	17.83 ± 0.64
NPN	2,791,054	12,378	11.38 ± 0.32
AGL	5,751,811	9,091	15.49 ± 0.50
MNP	1,701,907	6,562	21.43 ± 0.93
SOL	6,048,773	10,343	13.62 ± 0.43
SBK	9,427,755	7,441	18.93 ± 0.65
NED	4,518,354	7,090	19.85 ± 0.69
ABG	6,607,644	6,572	21.36 ± 0.78
SHP	3,758,655	5,549	25.35 ± 1.01
FSR	38,493,240	10,412	13.53 ± 0.39

Table 2.3: The table provides a summary of the 10 equities considered for the week from 24/06/2019 to 28/06/2019. The table indicates the volume traded, the number of unique trades and the mean inter-arrival time between trades measured in seconds with the 95% confidence interval provided.

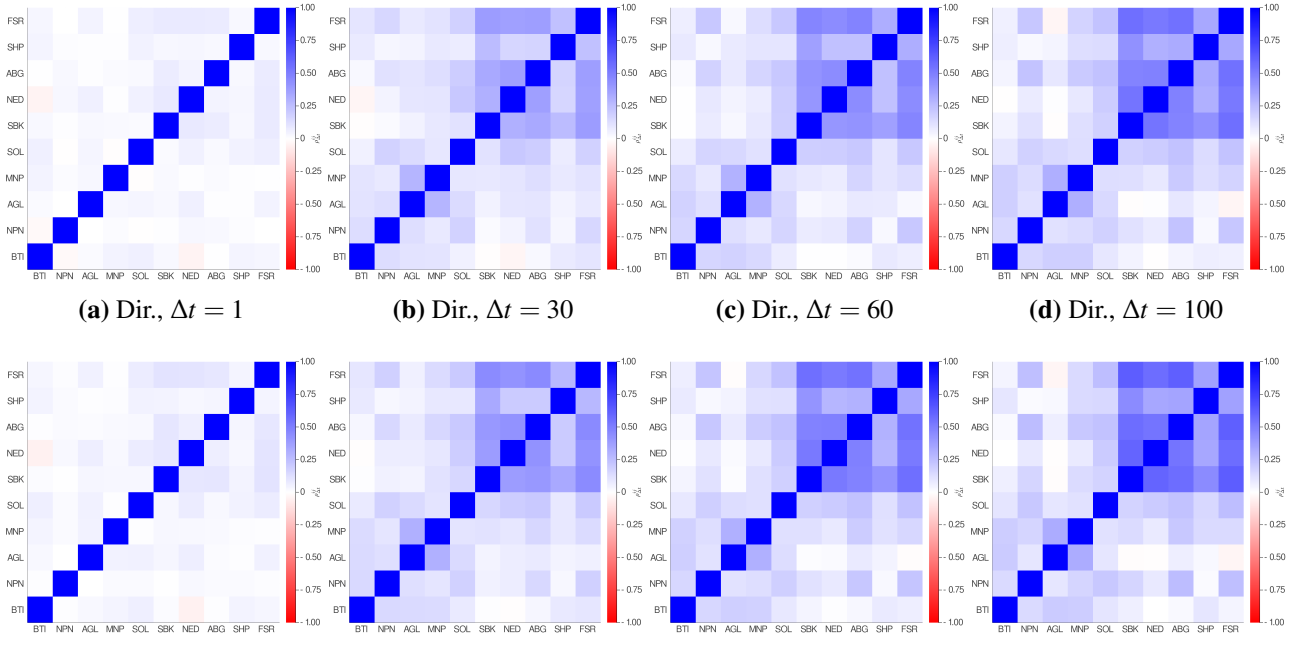
Table 2.3 reports the volume traded, the number of unique trades, and the mean inter-arrival times between the trades for the 10 equities used in the analysis over the five day trading period. The mean inter-arrivals are measured in seconds and a 95% confidence interval is provided using a t-distribution and the standard errors. Notice that the estimated mean inter-arrivals are not the same across the assets. Therefore, for simplicity in order to use eq. (2.31), I assume that the mean inter-arrivals are approximately the same and take on the smaller mean of the two assets, *i.e.* $1/\hat{\lambda} = \min\{1/\hat{\lambda}_i, 1/\hat{\lambda}_j\}$. I must highlight that different mean inter-arrivals for assets actually does not pose a problem for the theoretical Epps effect. Mastromatteo et al. (2011) has provided an extension which can deal with this issue, moreover my derivation of the Epps effect arising from asynchrony also deals with this issue. In this chapter, I am only interested in the general concave shape of the theoretical Epps curves. This is because here I want to demonstrate that under estimation, not all of the measured Epps curves conform to the concave shape from these theoretical models provided by Tóth and Kertész (2007) and Mastromatteo et al. (2011).

Before comparing the theoretical Epps effect against the measured Epps effect, let us first perform some Exploratory Data Analysis to identify the interesting correlation pairs out of the 45 available pairs. Figure 2.12 plots all 45 correlation pairs as a function of the sampling interval Δt ranging from 1 to 100 for the Dirichlet and Fejér basis kernel. In this instance, both the Dirichlet and Fejér kernel produced positive semi-definite covariance matrices. The conversion for Δt to N is given by eq. (2.32), assuming $T = 141,000$. The correlation estimates are obtained using the fast Gaussian gridding implementation of the NUFFT with $\varepsilon = 10^{-12}$. The compute time for 100 different N's took a total of 7.28 seconds using



(a) Correlation and sampling interval (Dirichlet) **(b)** Correlation and sampling interval (Fejér)

Figure 2.12: Here we investigate the Epps effect for 10 equities on the JSE by plotting the correlation estimates using the Malliavin-Mancino estimator as a function of the sampling interval Δt. The conversion between Δt and N is given by eq. (2.32), assuming $T = 141{,}000$ seconds in the 5 day period. Figure 2.12a is the estimates using the Dirichlet basis kernel and Figure 2.12b the Fejér basis kernel. We see that the Fejér kernel produces smoother estimates compared to the Dirichlet kernel. Moreover, we see that most correlation pairs exhibit the Epps effect, but there seems to be pairs where the correlation switches signs for different Δt. The figures can be recovered using the Julia script file Empirical_NUFFT.jl on the GitHub resource (Chang et al., 2020c).



(a) Dir., $\Delta t = 1$ **(b)** Dir., $\Delta t = 30$ **(c)** Dir., $\Delta t = 60$ **(d)** Dir., $\Delta t = 100$

(e) Fej., $\Delta t = 1$ **(f)** Fej., $\Delta t = 30$ **(g)** Fej., $\Delta t = 60$ **(h)** Fej., $\Delta t = 100$

Figure 2.13: Here we plot snapshots of the correlation structure of the 10 equities from the JSE as heat maps. The snapshots are taken for $\Delta t = 1, 30, 60$ and 100 seconds for Figures 2.13a to 2.13d and Figures 2.13e to 2.13h respectively. The first and second row are the Dirichlet and Fejér kernel respectively. We see that the top right quadrant is the banking sector and are highly correlated. More interestingly, we see that the correlation pair FSR/AGL goes from positively correlated to negatively correlated as Δt increases. The figures can be recovered using the Julia script file Empirical_NUFFT.jl on the GitHub resource (Chang et al., 2020c).

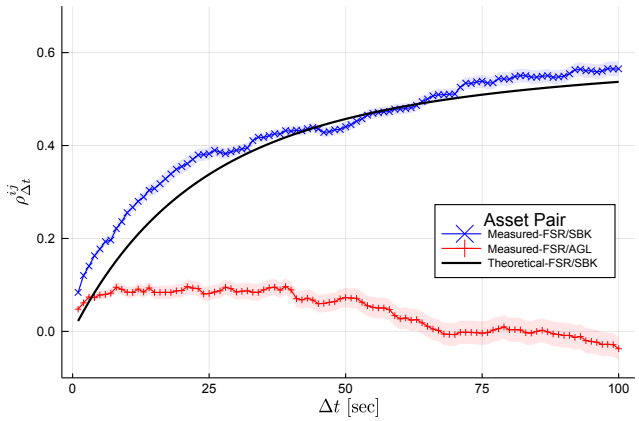
the Dirichlet basis and 9.10 seconds using the Fejér basis; demonstrating the efficacy of the NUFFT method. There are two observations to note from Figure 2.12. First, the Fejér kernel produces smoother estimates compared to the Dirichlet kernel due to the weighting of frequencies. Second, most of the correlation pairs exhibit the Epps effect where the correlation rises as Δt increases. However, there are exceptions where correlation pairs do not exhibit behaviours which can be easily accounted for by the prevailing Epps models. There is an correlation pair where the correlation drops as Δt increases to the point where the sign of the correlation switches.

To determine which pairs to investigate, I plot snapshots of the correlation structures as heat-maps for $\Delta t = 1, 30, 60$ and 100 seconds in Figure 2.13. Here there are two things to notice: first, the top right quadrant are equities from the banking sector and are highly correlated. Second, the correlation pair FSR/AGL is interesting. When $\Delta t = 1$, the pair is positively correlated but becomes negatively correlated when $\Delta t = 100$. This result does not fit easily into our current understanding behind the Epps effect.

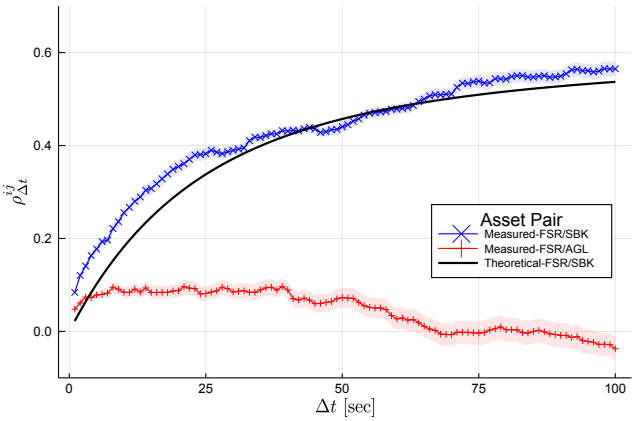
Figure 2.14 investigates this in more detail by plotting the correlation as a function of Δt (the Epps curves) for two particular assets pairs. The first pair FSR/SBK ($\times$) is from the banking sector and is a clear demonstration of the Epps effect. The second pair FSR/AGL ($+$) does not behave in accordance to the Epps effect. Indicative sample error bars are obtained through block bootstrap and the error bars represent 95% of the variability between the estimates at each Δt. The block bootstrap is achieved by splitting the data into 100 calendar time blocks and estimating the Epps curves with one block removed each time. Here, T remains the same across the replications so that the missing block is treated as missing data. The error bars are obtained using the sample standard deviations and a t-distribution with 99 degrees of freedom. The indicative errors are overlaid on the mean estimates from the block bootstrap. The FSR/SBK pair behaves in accordance with our understanding of the Epps effect, thus we compare it to a simple theoretical Epps model (black line) using eq. (2.31). Here I assume the inter-arrival time of trades follow an exponential distribution and both asset have approximately the same mean of $\min\{1/\hat{\lambda}_{FSR}, 1/\hat{\lambda}_{SBK}\} = 1/\hat{\lambda} = 13.5$. I found that $c = 0.621$ produced a relatively good fit for the measured correlations using the Dirichlet basis. Since the FSR/AGL pair does not behave in accordance to the Epps effect, there is little meaning trying to fit our theoretical Epps models to these estimates.

One could possibly try and argue that this behaviour is because there are stocks in the set with a low correlation where the sample error can generate measured sign changes. To test this, we simulate $T = 28,800$ data points for a three feature GBM with correlations $\rho^{12} = -0.5$ ($\times$), $\rho^{13} = 0.7$ ($\circ$) and $\rho^{23} = 0.01$ ($+$) for Figure 2.15a and $\rho^{12} = 0.5$ ($\times$), $\rho^{13} = 0.7$ ($\circ$) and $\rho^{23} = 0.01$ ($+$) for Figure 2.15b. The synchronous case is then sampled with an exponential inter-arrival with a mean of 5 seconds. The Dirichlet estimates are obtained for Δt ranging from 1 to 100 with the conversion to N given in eq. (2.32). This process is repeated 100 times and the average correlation estimate at each Δt is then plotted with error bars (computed using a t-distribution with 99 degrees of freedom and the sample standard deviation) representing 68% of the variability between the estimation paths. We see that when $\rho \approx 0$, we can get sign changes from the increased variability from smaller N. However, I argue this is insufficient as these estimates do not exhibit the pathological behaviour seen in the FSR/AGL pair.

The dynamics seen in Figure 2.14 is important because it illustrates a case where the Epps effect can be modelled and a case where our methods seem insufficient. Although Mastromatteo et al. (2011) caution that a significant portion of the measured Epps effect cannot be explained by current models of the Epps effect, the dynamics seen in FSR/AGL is not what they mean. They are referring to the fact that after correcting for known effects, the Epps effect is still present. What we see here suggests that either: *i.)* current theoretical explanations for the Epps effect are possibly insufficient, or *ii.)* there is more to high-frequency correlation dynamics other than just the Epps effect.

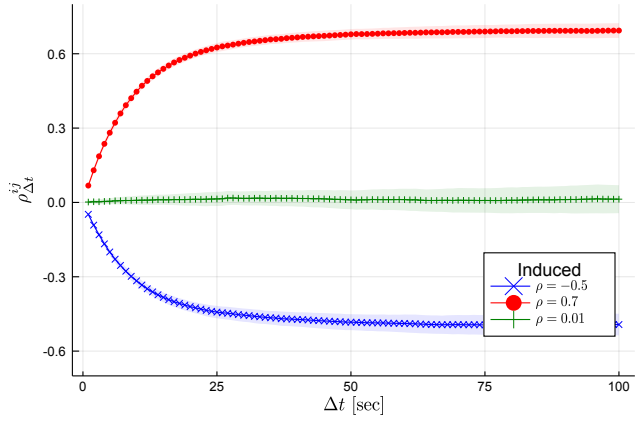


(a) Correlation and sampling interval (Dirichlet)

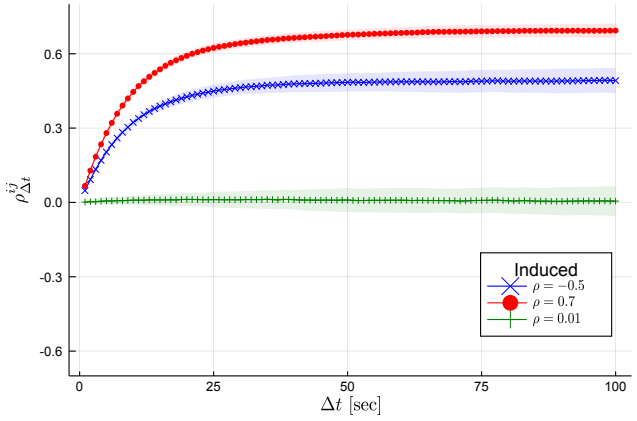


(b) Correlation and sampling interval (Fejér)

Figure 2.14: Here we investigate the Epps curves of the equity pairs FSR/SBK ($\times$) and FSR/AGL ($+$). The conversion between Δt and N is given by eq. (2.32) assuming $T = 141{,}000$. The lines and error bars are the mean estimates and 95% variability between the paths obtained from block bootstrap at each Δt. Furthermore, the theoretical Epps effect in eq. (2.31) is plotted for the FSR/SBK pair assuming $1/\hat{\lambda} = 13.5$. I found that $c = 0.621$ provided a relatively good fit. The theoretical Epps is not plotted for the FSR/AGL pair because the correlation dynamics do not exhibit the Epps effect. The empirical reality of curves, such as the FSR/SBK pair, suggest that current theoretical Epps effect models can plausibly model the correlation dynamics for some asset pairs; while curves such as the FSR/AGL pair, suggest that the current theoretical explanations for the Epps effect are possibly insufficient. The figures can be recovered using the Julia script file Empirical_NUFFT.jl on the GitHub resource (Chang et al., 2020c).



(a) Correlation and sampling interval (Dirichlet, $\lambda = 1/5$, $T = 28800$)



(b) Correlation and sampling interval (Dirichlet, $\lambda = 1/5$, $T = 28800$)

Figure 2.15: Here we investigate whether the interplay from correlation combinations or estimation uncertainty can explain the correlation dynamics found in FSR/AGL from Figure 2.14. A three feature GBM is simulated with $T = 28800$ data points. This is then sampled using Poisson sampling with the $1/\lambda = 5$. The induced correlations are: $\rho^{12} = -0.5$ ($\times$), $\rho^{13} = 0.7$ ($\circ$) and $\rho^{23} = 0.01$ ($+$) for Figure 2.15a and $\rho^{12} = 0.5$ ($\times$), $\rho^{13} = 0.7$ ($\circ$) and $\rho^{23} = 0.01$ ($+$) for Figure 2.15b. The estimates are obtained using the Dirichlet kernel for Δt ranging from 1 to 100 with the conversion to N given in eq. (2.32). The error bars are obtained from 100 replications (computed using a t-distribution with 99 degrees of freedom and the sample standard deviation) and represent 68% of the variability at each Δt. We see that when $\rho \approx 0$ and N is not large enough, estimation uncertainty arises which explains the switching of signs. However, it does not recover the pathological behaviour of the FSR/AGL pair. The figures can be recovered using the Julia script file 3Asset.jl on the GitHub resource (Chang et al., 2020c).

2.5 Closing remarks

In this chapter, I have applied non-uniform fast Fourier transforms in the context of the Malliavin-Mancino integrated estimator. This significantly improves the estimation speed for simulation and real-time applications. Since NUFFT methods are algorithms that allow for fast evaluation of Fourier coefficients of the type in eq. (2.4) at non-uniform grid points, I find that we can accurately recover the appropriate estimates provided we request a low enough tolerance level ε.

I have implemented the NUFFT methods using three averaging kernels: the Gaussian, Kaiser-Bessel, and exponential of semi-circle kernel. Based on the like-for-like algorithmic comparison, the fast Gaussian gridding is the fastest. I highlight there is further room for improvement when it comes to large scale estimation with many features through parallelisation.

Next, I have provided the clear link between the cutting frequency N and its relation to the time-scale of investigation Δt. This relation is compared to the analytic expression characterising the Epps effect arising from asynchrony for both the Dirichlet and Fejér kernel. Here we clearly see the difference between the basis kernels and their use cases. Therefore, I promote the use of the fast Gaussian gridding implementation with the Dirichlet basis if one wants to better recover the empirical observables from a data-informed perspective. Otherwise, the Fejér kernel proves useful when positive semi-definiteness is a requirement, or if one is interested in removing all sources of the Epps effect and avoiding market microstructure noise.

Finally, a preliminary analysis of ultra-high frequency correlation dynamics using Trade and Quote data from the JSE reveals that most correlation dynamics behave in accordance to the Epps effect; but there are correlations dynamics that cannot be simply explained with our current understanding around high-frequency correlation dynamics.

Even though the benchmark was only performed using GBMs, from our Honours work we know that the Malliavin-Mancino estimator works for a wide variety of diffusion models (Chang et al., 2019a). Therefore, by ensuring that the NUFFT indeed recovers the same Fourier coefficients, it is safe to conclude that the NUFFT implementation will also work for a wide variety of diffusion models.

Although this chapter applies NUFFT methods in the case of the integrated estimates, it is easy to apply it in the case of instantaneous estimators as is it just a matter of obtaining additional Fourier coefficients of the price processes in eq. (2.4). I will investigate the instantaneous estimators in the next chapter.

Chapter 3

Fourier instantaneous estimators and the Epps effect

This chapter is the extended version of Chang (2020) which has been accepted by PLOS ONE for publication. This chapter compares the Malliavin-Mancino and Cuchiero-Teichmann Fourier instantaneous estimator to investigate the impact of asynchrony on the instantaneous correlation estimates. Moreover, this chapter carefully examines the impact of the cutting frequencies N and M in the Fourier estimators and provides a simple method to correct for the Epps effect in the case of instantaneous estimators. The chapter begins with motivation of the work in Section 3.1. I then introduce the two Fourier instantaneous estimators in Section 3.2 and compare the two estimators for the various Stochastic models in Section 3.3. Section 3.4 investigates the impact of the cutting frequencies, the time-scale under asynchrony, and demonstrates how to deal with the Epps effect arising from asynchrony. Section 3.5 provides the empirical demonstration. Finally, Section 3.6 ends this chapter with some closing remarks.

3.1 Motivation

Volatility is a key parameter in quantitative finance as I have mentioned, but a large portion of the available tools focus on extracting the integrated quantities. To name a few, we have the classical Realised Volatility (RV) for the continuous-time Itô semi-martingale. Jump robust extensions such as the Bi- and Multi-Power variations studied by Barndorff-Nielsen and Shephard (2004) and Barndorff-Nielsen et al. (2006a,b), which has later been generalised by replacing the power function with different specifications (See Jacod (2008); Todorov and Tauchen (2012); Podolskij and Vetter (2009)). Extensions to deal with asynchrony such as the cumulative estimator proposed by Hayashi and Yoshida (2005), and Fourier estimators such as that proposed by Malliavin and Mancino (2002, 2009) which we investigated in Chapter 2.

These quantities although useful, average out interesting dynamics that occur in the volatility process over $[0,T]$. Instantaneous volatility estimates allow us to estimate the volatility at a particular point in time within $[0,T]$. Moreover, this leads to interesting estimation problems. For example, using the spot volatilities we can estimate the integrated stochastic volatility of volatility by using power variation estimators on the reconstructed volatility path (Cuchiero and Teichmann, 2015), or approaches that rely only on integrated quantities (Sanfelici et al., 2015). Additionally, this can also allow us to estimate the integrated leverage investigated in Curato and Sanfelici (2015). To go even further, Mancino et al. (2017) point out we can further obtain estimates for the instantaneous volatility of volatility and spot leverage. Most of the instantaneous estimators rely on double asymptotics with one of them being a numerical derivative on the integrated quantity to obtain the spot estimate (See Alvarez et al. (2012); Bandi and Renò (2018); Mykland and Zhang (2008)). These correspond to the *local realised volatility estimator*. Malliavin and Mancino (2009) point out that due to the differentiation, these estimators have strong numerical instabilities.

Fourier instantaneous estimators can overcome this issue as it relies on harmonics to reconstruct the spot estimates. They rely on appropriate cutting frequencies N and M. Most of the work regarding the cutting frequencies is focused on convergence and asymptotic properties (Chen, 2019; Cuchiero and Teichmann, 2015; Mancino et al., 2017), rather than practical guidance on how to pick the appropriate frequencies, especially when it comes to the estimators ability to deal with asynchrony and the Epps effect. Additionally, all the work regarding the Epps effect is done using the integrated correlation estimates. To the best of my knowledge, apart from an introduction to this problem by Mattiussi and Iori

(2010), there has yet been any investigation into this topic. The novel contribution of this chapter is to tie together these two threads of literature by thoroughly investigating the practical impact of the cutting frequencies and its effect on the instantaneous estimators and the Epps effect.

3.2 Instantaneous estimators

Instantaneous estimators based on Fourier transforms present several advantages over differentiation based local RV estimators. First, it relies on the integration of the time series rather than the differentiation which makes it numerically stable. Second, the reconstruction of the instantaneous covariance relies on harmonics. Therefore, the degree of smoothness is determined by the user through the choice of cutting frequency M for the reconstruction. Finally, the methods provide a global estimation of the spot volatility. Meaning the volatilities are estimated with similar accuracy at any time t within the interior of the domain (Mancino et al., 2017).

3.2.1 Malliavin-Mancino

As with before in Chapter 2, the Malliavin-Mancino instantaneous estimator aims to obtain the Fourier coefficients of the volatility process $\Sigma^{ij}(t)$ using the Fourier coefficients of the price process $X_t^i = \ln P_t^i$ where P_t^i is a generic asset price at time t. For the convenience of the reader, recall that Malliavin and Mancino (2009) show that the Fourier coefficients of the price process are given as:

$$\begin{aligned}\mathscr{F}(dX^i)(k) &\approx \frac{1}{2\pi}\sum_{h=0}^{n_i-1}\exp\left(-\mathrm{i}kt_h^i\right)\delta_i(I_h),\\ \mathscr{F}(dX^j)(k) &\approx \frac{1}{2\pi}\sum_{\ell=0}^{n_j-1}\exp\left(-\mathrm{i}kt_\ell^j\right)\delta_j(I_\ell),\end{aligned} \tag{3.1}$$

where $(t_h^i)_{h=0,\dots,n_i}$ and $(t_\ell^j)_{\ell=0,\dots,n_j}$ are the observation times. The price fluctuations are $\delta_i(I_h) = X^i_{t^i_{h+1}} - X^i_{t^i_h}$ and $\delta_j(I_\ell) = X^j_{t^j_{\ell+1}} - X^j_{t^j_\ell}$ for asset i and j respectively. Again, notice that $\mathrm{i} = \sqrt{-1}$ and should not be confused with integer indices i for the asset. Using Theorem 2.2.1, the estimates of the Fourier coefficients of the volatility process is given as:

$$\alpha_k\left(\Sigma^{ij}_{n_i,n_j,N}\right) = \frac{2\pi}{2N+1}\sum_{|s|\le N}\mathscr{F}(dX^i)(s)\mathscr{F}(dX^j)(k-s), \tag{3.2}$$

for $k \in \{-M,\dots,M\}$. Using the Fejér kernel, the instantaneous volatility/co-volatility can be reconstructed with eq. (3.2) as:

$$\hat{\Sigma}^{ij}_{n_i,n_i,N,M}(t) = \sum_{|k|\le M}\left(1-\frac{|k|}{M}\right)e^{\mathrm{i}tk}\alpha_k\left(\Sigma^{ij}_{n_i,n_j,N}\right). \tag{3.3}$$

Before in Chapter 2, the integrated quantities only require the evaluation of $\{-N,\dots,N\}$ Fourier coefficients. Here we need to evaluate $\{-M-N,\dots,N+M\}$ Fourier coefficients for each asset in eq. (3.1). This is computationally expensive, but we can use non-uniform fast Fourier transforms to speed up the evaluation. Specifically, I use the fast Gaussian gridding implementation to do so.

Require:

1. **P**: (n x D) matrix of sampled prices. Non-trade times are represented using *NaNs* or *NAs*.
2. **T**: (n x D) matrix of sampled times. Non-trade times are represented using *NaNs* or *NAs*.
3. **t**: vector of times to estimate the spot estimates. Elements of $\mathbf{t} \in [0, 2\pi]$.
4. N (Optional): cutoff frequency (Integer) used in the convolution. Default is set to be the Nyquist cutoff.
5. M (Optional): cutoff frequency (Integer) used in the reconstruction. Default is set to $\frac{1}{2\pi}\frac{1}{8}\sqrt{n}\log n$.
6. tol (Optional): error tolerance for NUFFTs. Determines the number of grid points to spread. Default is set to 10^{-12}.

Step I. Initialisation.

I.1. Re-scale the sampled times (**T**) (See Algorithm 14).

I.2. Compute the Nyquist cutoff (N)—unless specified otherwise through input parameter (See Algorithm 15).

Step F: Compute the Fourier coefficients, $k \in \{-N-M, ..., N+M\}$.

for i = 1 to D **do**

F.1. Extract the re-scaled sampled times for the i^{th} object: $\tilde{\mathbf{t}}^i = \mathbf{T}(i)$, excluding any *NaNs* or *NAs*.

F.2. Extract and compute the logarithm of the sampled prices for the i^{th} object: $\tilde{\mathbf{p}}_i = \ln\left(\mathbf{p}_i\left(\tilde{\mathbf{t}}^i\right)\right)$, excluding any *NaNs* or *NAs*.

F.3. Compute the returns: $\delta_i(I_h) = \tilde{p}_i(\tilde{t}^i_{h+1}) - \tilde{p}_i(\tilde{t}^i_h)$

F.4. Compute the Fourier coefficients:

$$c_k^+(i) = \sum_{h=1}^{n_i-1} e^{\mathrm{i}k\tilde{t}^i_h}\delta_i(I_h); \quad c_k^-(i) = \sum_{h=1}^{n_i-1} e^{-\mathrm{i}k\tilde{t}^i_h}\delta_i(I_h)$$

end for

Step C: Convolution.

C.1. Compute Fourier coefficient of the volatility, $k \in \{-M, ..., M\}$:

$$\alpha_k\left(\Sigma^{ij}_{n_i,n_j,N}\right) = \frac{1}{2N+1}\sum_{|s|\leq N} c_s(i)c_{k-s}(i)$$

Step R: Reconstruct.

R.1. Reconstruct the spot estimate at time t:

$$\hat{\Sigma}^{ij}_{n_i,n_i,N,M}(t) = \sum_{|k|\leq M}\left(1-\frac{|k|}{M}\right)e^{\mathrm{i}tk}\alpha_k\left(\Sigma^{ij}_{n_i,n_j,N}\right)$$

return $(\hat{\Sigma}^{ij}_{n_i,n_i,N,M}(t))$

Algorithm 8: The Malliavin-Mancino Fourier instantaneous estimator computes the spot volatility at time t. The algorithm follows the same outline as Algorithm 1 but adapted for spot estimates. The implementation can be found in the script file MM-Inst.jl on the GitHub resource (Chang, 2020).

Here there are two parameters N and M that require tuning in eq. (3.3). As with Chapter 2, N dictates how many Fourier modes of the price process are used in estimating the Fourier coefficients of the volatility. This controls the level of averaging and has a direct impact on the time-scale of investigation. The second parameter M is the reconstruction frequency. This determines how many Fourier coefficients of the volatility are used in approximating the spot volatility. I will investigate the impact of N and M with more care in Section 3.4. A reminder of Remark 2.2.2, n_i and n_j need not be the same as the

Malliavin-Mancino estimator performs the convolution in the Fourier domain. Thus there is no need to line the data up in the time domain by means of imputation.

3.2.2 Cuchiero-Teichmann

Cuchiero and Teichmann (2015) provide an extension based on the Malliavin-Mancino Fourier estimator to account for the presence of jumps. This is achieved by modifying jump robust estimators of integrated RVs considered by Barndorff-Nielsen et al. (2006a,b); Jacod (2008); Podolskij and Vetter (2009); Todorov and Tauchen (2012) in order to obtain estimates for the Fourier coefficients of the realised path of the instantaneous volatility. Rather than performing a convolution using the Fourier coefficients of the price process, Cuchiero and Teichmann (2015) modify the Bi- and Multi-Power variation so that the Fourier coefficients of $\rho_g(\Sigma)$ are directly obtained from the price observations X_t where $\rho_g(\cdot)$ is a continuous invertible function and Σ the volatility process.

The Cuchiero-Teichmann spot volatility is estimated through three steps. First, we need an estimator for the Fourier coefficients of $\rho_g(\Sigma)$ from the discrete price observations. Meaning we need an estimator for:

$$\mathscr{F}(\rho_g(\Sigma))(k) = \frac{1}{T}\int_0^T \rho_g(\Sigma(t))\, e^{-\mathrm{i}\frac{2\pi}{T}kt}. \tag{3.4}$$

Cuchiero and Teichmann (2015) show under various assumptions that the estimator of eq. (3.4) taking the form:

$$V(X,g,k)_T^n = \frac{1}{n}\sum_{h=1}^{\lfloor nT\rfloor} e^{-\mathrm{i}\frac{2\pi}{T}kt_{h-1}^n} g\left(\sqrt{n}\Delta_h^n X\right), \tag{3.5}$$

converges to the required Fourier coefficients in eq. (3.4) (See Theorem 3.4 of Cuchiero and Teichmann (2015) and the list of assumptions). Here $1/n = \Delta t$ is the discretisation interval, $\Delta_h^n X = X_{t_h^n} - X_{t_{h-1}^n}$, and the time grid for the observations of X_t in $[0,T]$ are equal and equidistant, *i.e.* $t_h^n = h/n$, $h = 0,...,\lfloor nT\rfloor$.

Remark 3.2.1 *Note that $\delta_i(I_h)$ and $\Delta_h^n X$ are both price fluctuations. Separate notation is used highlight that the observation times in $\delta_i(I_h)$ need not be equidistant, while $\Delta_h^n X$ must be strictly equidistant.*

Second, we can apply the Fourier-Fejér inversion to reconstruct the path of:

$$\widehat{\rho_g(\Sigma(t))}_{n,M} = \frac{1}{T}\sum_{|k|\le M}\left(1-\frac{|k|}{M}\right) e^{\mathrm{i}\frac{2\pi}{T}kt} V(X,g,k)_T^n. \tag{3.6}$$

Finally, the spot volatility can be obtained by inverting eq. (3.6) yielding:

$$\hat{\Sigma}_{n,M}(t) = \rho_g^{-1}\left(\widehat{\rho_g(\Sigma(t))}_{n,M}\right). \tag{3.7}$$

Up till now in the derivation of the Cuchiero-Teichmann estimator, I have avoided the use of asset indices i and j. This is because unlike the Malliavin-Mancino estimator which estimates the Fourier coefficients of the volatility process using a convolution with the Fourier coefficients of the price process, the Cuchiero-Teichmann estimator directly obtains the Fourier coefficients of the volatility process through the prices. This means that obtaining the spot estimates of the volatility $\hat{\Sigma}_{n,M}^{ij}(t)$ where $i \neq j$ requires the use of the polarisation identity. Therefore, the Cuchiero-Teichmann estimator does not have a method to over come the issue of asynchrony like the RV estimator which requires asynchronous data to be synchronised beforehand through imputation. For the imputation I use the most common previous tick interpolation as it is easy to implement and does not assume anything behind the genuine nature of the process.

The choice of the function $g(\cdot)$ plays an important role in the ability for the estimator to deal with jumps. Cuchiero and Teichmann (2015) provide several choices of $g(\cdot)$ that satisfy the assumptions, here I use the Todorov and Tauchen (2012) specification of the function $g(x) = \cos(x)$ which has the associate $\rho_g(\Sigma(t)) = e^{-\Sigma(t)/2}$. By using the polarisation identity, we have:

$$\begin{aligned}\hat{\Sigma}^{ii}_{n,M}(t) &= -2\log\left(\widehat{\rho_{g_{ii}}(\Sigma(t))}_{n,M}\right), \quad i \in \{1,2\},\\ \hat{\Sigma}^{12}_{n,M}(t) &= \frac{1}{2}\left(-2\log\left(\widehat{\rho_{g_{12}}(\Sigma(t))}_{n,M}\right) - \hat{\Sigma}^{11}_{n,M}(t) - \hat{\Sigma}^{22}_{n,M}(t)\right).\end{aligned} \tag{3.8}$$

Require:

1. **P**: (n x 2) matrix of prices. The trades must be strictly synchronous.
2. **t**: vector of times to estimate the spot estimates. Elements of $\mathbf{t} \in [0,T]$, $T = 1$ by default.
3. M: cutoff frequency (Integer) used in the reconstruction.

Step I. Initialisation.

I.1. Compute: $\tilde{\mathbf{P}} = \ln(\mathbf{P})$.

I.2. Compute: $\Delta^n_h = X^i = \tilde{P}^i_{h/n} - \tilde{P}^i_{(h-1)/n}, \quad i \in \{1,2\}$.

I.3. Compute: $\Delta^n_h = X^{12} = \left(\tilde{P}^1_{h/n} + \tilde{P}^2_{h/n}\right) - \left(\tilde{P}^1_{(h-1)/n} + \tilde{P}^2_{(h-1)/n}\right)$.

Step II: Compute $\widehat{\rho_g(\Sigma(t))}_{n,M}$.

II.1. Compute:

$$\widehat{\rho_{g_{ii}}(\Sigma(t))}_{n,M} = \frac{1}{T}\sum_{h=1}^{nT}\frac{1}{n}F_M\left(\frac{2\pi}{T}\left(t - \frac{h-1}{n}\right)\right)\cos\left(\sqrt{n}\Delta^n_h X^i\right)$$

II.2. Compute:

$$\widehat{\rho_{g_{12}}(\Sigma(t))}_{n,M} = \frac{1}{T}\sum_{h=1}^{nT}\frac{1}{n}F_M\left(\frac{2\pi}{T}\left(t - \frac{h-1}{n}\right)\right)\cos\left(\sqrt{n}\Delta^n_h X^{12}\right)$$

Step III: Invert $\widehat{\rho_g(\Sigma(t))}_{n,M}$.

III.1. Compute:

$$\hat{\Sigma}^{ii}_{n,M}(t) = -2\log\left(\widehat{\rho_{g_{ii}}(\Sigma(t))}_{n,M}\right), \quad i \in \{1,2\}$$

III.2. Compute:

$$\hat{\Sigma}^{12}_{n,M}(t) = \frac{1}{2}\left(-2\log\left(\widehat{\rho_{g_{12}}(\Sigma(t))}_{n,M}\right) - \hat{\Sigma}^{11}_{n,M}(t) - \hat{\Sigma}^{22}_{n,M}(t)\right)$$

return $(\hat{\Sigma}^{ii}_{n,M}(t), i \in \{1,2\}$ and $\hat{\Sigma}^{12}_{n,M}(t))$

Algorithm 9: The Cuchiero-Teichmann Fourier instantaneous estimator computes the spot volatility at time t. The algorithm is provided for two assets and uses the polarisation identity. The implementation can be found in the script file MM-JR.jl on the GitHub resource (Chang, 2020).

Remark 3.2.2 *Note that $F_M(\cdot)$ in Algorithm 9 is the Fejér kernel given by:*

$$F_M(x) := \frac{1}{M+1}\frac{\sin\left((M+1)\frac{x}{2}\right)^2}{\sin\left(\frac{x}{2}\right)^2} = \sum_{|k|\leq M}\left(1 - \frac{|k|}{M}\right)e^{\mathrm{i}kx}. \tag{3.9}$$

The Cuchiero-Teichmann estimator has a tuning parameter M which the same as the Malliavin-Mancino estimator, the reconstruction frequency of the spot volatility. However, there is no parameter N controlling the level of averaging. Therefore, the time-scale investigated depends on the discretisation interval $\Delta t = 1/n$.

3.3 Comparison

Here I compare the Malliavin-Mancino (MM) and Cuchiero-Teichmann (CT) spot volatility estimator under the presence of jumps and no jumps for the synchronous and asynchronous case. Here I use two types of volatility models: constant volatility and stochastic volatility. The stochastic models considered are the Geometric Brownian Motion (GBM) for constant volatility with no jumps; the Merton Model for constant volatility with jumps; the Heston Model for stochastic volatility with no jumps; and the Bates-type model for stochastic volatility with jumps. The parameters are given for the period $[0,T]$ where $T = 1$ can be thought of as a day.

Geometric Brownian Motion

Recall the bivariate Geometric Brownian Motion (GBM) satisfies the following system of SDEs

$$\frac{dP_t^i}{P_t^i} = \mu_i dt + \sigma_i dW_t^i, \quad i = 1,2, \tag{3.10}$$

where W_i are Brownian motions with $\text{Corr}(dW^1, dW^2) = \rho^{12}$. The parameters used in the simulation are given in Table 3.1, and the simulation is done using Algorithm 19.

Model	Parameter	Values
GBM/Merton	(μ_1, μ_2)	(0.01, 0.01)
	(σ_1^2, σ_2^2)	(0.1, 0.2)
	ρ^{12}	0.35
Merton	(a_1, a_2)	(-0.005, -0.003)
	(b_1, b_2)	(0.015, 0.02)
	(λ_1, λ_2)	(100, 100)

Table 3.1: Parameters used in the simulation for the Geometric Brownian Motion and the Merton Model.

Merton Model

The bivariate Merton model satisfies the following system of SDEs:

$$\frac{dP_t^i}{P_{t-}^i} = \mu_i dt + \sigma_i dW_t^i + dJ_t^i, \quad i = 1,2. \tag{3.11}$$

Here the correlation is $\text{Corr}(dW^1, dW^2) = \rho^{12}$ and the intervals J^i are independent of the W^i with piecewise constant paths (Glasserman, 2004). J_t^i is defined as:

$$J_t^i = \sum_{j=1}^{N_i(t)} (Y_j - 1), \tag{3.12}$$

where $N_i(t)$ is a Poisson process with rate λ_i, $Y_j \sim LN(a_i, b_i)$ i.i.d and also independent of $N_i(t)$. The t^- is used to indicate the Càdlàg nature of the process near jumps. The parameters used in the simulation are given in Table 3.1, and the simulation is performed using Algorithm 16.

Heston Model

The stochastic volatility models are simulated such that the entire volatility matrix is stochastic. Concretely, the bi-variate Heston model takes the form:

$$\begin{aligned} X_t &= X_0 + \int_0^t -\frac{1}{2}\Sigma^{\text{diag}}(s)ds + \int_0^t \sqrt{\Sigma(s)}dZ_s, \\ \Sigma(t) &= \Sigma(0) + \int_0^t \left(b + M\Sigma(t) + \Sigma(t)M^\top\right)dt + \sqrt{\Sigma(t)}dB_t H + HdB_t^\top \sqrt{\Sigma(t)}, \end{aligned} \tag{3.13}$$

where

- M and H are invertible matrices,
- $\Sigma(0), b - H^2 \in S_2^+$, and
- Z is a 2-dimensional Brownian motion correlated with the 2x2 matrix of Brownian motions B such that $Z = \sqrt{1-\rho^\top\rho}W + B\rho$, where $\rho \in [-1,1]^2$ such that $\rho^\top\rho \leq 1$ and W is a 2-dimensional Brownian motion independent of B.

Here S_2^+ refers to a 2-dimensional positive semi-definite symmetric matrix. The parameters for the simulation are given in Table 3.2, and the simulation is done using Algorithm 17.

Model	Parameter	Values
Heston/Bates	(X_0^1, X_0^2)	(4.6, 4.6)
	$\begin{pmatrix}\Sigma^{11}(0) & \Sigma^{12}(0)\\ \Sigma^{12}(0) & \Sigma^{22}(0)\end{pmatrix}$	$\begin{pmatrix}0.09 & -0.036\\ -0.036 & 0.09\end{pmatrix}$
	M	$\begin{pmatrix}-1.6 & -0.2\\ -0.4 & -1\end{pmatrix}$
	$\alpha = H^2$	$\begin{pmatrix}0.0725 & 0.06\\ 0.06 & 0.1325\end{pmatrix}$
	b	3.5α
	ρ	(-0.3, -0.5)
Bates	$(\lambda_1^X, \lambda_2^X)$	(100, 100)
	(a_1, a_2)	(-0.005, -0.003)
	(b_1, b_2)	(0.015, 0.02)
	$\lambda^{\Sigma^{11}}$	10
	θ	0.05

Table 3.2: Parameters used in the simulation for the Heston and the Bates-type Model. The parameters are borrowed from Cuchiero and Teichmann (2015).

Bates model

Let us consider a Bates-type model (henceforth referred to as the Bates model for simplicity) where jumps occur in both the log-price X_t and in the volatility process $\Sigma(t)$. A standard Bates model only has jumps in the log-price process. Here the jumps in the volatility are only happen for $\Sigma^{11}(t)$ as in Cuchiero

and Teichmann (2015). The 2-dimensional Bates model takes the form:

$$\begin{aligned} X_t &= X_0 + \int_0^t b_s ds + \int_0^t \sqrt{\Sigma(s^-)} dZ_s + \int_0^t \int_{\mathbb{R}^2} \xi \mu^X(d\xi, ds), \\ \Sigma(t) &= \Sigma(0) + \int_0^t \left(b + M\Sigma(t) + \Sigma(t)M^\top \right) dt \\ &\quad + \sqrt{\Sigma(t)} dB_t H + H dB_t^\top \sqrt{\Sigma(t)} + \int_0^t \int_{\mathbb{R}^2} \xi \mu^\Sigma(d\xi, ds), \end{aligned} \tag{3.14}$$

where

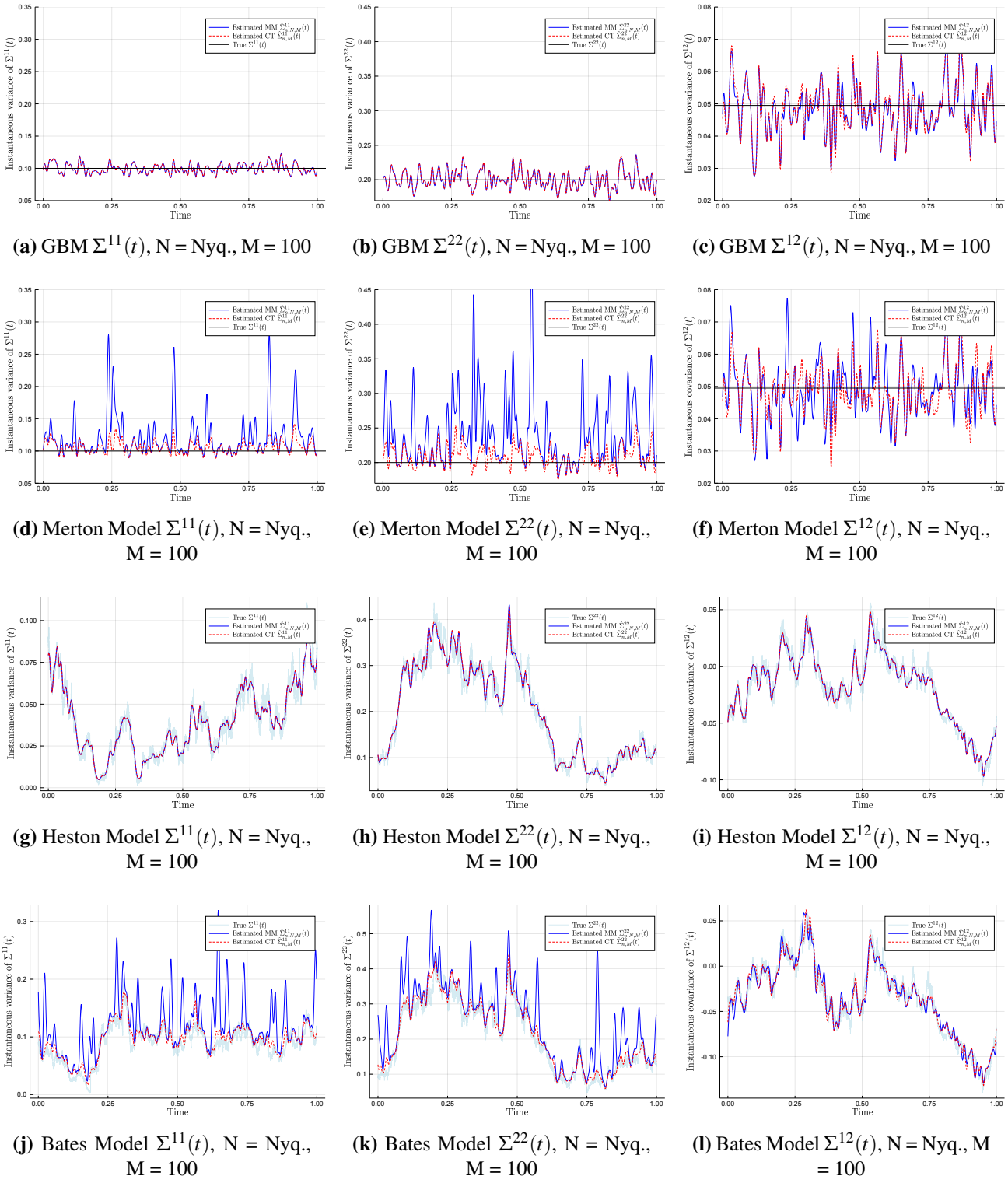
- the Brownian motions Z and B, and the parameters b, H and M are defined as before in the Heston model,
- $\mu^X(d\xi, dt)$ is the random measure associated with the jumps of X. The jumps are Gaussian with mean a_i, standard deviation b_i, and the rate of jumps is λ_i^X,
- $\mu^\Sigma(d\xi, dt)$ is the random measure associated with the jumps of Σ. The jumps are exponential with parameter θ and the rate of jumps is $\lambda^{\Sigma^{11}}$, and
- the drift of X is given by $b_{s,i} = -\frac{1}{2}\Sigma^{ii}(s) - \lambda_i^X \left(e^{a_i - \frac{1}{2}b_i^2} - 1 \right)$.

The parameters for the simulation are given in Table 3.2, and the simulation is performed using Algorithm 18.

3.3.1 Synchronous case

For the synchronous case, let us consider the impact of jumps against no jumps for the two spot estimates where the observations are on an equidistant synchronous grid ($n_1 = n_2 = n$). The number of grid points is set to $n = 28,800$. This corresponds to 1-second data for an eight hour trading day. We will compare the estimates on all four aforementioned stochastic models with the parameters given in Tables 3.1 and 3.2.

Figure 3.1 compares the true underlying volatility (black and light-blue lines) against the Malliavin-Mancino (blue lines) and Cuchiero-Teichmann (red dashes) spot volatility and co-volatility estimates. The sub-figures are organised as follows: the first to last rows are the GBM, Merton, Heston, and Bates model respectively; the first to last columns are $\Sigma^{11}(t)$, $\Sigma^{22}(t)$, and $\Sigma^{12}(t)$ respectively. The cutting frequencies used are N as the Nyquist frequency for the Malliavin-Mancino estimator, and $M = 100$ for both the Malliavin-Mancino and Cuchiero-Teichmann estimator. From the figure we see that for the GBM and Heston model (no jumps), both the Malliavin-Mancino and Cuchiero-Teichmann estimators recover the entire volatility matrix with high fidelity. On the other hand, for the Merton and Bates model (with jumps), the volatility estimates ($\Sigma^{ii}(t)$, $i = 1,2$) using the Malliavin-Mancino estimator presents regions with spikes in volatility due to the jumps. However, the Cuchiero-Teichmann estimator does not present these spikes in volatility because the estimator is robust to jumps. The effect of jumps can still be seen relative to the case with no jumps in the Cuchiero-Teichmann estimator, but overall it can still quite accurately recover the volatilities. Interestingly, both estimators are able to recover the co-volatility ($\Sigma^{12}(t)$) in the case with jumps. In my Honours work, we found that jumps result in a downward bias in the integrated correlation estimates. Here we can clearly see the reason for that. The spikes in the volatility estimates result in a larger normalisation factor for the correlation which ultimately leads to a downward bias.



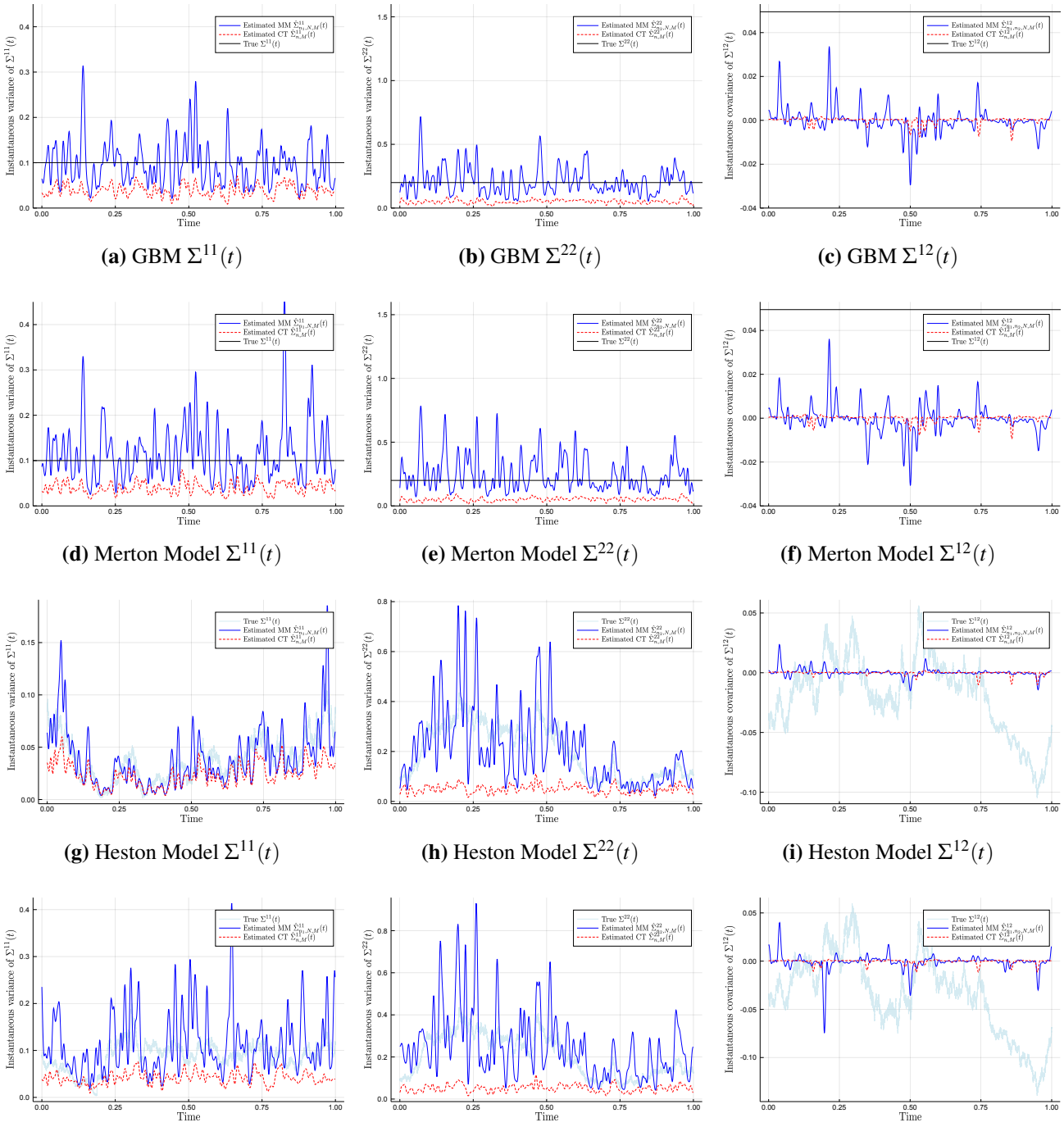
(a) GBM $\Sigma^{11}(t)$, N = Nyq., M = 100 **(b)** GBM $\Sigma^{22}(t)$, N = Nyq., M = 100 **(c)** GBM $\Sigma^{12}(t)$, N = Nyq., M = 100

(d) Merton Model $\Sigma^{11}(t)$, N = Nyq., M = 100 **(e)** Merton Model $\Sigma^{22}(t)$, N = Nyq., M = 100 **(f)** Merton Model $\Sigma^{12}(t)$, N = Nyq., M = 100

(g) Heston Model $\Sigma^{11}(t)$, N = Nyq., M = 100 **(h)** Heston Model $\Sigma^{22}(t)$, N = Nyq., M = 100 **(i)** Heston Model $\Sigma^{12}(t)$, N = Nyq., M = 100

(j) Bates Model $\Sigma^{11}(t)$, N = Nyq., M = 100 **(k)** Bates Model $\Sigma^{22}(t)$, N = Nyq., M = 100 **(l)** Bates Model $\Sigma^{12}(t)$, N = Nyq., M = 100

Figure 3.1: Here we compare the Malliavin-Mancino (blue line with label "Estimated MM") and Cuchiero-Teichmann (red dashes with label "Estimated CT") spot volatility/co-volatility estimator for the four stochastic models. The first to last row we have the GBM, Merton, Heston, and Bates model respectively. The first to last column we compare $\Sigma^{11}(t)$, $\Sigma^{22}(t)$, and $\Sigma^{12}(t)$ respectively. The models are simulated with $n = 28,800$ synchronous grid points. The spot estimates are compared against the true underlying instantaneous volatility matrix (black and light-blue lines with label "True"). In the case with no jumps, both estimators recover the entire volatility matrix. In the case with jumps, the Malliavin-Mancino estimator presents spikes in the volatility estimates while the Cuchiero-Teichmann volatility estimates are not severely affected. Both estimators recover the co-volatility estimates with and without jumps. The figures can be recovered using the Julia script files Synchronous.jl on the GitHub resource (Chang et al., 2020c).



(a) GBM $\Sigma^{11}(t)$ **(b)** GBM $\Sigma^{22}(t)$ **(c)** GBM $\Sigma^{12}(t)$

(d) Merton Model $\Sigma^{11}(t)$ **(e)** Merton Model $\Sigma^{22}(t)$ **(f)** Merton Model $\Sigma^{12}(t)$

(g) Heston Model $\Sigma^{11}(t)$ **(h)** Heston Model $\Sigma^{22}(t)$ **(i)** Heston Model $\Sigma^{12}(t)$

(j) Bates Model $\Sigma^{11}(t)$ **(k)** Bates Model $\Sigma^{22}(t)$ **(l)** Bates Model $\Sigma^{12}(t)$

Figure 3.2: Here we compare the Malliavin-Mancino (blue line with label "Estimated MM") and Cuchiero-Teichmann (red dashes with label "Estimated CT") spot volatility/co-volatility estimator for the four stochastic models under the influence of asynchrony. The first to last row we have the GBM, Merton, Heston, and Bates model respectively. The first to last column we compare $\Sigma^{11}(t)$, $\Sigma^{22}(t)$, and $\Sigma^{12}(t)$ respectively. Asynchrony is introduced by sampling each synchronous grid with $n = 28,800$ grid points using an exponential inter-arrival with mean 30, yielding $n_i \approx n/\lambda_i$. A time-scale of $\Delta t = 1$ second is investigated. The spot estimates are compared against the true underlying instantaneous volatility matrix (black and light-blue lines with label "True"). We see that the Cuchiero-Teichmann estimator underestimates the entire volatility matrix, while the Malliavin-Mancino estimator recovers the volatility but the estimates are more volatile. Both estimators have co-volatility estimates around zero due to the Epps effect arising from asynchrony. The figures can be recovered using the Julia script files Asynchronous.jl on the GitHub resource (Chang et al., 2020c).

3.3.2 Asynchronous case

Let us investigate the additional impact of asynchrony on top of jumps against no jumps for the spot volatility estimators. Asynchrony is introduced using an arrival time representation, where the inter-arrival time between trades follow an exponential distribution with parameter λ, and the mean inter-arrival time is given as $1/\lambda$.

As mentioned, the Cuchiero-Teichmann estimator does not have the ability to deal with asynchrony, thus we need a method to synchronise the data through an imputation scheme. To this end, I will use the previous tick interpolation to synchronise the asynchronous observations onto a new uniform grid. I use the previous tick interpolation because it is simple to implement and does not require any assumptions on the genuine nature of the process. Let $U^i = \{t_k^i\}_{k\in\mathbb{Z}}$, be the set of asynchronous arrival times observed between $[0,T]$ for asset i. The synchronised process is given by $\tilde{X}_t^i = X^i_{\gamma_i(t)}$, where $\gamma_i(t) = \max\{t_k^i : t_k^i \leq t\}$ for $t \in [0,T]$. The resulting synchronised process is piece-wise constant with jumps at $t_k^i \in U^i$. The new uniform grid has width Δt, which will be the time-scale of investigation.

The Malliavin-Mancino estimator deals with asynchrony by lining up the observations in the frequency domain and performing the operations there where the asynchrony in the time domain is not an issue anymore. As we have discussed in Chapter 2, the estimator can investigate various time-scales through an appropriate cutting frequency of N. For the convenience of the reader, recall that the relation between Δt and N is given by:

$$N = \left\lfloor \frac{1}{2}\left(\frac{T}{\Delta t} - 1\right)\right\rfloor, \tag{3.15}$$

where T is the entire interval of investigation. Note T should be measured in the same units as Δt, which is seconds in our case. This means $T = 28,800$ seconds for an 8 hour trading day. Again, the conversion may not be exact because N is an integer.

The experiment is set up by simulating $n = 28,800$ synchronous grid points for the four stochastic models. These processes then have each of their price paths sampled by an exponential inter-arrival with a mean of $1/\lambda_i = 30$ seconds. Thus $n_i \approx T/\lambda_i$ for each price path for the Malliavin-Mancino estimator. I pick N such that $\Delta t = 1$ second is investigated. For the Cuchiero-Teichmann estimator, I synchronise the grid with $\Delta t = 1$ second to get a new synchronised process $\tilde{X}_t^i$ with $n_i = 28,800$ observations. The reconstruction frequency M is chosen to again be 100 so that comparisons can be made with Figure 3.1.

Figure 3.2 compares the true underlying volatility (black and light-blue lines) against the Malliavin-Mancino (blue lines) and Cuchiero-Teichmann (red dashes) spot volatility estimates under the presence of asynchrony. The sub-figures are organised as follows: the first to last row we have the GBM, Merton, Heston, and Bates model respectively; the first to last columns are $\Sigma^{11}(t)$, $\Sigma^{22}(t)$, and $\Sigma^{12}(t)$. For the volatility estimates ($\Sigma^{ii}(t)$, $i = 1,2$), the Malliavin-Mancino estimator recovers the volatilities but asynchrony results in the estimates being more volatile with regions of large deviation from the true volatility. This effect is further confounded with jumps and the contributions from asynchrony and jumps are not clear. Here the Cuchiero-Teichmann estimator under-estimates the volatility. This is because of the previous tick interpolation and $\Delta t = 1$. Using the previous tick interpolation on small time-scales results in many zero returns and this leads to lower volatility estimates. For the co-volatility estimates, both estimators have estimates around zero. With the Malliavin-Mancino estimator, there are still spikes in co-volatility around zero and the spikes are enhanced by jumps. On the other hand, the Cuchiero-Teichmann estimates of the co-volatility are quite constant around zero. The reason behind this decay in the co-volatility is due to the Epps effect.

I will further investigate the impact of the Epps effect arising from asynchrony on the instantaneous correlation in Sections 3.4.2 and 3.4.3, but here I have demonstrated the effect of small Δt (large N) on the individual volatility components. In the bias-MSE analysis of Chapter 2, we found that under asynchronous observations, the Malliavin-Mancino estimator can recover the integrated volatility for

any choice of N with little to no bias. On the other hand, a downward bias appears for the integrated co-volatility as N increases. These results are consistent for the Malliavin-Mancino estimates in the case of the spot estimates.

We see that it is hard to even consider the impact of jumps in the case of asynchronous observations. Therefore, the remaining analysis will focus on understanding the various cutting frequencies and how we can deal with the Epps effect arising from asynchrony.

3.4 Cutting frequencies

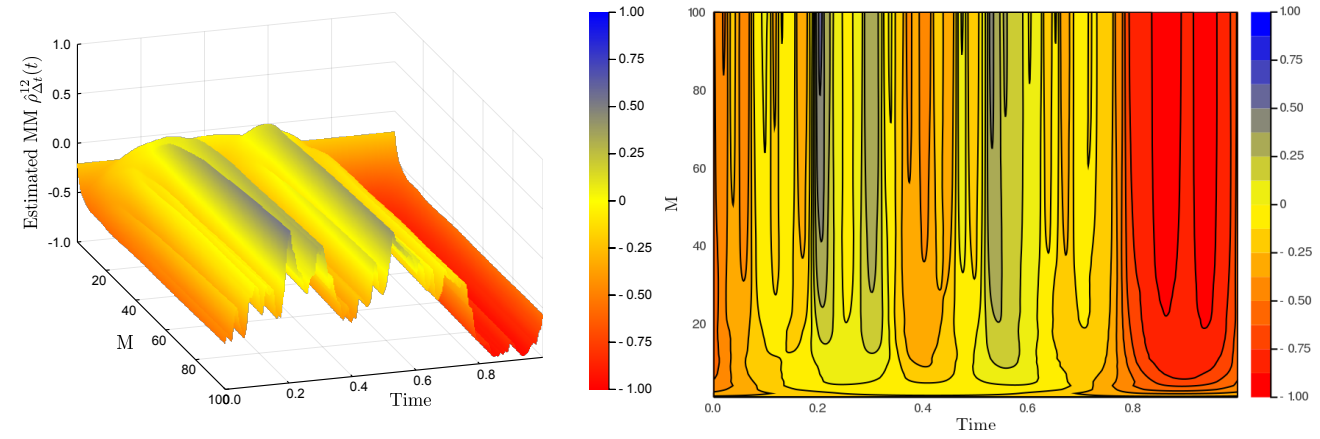
Section 3.3 I chose the cutting frequencies somewhat arbitrarily. However, these cutting frequencies N and M play a pivotal role in the estimators accuracy and its ability to deal with the Epps effect. For the Malliavin-Mancino estimator, N and M affects the asymptotic properties and convergence rates of the estimators (Mancino and Recchioni, 2015; Mancino et al., 2017; Chen, 2019). The level of averaging N determines the time-scale to estimate eq. (3.3). This controls for the impact of the Epps effect caused by asynchrony. The reconstruction frequency M determines the accuracy of the approximation of the spot estimates, the time-scale for the reconstruction of the volatility matrix, and it also affects the estimation of the volatility of volatility either through the reconstructed path (Cuchiero and Teichmann, 2015), or by the number of Fourier modes used in the estimation (Sanfelici et al., 2015).

The literature investigates these cutting frequencies through asymptotic properties and convergence rates. Moreover, the choices suggested in the literature are for very specific cases of asynchrony. Several choices for N and M have been put forward in the literature (for the Malliavin-Mancino estimator). Mancino et al. (2017) through asymptotic properties and simulation experiments suggest that $N = n/2$ and $M = \frac{1}{2\pi}\frac{1}{8}\sqrt{n}\log n$ should be used for the synchronous case; $N = 0.85n^{3/4}$ and $M = \frac{1}{2\pi}\frac{1}{8}\sqrt{n^{3/4}}\log n^{3/4}$ should be used for a special case of asynchrony where $n = n_i = n_j$ and one of the process is observed on an equidistant grid and the other on a non-equidistant grid. Chen (2019) does not concretely suggest choices but rather gives certain guidelines. He suggests that $N \leq \lfloor \underline{n}/2 \rfloor - M$ for the synchronous case, and $N = o(\underline{n}^{4/5})$ for the asynchronous case where $\underline{n} = \min_i n_i$. Chen (2019) places the additional restriction that $N + M$ must be less than equal the Nyquist frequency (of the price process). This is because eqs. (3.3) and (3.4) are estimated from the Fourier coefficients of the price process eq. (3.2). Thus he places this condition to avoid aliasing. Chen (2019) motivates these choices through convergence and asymptotic properties. Mattiussi and Iori (2010) use the Nyquist frequencies $N = n/2$ and $M = N/2$. However, they use a variant of the Malliavin-Mancino estimator with a modified Fejér kernel from Malliavin and Thalmaier (2006) which contains a positive parameter δ that smooths the process to the desired time-scale. Mattiussi and Iori (2010) adopt a different approach by choosing δ to minimise the MSE in an attempt to determine if there exists an optimal time-scale for reconstructing the instantaneous volatility matrix.

Here I want to thoroughly understand the impact of M and N from a practical viewpoint rather than through the convergence and asymptotic properties. From the practical understanding, I will provide a pragmatic approach and demonstration to pick N and M to ameliorate the Epps effect.

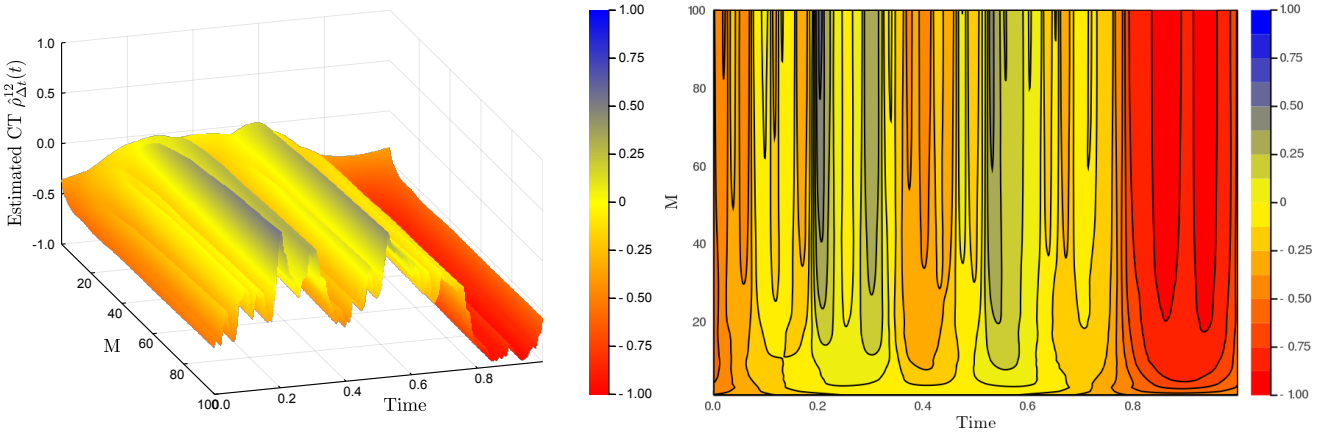
3.4.1 Impact of M

The cutting frequency M is the number of Fourier coefficients of the volatility process used in the reconstruction of the spot estimates. Let us visualise this through a simple demonstration. Here I simulate $n = 28,800$ grid points using a Heston model with parameters in Table 3.2. From here, let us fix N to be the Nyquist frequency, and I will visualise the impact of different values of M ranging from 1 to 100 on the instantaneous correlation estimate $\hat{\rho}^{12}_{\Delta t}(t)$.



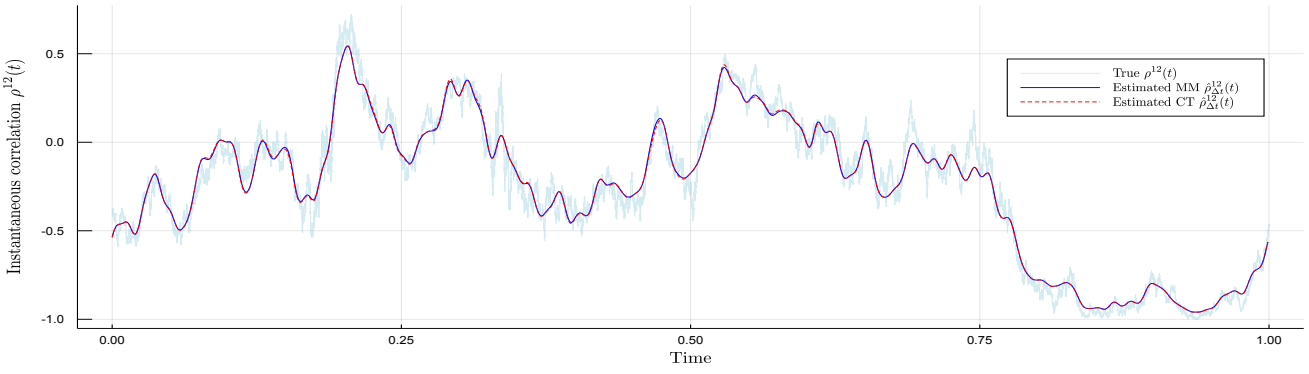
(a) MM Surface plot, N = Nyquist

(b) MM Contour plot, N = Nyquist



(c) CT Surface plot, N = Nyquist

(d) CT Contour plot, N = Nyquist



(e) Synchronous, N = Nyquist, M = 100

Figure 3.3: Here we investigate the impact of M ranging from 1 to 100 on the instantaneous correlation for the synchronous case of a Heston model with parameters in Table 3.2 and N as the Nyquist frequency. We simulate $n = 28,800$ synchronous grid points. The first row of figures is the Malliavin-Mancino estimates visualised using a surface and contour plot; the second row is that of the Cuchiero-Teichmann estimates. The figure in the last row is a comparison between the Malliavin-Mancino (blue line with label "Estimated MM") and the Cuchiero-Teichmann (red dashes with label "Estimated CT") estimates against the true instantaneous correlation (light-blue line with label "True") for $M = 100$. We see that as M increases the additional harmonics allow us to achieve a better approximation. The figures can be recovered using the Julia script file InstantaneousEpps.jl and the animated version can be found in SynchronousVaryingM.gif on the GitHub resource (Chang et al., 2020c).

Remark 3.4.1 *With the choice of the Nyquist frequency for N and $M \neq 0$, we do not satisfy the condition set up by Chen (2019). However, Mancino et al. (2017) argue that under the synchronous case and without microstructure noise, the Nyquist frequency is the best choice. This from their simulation experiments in Mancino and Recchioni (2015) where they found that N as the Nyquist frequency reduces the variance as opposed to a smaller N that prevents aliasing.*

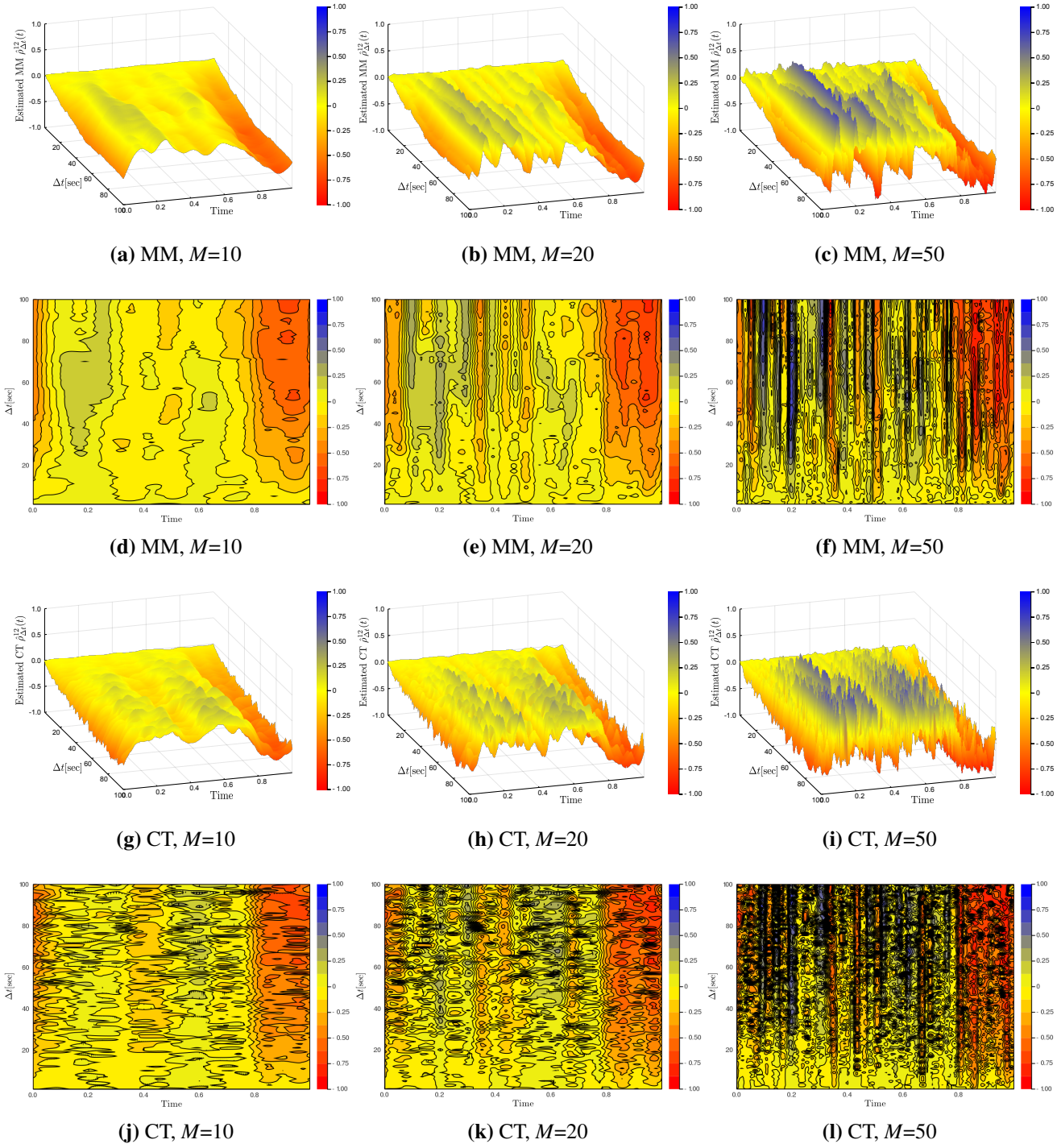
The sub-figures of Figure 3.3 are organised as follows: the first row is the Malliavin-Mancino spot estimates for various values of M plotted as surface and contour plots. The second row is the Cuchiero-Teichmann spot estimates for the same range of M as surface and contour plots. The last row is the Malliavin-Mancino (blue line) and Cuchiero-Teichmann (red dashes) spot estimates compared against the ground truth (light-blue line) for $M = 100$. From the surface and contour plots, we see how increasing M allows us to pick up the finer details of the instantaneous correlation. The harmonics build up to achieve a better approximation by allowing the estimates to reach higher peaks and lower troughs. Moreover, we see that when M is too small the approximation loses the finer details. However, it must be noted that when M is too large the spot estimates present a rapid zigzagging behaviour (increased volatility in the estimates). It must be noted that the choice of $M = N/2$ in Mattiussi and Iori (2010) does not present the rapid zigzagging behaviour due to the smoothing parameter δ in the modified Fejér kernel. The question of how we can achieve this subtle balance is still an open question relating to if there exists an optimal time-scale to reconstruct the spot volatility estimates (Mancino et al., 2017).

Based on my preliminary investigation and experiments, I argue that trying to find a one size fit all choice of M or some optimal time-scale for the reconstruction is the wrong question to ask and is not feasible. The appropriate choice of M should depend on the composition of the true spot parameter of interest. This is because the reconstruction is built upon harmonics, therefore the complexity of the underlying spot parameter of interest should dictate how many Fourier coefficients we need to achieve a good approximation. For example, the instantaneous correlation from the Heston model has a complex form and therefore requires more harmonics to achieve a better approximation. If we use the synchronous choice of M from Mancino et al. (2017) we obtain $M = 34$ which does not provide a good approximation as seen in Figure 3.3. On the other hand, if the instantaneous correlation takes on a simple shape such as Figure 3.5 then few harmonics are already enough to obtain a good approximation. Going even further, if the instantaneous parameter of interest is constant such as the case of the GBM and Merton model, then the appropriate choice of M should be zero. When few harmonics are sufficient in recovering the spot parameter, additional harmonics adds redundant information which results in an unsatisfactory approximation (Mattiussi and Iori, 2010).

The problem of picking an appropriate M such that it provides a sufficient approximation while not being too large causing zigzagging behaviour is all before adding the additional complication of asynchrony. With the added impact of asynchrony, one needs to first pick an appropriate N to avoid the Epps effect before deciding on an appropriate choice of M.

3.4.2 Impact of N

The cutting frequency N is the number of Fourier coefficients of the price process used to estimate the Fourier coefficients of the volatility. Following Remark 3.4.1, the impact of different values of N has been investigated by Mancino and Recchioni (2015) and Mancino et al. (2017) for the synchronous case. Here I am interested in the impact of N in the asynchronous case as a means to investigate the Epps effect on the instantaneous correlations through the implied time-scale Δt (See eq. (3.15)). In the case of the Cuchiero-Teichmann estimator, I will simply use the previous tick interpolation to synchronise the asynchronous samples to a particular time-scale. To set up this demonstration, I simulate $n = 28,800$ grid points using a Heston model with parameters in Table 3.2. Each of the synchronous price paths are then sampled with an exponential inter-arrival with a mean of 30 seconds. This yields $n_i \approx n/\lambda_i$ for



(a) MM, M=10 **(b)** MM, M=20 **(c)** MM, M=50

(d) MM, M=10 **(e)** MM, M=20 **(f)** MM, M=50

(g) CT, M=10 **(h)** CT, M=20 **(i)** CT, M=50

(j) CT, M=10 **(k)** CT, M=20 **(l)** CT, M=50

Figure 3.4: Here we investigate the impact of Δt ranging from 1 to 100 on the instantaneous correlation of a Heston model with parameters in Table 3.2 under the presence of asynchrony for three cases of $M = 10, 20$ and 50 corresponding to the first to third column. A synchronous grid with $n = 28,800$ observations is sampled with an exponential inter-arrival with a mean of 30 seconds. The first two rows are the Malliavin-Mancino estimates visualised as surface and contour plots; likewise, the last two rows are that of the Cuchiero-Teichmann estimates. We see the visualisation of the instantaneous Epps effect and see that the Cuchiero-Teichmann estimates are not stable for various values of Δt for fixed value of time t. The figures can be recovered using the Julia script file InstantaneousEpps.jl and the animated version can be found in AsynchronousVaryingN_M10.gif, AsynchronousVaryingN_M20.gif, and AsynchronousVaryingN_M50.gif on the GitHub resource (Chang et al., 2020c).

each price path. These observations will then be fed into the Malliavin-Mancino estimator, or into the previous tick interpolation for synchronisation for the Cuchiero-Teichmann estimator.

Figure 3.4 investigates the impact of Δt ranging from 1 to 100 seconds for three cases of M. These correspond to the three columns of the figure as $M = 10, 20$ and 50 respectively. The first two rows are the Malliavin-Mancino estimates as surface and contour plots; similarly the last two row are that of the Cuchiero-Teichmann estimates. There are three things to notice in Figure 3.4. First, there is a clear demonstration of the instantaneous Epps effect. When Δt is small the instantaneous correlations remain around zero, while the correlation structure starts to emerge as Δt increases for the various choices of M. Second, M is severely affected by asynchrony, meaning that the zigzagging behaviour occurs for much smaller choices of M relative to the synchronous case. As an example, the choice of $M = 100$ in Figure 3.3e still had relatively smooth estimates but here in the asynchronous case when $M = 50$, the estimates are more volatile in Figures 3.4c and 3.4i. Volatile here refers to the rapid changes in estimates for fixed Δt and different time points t. This leads to the third point specific for the Cuchiero-Teichmann estimates. The Cuchiero-Teichmann estimates not only presents rapid fluctuations for fixed Δt and varying time t (caused by larger M), it also presents rapid fluctuations for fixed t and varying Δt. This means that the Cuchiero-Teichmann estimates are highly unstable under the presence of asynchrony. This can be seen by the horizontal black marks on the contour plots. The black marks are caused by sudden changes in the estimates and the horizontal nature means that for a specific time point t in eq. (3.8), the estimates can take on very different values for similar values of Δt. This is different from the black vertical marks in the contour plot with Malliavin-Mancino estimates. These vertical marks are caused by large M rather than from different values of Δt. This means that for a fixed point in time t of eq. (3.4), various values of Δt do not cause sudden changes in the estimates. This provides a very interesting insight into the two methods of dealing with asynchrony. Through bypassing the time-domain the Malliavin-Mancino estimator is able to produce stable estimates while the previous tick interpolation produces highly unstable estimates for various Δt. This is due to the fact that the Malliavin-Mancino uses all available observations which are fixed. In the case of using the previous tick interpolation, the synchronised sample path can change under various choices of Δt. These changes in the synchronised price paths are more apparent for larger Δt which is where the instabilities are occurring in Figures 3.4j to 3.4l. This is the source causing the instability. The spot estimates provide an interesting perspective on this because this cannot be seen in the integrated estimates because it gets hidden away in the averaging.

The key take away from the two cutting frequencies is that M allows the estimates to build up through additional harmonics, it does not play a role in ameliorating the Epps effect. This is because when M is small the estimates are flat, but not necessary around zero; it is flat around the average correlation. Additional M allows us to resolve the finer features. However, M is affected by the Epps effect in that rapid fluctuations start to occur for smaller values of M compared to the synchronous case. On the other hand, N and the time-scale Δt is key when it comes to dealing with the Epps effect in that N must be small (the time-scale must be large) in order for the Epps effect to be ameliorated. Moreover, the Cuchiero-Teichmann estimator is unstable under asynchronous observations, therefore the next experiment I will focus on dealing with the Epps effect using the Malliavin-Mancino estimator.

Remark 3.4.2 *The reader may have noticed there are actually two time-scales at play. The time-scale in estimating the Fourier coefficients of the volatility determined by N or the discretisation in the Cuchiero-Teichmann estimator, and the time-scale in the reconstruction of the spot estimates determined by M. Unless otherwise specified, when I refer to the time-scale I mean the time-scale in estimating the Fourier coefficients of the volatility.*

3.4.3 Dealing with asynchrony

Chen (2019) provided a break down for the impact of asynchrony. There is a trade-off between the rate

of convergence and the bias caused by asynchrony which he calls the "curse of asynchrony". Moreover, he provides a sufficient (not necessary) condition to ameliorate the effect. Let us look at this trade-off through the lens of the Epps effect. The decay in correlation in the Epps effect arising from asynchrony is caused by the fact that the underlying co-variation is only extracted when the asynchronous observations overlap (Münnix et al., 2011). There are several methods to correct for this in the case of the integrated covariance such as using the Hayashi-Yoshida estimator (Hayashi and Yoshida, 2005), directly accounting for the non-overlapping distortions at a particular time-scale (Münnix et al., 2011), or by simply investigating a larger time-scale (Renò, 2003). When correcting for the Epps effect arising from asynchrony in the case of the instantaneous estimates, it is not clear how we can use the first two approaches. Thus, the most feasible approach is to follow Renò (2003) and find a small N (large time-scale) which corrects for the Epps effect. With this in mind, the choice of N should not be chosen as a one size fits all choice. This is based on the insight from the work in my Honours project where we found that different types of asynchrony and different levels of inhomogeneity will lead to different Epps curves. These different curves require different time-scales to remove the decay as seen in Figures 3.6a to 3.6c. Therefore, an appropriate choice of N should depend on the specific Epps curve from the specific case of asynchrony.

From the derivation of eq. (2.31) in Chapter 4, we will see that a property of the Epps curves is that after the time-scale Δt has reached the saturation level, the integrated correlations will remain at that level for even larger Δt (this can be seen in the asymptotic property of eq. (2.31)). This allows us to find an appropriate time-scale without knowledge of the "true" underlying correlation level such as the case of picking N to minimize the error with respect to the MSE. Concretely, I pick N based on the minimum Δt that achieves the saturation level in the Epps curves. This allows for the decay caused by the Epps effect arising from asynchrony to be removed, while retaining as many Fourier coefficients as possible to provide a better estimate of eq. (3.3) using the Bohr convolution product.

The remaining issue is how one should pick the appropriate M after a suitable choice of N. In the simulation scenario, we can pick M that minimises the MSE but this is problematic with empirical data as we have no knowledge of the true instantaneous dynamics. However, as a consequence of the sampling theorem, M can only be investigated for $M \leq N/2$ to avoid aliasing in the reconstruction of the spot estimates. The non-uniform fast Fourier methods can allow us to quickly perform EDA for various choices of M once N has been chosen, but it still remains unclear how one should pick an appropriate M as I have argued that an appropriate choice of M should depend on the underlying composition of the true instantaneous dynamics.

Remark 3.4.3 *Note that estimates of the Epps curves using the Malliavin-Mancino estimator can present deviations away from the saturation level for larger Δt. This is because small N increases the variability of the estimates seen in the bias-MSE analysis in Figure 2.9. This can be seen for example in Renò (2003) when the cutting frequency is too small.*

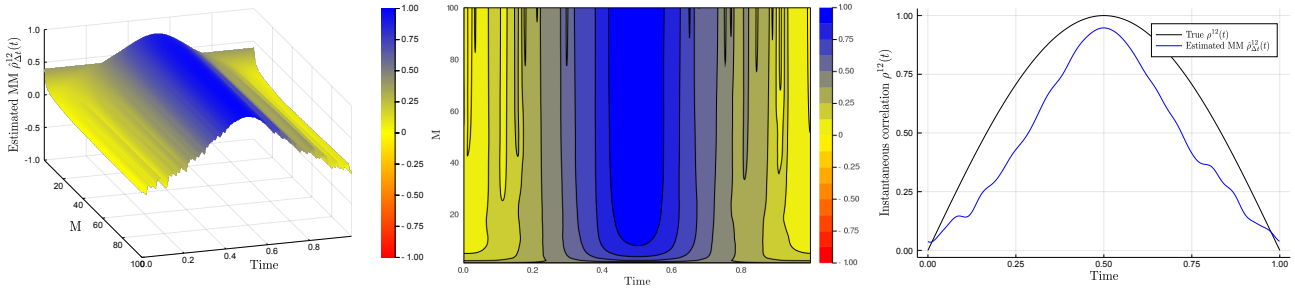
To demonstrate this approach, let us consider a simple bivariate diffusion defined as:

$$dX_t^i = \sigma_i dW_t^i, \quad i = 1,2, \tag{3.16}$$

where the Brownian motions W^1 and W^2 have a deterministic instantaneous correlation given by:

$$\rho^{12}(t) = \sin(t\pi T), \tag{3.17}$$

for $t \in [0,1]$. The process is simulated for $n = 28,800$ grid points, $T = 1$ trading day, and $(\sigma_1^2, \sigma_2^2) = (0.1, 0.2)$. To avoid excessive plots, let us demonstrate the procedure using the Malliavin-Mancino estimates as they are more stable for various choices of Δt. The ground truth is established in Figure 3.5 to see what choices of M recover a good approximation of eq. (3.17).

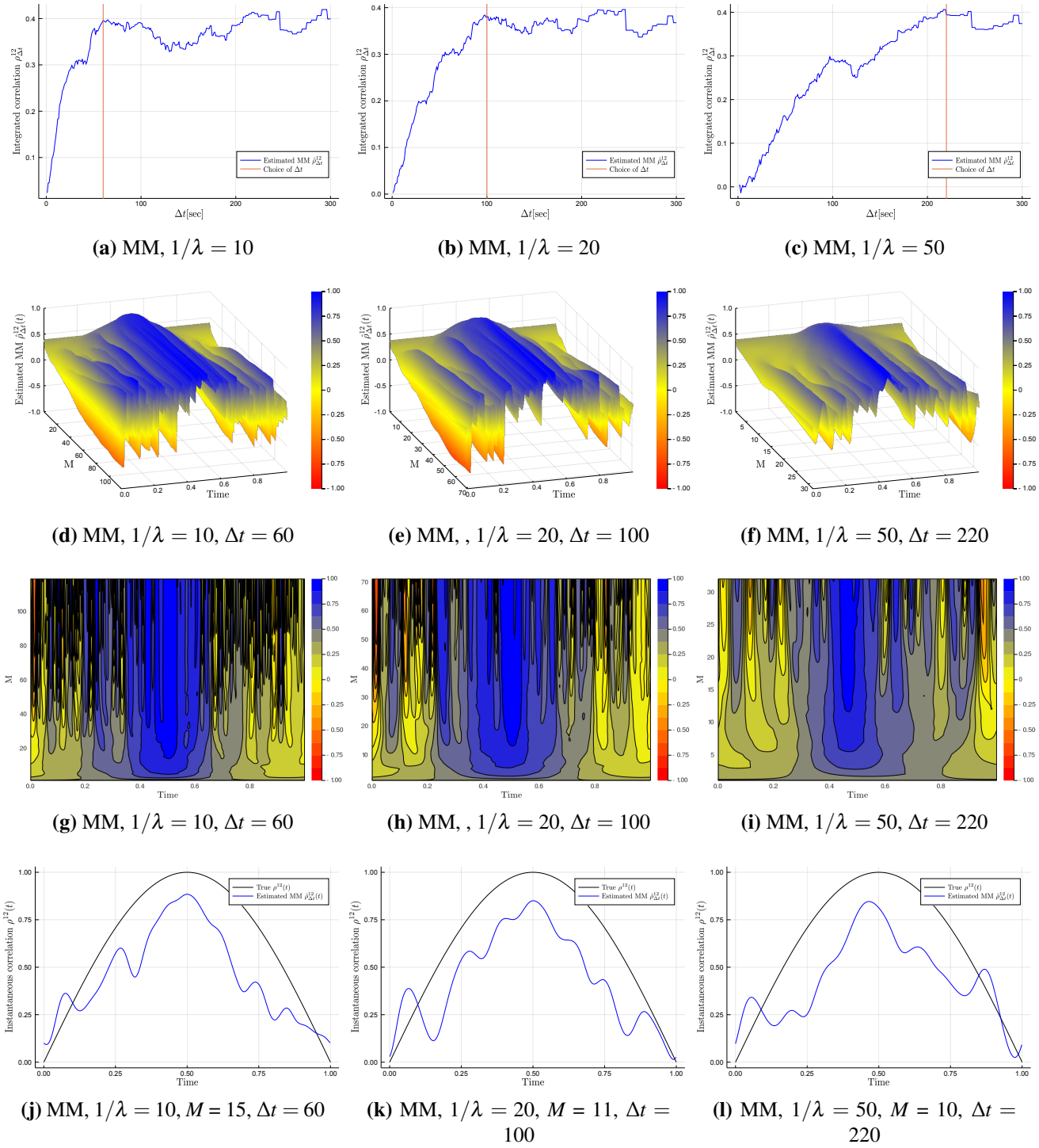


(a) MM Surface plot, $N =$ Nyquist **(b)** MM Contour plot, $N =$ Nyquist **(c)** MM, $N =$ Nyquist, $M = 20$

Figure 3.5: Here we investigate the ground truth for the simple diffusion model with deterministic correlation given by eqs. (3.16) and (3.17) using the synchronous case with $n = 28,800$ grid points. Figures 3.5a and 3.5b investigates the impact of different values of M when N is the Nyquist frequency as surface and contour plots for the Malliavin-Mancino estimates. Figure 3.5c compares the Malliavin-Mancino spot correlation (blue line with label "Estimated MM") using $M = 20$ and $N =$ Nyquist against the true instantaneous correlation (black line with label "True"). We see that when the underlying correlation is simple, small M provides an adequate approximation. The figures can be recovered using the Julia script file OptimalTimeScale.jl and the animated version can be found in OT_MM_syn.gif on the GitHub resource (Chang et al., 2020c).

Here I demonstrate the procedure using the Malliavin-Mancino estimates as they are more stable for various choices of Δt. Figure 3.5 establishes the ground truth to see what choices of M recover a good approximation of eq. (3.17). Figures 3.5a and 3.5b visualises the Malliavin-Mancino spot correlation estimates for various values of M with the Nyquist choice for N as the surface and contour plots respectively. Figure 3.5c compares the Malliavin-Mancino (blue line) correlation spot estimates for $M = 20$ and $N =$ Nyquist against the theoretical correlation (black line). We see that in this case where the instantaneous correlation is very simple, a small value of M is sufficient in approximating the spot correlation. Moreover, additional M adds redundant frequencies which creates fluctuations in the estimates.

To demonstrate the need to pick N based on the Epps curves, I will use a simple case where different levels of inhomogeneity leads to different Epps curves. I will consider three cases of asynchronous observations where each synchronous grid is sampled with an exponential inter-arrival with a mean of 10, 20 and 50 for the three cases. Each case will yield $n_i \approx n/\lambda$ for each asset. Next, by plotting the integrated correlation as a function of the time-scales Δt in Figures 3.6a to 3.6c, it becomes clear that these different cases of asynchrony each reach the saturation level at different time-scales. Hence, the time-scale required to ameliorate the Epps effect should be checked on a case-by-case basis. Figures 3.6a to 3.6c I use the Malliavin-Mancino integrated volatility/co-volatility estimates with the Dirichlet representation because in Chapter 2 we saw that the Dirichlet representation better recovers the theoretical Epps effect in eq. (2.31). The three time-scales (orange vertical line) which removes the Epps effect while preserving the largest N are $\Delta t = 60, 100$ and 220 seconds. These correspond to $N = 239, 143$ and 64 respectively. Therefore, the feasible range of M to investigate ranges from from 1 to 119, 71 and 32 respectively. Figures 3.6d to 3.6f and Figures 3.6g to 3.6i plots the Malliavin-Mancino instantaneous correlation estimates for the range of M and fixed Δt for the three cases as surface and contour plots respectively. With these choices of N we recover correlations that are no longer around zero. However, as we saw earlier, the rapid fluctuations from large M occur for much smaller values of M in the asynchronous case. In the synchronous case, $M = 100$ presented fluctuations but they were relatively small in size; here in the asynchronous case, large fluctuations appear for M larger than 20. It remains unclear as to why this happens, but it is clear that under the presence of asynchrony M needs to be smaller than the synchronous case. This can be problematic when trying to recover more complex instantaneous dynamics such as the instantaneous correlation from the Heston model as small M does not provide enough harmonics for an accurate approximation, but larger M is also not an appropriate option. For illustration, I compare the Malliavin-Mancino spot estimates (blue line) with $M = 15, 11$ and 10 for the respective three cases



(a) MM, $1/\lambda = 10$ **(b)** MM, $1/\lambda = 20$ **(c)** MM, $1/\lambda = 50$

(d) MM, $1/\lambda = 10$, $\Delta t = 60$ **(e)** MM, , $1/\lambda = 20$, $\Delta t = 100$ **(f)** MM, $1/\lambda = 50$, $\Delta t = 220$

(g) MM, $1/\lambda = 10$, $\Delta t = 60$ **(h)** MM, , $1/\lambda = 20$, $\Delta t = 100$ **(i)** MM, $1/\lambda = 50$, $\Delta t = 220$

(j) MM, $1/\lambda = 10$, $M = 15$, $\Delta t = 60$ **(k)** MM, $1/\lambda = 20$, $M = 11$, $\Delta t = 100$ **(l)** MM, $1/\lambda = 50$, $M = 10$, $\Delta t = 220$

Figure 3.6: Here I demonstrate how to pick N on a case-by-case basis. The simple diffusion model with deterministic correlation in eqs. (3.16) and (3.17) is simulated for $n = 28,800$ synchronous grid points. Three cases of asynchrony is introduced by sampling each price path with an exponential inter-arrival with mean 10, 20 and 50. These are the first to third columns respectively. The first row of figures is the Malliavin-Mancino integrated correlation with the Dirichlet representation plotted as a function of the time-scale Δt. The Δt which ameliorates the Epps effect are $\Delta t = 60, 100$ and 220 for the three cases. Thus the second and third row of figures are the Malliavin-Mancino spot estimates investigating M ranging from 1 to 119, 71 and 32 for the three cases plotted as surface and contour plots respectively. The last row compares the Malliavin-Mancino spot estimates (blue line with label "Estimated MM") with $M = 15, 11$ and 10 for the three cases against the true instantaneous correlation (black line with label "True") from eq. (3.17). We see that the instantaneous correlation can be recovered under the presence of asynchrony. The figures can be recovered using the script file OptimalTimeScale.jl and the animated version can be found in OT_MM_asyn_lam10.gif, OT_MM_asyn_lam20.gif, and OT_MM_asyn_lam50.gif on the GitHub resource (Chang et al., 2020c).

against the true instantaneous correlation (black line) from eq. (3.17) in Figures 3.6j to 3.6l. Although these estimates are able to recover the true correlations to a certain degree, the recovery is unsatisfactory compared to the synchronous case. Here I only knew what M to pick because I had knowledge of the true instantaneous correlation so that the choice of M can be adjusted for a satisfactory recovery. This is only possible under simulation conditions, and it remains unclear how to pick M for the empirical analysis. We only know that M should be small and must be less than $N/2$.

A second issue apart from picking M with empirical data is the issue of picking small N to remove the Epps effect. The approach is a naive approach to correct for the Epps effect which can conceal genuine sources of the effect. Picking small N to correct for the effect does not present an issue when the underlying correlation does not depend on Δt such as diffusion models or when there is no lead-lag. As I have mentioned before, the better approach when these genuine effects are present is to disentangle statistical effects causing the Epps effect from the genuine effects at various time-scales. However, it remains unclear how this can be performed for the instantaneous estimates. Therefore, as a preliminary investigation into the instantaneous Epps effect, I will only consider the impact from changing Δt rather than trying to disentangle the various effects.

Remark 3.4.4 *Notice that investigating various M once Δt is fixed is similar to investigating various values of M in the synchronous case. Once a time-scale has been chosen, M simply governs the level of approximation through the number of harmonics. Therefore, to re-emphasize, M is influenced by the Epps effect but does not ameliorate it. Dealing with the Epps effect is all in the time-scales Δt of estimating the Fourier coefficients of the volatility, not the reconstruction (as implied through M).*

3.5 Empirical analysis

Here I will demonstrate the instantaneous Epps effect and ameliorate the effect with empirical Trade and Quote (TAQ) data. To this end, I use two banking equities from the Johannesburg Stock Exchange (JSE) for the week from 24/06/2019 to 28/06/2019. The two equities considered are: Standard Bank Group Ltd (SBK) and FirstRand Limited (FSR). Each trading day is seven hours and 50 minutes (eight hour trading day and 10 minutes for the closing auction). Meaning $T = 1$ day equals $28,200$ seconds. The data was obtained from Bloomberg Pro and processed to remove repeated time stamps by aggregating trades using a volume wieghted average. This is discussed in more detail in Appendix C.2. I investigate each trading day separately since there is not clear how I can connect the trading days. This is because the spot estimates are built upon harmonics, so naively connecting the days can cause potential issues in the estimates. For example, if the correlation at the end of the day is very different to the start of the next day such as Tuesday 25/06/2019 and Wednesday 26/06/2019 in Figure 3.8, the harmonics will need to somehow connect. This can lead to different estimates. Thus, we cannot just stick the two days together, or just treat it as missing data like in Chapter 2 because the Cuchiero-Teichmann estimator cannot handle missing data.

I begin by plotting the integrated correlation estimates as a function of the time-scale Δt using the Malliavin-Mancino integrated volatility/co-volatility estimator with the Dirichlet representation. The time-scale is adjusted with an appropriate N using eq. (3.15) and ranges from 1 to 400 seconds. Figure 3.7 we see that the integrated correlation can vary considerably between the days for the same equity pair, but here they all exhibit the Epps effect. To avoid excessive figures, I will only investigate the instantaneous correlation for Tuesday 25/06/2019 (orange line) and Wednesday 26/06/2019 (green line) as they both reach the saturation level around $\Delta t = 300$ seconds and they present different saturation levels. However, the figures for the entire week can be found in the GitHub resource (Chang et al., 2020c).

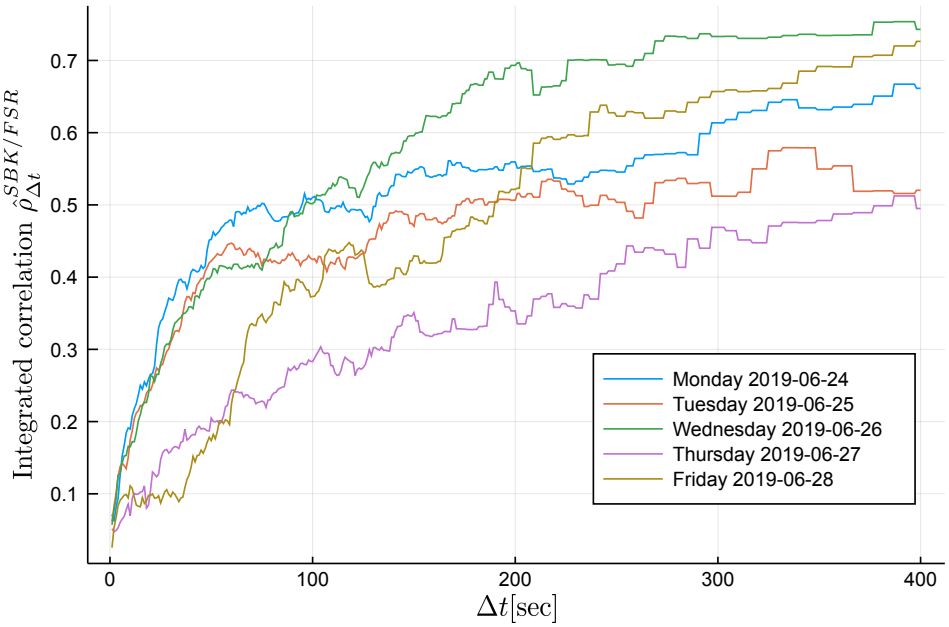
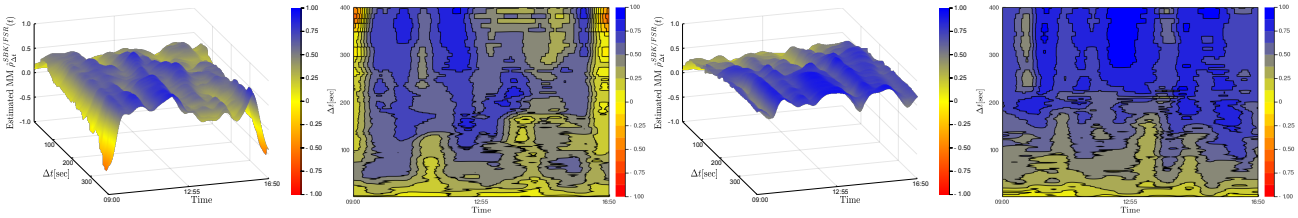


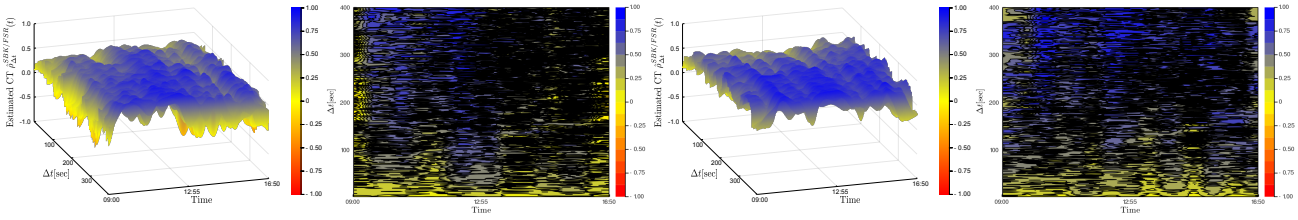
Figure 3.7: Here I plot the integrated correlation estimates for the equity pair SBK/FSR at various time-scales Δt for each trading day of the week from 24/06/2019 to 28/06/2019. The estimates are obtained using the Malliavin-Mancino integrated volatility/co-volatility estimate with the Dirichlet kernel and the time-scale ranges from 1 to 400 seconds controlled using eq. (3.15). As per the legend, Monday the 24/06 is the blue line, Tuesday the 25/06 is the orange line, Wednesday the 26/06 is the green line line, Thursday the 27/06 is the purple line, and Friday the 28/06 is the dark-green line. The figure can be recovered using the Julia script file Empirical_Inst.jl on the GitHub resource (Chang et al., 2020c).

Next, I demonstrate the existence of the instantaneous Epps effect with empirical data in Figure 3.8. The first and second row plots the surface and contour plots of the Malliavin-Mancino and Cuchiero-Teichmann for fixed $M = 10$ and varying Δt respectively. The first and second columns are for Tuesday the 25/06/2019, and the third and fourth columns are for Wednesday the 26/06/2019. The time-scale Δt is controlled using eq. (3.15) for the Malliavin-Mancino estimator while the Cuchiero-Teichmann estimator adjusts the time-scale using the discretisation size in the previous tick interpolation. Now to start the previous tick interpolation for each day, I set $t = 0$ once the equity pair has each made their first trade. This is discussed in more detail in Appendix C.3. We see similar results to the simulation experiments here. First, the instantaneous Epps effect is present. The spot estimates are around zero for small Δt and increase as the time-scale increases. Second, the previous tick interpolation presents instabilities in the Cuchiero-Teichmann estimates for fixed time t and varying Δt. This is seen again as the horizontal black marks in the contour plots. Third, and interestingly, the correlation dynamics within each day is different in the two days investigated. Looking closely at the Malliavin-Mancino estimates, we see that on Tuesday the 25/06/2019 there are large dips in correlations around the open and close of the trading day and also a smaller dip in the afternoon. However, on Wednesday the 26/06/2019 the correlation remains stable around the same level for the entire day. The spot correlations provide a lot more information in contrast to the integrated correlations. We see that these dips are aggregated and hidden into the integrated correlation resulting in a lower saturation level for Tuesday the 25/06/2019 in Figure 3.7.

From the Epps curves in Figure 3.7, I pick the time-scale $\Delta t = 300$ to ameliorate the Epps effect for Tuesday the 25/06/2019 and Wednesday the 26/06/2019. This results in $N = 46$ and thus I investigate M ranging from 1 to 23. The first and second row of Figure 3.9 plots the surface and contour plots of the Malliavin-Mancino and Cuchiero-Teichmann for fixed $\Delta t = 300$ and varying M respectively. The first and second columns are for Tuesday the 25/06/2019, and the third and fourth columns are for Wednesday

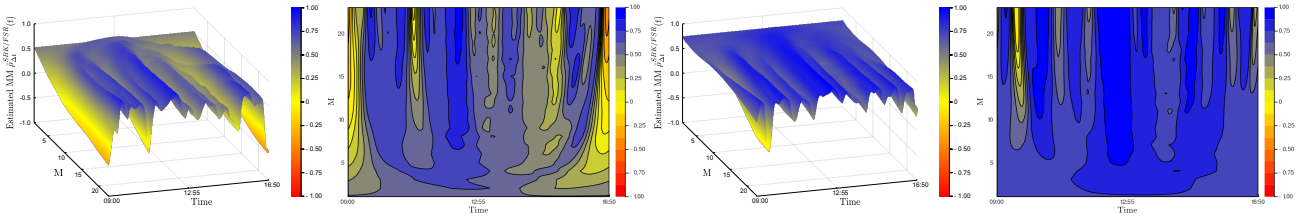


(a) MM, $M = 10$, 25/06 **(b)** MM, $M = 10$, 25/06 **(c)** MM, $M = 10$, 26/06 **(d)** MM, $M = 10$, 26/06

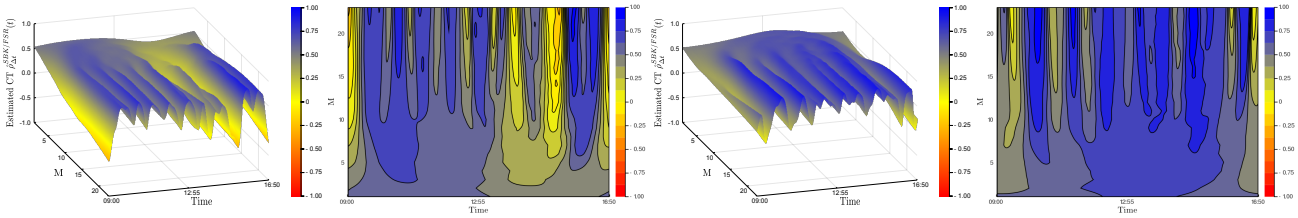


(e) CT, $M = 10$, 25/06 **(f)** CT, $M = 10$, 25/06 **(g)** CT, $M = 10$, 26/06 **(h)** CT, $M = 10$, 26/06

Figure 3.8: Here we demonstrate the instantaneous Epps effect for empirical data. The first and second row plots the surface and contour plots of the Malliavin-Mancino and Cuchiero-Teichmann for fixed $M = 10$ and varying Δt respectively. The first and second columns are for Tuesday the 25/06/2019; similarly, the third and fourth columns are for Wednesday the 26/06/2019. The time-scale Δt is controlled using eq. (3.15) for the Malliavin-Mancino estimator, while the Cuchiero-Teichmann estimator adjusts the time-scale using the previous tick interpolation. We see the Epps effect is present for the instantaneous correlations, and the Cuchiero-Teichmann presents instabilities in the estimates for varying Δt. The figures can be recovered using the Julia script file Empirical_Inst.jl on the GitHub resource (Chang et al., 2020c).



(a) MM, $\Delta t = 300$, 25/06 **(b)** MM, $\Delta t = 300$, 25/06 **(c)** MM, $\Delta t = 300$, 26/06 **(d)** MM, $\Delta t = 300$, 26/06



(e) CT, $\Delta t = 300$, 25/06 **(f)** CT, $\Delta t = 300$, 25/06 **(g)** CT, $\Delta t = 300$, 26/06 **(h)** CT, $\Delta t = 300$, 26/06

Figure 3.9: Here we investigate various values of M ranging from 1 to 23 after accounting for the Epps effect by picking $\Delta t = 300$. The first and second row plots the surface and contour plots of the Malliavin-Mancino and Cuchiero-Teichmann for fixed $\Delta t = 300$ and varying M respectively. The first and second columns are for Tuesday the 25/06/2019; similarly, the third and fourth columns are for Wednesday the 26/06/2019. We see that M builds the harmonics allowing us to achieve higher peaks and lower troughs. The figures can be recovered using the Julia script file Empirical_Inst.jl on the GitHub resource (Chang et al., 2020c).

the 26/06/2019. Since we have no knowledge of what the true correlation structure should be, it is unclear what choice of M provides an appropriate recovery. What is clear is that M adds additional frequencies allowing the estimates to achieve higher peaks and lower troughs.

Without a clear method to pick M, I simply compare the Malliavin-Mancino (solid lines) and Cuchiero-Teichmann (dashed lines) estimates for $M = 10$ and $\Delta t = 300$ in Figure 3.10. Tuesday the 25/06/2019 are in blue while Wednesday the 26/06/2019 are in red. The two estimators obtain similar estimates and capture the same general correlation dynamics. Keep in mind that the estimates of the Cuchiero-Teichmann are not the most reliable because these estimates are for a specific $\Delta t = 300$. The correlation dynamics obtained here can completely change for a different Δt. It remains unclear how one can compare the impact of jumps under the presence of asynchrony with the two estimators. In order to do so, we need to either some resolution for the instantaneous Epps effect, or we need a new estimator that can bypass the time domain like the Malliavin-Mancino estimator and is robust to jumps like the Cuchiero-Teichmann estimator.

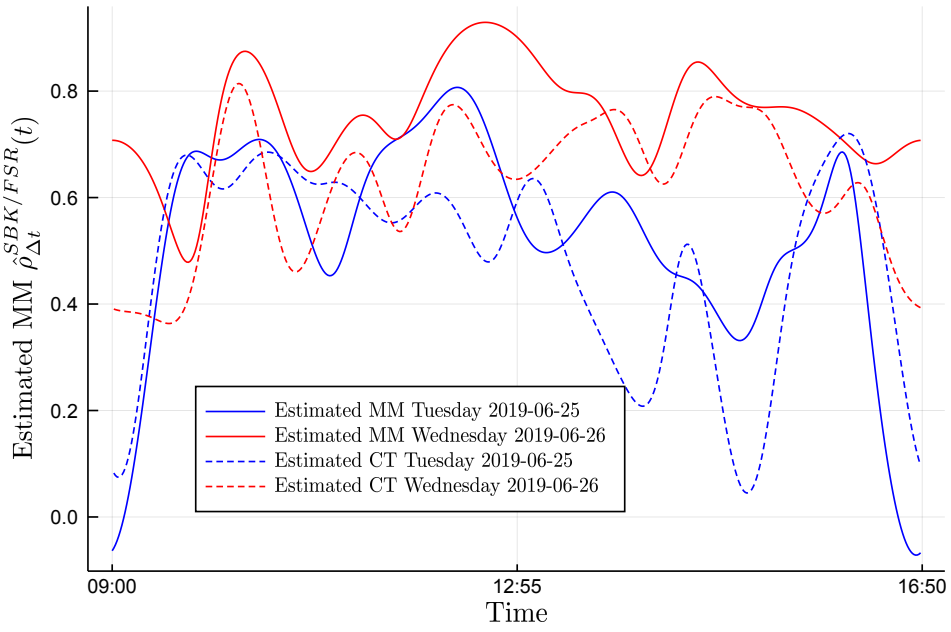


Figure 3.10: Here I compare the Malliavin-Mancino (solid lines) and Cuchiero-Teichmann (dashed lines) instantaneous estimates for $M = 10$ and $\Delta t = 300$. Tuesday the 25/06/2019 are in blue while Wednesday the 26/06/2019 are in red. The two estimators recover similar correlation structures which can vary considerably between the days. The figure can be recovered using the Julia script file Empirical_Inst.jl on the GitHub resource (Chang et al., 2020c).

At this stage, I argue the Malliavin-Mancino estimator is more useful for ultra-high frequency finance because at this scale, the impact of asynchrony outweighs the impact of jumps. Therefore, the ability to deal with asynchrony by bypassing the time domain and thus avoiding the need to impute data in the Malliavin-Mancino estimator makes it more attractive as the resulting estimates are more stable.

3.6 Closing remarks

In this chapter, I compared the Malliavin-Mancino and Cuchiero-Teichmann instantaneous estimator. I found that under the synchronous case, the Cuchiero-Teichmann can better recover the entire volatility matrix in both cases of jumps and no jumps. However, under the asynchronous case, the Cuchiero-Teichmann estimator under estimates the volatility while the Malliavin-Mancino estimates are able to

recover the volatility. Both estimators obtain estimates around zero for the co-volatility due to the Epps effect.

I then demonstrate the impact of the various cutting frequencies M and N in the estimators. I found that M plays a key role in the accuracy of the approximation of the spot estimates and argue that the choice of M should depend on the composition of the true spot parameter of interest. I then demonstrate the instantaneous Epps effect arising from asynchrony under simulation conditions and found that the Malliavin-Mancino estimates produce stable estimates for different Δt because it can bypass the time domain while the Cuchiero-Teichmann estimates are unstable for varying Δt due to the previous tick interpolation. I then provide an *ad hoc* approach to correct for the Epps effect on the instantaneous estimators. The approach is more general than the current choices presented in the literature which are for specific cases of asynchrony. An issue with this approach (and approaches in the current literature) is that it is a naive approach to correct for the Epps effect which can conceal genuine causes of the Epps effect from statistical ones.

Finally, the estimators are applied to Trade and Quote data from the Johannesburg Stock Exchange for two banking equities. The instantaneous Epps effect is demonstrated and once again find that the Malliavin-Mancino estimator obtains stable estimates for various Δt compared to the Cuchiero-Teichmann estimator. Lastly, the intraday correlation dynamics between days for the same asset pair can vary significantly and spot estimates allow us to see these differences.

I argue that the Malliavin-Mancino estimator is the preferred estimator for ultra-high frequency finance where the impact of asynchrony needs to be dealt with first before even considering the impact of jumps. The literature may benefit from a new estimator which is robust to jumps as with the Cuchiero-Teichmann estimator but also able to bypass the time domain like the Malliavin-Mancino estimator. An estimator like such can enable us to compare robustness to jumps under asynchrony. Cuchiero and Teichmann (2015) do mention the consideration of such estimator but also highlight that it is not obvious how this can be achieved (See Remark 3.2 of Cuchiero and Teichmann (2015)).

Initially we were hoping that the Epps effect may offer insight in disentangling jumps from the diffusion process. However, after the insights from this chapter it does not currently seem possible because we do not have the required tools to compare the impact of jumps under asynchronous observations. Even if we derive a new estimator as mentioned above (robust to jumps and able to bypass the time domain), it is not clear to me how we can use the correlations to detect whether or not there are jumps. The answer for this seems to be within the volatility estimates as seen in Figure 3.1. Therefore, in the next chapter I turn my attention to detect if the underlying process is made from events or diffusions.

Chapter 4

Using the Epps effect to detect discrete processes

This chapter is the extended version of the pre-print Chang et al. (2020e) and has been submitted to Quantitative Finance (QF) for review. This chapter is the main chapter of this dissertation tackling the question of whether or not Brownian diffusions are good-enough for high-frequency finance when it comes to recovering empirical correlation dynamics and the Epps effect. In this chapter, I derive the Epps effect arising from asynchrony and provide a refined method to correct the effect. The correction is compared against two existing methods correcting for the Epps effect arising from asynchrony. I then design two experiments to detect if the underlying process is discrete (proxied by a D-type Hawkes process), or if it comes from a Brownian diffusion. This chapter is structured as follows: Section 4.1 I start by motivating the importance of this work. Section 4.2 I derive the Epps effect arising from asynchrony and introduce three methods to correct for the effect. The correction methods are then compared in Section 4.3 for different underlying processes. Section 4.4 I discuss the importance of the residual Epps effect and I design three experiments to discriminate the underlying system. Section 4.5 applies these experiments to banking equities from the Johannesburg Stock Exchange. Finally, Section 4.6 ends the chapter with some closing remarks.

4.1 Motivation

It is important to remember that in the end, models are just are just an abstraction of reality aimed at trying to recover, explain, or predict a behaviour or dynamic. Models have assumptions, whether they are clearly stated or implied; these assumptions often govern how we treat certain empirical phenomenon. For example, in our case, how we treat the Epps effect. Brownian diffusions assume that correlations exist at infinitesimal scales and do not depend on the sampling interval Δt. In contrast, the fine-to-coarse model proposed by Bacry et al. (2013a) using a mutually exciting Hawkes process implicitly assumes that correlations depend on the sampling interval Δt. These two models have very different views on whether the Epps effect is a bias or not. In the case of Brownian diffusions, the Epps effect is a bias that requires correction. On the other hand, in the Hawkes model the Epps effect is a genuine effect that does not need to be corrected. Understanding which model is a better representation of reality is important when it comes to estimating covariance/correlation matrices using high-frequency data. Specifically, if it is even possible to use the abundance of high-frequency data to achieve better estimates compared to low-frequency estimates.

It is difficult to determine whether a model is good or not, but we can determine the usefulness of models in its ability to recover certain stylised facts and obtaining results from simulations that line up with what is seen empirically. In our case, a good model should be able to recover the Epps effect in its entirety which has been deeply problematic in the current literature relying on purely diffusive processes (Brownian diffusions). As I have mentioned in Section 1.2.2, Brownian diffusions can only explain a fraction of the empirically observed Epps effect and thus behavioural arguments have been brought up to defend this stance. My concern with a behavioural argument here is that it may conceal the possibility of an inappropriate underlying representation which can be detrimental when trying to correctly estimate covariance/correlation matrices with high-frequency data. Particularly, if Brownian diffusions are indeed an inappropriate representation then we need to be a lot more careful when considering the impact of time-scales when making decisions predicated on estimates of correlation matrices.

It is therefore important that we figure out what underlying representation is more appropriate when it comes to modelling high-frequency correlation dynamics. The novel aspects of this chapter are three-fold. First, I refine a correction for the Epps effect arising from asynchrony. Second, I demonstrate how seamlessly the Hawkes representation ties together the theory and empiricism. Finally, I design three experiments to test the source of the underlying process. Whether it is a diffusion process or discrete connected events.

4.2 Asynchrony: Compensating for the Epps effect

4.2.1 The Epps effect from asynchrony

Consider a multivariate diffusion process as the underlying representation of the stochastic nature of a log-price of the i^{th} asset at time t. Assuming that the process has stationary increments dX_t^i, let the finite variation over the interval Δt be given by:

$$X_{\Delta t}^i = \int_0^{\Delta t} dX_t^i. \tag{4.1}$$

The infinitesimal lagged correlation between asset i and j are taken to be:

$$\left\langle dX_t^i dX_{t'}^j \right\rangle = c_{t-t'}^{ij} dt dt', \tag{4.2}$$

where $c_{t-t'}^{ij}$ is defined as:

$$c_{t-t'}^{ij} = \begin{cases} \delta_{t-t'} & \text{if } i = j, \\ c\delta_{t-t'} & \text{if } i \neq j, \end{cases} \tag{4.3}$$

and δ_x denotes the delta function (unit impulse) at 0. Meaning δ_x equals 1 when $x = 0$, and equals 0 when $x \neq 0$. We want to examine the behaviour of the covariance process $C_{\Delta t}^{ij} = \left\langle X_{\Delta t}^i X_{\Delta t}^j \right\rangle$ on time-scales Δt. This can be computed using eq. (4.2) as:

$$C_{\Delta t}^{ij} = \int_0^{\Delta t} \int_0^{\Delta t} c_{t-t'}^{ij} dt dt'. \tag{4.4}$$

Following the double integration and unit impulse, it is clear that for the synchronous case we have the variance and covariances as:

$$\begin{aligned} C_{\Delta t}^{ij} &= c\Delta t, \\ C_{\Delta t}^{ii} &= \Delta t. \end{aligned} \tag{4.5}$$

Thus, the correlation of the synchronous process is:

$$\rho_{\Delta t}^{ij} = \frac{C_{\Delta t}^{ij}}{\sqrt{C_{\Delta t}^{ii} \cdot C_{\Delta t}^{jj}}} = c. \tag{4.6}$$

We see that for the synchronous process, both the variance and covariance scale linearly with the sampling interval Δt. Thus the correlation is independent of Δt. Mastromatteo et al. (2011) characterize the Epps effect by saying that the Epps effect is present whenever $\rho_{\Delta t}^{ij}$ depends on Δt, and is absent otherwise.

A property of high-frequency finance is that tick-by-tick trades arrive in an asynchronous fashion following an arrival type representation where the inter-arrivals are not equidistant. The Epps effect arising

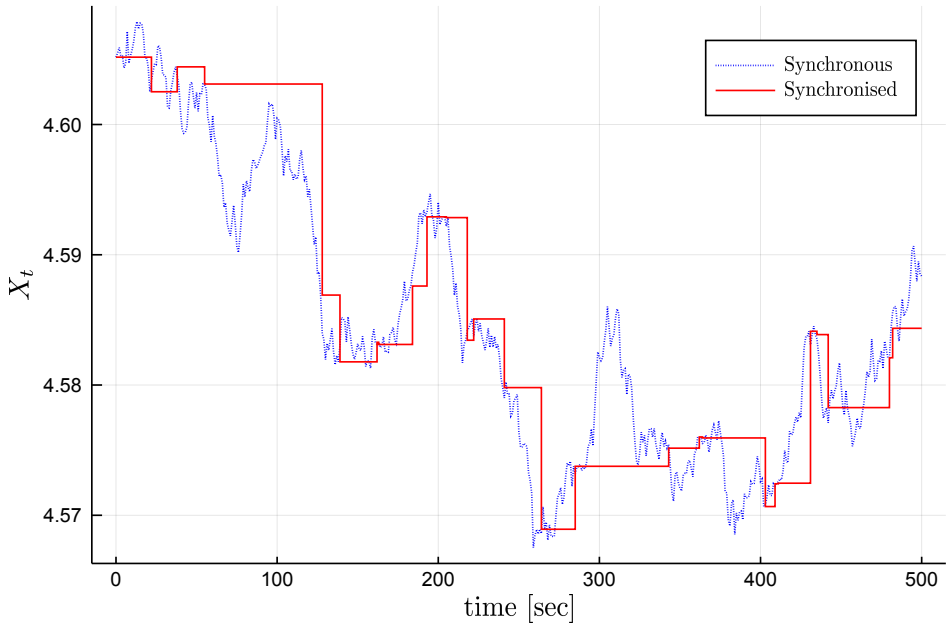


Figure 4.1: A comparison of a realisation of the synchronous process $X^i_{\Delta t}$ (dotted line) and the synchronised process $\tilde{X}^i_{\Delta t}$ (solid line). Here the inter-arrival distribution of U^i follows an exponential with $\lambda = 1/15$. The figure can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

from asynchrony is derived by synchronising the asynchronous observations using previous tick interpolation. To this end, let $U^i = \{t^i_k\}_{k\in\mathbb{Z}}$ be the set of asynchronous arrival times observed between $[0,T]$ for a realisation of the underlying synchronous process X^i_t. With these two components, we can construct the synchronised process:

$$\tilde{X}^i_{\Delta t} = \int_{t^i_1}^{t^i_2} dX^i_t, \tag{4.7}$$

where $t^i_1 = \max\{t^i_k \in U^i | t^i_k \leq 0\}$ and $t^i_2 = \max\{t^i_k \in U^i | t^i_k \leq \Delta t\}$. The resulting synchronised process is piece-wise constant with jumps at $t^i_k \in U^i$. A comparison of the two process $X^i_{\Delta t}$ and $\tilde{X}^i_{\Delta t}$ can be seen in Figure 4.1.

Given U^i and U^j, the covariance of the synchronised process can be defined as:

$$\begin{aligned}
\tilde{C}^{ij}_{\Delta t} &= \mathbb{E}\left[\left\langle \tilde{X}^i_{\Delta t}\tilde{X}^j_{\Delta t}\right\rangle \middle| \begin{matrix} U^i \\ U^j \end{matrix}\right] \\
&= \mathbb{E}\left[\int_{t^i_1}^{t^i_2}\int_{t^j_1}^{t^j_2} c^{ij}_{t-t'} dt dt' \middle| \begin{matrix} U^i \\ U^j \end{matrix}\right] \\
&= c \cdot \mathbb{E}\left[\left| \left[t^i_1, t^i_2\right] \cap \left[t^j_1, t^j_2\right] \right|\right].
\end{aligned} \tag{4.8}$$

Here $\mathbb{E}[\cdot]$ is the expectation of the sampling process. Similarly, the variance is defined as:

$$\tilde{C}^{ii}_{\Delta t} = \mathbb{E}\left[\left| \left[t^i_1, t^i_2\right] \cap \left[t^i_1, t^i_2\right] \right|\right]. \tag{4.9}$$

Given U^i and U^j, we can estimate the expectation of the overlap of the sampling process at a particular discretisation size Δt by defining a new variable $\gamma_i(t) = \max\{t^i_k : t^i_k \leq t\}$ for $t \in [0,T]$. The expectation can then be estimated as:

$$\hat{\kappa}^{ij}_{\Delta t} = \mathbb{E}\left[\left| \left[\gamma_i(t-\Delta t), \gamma_i(t)\right] \cap \left[\gamma_j(t-\Delta t), \gamma_j(t)\right] \right|\right]. \tag{4.10}$$

Thus, the correlation of the synchronised process is then given by:

$$\tilde{\rho}^{ij}_{\Delta t} = c \cdot \frac{\hat{\kappa}^{ij}_{\Delta t}}{\sqrt{\hat{\kappa}^{ii}_{\Delta t} \cdot \hat{\kappa}^{jj}_{\Delta t}}}. \tag{4.11}$$

This depends on Δt and therefore the Epps effect is present.

The intuition is best explained by Münnix et al. (2011) and in eq. (4.8). When the intervals $\left[t^i_1, t^i_2\right]$ and $\left[t^j_1, t^j_2\right]$ overlap, the correlation from the synchronous process $X_{\Delta t}$ is being extracted, while the non-overlapping intervals are uncorrelated. Therefore, the correlation of the synchronised process will be distorted by the non-overlapping intervals.

This approach has been previously investigated by Tóth and Kertész (2007), Mastromatteo et al. (2011) and Münnix et al. (2011). The key difference between the work of Tóth and Kertész (2007) and Mastromatteo et al. (2011) is that a specific distribution for the sampling process with stationary increments is chosen beforehand (usually a homogeneous Poisson), whereas here we only require that the sampling process have stationary increments. This can be useful because characterising the Epps effect with a specific *a priori* distribution requires knowledge about the distribution of $\mathscr{W}_{\Delta t}$ defined as:

$$\mathscr{W}_{\Delta t} = \min\left\{\gamma_i(t), \gamma_j(t)\right\} - \max\left\{\gamma_i(t-\Delta t), \gamma_j(t-\Delta t)\right\}, \tag{4.12}$$

This is not always easy to obtain. Using the Poisson example, to obtain eq. (4.12) we need to find the probability distribution of the minimum and maximum of two independently and exponentially distributed variables.[1] Let ν and η be such. We then have (Tóth and Kertész, 2007):

$$\begin{aligned} \mathbb{P}\left(\min\{\nu,\eta\} \in (x, x+dx)\right) &= 2\lambda e^{-2\lambda x} dx, \\ \mathbb{P}\left(\max\{\nu,\eta\} \in (x, x+dx)\right) &= 2\lambda \left(e^{-\lambda x} - e^{-2\lambda x}\right) dx. \end{aligned} \tag{4.13}$$

Using eqs. (4.11) and (4.13) we get:

$$\tilde{\rho}^{ij}_{\Delta t} = c\left(1 + \frac{1}{\lambda \Delta t}\left(e^{-\lambda \Delta t} - 1\right)\right). \tag{4.14}$$

The difference here and the work of Münnix et al. (2011) is that they apply the correction while estimating the correlation. They do so by compensating for the non-overlapping portions for return intervals that overlap. Here I separate the estimation of the correlation $\tilde{\rho}^{ij}_{\Delta t}$ and the overlapping intervals $\hat{\kappa}^{ij}_{\Delta t}$. This makes the correction simpler to compute.

4.2.2 Correcting for an Epps effect from asynchrony

Here the Epps effect arises from sampling; we can analytically quantify the deviation and the deviation can be compensated for accordingly. The Epps effect arising from asynchrony is a statistical bias causing the correlation to deviate from the "true" correlation of the synchronous process. I will discuss a few methods to correct for this effect. A key assumption underlying the correction is that the correlation from the synchronous process is the ground truth we are interested in and the observed samples are merely realisations from this underlying process, *i.e.* assuming there exists an underlying synchronous process generating the data, we are interested in recovering the correlation from the synchronous process using the asynchronous observations.

[1] The independence is not always exploitable, for example when using Hawkes sampling. Meaning that finding eq. (4.12) is non-trivial.

Remark 4.2.1 *As a follow up to Remark 2.2.6, this viewpoint is different to the data-informed perspective where in fact eq. (4.11) is the ground truth as it is the exact correlation from the observables. However, here it is more useful here to treat the bias in the usual traditional statistical sense. We will see that this actually allows us to detect when the underlying process is from a diffusion process or from connected events as this allows us to disentangle genuine and statistical sources of the Epps effect.*

Here I use the Malliavin and Mancino (2002, 2009) integrated estimator using non-uniform fast Fourier methods from Chapter 2 to efficiently measure the correlation $\tilde{\rho}^{ij}_{\Delta t}$ from the observables of the asynchronous process. In this chapter, I use the Malliavin-Mancino estimator differently. In Chapters 2 and 3 we saw its ability to deal with asynchrony, however we also saw in Figure 2.11 that even with the Dirichlet representation the estimator does not quite recover the theoretical Epps effect. This is because of the difference in how asynchrony is dealt with between the Malliavin-Mancino estimator and the previous tick interpolation for which the Epps effect is derived with (See Remark 2.4.1). Therefore, rather than exploiting its ability to deal with asynchrony, I use it as the Realised Volatility (RV) estimator by synchronising the observations using the previous tick interpolation. In my Honours project, we showed that when the Malliavin-Mancino estimator is applied to synchronous data with the Nyquist frequency, it recovers the same estimates as the Realised Volatility estimator. I make this choice for simplicity, so that we do not need to consider which method of dealing with asynchrony is more effective (although Chapter 3 we saw bypassing the time domain has some advantages); additionally, the derivation uses the previous tick interpolation thus it is sensible to also use the previous tick interpolation when estimating the correlations.

Concretely, we have:

$$\tilde{\Sigma}^{ij}_{T,\Delta t} = \int_0^T \tilde{\Sigma}^{ij}(t)dt = \frac{1}{2N+1} \sum_{\substack{|s|\leq N \\ \ell=1,k=1}}^{\lfloor T/\Delta t \rfloor} e^{\mathrm{i}s(\ell-k)\Delta t} \left(\tilde{X}^i_{\ell\Delta t} - \tilde{X}^i_{(\ell-1)\Delta t}\right)\left(\tilde{X}^j_{k\Delta t} - \tilde{X}^j_{(k-1)\Delta t}\right), \tag{4.15}$$

where $N = \lfloor \frac{T}{2\Delta t} \rfloor$ is the Nyquist frequency and $\tilde{X}^i_t = X^i_{\gamma_i(t)}$. Thus the measured correlation is:

$$\tilde{\rho}^{ij}_{\Delta t} = \frac{\tilde{\Sigma}^{ij}_{T,\Delta t}}{\sqrt{\tilde{\Sigma}^{ii}_{T,\Delta t} \cdot \tilde{\Sigma}^{jj}_{T,\Delta t}}}. \tag{4.16}$$

Arrival time (overlap) correction

The first correction method is to correct for the Epps effect directly from the characterisation derived in eq. (4.11). Thus, to recover the correlation of the true synchronous process we compute:

$$\rho^{ij}_{\Delta t} = \tilde{\rho}^{ij}_{\Delta t} \cdot \frac{\sqrt{\hat{\kappa}^{ii}_{\Delta t} \cdot \hat{\kappa}^{jj}_{\Delta t}}}{\hat{\kappa}^{ij}_{\Delta t}}, \tag{4.17}$$

where $\hat{\kappa}^{ij}_{\Delta t}$ can be directly estimated using U^i and U^j with discretisation size Δt. This defines the *overlap correction*.

Remark 4.2.2 *Notice the notational difference for the correlation estimates. Here $\tilde{\rho}^{ij}_{\Delta t}$ are the correlation estimates for the synchronised asynchronous process and $\rho^{ij}_{\Delta t}$ is the correlation estimate of the synchronous process.*

Require:
1. Δt: the discretisation interval.
2. T: time horizon of investigation.
3. $U^i = \{t_k^i\}_{k\in\mathbb{Z}}$: the set of asynchronous arrivals for asset i.
4. $U^j = \{t_k^j\}_{k\in\mathbb{Z}}$: the set of asynchronous arrivals for asset j.

Set: $n = \lfloor T/\Delta t \rfloor$
for k = 0 to n **do**
  Set: $\gamma_i(k\Delta t) = \max\{t_k^i : t_k^i \leq k\Delta t\}$.
  Set: $\gamma_j(k\Delta t) = \max\{t_k^j : t_k^j \leq k\Delta t\}$.
end for
for k = 1 to n **do**
  Set: $O_k^{ii} = |[\gamma_i((k-1)\Delta t), \gamma_i(k\Delta t)] \cap [\gamma_i((k-1)\Delta t), \gamma_i(k\Delta t)]|$.
  Set: $O_k^{jj} = \left|\left[\gamma_j((k-1)\Delta t), \gamma_j(k\Delta t)\right] \cap \left[\gamma_j((k-1)\Delta t), \gamma_j(k\Delta t)\right]\right|$.
  Set: $O_k^{ij} = \left|[\gamma_i((k-1)\Delta t), \gamma_i(k\Delta t)] \cap \left[\gamma_j((k-1)\Delta t), \gamma_j(k\Delta t)\right]\right|$.
end for
Set: $\hat{\kappa}_{\Delta t}^{ij} = \bar{O}^{ij}$; $\hat{\kappa}_{\Delta t}^{ii} = \bar{O}^{ii}$; $\hat{\kappa}_{\Delta t}^{jj} = \bar{O}^{jj}$.
return $(\hat{\kappa}_{\Delta t}^{ij}, \hat{\kappa}_{\Delta t}^{ii}, \hat{\kappa}_{\Delta t}^{jj})$

Algorithm 10: The Overlap correction Algorithm computes the overlap factor $\hat{\kappa}_{\Delta t}^{ij}$, $i, j = 1, 2$ for our arrival time correction in eq. (4.17). The Julia implementation can be found as an auxiliary function in the script file EppsCorrection.jl in the GitHub resource (Chang et al., 2020c).

Flat trade correction

The second correction method for the Epps effect is given by Buccheri et al. (2019). They lay down three assumptions.

Assumption 4.2.1 *Efficient price process:*
First assume that there exists two real-valued logarithmic efficient price processes, X^ℓ which are Brownian semi-martingales for $t \in [0, T]$ and satisfy:

$$dX_t^\ell = \mu_t^\ell dt + \sigma_t^\ell dW_t \quad \ell = i, j,$$

where W_t^ℓ is a standard Brownian motion, μ_t^ℓ and σ_t^ℓ are predictable and adapted.
Moreover, $d\left\langle W^i, W^j \right\rangle_t = \rho_t^{ij} dt$ and lastly σ_t^ℓ has semi-martingale dynamics.

Assumption 4.2.1 are standard assumptions for continuous-time stochastic processes. Moreover, they assume that the process is observed at $n = \lfloor \frac{T}{\Delta t} \rfloor + 1$ non-random times equispaced over $[0, T]$, *i.e.* $0 < t_{0,n} < t_{1,n} < ... < t_{n,n} = T$ where $\Delta t = t_{j,n} - t_{j-1,n}$ for $j \geq 1$.

Assumption 4.2.2 *Observed process:*
The observed price process is such that each price at each grid point has a Bernoulli random variable governing whether that price point is newly observed from the efficient price process, or retains the price point at the previous grid point. Meaning the observables are characterised as:

$$\tilde{X}_{t_{j,n}}^\ell = X_{t_{j,n}}^\ell \left(1 - B_{j,n}^\ell\right) + \tilde{X}_{t_{j-1,n}}^\ell B_{j,n}^\ell, \tag{4.18}$$

where $B_{j,n}^\ell$ for $\ell = 1, 2$ are pairwise-independent triangular arrays of i.i.d. Bernoulli variables such that

$$p_{\ell,n} = \mathbb{P}\left[B_{j,n}^\ell = 1\right] = \mathbb{E}\left[B_{j,n}^i\right] \to p_\ell,$$

and

$$n\left(p_{\ell,n}-p_{\ell}\right)\to 0,\quad as\quad n\to\infty.$$

Assumption 4.2.2 is slightly problematic in the sense that the validity of this assumption depends on the size of intervals Δt investigated. This assumption is better suited for small intervals where there are some bins with no new observables, but not suited for ultra-small intervals where there are lots of repeated bins with no new observables. Essentially, this assumption is better suited for a missing data representation of asynchrony, rather than the arrival time representation. This leads to their last assumption.

Assumption 4.2.3 *Dynamics of flat trading:*
For all $j=1,...,n$ and $\ell=1,2$. Let the number of consecutive flat trades for the ℓ^{th} asset before time $t_{j,n}$ be:

$$K_{j,n}^{\ell}=\min\left\{k\in\{0,...,j\}|B_{j,n}^{\ell}=1,B_{j-1,n}^{\ell}=1,...,B_{j-k+1,n}^{\ell}=1,B_{j-k,n}^{\ell}=0\right\}.$$

They assume that the maximum $K_n^{\ell}=\max_{j=1,...,n}K_{j,n}^{\ell}$ is such that

$$\frac{K_n^{\ell}\log(n)}{n}\xrightarrow{p}0,\quad as\quad n\to\infty.$$

Assumption 4.2.3 simply means that the maximum number of repeated trades is not large relative to the number of data points. This alludes to the fact that their correction is not well suited for tick-by-tick trade data. Nonetheless, I include the comparison as it is one of the few corrections of the Epps effect provided in the current literature.

Under these assumptions, Buccheri et al. (2019) show that the Realised Covariance estimator is biased.

Theorem 4.2.1 *Theorem 3.1 of Buccheri et al. (2019):*
Let the efficient price dynamics X^{ℓ} follow Assumption 4.2.1, the observed price dynamics $\tilde{X}^{\ell}$ follow Assumption 4.2.2, and the triangular array of Bernoulli variables follow Assumption 4.2.3.
Then the Realised Covariance estimator (RC) defined as:

$$RC_{\Delta t}=\sum_{k=1}^{\lfloor T/\Delta t\rfloor}\left(\tilde{X}_{k\Delta t}^{i}-\tilde{X}_{(k-1)\Delta t}^{i}\right)\left(\tilde{X}_{k\Delta t}^{j}-\tilde{X}_{(k-1)\Delta t}^{j}\right),$$

has the following limiting in probability:

$$RC_{\Delta t}\xrightarrow{p}\frac{(1-p_i)(1-p_j)}{(1-p_ip_j)}\int_0^T\sigma_s^i\sigma_s^j\rho_s^{ij}ds,\quad as\quad n\to\infty.$$

Therefore, Buccheri et al. (2019) provide a correction given as:

$$\rho_{\Delta t}^{ij}=\tilde{\rho}_{\Delta t}^{ij}\cdot\frac{\left(1-\hat{p}_{\Delta t,i}\hat{p}_{\Delta t,j}\right)}{\left(1-\hat{p}_{\Delta t,i}\right)\left(1-\hat{p}_{\Delta t,j}\right)},\tag{4.19}$$

where $\hat{p}_{\Delta t,i}$ is the estimate of p_i, computed as:

$$\hat{p}_{\Delta t,i}=\frac{1}{\lfloor T/\Delta t\rfloor}\sum_{j=1}^{\lfloor T/\Delta t\rfloor}\mathbb{1}_{\left\{\tilde{X}_{j\Delta t}^{i}-\tilde{X}_{(j-1)\Delta t}^{i}\right\}=0},\tag{4.20}$$

where $\mathbb{1}_A$ is an indicator function on event A. Convergence of $\hat{p}_{\Delta t,i}\xrightarrow{p}p_i$ is then shown to hold.

Require:
1. $\tilde{\mathbf{X}}$: (n x D) matrix of synchronised log-prices using the previous tick interpolation with discretisation size Δt.

Compute: δ the (n-1 x D) differenced matrix of log-returns, where $\delta^i_{k\Delta t} = \tilde{X}^i_{k\Delta t} - \tilde{X}^i_{(k-1)\Delta t}$ for the i^{th} asset.
for i = 1 to D **do**
  Set: count = 0.
  for k = 1 to n-1 **do**
    if $\delta^i_{k\Delta t} = 0$ **then**
      Set: count = count + 1.
    end if
  end for
  Set: $\hat{p}_{\Delta t,i} = \frac{\text{count}}{n-1}$.
end for
return $(\hat{\mathbf{p}}_{\Delta t})$

Algorithm 11: The Probability of Flat Trading Algorithm computes the estimates $\hat{p}_{\Delta t,i}$ for the Buccheri et al. (2019) correction in eq. (4.19). The Julia implementation can be found as an auxiliary function in the script file EppsCorrection.jl in the GitHub resource (Chang et al., 2020c).

Hayashi-Yoshida (baseline) correction

The last correction method for the Epps effect is given by Hayashi and Yoshida (2005). The estimator corrects for asynchrony by using a cumulative covariance estimator that allows for multiple contributions. First, Hayashi and Yoshida (2005) assume that the price process follows the standard semimartingale dynamics in Assumption 4.2.1. Moreover, let $\Pi^i = (I^i)_{i=1,2,\ldots}$ and $\Pi^j = (I^j)_{j=1,2,\ldots}$ be the sets of random intervals which partition $[0,T]$ dependent on the arrival dynamics U^i and U^j. Hayashi and Yoshida (2005) place some assumptions on the dynamics of $\Pi := (\Pi^i, \Pi^j)$, specifically:

Assumption 4.2.4 *Assume that* Π *satisfy the following:*

i.) (I^i) *and* (I^j) *are independent of the price dynamics, and*

ii.) As $n \to \infty$, $\max_i |I^i| \vee \max_j |J^j| \to 0$,

where $|I|$ *is the length of the interval* I.

Let the cumulative covariance estimator be defined as:

$$\hat{\Sigma}^{ij}_T = \int_0^T \Sigma^{ij}(t)dt = \sum_{\ell=1}^{\#U^i} \sum_{k=1}^{\#U^j} \left(X^i_{t^i_\ell} - X^i_{t^i_{\ell-1}}\right)\left(X^j_{t^j_k} - X^j_{t^j_{k-1}}\right) \mathbb{1}_{\left\{\left(t^i_{\ell-1}, t^i_\ell\right] \cap \left(t^j_{k-1}, t^j_k\right] \neq \emptyset\right\}}. \tag{4.21}$$

Here $\#U^i$ denotes the cardinality of set U^i. This leads to the main theorem in their paper.

Theorem 4.2.2 *Theorem 3.1 of Hayashi and Yoshida (2005):*
Suppose Assumption 4.2.4 holds

i.) If $\sup_{0\le t\le T} |\mu^\ell_t| \in L^4$ *for* $\ell = 1,2$, *then* $\hat{\Sigma}^{ij}_T \to \theta$ *as* $n \to \infty$.

ii.) If $\sup_{0\le t\le T} |\mu^\ell_t| < \infty$ *almost surely for* $\ell = 1,2$, *then* $\hat{\Sigma}^{ij}_T$ *is consistent for* θ, *that is,* $\hat{\Sigma}^{ij}_T \xrightarrow{p} \theta$ *as* $n \to \infty$.

Here $\theta = \int_0^T \sigma^i_s \sigma^j_s \rho^{ij}_s ds$. Therefore the correlation estimate is given by:

$$\rho^{ij}_{\Delta t} = \frac{\hat{\Sigma}^{ij}_T}{\sqrt{\hat{\Sigma}^{ii}_T \cdot \hat{\Sigma}^{jj}_T}}, \tag{4.22}$$

which is independent of Δt. This means that the Hayashi-Yoshida estimator cannot be used to investigate the correlation at various time-scales. It can only recover the correlation of the underlying synchronous process under the presence of asynchrony. For this reason I will use it as a baseline for comparison.

Although I do not deal with lead-lag, I highlight that the Hayashi-Yoshida estimator cannot account for lead-lag as it implicitly assumes that the correlation between two assets do not extend beyond the interval with full or partial overlap (Griffin and Oomen, 2011).

The implementation of the Hayashi-Yoshida estimator relies on the Kanatani weight matrix (Kanatani, 2004) which I used in my Honours project (Chang et al., 2019a). Taking a closer took at eq. (4.21), notice that we can write the indicator function with $w_{\ell k}$, given by:

$$w_{\ell k} = \begin{cases} 1 & \text{if } \left(t^i_{\ell-1}, t^i_{\ell}\right] \cap \left(t^j_{k-1}, t^j_{k}\right] \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{4.23}$$

Therefore, we can rewrite eq. (4.21) in matrix notation:

$$\begin{aligned} \hat{\Sigma}^{ij}_T &= \sum_{\ell=1}^{\#U^i} \sum_{k=1}^{\#U^j} \left(X^i_{t^i_\ell} - X^i_{t^i_{\ell-1}}\right)\left(X^j_{t^j_k} - X^j_{t^j_{k-1}}\right) w_{\ell k} \\ &= (\delta^i)^T \mathbf{W} (\delta^j), \end{aligned} \tag{4.24}$$

and δ^i is the differenced vector of log-returns, where $\delta^i_k = X^i_{t^i_k} - X^i_{t^i_{k-1}}$ for the i^{th} asset. When $i = j$, $\mathbf{W} = \mathbf{I}$ becomes the identity matrix.

Require:
1. $U^i = \{t^i_k\}_{k\in\mathbb{Z}}$: the set of asynchronous arrivals for asset i.
2. $U^j = \{t^j_k\}_{k\in\mathbb{Z}}$: the set of asynchronous arrivals for asset j.

for ℓ = 1 to $\#U^i$ **do**
  for k = 1 to $\#U^j$ **do**
    if $(t^i_{\ell-1}, t^i_{\ell}] \cap (t^j_{k-1}, t^j_k] \neq \emptyset$ **then**
      Set: $w_{\ell k} = 1$
    else
      Set: $w_{\ell k} = 0$
    end if
  end for
end for
return W

Algorithm 12: The Kanatani Weight matrix Algorithm computes a matrix $\mathbf{W}$ of 1's and 0's indicating when contributions to eq. (4.21) should be included using the implementation by Kanatani (2004). The Julia implementation can be found as an auxiliary function in HYcorr.jl in the GitHub resource (Chang et al., 2020c).

Require:

1. $U^i = \{t_k^i\}_{k\in\mathbb{Z}}$: the set of asynchronous arrivals for asset i.
2. $U^j = \{t_k^j\}_{k\in\mathbb{Z}}$: the set of asynchronous arrivals for asset j.
3. $X_t^i, t \in U^i$: the set of asynchronous log-price observations for asset i.
4. $X_t^j, t \in U^j$: the set of asynchronous log-price observations for asset j.

Compute: δ^i, the differenced vector of log-returns, where $\delta_k^i = X^i_{t_k^i} - X^i_{t_{k-1}^i}$ for the i^{th} asset.
Compute: $\mathbf{W}$, the Kanatani weight matrix using Algorithm 12.
Compute: $\Sigma^{ij} = \Sigma^{ji} = (\delta^i)^T \mathbf{W} (\delta^j)$.

Compute: $\Sigma^{ii} = (\delta^i)^T(\delta^i)$; $\Sigma^{jj} = (\delta^j)^T(\delta^j)$.
Compute: $\rho^{ij} = \frac{\Sigma^{ij}}{\sqrt{\Sigma^{ii}}\sqrt{\Sigma^{jj}}}$
return $(\mathbf{\Sigma}, \rho)$

Algorithm 13: The Hayashi Yoshida Algorithm computes the Hayashi-Yoshida estimate given by eq. (4.22) using the Kanatani weight matrix (Chang et al., 2019a). The Julia implementation can be found in HYcorr.jl in the GitHub resource (Chang et al., 2020c).

4.3 Simulation experiments

The experiments focus on comparing the correction methods of the Epps effect arising from asynchrony from two types of sampling. The first from a homogeneous Poisson process with exponential inter-arrivals (henceforth referred to as Poisson sampling), and the second from a 2-dimensional Hawkes process (henceforth referred to as Hawkes sampling). The sampling methods are performed on three types of price models:

i.) the standard Brownian diffusion price model,

ii.) the jump-diffusion price model, and

iii.) the Hawkes price model generated from bottom-up events, which in the limit converges in distribution towards the standard diffusion processes as shown by Bacry et al. (2013a,b).

The experiments in this section will be conducted using simulated paths where T is 20 hours (72,000 seconds) and correlations will be investigated at ultra-small intervals where Δt is measured in seconds. All the experiments here simulate a single realisation of the price paths. This is then re-sampled using either a Poisson or Hawkes sampling scheme for 100 replications. The figures plot the mean estimate at each Δt over 100 replications for the various correlation estimate/correction estimate. The error ribbons contain 95% of the estimates at each Δt from the replications computed using the Student t-distribution with 99 degrees of freedom and the standard deviation of the estimates between the replications at each Δt. Only in Figure 4.5, do I simulate 100 realisations of the price paths from the Hawkes price model given in eq. (4.29). The figure plots the mean correlation estimate and the error ribbons contain 95% of the estimates at each Δt over the 100 replications. A detailed discussion of the Hawkes process can be found in Appendix B.

4.3.1 Brownian price model

Let us consider a standard diffusive Brownian log-price model satisfying the following SDE:

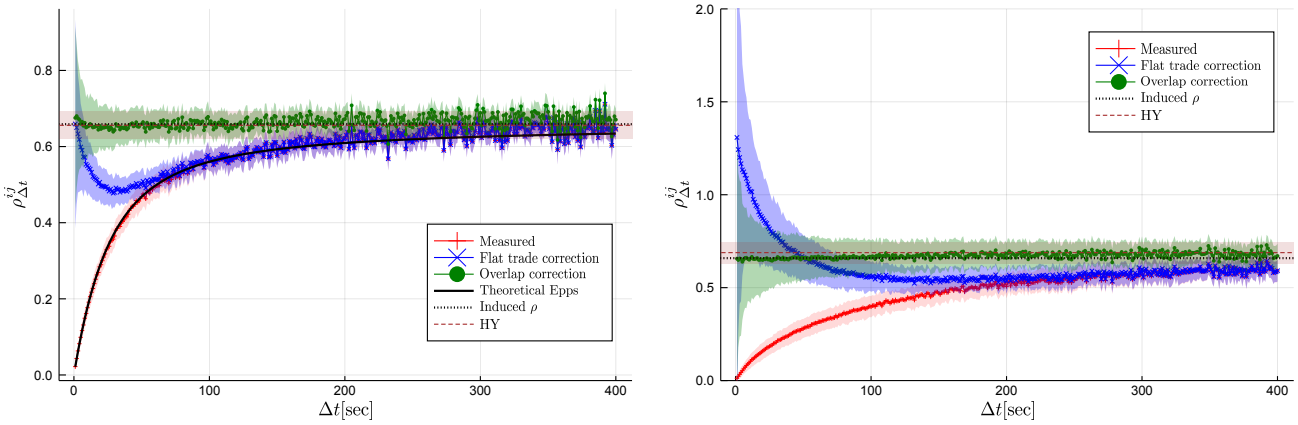
$$\begin{aligned} dX_t^1 &= \left(\mu_1 - \frac{\sigma_1^2}{2}\right) dt + \sigma_1 dW_t^1, \\ dX_t^2 &= \left(\mu_2 - \frac{\sigma_2^2}{2}\right) dt + \sigma_2 dW_t^2, \end{aligned} \tag{4.25}$$

where the infinitesimal correlation between dW_t^1 and dW_t^2 is ρ^{12} set to be approximately 0.65.[2] The process eq. (4.25) is simulated using Algorithm 19 with the discretisation size corresponding to one second intervals. The parameters used are $\mu_1 = \mu_2 = 0.01$, $\sigma_1^2 = 0.1$ and $\sigma_2^2 = 0.2$ (the parameters are given in daily intervals). The first 500 seconds of this process is shown in Figure 4.1.

This process is sampled using a Poisson process with a mean inter-arrival of $1/\lambda = 15$ seconds and a 2-dimensional Hawkes process with $\lambda_0^1 = \lambda_0^2 = 0.015$ and $\mathbf{\Phi}$ taking the form:

$$\mathbf{\Phi} = \begin{pmatrix} 0 & \phi^{(s)} \\ \phi^{(s)} & 0 \end{pmatrix}, \tag{4.26}$$

where $\phi^{(s)} = 0.023 e^{-0.11t} \mathbb{1}_{t \in \mathbb{R}^+}$.



(a) Brownian price model with Poisson sampling

(b) Brownian price model with Hawkes sampling

Figure 4.2: A Brownian price model with (a) Poisson sampling, and (b) Hawkes sampling is presented: (+) is the measured correlation from the asynchronous process (See eq. (4.16)), (×) is the flat trade correction (See eq. (4.19)), and (○) is the overlap correction (See eq. (4.17)). The thick solid line is the plot of eq. (4.14). The horizontal dotted line is the induced correlation of the synchronous system with $\rho \approx 0.65$. Lastly, the horizontal dashed line is the Hayashi-Yoshida estimate (See eq. (4.22)). Here we see that the overlap correction and Hayashi-Yoshida estimate correctly recover the underlying correlation, while the flat trade correction does not. The figures can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

Figure 4.2 compares the correction methods for (a) Poisson and (b) Hawkes sampling on a Brownian price model. Here the overlap correction (○) and the Hayashi-Yoshida estimator (horizontal dashed line) correctly recovers the underlying correlation of the synchronous process. On the other hand, the flat trade correction (×) does not perform well at recovering the induced underlying correlation. The flat trade correction does mitigate some of the Epps effect at ultra-high frequencies for the Poisson sampling,

[2] The value is chosen for suitable comparison against the limiting correlation of the Hawkes price model in Section 4.3.3.

but it is problematic under Hawkes sampling where the correction returns estimates outside the feasible range for correlations.[3] The reason for this can be attributed to the violation of Assumption 4.2.3 since the specification of the Hawkes sampling in eq. (4.26) results in long inter-arrivals. These long inter-arrivals under small sampling intervals Δt leads to extended periods of flat trading which makes the estimates in eq. (4.20) closer to 1. This leads to an over-compensation in the correction.

The theoretical Epps effect arising from asynchrony (thick solid line) is plotted for the Poisson sampling since the distribution of $\mathscr{W}_{\Delta t}$ can be recovered in this case. Thus we know the theoretical Epps effect is given by eq. (4.14). This theoretical Epps effect is not plotted for the Hawkes sampling as obtaining the distribution of $\mathscr{W}_{\Delta t}$ is not simple in this case. The main feature behind my correction method is that we do not need to obtain the distribution of $\mathscr{W}_{\Delta t}$ since the correction can be directly estimated using eq. (4.10) given U^i and U^j.

Remark 4.3.1 *Here I only use a Hawkes sampling process that mutually excites in eq. (4.26), but I highlight that similar results as Figure 4.2b is achieved using different variations of the Hawkes sampling process. Such as one that contains both self and mutual excitation.*

4.3.2 Jump diffusion model

Let us consider the case where the diffusion is dressed with jumps to determine if jumps will affect our ability to detect diffusions against discrete connected events. To this end, consider a Merton model where the prices P_t^i satisfy the following SDE:

$$\begin{aligned} \frac{dP_t^1}{P_{t^-}^1} &= \mu_1 dt + \sigma_1 dW_t^1 + dJ_t^1, \\ \frac{dP_t^2}{P_{t^-}^2} &= \mu_2 dt + \sigma_2 dW_t^2 + dJ_t^2, \end{aligned} \tag{4.27}$$

where the infinitesimal correlation between dW_t^1 and dW_t^2 is ρ^{12} set to be approximately 0.65, and J_t^i are independent of W_t^i with piece-wise constant paths. J_t^i is defined as

$$J_t^i = \sum_{j=1}^{N_i(t)} \left(Y_j - 1\right), \tag{4.28}$$

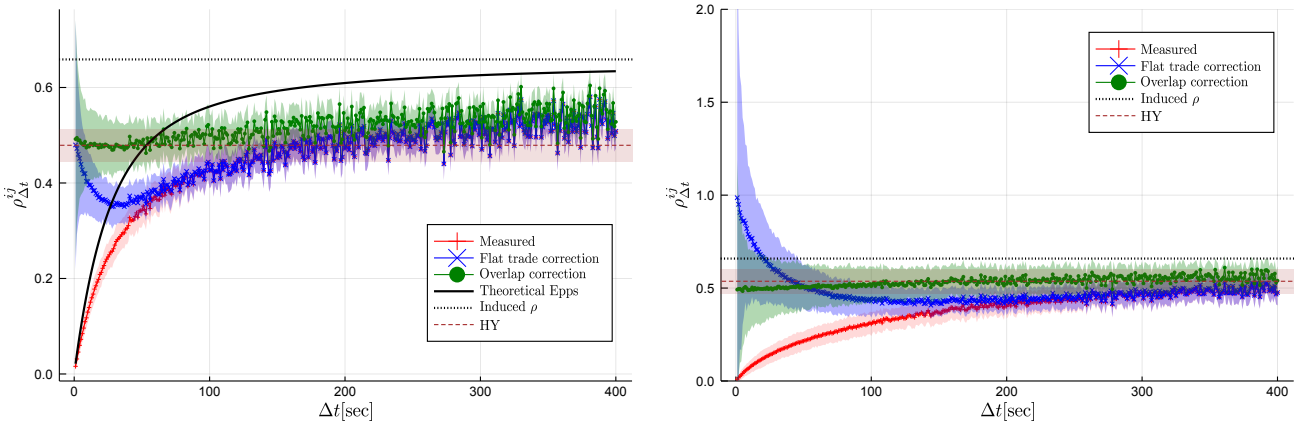
where $N_i(t)$ is a Poisson process with rate λ_i, $Y_j \sim LN(a_i, b_i)$ i.i.d and also independent of $N_i(t)$. The parameters used are $\mu_1 = \mu_2 = 0.01$, $\sigma_1^2 = 0.1$, $\sigma_2^2 = 0.2$, $a_1 = a_2 = 0$, $b_1 = b_2 = 0.001$ and $\lambda_1 = \lambda_2 = 0.2$.

Following the experiment as before, Figure 4.3 samples the Merton price model with (a) a Poisson process with a mean inter-arrival of $1/\lambda = 15$ seconds and (b) a 2-dimensional Hawkes process taking the form of eq. (4.26) with parameters $(\lambda_0^1 = \lambda_0^2, \alpha^{(s)}, \beta) = (0.015, 0.023, 0.11)$.

Figure 4.3 we see that the saturation level does not recover the induced correlation. This is consistent with what we found in my Honours project where jumps reduce the saturation level. In Chapter 3 we saw that this was because jumps cause spikes in the volatility estimates (for estimators which are not robust to jumps) which results in a larger normalisation factor and therefore a lower saturation level. We see that Figure 4.3 looks like Figure 4.2 with the exception that the saturation level does not recover the induced correlation. This is because ultimately with jump diffusions, the underlying component is still a diffusion process and therefore when the correction methods are applied we can ameliorate the

[3] None of the correction methods guarantee that the corrected estimates for the correlation lie within $[-1, 1]$.

Epps effect arising from asynchronous samples. Concretely, the overlap correction ($\circ$) and the Hayashi-Yoshida (horizontal dashed line) recover the saturation level, while the flat trade correction ($\times$) does not correctly recover the saturation level for Poisson sampling and over corrects for the Hawkes sampling. The ability to recover the saturation level here means that the Epps effect has been corrected. The lower saturation is a result from jumps which is a separate issue that can be solved with estimators robust to jumps such as those mentioned in Section 3.1.



(a) Merton price model with Poisson sampling

(b) Merton price model with Hawkes sampling

Figure 4.3: A Merton price model with (a) Poisson sampling, and (b) Hawkes sampling is presented: ($+$) is the measured correlation from the asynchronous process (See eq. (4.16)), ($\times$) is the flat trade correction (See eq. (4.19)), and ($\circ$) is the overlap correction (See eq. (4.17)). The thick solid line is the plot of eq. (4.14). The horizontal dotted line is the induced correlation of the synchronous system with $\rho \approx 0.65$. Lastly, the horizontal dashed line is the Hayashi-Yoshida estimate (See eq. (4.22)). Here we see that the overlap correction and Hayashi-Yoshida estimate recover the saturation level, while the flat trade correction does not. The figures can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

This means that the jumps do not affect our ability to correct for the Epps effect because of the underlying diffusion component. This means that jumps will not affect our ability to discriminate between diffusions and connected events.

4.3.3 Hawkes price model

Let us consider an alternative price model as opposed to the widely used diffusion processes. The price model considered is the fine-to-coarse model introduced by Bacry et al. (2013a) constructed using a multivariate Hawkes process. The price model arises from inter-connected counting processes (events). To this end, let the bivariate log-price be:

$$\begin{aligned} X_t^1 &= X_0^1 + N_1(t) - N_2(t), \\ X_t^2 &= X_0^2 + N_3(t) - N_4(t), \end{aligned} \tag{4.29}$$

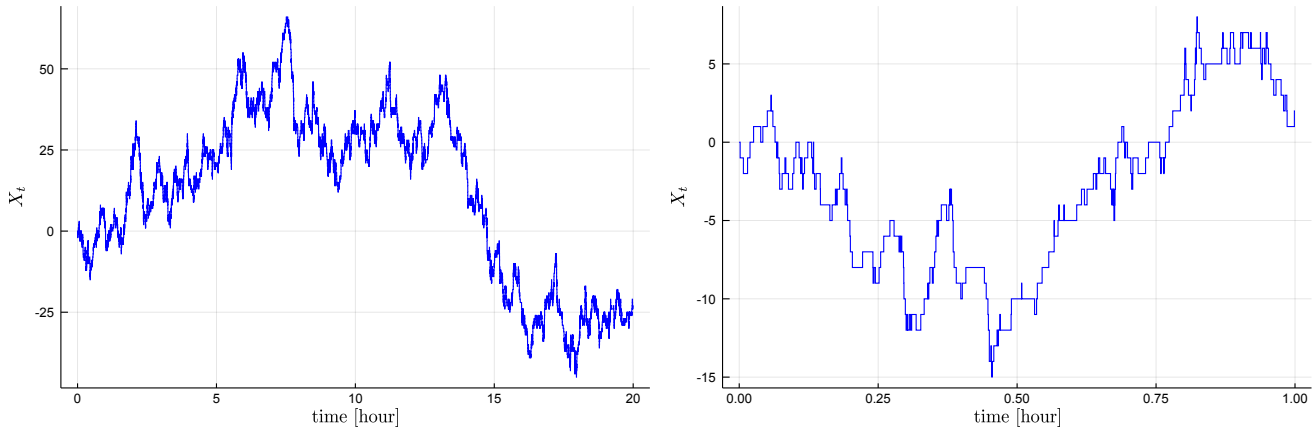
where $\{N_m(t)\}_{m=1}^4$ is a 4-dimensional mutually exciting Hawkes process, with $\mathbf{\Phi}$ taking the form:

$$\mathbf{\Phi} = \begin{pmatrix} 0 & \phi^{(r)} & \phi^{(c)} & 0 \\ \phi^{(r)} & 0 & 0 & \phi^{(c)} \\ \phi^{(c)} & 0 & 0 & \phi^{(r)} \\ 0 & \phi^{(c)} & \phi^{(r)} & 0 \end{pmatrix}. \tag{4.30}$$

The interpretation of eq. (4.30) is as follows: $\phi^{(r)}$ imitates mean reversion, because an uptick in X^1 by N_1 will lead to an increased intensity in the down tick N_2—allowing the price level to revert (similarly for X^2 through N_3 and N_4). While $\phi^{(c)}$ induces a correlation between the prices by connecting the two prices, since an uptick in X^1 by N_1 will lead to an increased intensity in the uptick of X^2 through N_3 (similarly for down ticks through N_2 and N_4).

I use the same parameters as Bacry et al. (2013a) for the construction of eq. (4.29). $\lambda_0^m = \mu, \forall m$, $\phi^{(r)} = \alpha^{(r)} e^{-\beta t} \mathbb{1}_{t\in\mathbb{R}^+}$ and $\phi^{(c)} = \alpha^{(c)} e^{-\beta t} \mathbb{1}_{t\in\mathbb{R}^+}$. Therefore, the set of parameters are $(\mu, \alpha^{(r)}, \alpha^{(c)}, \beta) = (0.015, 0.023, 0.05, 0.11)$.

Figure 4.4 plots a single realisation for the price model given by eq. (4.29). We see that over long horizons such as Figure 4.4a, the process begins to look like a diffusion process such as that in Figure 4.1. However, over short horizons such as Figure 4.4b, we can clearly see the jumps and that the price model is constructed based on discrete events. This happens because the Hawkes price model converges to Brownian motions as $T \to \infty$. The limit theorems for Hawkes price models have been derived by Bacry et al. (2013b).



(a) Full plot of 20 hours

(b) Zoomed plot of 1 hour

Figure 4.4: Here I plot a realisation of the Hawkes price model in eq. (4.29). (a) plots the full 20 hours where the path looks like that from a diffusion process, and (b) zooms in the first hour where the jumps are clearly seen. The figures can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

Bacry et al. (2013a) were able to derive the covariance matrix as a function of Δt for the Hawkes price model in eq. (4.29) given as:

$$\frac{C^{11}_{\Delta t}}{\Delta t} = \Lambda + \frac{RC_1}{2G_1} + \frac{RC_2}{2G_2} + R\frac{C_2G_1^2 e^{-\Delta t G_2} - C_1G_2^2 + Q_1G_2^2 e^{-\Delta t G_1} - C_2G_1^2}{2G_2^2G_1^2\Delta t},$$

and

$$\frac{C^{12}_{\Delta t}}{\Delta t} = \frac{-RC_1}{2G_1} + \frac{RC_2}{2G_2} + \frac{R\left(C_1G_2^2 - C_2G_1^2 - C_1G_2^2 e^{-G_1\Delta t} + C_2G_1^2 e^{-G_2\Delta t}\right)}{2G_2^2G_1^2\Delta t}.$$

Here the parameters are:

$$\Lambda = \frac{\mu}{1-\Gamma_{12}-\Gamma_{13}}, \qquad R = \frac{\beta\mu}{\Gamma_{12}+\Gamma_{13}-1},$$
$$C_1 = \frac{(2+\Gamma_{12}+\Gamma_{13})(\Gamma_{12}+\Gamma_{13})}{1+\Gamma_{12}+\Gamma_{13}}, \quad C_2 = \frac{(2+\Gamma_{12}-\Gamma_{13})(\Gamma_{12}-\Gamma_{13})}{1+\Gamma_{12}-\Gamma_{13}},$$
$$Q_1 = Q_4 = \frac{-\mu\left(\Gamma_{12}^2+\Gamma_{12}-\Gamma_{13}^2\right)}{\left((\Gamma_{12}+1)^2-\Gamma_{13}^2\right)(1-\Gamma_{12}-\Gamma_{13})},$$
$$Q_2 = Q_3 = \frac{-\mu\Gamma_{13}}{\left((\Gamma_{12}+1)^2-\Gamma_{13}^2\right)(1-\Gamma_{12}-\Gamma_{13})},$$

and

$$G_1 = \beta(1+\Gamma_{12}+\Gamma_{13}), \quad G_2 = \beta(1+\Gamma_{12}-\Gamma_{13}).$$

The correlation for the Hawkes price model is then given by:

$$\rho_{\Delta t}^{12} = \frac{C_{\Delta t}^{12}}{C_{\Delta t}^{11}}. \tag{4.31}$$

Moreover, in the limit we have that (Bacry et al., 2013a):

$$\lim_{\Delta t\to\infty} \rho_{\Delta t}^{12} = \frac{2\Gamma_{13}(1+\Gamma_{12})}{1+\Gamma_{13}^2+2\Gamma_{12}+\Gamma_{12}^2}. \tag{4.32}$$

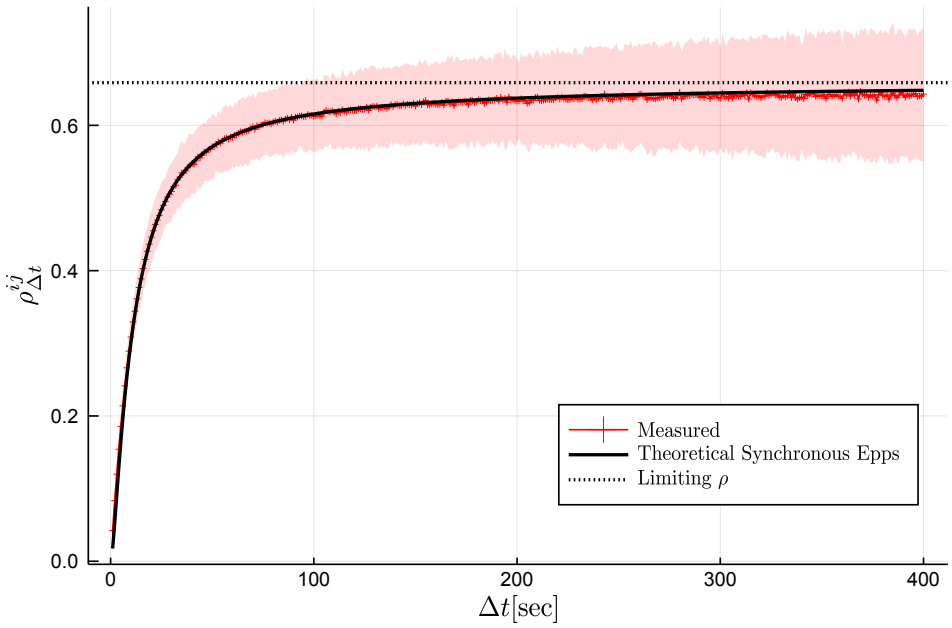
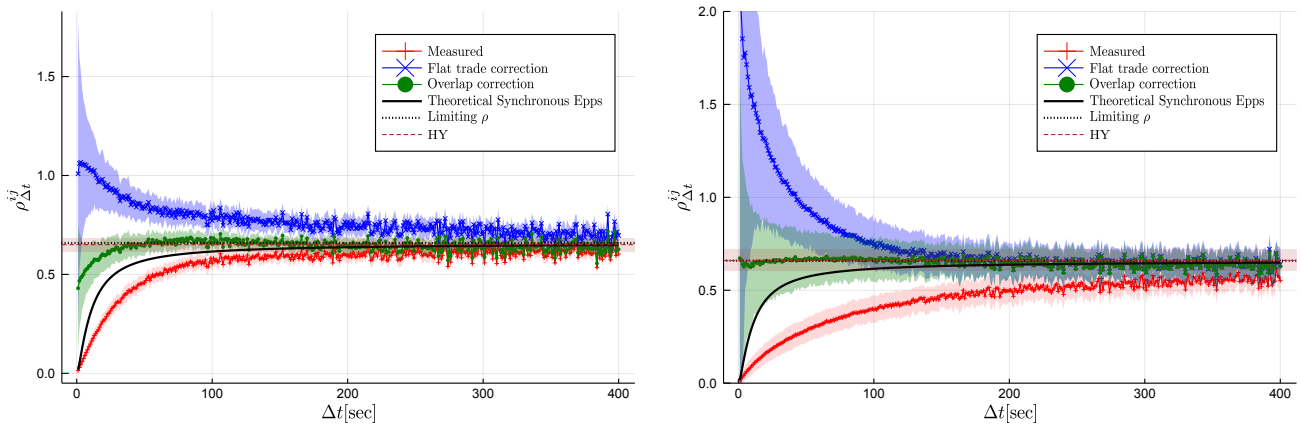


Figure 4.5: The Epps effect is demonstrated from the Hawkes price model where (+) is the measured correlation (See eq. (4.16)) from synchronous samples. The thick line is the plot of eq. (4.31), and the dotted horizontal line is the correlation as $\Delta t \to \infty$. Using the parameters $(\mu, \alpha^{(r)}, \alpha^{(c)}, \beta) = (0.015, 0.023, 0.05, 0.11)$, we have that $\rho_{\Delta t}^{12} \approx 0.65$ as $\Delta t \to \infty$. The figure can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

Figure 4.5 demonstrates the Epps effect arising from the Hawkes price model. A key feature here is that

the Epps effect arises from **synchronous** samples. There are no statistical causes such as asynchrony or tick-size, nor is the effect of lead-lag contributing towards the Epps effect here. In this case, rather than thinking of the Epps effect as a decay of existing correlations, it is more appropriate to think of it as the emergence of correlation as Δt increases. This is because at ultra-small time intervals, these events are merely random. The connection between the events can only be detected when there are sufficient events in a given sampling interval. This means a system created by an underlying web of inter-connected events take a finite time to correlate. This is a fundamental difference between the Hawkes price model and diffusion processes. Correlation from the Hawkes price model is a result of prices generated from discrete connected events and their connections can only be detected over larger intervals, while the diffusion based models assume the underlying correlation between prices exist at infinitesimal scales.



(a) Hawkes price model with Poisson sampling

(b) Hawkes price model with Hawkes sampling

Figure 4.6: The Hawkes price model with (a) Poisson sampling, and (b) Hawkes sampling are plotted where: $(+)$ is the measured correlation from the asynchronous process (See eq. (4.16)), $(\times)$ is the flat trade correction (See eq. (4.19)), and $(\circ)$ is the overlap correction (See eq. (4.17)). The thick solid line is the plot of eq. (4.31). The horizontal dotted line is when $\Delta t \to \infty$, making $\rho \approx 0.65$. Lastly, the horizontal dashed line is the Hayashi-Yoshida estimator (See eq. (4.22)). Here we see that the Hayashi-Yoshida estimator recovers the underlying limiting correlation, while the overlap correction can present a decaying residual Epps effect and the flat trade correction overcompensates. The figures can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

As with previous experiments, Figure 4.6 samples the Hawkes price model with (a) a Poisson process with a mean inter-arrival of $1/\lambda = 15$ seconds and (b) a 2-dimensional Hawkes process taking the form of eq. (4.26) with parameters $(\lambda_0^1 = \lambda_0^2, \alpha^{(s)}, \beta) = (0.015, 0.023, 0.11)$. The two processes sample the synchronous Hawkes price model where prices are extracted every second between $[0, T]$.

Figure 4.6a we see that the flat trade correction $(\times)$ over corrects the Hawkes price model with Poisson sampling this time. This is because the Hawkes price model only jumps to a new price state when an event occurs. Moreover, an uptick is likely followed by a down tick due to the mean reverting property; this interaction can lead to an amplification of an extended period of flat trades which results in the estimate of eq. (4.20) being closer to 1. Now this over correction is further amplified in Figure 4.6b with the Hawkes sampling as the above effect suffers further from the longer inter-arrivals for the specific Hawkes sampling used. The measured correlation $(+)$ from the asynchronous process exhibits a further Epps effect relative to the theoretical synchronous Epps effect (thick line) which is expected. This difference is naturally attributed towards the asynchronous sampling. What is more interesting is that in both Figures 4.6a and 4.6b, the Hayashi-Yoshida estimates (horizontal dashed line) recover the limiting correlation, but the overlap correction $(\circ)$ only recovers the limiting correlation in Figure 4.6b. Figure 4.6a the overlap correction at ultra-high frequencies under estimate the limiting correlation and over estimates

the synchronous Epps effect. This residual Epps effect[4] depends on the length of inter-arrivals in U^i and U^j and is the key factor to discriminate between diffusions and connected events. I will demonstrate why and how in the next section, and that it is not only unique to the overlap correction.

Remark 4.3.2 *The reader may have noticed that something does not sit right with Figure 4.6b. If we are correcting for asynchrony, why is the Hayashi-Yoshida estimate and overlap correction recovering the limiting correlation and not the correlation from the synchronous case (the thick black line)? Although the corrections are supposed to recover the correlation from the synchronous process, the key insight is realising that the sampling process can affect the recovery of the underlying synchronous process when correlations are emergent. This is the reason why the residual Epps effect depends on the length of inter-arrivals in U^i and U^j for the Hawkes process. I will further elaborate this in the next section. First, let me make an additional remark.*

Remark 4.3.3 *The reader may have noticed that the overlap correction is derived for diffusion processes but here I have applied it to the Hawkes process. This is because deriving an analogous version for the Hawkes process is extremely difficult since the underlying correlation depends on Δt and the asynchronous sampling affects the recovery of the underlying synchronous correlation. Nonetheless, applying the correction should not be an issue because the correction accounts for the distortion of the non-overlapping portions of overlapping intervals. Thus extracting the correlation from the synchronous process on the overlapping portions.*

4.4 The residual Epps effect

When asynchrony is introduced into the Hawkes price model, the effect of the corrections depend on the length of inter-arrivals from U^i and U^j. This is because as the average length of the inter-arrivals increase, larger intervals are implicitly investigated (through the asynchronous samples) thus allowing events to correlate towards its limiting correlation. This is why both the Hayashi-Yoshida estimate and overlap correction recover the limiting correlation in Figure 4.6b due to the long inter-arrivals from the specific specification of Hawkes sampling used. Therefore, the level of correction that can be achieved depends on the length of inter-arrivals in U^i and U^j which can affect the residual Epps effect.

The implicit time-scale from the sampling process only matters for the Hawkes price model and not for the diffusion model. This is because the underlying correlation of the synchronous process depends on the time-scale for the Hawkes price model while the correlation from diffusion models are independent of the time-scale. Therefore, using a diffusion model, the correction will recover the induced correlation regardless of the length of inter-arrivals from the sampling process. However, with a Hawkes price model, the correction will demonstrate a decaying residual Epps effect for sampling processes with smaller inter-arrivals. This realisation is key when designing experiments to discriminate between diffusion based processes against processes from connected events.

Experiment #1: Hayashi-Yoshida

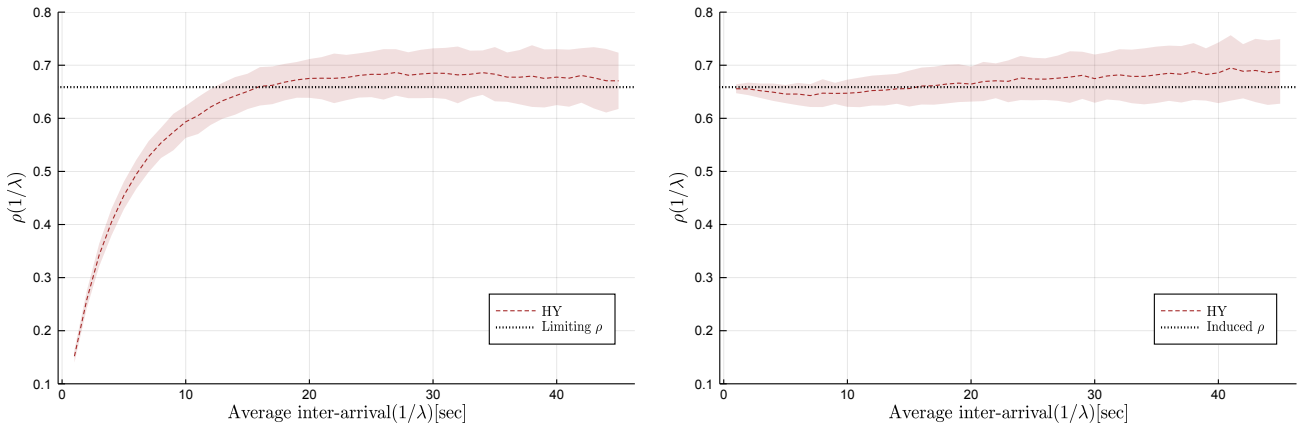
Using this insight, the next step is to design experiments to test the level of correction achieved for different lengths of inter-arrivals. In this experiment, the Hayashi-Yoshida estimator does not depend on Δt and returns a base-line correlation. Therefore, the residual Epps effect in this case is not the usual residual seen in the overlap correction of Figure 4.6a. This means we can directly plot the Hayashi-Yoshida estimate as a function of the mean inter-arrivals; rather than the various residual Epps curves for different mean inter-arrivals (See the next experiment).

[4]The residual Epps effect refers to the remaining Epps effect after correcting for asynchrony.

Now if the underlying process is discrete then we should see the Hayashi-Yoshida estimates decay for smaller inter-arrivals. On the other hand, if the underlying process is diffusion based then we should see that the Hayashi-Yoshida estimates are flat for any sized inter-arrivals.

Let us consider an experiment to demonstrate this point. Here I simulate a single realisation of price paths from the Brownian price model and the Hawkes price model. The length of inter-arrivals are controlled using the Poisson sampling with the mean ranging from 1 to 45 seconds. Each price model is then re-sampled with the different mean inter-arrivals 100 times. Figure 4.7 plots the mean Hayashi-Yoshida estimate on (a) the Hawkes price model and (b) the Brownian price model as a function of the mean inter-arrival time with error ribbons containing 95% of the estimates at each $1/\lambda$ over the 100 replications.

Figure 4.7a we see that for the Hawkes price model, the Hayashi-Yoshida estimator presents an Epps effect as a function of the mean inter-arrival time. Moreover, as $1/\lambda$ increases, the correlation tends towards the limiting correlation. This demonstrates the impact that implicit time-scales (from the arrival dynamics) have on the correction of asynchrony for discrete connected events. When the sampling process has smaller inter-arrivals, the correction will recover correlation estimates from the synchronous process at a smaller time-scale. Figure 4.7b we see that for the Brownian price model,[5] the Hayashi-Yoshida estimator recovers the induced correlation regardless of the mean inter-arrival time.



(a) Hawkes price model

(b) Brownian price model

Figure 4.7: The Hayashi-Yoshida estimate (dashes lines) given in eq. (4.22) plotted as a function of the mean inter-arrival for (a) the Hawkes price model and (b) the Brownian price model. The limiting/induced ρ (horizontal dotted line) is approximately 0.65 as before. Here we see how the length of inter-arrivals can affect the correction for the two price models. The figures can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

This means we can test the underlying process for an unknown system by re-sampling the process at different mean inter-arrivals and estimating the Hayashi-Yoshida estimates. If the plot presents decaying correlation for smaller $1/\lambda$ then the underlying process is from discrete connected events (a system where correlation is emergent); and if the plot is flat for various values of $1/\lambda$ then the underlying process is diffusion based.

[5] I highlight that the discretisation in the simulation of the diffusion process should be finer than the average inter-arrivals to avoid insufficient granularity in the diffusion process.

Experiment #2: Overlap correction

Since the Hayashi-Yoshida estimates do not depend on Δt, we could not plot the actual residual Epps curves. Therefore in this experiment, I use the overlap correction to examine the effect of different average inter-arrivals in relation to the synchronous Epps effect of the Hawkes price model. This allows us to better visualise the interplay between the sampling intervals used in estimating the correlation Δt and the implicit time-scale investigated through the asynchronous samples. Figure 4.8 plots the overlap correction for average inter-arrival times $1/\lambda = 1, 10, 25$ $(+, \times, \circ)$ respectively from Poisson sampling on the Hawkes price model (with mean and error ribbons from 100 replications). We see that as $1/\lambda$ decreases, the overlap correction starts to look like the synchronous Epps effect (thick black line), and as $1/\lambda$ increases, the overlap correction tends towards the limiting correlation. In other words, when the average inter-arrival decreases, the residual Epps effect decays further and we start to recover the correlation from the synchronous case.

In this experiment, the implicit time-scale from the asynchronous samples influences the ability of level of correction that can be achieved using the overlap correction. There is a band where the residual Epps curve will lie. This is between the theoretical synchronous Epps (decaying residual Epps curve), and the limiting correlation (flat residual Epps curve). However, in the first experiment, the level of correction achieved in the Hayashi-Yoshida estimate completely depends on the implicit time-scale as the estimator itself does not depend on Δt. Therefore, the Hayashi-Yoshida estimates in this case will not necessarily recover the same saturation level as the overlap correction. This can be seen in Figures 4.7a and 4.8, when $1/\lambda = 1$ the Hayashi-Yoshida estimate is around 0.15 which is not the saturation level achieved in the overlap correction.

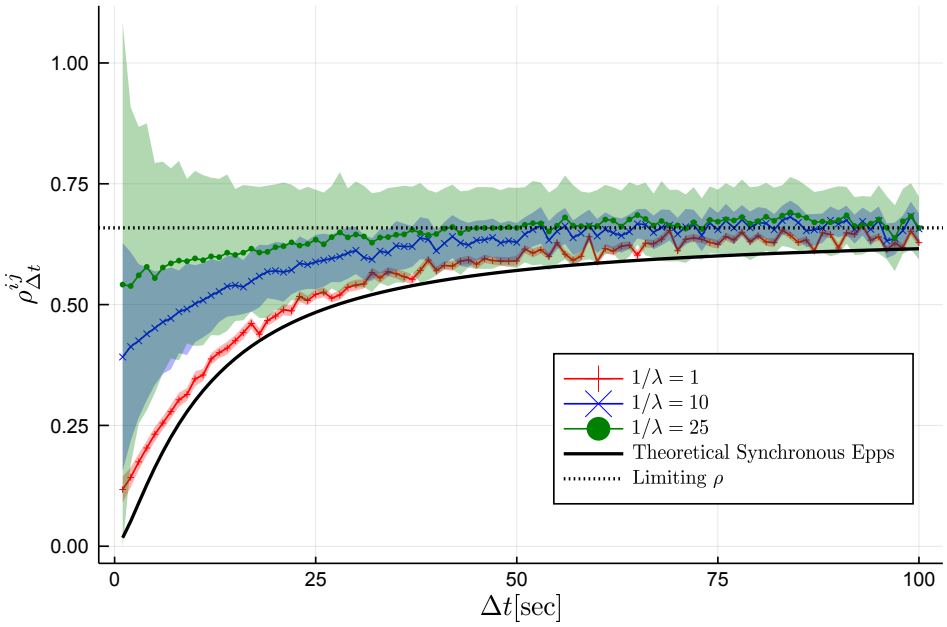


Figure 4.8: The overlap corrections (See eq. (4.17)) for three sampling frequencies are plotted. The three sampling frequencies chosen are: $1/\lambda = 1, 10, 25$ $(+, \times, \circ)$ respectively. The thick line is the theoretical synchronous Epps effect given by eq. (4.31) and the horizontal dotted line is the limiting ρ given by eq. (4.32) using the same parameters as before. Here we see that as the average inter-arrivals decrease, the residual Epps starts to look like the synchronous Epps effect; while when the average inter-arrivals increase, the residual Epps starts to recover the limiting correlation. The figure can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

In this case, there is no need to perform the experiment with the Brownian price model. This is because with the Poisson sampling, the Epps effect is characterised by eq. (4.14), therefore the residual Epps

effect here will be flat for the various inter-arrivals. This was the conundrum presented to researchers investigating the Epps effect. In their simulation experiments, the residual Epps effect was always flat because they used diffusion processes, but when the corrections are applied to empirical observations the residual Epps effect presented a decaying behaviour seen in Figure 4.8. Concretely, Mastromatteo et al. (2011) compensated the Epps effect arising from asynchrony and found that a significant fraction of the Epps effect cannot be explained through asynchrony alone. Münnix et al. (2011) compensated the Epps effect arising from asynchrony and tick-size, and still found that a portion of the Epps effect is still unaccounted for. This lead to Tóth and Kertész (2009) conjecturing that the unexplained fraction may be a result of time-scales relating to human reaction. Here I offer another possible explanation in Figure 4.8. If the empirical reality is that the price observations are from a discrete connected system, then the residual Epps effect has a natural explanation based on the underlying synchronous process itself having correlations that depend on Δt. The Epps effect from the synchronous Hawkes price model is a model formulation that captures the behavioural argument by Tóth and Kertész (2009).

This experiment can also give us a method to differentiate between systems based on diffusion processes and discrete connected events. This is achieved by sampling the unknown system with different mean inter-arrivals and then plotting the residual Epps curves using the overlap correction. If the residual Epps curves decay more for smaller $1/\lambda$ then the underlying process is from discrete connected events; and if the residual Epps curves are flat for all values of $1/\lambda$ then the underlying process is diffusion based.

Experiment #3: k-skip Hayashi-Yoshida

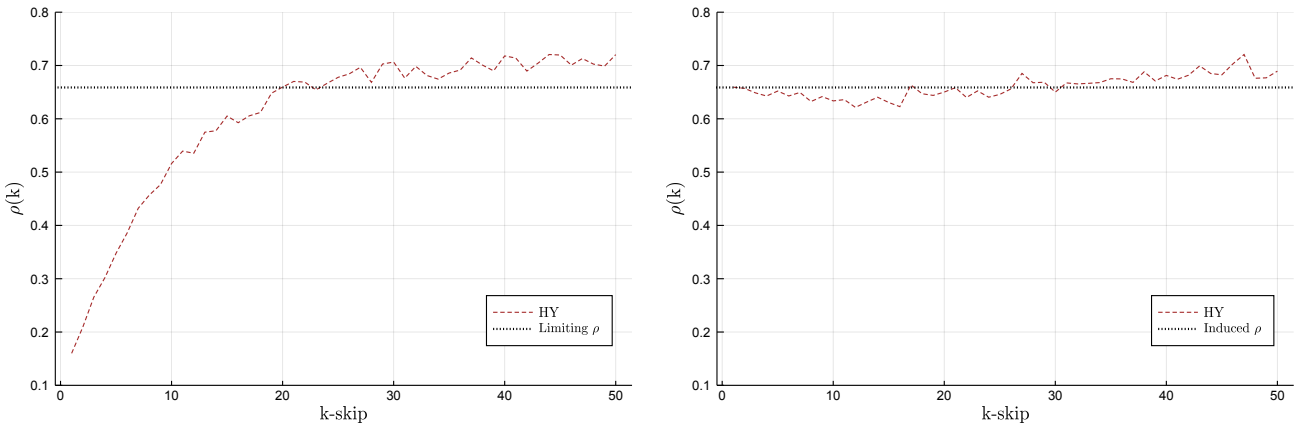
The first two experiments are well suited to determine the underlying process of an unknown system in a simulation scenario. This is because the first two experiments require the re-sampling of the process to obtain different sets of U^i and U^j with different sized inter-arrivals. This is problematic with real-world data because there is only one set of U^i and U^j. Moreover, the second experiment requires multiple replications to get a good gauge on the error ribbons. Using results based on one replication with the second experiment cannot meaningfully determine the underlying process; error bands are required to clearly see that the process consistently decays for a given sized inter-arrival process. The first experiment does not require replications to meaningfully determine the underlying process, this is because the error ribbons are very narrow. Meaning that the estimates are either flat or they decay.

This experiment is a variant of the first experiment but overcomes the issue of requiring multiple sets of U^i and U^j, by using a single set of U^i and U^j when discriminating. Griffin and Oomen (2011) provided the inspiration to over come this issue by using k-skip sampling. Here with one set of arrivals U^i and U^j, we can imitate multiple sets of arrivals with different sized inter-arrivals by sampling every k^{th} observation in the single set of arrivals. Concretely, from the single set of U^i, let the k^{th}-skip set of samples be the set $U^i_k = \left\{t^i_k, t^i_{2k}, ..., t^i_{\lfloor \#U^i/k \rfloor k}\right\}$. Again, note that $\#U^i$ is the cardinality of set U^i. Instead of re-sampling and computing the Hayashi-Yoshida estimate with different mean inter-arrival $1/\lambda$, we can now compute the Hayashi-Yoshida estimate as a function of the k^{th}-skip set of samples U^i_k and U^j_k.

Figure 4.9 plots the Hayashi-Yoshida estimates as a function of the k-skip sampling for (a) the Hawkes price model and (b) the Brownian price model. The k-skip sampling is performed from sampling every observation $k = 1$ to every 50^{th} observation $k = 50$. To re-create conditions similar with empirical data, there is only one set of asynchronous observations from each price process. Figure 4.9 recovers similar results as Figure 4.7. Figure 4.9a we see that for the Hawkes price model, the Hayashi-Yoshida estimator presents an Epps effect as a function of the k-skip sampling; while Figure 4.9b we see that for the Brownian price model, the Hayashi-Yoshida estimator recovers the induced correlation for any sized k-skip sampling.

This result allows us to test the underlying process for an unknown system with one set of asynchronous samples. If the plot presents a decaying correlation for smaller k then the underlying process is from

discrete connected events; and if the plot is flat for various values of k then the underlying process is diffusion based.



(a) Hawkes price model

(b) Brownian price model

Figure 4.9: The Hayashi-Yoshida estimate (dashes lines) given in eq. (4.22) plotted as a function of the k-skip sampling for (a) the Hawkes price model and (b) the Brownian price model. The limiting/induced ρ (horizontal dotted line) is approximately 0.65 as before. We see that k-skip sampling allows us to discriminate the two processes with one set of arrivals. The figures can be recovered using the Julia script file EppsCorrection.jl on the GitHub resource (Chang et al., 2020c).

Figure 4.9a is very interesting because Griffin and Oomen (2011) found this behaviour with empirical Trade and Quote (TAQ) data from the New York Stock Exchange (NYSE). They argue this behaviour is an issue with the Hayashi-Yoshida estimator because a key underlying assumption is that when a price update arrives it should fully incorporate all available information regarding the correlation (this is only true for the Brownian price model as they are assuming that the correlation should not depend on Δt). Therefore, the Hayashi-Yoshida estimate should be flat when using the k-skip sampling. They conjecture that this is because empirical data has a sluggish behaviour for price adjustments to reflect the appropriate correlation. This is essentially just re-stating the conjecture by Tóth and Kertész (2009) relating to time-scales of our human reaction. In the Hawkes price model, this sluggish behaviour is simply a result of correlations emerging as Δt increases.

It is clear how Figures 4.8 and 4.9a seamlessly ties together the theory and empirical observations when using a Hawkes representation. I argue that some of the literature has perhaps misconstrued some of the empirical observations as a result of an incorrect underlying representation. Moreover, the implied correlation dynamics from the conjectures of time-scales relating to our human reaction and sluggish price adjustments are encapsulated within the Hawkes representation.

Remark 4.4.1 *I must highlight that Griffin and Oomen (2011) were able to remove the decay in Figure 4.9a by adjusting the Hayashi-Yoshida estimator to account for lead-lags. They argue lead-lag effects arise as a result of sluggish price adjustments. Although lead-lags can arise within the framework of Hawkes price model by Bacry et al. (2013a,b), the specification I have used here in eq. (4.30) is fully symmetric and there is no lead-lag effects (See Bacry et al. (2013a) and Remark 9 of Bacry et al. (2013b)). Therefore, the sluggish behaviour in Figure 4.9a is due to the fact that correlation is emergent.*

4.5 Empirical investigation

In this section, I will compare the correction methods and detect the underlying processes using banking equities from the Johannesburg Stock Exchange (JSE) with Trade and Quote (TAQ) data for 40

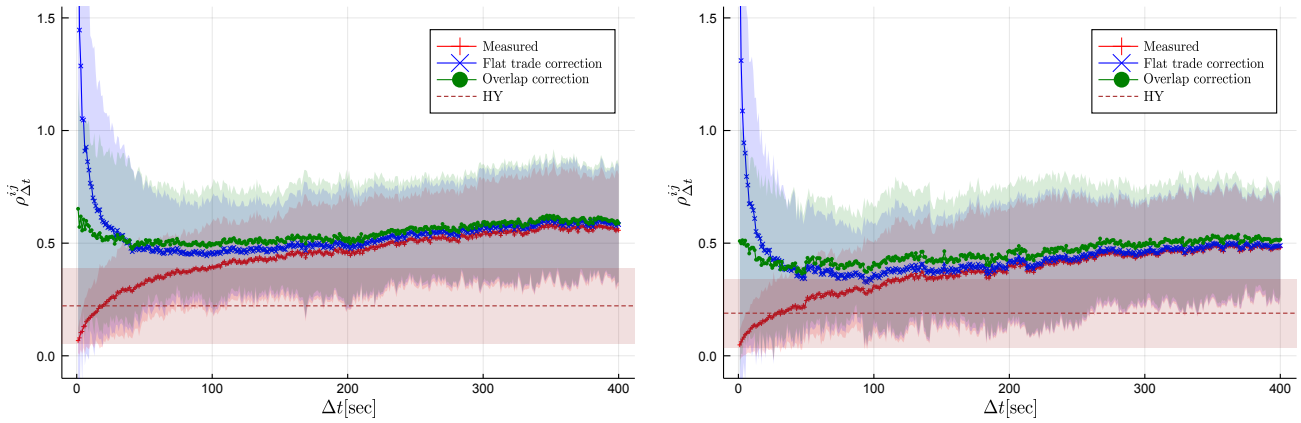
trading days starting from 2019/05/02 to 2019/06/28. The data is extracted from Bloomberg Pro and is processed so that trades with the same time stamp are aggregated using a volume weighted average (See Appendix C.2). The equities considered are: FirstRand Limited (FSR), Absa Group Ltd (ABG), Nedbank Group Ltd (NED) and Standard Bank Group Ltd (SBK).

Here the correlations are measured for each trading day and the ensemble is reported. The error ribbons are computed as before, they contain 95% of the estimates at each Δt between the trading days. They are computed using the Student t-distribution with 39 degrees of freedom and the standard deviation of the estimates between the days at each Δt. Additionally, $t = 0$ starts once the equity pair has each made their first trade (See Appendix C.3), and $T = 28200$ seconds from a seven hour and 50 minute trading day.

Tickers	$1/\hat{\lambda}$[sec]	$\hat{\sigma}(1/\lambda)$
FSR	12.09	19.53
SBK	13.06	21.51
NED	15.39	25.14
ABG	15.68	26.27

Table 4.1: The table reports the mean inter-arrival estimate and the associated standard deviation for the four banking equities over the 40 day trading period.

With empirical TAQ data we cannot re-sample the process ourselves, we only have one set of arrivals. Although the second experiment is not well suited for testing empirical data, it does reveal something interesting. Therefore, in order to apply the second experiment, I try to re-create the simulation setting by finding equity pairs with similar correlation levels but have different average inter-arrivals. Table 4.1 gives the estimate for the mean and standard deviation for the inter-arrival time over the 40 trading day period. It is clear that the pair FSR and SBK (NED and ABG) have on average smaller (larger) inter-arrival times respectively. Therefore, I will compare the corrections on FSR/SBK against NED/ABG to determine if there is a difference in the decay of the residual Epps effect following the spirit of the second experiment.



(a) SBK/FSR **(b)** NED/ABG

Figure 4.10: The corrections (See Section 4.2.2) are applied and compared for the equity pairs (a) SBK/FSR, and (b) NED/ABG. Here (+) is the measured correlation from the asynchronous process (See eq. (4.16)), (×) is the flat trade correction (See eq. (4.19)), and (○) is the overlap correction (See eq. (4.17)). Lastly, the horizontal dashed line is the Hayashi-Yoshida estimator (See eq. (4.22)). The figures can be recovered using the Julia script file Empirical_Epps.jl on the GitHub resource (Chang et al., 2020c).

Figure 4.10 I compare the correction methods on the pair (a) FSR/SBK and (b) NED/ABG to investigate the residual Epps effect. We see that the variability of the correlation estimates between days is very high (similar to Chapter 3) but largely remains positive. This is why I chose the banking equities since they have a strong correlation (seen in Chapter 2). Here if the equities are less correlated, the Epps curves can jump between negative and positive between the days. The flat trade correction ($\times$) once again over-compensates the correction. Here the Hayashi-Yoshida estimates (horizontal dashes) do not recover the same estimates as the overlap correction ($\circ$) or the saturation level. This is a sign that the underlying process is from discrete connected events as this also seen in Figures 4.7a and 4.8 where the underlying process is a Hawkes price model.

Figure 4.11 takes a closer look at the overlap correction ($\circ$) and the measured correlation ($+$) for the equity pairs SBK/FSR (dotted line) and NED/ABG (solid line) from Figure 4.10. Here I re-scale the estimates by the saturation level for better comparison since we have two different equity pairs. We see the measured and correction are similar for both the equity pairs. The similarity of the overlap correction may be due to the fact that the average inter-arrivals are not significantly different. What is more interesting is this U-shaped residual Epps curve which has not been reported in the broader literature. This might be because the correlations are usually not applied at such a fine resolution. Here when $\Delta t > 100$ seconds the decay is present, the decay starts to flatten out for Δt between 50 and 100 seconds, and below 50 seconds the residual correlation starts to rise back towards the saturation level. This behaviour could not be replicated in the simulations with the price models and sampling methods considered.

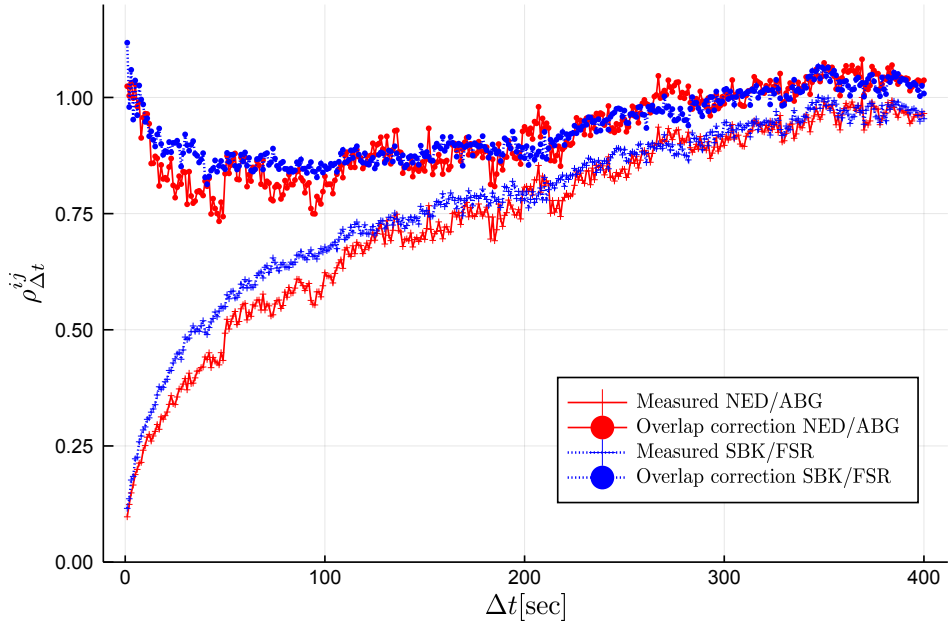


Figure 4.11: The overlap correction ($\circ$) is given by eq. (4.17) and show in the upper two curves for the two stock pairs investigated. The measured correlation ($+$) as given by eq. (4.16) are scaled by the saturation level (See section 4.5) for the equity pairs SBK/FSR (dotted line) and NED/ABG (solid line) and compared; they are the two lower curves. The figure can be recovered using the Julia script file Empirical_Epps.jl on the GitHub resource (Chang et al., 2020c).

We see that the first and second experiment from Section 4.4 are poorly adapted to detect the underlying process with empirical TAQ data. Thus let us now consider the k-skip sampling to detect the process. Figure 4.12 computes the Hayashi-Yoshida estimates with the k-skip sampling with k ranging from 1 to 50 on the equity pairs SBK/FSR ($+$) and NED/ABG ($\circ$). Both equity pairs present a decaying correlation for smaller k as seen in Figure 4.9a. This is strong evidence suggesting that empirical Trade and Quote data are better modelled as an underlying web of inter-related discrete events such as a Hawkes process.

Although the evidence in Figure 4.12 is quite convincing, we have yet to control for the other effects contributing towards the Epps effect such as tick-size and lead-lag. Therefore, additional research is required to design experiments that incorporate tick-size and lead-lag when discriminating the underlying process. This extension may possibly provide insight into this U-shaped residual Epps curve. Nonetheless, the Hawkes price model seems to be the preferred approach to model high-frequency correlation dynamics as the theory lines up with the empirical dynamics seen in Trade and Quote data.

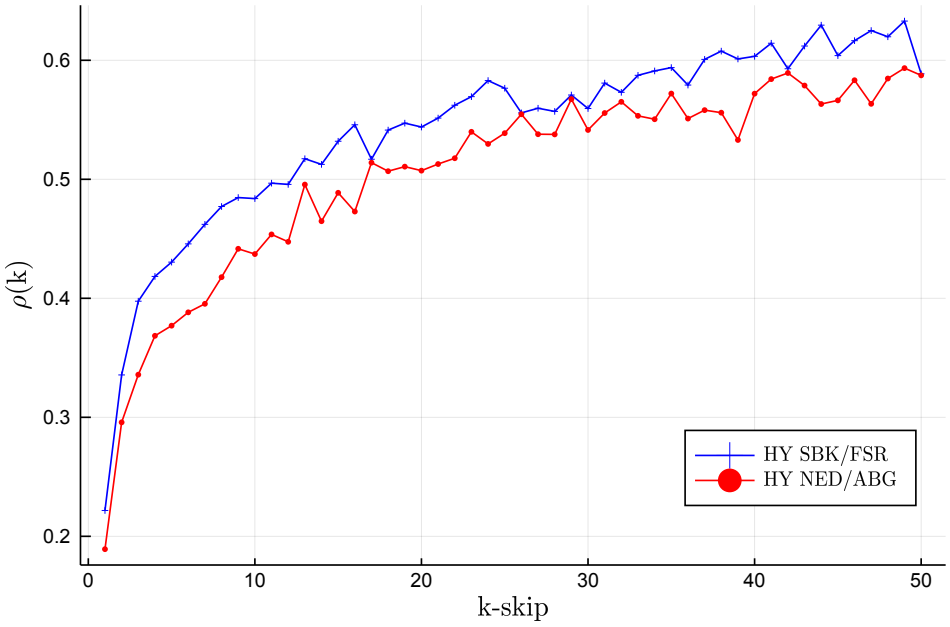


Figure 4.12: The k-skip sampling with the Hayashi-Yoshida estimator applied to the equity pairs SBK/FSR (+) and NED/ABG (○). The empirical reality suggests the underlying system is from discrete connected events. The figure can be recovered using the Julia script file Empirical_Epps.jl on the GitHub resource (Chang et al., 2020c).

4.6 Closing remarks

In this chapter, I derive the Epps effect arising from asynchrony and provide a refined method based on existing work (Münnix et al., 2010, 2011) to correct for this effect and is informed by the rich prior literature (Bacry et al., 2013a,b; Mastromatteo et al., 2011; Tóth and Kertész, 2009, 2007; Precup and Iori, 2007). The refinement is simpler to compute as it separates the estimation of correlation and correction factor, nor does it make strong assumption about the distributional properties of the sampling processes.

The method is compared against the Hayashi-Yoshida estimator (Hayashi and Yoshida, 2005) and a correction using the probability of flat trading (Buccheri et al., 2019) on a Brownian price model, a Merton price model, and a Hawkes price model. We found that our correction recovers similar estimates to the Hayashi-Yoshida estimator and outperforms the flat trade correction.

I then design three experiments to detect whether the underlying process is diffusion based or discrete connected events. The first two are better suited for simulation scenarios as it requires us to repeatedly re-sample the process. The third correction using k-skip sampling allows us to detect the underlying process with one set of arrivals. From these experiments, we see that the dynamics seen in the Hawkes price model seemingly recovers the empiricism reported in the literature that cannot be recovered using a Brownian representation.

The corrections and experiments are applied to Trade and Quote data from the Johannesburg Stock Exchange. Here I found a surprising U-shaped residual Epps curve which could not be replicated under

simulation. Moreover, I found strong evidence using the k-skip sampling suggesting that empirical Trade and Quote data is from a system of discrete connected events.

More research is required to convincingly conclude that Trade and Quote data is from a discrete connected system. Such research entails incorporating tick-size and lead-lag when discriminating the underlying process, and performing the discrimination over a wider variety of assets across a range of sectors. The work of Bacry et al. (2013b), Mastromatteo et al. (2011), and Münnix et al. (2010) will serve as a good starting point.

Chapter 5

Concluding remarks

What was achieved

Chapter 2

I applied non-uniform fast Fourier transforms in the context of the Malliavin-Mancino integrated estimator. Using NUFFT methods, I am able to quickly evaluate the Fourier coefficients of the price process at non-equispaced grid points. This allows fast Fourier methods to be used in all synchronous and asynchronous cases. The failure behind the zero-padded FFT is the motivation behind why non-uniform FFTs are required. NUFFTs preserve the power spectrum with an appropriate convolution and deconvolution, thus ensuring we obtain the appropriate Fourier coefficients of the price process used in the Malliavin-Mancino estimator.

The NUFFT implementation was performed using three types of averaging kernels: the Gaussian, Kaiser-Bessel, and exponential of semi-circle. Based on my implementations, the Gaussian kernel was the fastest using the fast Gaussian gridding implementation. Moreover, the NUFFT methods do not present an issue in terms of accuracy provided a low enough tolerance ε is requested.

I have demonstrated how the cutting frequency N relates to the time-scale of investigation Δt by comparing the Malliavin-Mancino estimates to the Epps effect arising from asynchrony derived in Chapter 4. We saw that the Dirichlet representation can better recover this theoretical Epps effect but does not exactly recover it. The difference is because of how the Malliavin-Mancino estimator deals with asynchrony compared to the previous tick interpolation used when deriving the theoretical Epps effect. Although in Chapter 3 we saw that the Malliavin-Mancino estimator handles asynchrony better by providing stable estimates, more research is required to understand the difference between the two methods and their relative efficacy. I argue that the Fejér representation is useful when positive semi-definiteness is a requirement, or if one wants to remove the Epps effect completely. However, this can be problematic as seen in Chapter 4 because there seems to be more to the Epps effect than simple statistical biases.

An empirical analysis investigating the integrated correlation dynamics at different time-scales on the Johannesburg Stock Exchange revealed an equity pair that does not conform to the Epps effect. This may mean that there may be more to high-frequency correlation dynamics than the Epps effect.

Chapter 3

I once again applied non-uniform fast Fourier transforms in the context of the Malliavin-Mancino instantaneous estimator. This is then compared against the Cuchiero-Teichmann instantaneous estimator. Here I pull together two threads of literature that has had little investigation. Specifically, the instantaneous Epps effect and the impact of cutting frequencies in the Fourier estimators.

I find that M plays a key role in the accuracy of the spot estimates and argue that the choice of M should not be a one size fit all choice but should rather depend on the underlying composition of the true spot parameter of interest. Moreover, I demonstrated the instantaneous Epps effect under simulation by investigating different time-scales through choice of N in the Malliavin-Mancino estimator and choice of discretisation Δt in the previous tick interpolation with the Cuchiero-Teichmann estimator. This demonstration revealed that the previous tick interpolation results in unstable estimates because the synchronised price path can change depending on the size of discretisation; while the ability to bypass

the time domain in the Malliavin-Mancino estimator makes the estimates more stable for different choice of time-scales (implied through the cutting frequency N). Therefore, I argue that the Malliavin-Mancino estimator is the preferred estimator for high-frequency finance because of its elegant method of dealing with asynchrony.

The reason behind why the instantaneous estimators break-down under asynchronous observations is not exactly clear. Nonetheless, I provide an *ad hoc* method of correcting the instantaneous Epps effect by picking an appropriate time-scale from the Epps curves. This approach is not without issues because it can conceal genuine causes of the Epps effect from statistical causes. More research is required into the instantaneous Epps effect in order to resolve this issue.

An empirical analysis found the existence of the instantaneous Epps effect on the Johannesburg Stock Exchange. After correcting for this by picking an appropriate time-scale, we saw that the instantaneous correlation between days for the same equity pair can display different dynamics. These differences cannot be clearly seen with the integrated correlations because the intraday dynamics get aggregated into a single metric, thus demonstrating the advantage of using instantaneous estimators when investigating correlation dynamics in the financial market.

Chapter 4

I derive the Epps effect arising from asynchrony and provide a refined method to correct for this effect based on the work in the existing literature. The refinement separates the compensation factor from the estimation of the correlation making it simpler to compute. Additionally, it does not have strong distributional assumptions regarding the sampling process. The correction derived is compared against the Hayashi-Yoshida estimator and a correction provided by Buccheri et al. (2019). We see that my correction recovers similar estimates as the Hayashi-Yoshida estimator and both outperform the correction from Buccheri et al. (2019).

Using these corrections to disentangle genuine effects causing the Epps effect from asynchrony allows us to design experiments to detect the underlying process generating the price observations. Concretely, I design two experiments to detect the underlying process under simulation conditions where the process can be re-sampled. Moreover, I design a third experiment to detect the underlying process with only one set of arrivals which is better suited for an empirical experiment using Trade and Quote data.

These experiments demonstrate how the theory of emerging correlations using a Hawkes representation seemingly recovers the empiricism reported in the literature. Specifically, these experiments demonstrate that using the Hawkes representation, we can recover the same empirical dynamics reported in the literature under simulation conditions. This could not be achieved by previous researchers because they focused reconciling the empirical dynamics with the Brownian representation. Moreover, the Hawkes representation captures the dynamics implied by behavioural arguments suggested in the literature when trying to reconcile the empirical dynamics with the Brownian representation. This is promising because a good model should be able to recover stylised facts and obtain results from simulations that line up with empirical observations.

The experiments were applied to Trade and Quote data from the Johannesburg Stock Exchange. The experiments revealed strong evidence suggesting that empirical Trade and Quote data are better modelled as as discrete but connected events (such as a Hawkes process). The evidence although strong, is not indisputable as I have yet to control for other sources such as lead-lag or tick-size. Further research is required to comprehensively include every known source of the Epps effect to discriminate the underlying process.

Are Brownian motions good enough for high-frequency finance?

I argue that the reason why the literature has not been able to account for the entirety of the Epps effect using empirical data is because of an incorrect underlying representation using Brownian diffusions. On the other hand, using the Hawkes representation we are able to explain the entirety of the Epps effect seen with empirical data. The Hawkes representation ties together the theory and empiricism. As Box et al. (2009) famously said: "All models are wrong, but some models are useful". Thus the real question is rather: "Is the model good enough for this particular application"? In this case, the Brownian diffusion representation is not good enough when recovering high-frequency correlation dynamics. The Hawkes representation however, seems to be better suited for this particular application.

The implication behind this is that the Epps effect is a fundamental property of market microstructure rather than a bias. This means we cannot use high-frequency data to achieve better estimates for a low-frequency correlation matrix because correlation in the financial markets are inherently tied to a particular time-scale. Thus we need to take additional care when making decisions in finance using correlation estimates. In particular, the time-scale of the application for decision making should determine the time-scale at which to estimate correlation matrices.

Ways forward

Since Bachelier (1900) introduced Brownian motions to the world, Brownian motions have largely formed the mathematical basis around the available tools we have for understanding the financial markets. A reason behind this is because of its analytical tractability, allowing us to gain insight through closed form solutions. Moving away from Brownian motions remain difficult because of how much work has been built upon it. Nonetheless, in recent decades because of the large amount of financial data and affordable computational power, there has been a branch of research trying to gain insight into financial markets using methods from complexity science. This is because ultimately, the financial market is the aggregation of individual agents and intuitions acting in their own interests. In particular, Agent-Based models (ABMs) has seen a rise in popularity. I am however cognisant of the fact that ABMs face issues particularly in the realm of high-frequency finance. In particular, Platt and Gebbie (2018) have argued that ABMs are over-specified in high-frequency finance and their parameters are degenerative. Therefore, the way forward is not exactly clear.

However, Hawkes processes do seem to be a promising avenue. Hawkes processes can capture the event based dynamics in high-frequency finance rather than the traditional view of a continuous process. It is simple enough to be analytically tractable thus allowing us to obtain closed form solutions. The issue of calibration is more manageable and we have the random time change theorem to test the specification (Bowsher, 2007). Moreover, Hawkes price models such as the one proposed by Bacry et al. (2013a,b) have this fine-to-coarse property where the process converges to Brownian diffusions. In particular, an extension to Bacry et al. (2013a,b) was proposed by Jaisson and Rosenbaum (2015) where the process converges to a Heston price model. This work is a promising stepping stone to a unified model that can recover stylised facts across the various scales. In other words, the model recovers the microstructure stylised facts such as the discreteness of prices and mean reversion, and large scale diffusive properties such as volatility clustering (Bacry et al., 2015).

Potential topics for future research

Jump robust estimator for asynchrony

Chapter 3 we saw that the Malliavin-Mancino estimator handles asynchrony better than the Cuchiero-Teichmann estimator, but the Cuchiero-Teichmann estimator handles jumps better than the Malliavin-Mancino estimator. Therefore, it is natural to try combine the strengths of each estimator into a new

Fourier estimator, one that is able to naturally deal with asynchrony while being robust to jumps. This estimator may prove useful in the current scope of the literature with applications in achieving better spot estimates relative to other known instantaneous estimators. Another possible estimator that may prove useful is an extension to the Hayashi and Yoshida (2005) estimator making it robust to jumps. To the best of my knowledge, I have yet to see an estimator of this type.

Discriminating under lead-lag

Chapter 4 only focused on discriminating the underlying process under the presence of asynchrony. Designing experiments to detect the underlying process under all known sources of the Epps effect such as asynchrony, tick-size, and lead-lag can allow one to fully disentangle various aspects of the Epps effect while allowing one to more rigorously conclude the underlying nature of Trade and Quote data from financial markets. The necessary tools are available in Münnix et al. (2010, 2011), Mastromatteo et al. (2011), and Bacry et al. (2013a,b). The question that remains is how does one combine these various aspects together.

Epps effect on cluster analysis

Correlation estimates can be applied into unsupervised algorithms such as the Agglomerative Super-Paramagnetic Clustering algorithm (Yelibi and Gebbie, 2019). These algorithms usually take in static correlation estimate aggregated over an interval $[0,T]$ (integrated correlation). The instantaneous estimators from Chapter 3 allows one to obtain correlation estimates for any point in time $t \in [0,T]$. This allows one to visualise the resulting cluster dynamics through time. This can be interesting for financial market crashes, particularly the cluster dynamics within the day of the crash.

Another potentially interesting application is to examine the impact of the Epps effect on the cluster dynamics, particularly as a function of Δt. This can be performed for different sources of the Epps effect and may provide an alternative method of discerning the underlying system.

Closing thoughts

There are a few tips I found that really helped to streamline and speed up the process when writing pre-prints. First, one really needs a thorough knowledge of the existing literature in the field to know the basic methods that exists, what has been done and what is missing. There is no quick way around this except reading the literature and combing through the reference list of one of the most relevant papers for the topic. A helpful method for this is to find a review type of paper summarising the literature when researching a new topic. For example, when I was researching Hawkes processes, the paper by Bacry et al. (2015) really helped speeding up the process. Second, one should have a goal in mind when combing through the literature. For example, our goal was to see if the Epps effect behaves differently for Hawkes processes and diffusion processes. This lets one know what one is searching for in the literature. A helpful method is to concisely summarise papers one has read (for future reference) and to draw a mind-map in how a paper can fit into one's topic. This helps with inspiration in how these papers can answer a possibly related question (in my case it gave the inspiration that we could discriminate the processes using the Epps curves), seeing everything together like a jigsaw puzzle really makes it a lot easier. Finally, when running simulations, always save the results no matter how trivial. Never prematurely delete results, you never know when it can come in handy later on.

This dissertation was my first experience into research and my first time going through a peer review process. I learned that research is a long and drawn out process; gratification and recognition does not come quickly or easily, but it is extremely fun if one is passionate about their research.

Appendix A

Supplementary Algorithms

A.1 Malliavin-Mancino estimators

Require:
1. **T**: (n x D) matrix of sampled times. Non-trade times are represented using *NaNs* or *NAs*.

Set: $t_{\min}$ = minimum value of **T**
Set: $t_{\max}$ = maximum value of **T**
for i = 1 to D **do**
  for h = 1 to n_i **do**

$$\tilde{t}_h^i = \frac{2\pi(t_h^i - t_{\min})}{t_{\max} - t_{\min}}$$

  end for
end for
return ($\tilde{\mathbf{t}}$)

Algorithm 14: The time re-scaling Algorithm re-scales the trading times from $[0,T]$ to $[0,2\pi]$. The Julia implementation can be found in any of the Dirichlet or Fejér implementations in the GitHub resource (Chang et al., 2020c) and is an auxiliary function based on the MATLAB implementation from Malherbe et al. (2005) and Hendricks et al. (2017).

Require:
1. $\tilde{\mathbf{t}}$: (n x D) of re-scaled sampled times. Non-trade times are represented using *NaNs* or *NAs*.

Set: Δt_0^i = minimum distance between sampled times for asset i.
Set: $N_0^i = \frac{2\pi}{\Delta t_0^i}$.
Set: $N_i = \lfloor \frac{N_0^i}{2} \rfloor$, where $\lfloor x \rfloor$ denotes the floor of x.
Set: $N = \min_i\{N_i\}$.
return (N)

Algorithm 15: The Nyquist frequency Algorithm computes the Nyquist cutoff for the Malliavin-Mancino estimator. The Julia implementation can be found in any of the Dirichlet or Fejér implementations in the GitHub resource (Chang et al., 2020c) and is an auxiliary function based on the MATLAB implementation from Malherbe et al. (2005) and Hendricks et al. (2017).

A.2 Simulation

Require:

1. n: number of price points to simulate.
2. μ: (D x 1) vector of drift parameters.
3. $\mathbf{\Sigma}$: (D x D) covariance matrix.
4. λ: (D x 1) vector of Poisson intensities.
5. **a**: (D x 1) vector of mean.
6. **b**: (D x 1) vector of standard deviation.
7. logarithm of start price: (D x 1) vector of $X(0)$.

Procedure for the i^{th} feature:

1. Generate: $\mathbf{Z} \sim \mathcal{N}_D(\mathbf{0}, \mathbf{I}_{DxD})$.
2. Generate: $W \sim \mathcal{N}_1(0,1)$.
3. Generate: $N_i \sim \mathcal{P}(\lambda_i(t_{k+1} - t_k))$.
4. Set: $d = \mu_i - \frac{1}{2}\sigma_i^2 - \lambda_i\left(\exp\left(a_i - 0.5b_i^2\right) - 1\right)$.
5. Set: $M_i = a_i N_i + \sqrt{N_i} b_i W$.
6. Set: $X^i(t_{k+1}) = X^i(t_k) + \left(d(t_{k+1} - t_k) + \sqrt{t_{k+1} - t_k}\sum_{k=1}^{d} A_{ik} Z_k + M_i\right)$.

return $(\exp(\mathbf{X}))$

Algorithm 16: The Merton model Algorithm simulates a correlated multivariate Merton model using the Euler–Maruyama scheme. It is subject to the initial condition $X(0) =$ logarithm of the start price. $\mathbf{A}$ is the Cholesky decomposition of $\mathbf{\Sigma}$. The Julia implementation can be found at Merton Model.jl in the GitHub resource (Chang et al., 2020c) and was provided by Glasserman (2004).

Require:

1. n: number of price points to simulate.
2. M: (2x2) invertible matrix.
3. H: (2x2) invertible matrix.
4. b: (2x2) matrix such that $b - H^2 \in S_2^+$.
5. ρ: (2x1) vector of correlations between volatility and price.
6. X_0: (2x1) vector of logarithm of start price.
7. $\Sigma(0)$: (2x2) matrix of starting volatility, such that $\Sigma(0) \in S_2^+$.

For each iteration:

1. Simulate: B, a (2x2) matrix of standard normal.
2. Simulate: $W \sim \mathcal{N}_2(\mathbf{0}, \mathbf{I}_{2x2})$, vector of standard normal.
3. Set: $Z = \sqrt{1 - \rho^\top \rho} W + B\rho$
4. Set:

$$\Sigma(t_{k+1}) = \Sigma(t_k) + \left(b + M\Sigma(t_k) + \Sigma(t_k)M^\top\right)(t_{k+1} - t_k) + \sqrt{\Sigma(t_k)}BH\sqrt{t_{k+1} - t_k} + HB^\top\sqrt{\Sigma(t_k)}\sqrt{t_{k+1} - t_k}.$$

5. Set:

$$Xt_{k+1} = Xt_k - \frac{1}{2}\Sigma^{\text{diag}}(t_k)(t_{k+1} - t_k) + \sqrt{\Sigma(t_k)}Z\sqrt{t_{k+1} - t_k}.$$

return $(\exp(\mathbf{X})$ and $\Sigma(t))$

Algorithm 17: The Heston model Algorithm simulates a bivariate Heston model using the Euler–Maruyama scheme. The Julia implementation can be found at Heston.jl in the GitHub resource (Chang et al., 2020c).

Require:

1. n: number of price points to simulate.
2. M: (2x2) invertible matrix.
3. H: (2x2) invertible matrix.
4. b: (2x2) matrix such that $b - H^2 \in S_2^+$.
5. ρ: (2x1) vector of correlations between volatility and price.
6. X_0: (2x1) vector of logarithm of start price.
7. $\Sigma(0)$: (2x2) matrix of starting volatility, such that $\Sigma(0) \in S_2^+$.
8. X_0: (2x1) vector of logarithm of start price.
9. λ^{X^i}: rate of jumps in price process.
10. $\lambda^{\Sigma^{11}}$: rate of jumps in the volatility process.
11. **a**: (2x1) vector of mean.
12. **b**: (2x1) vector of standard deviation.
13. θ: size of exponential jumps.

For each iteration:

1. Generate: $N^{\Sigma^{11}} \sim \mathscr{P}\left(\lambda^{\Sigma^{11}}(t_{k+1} - t_k)\right)$.
2. Generate: $N^{X^i} \sim \mathscr{P}\left(\lambda^{X^i}(t_{k+1} - t_k)\right)$.
3. Simulate: Z^J, a (2x1) matrix of standard normal.
4. Simulate: B, a (2x2) matrix of standard normal.
5. Simulate: $W \sim \mathscr{N}_2(\mathbf{0}, \mathbf{I}_{2x2})$, vector of standard normal.
6. Set: $Z = \sqrt{1 - \rho^\top}\rho W + B\rho$
7. Set:

$$\Sigma(t_{k+1}^-) = \Sigma(t_k^+) + \left(b + M\Sigma(t_k^+) + \Sigma(t_k^+)M^\top\right)(t_{k+1} - t_k) + \sqrt{\Sigma(t_k^+)}BH\sqrt{t_{k+1} - t_k} + HB^\top\sqrt{\Sigma(t_{k^+})}\sqrt{t_{k+1} - t_k}.$$

8. Set: $\Sigma(t_{k+1}^+) = \Sigma(t_{k+1}^-)$, and $\Sigma^{11}(t_{k+1}^+) = \Sigma^{11}(t_{k+1}^-) + \sum_{j=1}^{N^{\Sigma^{11}}} E_j$, where $E_j \sim \mathscr{E}(\theta)$.
9. Set: $b_{s,i} = -\frac{1}{2}\Sigma^{ii} - \lambda^{X^i}\left(\exp\left(a_i - 0.5b_i^2\right) - 1\right)$.
10. Set: $J_i = a_i N^{X^i} + \sqrt{N^{X^i}}b_i Z_i^J$.
11. Set:

$$Xt_{k+1} = Xt_k + b_s(t_{k+1} - t_k) + \sqrt{\Sigma(t_k^-)}Z\sqrt{t_{k+1} - t_k} + J.$$

return $(\exp(\mathbf{X})$ and $\Sigma(t^+))$

Algorithm 18: The Bates type model Algorithm simulates a bivariate Bates type model using the Euler–Maruyama scheme. The Julia implementation can be found at Bates2D.jl in the GitHub resource (Chang et al., 2020c).

Require:

1. n: number of price points to simulate.
2. μ: (D x 1) vector of drift parameters.
3. $\mathbf{\Sigma}$: (D x D) covariance matrix.
4. start price: (D x 1) vector of $P(0)$.

Procedure for the i^{th} feature:

1. Generate: $\mathbf{Z} \sim \mathscr{N}_D(\mathbf{0}, \mathbf{I}_{DxD})$.
2. Set: $P^i(t_{k+1}) = P^i(t_k)\exp\left[(\mu_i - \frac{1}{2}\sigma_i^2)(t_{k+1} - t_k) + \sqrt{t_{k+1} - t_k}\sum_{k=1}^d A_{ik}Z_k\right]$.

return (**P**)

Algorithm 19: The Geometric Brownian Motion (GBM) Algorithm simulates a correlated multivariate GBM using the Euler–Maruyama scheme. It is subject to the initial condition $S(0) =$ start price. **A** is the Cholesky decomposition of $\mathbf{\Sigma}$. The Julia implementation can be found at GBM.jl in the GitHub resource (Chang et al., 2020c) and was provided by Glasserman (2004).

A.3 Miscellaneous

Require:
1. d: (Integer) the dimension of the random correlation matrix.

Set: $\mathbf{A}$ = populate a (dxd) matrix with random uniforms.
Compute: $\mathbf{A} = \mathbf{A}\mathbf{A}^T$.
Compute: $\mathbf{A} = d \cdot \mathbf{I}_d$.
Set: $\sigma = (\sigma_i)_{i=1}^d$ where $\sigma_i = \sqrt{A_{ii}}$.
Set: $\rho_{ij} = \frac{A_{ij}}{\sigma_i \cdot \sigma_j}$
return (ρ)

Algorithm 20: The Random Correlation Algorithm generates a random d-dimensional correlation matrix. The function can be found in gencovmatrix.jl provided in our GitHub resources (Chang et al., 2020c).

Require:
1. d: (Integer) the dimension of the random correlation matrix.
2. $\sigma = (\sigma_i)_{i=1}^d$: $(dx1)$ vector of standard deviations.

Obtain: ρ from Algorithm 20
Set: $\Sigma_{ij} = \rho_{ij} \cdot \sigma_i \cdot \sigma_j$.
return (Σ)

Algorithm 21: The Random Covariance Algorithm generates a random d-dimensional covariance matrix. The function can be found in gencovmatrix.jl provided in our GitHub resources (Chang et al., 2020c).

Appendix B

Hawkes Process

This appendix serves as a short introduction (technical document) to Hawkes processes. Hawkes (1971) introduced a class of multivariate point process allowing for event-occurrence clustering through a stochastic intensity vector. The stochastic intensity is composed from an *exogenous* or *baseline intensity* component which is not influenced by prior events, and an *endogenous* component where prior events lead to an increased intensity (achieving the event-occurrence clustering). Within the endogenous component, *self-excitation* refers to an event type leading to more of the same event and *mutual-excitation* is an event driving the occurrence of other event types. Applications of the Hawkes process is plentiful, ranging from seismology (See Ogata (1988, 1999)) to social media (See Rizoiu et al. (2017)) and finance (See Hawkes (2018); Bacry et al. (2015) and the references therein). The application of the Hawkes process here is to construct a price model from bottom-up events and also applying the Hawkes process to model the arrival of asynchronous trades.

B.1 Definition

Consider a M-variate counting process $N(t) = \{N_m(t)\}_{m=1}^M$. The associated stochastic intensity $\lambda(t) = \{\lambda^m(t)\}_{m=1}^M$ takes the form:

$$\lambda^m(t) = \lambda_0^m(t) + \sum_{n=1}^{M} \int_{-\infty}^{t} \phi^{mn}(t-s) dN_s^n, \tag{B.1}$$

where $\lambda_0^m(t)$ is the time dependent baseline intensity for the m^{th} component and $\phi^{mn}(t)$ is the kernel function which governs the dependency of prior events to the current time t. Furthermore, $\mathbf{\Phi}(t) = \{\phi^{mn}(t)\}_{m,n=1}^M$ is such that the following conditions hold (Bacry et al., 2015):

(i) Component-wise positive, *i.e.* $\phi^{mn}(t) > 0$ for each $1 \leq m,n \leq M$;

(ii) Component-wise causal, *i.e.* if $t < 0$, then $\phi^{mn}(t) = 0$ for each $1 \leq m,n \leq M$;

(iii) Each component $\phi^{mn}(t)$ belongs to the space of L^1-integrable functions.

The popular kernels used include (Bacry et al., 2015) the exponential kernel:

$$\phi^{mn}(t) = \alpha^{mn} e^{-\beta^{mn} t} \mathbb{1}_{t \in \mathbb{R}^+}, \tag{B.2}$$

and the power law kernel:

$$\phi^{mn}(t) = \frac{\alpha^{mn} \beta^{mn}}{(1+\beta^{mn} t)^{1+\gamma^{mn}}} \mathbb{1}_{t \in \mathbb{R}^+}. \tag{B.3}$$

For $\alpha^{mn}, \beta^{mn} > 0$ in eqs. (B.2) and (B.3), we have that the Hawkes process is well defined. Here, let us consider the simplest case of the Hawkes process used by Bacry et al. (2013a): the exponential kernel with a constant baseline intensity λ_0^m.

B.2 Stability condition

Let the matrix of *branching ratios* be:

$$\mathbf{\Gamma} = \|\Phi\| = \{\|\phi^{mn}\|\}_{m,n=1}^{M}, \tag{B.4}$$

where $\|\phi^{mn}\|$ denotes the L^1-norm defined as $\int_0^\infty \phi^{mn}(t)dt$. Thus, for the exponential kernel we have $\mathbf{\Gamma} = \{\alpha^{mn}/\beta^{mn}\}_{m,n=1}^{M}$. The spectral radius of matrix $\mathbf{\Gamma}$ is defined as $\rho(\mathbf{\Gamma}) = \max_{a \in \mathscr{S}(\mathbf{\Gamma})} |a|$, where $\mathscr{S}(\mathbf{\Gamma})$ is the set of all eigenvalues of $\mathbf{\Gamma}$. The branching ratio can be thought of as the number of children event a parent event will have, thus the three phases of a Hawkes process is given by (Martins and Hendricks, 2016):

i.) $\rho(\mathbf{\Gamma}) > 1$, the process is *non-stationary*, or *super-critical*, where we will have an explosion of events.

ii.) $\rho(\mathbf{\Gamma}) = 1$, the process is *quasi-stationary*, or *critical*, where we will have a family that will live indefinitely without exploding.

iii.) $\rho(\mathbf{\Gamma}) < 1$, the process is *stationary*, or *sub-critical*, where if there is no exogenous component, each family will eventually die out.

Thus the condition to ensure stationarity is that the spectral radius of $\mathbf{\Gamma}$ must be strictly less than 1.

B.3 Simulation

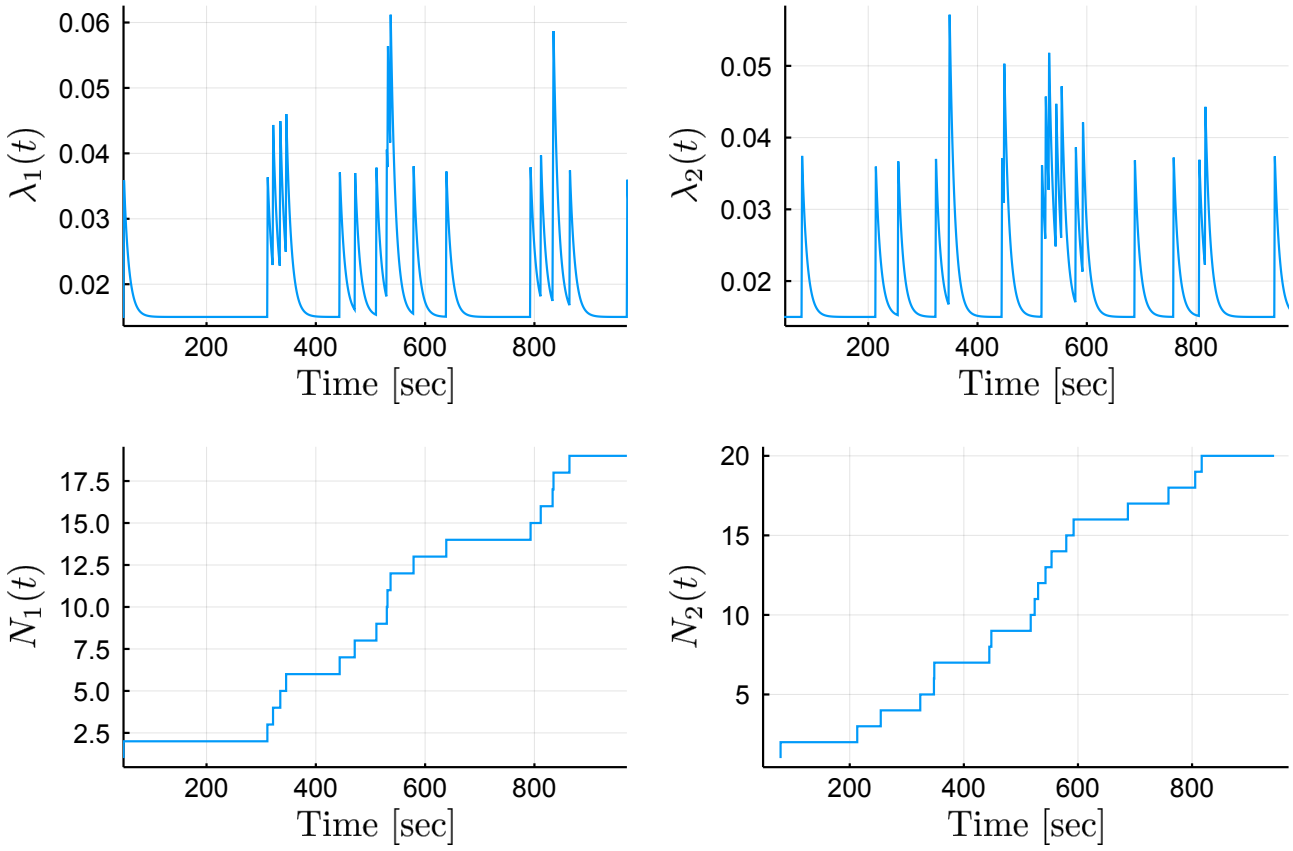


Figure B.1: The figure demonstrates a 2-variate Hawkes process and its associated intensity. $\mathbf{\Phi}$ takes the form of eq. (4.26), with parameters $(\lambda_0^1 = \lambda_0^2, \alpha^{(s)}, \beta^{(s)}) = (0.015, 0.023, 0.11)$ for $T = 1000$ seconds.

A number of simulation techniques have been developed over the years and each exploit different properties of the Hawkes process. Ogata (1981) developed the thinning procedure, followed by Møller and Rasmussen (2005) using the cluster representation and more recently Dassios and Zhao (2013) used the time-change method for the exponential kernel.

Here I use the thinning procedure in Algorithm 22 by Toke and Pomponio (2012). The thinning procedure can be summarized as follows: let $I^k(t) = \sum_{k=1}^{K} \lambda^k(t)$ and let t be the current time. We sample an exponential inter-arrival time with rate parameter $I^M(t)$, call it τ. A random uniform u is then sampled from $[0, I^M(t)]$, and if $u < I^M(t) - I^M(t+\tau)$ the arrival time $t+\tau$ is rejected. Otherwise, the arrival time $t+\tau$ is accepted and attributed to the i^{th} component, where i is such that $I^{i-1}(t+\tau) < u \leq I^i(t+\tau)$. The actual algorithm re-scales the cumulative intensities $I^k(t)$ so that we only need to simulate random uniforms between $[0,1]$.

The benefit for using the exponential kernel comes in when simulating and estimating the Hawkes process. This is because $\lambda(t)$ can be recast into Markovian form with the exponential kernel (Bacry et al., 2015). Meaning that keeping track of the entire history is unnecessary. Therefore reducing the computation of the likelihood from $\mathcal{O}(n^2 M)$ to $\mathcal{O}(nM)$, where n is the total number of arrival times.

Require:

1. λ: (M x 1) vector of baseline intensity.
2. α: (M x M) matrix of excitation.
3. β: (M x M) matrix of rates of relaxation.
4. T: time horizon of simulation.

Step 1. Initialisation.

1.1. Set $i = 1, i^1 = 1, ..., i^M = 1$.

1.2. Set $I^* = I^M(0) = \sum_{k=1}^{M} \lambda^k(0)$.

Step 2: Generate first event.

2.1. Generate $U \sim \mathcal{U}_{[0,1]}$, set $s = -\frac{1}{I^*} \ln U$.

2.2. **If** $s > T$ **Then** go to step 4.

2.3. Generate $D \sim \mathcal{U}_{[0,1]}$, set $t_1^{n_0}$ where n_0 is such that $\frac{I^{n_0-1}(0)}{I^*} < D \leq \frac{I^{n_0}(0)}{I^*}$.

2.4. Set $t_1 = t_1^{n_0}$.

Step 3: General routine.

3.1. Set $i^{n_0} = i^{n_0} + 1$ and $i = i + 1$.

3.2. Set $I^* = I^M(t_{i-1}) + \sum_{k=1}^{M} \alpha^{k n_0}$.

3.3. Generate $U \sim \mathcal{U}_{[0,1]}$, set $s = -\frac{1}{I^*} \ln U$.

3.4. **If** $s > T$ **Then** go to step 4.

3.5. Generate $D \sim \mathcal{U}_{[0,1]}$.

3.5.1 **If** $D \leq \frac{I^M(s)}{I^*}$,

3.5.2 **Then** set $t_{i^{n_0}}^{n_0} = s$ where n_0 is such that $\frac{I^{n_0-1}(0)}{I^*} < D \leq \frac{I^{n_0}(0)}{I^*}$, and $t_i = t_{i^{n_0}}^{n_0}$ and repeat step 3.

3.5.3 **Else** update $I^* = I^M(s)$ and repeat from step 3.3.

Step 4: **return** $(\{\{t_i^n\}_i\}_{n=1,...,M})$

Algorithm 22: The Hawkes Simulation Algorithm simulates a M-variate Hawkes process using the thinning procedure by Ogata (1981). The algorithm is from Toke and Pomponio (2012). The Julia implementation can be found at Hawkes.jl in the GitHub resource (Chang et al., 2020c).

I included a simple demonstration of the simulation in Figure B.1. The figure plots a 2-variate Hawkes process with its associated intensity vector. The figure uses $\mathbf{\Phi}$ which takes the form of eq. (4.26), with parameters $(\lambda_0^1 = \lambda_0^2, \alpha^{(s)}, \beta^{(s)}) = (0.015, 0.023, 0.11)$ for $T = 1{,}000$ seconds.

B.4 Calibration

Log-likelihood

As with the simulation, there exists a variety of methods to calibrate the Hawkes process. For a more detailed discussion on the various methods, I refer the reader to Appendix C of Bacry et al. (2015). Here I discuss the more popular parametric approach using the Maximum Likelihood Estimation (MLE) introduced by Ogata and Akaike (1982). The log-likelihood for the Hawkes process is defined as:

$$\ln \mathscr{L}\left(\{N(t)\}_{t\leq T}\right) = \sum_{m=1}^{M} \ln \mathscr{L}^m\left(\{N_m(t)\}_{t\leq T}\right), \tag{B.5}$$

with each term defined as:

$$\ln \mathscr{L}^m\left(\{N_m(t)\}_{t\leq T}\right) = \int_0^T (1-\lambda^m(s))ds + \int_0^T \ln \lambda^m(s) dN_m(s). \tag{B.6}$$

Using the exponential kernel, eq. (B.6) can be computed as:

$$\begin{aligned}\ln \mathscr{L}^m\left(\{N_m(t)\}_{t\leq T}\right) &= T - \int_0^T \lambda^m(s)ds \\ &+ \sum_{i:t_i\leq T} \mathbb{1}_{\{Z_i=m\}} \ln\left[\lambda_0^m(t_i) + \sum_{n=1}^{M}\sum_{t_k^n<t_i} \alpha^{mn} e^{-\beta^{mn}(t_i-t_k^n)}\right].\end{aligned} \tag{B.7}$$

Furthermore, eq. (B.7) can be computationally optimized using the Markov property by defining:

$$\begin{aligned}R^{mn}(l) &= \sum_{t_k^n<t_l^m} e^{-\beta^{mn}(t_l^m-t_k^n)} \\ &= \begin{cases} e^{-\beta^{mn}(t_l^m-t_{l-1}^m)} R^{mn}(l-1) + \sum_{t_{l-1}^m\leq t_k^n<t_l^m} e^{-\beta^{mn}(t_l^m-t_k^n)} & \text{if } m\neq n, \\ e^{-\beta^{mn}(t_l^m-t_{l-1}^m)}\left(1+R^{mn}(l-1)\right) & \text{if } m=n, \end{cases}\end{aligned} \tag{B.8}$$

subject to $R^{mn}(0)=0$.

Including eq. (B.8) in eq. (B.7) gives us:

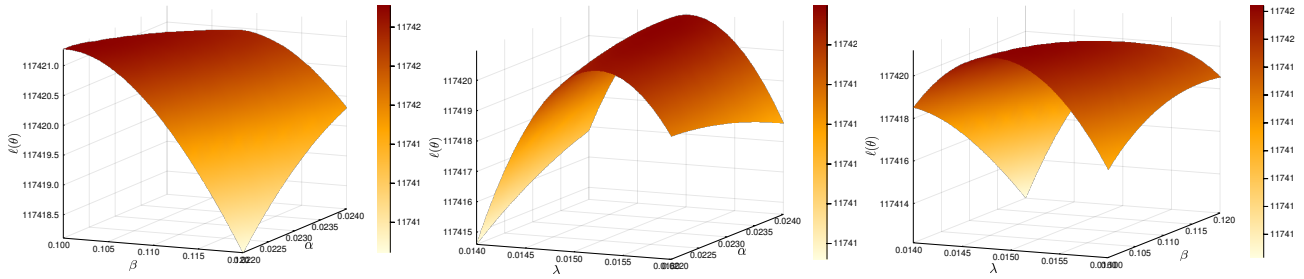
$$\begin{aligned}\ln \mathscr{L}^m\left(\{N_m(t)\}_{t\leq T}\right) &= T - \int_0^T \lambda^m(s)ds \\ &- \sum_{i:t_i\leq T}\sum_{m=1}^{M} \frac{\alpha^{mn}}{\beta^{mn}}\left(1-e^{-\beta^{mn}(T-t_i)}\right) \\ &+ \sum_{l:t_l^m\leq T} \ln\left[\lambda^m(t_l^m) + \sum_{n=1}^{M}\alpha^{mn}R^{mn}(l)\right].\end{aligned} \tag{B.9}$$

Now in order to calibrate the Hawkes process, we can simply maximize the log-likelihood:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ \ln \mathscr{L}(\theta),$$

where θ refers to the parameters of the Hawkes process which can include $(\lambda_0^m, \alpha^{mn}, \beta^{mn})$, depending on the specification of the model.

I include a simple example of the log-likelihood surface from eq. (B.5). The log-likelihood has its observations from a 2-variate Hawkes process with $\mathbf{\Phi}$ taking the form of eq. (4.26) for $T = 18$ hours. Each surface plots two parameters at a time, where $\theta = (\lambda_0^1 = \lambda_0^2, \alpha^{(s)}, \beta^{(s)})$ and the remaining parameter is fixed at its *true* value from the simulation. The true parameters are $(\lambda_0^1 = \lambda_0^2, \alpha^{(s)}, \beta^{(s)}) = (0.015, 0.023, 0.11)$.



(a) β and α **(b)** λ and α **(c)** λ and β

Figure B.2: The figure plots the log-likelihood surface using eq. (B.5). The log-likelihood is based off the observations from a 2-variate Hawkes process with $\mathbf{\Phi}$ taking the form of eq. (4.26) for $T = 18$ hours. Each surface plots two dimensions of the parameter space $\theta = (\lambda_0^1 = \lambda_0^2, \alpha^{(s)}, \beta^{(s)})$, with the remaining dimension fixed at its *true* value from the simulation. The true parameters are $(\lambda_0^1 = \lambda_0^2, \alpha^{(s)}, \beta^{(s)}) = (0.015, 0.023, 0.11)$.

Mis-specification testing

Given the various ways one can specify a Hawkes process, such as: different kernel choices and different structures for $\mathbf{\Phi}$, we need a method to test the specification. Bowsher (2007) provided a solution for this in a theorem on multivariate random time change.

Theorem B.4.1 *Theorem 4.1 of Bowsher (2007) (rephrased):*
Consider the M sequences of generalised residuals $\{e_i^{(m)}(\theta)\}_{m=1}^{M}$, *where*

$$e_i^{(m)}(\theta) = \int_{t_i^m}^{t_{i+1}^m} \lambda^m(s;\theta)ds. \tag{B.10}$$

When θ *is the true set of parameters, then each sequence* $e_i^{(m)}(\theta)$ *will be an independently distributed exponentially random variable with unit mean.*

Therefore, we can test for mis-specification by checking if the estimated generalised residuals $e_i^{(m)}(\hat{\theta})$ are indeed independently distributed exponential random variables with unit mean. This can be tested using statistical tests such as Kolmogorov-Smirnov, Ljung-Box Q, Excess Dispersion and Kwiatkowski-Phillips-Schmidt-Shin (Martins and Hendricks, 2016).

Empirical difficulties

Apart from the difficulties in determining the correct specification, estimation, and parameterisation; often when calibrating the Hawkes process to empirical data, one faces the issue of events falling within the same second (Large, 2007). This is the case when using data from Bloomberg Pro (see Appendix C.2) where measured data is rounded to the nearest second. This is problematic because in the framework of the Hawkes process, contemporaneous events can never occur. There are three methods to fix this problem (Large, 2007). The choice used will affect the estimation, specifically events that occur together within timescales of one second.

The first approach is to thin the data, so that only one event is retained at each time stamp. This is problematic because these contemporaneous events are prevalent and can take up to 40% of the data (see Large (2007)).

The second approach is to add a random uniform between $[-0.5, 0.5]$ (measured in seconds) to the contemporaneous time stamps. This however is problematic when one is trying to model the Limit Order Book (LOB) where the ordering of events are important. These two approaches are to ensure that the data is a simple point process (Large, 2007).

The last approach used by Large (2007) is to define $\phi(0) \equiv 0$ and maximize the log-likelihood without altering the data. This ensures the events at contemporaneous time stamps have no bearing on each other (which can be problematic, if one wants to faithfully recover the correct effects). However, doing so will yield many durations of length zero in eq. (B.10). This can affect the ability to test for mis-specification. Large (2007) overcomes this by discarding any term in the sequence eq. (B.10) which was less than 0.05, and subtracting 0.05 from the remaining terms in the sequence. This procedure is justified because if E is an exponential random variable with mean 1, then $E - 0.05$ truncated to $\mathbb{R}^+$ will also be an exponential random variable with mean 1.

The best method to overcome these issue is to find a data source with more accurate measurements, such as Thomson Reuters Tick History (TRTH) used in Martins and Hendricks (2016).

Appendix C

Data Engineering

The data sets used in this dissertation was inherited from my Honours project Chang et al. (2019a). The aggregation of raw data (See Appendix C.2) was performed in R and the code listing can be found in Bukuru et al. (2019), while the cleaning for the previous tick interpolation (See Appendix C.3) was performed in JuliaPro and the code listing can be found in Chang et al. (2020c).

C.1 Data Types

The Trade and Quote (TAQ) data presents three types of observables: the bid, ask and actual trades. The bids and ask are discarded because the work here is focused on modelling the trade data, rather than the mid quotes. I retain only the actual trades as they form the discrete samples of prices seen in high-frequency finance. Within the trades, there are various types. Examples include the Automated Trade (AT), Late Trade (LT), Post Contra Trade (LC) and Indicative Auction Information (IP) (JSE). I retain only the Automated Trades because only these form the continuous trading process. The other trade types are after-hour trades (LT), correction of previous days published off book trade (LC) and an indicative auction price based on the volume maximising auction algorithm used to determine the auction uncrossing price (IP) (JSE) - which is irrelevant to the analysis.

C.2 Aggregation

An issue with Bloomberg data which is not present with Thomson Reuters data is that timestamps are only shown up to seconds, therefore there are multiple trades with the same timestamp—illustrated in Figure C.1. This poses an issue for the Malliavin-Mancino and Hayashi-Yoshida estimators since they require a unique time stamp for each trade.

	times	type	value	size	condcode
3887	2018-12-28 12:26:39	TRADE	45337.15	162	AT
3888	2018-12-28 12:26:39	TRADE	45338.11	165	AT
3889	2018-12-28 12:26:39	TRADE	45338.11	200	AT
3890	2018-12-28 12:26:39	TRADE	45338.11	50	AT
3891	2018-12-28 12:26:39	TRADE	45338.11	53	AT
3892	2018-12-28 12:26:39	TRADE	45338.11	36	AT
3893	2018-12-28 12:26:39	TRADE	45338.11	169	AT
3894	2018-12-28 12:26:39	TRADE	45338.11	69	AT
3895	2018-12-28 12:26:39	TRADE	45338.11	35	AT
3896	2018-12-28 12:26:39	TRADE	45342.95	21	AT
3897	2018-12-28 12:26:39	TRADE	45342.95	19	AT
3898	2018-12-28 12:26:39	TRADE	45342.95	70	AT

Figure C.1: The figure illustrates some trades which have the same time stamp.

To overcome this issue, trades with the same time stamps need to be aggregated. The aggregation algorithm is presented in Algorithm 23 and it uses a Volume Weighted Average Price (VWAP) method

of aggregation. VWAP is employed because it gives a better representation of the data given by the fact that it weights each trade by the volume which is directly linked to the price impact (Easley et al., 2008).

Require:

1. T_i the trading times, $i = 1,...,N$
2. P_i the observed prices, $i = 1,...,N$
3. V_i the volume associated with the trade, $i = 1,...,N$

Identify the unique trading times t_j^*, $j = 1,...M$

Gather trades with the same trading times into a set J_j, $j = 1,...,M$

Procedure for the j^{th} set:

1. Set
$$p_j^* = \frac{\sum_{i \in J_j} \text{price}_i \cdot \text{volume}_i}{\sum_i \text{volume}_i}$$
2. Set $V_j^* = \sum_{i \in J_j} V_i$

return ($T^* = \{t_j^*\}_{j=1}^M, P^* = \{p_j^*\}_{j=1}^M, V^* = \{V_j^*\}_{j=1}^M$)

Algorithm 23: The algorithm aggregates trades with the same time stamp using a volume weighted average price. The implementation can be found in AsynchronousData.R.

Figure C.2 demonstrates the output of Algorithm 23 using the trades with the same time stamp in Figure C.1.

	times	type	value	size
1220	2018-12-28 12:26:39	TRADE	45338.47	1049

Figure C.2: Trades aggregated into a unique time stamp.

C.3 Previous tick interpolation

A method to synchronise the asynchronous tick-by-tick trades is to apply the previous tick interpolation for interval size Δt. Let X_t^i be the underlying process of asset i at time t. Let the set of asynchronous arrival times be given by $U^i = \{t_k^i\}_{k \in \mathbb{Z}}$. Therefore the process X^i is only observed at times $t \in U^i$. Now in order to construct the synchronised process, we need to define $\gamma_i(t) = \max\{t_k^i : t_k^i \leq t\}$ for $t \in [0,T]$. Now let the interval size (discretisation size) be Δt, then let the new synchronous grid be $0 = t_{0,\Delta t} < t_{1,\Delta t} < ... < t_{j,\Delta t} < ... < t_{n-1,\Delta t} < t_{n,\Delta t} = n\Delta t$ where $t_{j,\Delta t} - t_{j-1,\Delta t} = \Delta t$ and $n = \lfloor \frac{T}{\Delta t} \rfloor$. Then let the synchronised asynchronous process be defined as:

$$\tilde{X}^i_{t_{j,\Delta t}} = X^i_{\gamma_i(t_{j,\Delta t})}.$$

The implementation can be summarised in the following three steps:

i.) loop through $t_{j,\Delta t}$, for $j = 0,...,n$,

ii.) compute $\gamma_i(t_{j,\Delta t}) = \max\{t_k^i : t_k^i \leq t_{j,\Delta t}\}$,

iii.) set the j^{th} bin of the synchronised process $\tilde{X}^i_{t_{j,\Delta t}}$ as $X^i_{\gamma_i(t_{j,\Delta t})}$, where X_t^i are the observables.

The remaining detail is to synchronise $t = 0$ for the assets. This is because the first trade of each asset do not usually line up with each other (sometimes they do, and then it is a simple case of setting $t = 0$ when this occurs). The choice I make is to start the clock when both assets have each made their first trade. This is because the assets do not necessarily make their first trade at exactly 09:00:00 (when trading starts). There is usually a few seconds of delay before the first trade arrives. Therefore, $t = 0$ is set to be the first trade of the lagging asset. Let us consider an example to illustrate this.

Figures C.3 and C.4 demonstrates two possible scenarios. Here we have asset 1 ($\otimes$) as the lagging asset so $t = 0$ is set to be the first trade of asset 1 (the green $\otimes$). Figure C.3 demonstrates the first scenario when there is no trade from asset 2 at $t = 0$. The value asset 2 takes on at $t = 0$ is thus the blue $\oplus$ using the previous tick interpolation. Figure C.4 demonstrates the second scenario when there is a trade from asset 2 at $t = 0$. The value asset 2 takes on at $t = 0$ is thus the value of the new trade (the blue $\oplus$). These examples demonstrates how to synchronise $t = 0$ for the empirical data. Obviously the scenarios can be the other way around when asset 2 is the lagging asset. Hence, we have a total of 4 possible scenarios to consider.

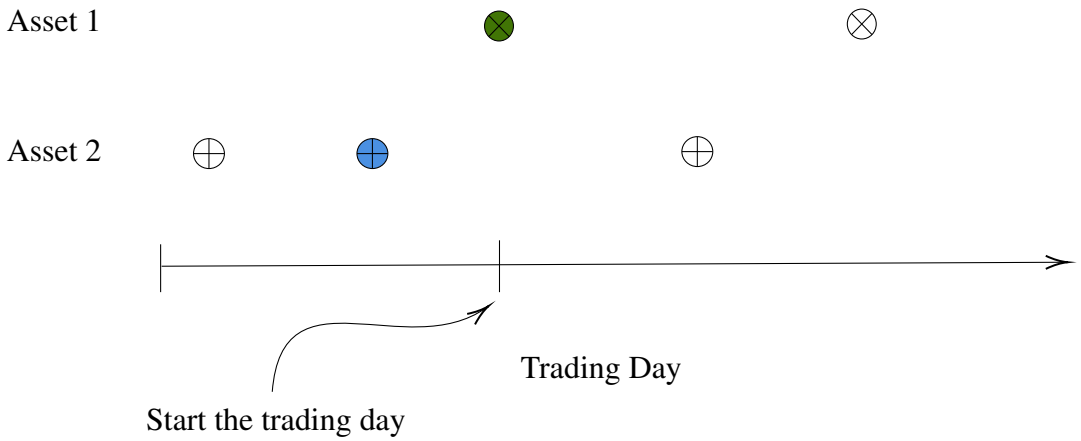


Figure C.3: The figure is a toy example to demonstrate how to synchronise $t = 0$ for two assets assuming asset 1 is the asset with the later first trade. Asset 1's trades are given by $\otimes$ and asset 2's trades are given by $\oplus$. Here $t = 0$ will be the green $\otimes$ as it is the lagging asset. The value asset 2 takes at that time will be the blue $\oplus$ from the previous tick interpolation.

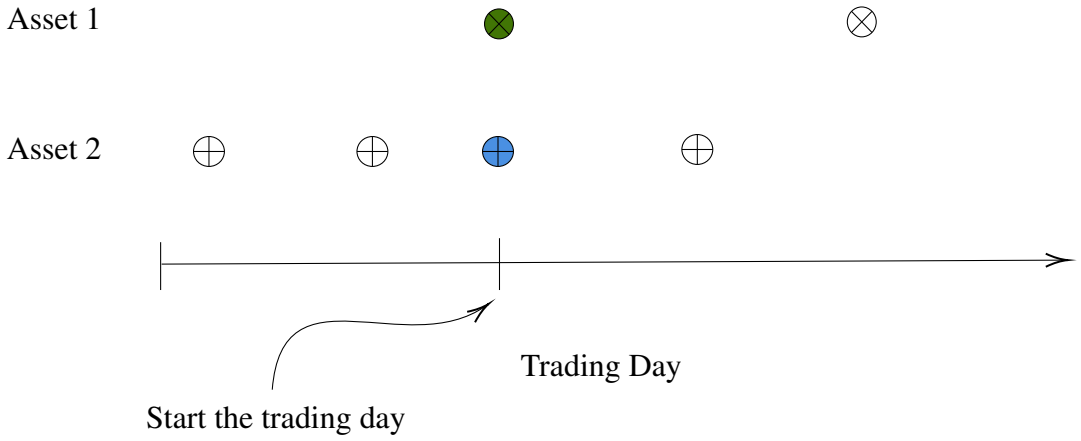


Figure C.4: The figure is a toy example to demonstrate how to synchronise $t = 0$ for two assets assuming asset 1 is the asset with the later first trade. Asset 1's trades are given by $\otimes$ and asset 2's trades are given by $\oplus$. Here $t = 0$ will be the green $\otimes$ as it is the lagging asset. The value asset 2 takes at that time will be the new trade blue $\oplus$ which occurred at $t = 0$.

The last point to mention is that each trading day has $T = 28,200$ seconds. This is 7 hours and 50 minutes because the last 10 minutes of the 8 hour trading day is the closing auction. Given that $t = 0$ can start slightly later, the "trading time" may be extend into the closing auction. This presents no issues because the data set only consists of Automated Trades, all the information regarding the closing auction between 16:50:00 and 17:00:00 has been removed. Therefore, the prices here are synchronised from the last trade within the regular trading hours (before 16:50:00) using previous tick interpolation.

Appendix D

JuliaPro usage

This appendix will serve as a technical document to guide a user through the basics of JuliaPro and serve as a simple demonstration on how to use the various functions built for the dissertation.

D.1 What is Julia?

Julia is an open source dynamically type scripting language that uses a Just In Time (JIT) compiler allowing it to achieve speeds similar to that of C. Julia at some level seems to solve the Ousterhout's dichotomy (Ousterhout, 1998), where the world of application development is divided into two groups: system programming languages and scripting languages. Each group has their distinct properties and use-case.

First, system programming languages such as C or Fortran are notoriously hard to use but extremely fast because it compiles code into machine language. In the case of C and Fortran, the code is compiled ahead of time. These languages are not user friendly as they are usually statically typed and require memory management. These languages are usually used for applications with large amounts of internal functionality such as operating systems and web browsers. Second, scripting languages such as Python is easy to use but slow because it interprets the code rather than compiling it into machine language. These programs usually offer user friendly functionalities such as dynamic typing, garbage collection and memory management. These languages are also known as *glue languages* and typically used for adding additional functionality on top of existing programs such as web page generation.

For context in the use-case of science, algorithms such as fast Fourier transforms are usually written in C or Fortran and are linked into languages such as Python for performing analysis. Julia offers a balance between these by using the JIT compiler where the code is compiled at run-time. MATLAB also offers the JIT compiler but is not an open source program.

D.2 Basic set up

Installation

JuliaPro can be downloaded from https://juliacomputing.com/products/juliapro. As with installing any program, simply follow the instructions to install. By default, the Juno IDE is bundled with JuliaPro. No separate installation is required. Upon launching JuliaPro, you will be presented with the Juno IDE (See Figure D.1). To start Julia, press enter in the REPL pane.

Remark D.2.1 *In my opinion, the Juno IDE is one of the best out there. Significantly better than R Studio. It is integrated with GitHub, streamlining the workflow. It has dynamic auto-completion providing suggestions for the code making coding that much simpler. Moreover, I argue the best feature it has is letting the user know where the code is breaking. This makes debugging and fixing code so much faster than for example, R Studio where the errors are simply stated.*

Remark D.2.2 *At JuliaCon 2020, the developers revealed that they are no longer developing Juno and it will only be in maintenance mode as development around Atom has slowed down during recent years. They decided to move to Visual Studio Code for future development.*

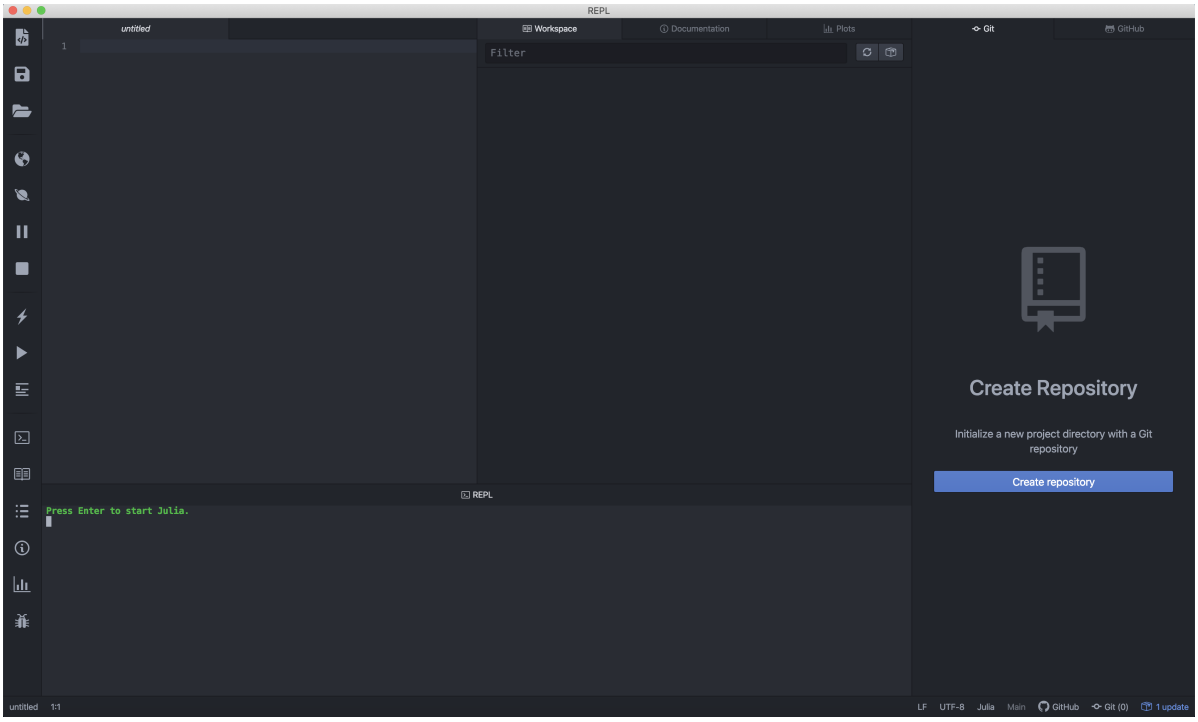


Figure D.1: Juno IDE.

Packages in Julia are all handled by the builtin package manager *Pkg*. By default, JuliaPro will download all the packages from https://pkg.juliacomputing.com/. This website requires authentication, therefore you will be asked to authenticate to receive a token (See Figure D.2). The authentication can be done using either a GitHub, Google or Linkedin account. Once this is done, you will have access to the range of packages made for JuliaPro.

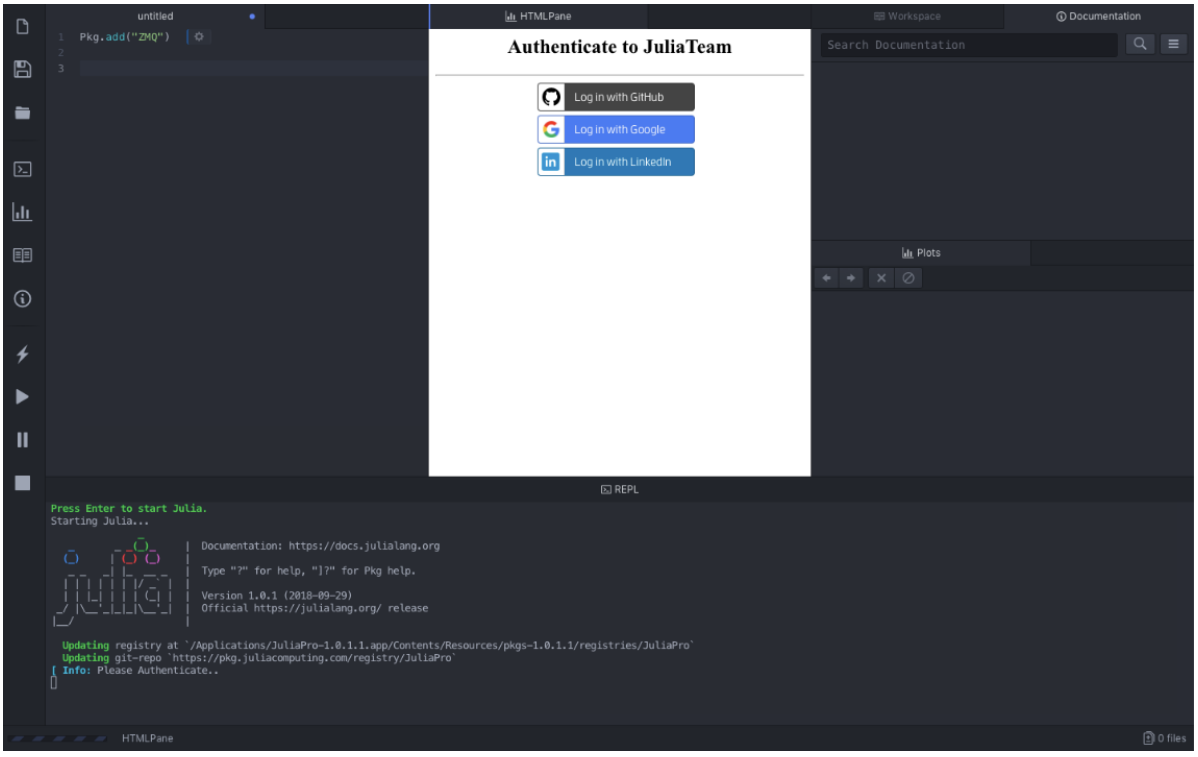


Figure D.2: Authentication prompt.

Registry management

One of the main issues I had with loading packages was the fact that I had two registries. This prevented me from loading any packages. To check that you only have one registry simply press `]` in the REPL to toggle *Pkg* operations. Then type in `registry st` to check the status of your registry.

Figure D.3: Checking the registry.

Figure D.3 we see that there are two registries: General and JuliaPro. We want to keep the General registry. In order to remove the JuliaPro registry, simply type `registry rm JuliaPro` (See Figure D.4). If the general registry is accidentally removed, you can add it back by simply typing `registry add https://github.com/JuliaRegistries/General`.

Figure D.4: Removing the registry.

Adding packages

Once the installation, authentication and registry setup has been completed; packages can now be added. In order to add a package, toggle to the *Pkg* operations and type `add PackageName` (See Figure D.5 for example adding the LinearAlgebra package). To toggle back to REPL from *Pkg*, simply press backspace. Once a package has been added, it needs to be loaded. This is done by typing `using PackageName` in the REPL.

Figure D.5: Adding a package.

Remark D.2.3 *The package that gave me the hardest time to install is the FINUFFT package by af Klinteberg (2018). This was no fault of the author but mine. The README.md very clearly states that the build script is hardwired to GCC 8 (it has since been changed to GCC 9). Therefore, I will briefly outline how to install GCC 8 on Mac-OS. First, Homebrew must be installed by following this link: https://brew.sh/. The website will provide a line of code which needs to be entered into the terminal. Once Homebrew has been installed, GCC 8 is installed by typing* `brew install gcc@8` *into the terminal.*

Plotting tips

In order to make plots in Julia, we need to first add and load the Plot package. Julia plots have several backends including: GR, PyPlot, Plotly and UnicodePlots to name a few. I use the GR backend which is not as well supported in the community as PyPlot. Therefore, I will try include some code listings of the plotting tricks I have found which are not very well documented. There is a very good documentation for the plots package in: tutorial. It covers all the basics and provides sufficient examples to perform the necessary basic plots.

Basics

Plotting in Julia is quite similar to ggplot in R. Plots can be stored as objects and additional arguments can be added to it. The difference is that the data does not need to be stored in the data-frame beforehand (as with ggplot), which makes it a lot more intuitive to work with. Below in the code listing, the plot is stored in the object `p1`. Additional lines and labels are then added to it.

```
# Load the required packages
using Plots, LaTeXStrings

# Save the first plot into an object p1
p1 = plot(tt, MM_GBM[2], label = L"\textrm{Estimated MM } \hat{\Sigma}^{22}_{n,N,M}(t)", color = :blue,
line=(1, [:solid]), ylims = (0.17, 0.45), dpi = 300)
# Add onto the plot using plot!()
plot!(p1, tt, JR_GBM[2], label = L"\textrm{Estimated CT } \hat{\Sigma}^{22}_{n,M}(t)", color = :red,
line=(1, [:dash]))
# Add a horizontal line onto the plot
hline!(p1, [0.2], label = L"\textrm{True } \Sigma^{22}(t)", color = :black, line=(1, [:solid]))
# Add a label for the x-axis
xlabel!(p1, L"\textrm{Time}")
# Add a label for the y-axis
ylabel!(p1, L"\textrm{Instantaneous variance of } \Sigma^{22}(t)")
```

Latex labels

In order to include latex symbols as labels, the LaTeXStrings package needs to be included. Notice in the above coding listing, there is an `L` before the string. This is to indicate that the string is going to be a Latex string. Now including Latex into the label, one simply needs to type out the symbols using the Latex notation. This is well documented. What is not well documented is how do you include text along with latex symbols. The solution to this is to use `\textrm` to wrap the text. This allows the inclusion of both text and symbols without producing any errors.

Annotations to plot

Including annotations into the plot such as in Figure 2.7 can be done using the code listing below. In the example below, the first two arguments `xcord` and `ycord` of `annotations` specify the co-ordinates on the plot. The third argument is the annotation. Here we include latex symbols mixed with an object (value) for which we want to annotate.

```
plot(p1, annotations=(xcord, ycord,
Plots.text(latexstring("\$\\overline{\\rho_{v}} = $(round(object, digits = 4))\$"), :left)))
```

Custom axis ticks

Including custom labels for the axis ticks such as the x-axis in Figure 2.7 can be done using the code listing below using the argument `xticks`.

```
n1 = collect(-1:-1:-14)
xticklabel = [L"10^{-1}", L"10^{-2}", L"10^{-3}", L"10^{-4}", L"10^{-5}", L"10^{-6}",
L"10^{-7}", L"10^{-8}", L"10^{-9}", L"10^{-10}", L"10^{-11}", L"10^{-12}", L"10^{-13}",
L"10^{-14}"]

p1 = plot(n1, object, xticks = (n1, xticklabel))
```

Surface plots

Strangely enough, the documentation does not cover surface plots. Here the z variable must be a matrix matching the dimensions of x and y. Two key arguments to ensure the surface plot looks nice is `fc` and `camera`. The first controls the color gradient and the second controls the camera angle.

```
p1 = surface(M, tt, MM_corr_syn', xlabel = L"\textrm{M}", ylabel = L"\textrm{Time}",
zlabel = L"\textrm{Estimated MM } \hat{\rho}^{12}_{\Delta t}(t)",
fc=ColorGradient([:red,:yellow,:blue]), camera = (65, 55), dpi = 300, clim=(-1,1), zlims = (-1, 1))
```

By default, the color range in the plot and legend will line up based on the maximum and minimum values from the z variable. However, if one wants to set a specific range of colors for certain values such as the surface plots in Chapter 3, then one needs to include both arguments `clim` and `zlims`. `clim` controls the color range in the legend while `zlims` controls the color range in the surface plot.

Contour plots

Contour plots are covered in the documentation, here a demonstration is included to show how one can control the color range in the contour plot. Once again, `fc` controls the color gradient for the plot and here one only needs the `clim` argument to control both the color range in the legend and the contour plot.

```
p1 = contour(tt, dt, MM_corr_asyn[:,:,1], fill = true, fc = ColorGradient([:red,:yellow,:blue]),
ylabel = L"\Delta t \textrm{[sec]}", xlabel = L"\textrm{Time}", dpi = 300, clim=(-1,1))
```

Heat maps

Heat maps are not covered in the documentation, here is a demonstration on how to plot heat maps in Figure 2.13. The first argument is the matrix of values, the color gradient is controlled through `color`, `xticks` and `xticks` control the labels on the x and y axis respectively, and `tickfontsize` controls the font size of the axis labels.

```
p1 = plot(Dir[:,:,1], st=:heatmap, clim=(-1,1), color=cgrad([:red, :white, :blue]),
colorbar_title=L"\rho", xticks = (1:10, equitynames), yticks = (1:10, equitynames),
dpi = 300, size = (800, 700), tickfontsize = 15)
```

Saving plots

Plots can be saved using the function `savefig`. The first argument is the plot object to save. The second argument is the path to save the plot (from where the current directory is).

Saving plots does not sound like a big deal, the problem is the size of the file when making publication quality figures. The usual standard is 300 dpi. Saving a simple figure at 300 dpi using the pdf extension can result in a file size of 6MB. These large file sizes make it difficult to compile documents and lead to an unnecessary large document size.

```
savefig(p1, "Plots/Instantaneous/SynMert11.svg")
```

The workaround for this is to save vector type images using the svg extension. This results in a file size of around 0.5MB compared to the 6MB using the pdf extension. Moreover, I use this online file converter: https://document.online-convert.com/convert/svg-to-pdf to convert the svg file into a pdf file. Converting the svg to pdf further compresses the file to a size of around 200KB which is much more manageable, although not as efficient as plots from R.

Remark D.2.4 *From what I can tell, there is no noticeable degrade in image quality when converting from svg to pdf using the online file converter.*

For figures such as heat-maps or contour plots the problem becomes more tricky. For these types of figures, saving it as a svg file is not efficient. The better extension to save the plot is png in this case. The only issue is that converting the png to pdf causes a very visible degrade in image quality. I have yet to find a solution to this problem.

Finally, for figures such as surface plots the problem is also tricky. Neither the pdf nor png extension works well for these types of figures. Moreover, converting from svg to pdf does not reduce the file size. The file sizes are around 0.5MB which is still manageable but not ideal. I have yet to find a solution to this problem.

D.3 Using the functions from the dissertation

NUFFT

The functions include the 1-Dimensional Type 1 non-uniform fast Fourier transforms using three types of kernels. These functions can be found under the folder /Functions/NUFFT on the GitHub resource (Chang et al., 2020c).

The functions require four input variables:

- cj: vector of source strength,
- xj: vector of source points,
- M: the number of Fourier coefficients you want returned (integer), and
- tol: the precision requested

```
include("Functions/NUFFT/NUFFT-FGG.jl")
include("Functions/NUFFT/NUFFT-KB.jl")
include("Functions/NUFFT/NUFFT-ES.jl")

# Simulate some non-uniform data
nj = 10
x = (collect(0:nj-1) + 0.5 .* rand(nj))
xj = (x .- minimum(x)) .* (2*pi / (maximum(x) - minimum(x)))    # Re-scale s.t. xj \in [0, 2\pi]
cj = rand(nj) + 0im*rand(nj)

# Parameter settings
M = 11 # Output size
tol = 10^-12 # Tolerance

# Output
fk_G = NUFFTFGG(cj, xj, M, tol)   # Gaussian kernel
fk_KB = NUFFTKB(cj, xj, M, tol)   # Kaiser-Bessel kernel
fk_ES = NUFFTES(cj, xj, M, tol)   # Exponential of semi-circle kernel
```

Integrated estimators

The functions include the Malliavin-Mancino and Hayashi-Yoshida estimator where both can deal with asynchrony. These functions can be found under the folder /Functions/Correlation Estimators on the GitHub resource (Chang et al., 2020c).

Malliavin-Mancino integrated estimators using NUFFTs

The implementation is performed using two representations: the Dirichlet and the Fejér. Both representations require two input variables:

- p: (nxD) matrix of prices, with non-trade times represented as NaNs,
- t: (nxD) corresponding matrix of trade times, with non-trade times represented as NaNs,

and two optional input variables:

- N: (optional input) for the number of Fourier coefficients used in the convolution of the Malliavin-Mancino estimator (integer) - defaults to the Nyquist frequency,
- tol: tolerance requested - defaults to 10^{-12}.

```
include("Functions/Correlation Estimators/Dirichlet/NUFFTcorrDK-FGG.jl")
```

```
include("Functions/Correlation Estimators/Fejer/NUFFTcorrFK-FGG.jl")
include("Functions/SDEs/GBM.jl")

# Create some data
mu = [0.01/86400, 0.01/86400]
sigma = [0.1/86400 sqrt(0.1/86400)*0.35*sqrt(0.2/86400);
        sqrt(0.1/86400)*0.35*sqrt(0.2/86400) 0.2/86400]

P = GBM(10000, mu, sigma, seed = 10)
t = reshape([collect(1:1:10000.0); collect(1:1:10000.0)], 10000, 2)

# Parameter settings
N = 500
tol = 10^-12

# Obtain results
output1 = NUFFTcorrDKFGG(P, t, N = N, tol = tol)    # Dirichlet
output2 = NUFFTcorrFKFGG(P, t, N = N, tol = tol)    # Fejer

# Extract results
cor1 = output1[1]   # correlation matrix
cov1 = output1[2]   # integrated covariance

cor2 = output2[1]   # correlation matrix
cov2 = output2[2]   # integrated covariance
```

Hayashi-Yoshida estimator

The Hayashi-Yoshida implementation is performed using the Kanatani weight matrix. The function only computes the integrated covariance for two assets at a time. The function requires four input variables:

- p1: vector of observed prices for the first asset,
- p2: vector of observed prices for the second asset,
- t1: vector of observed trading times for the first asset, and
- t2: vector of observed trading times for the second asset.

```
include("Functions/Correlation Estimators/HY/HYcorr.jl")
include("Functions/SDEs/GBM.jl")

# Create some data
mu = [0.01/86400, 0.01/86400]
sigma = [0.1/86400 sqrt(0.1/86400)*0.35*sqrt(0.2/86400);
        sqrt(0.1/86400)*0.35*sqrt(0.2/86400) 0.2/86400]

P = GBM(10000, mu, sigma, seed = 10)
t = collect(1:1:10000.0)

# Obtain results
output = HYcorr(p1 = P[:,1], p2 = P[:,2], t1 = t, t2 = t)

# Extract results
cor = output[1]   # correlation matrix
cov = output[2]   # integrated covariance
```

Instantaneous estimators

The functions include the Malliavin-Mancino and Cuchiero-Teichmann Fourier instantaneous estimators. These functions can be found under the folder /Functions/Instantaneous Estimators on the GitHub resource (Chang et al., 2020c).

Malliavin-Mancino instantaneous estimators using NUFFTs

The function requires three input variables:

- p: (nx2) matrix of prices, with non-trade times represented as NaNs,
- t: (nx2) corresponding matrix of trade times, with non-trade times represented as NaNs,
- outlength: the number of synchronous grid points to reconstruct the spot estimates.

and three optional input variables:

- N: (optional input) for the number of Fourier coefficients of the price process used in the convolution of the Malliavin-Mancino estimator (integer), controls the level of averaging and directly affects the time-scale investigated - defaults to the Nyquist frequency,
- M: (optional input) for the number of Fourier coefficients of the volatility process using in the reconstruction of the spot estimates - defaults to $\frac{1}{8}\frac{1}{2\pi}\sqrt{n}\log n$, and
- tol: tolerance requested - defaults to 10^{-12}.

```
include("Functions/SDEs/Heston.jl")
include("Functions/Instantaneous Estimators/MM-Inst.jl")

# Simulate some price observations from the Heston model.

nsim = 28800
P_Heston = Heston_CT(nsim, seed = 1, dt = nsim)
    # First variable in P_Heston is the price matrix,
    # Second to fourth variable are the true volatility and co-volatility.
t = collect(1:1:nsim)

# Parameter settings
outlength = 1000    # length of output vector
M = 100 # Cutting freq.
tol = 10^-12

# Output
MM_Heston = MM_inst(P_Heston[1], [t t], outlength, M = M, tol = tol)
    # First variable is the volatility estimates of asset 1.
    # Second variable is the volatility estimates of asset 2.
    # Third variable is the co-volatility estimates of asset 1 and 2.

# Extract results
vol11 = MM_Heston[1]    # Vector of spot volatility estimates for the first asset
vol22 = MM_Heston[2]    # Vector of spot volatility estimates for the second asset
vol12 = MM_Heston[3]    # Vector of spot co-volatility estimates
```

Cuchiero-Teichmann

The Cuchiero-Teichmann instantaneous estimator uses the specification of $g(x) = \cos(x)$. The estimator requires the data to be strictly synchronous, therefore the asynchronous data needs to be synchronised beforehand using the previous tick interpolation.

The function requires three input variables:

- p: (n x 2) double float matrix of price observations,
- N: the cutting frequency (integer) used in the reconstruction of the spot estimates, the dissertation uses the notation M, and
- outlength: the number of synchronous grid points to reconstruct the spot estimates.

```
include("Functions/SDEs/Heston.jl")
include("Functions/Instantaneous Estimators/MM-JR.jl")

# Simulate some price observations from the Heston model.

nsim = 28800
P_Heston = Heston_CT(nsim, seed = 1, dt = nsim)
    # First variable in P_Heston is the price matrix,
    # Second to fourth variable are the true volatility and co-volatility.
t = collect(1:1:nsim)

# Parameter settings
outlength = 1000    # length of output vector
M = 100 # Cutting freq.

# Output
JR_Heston = MM_JR(P_Heston[1], M, outlength)
    # First variable is the volatility estimates of asset 1.
    # Second variable is the volatility estimates of asset 2.
    # Third variable is the co-volatility estimates of asset 1 and 2.

# Extract results
vol11 = JR_Heston[1]    # Vector of spot volatility estimates for the first asset
vol22 = JR_Heston[2]    # Vector of spot volatility estimates for the second asset
vol12 = JR_Heston[3]    # Vector of spot co-volatility estimates
```

Hawkes

The functions include a variety of functions for the simulation and calibration of a M-variate Hawkes process. The Hawkes process uses an exponential kernel of the form:

$$\phi^{mn}(t) = \alpha^{mn} e^{-\beta^{mn} t} \mathbb{1}_{t \in \mathbb{R}^+},$$

Simulation

The function to simulate the Hawkes process requires four input variables:

- lambda0: the vector of constant base-line intensity,
- alpha: MxM matrix of alphas in the exponential kernel,
- beta: MxM matrix of betas in the exponential kernel, and
- T: the time horizon of the simulation.

```
include("Functions/Hawkes/Hawkes.jl")

# Setting the parameters for a 2-variate Hawkes process
lambda0 = [0.016 0.016]
alpha = [0 0.023; 0.023 0]
beta = [0 0.11; 0.11 0]
T = 3600

# Simulate the process
t = simulateHawkes(lambda0, alpha, beta, T)

# Extract the simulation results
t1 = t[1] # vector of arrival times for the first count process
t2 = t[2] # vector of arrival times for the second count process
```

Calibration

The calibration of the Hawkes process is a bit more intricate. When calibrating, we only have the observations from the M-variate count process. We do not know the true specification of the Hawkes process (See Appendix B.4). Nonetheless, suppose we do know the correct specification. We need to first create a function that takes in a vector of parameters and creates the matrix of α^{mn} and β^{mn} according to the specification. From there, we can send the parameters and observations into the log-likelihood for maximisation. Note that the observations is in the format of a vector of length M for each process and each element is a vector of observations from the specific process. In other words, the output format from the simulation example.

```
using Optim
include("Functions/Hawkes/Hawkes")

# Function to be used in optimization for the above simulation
# i.e. creating a function that takes in a vector of observations
# to create the Hawkes specification
function calibrateHawkes(param)
    lambda0 = [param[1] param[1]]
    alpha = [0 param[2]; param[2] 0]
    beta = [0 param[3]; param[3] 0]
    return -loglikeHawkes(t, lambda0, alpha, beta, T)   # t is the vector of vector of observations;
    # returns the negative because Optim minimises
end

# Optimize the parameters using Optim
res = optimize(calibrateHawkes, [0.01, 0.015, 0.15]) # function to minimise
    # and vector of initial parameters
par = Optim.minimizer(res)  # MLE estimates of the parameter
```

D.4 Reproducing the research

The data used in this dissertation can be found in Chang et al. (2020a,b). All the script files for obtaining the results can be found under /Scripts on the GitHub resource Chang et al. (2020c).

The first step is to clone/download the GitHub resource Chang et al. (2020c). Next, the data sets must be downloaded from Chang et al. (2020a,b). These CSV files must then be placed in the folder `/Real Data` from the GitHub resource. The results from the dissertation all have the script file that produces the results in the captions. Before running those script files, the directory should be changed from `cd("/Users/patrickchang1/PCEPTG-MSC")` to where you have stored the file `PCEPTG-MSC`. Once the directories have been changed, it is a simple matter of running the script file to reproduce the results. Note that for the results that have a long compute time, I have stored the results in `/Computed Data` so that the figures and tables can be reproduced without the long run time.

Bibliography

Ait-Sahalia, Y., Fan, J., Xiu, D., 2010. High-frequency covariance estimates with noisy and asynchronous financial data. Journal of the American Statistical Association 105, 1504–1517. doi:10.2139/ssrn.1631344.

Alvarez, A., Panloup, F., Pontier, M., Savy, N., 2012. Estimation of the instantaneous volatility. Statistical Inference for Stochastic Processes 15, 27–59. doi:10.1007/s11203-011-9062-2.

Aït-Sahalia, Y., Mykland, P.A., Zhang, L., 2005. How Often to Sample a Continuous-Time Process in the Presence of Market Microstructure Noise. The Review of Financial Studies 18, 351–416. doi:10.1093/rfs/hhi016.

Aït-Sahalia, Y., Mykland, P.A., Zhang, L., 2011. Ultra high frequency volatility estimation with dependent microstructure noise. Journal of Econometrics 160, 160 – 175. doi:https://doi.org/10.1016/j.jeconom.2010.03.028. realized Volatility.

Bachelier, L., 1900. Théorie de la spéculation. Annales scientifiques de l'École Normale Supérieure 3e série, 17, 21–86. doi:10.24033/asens.476.

Bacry, E., Delattre, S., Hoffmann, M., Muzy, J.F., 2013a. Modelling microstructure noise with mutually exciting point processes. Quantitative Finance 13, 65–77. doi:10.1080/14697688.2011.647054.

Bacry, E., Delattre, S., Hoffmann, M., Muzy, J.F., 2013b. Some limit theorems for hawkes processes and application to financial statistics. Stochastic Processes and their Applications 123.

Bacry, E., Mastromatteo, I., Muzy, J.F., 2015. Hawkes processes in finance. Market Microstructure and Liquidity 01, 1550005. doi:10.1142/S2382626615500057.

Bandi, F.M., Renò, R., 2018. Nonparametric stochastic volatility. Econometric Theory 34, 1207–1255. doi:10.1017/S0266466617000457.

Barndorff-Nielsen, O.E., Graversen, S.E., Jacod, J., Podolskij, M., Shephard, N., 2006a. A Central Limit Theorem for Realised Power and Bipower Variations of Continuous Semimartingales. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 33–68. URL: https://doi.org/10.1007/978-3-540-30788-4_3, doi:10.1007/978-3-540-30788-4_3.

Barndorff-Nielsen, O.E., Shephard, N., 2004. Power and Bipower Variation with Stochastic Volatility and Jumps. Journal of Financial Econometrics 2, 1–37. doi:10.1093/jjfinec/nbh001.

Barndorff-Nielsen, O.E., Shephard, N., Winkel, M., 2006b. Limit theorems for multipower variation in the presence of jumps. Stochastic Processes and their Applications 116, 796 – 806. doi:https://doi.org/10.1016/j.spa.2006.01.007.

Barnett, A.H., Magland, J.F., af Klinteberg, L., 2018. A parallel non-uniform fast Fourier transform library based on an "exponential of semicircle" kernel. SIAM J. Scientific Computing 41, C479–C504. doi:10.1137/18m120885x.

Barucci, E., Renò, R., 2002. On measuring volatility and the garch forecasting performance. Journal of International Financial Markets, Institutions and Money 12, 183–200. doi:10.1016/S1042-4431(02)00002-1.

Bouchaud, J.P., Farmer, J., Lillo, F., 2008. How markets slowly digest changes in supply and demand. Handbook of Financial Markets: Dynamics and Evolution arXiv. doi:10.2139/ssrn.1266681.

Bowsher, C., 2007. Modelling security market events in continuous time: Intensity based, multivariate point process models. Journal of Econometrics 141, 876–912.

Box, G.E., Luceno, A., del Carmen Paniagua-Quiñones, M., 2009. Statistical Control by Monitoring and Adjustment. 2nd ed., John Wiley & Sons, Inc., USA.

Buccheri, G., Livieri, G., Pirino, D., Pollastri, A., 2019. A closed-form formula characterization of the Epps effect. Quantitative Finance 20, 243–254. doi:10.1080/14697688.2019.1659992.

Bukuru, R., Chang, P., Gebbie, T., 2019. R code: Exploring the Epps effect in south african markets. URL: https://github.com/rogerbukuru/Exploring-The-Epps-Effect-R, doi:10.25375/uct.11310350.

Chang, P., 2020. Fourier instantaneous estimators and the epps effect. PLOS ONE 15, 1–24. doi:10.1371/journal.pone.0239415.

Chang, P., Bukuru, R., Gebbie, T., 2019a. An Exercise in R: High Frequency Covariance estimation using Malliavin-Mancino and Hayashi-Yoshida estimators. Project Report. University of Cape Town. doi:10.25375/uct.11310473.

Chang, P., Bukuru, R., Gebbie, T., 2019b. Revisting the Epps effect using volume time averaging: An exercise in r. URL: https://arxiv.org/abs/1912.02416.

Chang, P., Pienaar, E., Gebbie, T., 2020a. Johannesburg stock exchange trade and quote data. doi:10.25375/uct.11903442.

Chang, P., Pienaar, E., Gebbie, T., 2020b. Johannesburg stock exchange trade and quote data. doi:10.25375/uct.12315092.

Chang, P., Pienaar, E., Gebbie, T., 2020c. Julia code: Dissertation. URL: https://github.com/CHNPAT005/PCEPTG-MSC, doi:10.25375/uct.12832568.

Chang, P., Pienaar, E., Gebbie, T., 2020d. Malliavin-Mancino estimators implemented with non-uniform fast Fourier transforms, SIAM Journal on Scientific Computing. Forthcoming. URL: https://arxiv.org/abs/2003.02842.

Chang, P., Pienaar, E., Gebbie, T., 2020e. Using the Epps effect to detect discrete data generating processes. URL: https://arxiv.org/abs/2005.10568.

Chen, J., Revels, J., 2016. Robust benchmarking in noisy environments. arXiv e-prints .

Chen, R.Y., 2019. The Fourier transform method for volatility functional inference by asynchronous observations. arXiv:1911.02205.

Cooley, J., Tukey, J., 1965. An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation 19, 297–301. doi:10.1090/S0025-5718-1965-0178586-1.

Cuchiero, C., Teichmann, J., 2015. Fourier transform methods for pathwise covariance estimation in the presence of jumps. Stochastic Processes and their Applications 125, 116 – 160. doi:https://doi.org/10.1016/j.spa.2014.07.023.

Curato, I., Sanfelici, S., 2015. Measuring the Leverage Effect in a High-Frequency Trading Framework. chapter 24. p. 425–446. doi:10.1016/B978-0-12-802205-4.00024-5.

Dacorogna, M., Gencay, R., Muller, U., Olsen, R., Pictet, O., 2001. An Introduction to High-Frequency Finance. Academic Press.

Dassios, A., Zhao, H., 2013. Exact simulation of hawkes process with exponentially decaying intensity. Electron. Commun. Probab. 18, 13 pp. doi:10.1214/ECP.v18-2717.

Dutt, A., Rokhlin, V., 1993. Fast Fourier transforms for nonequispaced data. SIAM Journal on Scientific Computing 14, 1368–1393. doi:10.1137/0914081.

Easley, D., Engle, R.F., O'Hara, M., Wu, L., 2008. Time-Varying Arrival Rates of Informed and Uninformed Trades. Journal of Financial Econometrics 6, 171–207. doi:10.1093/jjfinec/nbn003.

Epps, T.W., 1979. Comovements in stock prices in the very short run. Journal of the American Statistical Association 74, 291–298.

Frigo, M., Johnson, S.G., 2005. The design and implementation of FFTW3. Proceedings of the IEEE 93, 216–231. doi:10.1109/JPROC.2004.840301. special issue on "Program Generation, Optimization, and Platform Adaptation".

Glasserman, P., 2004. Monte Carlo methods in financial engineering. Springer, New York.

Greengard, L., Lee, J.Y., 2004. Accelerating the nonuniform fast Fourier transform. SIAM Review 46, 443–454. doi:10.1137/S003614450343200X.

Griffin, J.E., Oomen, R.C., 2011. Covariance measurement in the presence of non-synchronous trading and market microstructure noise. Journal of Econometrics 160, 58–68.

Hawkes, A.G., 1971. Spectra of some self-exciting and mutually exciting point processes. Biometrika 58, 83–90.

Hawkes, A.G., 2018. Hawkes processes and their applications to finance: a review. Quantitative Finance 18, 193–198. doi:10.1080/14697688.2017.1403131.

Hayashi, T., Yoshida, N., 2005. On covariance estimation of non-synchronously observed diffusion processes. Bernoulli 11, 359–379. doi:10.3150/bj/1116340299.

Hendricks, D., 2016. An online adaptive learning algorithm for optimal trade execution in high-frequency markets. Ph.D. thesis. University of the Witwatersrand. URL: http://hdl.handle.net/10539/21710.

Hendricks, D., 2017a. Using real-time cluster configurations of streaming asynchronous features as online state descriptors in financial markets. Pattern Recognition Letters 97, 21 – 28. doi:https://doi.org/10.1016/j.patrec.2017.06.026.

Hendricks, D., 2017b. Using real-time cluster configurations of streaming asynchronous features as online state descriptors in financial markets. Pattern Recognition Letters 97, 21 – 28.

Hendricks, D., Gebbie, T., Wilcox, D., 2016a. Detecting intraday financial market states using temporal clustering. Quantitative Finance 16, 1657–1678. doi:10.1080/14697688.2016.1171378.

Hendricks, D., Gebbie, T., Wilcox, D., 2016b. Detecting intraday financial market states using temporal clustering. Quantitative Finance 16, 1657–1678.

Hendricks, D., Gebbie, T., Wilcox, D., 2017. High-speed Fourier method estimation of covariances from asynchronous data. Working paper.

Hendricks, D., Wilcox, D., 2014. A reinforcement learning extension to the almgren-chriss framework for optimal trade execution. 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr) , 457–464.

Jackson, J.I., Meyer, C.H., Nishimura, D.G., Macovski, A., 1991. Selection of a convolution function for Fourier inversion using gridding (computerised tomography application). IEEE Transactions on Medical Imaging 10, 473–478.

Jacod, J., 2008. Asymptotic properties of realized power variations and related functionals of semimartingales. Stochastic Processes and their Applications 118, 517 – 559. doi:https://doi.org/10.1016/j.spa.2007.05.005.

Jaisson, T., Rosenbaum, M., 2015. Limit theorems for nearly unstable hawkes processes. The Annals of Applied Probability 25, 600–631.

JSE, 2019. Volume 00E-Trading and Information Overview. Johannesburg Stock Exchange. Sandown, Johannesburg, South Africa.

Kanatani, T., 2004. Optimally weighted realized volatility.

Karatzas, I., Shreve, S.E., 1998. Brownian Motion and Stochastic Calculus. Springer-Verlag. doi:10.1007/978-1-4612-0949-2.

af Klinteberg, L., 2018. Julia interface to finufft. URL: https://github.com/ludvigak/FINUFFT.jl.

Kloeden, P.E., Platen, E., 2013. Numerical solution of stochastic differential equations. volume 23. Springer Science & Business Media.

Large, J., 2007. Measuring the resiliency of an electronic limit order book. Journal of Financial Markets 10.

Lindskog, F., 2001. Linear correlation estimation. RiskLab Report, ETH Zurich.

Malherbe, C., Hendricks, D., Gebbie, T., Wilcox, D., 2005. Matlab functions `ftcorrgpu.m` and `fftcorrgpu.m`.

Malliavin, P., Mancino, M., 2002. Fourier series method for measurement of multivariate volatilities. Finance and Stochastics 6, 49–61. doi:10.1007/s780-002-8400-6.

Malliavin, P., Mancino, M., 2009. A Fourier transform method for nonparametric estimation of multivariate volatility. Ann. Statist. 37, 1983–2010. doi:10.1214/08-AOS633.

Malliavin, P., Thalmaier, A., 2006. Stochastic Calculus of Variations in Mathematical Finance. Springer Finance, Springer Berlin Heidelberg. doi:10.1007/3-540-30799-0.

Mancino, M., Recchioni, M., 2015. Fourier spot volatility estimator: Asymptotic normality and efficiency with liquid and illiquid high-frequency data. PloS one 10, e0139041. doi:10.1371/journal.pone.0139041.

Mancino, M., Recchioni, M., Sanfelici, S., 2017. Fourier-Malliavin Volatility Estimation Theory and Practice. Springer International Publishing. doi:10.1007/978-3-319-50969-3.

Mancino, M.E., Sanfelici, S., 2011. Estimating Covariance via Fourier Method in the Presence of Asynchronous Trading and Microstructure Noise. Journal of Financial Econometrics 9, 367–408. doi:10.1093/jjfinec/nbq031.

Markowitz, H., 1952. Portfolio selection*. The Journal of Finance 7, 77–91. doi:10.1111/j.1540-6261.1952.tb01525.x.

Martins, R., Hendricks, D., 2016. The statistical significance of multivariate hawkes processes fitted to limit order book data. arXiv:1604.01824.

Mastromatteo, I., Marsili, M., Zoi, P., 2011. Financial correlations at ultra-high frequency: theoretical models and empirical estimation. The European Physical Journal B 80, 243–253. doi:10.1140/epjb/e2011-10865-y.

Matoti, L., 2009. Building a statistical linear factor model and a global minimum variance portfolio using estimated covariance matrices. MSc. Dissertation. University of Cape Town.

Mattiussi, V., Iori, G., 2010. A nonparametric approach to estimate volatility and correlation dynamics. Working paper.

McNeil, A., Frey, R., Embrechts, P., 2005. Quantitative Risk Management: Concepts, Techniques, and Tools. Princeton University Press.

Münnix, M.C., Schäfer, R., Guhr, T., 2010. Impact of the tick-size on financial returns and correlations. Physica A: Statistical Mechanics and its Applications 389, 4828 – 4843. doi:https://doi.org/10.1016/j.physa.2010.06.037.

Münnix, M.C., Schäfer, R., Guhr, T., 2011. Statistical causes for the Epps effect in microstructure noise. International Journal of Theoretical and Applied Finance 14, 1231–1246. doi:10.1142/S0219024911006838.

Mykland, P.A., Zhang, L., 2008. Inference for volatility-type objects and implications for hedging. Statistics and Its Interface 1, 255 – 278.

Møller, J., Rasmussen, J.G., 2005. Perfect simulation of hawkes processes. Advances in Applied Probability 37, 629–646.

Ogata, Y., 1981. On lewis' simulation method for point processes. IEEE Trans. Inf. Theory 27, 23–30.

Ogata, Y., 1988. Statistical models for earthquake occurrences and residual analysis for point processes. Journal of the American Statistical Association 83, 9–27. doi:10.1080/01621459.1988.10478560.

Ogata, Y., 1999. Seismicity analysis through point-process modeling: A review. pure and applied geophysics 155, 471–507. doi:10.1007/s000240050275.

Ogata, Y., Akaike, H., 1982. On linear intensity models for mixed doubly stochastic poisson and self-exciting processes. Journal of the Royal Statistical Society. Series B 44. doi:10.1007/978-1-4612-1694-0_20.

Ousterhout, J.K., 1998. Scripting: Higher-level programming for the 21st century. Computer 31, 23–30. doi:10.1109/2.660187.

Park, S., Hong, S.Y., Linton, O., 2016. Estimating the quadratic covariation matrix for asynchronously observed high frequency stock returns corrupted by additive measurement error. Journal of Econometrics 191, 325 – 347. doi:https://doi.org/10.1016/j.jeconom.2015.12.005. innovations in Measurement in Economics and Econometrics.

Platt, D., Gebbie, T., 2018. Can agent-based models probe market microstructure? Physica A: Statistical Mechanics and its Applications 503, 1092 – 1106. doi:https://doi.org/10.1016/j.physa.2018.08.055.

Podolskij, M., Vetter, M., 2009. Estimation of volatility functional in the simultaneous presence of microstructure noise and jumps. Bernoulli 15, 634–658. doi:10.3150/08-BEJ167. cited By 138.

Potts, D., Steidl, G., 2003. Fast summation at nonequispaced knots by nfft. SIAM Journal on Scientific Computing 24, 2013–2037. doi:10.1137/S1064827502400984.

López de Prado, M., 2016. Building diversified portfolios that outperform out of sample. The Journal of Portfolio Management 42, 59–69. doi:10.3905/jpm.2016.42.4.059.

Precup, O.V., Iori, G., 2007. Cross-correlation measures in the high-frequency domain. The European Journal of Finance 13, 319–331. doi:10.1080/13518470600813565.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 1992. Numerical Recipes in C. Second ed., Cambridge University Press, Cambridge, USA.

Renò, R., 2003. A closer look at the Epps effect. International Journal of Theoretical and Applied Finance 06, 87–102. doi:10.2139/ssrn.314723.

Rizoiu, M.A., Lee, Y., Mishra, S., Xie, L., 2017. A Tutorial on Hawkes Processes for Events in Social Media. arXiv:1708.06401 [cs, stat] ArXiv: 1708.06401.

Saichev, A., Sornette, D., 2014. A simple microstructure return model explaining microstructure noise and Epps effects. International Journal of Modern Physics C 25, 1450012. doi:10.1142/S0129183114500120.

Sanfelici, S., Curato, I.V., Mancino, M.E., 2015. High-frequency volatility of volatility estimation free from spot volatility estimates. Quantitative Finance 15, 1331–1345. doi:10.1080/14697688.2015.1032542.

Todorov, V., Tauchen, G., 2012. The realized laplace transform of volatility. Econometrica 80, 1105–1127. doi:10.3982/ECTA9133. cited By 34.

Toke, I., Pomponio, F., 2012. Modelling trades-through in a limit order book using hawkes processes. Economics: The Open-Access, Open-Assessment E-Journal 6, 1–23. doi:10.2139/ssrn.1973856.

Tóth, B., Kertész, J., 2007. Modeling the Epps effect of cross correlations in asset prices, in: Kertész, J., Bornholdt, S., Mantegna, R.N. (Eds.), Noise and Stochastics in Complex Systems and Finance, International Society for Optics and Photonics. SPIE. pp. 89 – 97. URL: https://doi.org/10.1117/12.727127, doi:10.1117/12.727127.

Tóth, B., Kertész, J., 2009. The Epps effect revisited. Quantitative Finance 9, 793–802. doi:10.1080/14697680802595668.

Yelibi, L., Gebbie, T., 2019. Agglomerative fast super-paramagnetic clustering. URL: https://arxiv.org/abs/1908.00951.

Zhang, L., 2010. Estimating covariation: Epps effect, microstructure noise. Journal of Econometrics 160, 33–47. doi:10.1016/j.jeconom.2010.03.012.

Zhang, L., Mykland, P.A., Aït-Sahalia, Y., 2005. A tale of two time scales: Determining integrated volatility with noisy high-frequency data. Journal of the American Statistical Association 100, 1394–1411.