

Deposit & Copying of Thesis Declaration



UNIVERSITY OF
CAMBRIDGE

Please note that you will also need to bind a copy of this Declaration into your final, hardbound copy of thesis - this has to be the very first page of the hardbound thesis.

1	Surname (Family Name)	Forenames(s)	Title
	LI	CHENHAO	MR
2	Title of Thesis as approved by the Degree Committee		
	Sequential Modelling and Inference of High-frequency Limit Order Book with State-space Models and Monte Carlo Algorithms		

In accordance with the University Regulations in *Statutes and Ordinances* for the PhD, MSc and MLitt Degrees, I agree to deposit one print copy of my thesis entitled above with the Secretary of the Postgraduate Committee who shall deposit the thesis in the University Library under the following terms and conditions:

1. Thesis Author Declaration

I am the author of this thesis and hereby give the University the right to make my thesis available in print form as described in 2. below.

My thesis is my original work and a product of my own research endeavours and includes nothing which is the outcome of work done in collaboration with others except as declared in the Preface and specified in the text. I hereby assert my moral right to be identified as the author of the thesis.

The deposit and dissemination of my thesis by the University does not constitute a breach of any other agreement, publishing or otherwise, including any confidentiality or publication restriction provisions in sponsorship or collaboration agreements governing my research or work at the University or elsewhere.

2. Access to Dissertation

I understand that one print copy of my thesis will be deposited in the University Library for archival and preservation purposes, and that, unless upon my application restricted access to my thesis for a specified period of time has been granted by the Postgraduate Committee prior to this deposit, the thesis will be made available by the University Library for consultation by readers in accordance with University Library Regulations and copies of my thesis may be provided to readers in accordance with applicable legislation.

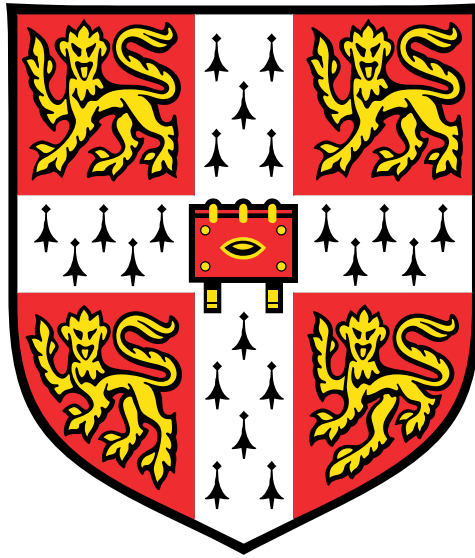
3	Signature	Date
		8 May 2021

Corresponding Regulation

Before being admitted to a degree, a student shall deposit with the Secretary of the Postgraduate Committee one copy of his or her hard-bound thesis, in a form approved by the Committee. The Secretary shall deposit the copy of the thesis in the University Library where, subject to restricted access to the thesis for a specified period of time having been granted by the Postgraduate Committee, it shall be made available for consultation by readers in accordance with University Library Regulations and copies of the thesis provided to readers in accordance with applicable legislation.

Sequential Modelling and Inference of High-frequency Limit Order Book with State-space Models and Monte Carlo Algorithms

Signal Processing and Communications Laboratory, CUED



Chenhao Li
St. John's College

September 2020

This thesis is submitted for the degree of Doctor of Philosophy.

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Chenhao (James) Li

Date: September 2020

Abstract

Sequential Modelling and Inference of High-frequency Limit Order Book with State-space Models and Monte Carlo Algorithms

Chenhao Li

The high-frequency limit order book (LOB) market has recently attracted increasing research attention from both the industry and the academia as a result of expanding algorithmic trading. However, the massive data throughput and the inherent complexity of high-frequency market dynamics also present challenges to some classic statistical modelling approaches. By adopting powerful state-space models from the field of signal processing as well as a number of Bayesian inference algorithms such as particle filtering, Markov chain Monte Carlo and variational inference algorithms, this thesis presents my extensive research into the high-frequency limit order book covering a wide scope of topics.

Chapter 2 presents a novel construction of the non-homogeneous Poisson process to allow online intensity inference of limit order transactions arriving at a central exchange as point data. Chapter 3 extends a baseline jump diffusion model for market fair-price process to include three additional model features taken from real-world market intuitions. In Chapter 4, another price model is developed to account for both long-term and short-term diffusion behaviours of the price process. This is achieved by incorporating multiple jump-diffusion processes each exhibiting a unique characteristic. Chapter 5 observes the multi-regime nature of price diffusion processes as well as the non-Markovian switching behaviour between regimes. As such, a novel model is proposed which combines the continuous-time state-space model, the hidden semi-Markov switching model and the non-parametric Dirichlet process model. Additionally, building upon the general structure of the particle Markov chain Monte Carlo algorithm, I further propose an algorithm which achieves sequential state inference, regime identification and regime parameters learning requiring minimal prior assumptions. Chapter 6 focuses on the development of efficient parameter-learning algorithms for state-space models and presents three algorithms each demonstrating promising results in comparison to some well-established methods.

The models and algorithms proposed in this thesis not only are practical tools for analysing high-frequency LOB markets, but can also be applied in various areas and disciplines beyond finance.

Dedication

I dedicate my thesis to my beloved and loving wife, Shiqi Li who has offered me unwavering support and encouragement throughout the past four years of my doctoral journey. She keeps me company when I fail and cheers for even the tiniest success of my research. She always believes in me even in the times when I doubt myself. She is the light of my world.

I also dedicate this thesis to my parents, Yu Li and Meifen Lu who always offer me their unconditional and undying love. They provided for me and supported me to become whoever I wanted to be. Their wisdom inspires me and their attitude to life shapes me. They are my twin pillars, without whom I could not stand.

Contents

1	Introduction	12
1.1	Motivations	12
1.2	Background	13
1.2.1	Basics of limit order book	15
1.2.2	State-space modelling and particle filtering	16
1.3	Summary of contributions	17
1.4	Thesis outline	18
2	Bayesian sequential inference for non-homogeneous Poisson process intensity	20
2.1	Background	20
2.1.1	Non-homogeneous Poisson process	21
2.1.2	Thinning and simulation	22
2.2	Review of existing methods	24
2.2.1	Sigmoidal Gaussian Cox process (SGCP)	26
2.3	Model	27
2.3.1	Continuous-time state space model	27
2.3.2	Doubly-stochastic process with SSM dynamics	28
2.4	Inference	29
2.4.1	A generic SMC MC algorithm	31
2.4.2	Joint proposal of latent variables	34
2.4.3	Metropolis-within-Gibbs refinement	35
2.4.4	Refinement with rejection sampling (RS)	39
2.4.5	Sequential batch scheme	41
2.5	Results and discussions	42
2.5.1	Synthetic Data	43
2.5.2	Application to Order Book Data	45
2.5.3	Convergence Evaluation	47
2.5.4	Hyperparameter Settings	49
2.6	Conclusions and future work	51

Appendices	54
2.A Proof of Theroem 1	54
2.B SDE solution	55
2.C Thinning from rejection sampling	56
3 Extended state-space model for LOB market price inference	58
3.1 Background and LOB imbalance investigation	58
3.1.1 The effect of LOB imbalance on price movements and trends	59
3.2 Proposed state-space models	61
3.2.1 Review of the jump diffusion model	61
3.2.2 Imbalance-driven volumetric model	64
3.2.3 Extended trend model	66
3.2.4 Jump trend resetting model	67
3.2.5 Full model	67
3.3 Inference	68
3.3.1 Generic bootstrap particle filter	68
3.3.2 Rao-Blackwellised particle filter (RBPF)	69
3.4 Results and discussions	72
3.5 Conclusions and future work	78
4 Multi-jump diffusion process for long-short term price dynamics	81
4.1 Introduction and motivation	81
4.2 Model formulation	83
4.2.1 Multi-jump reversion process	83
4.3 Inference	86
4.3.1 Jump proposals and semi-deterministic particle filtering	87
4.4 Results and discussions	92
4.4.1 Simulation and inference	93
4.4.2 FOREX market price	97
4.4.3 Connections to the NHPP intensity inference model	101
4.5 Chapter summary and future work	102
5 State-space regime-switching model with infinite mixture dynamics	103
5.1 Introduction and motivations	104
5.2 Background and review	105
5.2.1 Hidden semi-Markov model	106
5.2.2 Dirichlet process model	108
5.3 Model	111
5.3.1 Continous-time State-space Model	111
5.3.2 Integration of models	112

5.4	Inference	116
5.4.1	RBPF: time-series state inference	117
5.4.2	Blocked Metropolis-within-Gibbs: DPM inference	120
5.4.3	Particle-MCMC: an iterative framework	124
5.4.4	Deterministic filtering and optimal resampling	125
5.5	Results and discussions	128
5.5.1	Synthetic data	128
5.5.2	Animal GPS data	132
5.5.3	FOREX market price	134
5.6	Generalisation of diffusion models	138
5.7	Conclusions	140
Appendices		142
5.A	Stratified sampling algorithm	142
6	Marginal filters and variational parameter learning for state-space models	143
6.1	Marginal Kalman filter	144
6.1.1	Model setup	145
6.1.2	Marginal filtering	146
6.1.3	Posteriors, backward smoothing and marginal state inference	147
6.1.4	Simulation results	150
6.2	Parameter learning in the particle filter	153
6.2.1	Marginal Rao-Blackwellised particle filter	154
6.2.2	Particle filter based variational inference	161
6.2.3	Hybrid PF-VI	171
6.3	Results and discussions	173
6.3.1	Jump-diffusion example	173
6.3.2	Gordon-Kitagawa example	179
6.4	Conclusions and future work	184
Appendices		186
6.A	Expected transition density and likelihood	186
6.B	ELBO evaluation for jump-diffusion model	187
7	Summary	190
Bibliography		193

List of Acronyms

ACF : Autocorrelation Function	MCMC : Markov Chain Monte Carlo
AD : Automatic Differentiation	MH : Metropolis-Hastings
APF : Auxiliary Particle Filter	MKF : Marginal Kalman Filter
AR : Autoregressive (model)	ML : Maximum Likelihood
BBO : Best Bid Offer (data)	MwG : Metropolis-within-Gibbs
BMwG : Blocked Metropolis-within-Gibbs	MSE : Mean Square Error
CAVI : Coordinate Ascent Variational Inference	NHPP : Non-homogeneous Poisson Process
CI : Confidence Interval	\mathcal{NIG} : Normal-inverse-Gamma (distribution)
CDA : Continuous Double Auction	OU : Ornstein-Uhlenbeck (process)
CDF : Cumulative Density Function	PDF : Probability Density Function
DP : Dirichlet Process	PED : Prediction Error Decomposition
DPM : Dirichlet Process Model	PF : Particle Filter
EM : Expectation Maximisation	PF-VI : Particle Filter Variational Inference (algorithm)
ET : Eastern Time	PG : Particle Gibbs
EUR : Euro	PL : Particle Learning
ELBO : Evidence Lower Bound	PMCMC : Particle Markov Chain Monte Carlo
FOREX : Foreign Exchange	PMMH : Particle Marginal Metropolis Hastings
GP : Gaussian Process	RBPF : Rao-Blackwellised Particle Filter
GPS : Global Positioning System	RMSE : Root Mean Square Error
GARCH : generalised autoregressive conditional heteroskedasticity (model)	RS : Rejection Sampling
HDP : Hierarchical Dirichlet Process	RTS : Rauch-Tung-Striebel (smoother)
HMM : Hidden Markov Model	SDE : Stochastic Differential Equation
HsMM : Hidden Semi-Markov Model	SGCP : Sigmoidal Gaussian Cox Process
HPP : Homogeneous Poisson Process	S-LD : Sequential Langevin
IACT : Integrated Autocorrelation Time	SMC : Sequential Monte Carlo
\mathcal{IG} : Inverse-Gamma (distribution)	SMCMC : Sequential Markov Chain Monte Carlo
IRS : Importance Resampling	SNP : Snapshot (data)
IMM : Interacting Multiple Model	SSM : State-space Model
KDE : Kernel Density Estimation	SSSM : Switching State-space Model
KF : Kalman Filter	TKS : Ticks (data)
KL-divergence : Kullback-Leibler divergence	USD : United States Dollar
LGCP : Log Gaussian Cox Process	VB : Variational Bayes
LOB : Limit Order Book	VI : Variational Inference
LSTM : Long Short-term Memory	VRPF : Variable Rate Particle Filter
MAE : Mean Absolute Error	
MALA : Metropolis-adjusted Langevin Algorithm	

Chapter 1

Introduction

1.1 Motivations

Financial market, as one of the most complicated and volatile systems in the world, attracts more than just practitioners but also researchers from various disciplines. While most of the mid and long-duration markets and commodities are still strongly correlated to macroeconomic conditions, high-frequency financial markets, on the other hand, are constantly subject to short-term or even instantaneous volatility and pressure from different sources. Consequently, their trends or behaviours cannot typically be captured by human eye and identified with only macroeconomic intuitions.

As a result of recent advancements in algorithmic trading and quantitative finance, trading operations based on algorithm-generated decisions and algorithmically identified signals have become more and more common especially in high-frequency markets. A study in 2016 [1] showed that over 80% of the trading volume in the FOREX (foreign exchange) market was performed by trading algorithms, while the report from [2] predicts the total market worth of algorithmic trading to grow from \$11.1 billion in 2019 to \$18.8 billion by 2024. However in recent years, major failures of algorithmic trading had also resulted in substantial losses, fines and reputational damage for credit institutions and investment firms e.g. the Knight Capital trading glitch in 2012 [3]. Such factors have motivated the growing interest in formulating robust models and reliable algorithmic solutions suitable for high-frequency financial markets not only by the practitioners for better predictive performance and risk evaluation capability, but also by the market regulators for necessary supervision and improved market stability.

In particular and evolving over the past few decades, the limit order book (LOB) plays a key role in modern financial markets. The open LOB market, as a financial structure, has replaced various financial and economic systems that had dominated for several decades, to meet the rising demand for improved transparency and liquidity, more centralised clear-


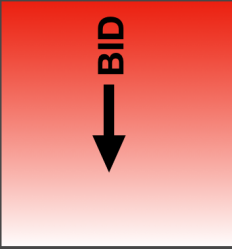
Limit Order Book		
	Price (\$)	Quantity (million)
	1.12962	6
	1.12961	9
	1.12960	9.5
	1.12959	19
	1.12958	22
	1.12954	20
	1.12953	4
	1.12952	2
	1.12951	3
	1.12950	2

Figure 1.1: A typical example of LOB at a single timestamp from the EUR-USD FOREX market. It only shows the five closest price levels to the mid-price from each of the bid and ask sides.

ing and lower transaction cost. In an order-driven LOB market, all buying/biding and selling/asking orders submitted by the market participants to the exchange are aggregated and displayed in the LOB. This provides a transparent and complete view of current and historic market information. The accurate modelling of LOBs has hence become the key to high-frequency and algorithmic trading.

Due to the modest margins that the high-frequency traders and trading firms often operate with, the fluctuation in the performance of the applied algorithms pertaining to an adopted model to be carefully quantified to avoid any unnecessary risks. Moreover with order execution times now measured in microseconds and market dynamics changing constantly intraday or even intra-hour, the online inference of price models and market structures becomes increasingly important in modern trading strategies and algorithms. These requirements have vitally challenged classic frequentist approaches. Thus in this thesis, technical tools from statistical signal processing and Bayesian stochastic modelling are utilised to achieve effective simulation and inference of the LOBs in high-frequency finance. Next, the background information on LOB as well as stochastic modelling and inference approaches are provided.

1.2 Background

In this section, selected relevant basic concepts from the open LOB market are first outlined with demonstrations of LOBs reconstructed from real-world market data. Secondly, some related work on state-space modelling and sequential Monte Carlo (SMC)/particle

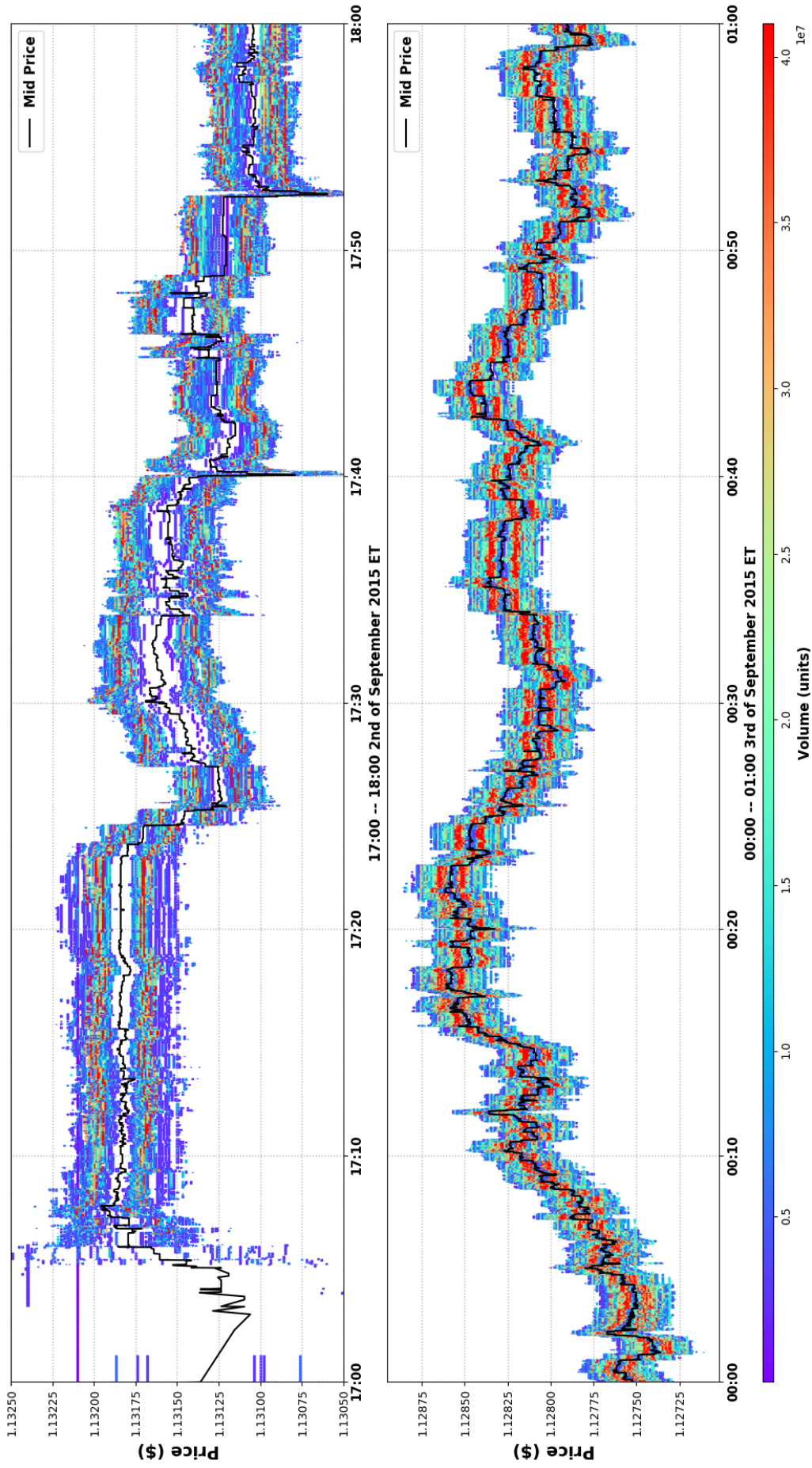


Figure 1.2: Two example plots of LOB snapshots, each for a duration of one hour in the EUR-USD FOREX market. The colormap indicates the relative outstanding limit order volumes remaining on the market.

filtering methods are highlighted.

1.2.1 Basics of limit order book

In an order-driven LOB market, market participants trade by submitting orders to the exchange, where centralised processing takes place. Regardless of buying or selling, orders can generally be divided into two classes based on their effective time of existence on the market: *limit orders* and *market orders*. A *limit order* states the intention to buy or sell (i.e. type) a certain amount (i.e. volume) of an asset at a specific price for a limited period of time (i.e. duration), and hence the name “*limit order*”. Such an order can be executed by any matching orders of opposite type from other traders, or cancelled before its expiration. Any outstanding limit orders are hence aggregated according to their prices and volumes to form the *limit order book* as shown in **Figure 1.1**. The highest buying price and the lowest selling price at a given timestamp are called the *best bid price* and the *best ask price* respectively; in this thesis, the *mid-price* is defined as the arithmetic mean of the best prices. Thereby, limit orders provide liquidity to the market by listing traders’ offers at a cost of time and uncertainty.

On the other hand, a *market order*, after its submission, is executed immediately at the current *best price* in the market. A market order will always reduce the amount/volume of the existing limit orders in the market and therefore demands/consumes liquidity. Any limit buying/selling orders submitted with prices higher/lower than the best ask/bid price will be made into *marketable limit orders* and executed immediately at the best prices. The outstanding quantities of limit orders get updated accordingly every time a market order is executed. Once the limit orders at the best prices deplete, the mid-price correspondingly shifts. The continuous submission and depletion of limit orders over time give rise to the complex price dynamics of a high-frequency financial market.

Figure.1.2 shows two plots of one-hour LOB reconstructed using raw data collected from the EUR-USD FOREX market. With the colours indicating the outstanding volumes of limit orders at different timestamps and price levels, there are empty (i.e. white) regions around the mid-price where no limit order exists. The width of the region, i.e. the difference between best prices, is called the *bid-ask spread*. The bid-ask spread can be used as an indicator for current market liquidity as can be seen from the figure. The top panel shows the LOB during the first hour since the opening of the market; whilst the bottom panel shows a maturer market with much higher order volumes (i.e. higher liquidity) and much narrower bid-ask spread. This spread in any highly liquid markets can be as small as one *ticksize* which is defined as the smallest price increment of that specific market, e.g. $one\ tick = 1 \times 10^{-5}$ in the EUR-USD FOREX market.

The terms introduced above will be referred to directly or indirectly throughout this thesis. While there are still many other concepts when it comes to trading in an open

LOB market such as *stop-limit order* and *one-cancels-the-other order* (OCO) that are often used by experienced practitioners to mitigate trading risk, my research so far does not feature the implications of these more practical concepts. For further information on LOB and their use in real-world financial markets, readers may refer to [4, 5] and the Investopedia website [6].

Based on the information content, LOB datasets used here for real-data testing can generally be classified into three different types:

- (i) ***Best bid-offer (BBO) data:*** The BBO data includes only the orders (with volumes) that are placed at the *best bid and ask prices* for each timestamp. The BBO data gives a concise view of the general trend of the market with necessary information for market-order traders.
- (ii) ***Snapshot (SNP) data:*** The SNP data gives the full LOB including prices and volumes of all limit orders from both sides of the mid-price at all timestamps when the snapshots are taken. **Figure 1.2** is a typical visualisation of the SNP data.
- (iii) ***Ticks (TKS) data:*** The TKS contains the most complete information of the LOB market. The dataset comprises a high-frequency stream of order transactions data including limit order arrivals/submissions, cancellations and modifications.

As each transaction necessarily changes the LOB structure, the TKS data can be used for LOB analysis at a much finer temporal resolution than the SNP data. However, as most order operations are noise to the actual market dynamics, the crude use of TKS data usually requires a large amount of computation and is inefficient for models and algorithms. For instance, later in **Chapter 2** a model which helps transform the TKS data into higher-level indicative information to LOB dynamics is introduced.

1.2.2 State-space modelling and particle filtering

The research reported in this thesis on high-frequency LOB closely revolves around two key technical concepts: the state-space models and the particle filter algorithm. Thus in this subsection, I present a concise literature review on these two terms, with more detailed and focused reviews presented in the later chapters.

State-space model (SSM) refers to a general class of probabilistic models [7] that describes the probabilistic dependence between the latent state variables and observations. The concept of “state space” originates from control engineering [8], which defines the state space as the Euclidean space where the variables on the axes are the state variables. Since its development, SSM in conjunction with its inference routine has been employed in many areas including system control [9], object tracking [10], image processing [11], economics [12], finance [13] and neuroscience [14]. Among these, the most well-known

and most studied inference algorithm is the Kalman filter [8] which is based on linear Gaussian SSMs and delivers optimal closed-form estimation performance. In addition to its high inference accuracy in certain settings, Kalman filter’s popularity is also owing to its simplicity and low computational complexity.

An important variant of the SSM is the hidden Markov model (HMM) where the sequential states are specified to be discrete variables. Similarly a mixed continuous-discrete state variable leads to another variant called the switching SSM [15].

With the increasing demand of realistic modelling of dynamic systems, the SSM is developed further to accommodate non-linear and non-Gaussian systems. The need of efficient sequential inference algorithm to handle intractable likelihoods and transition densities led to the development of the particle filter. Particle filter, which is also referred to as the sequential Monte Carlo (SMC) method [16], is a sampling-based algorithm for Bayesian state inference in sequential models. The technique was first proposed in [17] for Bayesian state estimation in a non-linear non-Gaussian setting of the SSM. Due to its ability to handle intractable probability densities, particle filtering received enormous attention and has been studied extensively in various aspects. The authors of [18] present an overview of various developments on the generic algorithm since its introduction; while the authors of [19] compare the characteristics of different resampling schemes within the particle filter. A more recent work [20] provides a concise survey of the ideas behind the particle filtering method from 1930 up to the present day.

1.3 Summary of contributions

Here, I briefly summarise the contributions of this thesis:

- The thesis proposes several novel models for different tasks in the modelling and inference of high-frequency LOB markets including intensity inference of limit order transactions, market fair price inference and learning and identification of market dynamic regimes.
- Capitalising on the sequential framework of the proposed model, I adopt various sequential Monte Carlo approaches such as sequential Markov chain Monte Carlo (SMCMC) and Rao-Blackwellised particle filter (RBPF) and propose novel extensions on the generic algorithms to better address the specific needs of the models and applications. I also propose adaptations of iterative algorithms, e.g. particle Markov chain Monte Carlo (PMCMC) and variational Bayes (VB), to perform accurate learning of the SSM parameters with intractable posteriors.
- The performance of the proposed models and inference algorithms are evaluated on both synthetic datasets and real-world high-frequency LOB datasets.

- The introduced models provide a solid foundation for practical applications to the real-world LOB markets by achieving online (continuous-time) state inference as well as batch-wise parameters learning. I also point out clear directions for future work and present some preliminary examples of these potential extensions.

1.4 Thesis outline

This thesis studies various aspects related to the modelling and the inference of high-frequency LOB market from a signal processing perspective. It is nonetheless emphasised that the novel stochastic models and sequential inference algorithms proposed and introduced here can be readily applied in various areas and disciplines beyond finance, such as object tracking, localisation, dynamic group clustering and others. The layout of chapters in this thesis attempts to accurately reflect the progression of the undertaken research over the period of my PhD research. Here, I outline the main focus of research for each chapter and present the continuity of investigation from chapter to chapter.

Chapter 2 focuses on the fundamental element of LOB market, namely the limit orders and their operations. In order to process the highly granular point data of order operations and translate them into interpretable market indicators, the non-homogeneous Poisson process is adopted together with the development of a novel sequential approach of performing Bayesian inference on the intractable intensity functions. By combining the continuous-time SSM with the SMC MC algorithm, the proposed method is able to efficiently infer the time-varying intensities of various limit order operations on the market and consequently serves as a foundation for the study of LOB structures and the prediction of market trends.

Both **Chapter 3** and **4** studies the fair price process of LOB market. Based on the fundamental structure of the market, the research aims to capture the price dynamics by incorporating various real-world market intuitions. **Chapter 3** proposes several new models that account for dynamical supply-demand imbalance, momentum/trends of different time scales, as well as a jump-resetting mechanism. These models help achieve more realistic modelling, more accurate predictions and better retrospective analyses of the underlying fair price process. In **Chapter 4**, an alternative perspective to the market is taken with the aim to connect the arrival of limit orders operations with the innovation of price movements. This leads to the development of a multi-jump diffusion model with the ability to capture both long-term and short-term price dynamics. Moreover, this formulation allows the stochastic processes of order operations studied in **Chapter 2** to be connected with their potential impacts on the market price. The inference of the multi-jump diffusion model is handled by the semi-deterministic particle filter algorithm which is a novel adaption of the classic deterministic particle filter [21] to the continuous jump-time space.

Whilst **Chapter 3** and **4** each focuses on the design of a single diffusion model to incorporate more real-world market features, the multi-regime nature of the LOB market is recognised and researched in **Chapter 5**. Like many other complex dynamical systems, price processes in financial markets constantly exhibit different movement patterns intra-day or even intra-hour. In order to accommodate this multi-regime nature while being able to learn the diffusion parameters without significant prior assumptions, a model is proposed in this chapter, which elegantly integrates the continuous-time SSM, the hidden semi-Markov model (HsMM) and the non-parametric Dirichlet process model (DPM). By utilising the PMCMC algorithm framework, a sophisticated Bayesian inference algorithm is also proposed to achieve accurate online state inference, regime allocation as well as off-line parameters learning. Applied to both financial LOB data and animal movement data, it is also demonstrated that the model can provide useful insights to various not priorly known modes of motion.

Due to the ever-changing nature of the LOB market, parameter learning and hyperparameter tuning play crucial roles in the modelling of market dynamics. The continuous-time construction of the model and the amount of time-series data have both challenged the tractability and the complexity of parameter-learning algorithms. Therefore in **Chapter 6**, I propose novel inference algorithms that, in both online and off-line settings, perform Bayesian parameters learning on the dynamics-controlling parameters in SSMs. Additionally, the algorithms' performance is benchmarked against some well-established algorithms.

Chapter 2

Bayesian sequential inference for non-homogeneous Poisson process intensity

2.1 Background

The predictions of LOB structure and the market fair price are two of the most challenging tasks in studying high-frequency financial market. While the long-term prediction is affected by various factors either within or outside the market through some extremely convoluted links, the short-term evolution of LOB is closely related to different operations on limit orders such as submissions, cancellations and executions. As briefly outlined in Section 1.2, the depletion of best order queues not only changes the general structure of LOB but also shifts the market mid-price. The rate of depletion, injection and replacement of limit order queues predominately determines the (short-term) dynamics of the high-frequency LOB market. Therefore in this chapter, the research focuses on the orders, and more specifically on the inference of intensities of limit order operations. The research covered in this chapter has been published in [22]¹ and [23]² and the content of this chapter reuses a proportion of the texts, figures and tables in these two papers.

In order to infer the sought intensities, it is necessary to work with the point data of basic order operations. The datasets used in this chapter are thus extracted from the LOB *ticks data*, which contains the fullest information of the market by recording every single transaction/operation. The occurrences of these transactions are grouped together based on their operation types and price to provide suitable datasets for evaluating the efficacy of the developed modeling and corresponding inference algorithms.

¹© 20XX IEEE. Reprinted, with permission, from [22]

²© 20XX IEEE. Reprinted, with permission, from [23]

With the aim of developing a generic approach to perform intensity inference on order operations, Poisson process stands out for its ability to model point data in both temporal and spatial settings with applications across various disciplines. For instance, the neuronal spike trains are often modelled as Poisson processes [24], as well as the earthquake sequences [25]. It is also widely studied and applied in the field of teletraffic engineering [26]. Owing to the convenient mathematical properties of the Poisson process, it was often combined with other sophisticated models like SSMs to capture salient features of dynamical systems with jumps [27, 13] and/or discrete occurrences such as signal detection and false alarms [28] in the field of tracking.

Instead of using the standard Poisson process, a more powerful variant is employed here – the non-homogenous Poisson process (NHPP). As its name suggests, the NHPP provides additional flexibility by allowing the intensity function to vary across time and/or space instead of being constant (i.e. homogeneous). This gives a more realistic modelling option for the application on high-frequency limit orders as intensities are typically expected to change significantly intra-day and even intra-hour. Next, the definition of NHPP and its simulation procedure is briefly introduced before reviewing the related work on such processes.

2.1.1 Non-homogeneous Poisson process

The NHPP can be defined in several equivalent ways, each fitting the general definitions of a Poisson process. An intuitive definition of the NHPP is introduced here as in [29]:

Definition 1. *Non-homogeneous Poisson Process: For a domain $\mathcal{S} = \mathbb{R}^D$, define a NHPP with an intensity function $\lambda(s) \geq 0, s \in \mathcal{S}$, and the counting measure $N(\mathcal{T})$ (i.e. number of occurrences) in any bounded region $\mathcal{T} \subset \mathcal{S}$ such that:*

1. $N(\emptyset) = 0$
2. $\{N(\mathcal{T}_i)\}_i$ are independent for any disjoint subsets $\{\mathcal{T}_i\} \subset \mathcal{S}$
3. $N(\mathcal{T}) \sim \text{Poisson}(\Lambda)$, with $\Lambda = \int_{\mathcal{T}} \lambda(s) ds$

Let $\{s_k\}_{k=1}^K$ be a set of K events/occurrences in a region \mathcal{T} ; then with the definition above, the likelihood function of the NHPP with intensity $\lambda(s)$ can be written as the product of three probabilities: (1) the Poisson probability of observing K events in \mathcal{T} : $\frac{e^{-\Lambda} \Lambda^K}{K!}$; (2) the density of the events $\{s_k\}_{k=1}^K$: $\prod_{k=1}^K \frac{\lambda(s_k)}{\Lambda}$; and (3) the $K!$ number of possibilities of ordering the K events. Thus the likelihood can be expressed as:

$$\begin{aligned}
 p(\{s_k\}_{k=1}^K | \lambda(s), \mathcal{T}) &= \frac{e^{-\Lambda} \Lambda^K}{K!} \times \prod_{k=1}^K \frac{\lambda(s_k)}{\Lambda} \times K! \\
 &= \exp\left\{-\int_{\mathcal{T}} \lambda(s) ds\right\} \prod_{k=1}^K \lambda(s_k)
 \end{aligned} \tag{2.1}$$

Clearly, the above likelihood cannot be computed directly as it requires not only pointwise evaluations of the intensities at all event times/locations $\{s_k\}_{k=1}^K$, but also an integration of the intensity function $\lambda(s)$ over the entire domain of interest \mathcal{T} . This integration is generally intractable, except for certain intensity functions of simple forms. Such intractability severely inhibits the inference of intensity via most likelihood-based or Bayesian inference approaches.

It is nonetheless possible to obtain a tractable integration by assume a simple known functional form (e.g. linear) of the intensity function with unknown parameters and perform certain inference on these parameters. However, such assumption can largely limit the practicality of the NHPP and can give a poor approximation of the true intensity curve.

2.1.2 Thinning and simulation

Despite the intractability in the likelihood of NHPP, it is still possible to perform exact (i.e. unbiased) simulation of a NHPP from a very board class of intensity functions without computing the integral. This can be achieved by *thinning* [30]. The thinning operation entails removing point(s) from an existing point process based on some predefined rules in order to produce a new point process.

While there are many variants of thinning operations, this thesis focuses on independent thinning, where the decision about removing each point is made by independent Bernoulli trials and the interaction between points has no effect on this decision [31]. Starting with the following theorem:

Theorem 1. *Consider a homogeneous Poisson process (HPP) with constant intensity λ^* over a domain $S = \mathbb{R}^D$, so that the counting measure over any bounded region $\mathcal{T} \subset S$ is $N^*(\mathcal{T}) \sim \text{Poisson}(\lambda^*|\mathcal{T}|)$, where $|\mathcal{T}|$ is the Lebesgue measure of \mathcal{T} . If the points of this process undergo an independent thinning operation with a spatially varying deletion probability $1 - p(s)$, the remaining points form a NHPP with intensity function $\lambda^*p(s)$ within region \mathcal{T} .*

The proof of **Theorem 1** can be found in **Appendix 2.A**. Following this theorem to generate a NHPP with the desired intensity function $\lambda(s)$, the simulation can simply start with a HPP having intensity λ^* and perform the described thinning operation on each homogeneous with a temporally or spatially varying Bernoulli (retaining) probability:

$$p(s) = \frac{\lambda(s)}{\lambda^*}, \quad \lambda^* \geq \sup_{s \in S} \{\lambda(s)\} \quad (2.2)$$

The value of λ^* should be chosen such that it serves as an upper bound on $\lambda(s)$. It is also worth noting that the starting process (of counting measure $N(\mathcal{T})$) is not restricted to

Algorithm 1 Simulation of a NHPP

Inputs: domain of interest $\mathcal{T} \subset \mathcal{S}$, intensity $\lambda(s)$, upper-bound intensity λ_{\max}
Outputs: A set of (random) K non-homogeneous events $\Phi = \{s_k\}_{k=1}^K$ within \mathcal{T}

- 1: $N \sim \text{Poisson}(\lambda_{\max}|\mathcal{T}|)$ ▷ Compute the number of occurrences in homogeneous Poisson process
- 2: $\{s_n\}_{n=1}^N \sim \text{Uniform}(\mathcal{T})$ ▷ Sample the homogeneous occurrence locations
- 3: $\Phi \leftarrow \emptyset$ ▷ Initialise an empty set for NHPP events
- 4: **for** $n = 1 : N$ **do**
- 5: $\rho_n \sim \text{Uniform}(0, 1)$
- 6: **if** $\rho_n \leq \lambda(s_n)/\lambda_{\max}$ **then** ▷ Apply the thinning operation
- 7: $\Phi \leftarrow \Phi \cup s_n$
- 8: **end if**
- 9: **end for**
- 10: **Return:** Φ

HPP and the method of generating non-homogeneous realisations via thinning can generalise to any NHPP as long as the starting intensity $\lambda^*(s)$ is an envelope of the desired intensity $\lambda(s)$ such that $\lambda^*(s) \geq \lambda(s)$, $\forall s \in \mathcal{T}$ [30]. From this, a close relation between *thinning* and *rejection sampling* (RS) can be observed; the tighter the envelope $\lambda^*(s)$ is, the more efficient simulation will be.

Consequently, without requiring any form of integration any NHPP, whose intensity function is upper-bounded and which can be evaluated point-wise, may be simulated. A set of N homogeneous Poisson points $\{s_n\}_{n=1}^N$ are generated by first drawing the variable N from a Poisson distribution with parameter $(\lambda^*|\mathcal{T}|)$, followed by N independent uniform random draws within \mathcal{T} . The homogeneous points are then thinned with Bernoulli probabilities $(1 - \lambda(s)/\lambda^*)$ to provide the NHPP events having the desired intensity function $\lambda(s)$. **Algorithm 1** shows the detailed procedure of simulating a NHPP, accompanied by a graphical illustration in **Figure 2.1**.

It can be noted that the application of **Theorem 1** requires an assumption of an attainable maximum intensity λ^* of the NHPP. Although this premise may seem restrictive, it can generally be satisfied in most real-world applications especially as λ^* is not necessarily a *tight upper bound* or a *supremum* of the intensity function $\lambda(s)$. In practice, an intuitive value of λ^* is often inherently defined by the modelled systems, e.g. the maximum number of phone calls that can be processed by the call center at one time. For the real data experiment presented in Section 2.5.2, the maximum number of limit order arrivals is often restricted by the exchange. However, this intuitive bound may not be the best choice for λ^* since the tightness of this upper bound is closely related to algorithm efficiency (for both simulation and intensity inference). The value of λ^* should be tuned to accommodate the requirements of specific application.

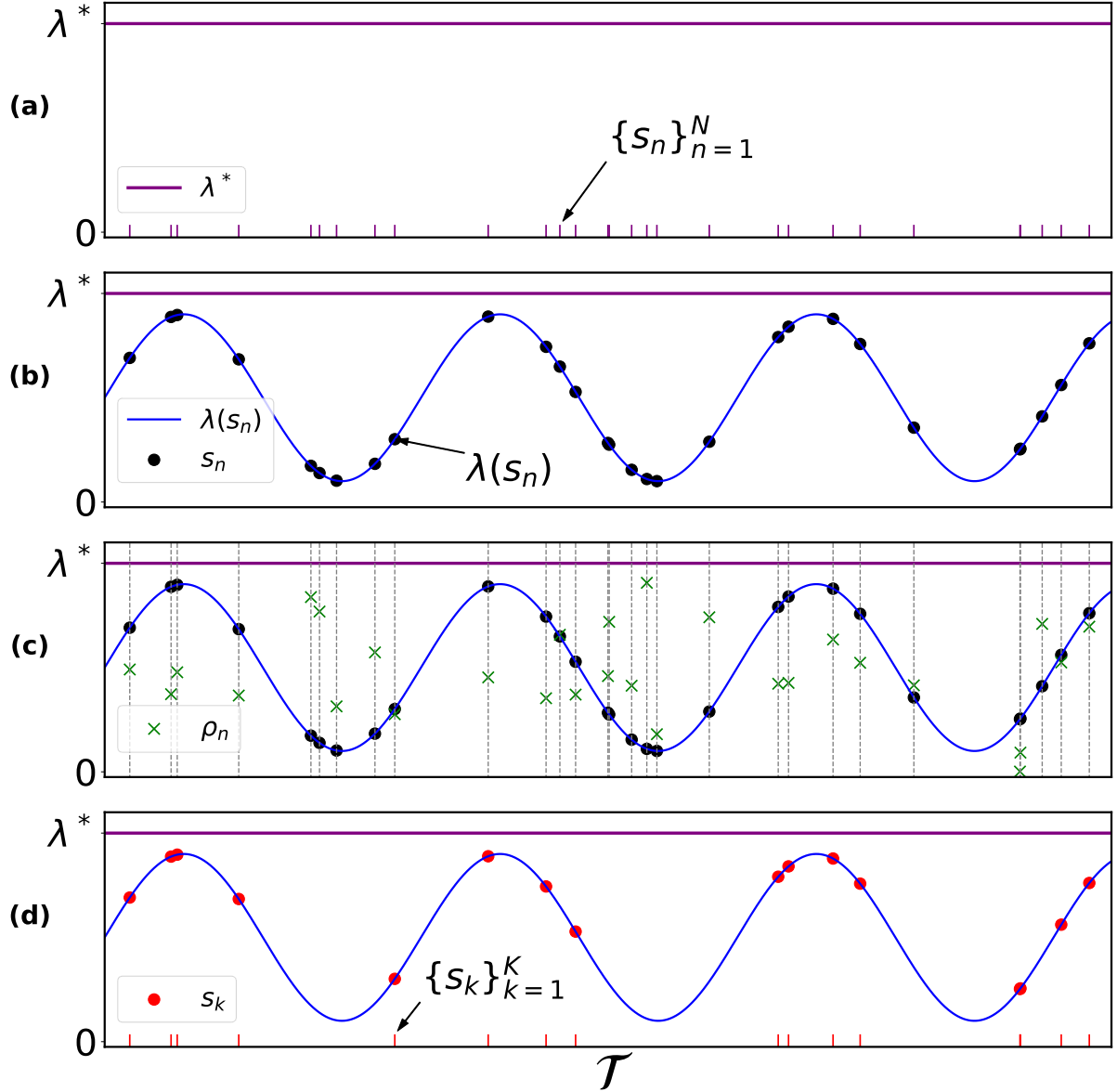


Figure 2.1: The figure shows the generative procedure of the NHPP from a periodical intensity function: (a) The homogeneous Poisson points generated from the upper-bound intensity λ^* . (b) For each point, the non-homogeneous intensity is evaluated with the analytical function. (c) Each point samples a variate uniformly from $(0, \lambda^*)$. (d) Only points with the proposed variates lower than their corresponding intensity $\lambda(s_n)$ get accepted.

2.2 Review of existing methods

The attractive features of NHPP have led to an extensive amount of research interests into the intensity inference of the NHPP. For example, the early frequentist approach proposed in [32] uses kernel densities to construct a non-parametric intensity estimator of the NHPP. Another frequentist method developed in [33] assumes a piecewise-linear parametric form of the intensity function and estimates the function parameters via regression. The continuous piecewise-linear method was later advanced in [34] and [35] by

formulating the intensity inference problem as a constrained quadratic programming problem and a convex optimisation problem respectively. Both of these frequentist methods have achieved relatively fast estimation of intensity, however without providing any uncertainty quantification. This renders their inference accuracy rather sensitive to the choice of hyperparameters and model assumptions.

The popularity of Poisson process in the field of tracking has also encouraged the development of intensity inference approaches. The early work in [28] introduced a method called the probability hypothesis density (PHD) filter which recursively estimates the unknown time-varying number of targets (and their states) at discrete intervals of time. This filter was later extended in [36, 37] (with practical applications in [38]) to the intensity filter which infers the posterior intensity of the Poisson process approximation as the scaled marginal probability density. Although the intensity filter and its variants are formulated as online recursive approaches and require little computation, such methods typically assume a constant intensity value within the update interval and hence a piecewise constant intensity function.

The piecewise constant assumption on intensity function is fairly common for inference of the NHPP. The change-point approaches for the NHPP rely heavily on this assumption with early studies in [39, 40, 41] where the models assume only two (homogeneous) Poisson intervals (i.e. one change point). Based on these studies, multiple-interval models were later developed [42] and a more recent paper [43] proposed a Bayesian inference approach which also includes an optimised/learned number of intervals. Despite the similarity, the change-point methods focus mainly on the inference of exact timings of the changes; while the computation of intensity is often trivial. It is worth noting that the work in this chapter assumes a general unknown continuous intensity function, which involves a completely different inferential task to the change-point methods.

One major challenge of inference for the NHPP is that it often involves a non-preconceived form of the intensity function. This gives rise to the idea of doubly-stochastic Poisson process, or a Cox process [29]. The Cox process allows its unknown intensity function to be governed by another (continuous) stochastic process. The Gaussian Cox process, where the intensity function is a transformation of a random realisation from a Gaussian process (\mathcal{GP}) [44], provides a convenient way to specify general prior belief on the unknown intensity function. Non-parametric Bayesian intensity inference approaches of such a process were studied in both [45] and [46] using the log Gaussian Cox process (LGCP) model. Both approaches introduced finite-dimensional proxy distribution via discretisation. The authors of [47], for the first time, proposed the sigmoidal Gaussian Cox process (SGCP) model and a tractable approach to perform fully Bayesian intensity inference via Markov chain Monte Carlo (MCMC) algorithms. Also in the paper, the SGCP model demonstrated superior intensity inference performance than the LGCP model. Based on the ideas in [47], the SGCP model was later extended in [48] to include

variational sampling of hyperparameters and parallel inference for multiple correlated processes. However, both approaches scale poorly with the size of the dataset due to the high complexity of the Gaussian process prior and Bayesian computation. Inspired by sparse Gaussian process models, [49] uses generative *inducing points* to perform tractable variational inference on NHPP likelihood function, which achieves a complexity that scales better with input dataset size. A more recent paper of [43] also proposes a Bayesian inference approach for Poisson point processes but requires a piecewise constant assumption on intensity function.

2.2.1 Sigmoidal Gaussian Cox process (SGCP)

Before introducing the novel model developed in this work, I present a brief review on the SGCP model in [47], which forms a starting point of the proposed approach.

Denote the unknown varying intensity function of a doubly-stochastic Poisson process as $\lambda(s)$ which is governed by another stochastic process $\{g(s), s \in \mathcal{T}\}$. The SGCP model incorporates a Gaussian process (\mathcal{GP}) as the intensity prior:

$$g(s) : \mathcal{S} \rightarrow \mathbb{R} \quad , \quad g \sim \mathcal{GP}\left(m(\cdot), C(\cdot, \cdot)\right) \quad (2.3)$$

where $m(\cdot) : \mathcal{S} \rightarrow \mathbb{R}$ is the mean function and $C(\cdot, \cdot) : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ is the positive definite covariance function. The scalar function $g(s)$ is then mapped to the non-negative intensity function through a scaled sigmoid function $\lambda(s) = \lambda^* \sigma(g(s))$, with $\sigma(x) = (1 + e^{-x})^{-1}$.

Inspired by the constructive generative process of the NHPP introduced in Section 2.1.2, the SGCP model achieves tractability by considering the observed NHPP as the output from a thinning operation applied to a latent HPP. In such a case, the retaining probability for an observed (i.e. input) point is:

$$p(s) = \frac{\lambda(s)}{\lambda^*} = \sigma(g(s)) \quad (2.4)$$

Define $i_n \in \{0, 1\}$ as an indicator associated with each Poisson event, taking value 0 for an observed data point and 1 for a ‘thinned’ point. And thus write the augmented joint probability of the SGCP model as:

$$\begin{aligned} & p(\{s_n\}_{n=1}^N, \mathbf{g}_{1:N}, \{i_n\}_{n=1}^N \mid \lambda^*, \mathcal{T}) \\ &= \underbrace{(\lambda^*)^N e^{-\lambda^*|\mathcal{T}|}}_{(1)} \underbrace{p(\mathbf{g}_{1:N} \mid \{s_n\}_{n=1}^N)}_{(2)} \underbrace{\prod_{n=1}^N \sigma\{(-1)^{i_n} g(s_n)\}}_{(3)} \end{aligned} \quad (2.5)$$

where $\mathbf{g}_{1:N}$ is the concatenated vector with $\mathbf{g}_{1:N} = [g(s_1), g(s_2), \dots, g(s_N)]^T$. The for-

mulation of Eq. (2.5) follows similar steps to the simulation in **Algorithm 1**, with the addition of random simulation from the intensity function: (1) represents the probability of generating N ordered points in \mathcal{T} according to the upper-bound intensity λ^* ; (2) is the probability of generating stochastic process values $\mathbf{g}_{1:N}$ at times $\{s_n\}_{n=1}^N$ from the prior; and (3) is the probability of the Bernoulli trials, since $1 - \sigma(x) = \sigma(-x)$. Inference for the SGCP model is achieved in [47] by offline batch-based MCMC samplers for each latent variable which alternate in a Gibbs sampling manner.

The SGCP model facilitates a tractable inference procedure for NHPPs. However the practical application of the model is limited by the $\mathcal{O}(N^3)$ complexity arising from the \mathcal{GP} prior and the corresponding MCMC inference routine. The value of N here is the total number of homogeneous points, and this can be much larger compared to the number of input points when the intensity function has regions of low values compared to the upper-bound intensity λ^* . Furthermore, the batch-based MCMC sampler provides only retrospective knowledge of the NHPP and cannot be readily adapted to a sequential, online setting.

2.3 Model

In order to alleviate the aforementioned limitations of NHPP modelling, a new state-space intensity model is developed in this thesis under the generative thinning framework. As illustrated here, this allows us to employ efficient sequential Bayesian inference techniques.

2.3.1 Continuous-time state space model

In order to construct a computationally tractable sequential framework for NHPP data while allowing flexibility in the choice of prior characteristics, a continuous-time SSM is employed to replace the fully correlated Gaussian process prior. Denoting $\mathbf{g}(t)$ as the state vector at time t , the SSM can be formulated as the following stochastic differential equation (SDE):

$$d\mathbf{g}(t) = \mathbf{A}\mathbf{g}(t) dt + \mathbf{h} dW_t \quad (2.6)$$

where $\{W_t\}$ is a Wiener process. Such a model, which is linear and Gaussian, can be discretised exactly in closed form using Itô calculus. The solution of the SDE (see **Appendix 2.B** for derivations), integrating from arbitrary time P to Q for $Q \geq P$, is:

$$\mathbf{g}(Q) = e^{\mathbf{A}(Q-P)} \left[\mathbf{g}(P) + \int_0^{Q-P} e^{-\mathbf{A}\tau} \mathbf{h} dW_\tau \right] \quad (2.7)$$

and the conditional transition density $p(\mathbf{g}(Q)|\mathbf{g}(P))$ can be readily computed to be Gaussian and Markovian:

$$p(\mathbf{g}(Q) | \mathbf{g}(P)) \sim \mathcal{N}(\mathbf{g}(Q) | \boldsymbol{\mu}(Q, P), \mathbf{C}(Q, P)) \quad (2.8)$$

with mean and covariance calculated directly from the stochastic integral in Eq. (2.7):

$$\boldsymbol{\mu}(Q, P) = \mathbb{E}\{\mathbf{g}(Q) | \mathbf{g}(P)\} = e^{\mathbf{A}(Q-P)} \mathbf{g}(P) \quad (2.9)$$

$$\begin{aligned} \mathbf{C}(Q, P) &= \mathbb{E}\left\{ [\mathbf{g}(Q) - \boldsymbol{\mu}(Q, P)] [\mathbf{g}(Q) - \boldsymbol{\mu}(Q, P)]' \right\} \\ &= e^{\mathbf{A}(Q-P)} \mathbf{K}(Q, P) (e^{\mathbf{A}(Q-P)})' \end{aligned} \quad (2.10)$$

where

$$\mathbf{K}(Q, P) = \int_0^{(Q-P)} e^{-\mathbf{A}\tau} \mathbf{h} \mathbf{h}' (e^{-\mathbf{A}\tau})' d\tau \quad (2.11)$$

The computation of $\mathbf{K}(Q, P)$ is non-trivial and can be obtained using matrix fraction decomposition [50] or approximated by series expansion of the exponential functions [51]. With the above definition of the conditional transition density, and a Gaussian initial state prior, it is possible to obtain the joint probability of all or part of the state vector by conditioning and probability chain rule, capitalising on the Markovian property.

While any Gaussian SSM could in principle be applied in the above framework, this chapter have adopted for illustration a Langevin dynamical model that is similar to that used in [13]. In such a model, the state vector $\mathbf{g}_t = [g_{1,t}, g_{2,t}]^T$ contains a value term $g_{1,t}$ and a stochastic trend term $g_{2,t}$ at time t . The general SDE in Eq. (2.6) is reformulated as:

$$d \begin{bmatrix} g_{1,t} \\ g_{2,t} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \theta \end{bmatrix} \begin{bmatrix} g_{1,t} \\ g_{2,t} \end{bmatrix} dt + \begin{bmatrix} 0 \\ \sigma \end{bmatrix} dW(t) \quad (2.12)$$

where θ is the non-positive reversion coefficient and $\sigma > 0$ is the scale of the trend process. The Langevin dynamics have the advantage of being analytically tractable while allowing either long-term or short-term trend behaviours of the intensity depending on the choice of θ . Denote the joint prior under this model omitting the dependency on hyperparameters as:

$$p(\mathbf{g}_{1:N} | \{s_n\}_{n=1}^N) = \mathcal{LD}(\mathbf{g}_{1:N} | \{s_n\}_{n=1}^N) \quad (2.13)$$

where $\mathbf{g}_n = [g_{1,s_n}, g_{2,s_n}]^T$ and $\{s_n\}_{n=1}^N$ are the time stamps.

2.3.2 Doubly-stochastic process with SSM dynamics

Under the Gaussian assumption, the stochastic process values $g_{1,s}$ can lie anywhere on the real line. Hence, as in [47], the sigmoidal mapping onto $[0, \lambda^*]$ is also adopted in this

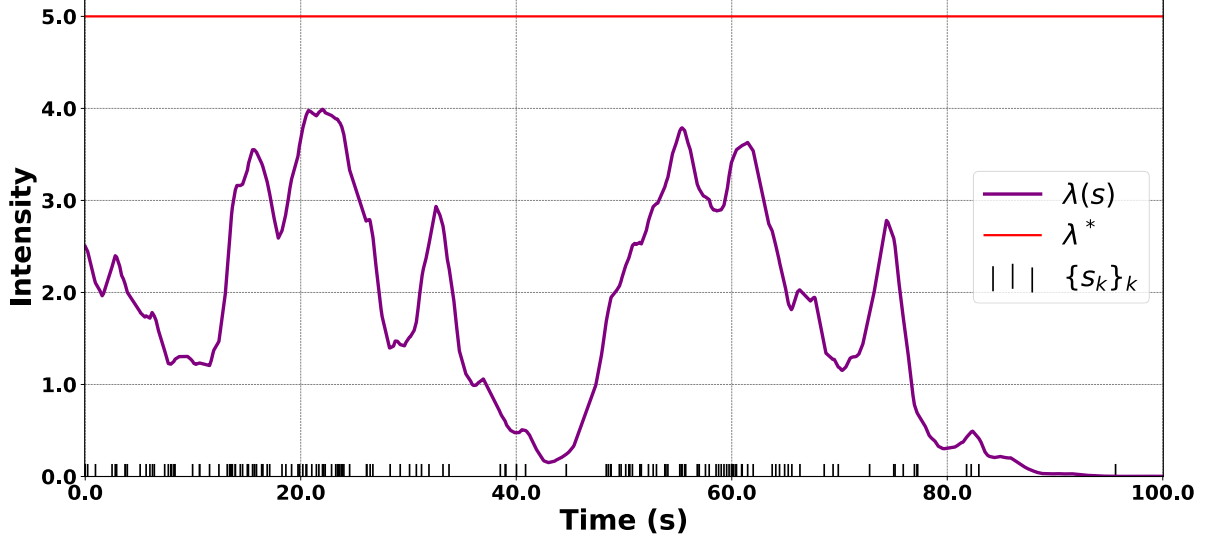


Figure 2.2: A random realisation of the doubly-stochastic Poisson process with a Langevin prior on $\mathcal{T} = [0, 100]$. Simulated NHPP events are shown as scatter lines on the time/ x axis.

model:

$$\lambda(s) = \lambda^* \sigma(g_{1,s}) \quad (2.14)$$

although note that any suitable function $\sigma(\cdot)$ that maps to $[0, 1]$ could be used in place of the sigmoidal function. Under this formulation, one may use **Algorithm 1** to generate realisations of the NHPP with dynamics specified in (2.12). With event timestamps $\{s_n\}_{n=1}^N$ proposed from the HPP with intensity λ^* , the state vectors $\{\mathbf{g}_n\}_{n=1}^N$ and corresponding intensities $\{\lambda(s_n)\}_{n=1}^N$ are evaluated sequentially through the conditional transition density in (2.8) and the mapping function in (2.14). **Figure 2.2** shows a typical realisation of a NHPP generated with Langevin governing stochastic intensity (in solid purple line). It is worth noting that the generation of both intensity process and NHPP events are random/stochastic (and hence the name doubly-stochastic). Different realisations of this process are used below in testing of the proposed model and inference methods.

2.4 Inference

As in the SGCP model, the proposed model works with the tractable augmented joint probability in Eq. (2.5), with the prior being the SSM. However unlike the \mathcal{GP} in [47], the Markovian property of the SSM allows efficient sequential inference of the intensity. By inputting short batches of data delineated by times t_k , $k = 0, 1, \dots$, inference is updated with arriving data for each batch. The time intervals $\mathcal{T}_k = (t_{k-1}, t_k] \subset \mathcal{T}$ could be e.g. regularly spaced, $t_k = k\delta_T$, or spaced according to the timings of the input (observed) points $\{s_n; i_n=0\}$ supporting asynchronous data processing.

Further define the notation $x_k = \{s_n, \mathbf{g}_n, i_n; s_n \in \mathcal{T}_k\}$ as the locations, the state

vectors, and the indicators corresponding to all events in the interval \mathcal{T}_k . Note that x_k includes both unobserved (latent) and observed components of the model. One can write the recursion for the joint distribution as:

$$p(x_{1:k} | \lambda^*, \mathcal{T}_{1:k}) = p(x_{1:k-1} | \lambda^*, \mathcal{T}_{1:k-1}) p(x_k | \lambda^*, x_{1:k-1}, \mathcal{T}_k) \quad (2.15)$$

where the second term of the conditional propagation can be conveniently factorised based on Eq. (2.5) as:

$$\begin{aligned} & p(x_k | \lambda^*, x_{1:k-1}, \mathcal{T}_k) \\ &= p(x_k | \lambda^*, x_{k-1}, \mathcal{T}_k) \\ &= (\lambda^*)^{N_k} e^{-\lambda^* |\mathcal{T}_k|} \times \mathcal{LD}(\{\mathbf{g}\}_{\mathcal{T}_k} | \{\mathbf{g}\}_{\mathcal{T}_{k-1}}, \{s\}_{\mathcal{T}_k}) \\ &\quad \times \prod_{n: s_n \in \mathcal{T}_k} \sigma\{(-1)^{i_n} g_{1,n}\} \end{aligned} \quad (2.16)$$

and $N_k = |\{n; s_n \in \mathcal{T}_k\}|$ is the total number of HPP events in \mathcal{T}_k . Note that the number of events N_k in \mathcal{T}_k is itself a random variable, therefore extra care is needed in performing the correct sequential inference. Thereby, a suitable scheme is proposed below.

Suppose that at time interval $k-1$, there is a large collection of random and possibly weighted samples (‘particles’) drawn from the posterior joint distribution at $k-1$ with the p th particle and its corresponding weight denoted as:

$$\{x_{1:k-1}^p, w_{k-1}^p\} \sim p(x_{1:k-1} | \lambda^*, \mathcal{T}_{1:k-1}), \quad p = 1, \dots, N_p \quad (2.17)$$

One can therefore approximate $p(x_{1:k-1} | \lambda^*, \mathcal{T}_{1:k-1})$ (the ‘smoothing’ distribution at $k-1$) with the empirical distribution of the particles:

$$p(x_{1:k-1} | \lambda^*, \mathcal{T}_{1:k-1}) \approx \sum_{p=1}^{N_p} w_{k-1}^p \delta_{x_{1:k-1}^p}(x_{1:k-1}) \quad (2.18)$$

with $w_{k-1}^p \geq 0$ and $\sum_p w_{k-1}^p = 1$. Combining the above approximated posterior with the factorised joint recursion of Eq. (2.15) and (2.16), an updated particle posterior distribution at interval k is obtained:

$$p(x_{1:k} | \lambda^*, \mathcal{T}_{1:k}) \approx \sum_{p=1}^{N_p} w_{k-1}^p \delta_{x_{1:k-1}^p}(x_{1:k-1}) \times p(x_k | \lambda^*, x_{k-1}^p, \mathcal{T}_k) \quad (2.19)$$

The above equation gives a mixed discrete-continuous distribution containing the point masses for the “past history” variables $x_{k-1} = x_{k-1}^p = \{s_n, \mathbf{g}_n, i_n; s_n \in \mathcal{T}_{k-1}\}^p$ and the conditional distributions for the “new variables” x_k . One can now propose samples jointly from this entire approximated distribution of past and new variables and compute the importance weights or MCMC acceptance probabilities, leading to standard particle fil-

tering methods or SMC MC procedures respectively. In either case, it is helpful to keep in view that the samples produced are *joint* samples approximating the posterior for all $\mathcal{T}_{1:k}$.

The posterior propagation proceeds by selecting one particle, say $p = \tilde{p}$, randomly from the smoothing distribution in Eq. (2.18) represented empirically by the ‘history’ collection at the end of interval \mathcal{T}_{k-1} . Based on the drawn particle $x_{1:k-1}^{\tilde{p}}$, a new set of variables x_k is proposed from either priors or pre-assigned proposal distributions with its weight/acceptance ratio computed accordingly. Enough repetitions of this procedure during interval \mathcal{T}_k will yield a set of importance-weighted/converged samples from the joint smoothing distribution $p(x_{1:k}|\lambda^*, \mathcal{T}_{1:k})$. Note that for mathematical convenience, I treat the observed events $\{s_n; i_n = 0\}$ jointly with the latent events as random variables whose values are known with probability 1, and hence simply chosen deterministically in the proposal step. Note that Eq. (2.19) approximates the joint distribution of input points $\{s_n; i_n = 0\}$ and all the remaining unknowns in the system. Since this joint distribution is directly proportional to the posterior distribution of the unknown state elements, conditional on a particular realisation of the input points (the ‘data’), one may obtain posterior Monte Carlo samples simply by extracting the Monte Carlo samples of the unknowns and excluding the known fixed input points. **Figure 2.3** shows graphically the scheme of propagation as described in Eq. (2.19) across batches for different particles in the algorithm.

So far, the described scheme can be easily implemented with the variable-rate particle filter (VRPF) [27]. Such an approach however is not especially effective for this task as the factorisation in Eq. (2.16) requires the proposal of multiple latent variables of varying dimension in a single propagation step, which will inevitably result in the inherent weight degeneracy problem of the particle filter. Instead here, this high-dimensional proposal is addressed with the SMC MC algorithm which targets sequentially the joint distributions of Eq. (2.19), using both local and global Metropolis-Hastings (MH) accept-reject moves instead of importance sampling or resampling [52, 53, 54, 55].

Furthermore, a mixture sampling procedure is adopted: at each MCMC iteration, a decision is made on performing either a joint MH proposal step with probability P_J or a sequence of individual refinement Metropolis-within-Gibbs (MwG) transitions with probability $1 - P_J$. Such a scheme provides an effective trade-off between the speed and accuracy of the inference via adjusting the value of P_J . Next, the generic algorithm of SMC MC is briefly covered before proceeding to the detailed design of joint proposal and refinement steps in Section 2.4.2, 2.4.3 and 2.4.4.

2.4.1 A generic SMC MC algorithm

Similar to many other Monte Carlo methods, the generic algorithm of SMC MC also tries to approximate the posterior density with an empirical representation of the particles.

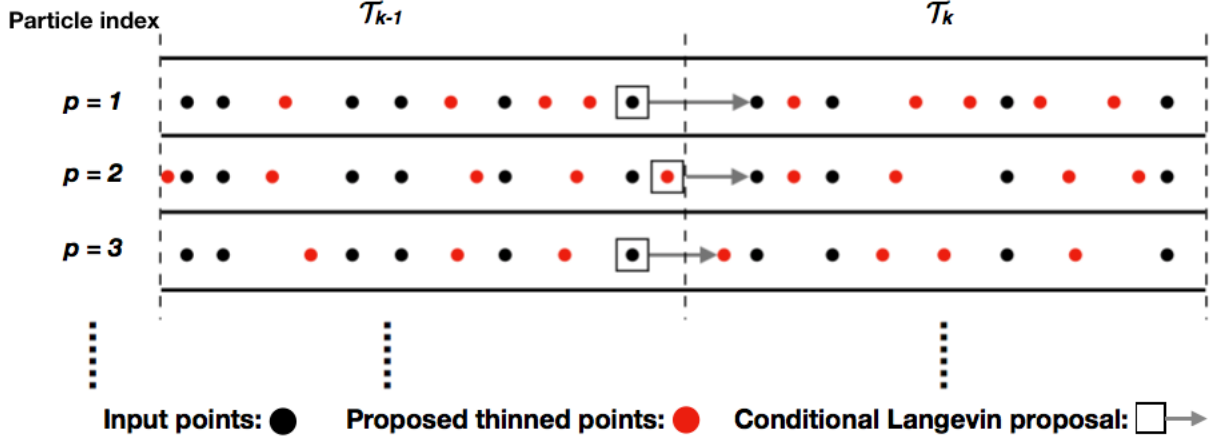


Figure 2.3: Propagation scheme of the SMCMC algorithm across the batch boundary between \mathcal{T}_{k-1} and \mathcal{T}_k . Number and locations of ‘thinned’ events are proposed independently for each particle p .

However in contrast to a standard particle filter, the particles in SMCMC are not weighted and each is guaranteed to be a representative sample of the posterior density with the assumption that the MCMC run has converged.

Targeting the posterior joint density, e.g. Eq. (2.19) at each propagation, the SMCMC algorithm selects historic particles from $\{x_{1:k-1}^p\}_{p=1}^{N_p}$ and conditionally proposes new latent variables x_k from one joint MCMC kernel or several conditional kernels in turn. The samples accepted after a burn-in period are included into the particle collection for the next propagation step requiring neither weight assignments nor a resampling step in the manner of the regular particle filter. However, one should note that any of the MCMC moves that involve proposing the sequence $x_{1:k-1}$ (or $x_{0:k-1}$ if involves a random initialisation), i.e. the discretely approximated ancestor states, are in some sense equivalent to a resampling operation. Thus, some degree of path/time degeneracy in the SMCMC method should be expected, as for regular particle filtering. To the best of my knowledge there is no theory that proves which method would be less degenerate. However, I would postulate that the wide range of MCMC moves available in a single SMCMC scheme may improve on path degeneracy in the SMCMC case compared with particle filtering, although a full exploration of is a topic for future work.

Both generic algorithms of particle filter and SMCMC are shown in **Algorithm 2** and **3** respectively. SMCMC may be favourable in the case of high-dimensional latent variables, as in practice the proposal $q(x_k | x_{0:k-1}) q(x_{0:k-1})$ will be split up into a number of blockwise Gibbs steps and Metropolis-within-Gibbs (MwG) moves. This allows us to adopt suitable MCMC schemes based on domain knowledge, which is likely to lead to better convergence [56]. In later sections, I will show how to design both joint MH samplers and MwG samplers to give good inference performance with the SMCMC algorithm.

Algorithm 2 A generic particle filter (PF) algorithm

Inputs: A set of (observed) events $\{s_k\}_{k=1}^K$ in \mathcal{T} .

Outputs: A collection of weighted particles $\{x_{0:K}^p\}_{p=1}^{N_p}$ with normalised weights $\{w_K^p\}_{p=1}^{N_p}$

- 1: **Initialisation:** ($k = 0$) Sample N_p particles $\{x_0^p\}_{p=1}^{N_p}$ from the prior $p(x_0)$ and assign uniform weights $w_0^p = 1/N_p$.
 - 2: **for** $k = 1 : K$ **do**
 - 3: **for** particle $p = 1 : N_p$ **do**
 - 4: Sample $x_k^p \sim q(x_k | x_{0:k-1}^p)$
 - 5: Compute weight $\tilde{w}_k^p = w_{k-1}^p \times \frac{p(x_k^p | x_{0:k-1}^p)}{q(x_k^p | x_{0:k-1}^p)}$
 - 6: **end for**
 - 7: Normalise weights: $w_k^p = \tilde{w}_k^p / \sum_{p'=1}^{N_p} \tilde{w}_k^{p'}$
 - 8: **Resample if necessary**
 - 9: **end for**
 - 10: **Return:** $\{x_{0:K}^p\}_{p=1}^{N_p}, \{w_K^p\}_{p=1}^{N_p}$
-

Algorithm 3 A generic SMC MC algorithm

Inputs: A set of (observed) events $\{s_k\}_{k=1}^K$ in \mathcal{T} .

Outputs: A collection of (unweighted) particles Ω .

- 1: **Initialisation:** ($k = 0$) Sample N_p particles $\{x_0^p\}_{p=1}^{N_p}$ from the prior $p(x_0)$ to form the initial particle collection Ω_0 .
 - 2: **for** $k = 1 : K$ **do**
 - 3: $\Omega_k = \emptyset$
 - 4: **for** iteration $p = 1 : (N_p + N_{\text{burn}})$ **do**
 - 5: Sample $x_{0:k}^* \sim q(x_k | x_{0:k-1}) q(x_{0:k-1})$
 - 6: **if** $p = 1$ **then**
 - 7: $x_{0:k}^p = x_{0:k}^*$ \triangleright Accept the initial condition
 - 8: **else**
 - 9: Compute $\rho = \min\left\{1, \frac{p(x_{0:k}^*)q(x_k^{p-1} | x_{0:k-1}^{p-1})q(x_{0:k-1}^{p-1})}{p(x_{0:k}^{p-1})q(x_k^* | x_{0:k-1}^*)q(x_{0:k-1}^*)}\right\}$
 - 10: Draw $u \sim \text{Uniform}(0, 1)$
 - 11: **if** $u < \rho$ **then** \triangleright MH accept-reject
 - 12: $x_{0:k}^p = x_{0:k}^*$
 - 13: **else**
 - 14: $x_{0:k}^p = x_{0:k}^{p-1}$
 - 15: **end if**
 - 16: **end if**
 - 17: **if** $p > N_{\text{burn}}$ **then**
 - 18: $\Omega_k \leftarrow \Omega_k \cup x_k^p$
 - 19: **end if**
 - 20: **end for**
 - 21: **end for**
 - 22: **Return:** $\Omega = \Omega_0 \cup \Omega_1 \cup \dots \cup \Omega_K$
-

Algorithm 4 SMC MC algorithm for sequential intensity inference

Inputs: A set of events $\{s_k\}_{k=1}^K$ in \mathcal{T} .

Outputs: Posterior filtering samples of underlying intensity.

```

1: Initialisation: ( $k = 0$ ) Create a particle collection  $\Omega_0$  of  $N_p$  particles from the prior.
2: for batch  $\mathcal{T}_k = \mathcal{T}_1 : \mathcal{T}_K$  do
3:   Initialise a new (empty) particle collection  $\Omega_k = \emptyset$ 
4:   for iteration  $p = 1 : (N_p + N_{\text{burn}})$  do
5:     if  $p = 1$  then ▷ Initial condition
6:       Draw a sample  $x_{k-1}^*$  discretely from collection  $\Omega_{k-1}$ 
7:       Propose No. thinned points  $\widetilde{M}^* \sim \text{Poisson}\{\lambda^*|\mathcal{T}_k|\}$ 
8:       Propose positions of thinned points  $\{\tilde{s}_m^*\}_{m=1}^{\widetilde{M}^*} \sim \widetilde{M}^* \times \text{Uniform}(\mathcal{T}_k)$ 
9:       Propose  $\{\mathbf{g}\}_{\mathcal{T}_k}^*$  from  $\mathcal{LD}$  prior (2.8) conditioned on  $x_{k-1}^*$ 
10:       $x_k^p = \{s, \mathbf{g}, i\}_{\mathcal{T}_k}^*$ 
11:    else
12:       $u \sim \text{Uniform}(0, 1)$ 
13:      if  $u < P_J$  then ▷ A joint proposal
14:        Draw a sample  $x_{k-1}^*$  discretely from collection  $\Omega_{k-1}$ 
15:        Propose No. thinned points  $\widetilde{M}^* \sim \text{Poisson}\{\lambda^*|\mathcal{T}_k|\}$ 
16:        Propose positions of thinned points  $\{\tilde{s}_m^*\}_{m=1}^{\widetilde{M}^*} \sim \widetilde{M}^* \times \text{Uniform}(\mathcal{T}_k)$ 
17:        Propose  $\{\mathbf{g}\}_{\mathcal{T}_k}^*$  from  $\mathcal{LD}$  prior (2.8) conditioned on  $x_{k-1}^*$ 
18:         $x_k^* = \{s, \mathbf{g}, i\}_{\mathcal{T}_k}^*$ 
19:        Compute MH acceptance probability  $\rho_J$  from Eq. (2.21)
20:        if  $\text{Uniform}(0, 1) < \rho_J$  then
21:           $x_k^p = x_k^*$  ▷ Accept proposed variables
22:        else
23:           $x_k^p = x_k^{p-1}$  ▷ Reject proposed variables
24:        end if
25:      else ▷ Metropolis-within-Gibbs
26:        Perform MwG refinement moves
27:      end if
28:    end if
29:    if  $p > N_{\text{burn}}$  then
30:       $\Omega_k \leftarrow \Omega_k \cup x_k^p$  ▷ Include the converged sample into particle collection
31:    end if
32:  end for
33: end for
34: Map all posterior state vector samples to intensity  $\lambda^p(s_k)$  with Eq. (2.14)
35: Return: Intensity samples  $\{\lambda^p(s_k)\}_{p=1}^{N_p}$  at each input event  $s_k$ 

```

2.4.2 Joint proposal of latent variables

The joint MH kernel is firstly introduced here. It provides fast proposals of the ‘new’ latent variables x_k in each interval \mathcal{T}_k and consists of a discrete uniform draw of the converged sample $x_{1:k-1}$ from the ‘past’ particle collection obtained from the previous step at the end of interval \mathcal{T}_{k-1} , followed by proposals of x_k conditioned on the sampled particle

$x_{1:k-1}^p$.

More specifically, this latter proposal step is split into three sampling sub-steps, applied in sequence: 1) the total number of thinned events \widetilde{M} in \mathcal{T}_k sampled from a Poisson distribution; 2) the locations of thinned events $\{\tilde{s}_m\}_{m=1}^{\widetilde{M}}$ sampled uniformly within \mathcal{T}_k ; and 3) the state vectors $\{\mathbf{g}\}_{\mathcal{T}_k}$ of all events (both observed and latent) in \mathcal{T}_k sampled from the \mathcal{LD} prior conditioned on the events' locations and the sampled particle $x_{1:k-1}^p$. This gives an overall proposal density as:

$$q_J(x_k) = \frac{\text{Poisson}(\widetilde{M} | \lambda^*, \mathcal{T}_k)}{|\mathcal{T}_k|^{\widetilde{M}}} \mathcal{LD}(\{\mathbf{g}\}_{\mathcal{T}_k} | \{\mathbf{g}\}_{\mathcal{T}_{k-1}}, \{s\}_{\mathcal{T}_k}) \quad (2.20)$$

Note that since the thinned events and input events jointly contribute to form the prior homogeneous Poisson process, there is no tractable prior distribution for the number of thinned events \widetilde{M} . However, it can still be sampled from the Poisson distribution with the upper-bound intensity λ^* (or an arbitrary discounted intensity) as the MH acceptance probability will adjust for the values of \widetilde{M} proposed. Incorporating Eq. (2.16), one can write down the MH acceptance probability for joint latent variable x_k at the p th MCMC iteration:

$$\rho_J = \min \left\{ 1, \frac{(\lambda^*)^{N_k^*} \mathcal{LD}(\{\mathbf{g}\}_{\mathcal{T}_k}^* | \{\mathbf{g}\}_{\mathcal{T}_{k-1}}^*, \{s\}_{\mathcal{T}_k}^*) \prod_n \sigma\{(-1)^{i_n^*} g_{1,n}^*\} q_J(x_k^{p-1})}{(\lambda^*)^{N_k^{p-1}} \mathcal{LD}(\{\mathbf{g}\}_{\mathcal{T}_k}^{p-1} | \{\mathbf{g}\}_{\mathcal{T}_{k-1}}^{p-1}, \{s\}_{\mathcal{T}_k}^{p-1}) \prod_n \sigma\{(-1)^{i_n^{p-1}} g_{1,n}^{p-1}\} q_J(x_k^*)} \right\} \quad (2.21)$$

where the superscript '*' indicates the samples proposed in the current iteration and 'p-1' indicates the previous iteration of the MCMC. Note that the nature (i.e. latent or observed) of the events are known a priori, hence the indicators $\{i\}_{\mathcal{T}_k}$ of the events in \mathcal{T}_k are assigned deterministically with values of either 0 or 1.

As is common practice, it is suggested to take N_{burn} iterations before including any MCMC output into the new particle set, in order to neglect non-converged MCMC samples. **Algorithm 4** outlines the pseudo-code of the general scheme for performing SMCMC inference with the joint proposal. Tuning of the proposal intensity and alternative proposals incorporating domain knowledge could improve the convergence rate and inference performance, although this is not investigated here.

2.4.3 Metropolis-within-Gibbs refinement

A second step in the MCMC procedure involves refinement moves using MwG. As the dimension and number of latent variables in the batch increases, the joint proposal described in the previous section can sometimes have extremely low acceptance rates and consequently result in low particle diversity, which is analogous to the weight degeneracy problem encountered in importance sampling particle filters [18], although I stress that the SMCMC procedure operates entirely without particle weights. Therefore in this section,

I propose the use of a MwG refinement step in conjunction with the joint MH kernel, as indicated in **Algorithm 4**.

The refinement moves are designed to consist of: a reversible-jump step for adding or removing thinned events (positions and state vectors); a MwG step for refining the positions of the thinned events; and finally a MwG step for moving the state vectors $\{\mathbf{g}\}_{\tau_k}$ using a Metropolis-adjusted-Langevin-algorithm (MALA) procedure (see [55] and references therein). While these three Gibbs sampling steps are likely to make smaller moves than those of the joint MH sampler, they are also able to achieve more local exploration of the latent sample space through higher acceptance probabilities. A similar MwG construction was adopted in the non-sequential SGCP model of [47]. These three sub-steps are detailed below.

Reversible-jump MCMC for \widetilde{M}

The value of \widetilde{M} determines the number of thinned event locations and state vectors that need to be proposed in the other two MwG samplers. Therefore, the reversible-jump MCMC [57] algorithm is adopted to navigate the variable dimension of the sample space.

The sampler first makes a Bernoulli decision on whether to insert or delete a latent event. An insertion proposal q_{ins} consists of a uniform proposal of the event location \tilde{s}' in \mathcal{T}_k , followed by a draw of its corresponding state vector $\mathbf{g}(\tilde{s}')$ from the \mathcal{LD} prior conditioned on the state vectors of the two events immediately preceding and following \tilde{s}' , whilst a deletion proposal q_{del} simply consists of a uniform random selection and removal of an existing latent event, say \tilde{s}_m , out from a total of \widetilde{M} events. Thus, the proposal densities are as follows

$$q_{\text{ins}}(\widetilde{M}+1 \leftarrow \widetilde{M}) = \frac{P_B}{|\mathcal{T}_k|} \mathcal{LD}(\mathbf{g}(\tilde{s}') | \tilde{s}', \{\mathbf{g}\}_{\tau_k}) \quad (2.22)$$

$$q_{\text{del}}(\widetilde{M}-1 \leftarrow \widetilde{M}) = \frac{1 - P_B}{\widetilde{M}} \quad (2.23)$$

where P_B is the Bernoulli probability of making an insertion move which is set to 0.5 throughout this chapter. Incorporating the joint recursion in (2.15) and (2.16), the acceptance ratios for both moves are:

$$\rho_{\text{ins}} = \min\left\{1, \frac{(1 - P_B) |\mathcal{T}_k| \lambda^*}{P_B (\widetilde{M} + 1) (1 + \exp\{g_1(\tilde{s}')\})}\right\} \quad (2.24)$$

$$\rho_{\text{del}} = \min\left\{1, \frac{P_B \widetilde{M} (1 + \exp\{g_1(\tilde{s}_m)\})}{(1 - P_B) |\mathcal{T}_k| \lambda^*}\right\} \quad (2.25)$$

Algorithm 5 shows the pseudo-code for performing one iteration of the reversible-jump move. It is found advisable to perform several iterations of this MH kernel before pro-

Algorithm 5 Single-iteration reversible-jump MCMC for \widetilde{M}

Inputs: Event positions $\{s\}_{\mathcal{T}_k}$ and state vectors $\{\mathbf{g}\}_{\mathcal{T}_k}$ in \mathcal{T}_k ; the number of thinned events \widetilde{M}

Outputs: Updated number of thinned events \widetilde{M} (and corresponding event positions and state vectors).

```

1: Draw  $u \sim \text{Uniform}(0, 1)$ 
2: if  $u < P_B$  then ▷ Insertion
3:   Draw  $\tilde{s}' \sim \text{Uniform}(\mathcal{T}_k)$ 
4:   Draw  $\mathbf{g}(\tilde{s}') \sim \mathcal{LD}(\mathbf{g}(\tilde{s}') | \tilde{s}', \{\mathbf{g}\}_{\mathcal{T}_k})$ 
5:   Compute  $\rho_{\text{ins}}$  from Eq. (2.24)
6:   if  $\text{Uniform}(0, 1) < \rho_{\text{ins}}$  then
7:     Accept  $\tilde{s}'$  and  $\mathbf{g}(\tilde{s}')$  as a new thinned event
8:      $\widetilde{M} = \widetilde{M} + 1$ 
9:   end if
10: else ▷ Deletion
11:   Draw  $\tilde{s}_m$  discretely uniformly from  $\{m = 1, 2, \dots, \widetilde{M}\}$ 
12:   Compute  $\rho_{\text{del}}$  from Eq. (2.25)
13:   if  $\text{Uniform}(0, 1) < \rho_{\text{del}}$  then
14:     Remove  $\tilde{s}_m$  and  $\mathbf{g}(\tilde{s}_m)$  from the thinned events
15:      $\widetilde{M} = \widetilde{M} - 1$ 
16:   end if
17: end if
18: Return:  $\widetilde{M}$ 

```

ceeding to the other two samplers.

Metropolis-Hastings for $\{\tilde{s}_m\}_{m=1}^{\widetilde{M}}$

Conditioned on the total number of thinned events \widetilde{M} , the posterior thinned event locations are sampled from a standard MH kernel. For each thinned event \tilde{s}_m , a new location \tilde{s}'_m is proposed from a pre-assigned conditional transition kernel $q_{\text{loc}}(\tilde{s}'_m \leftarrow \tilde{s}_m)$ followed by a draw of the new state vector $\mathbf{g}(\tilde{s}'_m)$ at time \tilde{s}'_m from the conditional Langevin prior:

$$\mathcal{LD}\left(\mathbf{g}(\tilde{s}'_m) \mid \tilde{s}'_m, \{\mathbf{g}\}_{\mathcal{T}_k \setminus \mathbf{g}(\tilde{s}_m)}\right) \quad (2.26)$$

where $\{\mathbf{g}\}_{\mathcal{T}_k \setminus \mathbf{g}(\tilde{s}_m)}$ stands for all state vectors in the batch \mathcal{T}_k except for the one at \tilde{s}_m . The acceptance probability can then be described as:

$$\rho_{\text{loc}} = \min\left\{1, \frac{q_{\text{loc}}(\tilde{s}_m \leftarrow \tilde{s}'_m)(1 + \exp\{g_1(\tilde{s}_m)\})}{q_{\text{loc}}(\tilde{s}'_m \leftarrow \tilde{s}_m)(1 + \exp\{g_1(\tilde{s}'_m)\})}\right\} \quad (2.27)$$

In the case where q_{loc} is symmetric, the acceptance probability is further reduced to the ratio of two sigmoidal thinning probabilities.

Metropolis-adjusted-Langevin-algorithm (MALA) for state vectors

Conditioned on the number and locations of the thinned events, the posterior state vectors of all events within the batch can now be sampled. The exploration of the state vectors takes place in a multi-dimensional continuous space and hence requires a well-tuned sampling method to ensure fast convergence. Based on the conditional propagation equation in Eq. (2.16), one can write the *Log*-posterior of the state vector subject to an additive constant (normalising constant):

$$\begin{aligned} & \mathcal{L}\left(\{\mathbf{g}\}_{\mathcal{T}_k} \mid x_{k-1}, \{s\}_{\mathcal{T}_k}, \{i\}_{\mathcal{T}_k}, \lambda^*, \mathcal{T}_k\right) \\ &= \ln\left\{\mathcal{LD}(\{\mathbf{g}\}_{\mathcal{T}_k} \mid \{\mathbf{g}\}_{\mathcal{T}_{k-1}}, \{s\}_{\mathcal{T}_k})\right\} - \sum_{n=1}^{N_k} \ln[1 + (-1)^{i_n} \exp\{g_{1,n}\}] + \text{Const.} \end{aligned} \quad (2.28)$$

As Eq. (2.8) shows the conditional progression of the state vectors, one can concatenate $\{\mathbf{g}\}_{\mathcal{T}_k}$ into a $2N_k$ -dimensional multivariate Gaussian vector \mathbf{G}_k :

$$p(\{\mathbf{g}\}_{\mathcal{T}_k} \mid \{\mathbf{g}\}_{\mathcal{T}_{k-1}}, \{s\}_{\mathcal{T}_k}) = \mathcal{N}(\mathbf{G}_k \mid \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}) \quad (2.29)$$

which essentially allows further simplification over the *Log*-posterior:

$$\begin{aligned} & \mathcal{L}\left(\{\mathbf{g}\}_{\mathcal{T}_k} \mid x_{k-1}, \{s\}_{\mathcal{T}_k}, \{i\}_{\mathcal{T}_k}, \lambda^*, \mathcal{T}_k\right) \\ &= -\frac{1}{2}(\mathbf{G}_k - \hat{\boldsymbol{\mu}})^T \hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{G}_k - \hat{\boldsymbol{\mu}}) - \sum_{n=1}^{N_k} \ln[1 + \exp\{(-1)^{i_n} g_{1,n}\}] + \text{Const.} \end{aligned} \quad (2.30)$$

Taking advantage of the *Log*-gradient information, the MALA is used to accelerate the convergence by proposing from a gradient-adjusted transition kernel:

$$q(\mathbf{G}_k^* \mid \mathbf{G}_k^{p-1}) = \mathcal{N}\left(\mathbf{G}_k^* \mid \mathbf{G}_k^{p-1} + \frac{\epsilon^2}{2} \nabla \log \tilde{\pi}(\mathbf{G}_k^{p-1}), \epsilon^2 \Sigma_c\right) \quad (2.31)$$

where $\nabla \log \tilde{\pi}(\cdot)$ is the gradient of Eq. (2.30), ϵ is the integration step size and Σ_c is a pre-defined (constant) covariance matrix. MALA is then completed with a standard accept/reject step with acceptance probability:

$$\rho_{\text{MALA}} = \min\left\{1, \frac{\tilde{\pi}(\mathbf{G}_k^*) q(\mathbf{G}_k^{p-1} \mid \mathbf{G}_k^*)}{\tilde{\pi}(\mathbf{G}_k^{p-1}) q(\mathbf{G}_k^* \mid \mathbf{G}_k^{p-1})}\right\} \quad (2.32)$$

Additional, the gradient calculation and the MALA diffusion is carried out over the ‘whitened’ space of the variable \mathbf{G}_k . This is achieved by applying Cholesky decomposition on the precision matrix $\hat{\boldsymbol{\Sigma}}^{-1} = \mathbf{L}\mathbf{L}^T$ and rewrite the *Log*-posterior of Eq. (2.30)

Algorithm 6 Rejection Sampling

Inputs: target density $f(x)$, proposal density $q(x)$, bound B

Outputs: X as a sample from $f(x)$

```
1: flag = False
2: while not flag do
3:   Draw  $X \sim q(x)$ 
4:   Compute  $\rho(X) = \frac{f(X)}{B \times q(X)}$ 
5:   Draw  $u \sim \text{Uniform}(0, 1)$ 
6:   if  $u < \rho(X)$  then
7:     Accept sample  $X$ 
8:     flag = True
9:   end if
10: end while
Return:  $X$ 
```

as:

$$\begin{aligned} & \mathcal{L}\left(\{\mathbf{g}\}_{\mathcal{T}_k} \mid x_{k-1}, \{s\}_{\mathcal{T}_k}, \{i\}_{\mathcal{T}_k}, \lambda^*, \mathcal{T}_k\right) \\ &= -\frac{1}{2}(\mathbf{G}_k - \hat{\boldsymbol{\mu}})^T \mathbf{L} \mathbf{L}^T (\mathbf{G}_k - \hat{\boldsymbol{\mu}}) - \sum_{n=1}^{N_k} \ln[1 + \exp\{(-1)^{i_n} g_{1,n}\}] + \text{Const.} \\ &= -\frac{1}{2}(\mathbf{G}_k^w - \hat{\boldsymbol{\mu}}^w)^T (\mathbf{G}_k^w - \hat{\boldsymbol{\mu}}^w) - \sum_{n=1}^{N_k} \ln[1 + \exp\{(-1)^{i_n} [\mathbf{L}^{-T} \mathbf{G}_k^w]_{2n-1}\}] + \text{Const.} \end{aligned} \quad (2.33)$$

where $\mathbf{G}_k^w = \mathbf{L}^T \mathbf{G}_k$, $\hat{\boldsymbol{\mu}}^w = \mathbf{L}^T \hat{\boldsymbol{\mu}}$ and the subscript $2n-1$ represents the $(2n-1)$ th element of the $2N_k$ -dimensional vector. This allows us to carry out the same MH routine of MALA on the whitened variable \mathbf{G}_k^w instead of \mathbf{G}_k , which gives a better-conditioned covariance matrix (\mathbf{I}) and fastens the convergence.

Another possible improvement that can be done on MALA is considering a Langevin diffusion on a Riemannian manifold [58] so that the pre-defined matrix Σ and step size ϵ take account of the local curvature (e.g. Hessian) of the target density to speed up the convergence of the Markov chain. Such approach generally demands more intensive computations, more details can be found in [55].

2.4.4 Refinement with rejection sampling (RS)

Recall the close relation between *thinning* and RS as mentioned in Section 2.1.2, it is in fact possible to derive the thinning (or retaining) probability for the NHPP from the RS perspective (see **Appendix 2.C**). Therefore, the possibility of using RS as an alternative refinement approach is explored here.

RS is a technique used to generate samples from distributions that cannot be sampled directly. Denoting the target density as $f(x)$ and the proposal density where samples can

be readily drawn as $q(x)$, the rejection sampling compute the acceptance probability $\rho(x)$ as:

$$\rho(x) = \frac{f(x)}{B \times q(x)} \quad (2.34)$$

where $1 \leq B < \infty$ is a finite bound over the probability ratio $f(x)/q(x)$, i.e. $f(x) \leq Bq(x), \forall x$. Therefore RS is usually more restrictive to use than other MCMC methods as it requires the calculation of a tractable bound B . However, RS does not require any burn-in for convergence as the generated samples are guaranteed to come from the target distribution. **Algorithm 6** shows the standard RS procedure to generate one sample from the target distribution, the general structure of which is used repeatedly in the refinement step below.

To apply RS in the refinement step, I again use three separate rejection samplers in a Gibbs manner as in MwG: (i) sampling the number thinned events \widetilde{M} ; (ii) sampling the location of the thinned events $\{\tilde{s}_m\}_{m=1}^{\widetilde{M}}$; and (iii) sampling the state vectors of all events in the batch $\{\mathbf{g}\}_{T_k}$.

(i): Denoting the number of observed events in the batch \mathcal{T}_k as \hat{K} , these together with the \widetilde{M} thinned events should constitute a HPP with counting measure $N(\mathcal{T}_k) \sim \text{Poisson}(\lambda^*|\mathcal{T}_k|)$. The target density can thus be written as:

$$f(\widetilde{M}) = \text{Poisson}\left((\widetilde{M} + \hat{K}) \mid \lambda^*|\mathcal{T}_k|\right) \quad (2.35)$$

With a simple proposal of \widetilde{M} from a discretely uniform distribution with $G \in \mathbb{N}^+$ bins i.e. $\{0, 1, 2, \dots, G-1\}$, the bound $B(\widetilde{M})$ and corresponding acceptance ratio $\rho(\widetilde{M})$ are:

$$B(\widetilde{M}) = \frac{(\lambda^*|\mathcal{T}_k|)^{\lfloor \lambda^*|\mathcal{T}_k| \rfloor} \exp\{-\lambda^*|\mathcal{T}_k|\} G}{(\lfloor \lambda^*|\mathcal{T}_k| \rfloor)!} \quad (2.36)$$

$$\rho(\widetilde{M}) = \frac{\lfloor \lambda^*|\mathcal{T}_k| \rfloor! \times (\lambda^*|\mathcal{T}_k|)^{\widetilde{M} + \hat{K} - \lfloor \lambda^*|\mathcal{T}_k| \rfloor}}{(\widetilde{M} + \hat{K})!} \quad (2.37)$$

Note that unlike the corresponding MH sampler in MwG, the described rejection sampler does not propose locations and state vectors of the \widetilde{M} thinned events.

(ii): As it is impossible to ‘fill’ thinned events into a NHPP to make it homogeneous without knowledge of intensity function (or state vectors), the location \tilde{s}_m and the state vector $\mathbf{g}(\tilde{s}_m)$ are jointly proposed for \widetilde{M} times conditioned on the existing state vectors in the batch (for input points and already proposed latent points).

To this end, one can write the target density and the proposal density respectively as:

$$f(\tilde{s}_m, \mathbf{g}(\tilde{s}_m)) = \frac{1}{|\mathcal{T}|} \times \mathcal{LD}\left\{\mathbf{g}(\tilde{s}_m)|\tilde{s}_m, \{\mathbf{g}\}_{\mathcal{T}_k}\right\} \times \sigma(-g_1(\tilde{s}_m)) \quad (2.38)$$

$$q(\tilde{s}_m, \mathbf{g}(\tilde{s}_m)) = \frac{1}{|\mathcal{T}|} \times \mathcal{LD}\left\{\mathbf{g}(\tilde{s}_m)|\tilde{s}_m, \{\mathbf{g}\}_{\mathcal{T}_k}\right\} \quad (2.39)$$

As the additional term in the target density is always less than 1, the proposal itself is already a finite bound of the target with $B = 1$, giving the acceptance ratio $\rho(\tilde{s}_m, \mathbf{g}(\tilde{s}_m)) = \sigma(-g_1(\tilde{s}_m))$

(iii): With the locations of both input and latent events in the batch fixed, the proposal of state vectors $\{\mathbf{g}\}_{\mathcal{T}_k}$ can simply be the conditional prior of the Langevin dynamics. In this case, a simplified acceptance ratio is obtained as the product of Bernoulli probability of each event, with the bound B equal to unity. Hence, one joint proposal of $\{\mathbf{g}\}_{\mathcal{T}_k}$ will give an acceptance probability:

$$\rho(\{\mathbf{g}\}_{\mathcal{T}_k}) = \prod_{n:s_n \in \mathcal{T}_k} \sigma\{(-1)^{i_n} g_{1,n}\} \quad (2.40)$$

It is worth noting that the product of multiple sigmoid functions could result in an extremely low acceptance ratio which stagnates the algorithm and increases computation. To circumvent this issue, the rejection sampler is applied individually to each state vector conditional on all others e.g. propose from the conditional prior as in Eq. (2.26). Such rejection sampler provides a better acceptance ratio as a single sigmoid function at the cost of reduced mixing among state vectors.

Furthermore, this RS process can be regarded as the reverse of the thinning operation: instead of deciding whether the point should be thinned or retained given intensity, one now tries to find the appropriate intensity (state vector) value given the fact that a point is either latent ($i_n = 1$) or observed ($i_n = 0$).

2.4.5 Sequential batch scheme

Regarding the interval \mathcal{T}_k , the choice of how to delineate the entire domain of interest is largely arbitrary. However, the most intuitive choice that fixes these intervals to correspond to the observed event times, may not yield best algorithmic performance. I propose the use of regular sized batches of duration δT here. With the appropriate choice of δT , the scheme recovers the temporal correlation among points within the same batch and thus tends to improve the sequential inference accuracy when compared to the point-wise propagation scheme.

Moreover, the batch scheme provides the possibility to replace the global maximum

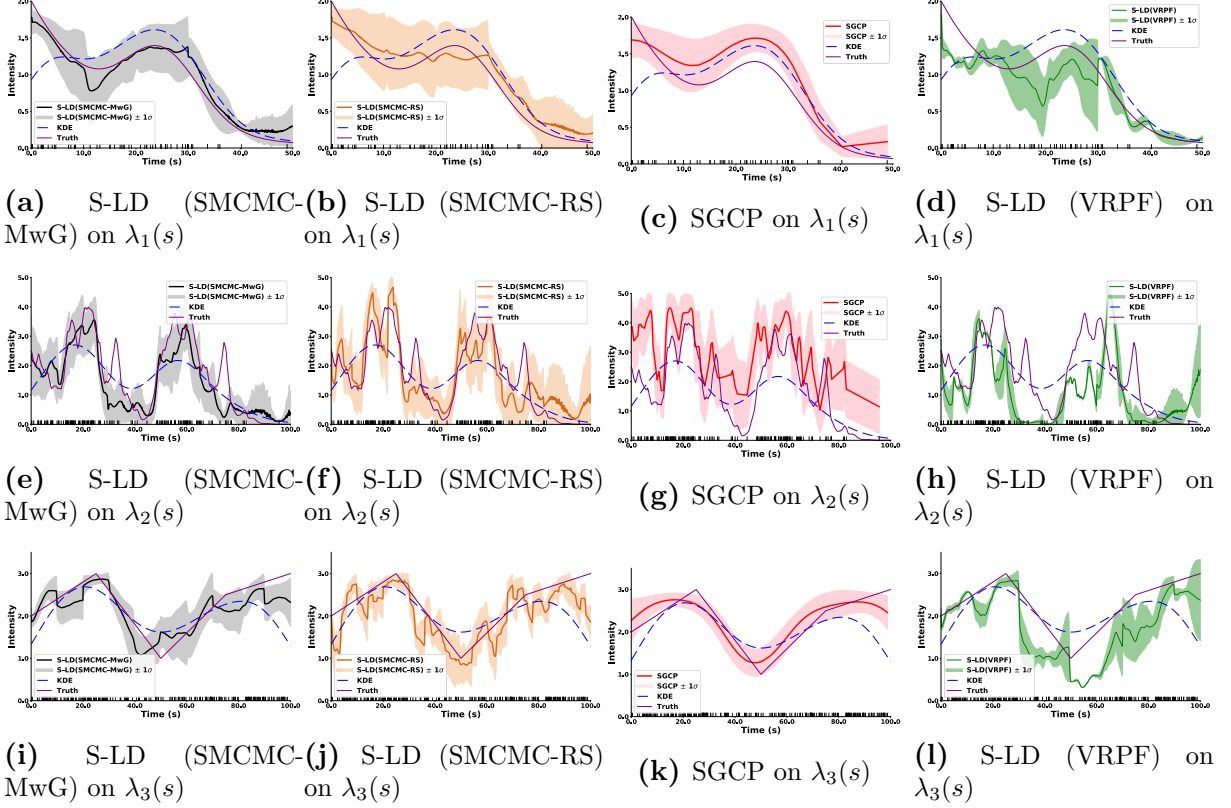


Figure 2.4: The intensity inference results estimated using different methods on three synthetic datasets. True intensity curves and KDE results are displayed in all panels. Three Bayesian approaches are shown individually with corresponding means in solid lines and $\pm 1\sigma$ (68%) confidence intervals as shaded regions. The occurrences of input events are shown as scatter lines on the x -axes

intensity λ^* with maxima λ_k^* that vary with batch number k and these can be updated individually in a Gibbs manner with a Gamma prior specified for each λ_k^* . The Gibbs conditional parameters for the posterior are: $\alpha_{\text{post}} = \alpha_{\text{prior}} + N_k$, $\beta_{\text{post}} = \beta_{\text{prior}} + |\mathcal{T}_k|$. This local maximum intensity considerably reduces the number of latent variables proposed for inference and thus enables computational efficiency.

2.5 Results and discussions

In this section, I present empirical performance analysis of the proposed sequential-Langevin (S-LD) model. Section 2.5.1 assesses the relative performance of the S-LD, the SGCP [47] and a baseline kernel density estimation (KDE) method [32] with synthetic datasets where the ground truth intensity $\lambda(s)$ is available. In the same section, I also demonstrate the time-degeneracy behaviour that exists in applying more standard sequential Monte Carlo algorithms to the proposed model. The S-LD model is then applied to a financial dataset with high frequency input events in Section 2.5.2. Finally, the convergence behaviour of the SMCMC algorithm is examined under different refinement

Table 2.1: Hyperparameters settings for each inference approach on synthetic datasets

	S-LD (SMCMC)	SGCP	S-LD (VRPF)
$\lambda_1(s)$	$\theta = -0.7, \sigma = 0.5, K = 5$ $P_j = 0.5, N_{\text{burn}} = 200, N_p = 200$	$N_{\text{iter}} = 400,$ $N_{\text{burn}} = 200, l_k = 2.0$	$\theta = -0.7, \sigma = 0.5,$ $K = 5, N_p = 800$
$\lambda_2(s)$	$\theta = -0.5, \sigma = 0.8, K = 20$ $P_j = 0.7, N_{\text{burn}} = 200, N_p = 200$	$N_{\text{iter}} = 400,$ $N_{\text{burn}} = 200, l_k = 1.0$	$\theta = -0.5, \sigma = 0.8,$ $K = 20, N_p = 800$
$\lambda_3(s)$	$\theta = -0.2, \sigma = 0.2, K = 10$ $P_j = 0.5, N_{\text{burn}} = 200, N_p = 200$	$N_{\text{iter}} = 400,$ $N_{\text{burn}} = 200, l_k = 15.0$	$\theta = -0.2, \sigma = 0.2,$ $K = 10, N_p = 1500$

Table 2.2: Numerical results for models. **Bold** is the best.

		S-LD (SMCMC-MwG)	S-LD (SMCMC-RS)	KDE	SGCP	S-LD (VRPF)
$\lambda_1(s)$	MSE	0.0257	0.0342	0.129	0.0704	0.187
	$\mathcal{L}(p)$	1.825	-6.379	–	-9.440	-5498
	Time (s)	15.86	36.46	0.01	60.23	14.89
$\lambda_2(s)$	MSE	0.6531	0.6018	0.8599	1.5257	1.7004
	$\mathcal{L}(p)$	-248.1	-233.85	–	-326.6	-9.3×10^{34}
	Time (s)	60.05	490.4	0.05	1326.28	64.25
$\lambda_3(s)$	MSE	0.0986	0.1157	0.2166	0.0637	0.4286
	$\mathcal{L}(p)$	-69.20	-74.54	–	-28.34	-5.93×10^{29}
	Time (s)	100.3	125.3	0.05	522.2	98.38

schemes (MwG or RS); and the effect of hyperparameters on the S-LD model performance is evaluated.

2.5.1 Synthetic Data

Three sets of one-dimensional data are generated based on **Algorithm 1** with the following intensity functions:

1. A sum of an exponential and a Gaussian bump:

$$\lambda_1(s) = 2 \exp\{-s/15\} + \exp\{-((s-25)/10)^2\} \text{ on the interval } [0, 50] \text{ with 55 events.}$$

2. A doubly-stochastic process with $\lambda_2(s)$ governed by Langevin dynamics with parameters $\theta = -0.5, \sigma = 0.5$ and $\lambda^* = 5$ on interval $[0, 100]$ with 156 events.

3. A piece-wise linear intensity function $\lambda_3(s)$ on interval $[0, 100]$ with 230 events

Synthetic datasets similar to 1 and 3 were also used in the original SGCP paper [47]; while 2 is the dataset generated from the matching prior model. Furthermore, the three synthetic intensity functions also test the models' ability in generalising to underlying intensities not drawn from the assumed prior structure of the model.

In these experiments, the intensity functions of the S-LD model are inferred using both the proposed SMCMC algorithm and a batch-based variable-rate particle filter (VRPF). Additionally, the SMCMC algorithm is tested separately using both MwG refinement and RS refinement schemes. The number of particles used in VRPF is tuned to roughly

match the computational cost of SMCMC algorithm. The results are compared with those obtained by the SGCP model using a square-exponential covariance function (with lengthscale l_k) and by the KDE approach with Gaussian smoothing kernel [59]. **Table 2.1** lists the hyperparameter values used by each approach, except for KDE, for all three synthetic cases. The hyperparameters are heuristically tuned to provide representative results for comparison purpose. **Figure 2.4** shows the graphical results of the four approaches and **Table 2.2** quantitatively reports the performance averaged across 10 trials in terms of the computational time, the mean squared error (MSE) to the true intensity function, and a probabilistic metric $\mathcal{L}(p)$. The log-probability $\mathcal{L}(p)$ is computed as follows:

$$\mathcal{L}(p) = \sum_{k=1}^K \log \left\{ \mathcal{N}(\lambda(s_k) \mid \hat{\mu}_k, \hat{\sigma}_k^2) \right\} \quad (2.41)$$

where $\lambda(s_k)$ is the true intensity value at s_k ; $\hat{\mu}_k$ and $\hat{\sigma}_k^2$ are mean and variance empirically approximated by the particles obtained from the inference algorithm. In addition to the mean error, $\mathcal{L}(p)$ also quantifies the uncertainty of different Bayesian approaches.

Inferred with SMCMC, the proposed S-LD model is able to outperform the other two models in both MSE and $\mathcal{L}(p)$ for the first two synthetic datasets while giving satisfactory accuracy for the third dataset. VRPF on the other hand fails to provide good inference for the S-LD model due to particle weight degeneracy as discussed in Section 2.4, which can be seen from the noticeably low $\mathcal{L}(p)$ values. Computationally, the KDE always gives the fastest run-time because of its algorithmic simplicity, but gives no confidence intervals as a frequentist approach. The SGCP model is significantly more computationally expensive and scales poorly with λ^* , which can be observed from the results for datasets 2 and 3. Comparing between MwG and RS refinement, both schemes give similar estimation accuracy whilst MwG refinement requires lower computational cost than RS. It is typically more difficult to maintain a salable computational cost using RS algorithms due to its inherent sampling mechanism. The MwG refinement on S-LD model shows roughly linear computational cost with the number of input events.

Visually from **Figure 2.4**, despite the prior mismatches for $\lambda_1(s)$ and $\lambda_3(s)$ compared with the model used in inference, the S-LD model inferred with both refinements of the SMCMC algorithm still captures the overall shape of the true intensity function and gives a reasonable estimate of uncertainty. The SGCP model on the other hand, was found to be sensitive to the choice of hyperparameters e.g. different forms of covariance functions and values of lengthscale. The KDE method tends to over-smooth the intensity and ignores short-term variations. From the figure, one can again notice the degeneracy in particles for the S-LD(VRPF) as it provides overly narrow confidence intervals.

I also apply the S-LD model (inferred by SMCMC-MwG) to additional 4 realisations of the doubly-stochastic Poisson process with the same parameters used to generate $\lambda_2(s)$.

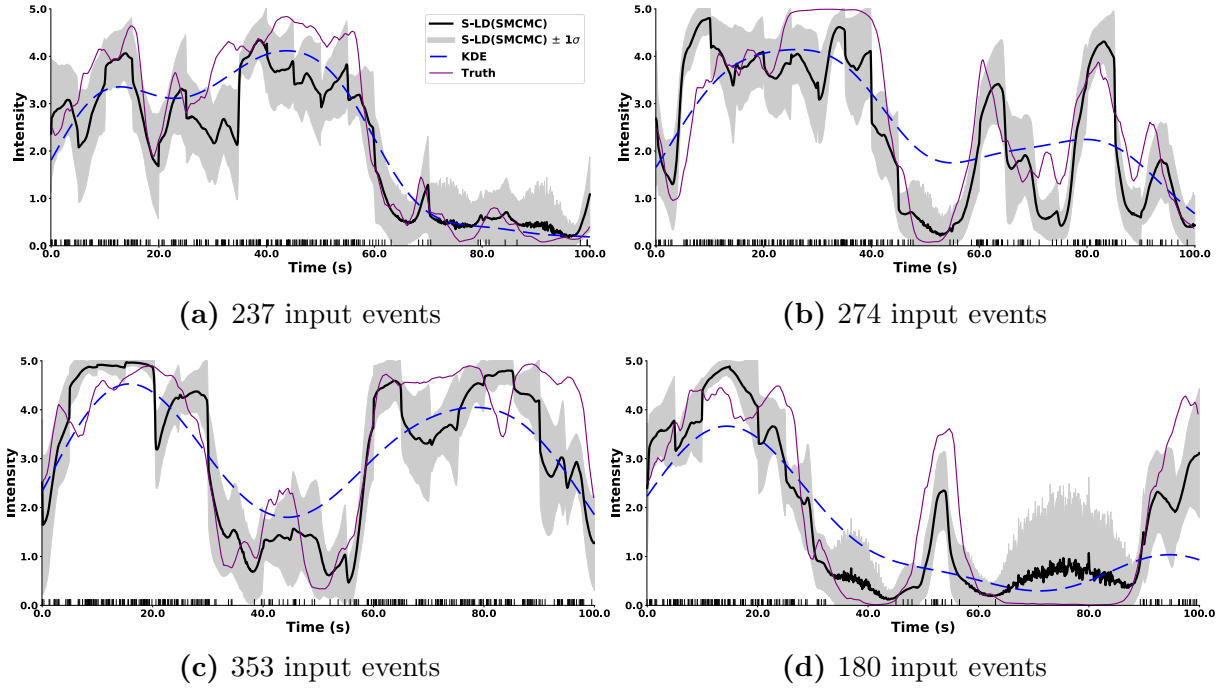


Figure 2.5: Additional tests of the S-LD model on doubly-stochastic Poisson processes. In comparison to KDE, the S-LD model is able to achieve an average MSE of **0.6806** and an average computational time of **0.863s** per input point; while KDE only provides an average MSE of **1.032** with an average computational time of 1.576×10^{-4} s per input point.

Figure 2.5 shows the results obtained using the same set of inference hyperparameters for $\lambda_2(s)$ as listed in **Table 2.1**. The S-LD model is compared to KDE only as the SGCP model was found to be impractically slow. In terms of both accuracy and computational time, the results are consistent with those obtained from earlier three experiments on synthetic datasets.

For the SMCMC algorithm, one can also obtain the empirical online smoothing distribution $\hat{p}(x_{1:k} | \mathcal{T}_{1:k}, \lambda^*)$, which keeps track of the entire trajectory (historic lineage) of each particle [60]. Although such results usually give more continuous estimated intensities than the filtering results shown in **Figure 2.4**, they are known to exhibit some degeneracy in time and give poorer uncertainty estimates for early data points as k increases. **Figure 2.6** depicts five plots of the intensity trajectories for all particles at different times k of the SMCMC algorithm applied on the S-LD model. It is clear that the trajectories prior to the filtering batch are slightly degenerate and hence one can guess that posterior means and uncertainty estimates would not be reliable as the batch inference progresses.

2.5.2 Application to Order Book Data

Limit order books [61] in modern financial markets, record and display order operations performed by market participants all over the world. With momentum strategy as a common technique used by the traders in high-frequency finance [62], being able to infer

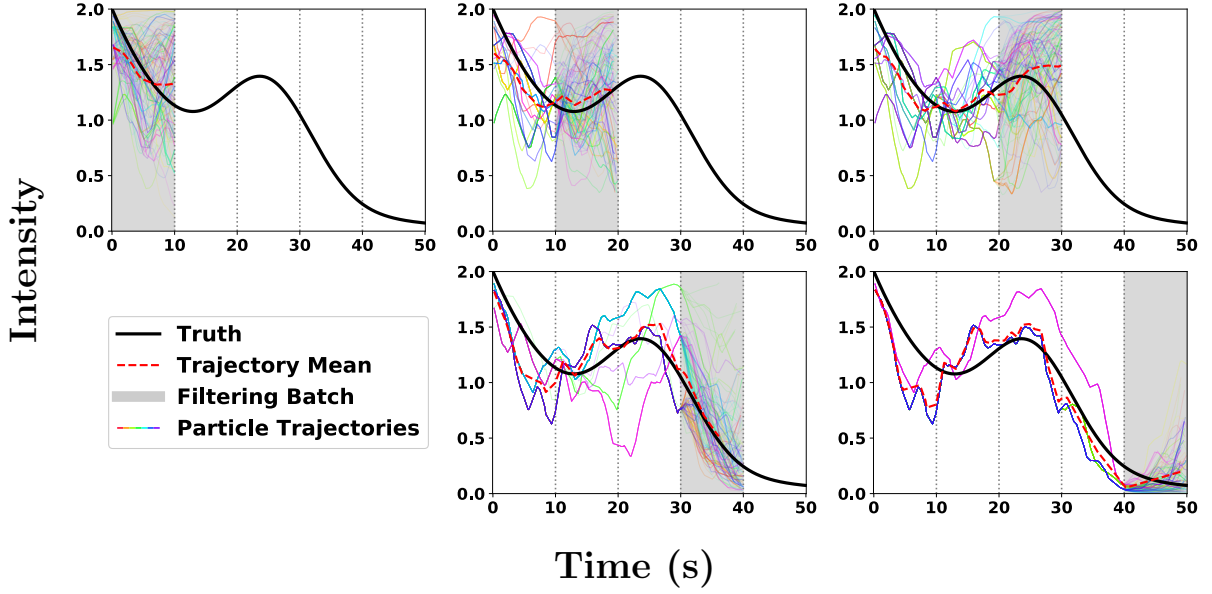


Figure 2.6: Plots show smoothing trajectories of 200 particles in coloured lines. Dashed red lines represent the mean intensity of the trajectories while the solid black lines indicate the true intensity. The variance of the trajectories reduces as the SMC algorithm progresses.

the intensities of limit order arrivals, cancellations and executions provides crucial insights into the future market structure and price trends. This demands the development of a computational-efficient online intensity inference method for market analysis based on time-of arrival trading data.

I apply the S-LD model on a set of LOB data collected from the EUR-USD FOREX market on the 2nd of September 2015. The ticks data used in the experiments is a record of every single limit order arrival at 51 different price levels (‘ticks’) around the mid-price for a duration of 5 minutes (19:35–19:40) from one of the busiest hours of the day.

51 independent S-LD models are constructed for the 51 price levels of interest. A volatile Langevin prior with $\sigma = 1.0$, $\theta = -0.7$ and $\lambda^* = 5.0$ is assigned to all models to accommodate possible drastic changes of the intensity curves in the highly stochastic market. The SMC algorithm with MwG refinement is used for inference with $P_J = 0.7$, $K = 20$, $N_p = 400$ and $N_{\text{burn}} = 400$ to ensure convergence.

Figure 2.7 shows the inference results from the above experiment presented as a 3D surface plot and a heatmap. The surface plot appears to exhibit reasonable behaviours with the mid-price lying in a ‘valley’ formed between the high-intensity ridges and peaks at prices above and below the mid price: this price region close to the mid-price is where the market orders are typically placed and matched (executed) immediately, presenting little interest for limit-order traders. In contrast, the limit order arrivals have high intensity a few *ticksizes* away from the mid-price as these levels are most likely to become the best prices in future price fluctuations. Moreover from the heatmap, one can also observe that the high intensity of bid or ask arrivals exerts pressure on the mid-price to go in the

opposite direction, as would be expected from the market supply-demand relationship.

Figure 2.8 shows in more details the intensity of limit order arrivals at a fixed price of \$1.12960 with the red solid line being bid orders and blue dashed line being ask orders. It is easy to observe a similar distribution of intensities as a function of the distance to the mid-price as described above. Within the 5-minutes period, the mid-price crosses the selected price four times with different responses in the intensity. The first two transitions are almost instantaneous but the start of a small bump in intensity can still be observed prior to the change of bid-ask sides. The arrival of the third transition is progressive and hence the intensity slowly drops as practitioners' interests gradually shift away from the selected price. The final transition is abrupt causing the intensity to spike continuously even after the side changes as the market is very active after experiencing the jump.

Based on the results obtained from this example, it is reasonable to conclude that the proposed S-LD model may be useful for providing predictive inference about market behaviour, although I leave a full investigation of this for future work. The SGCP model was not tested on this dataset due to its impractical computational cost.

2.5.3 Convergence Evaluation

When it comes to an iterative sampling method such as SMC MC, it is important to ensure its convergence while maintaining reasonable computational efficiency. In this section, the convergence behaviours of different SMC MC refinement setups are evaluated by computing their corresponding integrated autocorrelation times (IACFs).

The IACF for a sequence $f(t)$ is defined as:

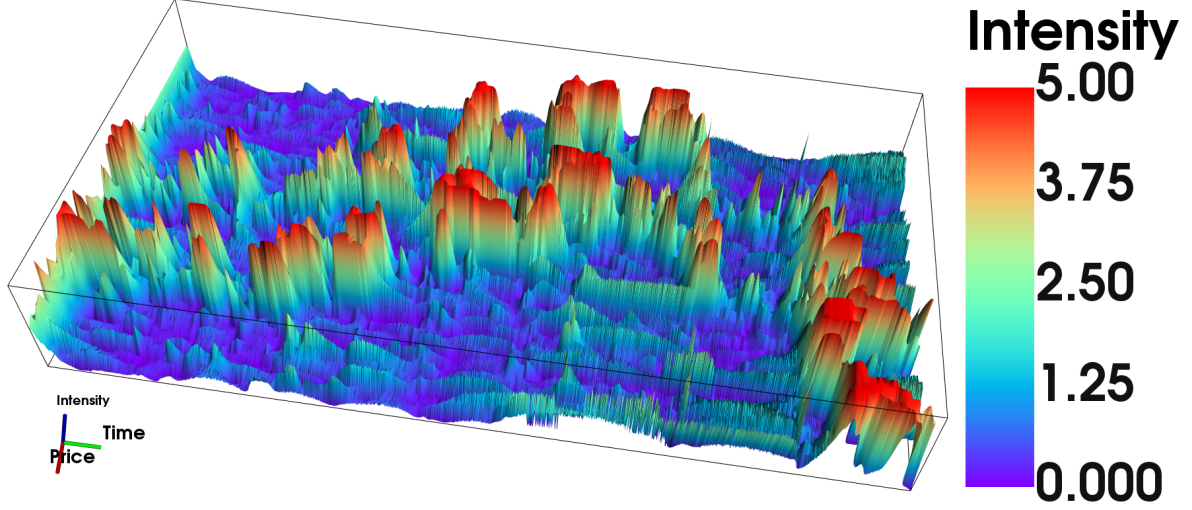
$$\tau_f = \sum_{k=-\infty}^{\infty} \rho_f(k) = 1 + 2 \sum_{k=1}^{\infty} \rho_f(k) \quad (2.42)$$

where $\rho_f(k)$ is the normalised autocorrelation function (ACF):

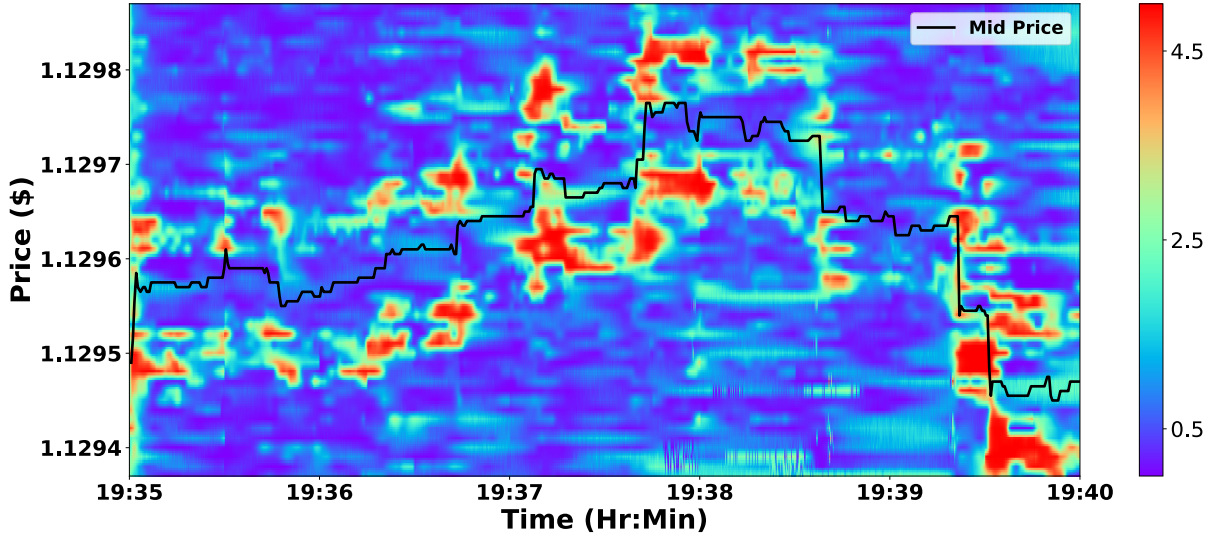
$$\rho_f(k) = \frac{\mathbb{E}[(f(t) - \mu_f)(f(t+k) - \mu_f)]}{\sigma_f^2} \quad (2.43)$$

with μ_f and σ_f^2 being the mean and variance of the sequence. With a sequence of values (e.g. intensities) output from the SMC MC algorithm, τ_f quantifies the factor by which MCMC chain's Monte Carlo error is degraded comparing to standard i.i.d. Monte Carlo from the target posterior distribution. Therefore, the SMC MC refinement setup with better mixing and faster convergence would give a smaller value of τ_f . The IACF is approximated with a finite summation using the method suggested in [63].

Running the SMC MC algorithm for 6000 iterations (i.e. particles + burn-in) on synthetic dataset $\lambda_1(s)$, **Table 2.3** reports the IACFs of root-mean-square errors (RMSE) and intensities under different configurations of the refinement move. Both RS and MCMC



(1) 3D surface plot of the limit order arrival intensities



(2) 2D intensity heatmap

Figure 2.7: Results obtained from parallel runs of S-LD model on limit order arrival data at 51 different price levels. The top figure shows the surface plot of the inferred (filtering) intensity with time and price on x, y axes and intensity on z axis; the bottom figure shows the corresponding heatmap with market mid-price plotted in black solid line as reference

refinement show relatively poor convergence at low P_J due to the lack of resampling of the ‘ancestor’ particle. At high P_J , MCMC suffers from inadequate mixing of Markov chains (i.e. MwG); while RS provides lower IACTs as it is guaranteed to produce representative posterior samples and it only requires convergence in the conditional Gibbs step. An intermediate value of P_J yields the best result for MCMC and outperforms all other configurations.

In summary, the refinement applied in the SMCMC algorithm provide improved mixing of the Markov chains without dramatic increase in computation. Meanwhile, it is also necessary to maintain enough resampling of the ‘ancestor’ particles from the previous

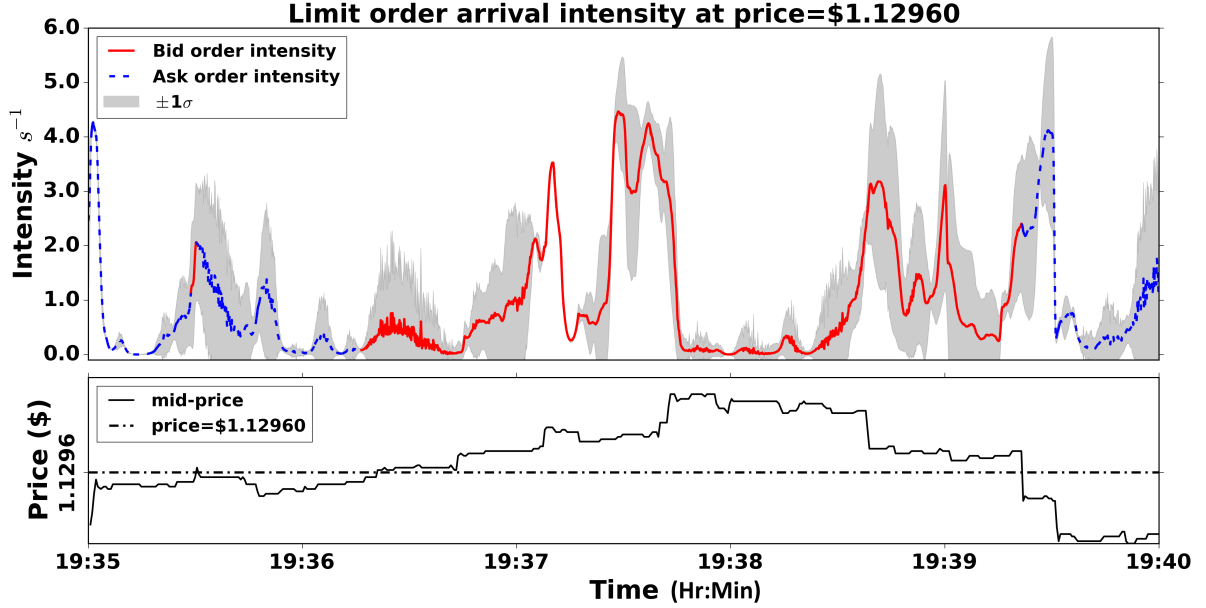


Figure 2.8: Top panel shows the S-LD model intensity inference result on limit order (both bid and ask) arrival data at price \$1.12960. Bottom panel shows the market mid-price for the same duration.

Table 2.3: IACT values computed from RMSE sequence and intensity sequences obtained from 6000 iterations of SMC MC run on $\lambda_1(s)$. Intensity IACT values are averaged across all input points/events.

<i>Refinement method</i>		<i>Joint move ratio</i>		
		$P_J = 0.1$	$P_J = 0.5$	$P_J = 0.9$
MCMC	RMSE IACT	37.23	7.46	18.37
	Intensity avg. IACT	43.91	15.88	29.88
RS	RMSE IACT	49.17	18.92	12.97
	Intensity avg. IACT	50.08	15.24	15.08

batch. The overall low values of IACT computed in this experiment suggest that the proposed SMC MC algorithm requires only a moderate amount of particles and burn-in for inference.

2.5.4 Hyperparameter Settings

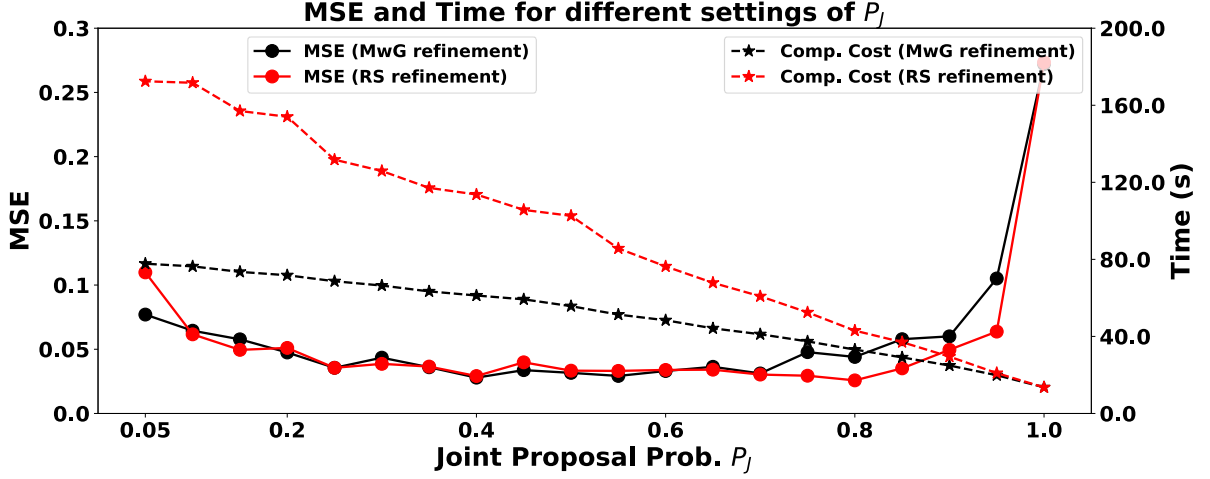
Tuning of hyperparameters is crucial in Bayesian inference and learning. The hyperparameters in the SSM determine loosely the prior dynamics of the state vector diffusion, and hence case-specific domain knowledge should be incorporated to improve the fit of the prior model to the data. Alternatively, SSM hyperparameters can be learned directly from the data with an extension of a variational structure [15] or particle-MCMC methods [64]. In this section however, I present a focused analysis on the algorithm-related hyperparameters.

In **Algorithm 4**, the SMC MC inference routine is controlled by two hyperparameters: joint proposal ratio P_J and batch size \mathcal{T}_k . Both values affect the inference accuracy and computational speed. The S-LD model is run on the same set of NHPP realisations from $\lambda_1(s)$ with the same SSM settings described in **Table 2.1**. Algorithm-wise, MCMC refinement is used with $N_p = 200$ and $N_{\text{burn}} = 800$, which constitute to a total of 1000 iterations to ensure convergence. Each result presented here is averaged across 10 random runs of the SMC MC algorithm.

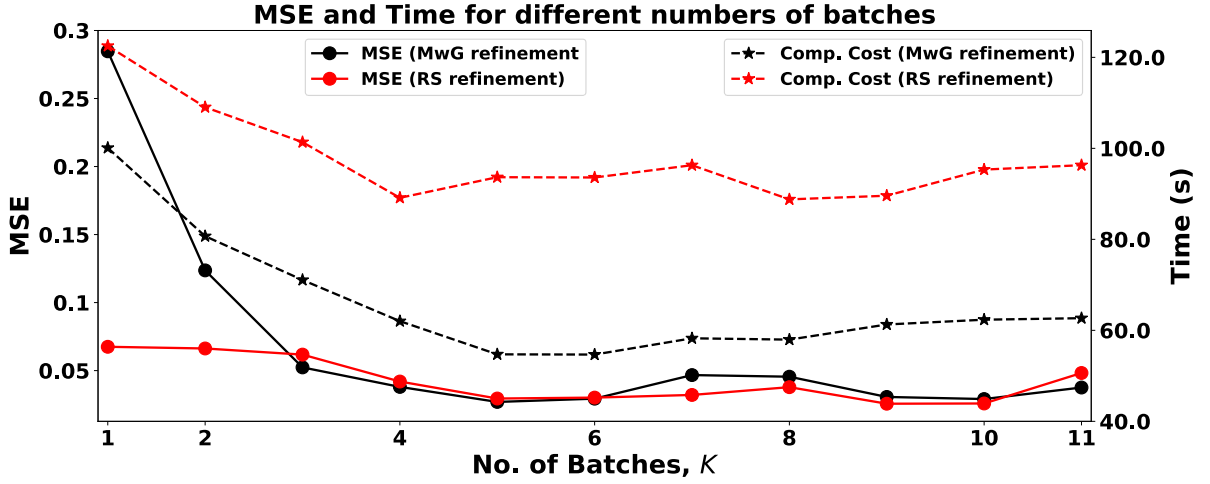
Figure 2.9 shows two plots of computational cost and MSEs under a range of values of P_J and K . The top panel shows that the inference time reduces linearly with the increase of P_J as both MwG and RS refinements require more computation especially in the MALA step and rejection sampling steps. MSE shows weaker correlation with P_J but deteriorates drastically without refinement (i.e. $P_J = 1$), which emphasises the importance of refinement steps in the SMC MC algorithm. At low values of P_J , the MSEs for both refinement schemes rise slightly due to the lack of resampling of ‘ancestor’ particle. Despite the small difference, RS refinement gives lower MSEs at high values of P_J which supports the findings in Section 2.5.3.

With MALA being the computational bottleneck of MwG refinement, the sequential batch scheme changes the complexity from $\mathcal{O}(N^3)$ to roughly $\mathcal{O}(\frac{N^3}{K^3}) \times \mathcal{O}(K)$. This gives obvious drops in computational cost especially upon using the batch scheme (i.e. K changes from 1 to 2) as shown in the bottom plot of **Figure 2.9**. RS refinement’s cost is relatively less sensitive to this change as RS does not involve the $\mathcal{O}(N^3)$ matrix inversion. However, the low acceptance rate of the rejection sampler still renders it slower than MwG. The cost later increases slightly with K , as the $\mathcal{O}(K)$ part becomes more dominant. For MwG refinement, MSE generally improves with an increasing K value because the sequential batch scheme reduces the dimension of latent variables in each batch and hence improves MCMC samplers’ performance. However, the rising trend in MSE that is just observable as K increases becomes more acute at larger K values (not shown on the plot), as the batches tend to de-correlate local location information of the input points. As for RS refinement, the problem of high-dimensional latent space is primarily reflected in the high computational cost (i.e. low acceptance probability in rejection samplers) and hence the small value of K has little influence on its MSE. The de-correlation of input points will also have a negative impact on the MSE for RS refinement at larger K values.

Based on above the analyses, it is fair to conclude that despite higher computational cost, RS refinement is able to provide more robust inference results across a range of algorithmic tuning hyperparameters. A drawback of the current RS refinement scheme is its relative slowness compared to MwG. This could potentially be overcome by using more sophisticated proposals such as adaptive rejection sampling (ARS) [65, 66] and its variants. In practice, one may choose to use a mixture of both RS and MwG refinement moves to achieve the best trade-off between computational time per iteration and convergence over



(1) Varying P_J with fixed $K = 5$



(2) Varying K with fixed $P_J = 0.5$

Figure 2.9: The S-LD model performances under different settings of hyperparameters on synthetic dataset $\lambda_1(s)$

iterations.

2.6 Conclusions and future work

In this chapter, a novel approach of modelling the intensity function of a NHPP with a continuous-time SSM has been presented. In addition to using a generative prior with latent variables to mitigate the inherent intractability of NHPP, I utilised the Markovian property of the SSM and performed sequential Bayesian inference for the intensity function with a novel design of SMCMC algorithm. The proposed algorithm not only deals with the degeneracy problem caused by high-dimensional latent variables, but is also favourable in practical applications that require online intensity estimations such as LOB prediction.

In addition to the basic SMCMC algorithm, a MwG refinement scheme and a sequen-

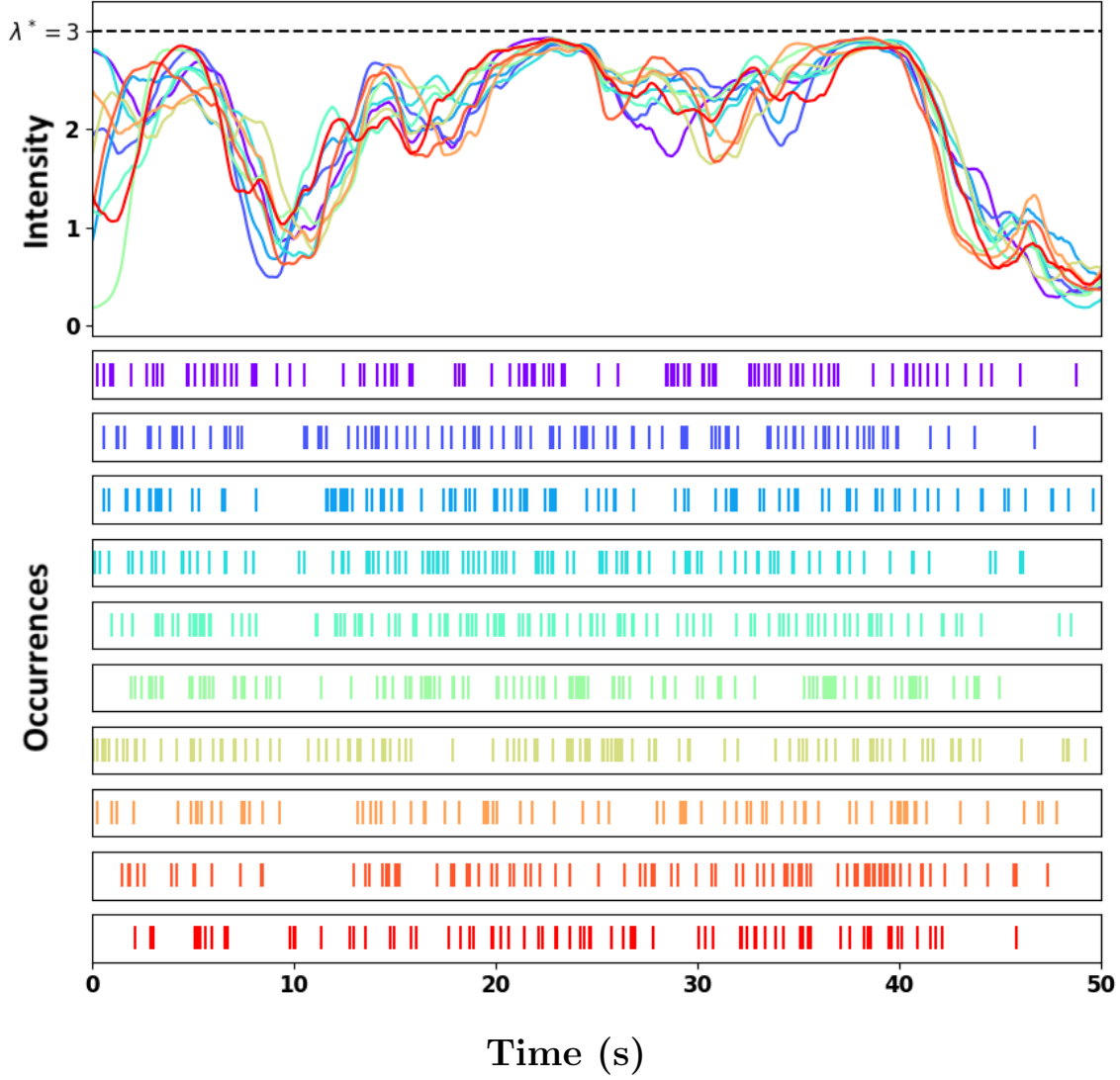


Figure 2.10: Realisations of ten NHPPs (in different colours) with correlated intensities shown in the top panel. The intensity functions are governed by a virtual leader SSM dynamics. Events are shown as scatter lines.

tial batch scheme have been further proposed to improve the inference performance by increasing Markov chain mixing. In comparison with the KDE and SGCP approaches on synthetic datasets, the proposed S-LD model has demonstrated better inference accuracy and reasonable computational cost while maintaining a fully Bayesian framework. Furthermore, the rejection sampling (RS) algorithm has also been studied in this chapter as an alternative to the MH samplers for the refinement step. Despite its “uncontrollable” computational cost, the RS algorithm is still able to provide competitive performance in terms of accuracy to the standard MwG refinement especially for low values of P_J and/or high dimension of the latent variables. This gives the opportunity of trade-off between computational time and algorithm convergence.

The results obtained using FOREX data has demonstrated that the proposed mod-

elling approach is capable of handling real-world challenging intensity inference tasks while giving plausible interpretations of the data. In particular, the inferred intensities (e.g. 3D surface) around mid-price match the typical behaviours of major liquidity providers/market makers who profit from the bid-ask spread. The inherent sequential framework constructed for the S-LD model allows it to be readily integrated with other online price models, such as those presented later in **Chapter 3** and **4**, to provide more predictive power and better model interpretability.

For future work, one possible extension upon the current S-LD model is to add the learning of SSM hyperparameters from the data, including the possibility that the parameters are themselves dynamic, i.e. time-varying. In the proposed framework this would introduce an extra methodological challenge since dynamical (online) parameter learning can be even more demanding than static cases.

Finally, a further promising possibility is to encapsulate multiple correlated point processes within a single SSM (e.g. virtual leader, group dynamics [67]) and perform coupled sequential inference simultaneously across multiple processes (see also [48]), as would be highly beneficial in applications such as the financial order book examples considered earlier. An interesting simulation of such correlated NHPPs is shown in **Figure 2.10**. This and the parameter learning task are topics of my current research.

Appendix

2.A Proof of Theroem 1

Proof. In order to prove the resultant process after thinning is a NHPP with intensity $\lambda^*p(s)$, it is necessary to ensure its counting measure, denoted as $N_d(\mathcal{T})$, satisfies the three conditions specified in **Definition 1**. It is clear that via independent thinning, the first two conditions are easily satisfied. The focus is thus on the proof of Poissonly distributed counting measure i.e. $N_d(\mathcal{T}) \sim \text{Poisson}\left(\int_{\mathcal{T}} \lambda^*p(s)\right)$. First, write the probability of deleting any one point from the original HPP:

$$\begin{aligned}
 & P(\text{delete any one point}) \\
 &= \int_{\mathcal{T}} P(\text{deletion} \mid \text{point at } s) \times P(\text{point at } s) ds \\
 &= \int_{\mathcal{T}} \frac{1 - p(s)}{|\mathcal{T}|} ds \\
 &= 1 - \frac{\int_{\mathcal{T}} p(s) ds}{|\mathcal{T}|}
 \end{aligned} \tag{2.44}$$

Denoting this probability as \hat{p} , one can therefore write the following conditional probability for non-negative values of n and k :

$$\begin{aligned}
 & P\{N_d(\mathcal{T}) = n \mid N^*(\mathcal{T}) = k\} \\
 &= \begin{cases} \binom{k}{n} (1 - \hat{p})^n \hat{p}^{(k-n)} & \text{if } k \geq n \geq 0 \\ 0 & \text{if } k < n \end{cases}
 \end{aligned} \tag{2.45}$$

To obtain the probability for counting measure $N_d(\mathcal{T})$, simply multiply the above conditional probability by the Poisson probability of the counting measure $N^*(\mathcal{T})$ and marginalise

out the variable k for $k \geq n$:

$$\begin{aligned}
& P\{N_d(\mathcal{T}) = n\} \\
&= \sum_{k=n}^{\infty} \binom{k}{n} \frac{\left(\int_{\mathcal{T}} p(s) ds\right)^n}{|\mathcal{T}|^n} \times \frac{\left(|\mathcal{T}| - \int_{\mathcal{T}} p(s) ds\right)^{k-n}}{|\mathcal{T}|^{k-n}} \times \frac{e^{-\lambda^*|\mathcal{T}|} (\lambda^*|\mathcal{T}|)^k}{k!} \\
&= \frac{e^{-\lambda^*|\mathcal{T}|} \left(\int_{\mathcal{T}} \lambda^* p(s) ds\right)^n}{n!} \sum_{k=n}^{\infty} \frac{\left(\lambda^*|\mathcal{T}| - \int_{\mathcal{T}} \lambda^* p(s) ds\right)^{k-n}}{(k-n)!} \\
&= \frac{e^{-\lambda^*|\mathcal{T}|} \left(\int_{\mathcal{T}} \lambda^* p(s) ds\right)^n}{n!} \times \exp\left\{\lambda^*|\mathcal{T}| - \int_{\mathcal{T}} \lambda^* p(s) ds\right\} \\
&= \frac{\exp\{-\Lambda_d\} (\Lambda_d)^n}{n!}
\end{aligned} \tag{2.46}$$

where $\Lambda_d = \int_{\mathcal{T}} \lambda^* p(s) ds$. From Eq. (2.46), it can be seen that the counting measure $N_d(\mathcal{T})$ of the resultant process follows exactly the Poisson distribution with parameter Λ_d and hence the resultant process is a NHPP with intensity function $\lambda^* p(s)$. \square

2.B SDE solution

For a stochastic process X_t governed by the following SDE:

$$dX_t = AX_t dt + b dW_t \tag{2.47}$$

Re-write the noise process as $dW_t = \xi(t) dt$:

$$\begin{aligned}
& dX_t = AX_t dt + b \xi(t) dt \\
& \frac{dX_t}{dt} - AX_t = b \xi(t) \\
& e^{-At} \frac{dX_t}{dt} - e^{-At} AX_t = e^{-At} b \xi(t) \\
& \frac{d(e^{-At} X_t)}{dt} = e^{-At} b \xi(t)
\end{aligned} \tag{2.48}$$

Integrate both sides of the above equation w.r.t. t , from an arbitrary time S to time T with $S < T$:

$$\begin{aligned}
\int_S^T \frac{d(e^{-At} X_t)}{dt} dt &= \int_S^T e^{-At} b \xi(t) dt \\
(e^{-AT} X_T) - (e^{-AS} X_S) &= \int_S^T e^{-At} b dW_t \\
X_T &= e^{AT} e^{-AS} X_S + e^{AT} \int_S^T e^{-At} b dW_t \\
X_T &= e^{A(T-S)} X_S + e^{A(T-S)} \int_S^T e^{AS} e^{-At} b dW_t \\
X_T &= e^{A(T-S)} \left[X_S + \int_S^T e^{-A(t-S)} b dW_t \right]
\end{aligned} \tag{2.49}$$

Replace $\tau = t - S \rightarrow \frac{d\tau}{dt} = 1$, $t \in [S, T] \mapsto \tau \in [0, T-S]$, and obtain:

$$\begin{aligned}
X_T &= e^{A(T-S)} \left[X_S + \int_0^{T-S} e^{-A\tau} b dW_{\tau+S} \right] \\
&= e^{A(T-S)} \left[X_S + \int_0^{T-S} e^{-A\tau} b dW_\tau \right]
\end{aligned} \tag{2.50}$$

as the required solution to SDE.

2.C Thinning from rejection sampling

Consider the case where a N -event HPP $\{s_n\}_{n=1}^N$ is generated according to a constant upper-bound intensity λ^* ; and $\lambda(s) \leq \lambda^*, \forall s \in \mathcal{T}$ is the varying intensity of the desired NHPP. With aim to sample K ($K \leq N$) events $\{s_k\}_{k=1}^K$, one can write the NHPP target density as:

$$f(\{s_k\}_{k=1}^K) = \exp\left\{-\int_{\mathcal{T}} \lambda(s) ds\right\} \prod_{k=1}^K \lambda(s_k) \tag{2.51}$$

Since it is impossible to sample directly from the target density, one can instead sample a random combination of K events out from the total N events, with proposal:

$$q(\{s_k\}_{k=1}^K) = \frac{1}{\binom{N}{K}} \tag{2.52}$$

To find the tractable bound B for RS, notice that $f(\{s_k\}_{k=1}^K) \leq \exp\left\{-\int_{\mathcal{T}} \lambda(s) ds\right\} (\lambda^*)^K$ and thus the value of B can be chosen such that:

$$B \times q(\{s_k\}_{k=1}^K) = \exp\left\{-\int_{\mathcal{T}} \lambda(s) ds\right\} (\lambda^*)^K \tag{2.53}$$

and gives the acceptance probability:

$$\rho(\{s_k\}_{k=1}^K) = \prod_{k=1}^K \frac{\lambda(s_k)}{\lambda^*} \quad (2.54)$$

which is essentially the probability of the sampled combination $\{s_k\}_{k=1}^K$ being retained in the K independent thinning operations.

Chapter 3

Extended state-space model for LOB market price inference

While the intensity inference model introduced in the previous chapter has demonstrated notable potential for enabling reliable inference routines for LOB, it mainly serves as a generic tool for online intensity estimation/prediction tasks of non-homogeneous Poisson processes. In this chapter, I present a novel state-space modeling approach which facilitates incorporation of additional information from LOB markets such as supply-demand imbalance and trends at varying time scales, with suitable SMC inference algorithms for price estimation/prediction in LOB markets. The performance is evaluated on real LOB datasets. In Section 3.1, I briefly explain the context of LOB inference with a concise review of existing literature. A study of the effect of LOB imbalance on the price movements and trends is then presented, which gives the intuition behind one of the features of the proposed models. Section 3.2 and 3.3 describe the novel state-space models and the corresponding sequential Monte Carlo (SMC) strategy, specifically the Rao-Blackwellised particle filter (RBPF). Results are presented and discussed in Section 3.4.

3.1 Background and LOB imbalance investigation

The continuous evolution of a LOB is the joint result of heterogeneous order operations submitted by a large number of traders and institutions all over the world. The inherent complexity of the LOB allows various approaches to the task of order book inference under different disciplines, which has resulted in an extensive body of literature.

The authors of [68] present studies on the steady-state distributions and limiting statistical properties in a trading system known as the *continuous double auction* (CDA) which is considered as the early prototype of the modern LOB market. In [69], the authors try to tackle a similar task via a sequential model for the distributions of limit order volumes around mid-price and by tracking the time-varying parameters of the proposed

Gamma/inverse-Gamma distributions with a particle filter. In particular, the simulation and reconstruction of LOBs are both important research topics. Poisson processes are employed in [70] and [71] to simulate the LOB structure evolution and price dynamics. Whereas, a variation of the hidden Markov model (HMM) is proposed in [72] to infer the trading information lost during asynchronous LOB updates. The paper of [73] takes a different perspective and derives the optimal order submission strategies for agents trading in the high-frequency LOB market. While the classic price inference models such as GARCH (reviewed in [74]) have been widely used in the financial industry for investment decisions, black-box approaches from the opposite extreme such as deep learning models have also been considered, e.g. in [75], with the use of long short-term memory (LSTM) units. However, the lack of revealing uncertainty information and little transparency in relation to market fundamentals in deep learning models has typically prevented them from being applied to many financial problems in practice.

Since its first introduction in 1993 [76], the particle filter (PF) has been mostly studied and developed to accommodate non-linear and/or non-Gaussian settings in object tracking problems [77]. It has been applied in numerous applications in finance e.g. [78, 79, 80]. In [13], the authors use the Rao-Blackwellised particle filter [81, 82, 83] to infer the market fair price modelled as a non-linear continuous-time jump diffusion process. The latter modelling framework provides a starting point for the work in this chapter.

3.1.1 The effect of LOB imbalance on price movements and trends

Despite promising performance achieved by the model in [13], it conducts *fair price* inference solely based on the dynamics of the price process and does not account for the state of LOB, which somewhat limits the model’s predictive ability when applied to dynamically changing LOBs. At any given timestamp, the limit order book contains the state of all open limit orders for the traded commodity in that particular market. As limit orders are entered, executed (i.e. fulfilled) and cancelled at different price levels, the LOB evolves with these changes in the market micro-structure, which consequently impacts the price dynamics and market trends. **Figure 3.1.1** shows a typical snapshot of limit order volumes at a range of price levels around the mid-price. As can be seen from the figure, asymmetric limit order volumes cause imbalance of the supply and demand (i.e. ask and bid) of the traded commodity, thereby inducing a price pressure towards the opposite side. This can be more obviously observed in the 5 consecutive volume plots in **Figure 3.1.2** as the bid volume depletes and mid-price shifts towards the buyer’s side. The design of the LOB price model capitalises on this simple intuition of supply-demand relation to achieve better prediction performance.

To incorporate the LOB volume imbalance aspect effectively, a weighted contribution of order volumes at different price levels is considered here. Denoting the ‘weight’ of

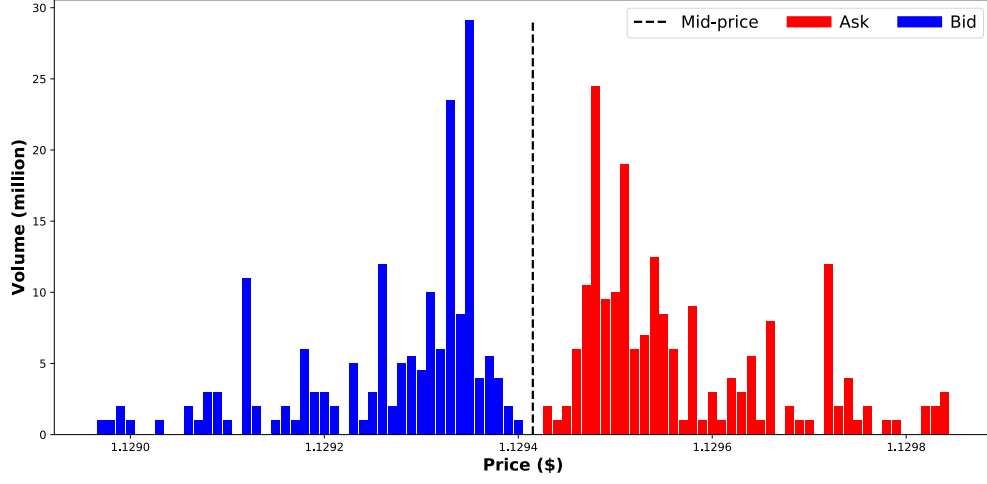


Figure 3.1.1: Volume snapshot of limit orders at a given timestamp in EUR-USD FOREX market.

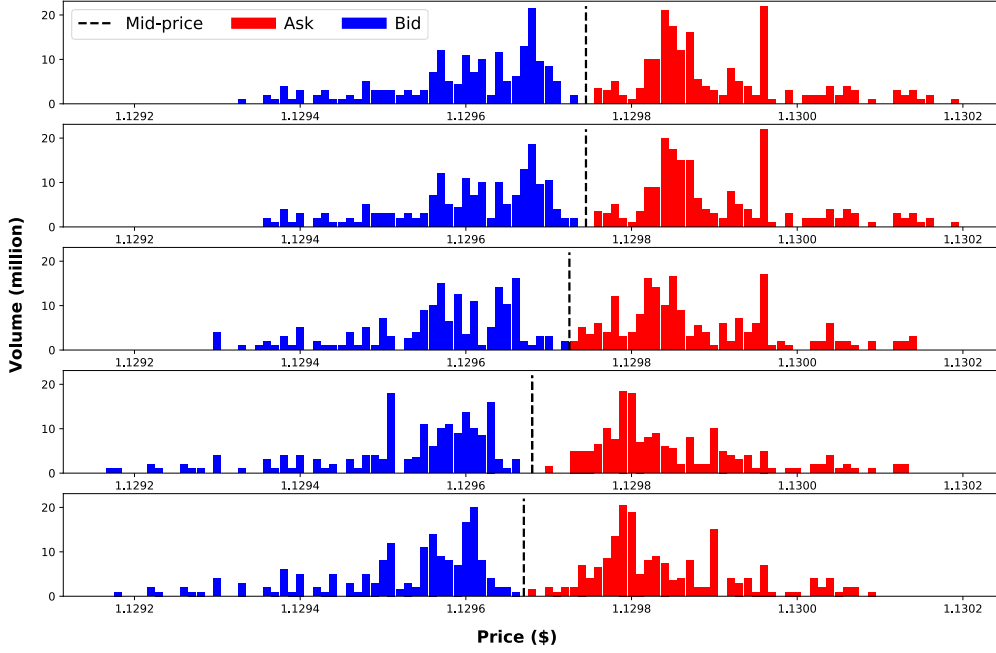


Figure 3.1.2: Volume snapshots taken from 5 consecutive timestamps with intervals less than 0.5s. Time progresses from top to bottom.

volume at price P_k at time t as $w_t^{(k)}$, the following exponential function is assigned to the weight:

$$w_t^{(k)} = \exp\{-c \times d_t^{(k)}\} \quad (3.1)$$

where the $d_t^{(k)}$ is the distance (i.e. number of *ticks*) between P_k and the mid-price at time t and c is a constant. The instantaneous weights decrease exponentially with the

increasing distance to mid-price, which reflects the practitioners’ interests in different price levels as well as the amount of price pressure imposed by each limit order. Similar exponentially decaying functions have also been adopted in [73] and [84] to represent traders’ interest/attention and they can be readily applied within the proposed modelling framework in this thesis.

Define a function $I(t)$ as the difference between the aggregated bid volume pressure and the ask volume pressure:

$$I(t) = \sum_{k=1}^{N_B} w_{t,b}^{(k)} V_{t,b}^{(k)} - \sum_{k=1}^{N_A} w_{t,a}^{(k)} V_{t,a}^{(k)} \quad (3.2)$$

where $V_{t,\cdot}^{(k)}$ is the order volume at price P_k and the subscript a or b represents order type (ask/bid); N_A and N_B are the “depths” or the number of price levels at ask and bid sides respectively. **Figure 3.1.3** shows two overlaid histograms of the function values computed from one-hour high-frequency LOB data with $c = 0.25$. With zero being the reference line, it is visible that the price drop (green) histogram is skewed towards the negative direction of $I(t)$, which implies that higher ask volume pressure is more likely to cause a price drop; and vice versa for the red histogram. This confirms the importance of incorporating LOB volume information for achieving enhanced price inference and prediction.

3.2 Proposed state-space models

In this section, I first introduce the basic jump diffusion model presented in [13] which is used as a starting point and baseline model for benchmarking purposes. Three novel models are then proposed: 1) imbalance-driven volumetric, 2) extended trend and 3) jump trend resetting models. Their objective is to account better for stochastic dependence of the price on the order book state, and incorporation of more behavioural information through Bayesian priors, thereby leading to more realistic modelling and accurate corresponding inference procedures. It is noted that the proposed features from the three models can be combined into one single model, termed the full model.

3.2.1 Review of the jump diffusion model

Due to heterogeneous operations performed by traders, the structure and price of LOB evolve constantly in continuous time, which demands the model’s ability to take irregular inputs at arbitrary timestamps and make continuous inference. In order to accommodate this, the authors of [13] proposed the use of a continuous-time SSM that is commonly adopted for object tracking applications, e.g. the object temporal-spatial characteristics such as position, velocity, and higher order kinematics. With a state vector \mathbf{X}_t containing the “value” $x_{1,t}$ (e.g. sought fair price in LOB) and its “velocity” $x_{2,t}$, the SSM is

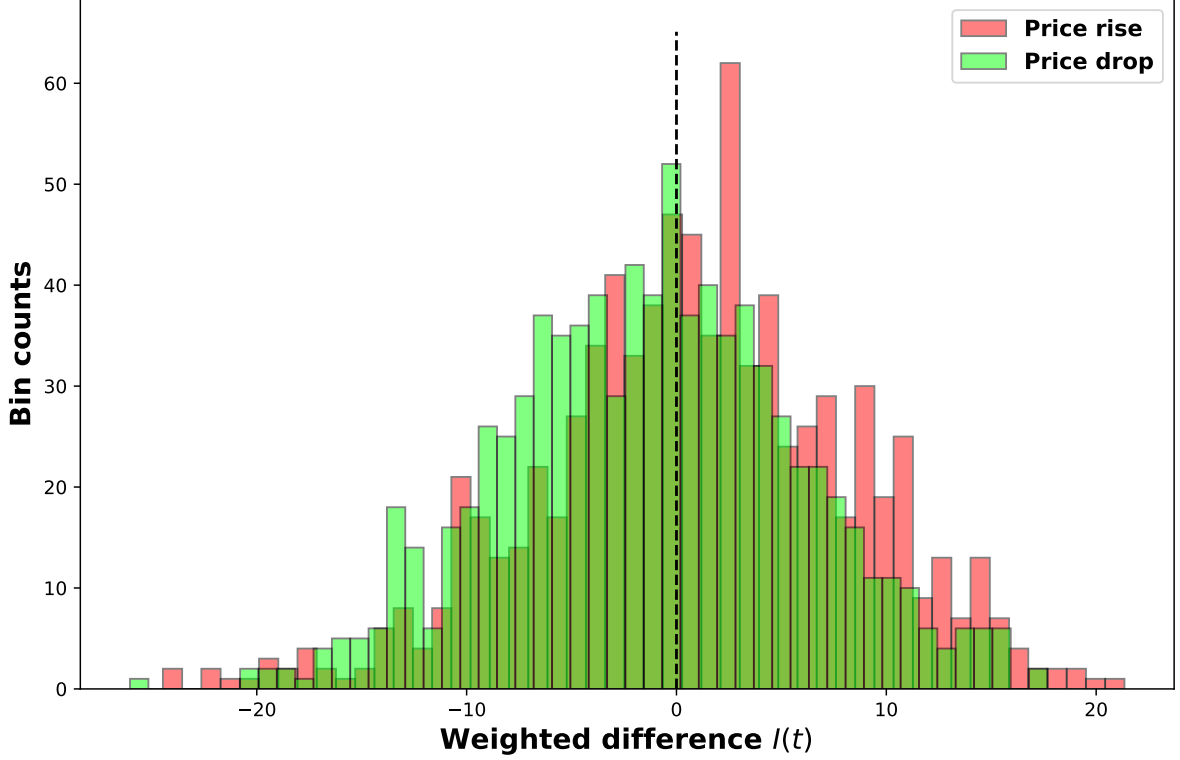


Figure 3.1.3: Two overlaid histograms of the function values $I(t)$ computed from EUR-USD FOREX market for the duration of one hour. The green histogram contains the indicator values which are followed by an immediate mid-price drop; while the red histogram contains those followed by an immediate rise.

formulated with the following dynamics:

$$d\mathbf{X}_t = \mathbf{A}\mathbf{X}_t dt + \mathbf{b} dW_t + \mathbf{c} dJ_t \quad (3.3)$$

where dW_t is a Wiener process accounting for the constant volatility of the market; and dJ_t is the non-linear jump process accounting for impulsive changes in the price process. With y_t being the observed market mid-price at time t , the observation model is constructed as the hidden “fair price” $x_{1,t}$ subject to Gaussian noise with a fixed variance:

$$y_t \sim \mathcal{N}(y_t | \mathbf{G}\mathbf{X}_t, \sigma_{\text{obs}}^2) \quad (3.4)$$

where $\mathbf{G} = [1 \ 0]$ is the fixed observation matrix. With the inherent Markovian property of the SSM construction, this model has provided a sequential framework for online prediction and posterior tracking of the hidden state vector \mathbf{X}_t .

It is worth noting that the SDE in Eq. (3.3) cannot be solved in a closed-form due to the non-linear jump process dJ_t . However, the solution to the SDE without jumps is readily available as detailed in **Appendix 2.B** of **Chapter 2**. It is thus possible to compute the conditional transition density of \mathbf{X}_t between two arbitrary timestamps S

and $T, S \leq T$, conditioned on the jump(s) $\{\tau_i\}_i$ (where i is the index of jumps) occurred in the interval $(S, T]$ and state of \mathbf{X}_S :

$$p(\mathbf{X}_T | \mathbf{X}_S, \{\tau_i\}_i) \quad (3.5)$$

With no jumps in the interval i.e. $\{\tau_i\}_i = \emptyset$, the above density takes exactly the same Gaussian form $\mathcal{N}(\mathbf{X}_T | \boldsymbol{\mu}(T, S), \mathbf{C}(T, S))$ as derived in Eq. (2.8) with parameters:

$$\boldsymbol{\mu}(T, S) = e^{\mathbf{A}(T-S)} \mathbf{X}_S, \quad \mathbf{C}(T, S) = e^{\mathbf{A}(T-S)} \mathbf{K}(T, S) (e^{\mathbf{A}(T-S)})' \quad (3.6)$$

$$\mathbf{K}(T, S) = \int_{t=0}^{T-S} e^{-\mathbf{A}t} \mathbf{b} \mathbf{b}' (e^{-\mathbf{A}t})' dt \quad (3.7)$$

The calculation of transition density with jumps is more complicated as it is necessary to consider diffusion processes in each of the sub-intervals delineated by the jump times $\{\tau_i\}_i$. Denoting the instance right before a jump τ as τ_- and after as τ_+ , the above formulae can be used to obtain the pre-jump distribution parameters $\boldsymbol{\mu}(\tau_{1,-}, S)$ and $\mathbf{C}(\tau_{1,-}, S)$ for the first sub-interval $(S, \tau_1]$. While any jump process can be employed, the original model proposed in [13] utilised a Poisson jump process with a constant intensity λ_J and normally distributed jump size with parameters (μ_J, σ_J^2) . This conveniently leads to a post-jump distribution that is also Gaussian with mean and covariance given by:

$$\begin{aligned} \boldsymbol{\mu}(\tau_{1,+}, S) &= \boldsymbol{\mu}(\tau_{1,-}, S) + \mathbf{c} \mu_J \\ \mathbf{C}(\tau_{1,+}, S) &= \mathbf{C}(\tau_{1,-}, S) + \mathbf{c} \mathbf{c}' \sigma_J^2 \end{aligned} \quad (3.8)$$

The same process is repeated sequentially across all sub-intervals until time T is reached, which leads to a Gaussian transition density from S to T , owing to the Gaussianity maintained between and “during” jumps:

$$p(\mathbf{X}_T | \mathbf{X}_S, \{\tau_i\}_i) = \mathcal{N}(\mathbf{X}_T | \boldsymbol{\mu}(T, S, \{\tau_i\}_i), \mathbf{C}(T, S, \{\tau_i\}_i)) \quad (3.9)$$

with

$$\begin{aligned} \boldsymbol{\mu}(T, S, \{\tau_i\}_i) &= e^{\mathbf{A}(T-S)} \mathbf{X}_S + \sum_i e^{\mathbf{A}(T-\tau_i)} \mathbf{c} \mu_J \\ \mathbf{C}(T, S, \{\tau_i\}_i) &= e^{\mathbf{A}(T-S)} \mathbf{K}(T, S) (e^{\mathbf{A}(T-S)})' + \sum_i e^{\mathbf{A}(T-\tau_i)} \mathbf{c} \mathbf{c}' \sigma_J^2 (e^{\mathbf{A}(T-\tau_i)})' \end{aligned} \quad (3.10)$$

Since the jumps are assumed to follow a homogeneous Poisson process, the interval between two consecutive jumps follows a memoryless exponential distribution:

$$\begin{aligned} p(\tau_1 - S) &= \text{Exponential}(\tau_1 - S | \lambda_J) \\ p(\tau_i - \tau_{i-1} | \tau_{1:i-1}) &= \text{Exponential}(\tau_i - \tau_{i-1} | \lambda_J) \end{aligned} \quad (3.11)$$

Given a set of observed mid-prices $Y = \{y_n\}_{n=1}^N$ at timestamps $\{t_n\}_{n=1}^N$ and denoting $\mathbf{X} = \{\mathbf{X}_{t_n}\}_{n=0}^N$, the joint probability of the model and its conditional propagation factorisation can hence be written as:

$$\begin{aligned} & p(Y, \mathbf{X}, \{\tau\}_{0:N} \mid \Omega) \\ &= p(\mathbf{X}_{t_0}) \times \prod_{n=1}^N p(\{\tau\}_{n-1:n} \mid t_{n-1}, \{\tau\}_{0:n-1}) p(\mathbf{X}_{t_n} \mid \mathbf{X}_{t_{n-1}}, \{\tau\}_{n-1:n}) p(y_n \mid \mathbf{X}_{t_n}) \end{aligned} \quad (3.12)$$

where t_0 is the starting time; Ω is the collection of hyperparameters; and $\{\tau\}_{n-1:n} = \{\tau_i \mid t_{n-1} < \tau \leq t_n\}$.

3.2.2 *Imbalance-driven volumetric model*

The above baseline jump diffusion model is constructed to perform price inference and predictions using only the price input. It is reasonable to anticipate that the performance of the model is largely dependent on the chosen prior dynamics and the tuning of hyperparameters to the specific market and period. Despite the novel addition of a jump process to account for unexpected disturbances on top of the steady diffusion in [13], the model still lacks adaptability to the dynamic market and the LOB.

Recall that the fair price in free markets can be significantly affected by the supply-demand mismatch in both long-term and short-term. In the case of LOB markets, this mismatch is directly represented by the imbalance of limit orders placed on the two sides of the mid-price. Accordingly, a novel imbalance-driven volumetric model is proposed, which takes sequential order volume information from LOB to account for imbalance-innovated price changes.

The use of the following SSM dynamics was proposed in my early studies on volumetric models:

$$d \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \theta_0 \end{bmatrix} \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} dt + \begin{bmatrix} 0 \\ \sigma_0 \end{bmatrix} dW_t + \begin{bmatrix} 0 & 0 & \dots & 0 \\ \phi_1 & \phi_2 & \dots & \phi_K \end{bmatrix} \begin{bmatrix} V_{1,t} \\ V_{2,t} \\ \vdots \\ V_{K,t} \end{bmatrix} dt + \begin{bmatrix} 0 \\ 1 \end{bmatrix} dJ_t \quad (3.13)$$

where θ_0 is the non-positive zero-mean-reverting coefficient on price velocity which represents the resistance force from market to prevent price from going unidirectional (i.e. to maintain a stable price process); σ_0 is the positive scale of the market's constant volatility. The volume vector $[V_{1,t}, \dots, V_{K,t}]^T$, which takes sequential inputs from different levels of the LOB, is scaled linearly by coefficients $\{\phi_k\}_{k=1}^K$.

Although the volumetric model in Eq. (3.13) can provide a means for including the

volume inputs in the modelling of price dynamics, it uses the crude volume information directly from LOB and a potentially high-dimensional coefficient vector $[\phi_1, \dots, \phi_K]$ which needs careful tuning or learning to avoid overfitting effects. To improve such a formulation, I take the intuition from the imbalance analysis presented in Section 3.1.1 by exponentially scaling the volumes from two sides of the mid-price and separately summarising them with the non-linear *log*-function:

$$V_{A,t} = \log \left(\sum_{n=1}^{N_A} e^{-c \times n} V_{n,t} \right), \quad (3.14)$$

$$V_{B,t} = \log \left(\sum_{n=1}^{N_B} e^{-c \times n} V_{-n,t} \right), \quad (3.15)$$

where $V_{j,t}$ is the volume of limit orders at j ticks away from the mid-price at time t ; a positive j represents a ask/sell price and a negative j represents a bid/buy price; N_A and N_B are the total number of price levels for ask and bid respectively. While the exponent c in effect modulates the pressure passed by order volumes, the non-linear *log*-function penalises its contribution towards price trend $x_{2,t}$ when large volumes are presented on both sides (i.e. selling and buying).

Such input of volumetric information not only reduces the dimension/number of model parameters, but also incorporates important prior knowledge of the relative weightings (e.g. traders' attention) of order volumes at different price levels. The volumetric coefficients are captured by the two parameters $\Phi = \{\phi_A, \phi_B\}$. It is possible to re-parameterise the model to include these two coefficients in an augmented state vector, i.e. $\mathbf{X}_t = [x_{1,t} \ x_{2,t} \ \phi_A \ \phi_B]^T$ and perform joint inference on both sequential states \mathbf{X}_t and static parameters Φ via PF. However, such Φ in the extended state has no *forgetting* property, which means that the parameter space is only explored during the initialisation of the PF and each subsequent resampling step will monotonically reduce the diversity of the particles (i.e. degeneracy).

In order to mitigate algorithm degeneracy while allowing a more dynamic input of the LOB volume information, the following improved volumetric SSM is proposed:

$$d\mathbf{X}_t = \mathbf{A}\mathbf{X}_t dt + \mathbf{b} dW_t + \mathbf{c}_A dW_{A,t} + \mathbf{c}_B dW_{B,t} + \mathbf{d} dJ_t$$

$$d \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ \phi_{A,t} \\ \phi_{B,t} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \theta_0 & V_{A,t} & V_{B,t} \\ 0 & 0 & \theta_\phi & 0 \\ 0 & 0 & 0 & \theta_\phi \end{bmatrix} \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ \phi_{A,t} \\ \phi_{B,t} \end{bmatrix} dt + \begin{bmatrix} 0 \\ \sigma_0 \\ 0 \\ 0 \end{bmatrix} dW_t + \begin{bmatrix} 0 \\ 0 \\ \sigma_\phi \\ 0 \end{bmatrix} dW_{A,t} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \sigma_\phi \end{bmatrix} dW_{B,t} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} dJ_t \quad (3.16)$$

where both $\phi_{A,t}$ and $\phi_{B,t}$ are time-varying and each of them follows an independent Ornstein–Uhlenbeck (OU) process [85] instead of being static. It can also be noticed that the transferred volume information $\{V_{A,t}, V_{B,t}\}$ is absorbed into the transition matrix \mathbf{A}

which is now updated seamlessly with observations.

The new SSM in Eq. (3.16) preserves the conditional linear-Gaussian form and consequently the solution to the above SDE and the model's transition density can be readily derived following the same procedure as in [13]. This nonetheless requires a change in the calculation of the integral $\mathbf{K}(T, S)$:

$$\mathbf{K}(T, S) = \int_{t=0}^{T-S} e^{-\mathbf{A}t} (\mathbf{b}\mathbf{b}' + \mathbf{c}_A \mathbf{c}_A' + \mathbf{c}_B \mathbf{c}_B') e^{-\mathbf{A}^T t} dt \quad (3.17)$$

With the observed price normally distributed around the hidden fair price $x_{1,t}$, the proposed imbalance-driven volumetric model in this thesis shares the same form of the joint probability as the baseline model.

3.2.3 Extended trend model

Small and fast fluctuations comprise a large proportion of the price movements within a liquid LOB market. These behaviours are well-captured in the baseline model by the OU process assigned to the price velocity $x_{2,t}$. However, substantial market fluctuations can not only spur larger and sharper changes to the price trend, but may also induce longer-lasting residual effects than general volatility does.

Here, the state vector is extended with an additional term $x_{3,t}$ as the ‘trend regulator’:

$$d \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \theta_0 & -\theta_0 \\ 0 & 0 & \theta_e \end{bmatrix} \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{bmatrix} dt + \begin{bmatrix} 0 \\ \sigma_0 \\ 0 \end{bmatrix} dW_t + \begin{bmatrix} 0 \\ 0 \\ \sigma_e \end{bmatrix} dW_{e,t} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} dJ_t \quad (3.18)$$

where θ_e is the non-positive mean-reverting coefficient for $x_{3,t}$, σ_e is the innovation scale and $dW_{e,t}$ is a unit-variance Wiener process. Whereas, the dynamics of $x_{2,t}$ are formulated as:

$$dx_{2,t} = \theta_0(x_{2,t} - x_{3,t}) dt + \sigma_0 dW_t + dJ_t \quad (3.19)$$

In the case where θ_0 is non-negative and $x_{3,t}$ is a constant, the above dynamics resemble a constant-mean OU process. However, by fitting another zero-mean OU process to $x_{3,t}$ with perhaps less negative θ_e compared to θ_0 , the momentum/trend term $x_{2,t}$ can additionally gains longer-term trends (of a maybe smaller scale if $\sigma_e < \sigma_0$) by following the regulator $x_{3,t}$. And by tuning θ_0 within $(-\infty, 0]$, it is possible for $x_{2,t}$ to achieve any dynamics between a Brownian diffusion with jumps (i.e. $\theta_0 = 0$) and a quick follower of $x_{3,t}$ (i.e. $\theta_0 \rightarrow -\infty$). The new extended trend model allows trends at different time scales to be maintained and thus gives richer dynamics to the modelled fair-price process $x_{1,t}$.

3.2.4 Jump trend resetting model

Sharp jumps in the market are often triggered by non-market factors which are hard to account for. These occurrences are likely to disturb any balance or correlation established in the market, which then reforms gradually afterwards.

In order to accommodate this particular characteristic of the market, a trend-resetting mechanism is proposed for the jump process. For a jump at time τ between two input timestamps S and T (i.e. $S \leq \tau \leq T$), one can consider the segmented diffusion from S to the instance τ_- right before the jump and obtain the Gaussian propagation density with parameters $\boldsymbol{\mu}(\tau_-, S)$ and $\mathbf{C}(\tau_-, S)$ according to Eq. (3.6). Instead of applying a normally distributed jump size to the post-jump parameters, the new trend reset model proceeds as follows:

$$\boldsymbol{\mu}(\tau_+, S) = \begin{bmatrix} \boldsymbol{\mu}(\tau_-, S)^{(1)} \\ \boldsymbol{\mu}_{\text{prior}} \end{bmatrix}, \quad \mathbf{C}(\tau_+, S) = \begin{bmatrix} \mathbf{C}(\tau_-, S)^{(1,1)} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{C}_{\text{prior}} \end{bmatrix}. \quad (3.20)$$

where the superscripts indicate the indices of elements taken from the pre-jump parameters. For a M -dimensional state vector, both $\boldsymbol{\mu}_{\text{prior}}$ and $\mathbf{0}$ are $(M-1)$ -dimensional (column) vector and $\mathbf{C}_{\text{prior}}$ is a $(M-1)$ -by- $(M-1)$ matrix. In practice, $\mathbf{C}_{\text{prior}}$ can take the form of $\sigma_{\text{prior}}^2 \times \mathbf{I}$, where \mathbf{I} is the identity matrix and σ_{prior}^2 should take a much larger value (e.g. $\geq 10\times$) than σ_0^2 . Such a trend-resetting jump process helps avoid overfitting from the converged dynamic parameters and allows faster adaptation towards a new market dynamics. Multiple jumps within a single update interval can be handled analogously in sequence.

3.2.5 Full model

Each of the three models proposed above accounts for different market behaviours with a specific model design, e.g. extended state vector or Bayesian priors. The orthogonality in the designs allows us to combine the models into a “full model”. In this case, the state vector $\mathbf{X}_t = [x_{1,t} \ x_{2,t} \ x_{3,t} \ \phi_{A,t} \ \phi_{B,t}]^T$ has both extended trend regulator and volume coefficients in addition to the value and trend terms:

$$d \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \\ \phi_{A,t} \\ \phi_{B,t} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & \theta_0 & -\theta_0 & V_{A,t} & V_{B,t} \\ 0 & 0 & \theta_e & 0 & 0 \\ 0 & 0 & 0 & \theta_\phi & 0 \\ 0 & 0 & 0 & 0 & \theta_\phi \end{bmatrix} \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \\ \phi_{A,t} \\ \phi_{B,t} \end{bmatrix} dt + \begin{bmatrix} 0 \\ \sigma_0 \\ 0 \\ 0 \\ 0 \end{bmatrix} dW_t + \begin{bmatrix} 0 \\ 0 \\ \sigma_e \\ 0 \\ 0 \end{bmatrix} dW_{e,t} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \sigma_\phi \\ 0 \end{bmatrix} dW_{A,t} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \sigma_\phi \end{bmatrix} dW_{B,t} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} dJ_t \quad (3.21)$$

The jump resetting mechanism operates on all states except the “value” term $x_{1,t}$ following Eq. (3.20). Section 3.4 also presents the results obtained from this full model and the qualitative analyses on the interpretations of each individual feature.

3.3 Inference

The models proposed in this chapter all focus on the specific design of SSM and price dynamics, which allows them to share the same joint probability format in (3.12). Given the models' formulations, the same inference routine can be applied to all of the proposed models; two suitable particle filtering algorithms are detailed in this section.

3.3.1 Generic bootstrap particle filter

Taking advantage of the Markovian property of the SSM, sequential prediction and posterior inference of the state vectors and jumps can be achieved by first writing the recursion of the joint posterior (online) smoothing density as:

$$\begin{aligned} & p(\mathbf{X}_{0:t_n}, \{\tau\}_{0:n} \mid y_{1:n}) \\ &= p(\mathbf{X}_{0:t_{n-1}}, \{\tau\}_{0:n-1} \mid y_{1:n-1}) \times \\ & \quad \frac{p(\{\tau\}_{n-1:n} \mid \{\tau\}_{0:n-1}) p(\mathbf{X}_{t_n} \mid \mathbf{X}_{t_{n-1}}, \{\tau\}_{n-1:n}) p(y_n \mid \mathbf{X}_{t_n})}{p(y_n \mid y_{1:n-1})} \end{aligned} \quad (3.22)$$

where the dependency on hyperparameters Ω has been omitted for simplicity. Although the recursive posterior cannot be computed in closed-form due to the non-linear jumps, empirical approximation of it can be readily obtained with a PF. Suppose that the posterior filtering density at time t_{n-1} can be approximated by a large and random collection of N_p weighted particles:

$$\begin{aligned} p(\mathbf{X}_{0:t_{n-1}}, \{\tau\}_{0:n-1} \mid y_{1:n-1}) &\approx \hat{p}(\mathbf{X}_{0:t_{n-1}}, \{\tau\}_{0:n-1} \mid y_{1:n-1}) \\ &= \sum_{p=1}^{N_p} w_{n-1}^p \delta(\mathbf{X}_{0:t_{n-1}}^p, \{\tau\}_{0:n-1}^p) \end{aligned} \quad (3.23)$$

where $w_{n-1}^p \geq 0$ and $\sum_{p=1}^{N_p} w_{n-1}^p = 1$. By substituting the empirical posterior (3.23) at t_{n-1} into the posterior recursion (3.22), the propagation scheme of a bootstrap PF for a particular particle p can be obtained as:

1. Initialise $\tau' = t_{n-1}$ and $\{\tau\}_{n-1:n}^p = \emptyset$.
2. While $\tau' < t_n$, sample a new jump interval from the prior $\Delta\tau \sim \text{Exponential}(\lambda_J)$; compute the new jump time $\tau' = \tau' + \Delta\tau$; and append τ' to $\{\tau\}_{n-1:n}^p$ if $\tau' \leq t_n$.
3. Sample state vector from the conditional transition density in Eq. (3.9):
 $\mathbf{X}_{t_n}^p \sim p(\mathbf{X}_{t_n} \mid \mathbf{X}_{t_{n-1}}^p, \{\tau\}_{n-1:n}^p)$
4. Compute the unnormalised weight $\tilde{w}_n^p = w_{n-1}^p \times p(y_n \mid \mathbf{X}_{t_n}^p, \sigma_{\text{obs}}^2)$

where the weights are re-normalised after all particles have been propagated $w_n^p = \tilde{w}_n^p / \sum_j \tilde{w}_n^j$; and the resultant weights and particle form the collection for the next propagation from t_n to t_{n+1} .

3.3.2 Rao-Blackwellised particle filter (RBPF)

The bootstrap PF is known to suffer from the curse of dimensionality [86] and its performance can be compromised when the state vector dimensionality grows (e.g. in the volumetric model) due to the lack of exploration of the continuous sample space. However owing to the linear Gaussian form of both state transition density and likelihood defined in Section 3.2, it is possible to marginalise out the state vectors in each propagation step conditional on a given set of jump times e.g. $\{\tau\}_{0:n}$ and utilise the RBPF algorithm as a combination of Kalman filter and particle filter [18, 27, 87]. Suppose that the posterior filtering density of the vector $\mathbf{X}_{t_{n-1}}$ at time t_{n-1} can be expressed as a Gaussian distribution:

$$p(\mathbf{X}_{t_{n-1}} | y_{1:n-1}, \{\tau\}_{0:n-1}) = \mathcal{N}(\mathbf{X}_{t_{n-1}} | \boldsymbol{\mu}_{n-1|0:n-1}, \boldsymbol{\Sigma}_{n-1|0:n-1}) \quad (3.24)$$

Given the jump times $\{\tau\}_{n-1:n}$ within the update interval $(t_{n-1}, t_n]$, one can write the predictive distribution of \mathbf{X}_{t_n} as:

$$\begin{aligned} p(\mathbf{X}_{t_n} | y_{1:n-1}, \{\tau\}_{0:n}) &= \int p(\mathbf{X}_{t_{n-1}} | y_{1:n-1}, \{\tau\}_{0:n-1}) p(\mathbf{X}_{t_n} | \mathbf{X}_{t_{n-1}}, \{\tau\}_{n-1:n}) d\mathbf{X}_{t_{n-1}} \\ &= \mathcal{N}(\mathbf{X}_{t_n} | \boldsymbol{\mu}_{n|0:n-1}, \boldsymbol{\Sigma}_{n|0:n-1}) \end{aligned} \quad (3.25)$$

where the Gaussian parameters can be written as linear transformations of the posterior parameters $\boldsymbol{\mu}_{n-1|0:n-1}$ and $\boldsymbol{\Sigma}_{n-1|0:n-1}$:

$$\boldsymbol{\mu}_{n|0:n-1} = e^{\mathbf{A}(t_n - t_{n-1})} \boldsymbol{\mu}_{n-1|0:n-1} + \sum_{\tau_i \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_i)} \mathbf{c} \mu_J \quad (3.26)$$

$$\boldsymbol{\Sigma}_{n|0:n-1} = e^{\mathbf{A}(t_n - t_{n-1})} \boldsymbol{\Sigma}_{n-1|0:n-1} e^{\mathbf{A}^T(t_n - t_{n-1})} + \mathbf{C}(t_n, t_{n-1}, \{\tau\}_{n-1:n}) \quad (3.27)$$

where \mathbf{c} is defined as the vector coefficient of the jump process dJ_t in each model and $\mathbf{C}(t_n, t_{n-1}, \{\tau\}_{n-1:n})$ is defined in (3.10). As an exception, the jump trend resetting model (and the full model) has a slightly different transition density conditional on jumps. Nonetheless, the closed-form predictive Gaussian densities can still be obtained as the resetting mechanism also preserves Gaussianity. Utilising these formulations, a classic Kalman filter can be applied on the state vectors (conditional on the jump times) and

obtain the up-to-date posterior parameters that are “corrected” by the observation y_n :

$$\mathbf{Q}_n = \Sigma_{n|0:n-1} \mathbf{G}^T \left(\mathbf{G} \Sigma_{n|0:n-1} \mathbf{G}^T + \sigma_{\text{obs}}^2 \right)^{-1} \quad (3.28)$$

$$\boldsymbol{\mu}_{n|0:n} = \boldsymbol{\mu}_{n|0:n-1} + \mathbf{Q}_n \left(y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1} \right) \quad (3.29)$$

$$\Sigma_{n|0:n} = \left(\mathbf{I} - \mathbf{Q}_n \mathbf{G} \right) \Sigma_{n|0:n-1} \quad (3.30)$$

Similar to the generic Kalman filter, it is necessary to specify the initial distribution for the state \mathbf{X}_{t_0} at time t_0 in order to start the filtering process. The conditional Kalman filter requires this initialisation to be Gaussian. Therefore, a prior mean $\boldsymbol{\mu}_{\text{prior}}$ and a prior covariance matrix Σ_{prior} are adopted as the initial parameters $\boldsymbol{\mu}_{0|0}$ and $\Sigma_{0|0}$ for the first prediction step of the filter i.e. (3.26) and (3.27). It is also worth noting that the initialisation of the filter can vary depending on the specific application of the model. For example, certain applications may demand an accurate initial state; other online inference scenarios are less sensitive to initialisation.

With the state vectors Rao-Blackwellised/marginalised, one can then use a variable-rate particle filter (VRPF) for the posterior recursion that targets only the jump times [27, 87]:

$$\begin{aligned} & p(\{\tau\}_{0:n} | y_{1:n}) \\ & \propto p(\{\tau\}_{0:n-1} | y_{1:n-1}) p(\{\tau\}_{n-1:n} | \{\tau\}_{0:n-1}) \int p(\mathbf{X}_{t_n} | y_{1:n-1}, \{\tau\}_{0:n}) p(y_n | \mathbf{X}_{t_n}) d\mathbf{X}_{t_n} \quad (3.31) \\ & \propto p(\{\tau\}_{0:n-1} | y_{1:n-1}) p(\{\tau\}_{n-1:n} | \{\tau\}_{0:n-1}) \times p(y_n | y_{1:n-1}, \{\tau\}_{0:n}) \end{aligned}$$

where the integral can be spotted as the *prediction error decomposition* (PED) of the Kalman filter [88]:

$$p(y_n | y_{1:n-1}, \{\tau\}_{0:n}) = \mathcal{N}(y_n | \mu_{y_n}, \Sigma_{y_n}) \quad (3.32)$$

$$\mu_{y_n} = \mathbf{G} \boldsymbol{\mu}_{n|0:n-1} \quad (3.33)$$

$$\Sigma_{y_n} = \mathbf{G} \Sigma_{n|0:n-1} \mathbf{G}^T + \sigma_{\text{obs}}^2 \quad (3.34)$$

Subsequently, the posterior recursion of (3.31) can be empirically approximated in a similar manner as for a generic bootstrap PF algorithm. Therefore, an arbitrary particle p at timestamp t_n that will be propagated in the RBPF algorithm should include: (1) a sequence of proposed jump times $\{\tau\}_{0:n}^p$; (2) the posterior filtering parameters for state vector \mathbf{X}_{t_n} , $(\boldsymbol{\mu}_{n|0:n}^p, \Sigma_{n|0:n}^p)$; and (3) its current normalised weight w_n^p . Express this as:

$$P_n^{(p)} = \left\{ \{\tau\}_{0:n}^p, \boldsymbol{\mu}_{n|0:n}^p, \Sigma_{n|0:n}^p, w_n^p \right\} \quad (3.35)$$

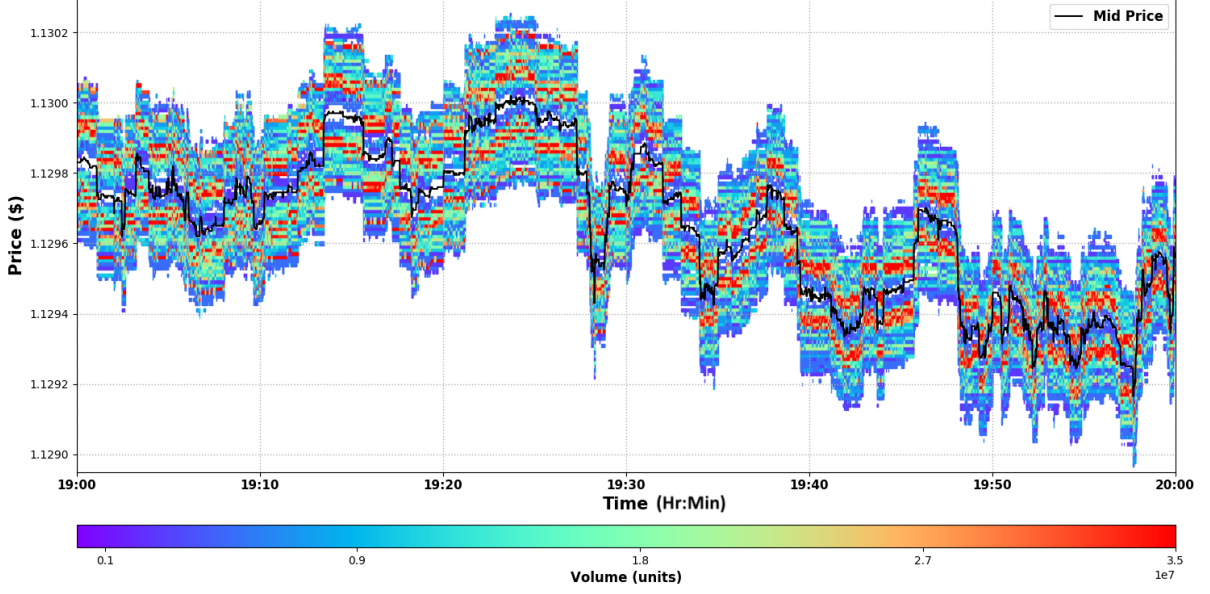


Figure 3.4.1: The snapshot plot of the one-hour FOREX data used for the experiment. The orders are plotted as rectangles around the mid-price with the corresponding volumes indicated by the colourmap. Orders that are placed farther away from the mid-price (i.e. deeper into the LOB) are not shown as they contribute negligibly to the market dynamics.

and the re-weighting of the particle at timestamps t_n can be performed as follows:

$$\tilde{w}_n^p = w_{n-1}^p \times \frac{p(\{\tau\}_{n-1:n}^p | \{\tau\}_{0:n-1}^p) \times p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^p)}{q(\{\tau\}_{n-1:n}^p | \{\tau\}_{0:n-1}^p)} \quad (3.36)$$

where $p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^p)$ is the normal PED in (3.32) conditioned on the particular sequence of jump times in the p th particle; and $q(\cdot)$ is the proposal density of the jumps. In the case of a bootstrap RBPF, the proposal used for jumps is identical to the jump prior and the weight update essentially reduces to

$$\tilde{w}_n^p = w_{n-1}^p \times p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^p) \quad (3.37)$$

Algorithm 7 demonstrates the pseudo-code of the RBPF algorithm with an adaptive resampling scheme used for the proposed models. The adaptive importance resampler computes the effective sample size $N_{\text{eff}} = 1 / \sum_p (w_n^p)^2$ at each timestamp t_n and performs resampling only if N_{eff} falls below a certain threshold N_{thr} , see [89] for further details on adaptive importance resampling. With state vectors marginalised, RBPF efficiently mitigates the curse of dimensionality which is detrimental to the performance of particle filters [86]. In the next section, the performance of the introduced SSMs and inference algorithms are evaluated using real data.

Algorithm 7 Bootstrap Rao-Blackwellised particle filter

Inputs: timestamps $\{t_n\}_{n=0}^N$; mid-price $\{y_n\}_{n=1}^N$
Outputs: A collection of inferred particles $\{P_N^{(p)}\}_{p=1}^{N_p}$

- 1: **Initialisation:** (n=0) Create a collection of identical particles: $\{\tau\}_0^p = \emptyset$, $\mu_{0|0}^p = \mu_{\text{prior}}$, $\Sigma_{0|0}^p = \Sigma_{\text{prior}}$, $w_0^p = 1/N_p$.
- 2: **for** $n = 1 : N$ **do**
- 3: **for** $p = 1 : N_p$ **do**
- 4: Set $\tau' = t_{n-1}$ and $\{\tau\}_{n-1:n}^p = \emptyset$
- 5: **while** $\tau' < t_n$ **do**
- 6: Sample jump interval $\Delta\tau \sim \text{Exponential}(\lambda_J)$
- 7: $\tau' = \tau' + \Delta\tau$
- 8: **if** $\tau' \leq t_n$ **then**
- 9: Append τ' to $\{\tau\}_{n-1:n}^p$
- 10: **end if**
- 11: **end while**
- 12: Compute predictive parameters $(\mu_{n|0:n-1}^p, \Sigma_{n|0:n-1}^p)$ conditioned on $\{\tau\}_{n-1:n}^p$ using (3.26) and (3.27).
- 13: Compute (unnormalised) weight $\tilde{w}_n^p = w_{n-1}^p \times p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^p)$
- 14: Kalman update $(\mu_{n|0:n}^p, \Sigma_{n|0:n}^p)$ with (3.29) and (3.30)
- 15: **end for**
- 16: Normalise weights $w_n^p = \tilde{w}_n^p / \sum_j \tilde{w}_n^j$
- 17: Compute effective sample size $N_{\text{eff}} = 1.0 / \sum_p (w_n^p)^2$
- 18: **if** $N_{\text{eff}} < N_{\text{thr}}$ **then**
- 19: **Perform Importance Resampling**
- 20: **end if**
- 21: **end for**
- 22: **Return:** $\{P_N^{(p)}\}_{p=1}^{N_p}$

Table 3.4.1: Hyperparameters used for predictive performance model comparisons.

σ_0	θ_0	σ_e	θ_e	σ_ϕ	θ_ϕ	σ_{obs}	σ_J	μ_J	λ_J	μ_{prior}	$\mathbf{C}_{\text{prior}}$
0.08	-0.7	0.03	-0.4	0.005	-0.5	0.1	1.0	0	0.1	$\vec{0}$	$0.1 \times \mathbf{I}$

3.4 Results and discussions

In this section, I apply the proposed models as well as the baseline model in [13] on a set of real-world high-frequency FOREX data and present empirical results obtained from this dataset. The predictive outcome of the models are numerically compared based on different metrics and the interpretations of the added model features are qualitatively investigated.

The models are tested on a set of one-hour (19:00 – 20:00 ET) LOB data from the EUR-USD FOREX market on the 2nd of September, 2015. The one-hour snapshot is reconstructed using the raw LOB data depicted in **Figure 3.4.1**. As shown in the figure, the *bid-ask spread* is very small during this period indicating high liquidity of the market. It is also visible that the market matures progressively over the one-hour period as the

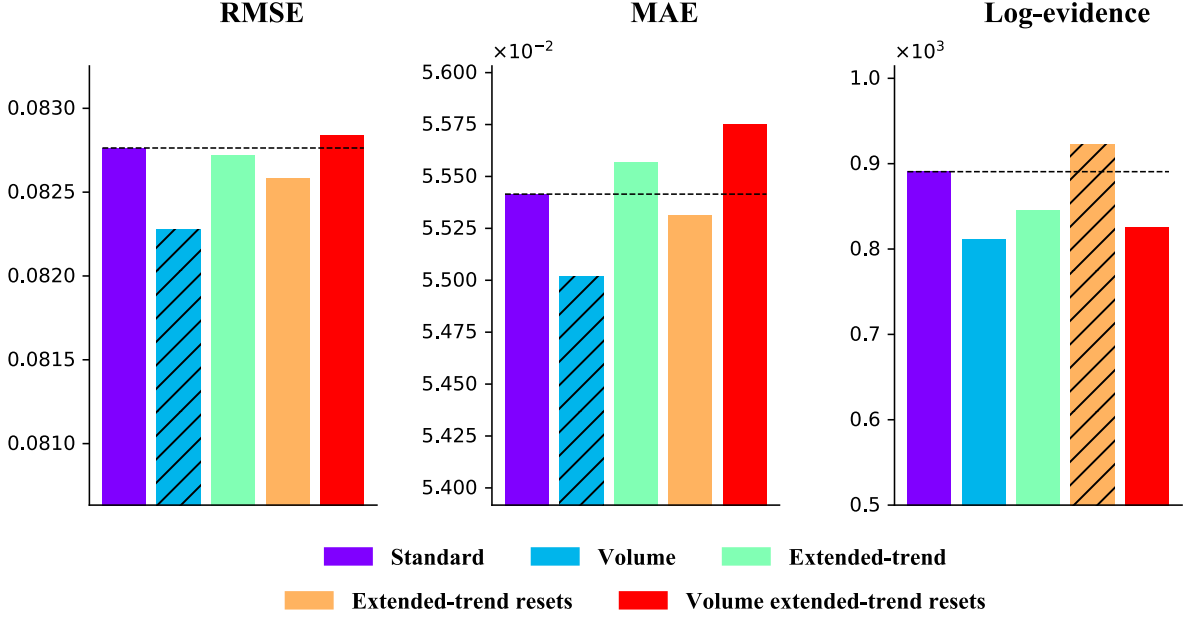


Figure 3.4.2: Figure shows the root mean squared error (RMSE), mean absolute error (MAE) and approximated *log*-marginal evidence of the models. The best models are indicated by hatching. The dashed lines give the values of the standard model as references.

volumes of limit orders increase i.e. more solid red rectangles and the *bid-ask spread* closes up further.

In order to ensure the model operates on a sensible numerical scale and to ease the tuning of hyperparameters, the mid-price data are normalised to have zero mean and unit variance. Additionally, the summarised LOB volume information is normalised by the maximum value (i.e. the resultant range is from 0 to 1). In particular, the performance of the following models is investigated:

- the baseline jump diffusion model as proposed in [13];
- the volumetric model with summarised LOB volume inputs;
- the extended trend model;
- the extended trend model with jump trend resetting mechanism;
- and the volumetric model with extended trend and jump trend resetting mechanism, which referred to as the “full model”.

The hyperparameters are heuristically tuned for comparative model evaluations and are shown in **Table 3.4.1** (i.e. no fine tuning is undertaken or learning routine for optimisation utilised). The RBPF algorithm is applied for inference using 100 particles (i.e. $N_p = 100$) and 50%-threshold (i.e. $N_{thr} = 0.5N_p$) adaptive resampling scheme. **Figure 3.4.2** shows the predictive performance of the five models evaluated numerically based on the three metrics:

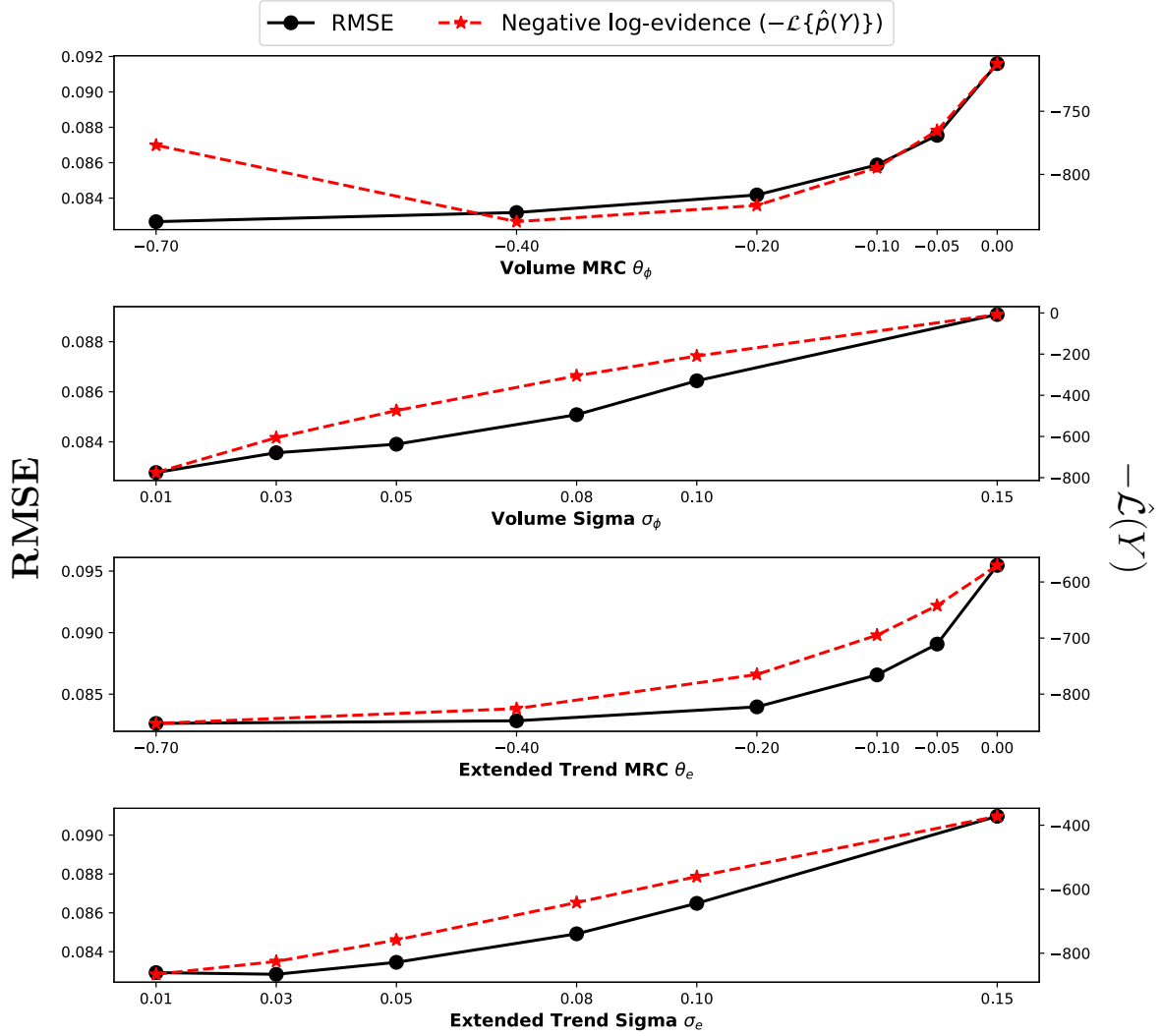


Figure 3.4.3: The prediction RMSE and the (approximated) log-evidence of the full model under different settings of hyperparameters $\{\sigma_\phi, \theta_\phi, \sigma_e, \phi_e\}$. The model hyperparameters took the values listed in **Table 3.4.1** except the one investigated which is indicated on x -axis.

1. The root mean squared error (RMSE) of the predictive mean $\mu_{n|0:n-1}$.

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{n=1}^N \left(y_n - \mathbf{G} \sum_{p=1}^{N_p} w_{n-1}^p \mu_{n|0:n-1}^p \right)^2} \quad (3.38)$$

2. The mean absolute error (MAE) of the predictive mean.

$$\text{MAE} = \frac{1}{N} \sum_{n=1}^N \left| y_n - \mathbf{G} \sum_{p=1}^{N_p} w_{n-1}^p \mu_{n|0:n-1}^p \right| \quad (3.39)$$

3. The approximated (*log*) marginal model evidence computed from the unnormalised

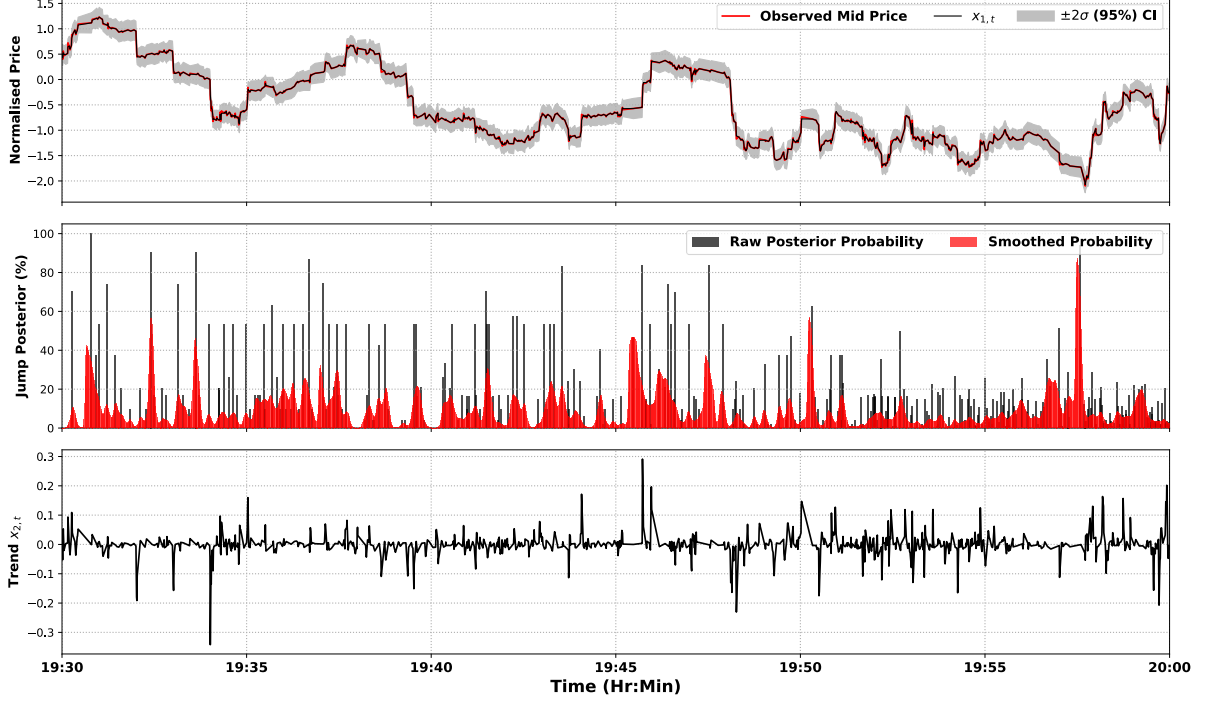


Figure 3.4.4: The posterior (smoothing) results of the standard model for the second half hour (19:30 – 20:00 ET). The middle panel shows both raw and smoothed posterior probability of jumps within each update interval i.e. $\{(t_{n-1}, t_n]\}_{n=1}^N$.

particle weights:

$$\begin{aligned} \log\{p(Y)\} &\approx \hat{\mathcal{L}}(Y) = \log\left\{\prod_{n=0}^N \left[\sum_{p=1}^{N_p} \tilde{w}_n^p\right]\right\} \\ &= \sum_{n=0}^N \log\left\{\sum_{p=1}^{N_p} \tilde{w}_n^p\right\} \end{aligned} \quad (3.40)$$

The first two metrics measure the mean error of models' prediction; while the marginal evidence quantifies the model fit from a Bayesian perspective, which is analogous to the *p-value* used in evaluating frequentist approaches but accounts for both potential overfit and underfit. Comparing to the standard model, it can be seen from **Figure 3.4.2** that the inclusion of volume information helps improve the accuracy of mean predictions made by the model. However, the log-evidence is reduced due to perhaps the sequential dynamics and stochasticity introduced into the volumetric variables $\{\phi_{A,t}, \phi_{B,t}\}$. The extended trend term $x_{3,t}$ provides minor improvement on mean-prediction accuracy; while its combination with trend-resetting jumps improves the overall model fit.

It can also be observed that for this dataset, the collective use of all proposed features (i.e. the full model) does not necessarily yields better results. Therefore, additional tests are performed on the full model with different settings of the hyperparameters. **Figure 3.4.3** shows the RMSE and negative log-evidence obtained on the same set of data. It is clear from the figure that both volumetric variables $\{\phi_{A,t}, \phi_{B,t}\}$ and extended trend $x_{3,t}$

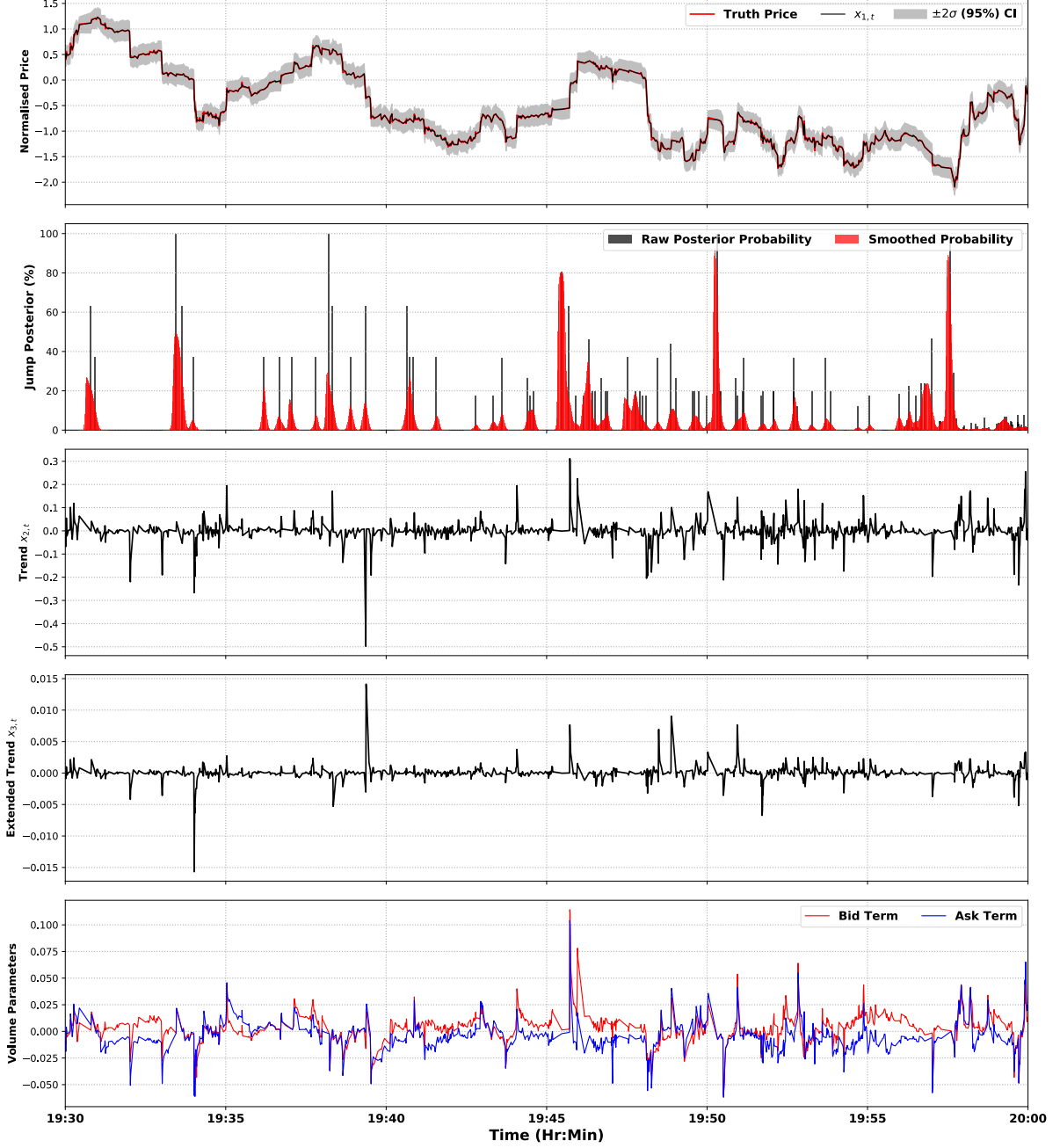


Figure 3.4.5: The posterior (smoothing) results of the full model for the second half hour (19:30 – 20:00 ET). In addition to the variables in the standard model, the figure also displays the posterior mean of the extended trend $x_{3,t}$ and the two volumetric variables $\{\phi_{A,t}, \phi_{B,t}\}$.

in the full model favour dynamics with relatively strong zero-mean reversion and small innovation. The small σ_e and large magnitude of θ_e both indicate that the price process in the selected interval exhibits very little long-term trend. Even though the small σ_ϕ may be favoured due to the large scale of order volumes, the comparison of different θ_ϕ 's indicates that the volume-driven trend is also non-persistent throughout the one-hour period of interest. This lack of long/medium-term dynamics in the price process is easily observed from the snapshot plot in **Figure 3.4.1**.

In addition to the predictive performance, one may also be interested in the retrospective interpretations of the posteriors of the model's variables. Here, I compare the posteriors of the standard model and the full model on the one-hour LOB data. **Figure 3.4.4** and **3.4.5** show the posterior tracking results of the second half hour obtained using the standard model and the full model respectively. In this experiment, the mean-reverting coefficient θ_ϕ in the full model is set to -0.05 such that more characteristics of the volumetric variables $\{\phi_{A,t}, \phi_{B,t}\}$ can be expressed. The posterior jump probability of each interval $(t_{n-1}, t_n]$ is computed by summing the weights of all particles $P_N^{(p')}$ that each has at least one jump within $(t_{n-1}, t_n]$, i.e.:

$$P(\text{jump(s) in } (t_{n-1}, t_n]) = \sum_{p' \in \Omega_n} w_N^{p'} \quad , \quad \Omega_n = \{p \mid \{\tau\}_{0:N}^p \cap (t_{n-1}, t_n] \neq \emptyset\} \quad (3.41)$$

It can be seen from the bottom panel of **Figure 3.4.5** that even without constraining the positivity of $\phi_{B,t}$ (or the negativity of $\phi_{A,t}$) in its dynamics, the posterior means of the volumetric variables are still able to take opposite directions with respect to zero. The bid term $\phi_{B,t}$ modulates the buying pressure and contributes positively towards the price trend; while the ask term $\phi_{A,t}$ modulates the selling pressure and takes mostly negative values for downward trend. In the case of significant jumps in the price, both bid and ask terms jointly drive the trend in the jump direction (e.g. the blue and red curves coincide). In real-world markets, the pre-jump outstanding orders are likely to be either cancelled (e.g. via *stop-limit order*) or fulfilled quickly when jumps take place and new orders are often placed for market probing purposes. Therefore the volume coefficients barely reveal any adversarial imbalance information during these periods

In the standard model, as the single velocity process $x_{2,t}$ is unable to capture all the price fluctuations, the designed jump process is employed to account for many small/medium fluctuations in the price in addition to the significant jumps as can be seen from **Figure 3.4.4**. On the other hand, the posterior jumps inferred for the full model are much clearer since the fluctuations are accounted by instantaneous LOB imbalance. Moreover, recall that the “jumps” in the full model in fact represent the trend-resetting mechanism, thus it can be observed in **Figure 3.4.5** that some “jumps” happen prior to major price changes. This resets the volumetric variables $\{\phi_{A,t}, \phi_{B,t}\}$ (to prior) to allow adaptation of the sharp trends. Due to the lack of long/medium-term price dynamics in the period, the extended trend term $x_{3,t}$ shows little interesting characteristics in its posterior.

Figure 3.4.6 depicts the posterior means of the volumetric variables using different settings of $(\sigma_\phi, \theta_\phi)$, together with the corresponding aggregated signal $(\phi_{A,t} V_{A,t} + \phi_{B,t} V_{B,t})$ that contributes to the trend $x_{2,t}$. More specifically, the model is tested under two other extreme settings: (i) no dynamics assigned to volumetric variables i.e. $\theta_\phi = \sigma_\phi = 0$; (ii)

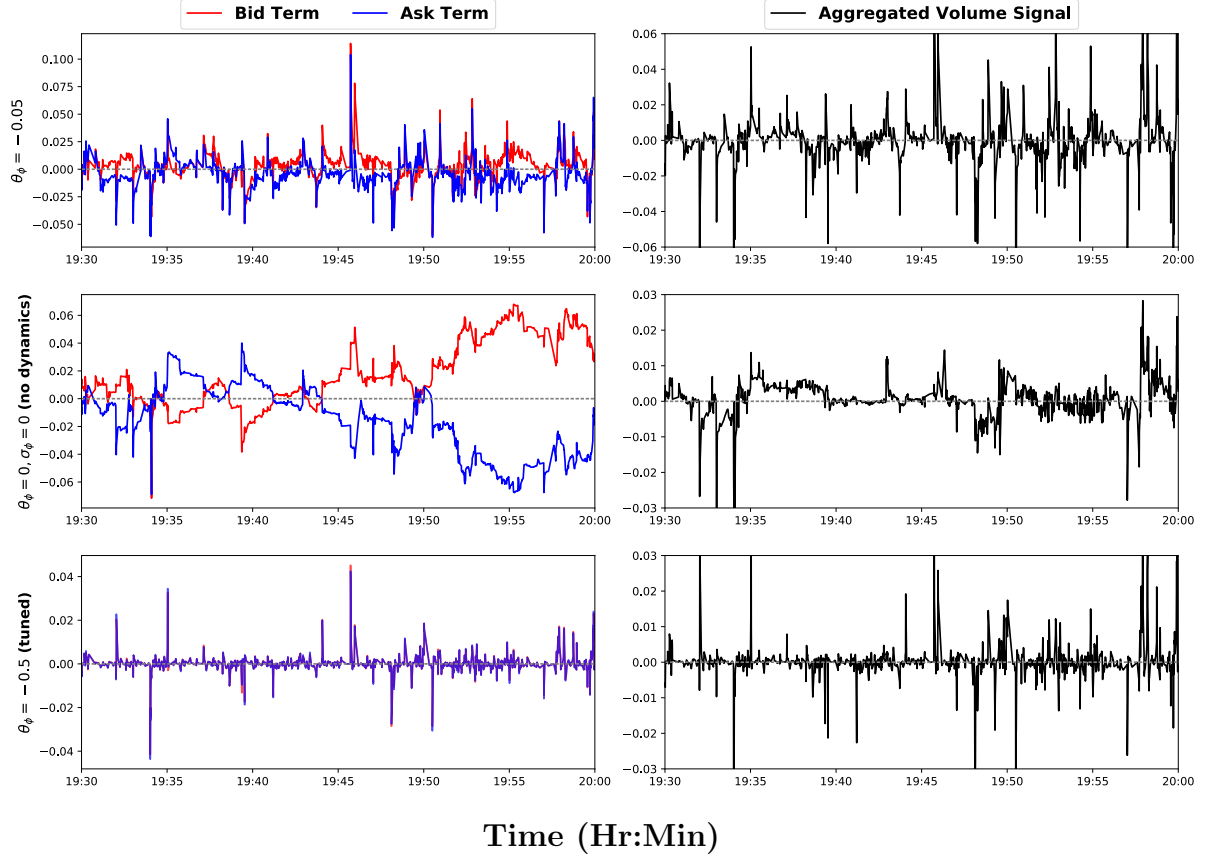


Figure 3.4.6: Figure shows the posterior means of volumetric variables $\{\phi_{A,t}, \phi_{B,t}\}$ and the aggregated volume signal under different settings of hyperparameters.

the tuned fast reverting dynamics in **Table 3.4.1** i.e. $\theta_\phi = -0.5$. It can be observed from **Figure 3.4.6** that stronger reverting force tends to close up the separation between $\phi_{A,t}$ and $\phi_{B,t}$ and remove the longer-term component within the aggregated signal. Without dynamics (middle plots), the volumetric variables can still adapt to the LOB inputs to some extent with occasional resets. However in such a case, it is hard for $\phi_{A,t}$ and $\phi_{B,t}$ to capture any high-frequency sharp changes in the trend that may be caused by LOB imbalance as shown in the aggregated signal plot.

3.5 Conclusions and future work

In this chapter, the baseline jump diffusion model [13] was firstly review, which is a part of the pioneer work in using the SSMS and the particle filter for financial price modelling. Based upon this work, this chapter has presented three novel models each incorporating an unique market intuition: (i) an imbalance-driven trend model that takes in dynamical changes in LOB structure and translates the imbalance into price innovations; (ii) an extended trend model which aims to capture trends at two different time scales; and (iii) a jump trend resetting model that allows better re-adaptation of state variables after

significant market changes (i.e. jumps).

The imbalance-driven trend model employs a novel construction of the SSM by including a set of time-varying parameters $\{\phi_{A,t}, \phi_{B,t}\}$ in the state vector to modulate the trend contribution taken from the instantaneous order volume imbalance at each observation of the market. This inclusion allows the two parameters to be sequentially tracked together with the price process in the generic inference algorithm. The extended trend model further adopts a varying-mean OU process on the original trend term to incorporate additional mean-reverting characteristics and consequently captures the price momentum at different time scales. The jump resetting mechanism resets the filtering mean and covariance of the state vector to prior values and enables faster adaptation to the new market condition after significant price changes. All three models, as well as their combinations (e.g. the full model), have maintained the linear-Gaussian structure of the SSM conditioned on the non-linear jump times allowing efficient sequential inference to be achieved using the well-established RBPF algorithm.

The experimental results obtained using real-world LOB data extracted from the FOREX market show that the proposed models/features improve both predication accuracy as well as model fit compared to the baseline jump diffusion model [13]. This is due to incorporating salient Bayesian priors on the LOB market behaviours via a SSM approach. Furthermore, qualitative analyses on the posteriors have also demonstrated that the new state variables are able to provide reasonable real-world market interpretations.

It is also important to note that the presented models in this chapter can be easily extended to achieve simultaneous fair price inference of multiple (dependent or independent) markets. The pairwise correlations (e.g. correlated volatility and/or jumps of two markets) of the inferred markets can be either encoded in the SSM construction using prior knowledge or learnt with optimisation algorithms.

However, as can be seen from the posterior tracking results in Section 3.4, the volumetric variables in the proposed model can sometimes take unusual values (e.g. long-lasting negative $\phi_{B,t}$ and positive $\phi_{A,t}$) that cannot be explained by market intuitions. Thus in future work, assigning constrained diffusion dynamics, such as a geometric Brownian motion, to the volumetric variables can potentially be an interesting extension to the current models. However, it would be challenging to fit the constrained diffusion model into the general RBPF framework.

Another goal of future work extending this research could be to examine the versatility of the proposed models, not just to FOREX markets, but also to other types of LOBs such as stocks and derivatives. It would also be desirable to implement a more comprehensive mock trading algorithm to backtest the models and the inference algorithms in more realistic scenarios, although this would make it necessary to consider more complex trading effects like slippage (execution delays). Studying (e.g. empirically) the sensitivity of the

proposed modeling and inference approach to variations in the model hyperparameters is another potential area of future investigation.

Chapter 4

Multi-jump diffusion process for long-short term price dynamics

The jump-diffusion process introduced in **Chapter 3** is a popular choice in the area of financial modelling to account for potential risks and uncertainties by generalising the unknown impulsive impacts in the market with a simple jump process. In this chapter, I propose an extension and generalisation of such a modelling approach, and then connect the jump-diffusion model with the heterogeneous arrivals/operations of limit orders in a high-frequency LOB market.

4.1 Introduction and motivation

The field of financial modelling pays immense amount of attention to the modelling of outliers. From a modelling perspective, the outliers indicate potential model mismatch and overfitting which can be detrimental to the performance of model predictions. In practice, a single observation that is considered to be statistically insignificant by the model may still be fatal to the investment decisions made by practitioners. An empirical motivation for using jump-diffusion model in finance is to gain better accountability for the occasional heavy-tailed behaviour of asset returns or commodity prices, which cannot be reliably captured by a conventional diffusion model.

Jump-diffusion processes have been discussed in multiple studies since their introduction to finance in [90] as an improvement on the classic Black-Scholes option pricing formulae (reviewed in [91]). A Gaussian/normally distributed jump size was considered in the first iteration of the model. This distribution of the jump size was later extended in [92] and [93] to a double-exponential distribution and an asymmetric Laplace distribution respectively, which were empirically shown to have better accountability for heavier-tailed data. Jump processes were also integrated with stochastic-volatility models to provide a general family of affine jump-diffusion models [94]. The author of [95] generalised the

one-dimensional jump-diffusion process to a multivariate case where both individual and “common” jumps can occur on multiple assets. These early models often pursued the closed-form statistical properties of jump-diffusion process that can subsequently simplify the applied estimation/prediction algorithm. With the development of particle filter (PF) as SMC inference procedure, more diverse dynamics are incorporated in the jump-diffusion model for more realistic modelling of the financial market [96, 13].

Another motivation of the jump-diffusion model comes from behavioral finance. It has been well-studied that the market tends to respond actively towards the news outside the market such as the changes of macroeconomic policies, etc.. With the help of Bayesian modelling and posterior inference, the posterior jump (both times and size) estimation of the model can also provide important retrospective information on market responses to various known or unknown external sources. This idea was adopted in [13] and the price models introduced in **Chapter 3**.

In a high-frequency financial market, the price dynamics are studied at a much finer time-scale. While jump processes can still be incorporated to account for unknown disturbances from outside the market, such occurrences can be rare compared to the jumps or trends that are initiated within the market. On the other hand, large institutional orders which create significant buying/selling pressure are more frequently placed by market makers to drive the price process. Thus in this research, I connect the stochastic jump process on a continuous time-grid with the heterogeneous submissions of (institutional) limit orders. However, unlike the impulsive market responses to outside factors which are often generalised with a single jump process, institutional orders are likely to induce jumps with different scales, biases, frequencies and residual effects depending on the levels, quantities and duration of the limit orders placed. It is clear that the classic jump-diffusion process can no longer capture these specific characteristics of the high-frequency LOB market.

Therefore in this chapter, I consider jumps as a major source of price innovation and propose a novel state-space model which incorporates multiple jump processes to account for the part of market dynamics initiated by injection or cancellation of limit orders at different levels of the LOB. The model also aims to capture the long-short term trends of the price dynamics with the multi-jump construction. For inference, a semi-deterministic particle filter with stratified resampling is proposed to allow more thorough exploration of the continuous sample space of the latent variables.

The remainder of this chapter is organised as follows. I first introduce the general formulation of the multi-jump diffusion model in Section 4.2. Section 4.3 briefly recaps the standard RBPF algorithm [13] introduced in **Chapter 3** before focusing on its adaptation to jump proposals with memory and the semi-deterministic filtering scheme. Results obtained on both synthetic and real datasets are presented in Section 4.4.

4.2 Model formulation

Similar to the classic jump-diffusion model, the proposed multi-jump diffusion model can also be disaggregated into two connected parts: (1) a constant-volatility mean-reverting process; and (2) a jump-driven process. To begin with, first consider the following dynamics for the (latent) market fair price process $x(t)$:

$$dx(t) = \theta[x(t) - z(t)] dt + \sigma_v dW_t \quad (4.1)$$

where σ_v is the scale of innovation noise which represents the constant volatility in the market; θ is the non-positive mean-reversion coefficient; and dW_t is a unit-variance Wiener process. When the function $z(t)$ takes a constant value, the above dynamics is simply an Ornstein–Uhlenbeck (OU) process [85].

It can be seen from Eq. (4.1) the fair price process $x(t)$ will stochastically adapt to the “guiding” function $z(t)$ with a rate depending on the value of θ . Such a formulation allows the clear (i.e. volatility-free) market/price signal $z(t)$ to be explicitly modelled by a different stochastic process. The fair price $x(t)$ is constructed such that it is subject to the inherent volatility of the market and reflects a delayed and cluttered version of the clear signal. The value of θ thus determines the extent to which the fair price process follows the volatility-free signal. In the case of $\theta = 0$, $x(t)$ simply follows a Brownian diffusion model taking no input from the jumps induced by the changes of LOB structure; while a negative θ of large magnitude will make $x(t)$ follow $z(t)$ closely (or exactly as $\theta \rightarrow -\infty$ and $\sigma_v = 0$).

4.2.1 Multi-jump reversion process

Due to the various forms that $z(t)$ can take, a closed-form solution to the stochastic differential equation (SDE) in (4.1) is not guaranteed in general. In order to ensure the tractability of the SDEs while allowing salient features of the market to be described by the process $z(t)$, the following SSM is constructed:

$$dz_t = \mathbf{A}_z \mathbf{z}_t dt + \sum_{k=1}^K \mathbf{c}_k dJ_t^k \quad (4.2)$$

where:

$$\mathbf{z}_t = \begin{bmatrix} z_{1,t} \\ z_{2,t} \\ \vdots \\ z_{K+1,t} \end{bmatrix}, \quad \mathbf{A}_z = \begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & \theta_1 & 0 & \cdots & 0 \\ \vdots & 0 & \theta_2 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \theta_K \end{bmatrix} \quad (4.3)$$

where $z_{1,t}$ is exactly the signal $z(t)$ in (4.1) that “guides” $x(t)$; $\{\theta_k\}_{k=1}^K$ are the non-positive mean-reverting coefficients; and dJ_t^k is the k th jump process. The scaling vector \mathbf{c}_k for each jump process is constructed as follows:

$$\mathbf{c}_k^i = \begin{cases} 1 & \text{if } i = k + 1 \\ 0 & \text{otherwise} \end{cases}, \quad \text{e.g. } \mathbf{c}_k = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow (k+1)^{\text{th}} \text{ element} \quad (4.4)$$

where the superscript i here indicates the i th element of the vector. The design of matrix \mathbf{A}_z is based on the intuition that large changes in LOB structure (i.e. submission or cancellation) often create momentum for the shift of market price; while this momentum fades gradually with the depletion and fill-in of limit orders as the market’s reaction. In order to model the characteristic responses to different levels of LOB changes, $\{z_{j,t}\}_{j=2}^{K+1}$ are each governed by an independent jump reversion process with a distinct set of parameters. This use of multiple jump reversion processes provides an aggregated momentum/trend of $z_{1,t}$ with multiple layers of time structures:

$$\frac{dz_{1,t}}{dt} = \sum_{j=2}^{K+1} z_{j,t} \quad (4.5)$$

Although it is not immediately clear on how this multi-jump SSM can be integrated in the constant-volatility dynamics of the fair price x_t to give a tractable solution, it is possible to concatenate \mathbf{z}_t and x_t into a single state vector denoted as \mathbf{x}_t and construct a joint SSM:

$$d\mathbf{x}_t = \mathbf{A} \mathbf{x}_t dt + \mathbf{b} dW_t + \sum_{k=1}^K \tilde{\mathbf{c}}_k dJ_t^k \quad (4.6)$$

where:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{z}_t \\ x_t \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_z & \vdots & 0 \\ -\theta & 0 & \dots & \theta \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \sigma_v \end{bmatrix}, \quad \tilde{\mathbf{c}}_k = \begin{bmatrix} \mathbf{c}_k \\ 0 \end{bmatrix} \quad (4.7)$$

This resembles the similar general SDE for jump-diffusion process described in Eq. (3.3) of **Chapter 3**, except that now there are a total of K jump processes. Using the same derivations in **Appendix 2.B** and Section 3.2, the SDE can be readily solved in closed-form conditioned on the jumps in the K jump processes. The jump size also follows a

normal distribution with parameters determined by the specific process:

$$\mathcal{J}_i^k \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(\mathcal{J}_i^k | \mu_k, \sigma_k^2) \quad (4.8)$$

where \mathcal{J}_i^k denotes the i th jump size for the k th jump process. Given the interval $(S, T]$ between two arbitrary timestamps $S \leq T$ and further denoting the jump times (from all K processes) in this interval as $\{\tau_i^k\}_{k,i}$, where $k \in \{1, \dots, K\}$ is the jump type and i is the index of jumps of type k , the conditional transition density of the state vector \mathbf{x}_T can be written as:

$$p(\mathbf{x}_T | \mathbf{x}_S, \{\tau_i^k\}_{k,i}) = \mathcal{N}\left\{\mathbf{x}_T | \boldsymbol{\mu}(\mathbf{x}_S, T - S, \{\tau_i^k\}_{k,i}), \mathbf{C}(T - S, \{\tau_i^k\}_{k,i})\right\} \quad (4.9)$$

The subscripts (k, i) represent all possible combinations for $k \in \mathbb{N}^+, i \in \mathbb{N}^+$ to index a jump in the interval $(S, T]$. The mean and covariance for the above density are:

$$\boldsymbol{\mu}(\mathbf{x}_S, T - S, \{\tau_i^k\}_{k,i}) = e^{\mathbf{A}(T-S)} \mathbf{x}_S + \sum_{k,i} e^{\mathbf{A}(T-\tau_i^k)} \tilde{\mathbf{c}}_k \mu_k \quad (4.10)$$

$$\mathbf{C}(T - S, \{\tau_i^k\}_{k,i}) = e^{\mathbf{A}(T-S)} \mathbf{K}(T, S) (e^{\mathbf{A}(T-S)})' + \sum_{k,i} \sigma_k^2 e^{\mathbf{A}(T-\tau_i^k)} \tilde{\mathbf{c}}_k \tilde{\mathbf{c}}_k' (e^{\mathbf{A}(T-\tau_i^k)})' \quad (4.11)$$

The expression for $\mathbf{K}(T, S)$ is the same as that in the standard jump-diffusion model:

$$\mathbf{K}(T, S) = \int_{t=0}^{T-S} e^{-\mathbf{A}t} \mathbf{b} \mathbf{b}' (e^{-\mathbf{A}t})' dt$$

Many of the jump-diffusion models use the Poisson process to model the jump times because of its convenient memoryless property. However in real-world markets, the jumps (i.e. injection or cancellation of large limit orders) are rarely memoryless and the future occurrences are likely to depend on past history. Therefore, I propose the use of a shifted Gamma distribution to model the interval between jumps instead of an exponential distribution:

$$\begin{aligned} p(\tau_i^k | \{\tau_j^k\}_{j < i}) &= p(\tau_i^k | \tau_{i-1}^k) \\ \implies \tau_i^k - \tau_{i-1}^k - d_k &\stackrel{\text{i.i.d.}}{\sim} \text{Gamma}(\alpha_k, \beta_k) \end{aligned} \quad (4.12)$$

where d_k is the non-negative shift parameter for the k th jump process; α_k and β_k are the shape and rate parameters of the Gamma distribution respectively. The shift parameter is used here to ensure a minimum duration d_k of “cool-down” time before the next jump takes place i.e. $p([\tau_i^k - \tau_{i-1}^k] < d_k) = 0$. Institutional orders are introduced to the market often as an incentive for trend or to push the price in a desired direction. These orders of large volume are unlikely to be repeatedly injected within a short time-frame. Hence, a shifted Gamma distribution with the memory of the last jump gives an intuitive representation of the jumps in a real-world LOB market.

Finally, define the observational model for an observed market mid-price y_t as the fair price x_t with additive Gaussian observation noise:

$$y_t = x_t + \epsilon_t \quad , \quad \epsilon_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{\text{obs}}^2) \quad (4.13)$$

$$p(y_t | \mathbf{x}_t) = \mathcal{N}(y_t | \mathbf{G} \mathbf{x}_t, \sigma_{\text{obs}}^2) \quad (4.14)$$

where $\mathbf{G} = [0, \dots, 0, 1]$. For simplicity of the expressions, let us allow the notations y_n and \mathbf{x}_n to denote the observed mid-price and latent state vector at time t_n respectively. Furthermore, denote all jump times in interval $(t_n, t_m]$ from the k th jump process as $\{\tau\}_{n:m}^k$. Given a set of N mid-price observations $Y = \{y_n\}_{n=1}^N$ at timestamps $\{t_n\}_{n=1}^N$ and an initialisation of \mathbf{x}_0 at t_0 , the joint model probability can thus be written as:

$$\begin{aligned} & p(Y, \{\mathbf{x}_n\}_{n=1}^N, \{\tau\}_{0:N}^{1:K} | \mathbf{x}_0, \Omega) \\ &= \prod_{n=1}^N \left\{ \prod_{k=1}^K [p(\{\tau\}_{n-1:n}^k | \{\tau\}_{0:n-1}^k)] p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}^{1:K}) p(y_n | \mathbf{x}_n) \right\} \end{aligned} \quad (4.15)$$

where Ω represents all hyperparameters of the model.

4.3 Inference

Clearly, the proposed multi-jump diffusion model shares a similar model structure to the jump-diffusion models reviewed and proposed in the previous chapter. This means that the same general framework of the Rao-Blackwellised particle filter (RBPF) can also be adopted to achieve sequential inference for this model.

As the multi-jump diffusion SSM has the same linear-Gaussian form of the transition density (4.9) and observational likelihood (4.14) as the standard jump-diffusion model, the same derivation of the conditional Kalman filter described in Section 3.3 can be applied to obtain the recursive update formulae for the predictive distribution of the state vector \mathbf{x}_n :

$$p(\mathbf{x}_n | y_{1:n-1}, \{\tau\}_{0:n}^{1:K}) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:n-1}, \boldsymbol{\Sigma}_{n|0:n-1}) \quad (4.16)$$

with predictive parameters:

$$\boldsymbol{\mu}_{n|0:n-1} = e^{\mathbf{A}(t_n - t_{n-1})} \boldsymbol{\mu}_{n-1|0:n-1} + \sum_{\{k, i | \tau_i^k \in \{\tau\}_{n-1:n}^{1:K}\}} e^{\mathbf{A}(t_n - \tau_i^k)} \tilde{\mathbf{c}}_k \mu_k \quad (4.17)$$

$$\boldsymbol{\Sigma}_{n|0:n-1} = e^{\mathbf{A}(t_n - t_{n-1})} \boldsymbol{\Sigma}_{n-1|0:n-1} e^{\mathbf{A}^T(t_n - t_{n-1})} + \mathbf{C}(t_n, t_{n-1}, \{\tau\}_{n-1:n}^{1:K}) \quad (4.18)$$

The posterior filtering density is subsequently obtained after the correction by the observation y_n :

$$p(\mathbf{x}_n | y_{1:n}, \{\tau\}_{0:n}^{1:K}) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:n}, \boldsymbol{\Sigma}_{n|0:n}) \quad (4.19)$$

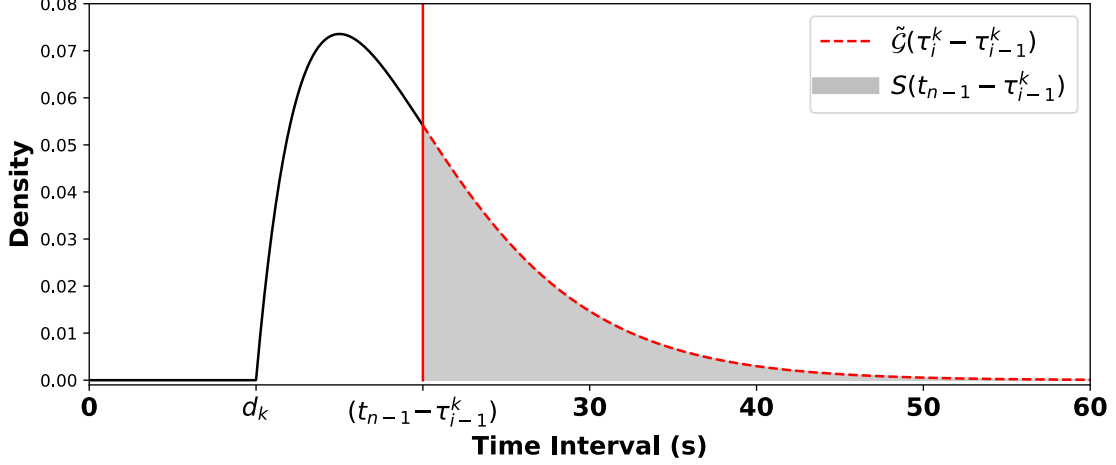


Figure 4.3.1: The PDF of a shifted Gamma distribution with shift parameter d_k . The red dashed curve is the unnormalised density of jump interval that gives a jump time $\tau_i^k \geq t_{n-1}$; while the area of the shaded region is the survival function value.

with the posterior parameters obtained with the same Kalman update formulae:

$$\mathbf{Q}_n = \Sigma_{n|0:n-1} \mathbf{G}^T \left(\mathbf{G} \Sigma_{n|0:n-1} \mathbf{G}^T + \sigma_{\text{obs}}^2 \right)^{-1} \quad (4.20)$$

$$\boldsymbol{\mu}_{n|0:n} = \boldsymbol{\mu}_{n|0:n-1} + \mathbf{Q}_n \left(y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1} \right) \quad (4.21)$$

$$\Sigma_{n|0:n} = \left(\mathbf{I} - \mathbf{Q}_n \mathbf{G} \right) \Sigma_{n|0:n-1} \quad (4.22)$$

The initialisation of $\boldsymbol{\mu}_{0|0}$ and $\Sigma_{0|0}$ can be readily set to some pre-determined values without influencing the overall performance of the RBPF algorithm.

4.3.1 *Jump proposals and semi-deterministic particle filtering*

The distinct feature of the proposal model lies in the use of multiple jump processes. From the inference perspective, the change is not only in the number of latent processes but also in the jump dynamics which is no longer memoryless. The former increases the dimension of latent sample space and imposes a bigger challenge in handling the inherent particle degeneracy problem; while the latter requires an alternative design for the jump proposals. In this subsection, I present solutions to these two problems within the variable-rate particle filter (VRPF) part of the RBPF algorithm.

Inverse-CDF jump proposal

Unlike the memoryless Poisson process where jump intervals can simply be proposed from an (unconditional) exponential distribution, the jump intervals in the proposed model follow a shifted Gamma distribution. Due to the asynchronous arrivals of the observations (i.e. update timestamps) and the jump times, the jumps proposed at time t_n in the VRPF should be later than the timestamp of the previous observation t_{n-1} . This results in a

conditional (bootstrap) jump proposal from the prior (4.12):

$$\begin{aligned}
p(\tau_i^k | \tau_{i-1}^k, t_{n-1}) &\Rightarrow p(\tau_i^k - \tau_{i-1}^k | \tau_i^k > t_{n-1}) \\
&= p(\tau_i^k - \tau_{i-1}^k | [\tau_i^k - \tau_{i-1}^k] > [t_{n-1} - \tau_{i-1}^k]) \\
&= \frac{p(\tau_i^k - \tau_{i-1}^k \cap [\tau_i^k - \tau_{i-1}^k] > [t_{n-1} - \tau_{i-1}^k])}{p([\tau_i^k - \tau_{i-1}^k] > [t_{n-1} - \tau_{i-1}^k])} \\
&= \frac{\tilde{\mathcal{G}}(\tau_i^k - \tau_{i-1}^k | d_k, \alpha_k, \beta_k, [t_{n-1} - \tau_{i-1}^k])}{S(t_{n-1} - \tau_{i-1}^k)}
\end{aligned} \tag{4.23}$$

where $\tilde{\mathcal{G}}(\cdot)$ is the unnormalised density function of the jump interval which gives a $\tau_i^k > t_{n-1}$; and $S(\cdot)$ is the survival function of the shifted Gamma distribution. The relationship between these two functions and the original shifted Gamma distribution for jump interval is shown in **Figure 4.3.1**. Clearly, the resultant conditional probability for the jump interval is a normalised density function which can be sampled analytically using the inverse-CDF (cumulative density function) method. Denoting $\Delta\tau_i^k = \tau_i^k - \tau_{i-1}^k$, the CDF is: (4.23) as:

$$\begin{aligned}
F(T | \tau_i^k > t_{n-1}) &= \int_{\Delta\tau_i^k=0}^T \frac{\tilde{\mathcal{G}}(\Delta\tau_i^k | d_k, \alpha_k, \beta_k, [t_{n-1} - \tau_{i-1}^k])}{S(t_{n-1} - \tau_{i-1}^k)} d\Delta\tau_i^k \\
&= \begin{cases} 0 & \text{if } T \leq (t_{n-1} - \tau_{i-1}^k) \\ \frac{F(T) - [1 - S(t_{n-1} - \tau_{i-1}^k)]}{S(t_{n-1} - \tau_{i-1}^k)} & \text{otherwise} \end{cases}
\end{aligned} \tag{4.24}$$

where $F(T)$ is the CDF for the (unconditional) shifted Gamma distribution defined in (4.12); the inverse-CDF sampling thus proceeds as follows:

1. Sample $u \sim \text{Uniform}(0, 1)$;
2. Compute $u' = (u - 1) \times S(t_{n-1} - \tau_{i-1}^k) + 1$
3. Obtain the sample $\Delta\tau_i^k = F^{-1}(u')$

It can be easily verified that $\Delta\tau_i^k$ sampled from this algorithm always guarantees a τ_i^k greater than t_{n-1} . And in the case where $\tau_{i-1}^k \geq t_{n-1}$ and consequently $S(t_{n-1} - \tau_{i-1}^k) = 1$, the jump interval will be automatically sampled from the unconditional shifted Gamma distribution, i.e. $u' = u \sim \text{Uniform}(0, 1)$.

Now with jump proposals sorted, the RBPF can be carried out with almost the same procedure as that for the jump-diffusion models introduced in **Chapter 3**. However as the model incorporates K independent jump processes, each particle in the algorithm should also contain the jump-time sequences of the K jump processes during the sequential propagation. Denote the content of a particle p at time t_n as:

$$P_n^{(p)} = \left\{ (\{\tau\}_{0:n}^{1:K})^p, \boldsymbol{\mu}_{n|0:n}^p, \boldsymbol{\Sigma}_{n|0:n}^p, w_n^p \right\} \tag{4.25}$$

where $\boldsymbol{\mu}_{n|0:n}^p, \boldsymbol{\Sigma}_{n|0:n}^p$ are the posterior filtering parameters obtained from the Kalman filter conditioned on the jump sequences $(\{\tau\}_{0:n}^{1:K})^p$; and w_n^p is the normalised particle weight. A standard RBPF algorithm for the proposed multi-jump diffusion model can be adapted from **Algorithm 7** in **Chapter 3** with a slight change in the jump proposals.

Semi-deterministic particle filtering

Although Rao-Blackwellisation is adopted in PF to ease the curse of dimensionality arising from the state vectors, it remains important to handle the high-dimensional sample-space of the jump times which are proposed in the VRPF algorithm. Here, I propose a scheme of semi-deterministic particle filtering inspired by the deterministic particle filter introduced in [21].

Simply reviewing the conditional prior jump proposals (4.23) for VRPF algorithm, it can be noticed that the proposed jump times τ_i^k can exceed the timestamps of the current update step t_n . And based on the definition of the VRPF, any jumps (i.e. particles) proposed outside the update interval $(t_{n-1}, t_n]$ will be omitted and considered as no-jump for the current propagation. This means that the effective number of particles used to explore the continuous spectrum/sample space of the jump times can be significantly lower than the actual particle number N_p specified in the VRPF algorithm. This lack of exploration can lead to a number of undesirable outcomes including poor inference accuracy and particle weight degeneracy.

The deterministic PF was originally applied to a hidden Markov model (HMM) in [21] to allow online inference for the hidden state. A salient difference between the HMM and the SSM is that the hidden state in the HMM is discretely distributed with a finite support. The deterministic PF thus performs thorough exploration of all possible descendants (i.e. hidden states) of each ancestor particle from the previous timestamp, which gives a significant improvement in inference accuracy at the cost of extra computation. A novel variant of the deterministic PF is employed in **Chapter 5**, in which the algorithm is more thoroughly studied. Here, let us focus on the adaptation of the generic algorithm to further accommodate the continuous latent jumps in the proposed model.

Clearly, it is impossible to apply the deterministic PF directly to explore the sample space of jump times as it has an infinite support. Assuming for now that the model has only one jump process (i.e. $K = 1$), a semi-deterministic approach is developed with the following steps at stage t_n of the VRPF:

1. For each particle $p = 1, \dots, N_p$, deterministically consider both possibilities of **(i)** no jump within $(t_{n-1}, t_n]$; and **(ii)** having at least 1 jump within $(t_{n-1}, t_n]$.
2. Compute the jump probability $q_J^{k,p}$ (of particle p) given the time of the last jump

Algorithm 8 Semi-deterministic RBPF

Input: Time-stamps $\{t_n\}_{n=1}^N$; observation $\{y_n\}_{n=1}^N$; K -digits binary numbers encoding the jump combinations $\{B_j\}_{j=1}^{2^K}$.

Output: A collection of weighted particles $\{P_N^{(p)}\}_{p=1}^{N_p}$

- 1: **Initialisation:** ($n=0$) Create initial particle collection $P_0^{(p)} = \{\emptyset, \boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\Sigma}_{\text{prior}}, \frac{1}{N_p}\}$ for $p = 1, \dots, N_p$
- 2: **for** $n = 1, \dots, N$ **do**
- 3: **for** $p = 1, \dots, N_p$ **do**
- 4: Compute predictive parameters $\hat{\boldsymbol{\mu}}_{n|0:n-1}^p, \hat{\boldsymbol{\Sigma}}_{n|0:n-1}^p$ conditioned on no jump.
- 5: Pre-set all predictive parameters $\{\boldsymbol{\mu}_{n|0:n-1}^{p,j}\}_{j=1}^{2^K}, \{\boldsymbol{\Sigma}_{n|0:n-1}^{p,j}\}_{j=1}^{2^K}$ to $\hat{\boldsymbol{\mu}}_{n|0:n-1}^p, \hat{\boldsymbol{\Sigma}}_{n|0:n-1}^p$.
- 6: Pre-set all (unnormalised) weights $\{\tilde{w}_n^{p,j}\}_{j=1}^{2^K}$ to w_{n-1}^p .
- 7: Pre-set all jump times $(\{\tau\}_{0:n}^{1:K})^{p,j}$ to $(\{\tau\}_{0:n-1}^{1:K})^p$
- 8: **for** $k = 1, \dots, K$ **do**
- 9: Propose jumps $\{\tau\}_{n-1:n}^{k,p}$ from the conditional prior (4.28) of the k th jump process.
- 10: Compute probability $q_J^{k,p}$ using (4.26)
- 11: **for** $j = 1, \dots, 2^K$ **do**
- 12: **if** (the k th digit $B_j^k = 1$) **then**
- 13: $\boldsymbol{\mu}_{n|0:n-1}^{p,j} = \boldsymbol{\mu}_{n|0:n-1}^{p,j} + \sum_{\tau_i \in \{\tau\}_{n-1:n}^{k,p}} e^{\mathbf{A}(t_n - \tau_i)} \tilde{\mathbf{c}}_k \mu_k$
- 14: $\boldsymbol{\Sigma}_{n|0:n-1}^{p,j} = \boldsymbol{\Sigma}_{n|0:n-1}^{p,j} + \sum_{\tau_i \in \{\tau\}_{n-1:n}^{k,p}} \sigma_k^2 e^{\mathbf{A}(t_n - \tau_i)} \tilde{\mathbf{c}}_k \tilde{\mathbf{c}}_k' (e^{\mathbf{A}(t_n - \tau_i)})'$
- 15: $\tilde{w}_n^{p,j} = \tilde{w}_n^{p,j} \times q_J^{k,p}$
- 16: Append proposed jump time(s): $(\{\tau\}_{0:n}^{k,p})^{p,j} \leftarrow (\{\tau\}_{0:n-1}^{k,p})^{p,j} \cup \{\tau\}_{n-1:n}^{k,p}$
- 17: **else**
- 18: $\tilde{w}_n^{p,j} = \tilde{w}_n^{p,j} \times (1 - q_J^{k,p})$
- 19: **end if**
- 20: **end for**
- 21: **end for**
- 22: Update the weights with Kalman PED: $\tilde{w}_n^{p,j} = \tilde{w}_n^{p,j} \times \text{PED}(y_n | \boldsymbol{\mu}_{n|0:n-1}^{p,j}, \boldsymbol{\Sigma}_{n|0:n-1}^{p,j})$, for all $j = 1, \dots, 2^K$
- 23: **end for**
- 24: Normalise weights $w_n^{p,j} = \tilde{w}_n^{p,j} / (\sum_p \sum_j \tilde{w}_n^{p,j})$
- 25: **Resample** N_p particles $\{P_n^{(p)}\}_{p=1}^{N_p}$ from the total $2^K \times N_p$ particles.
- 26: Kalman update posterior parameters $\{\boldsymbol{\mu}_{n|0:n}^p, \boldsymbol{\Sigma}_{n|0:n}^p \mid y_n\}$ from Eq. (4.21) and (4.22)
- 27: (Re-normalise weights $\{w_n^p\}_{p=1}^{N_p}$ if necessary)
- 28: **end for**
- 29: **Return:** $\{P_N^{(p)}\}_{p=1}^{N_p}$

$$\tau_{i-1}^{k,p}:$$

$$q_J^{k,p} = \Pr\{\tau_i^{k,p} \in (t_{n-1}, t_n] \mid \tau_{i-1}^{k,p}\} = 1 - \frac{S(t_n - \tau_{i-1}^{k,p})}{S(t_{n-1} - \tau_{i-1}^{k,p})} \quad (4.26)$$

where $S(\cdot)$ is the survival function of the prior density.

3. For **(i)**, propagate the state vector conditioned on no jump (i.e. standard SSM diffusion). For **(ii)**, propose jump time(s) from an augmented conditional prior $p(\tau_i^k \mid \tau_{i-1}^{k,p}, \tau_i^{k,p} \in (t_{n-1}, t_n])$; and propagate the state vector conditioned on the proposed jump time(s).
4. Evaluate the (unnormalised) weights for both scenarios: (i) \tilde{w}_n^p (no jump) = $w_{n-1}^p \times \text{PED}(y_n) \times (1 - q_J^{k,p})$; (ii) \tilde{w}_n^p (with jump) = $w_{n-1}^p \times \text{PED}(y_n) \times q_J^{k,p}$.
5. Normalise weights and resample N_p particles out of the total $2 N_p$ particles for the next propagation step.

where PED is the *prediction error decomposition* obtained from the conditional Kalman filter [88] with its formula derived in (3.32). In the $K = 1$ deterministic filtering case, for each of the **2** (2^K) descendants of particle p , its PED can be expressed as:

$$\begin{aligned} & \text{PED}(y_n \mid \boldsymbol{\mu}_{n|0:n-1}^{p,j}, \boldsymbol{\Sigma}_{n|0:n-1}^{p,j}) \\ &= p(y_n \mid y_{1:n-1}, (\{\tau\}_{0:n}^{1:K})^{p,j}) \\ &= \mathcal{N}(y_n \mid \mathbf{G} \boldsymbol{\mu}_{n|0:n-1}^{p,j}, \mathbf{G} \boldsymbol{\Sigma}_{n|0:n-1}^{p,j} \mathbf{G}^T + \sigma_{\text{obs}}^2) \end{aligned} \quad (4.27)$$

where j is the descendant index; and $\boldsymbol{\mu}_{n|0:n-1}^{p,j}$ and $\boldsymbol{\Sigma}_{n|0:n-1}^{p,j}$ are the predictive parameters computed from Eq. (4.17) and (4.18) conditioned on jump(s) or no jump. Such a filtering scheme deterministically explores the existence of jumps within the update interval, conditioned on which the jump particles are randomly sampled and re-weighted. This enables effective usage of particles by proposing jumps that only occur between t_{n-1} and t_n , with proposal density:

$$\begin{aligned} p(\Delta \tau_i^k \mid t_{n-1} < \tau_i^k \leq t_n) &= \frac{p(\Delta \tau_i^k \cap t_{n-1} < \tau_i^k \leq t_n)}{S(t_{n-1} - \tau_{i-1}^k) - S(t_n - \tau_{i-1}^k)} \\ &= \frac{\tilde{\mathcal{G}}(\Delta \tau_i^k \mid d_k, \alpha_k, \beta_k, [t_{n-1} - \tau_{i-1}^k], [t_n - \tau_{i-1}^k])}{S(t_{n-1} - \tau_{i-1}^k) - S(t_n - \tau_{i-1}^k)} \end{aligned} \quad (4.28)$$

where samples can also be analytically obtained using the inverse-CDF method. However, it is important to monitor the value of u' in this case because it is possible that there is no valid sample to draw from this conditional prior i.e. $(t_{n-1} - \tau_{i-1}^k) < (t_n - \tau_{i-1}^k) \leq d_k$ and $S(t_{n-1} - \tau_{i-1}^k) - S(t_n - \tau_{i-1}^k) = 0$. It is also worth noting that with a single jump process, the superscript k is redundant as it only takes one value and there are only two deterministic branches for each “ancestral” particle. While in the case where $K > 1$,

the superscript is then used to indicate the values and random variables of the k th jump process. Furthermore with $K > 1$, the number of deterministic branches that require exploration will be increased to 2^K combinations of the binary states of the K jump processes. This can be expressed with the K -digits binary numbers, for example:

00...00	No jump in any process
00...01	Jump(s) in the 1st process
00...10	Jump(s) in the 2nd process
\vdots	\vdots
11...10	Jump(s) in all processes but the 1st
11...11	Jump(s) in all processes

This allows the combinations of different jumps to be thoroughly explored when propagating the state, which can subsequently improve the inference accuracy. However, the limitation of this semi-deterministic filtering scheme is that it requires exponentially growing computation with the increasing number of jump processes. Therefore, the scheme is only practical for a relatively small K . **Algorithm 8** demonstrates the pseudo-code of a semi-deterministic RBPF for the proposed multi-jump diffusion model with an arbitrary K number of jump processes. The algorithm utilises the linear structures of the predictive mean and covariance, which means that certain “for” loops can be vectorised to provide more efficient computation.

Near the end of each propagation, the semi-deterministic filtering scheme requires the resampling of $2^K \times N_p$ particles to produce N_p particles as the posterior collection for the next propagation step. While this can be achieved with a simple importance resampler, the authors of [21] also proposed an “optimal” stratified resampling scheme which has shown improved performance on HMM. The details of this resampler is introduced and discussed in Appendix 5.A and Section 5.4.4 of this thesis respectively. In the next section, a comparison between the performance of the two resamplers on a synthetic dataset is presented.

4.4 Results and discussions

In this section, I present experimental results obtained with the proposed multi-jump diffusion model. First, the model’s behaviour is examined by simulating the process with a fixed set of hyperparameters and the inference performance of different RBPF configurations is evaluated on this synthetic dataset. Furthermore, the proposed model is applied to a set of FOREX mid-price data and the model’s interpretability on a real-world

Table 4.4.1: Simulation hyperparameters

Process	Hyperparameters
$z_{2,t}$	$\alpha_1 = 60.0$, $\beta_1 = 1.0$, $d_1 = 1.5$, $\theta_1 = -0.05$, $\mu_1 = 0$, $\sigma_1 = 0.5$
$z_{3,t}$	$\alpha_2 = 30.0$, $\beta_2 = 1.0$, $d_2 = 1.5$, $\theta_2 = -0.3$, $\mu_2 = 0$, $\sigma_2 = 1.0$
$z_{4,t}$	$\alpha_3 = 10.0$, $\beta_3 = 1.0$, $d_3 = 10.0$, $\theta_3 = -0.9$, $\mu_3 = 0$, $\sigma_3 = 2.5$
x_t	$\theta = -0.5$, $\sigma_v = 0.5$, $\sigma_{\text{obs}} = 0.05$

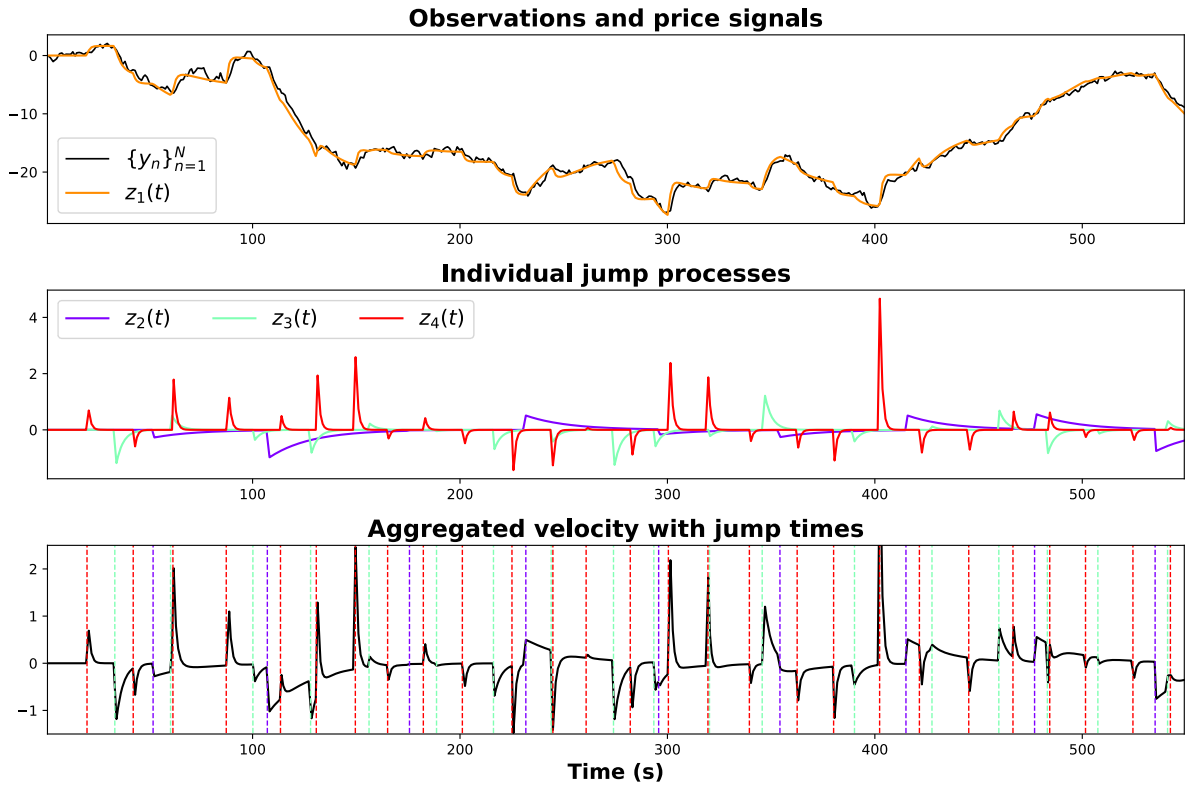


Figure 4.4.1: The simulated multi-jump diffusion process using pre-assigned hyperparameters. The top panel shows the synthetic observations and the volatility-free signal $z_{1,t}$. The middle panel shows the individual jump processes. While the bottom panel shows the aggregated velocity ($z_{2,t} + z_{3,t} + z_{4,t}$) with true jump times indicated in dashed lines (same colour codes for jumps)

financial market is studied.

4.4.1 *Simulation and inference*

The multi-jump diffusion model is designed to capture the trends in price dynamics at different time-scales. Therefore, I perform the first experiment by simulated the price process from the proposed model and study its behaviour in relation to the price dynam-

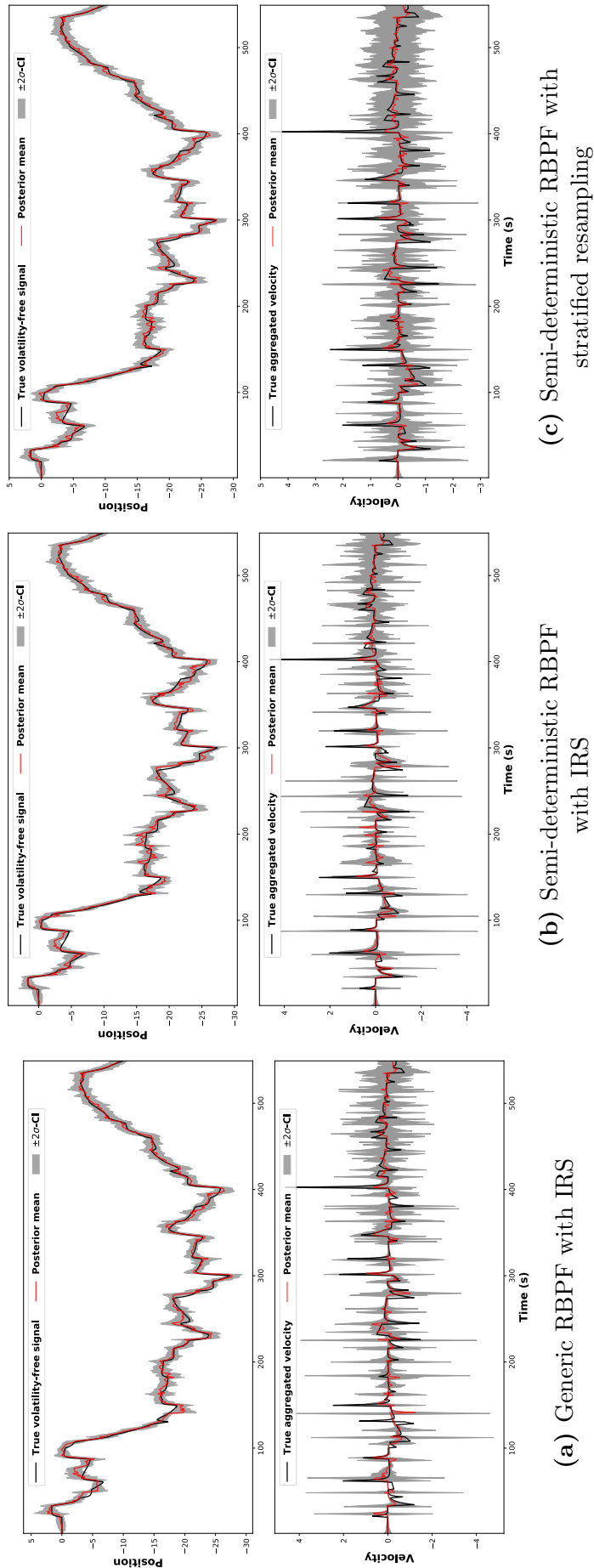


Figure 4.4.2: The posterior inference results obtained using different configurations of the RBPF algorithm. The top panels show the posterior means (in red) of the clear signal process $z_{1,t}$ and their corresponding 95%-CI's (in grey shaded) in comparison to the truth (in black). The bottom panels show the posterior means (in red) and 95%-CI's (in grey shaded) of the aggregated velocity in comparison to the truth (in black).

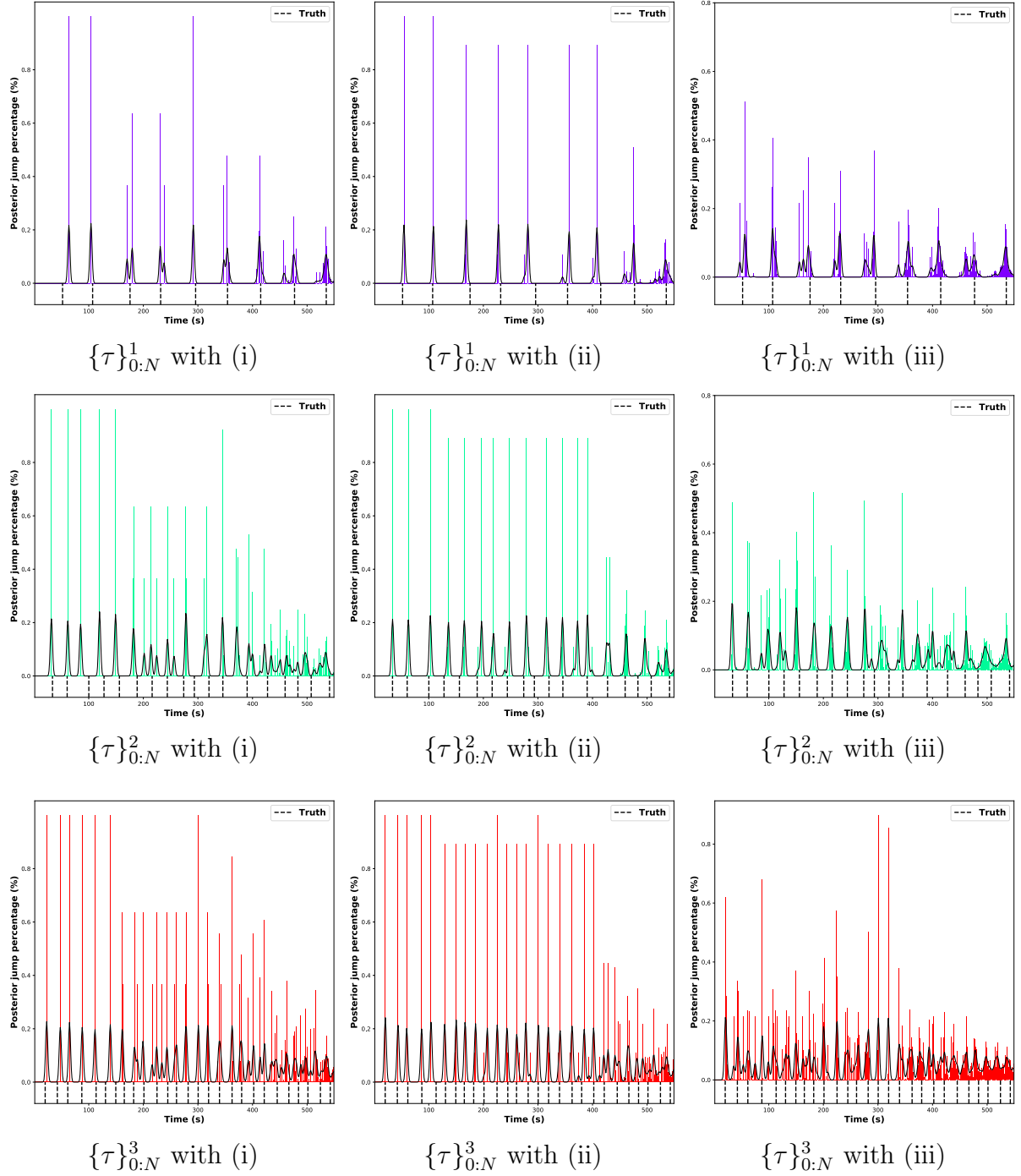


Figure 4.4.3: The posterior probabilities of jump times for each jump process. The probabilities are calculated on time bins formed by input timestamps $\{t_n\}_{n=0}^N$ (in coloured bars); while the (Gaussian) smoothed probabilities are plotted in black lines. (i), (ii), (iii) correspond to the RBPF configurations described in Section 4.4.1

Table 4.4.2: Numeric results for different inference configurations. **Bold** is the best.

	(i)	(ii)	(iii)	std. RBPF + (50%) adaptive IRS
l_2 -norm	0.5746	0.5767	0.5079	0.5560
avg. jump hit rate	14.5%	15.7%	16.7%	10.5%
avg. jump detection rate	20.8%	18.8%	25.4%	18.93%
avg. $\hat{\mathcal{L}}(Y)(1\sigma)$	-389.91 (1.977)	391.37 (1.363)	-388.75 (0.443)	-390.12 (1.171)

ics in a LOB market. The dataset is simulated to have three jump processes governed by distinct sets of hyperparameters as shown in **Table 4.4.1**. The model is designed specifically to allow trends of different time-scales to be induced by jumps arriving at different frequencies i.e. $z_{2,t}$ causes long-term trends but has a high expected jump interval while $z_{4,t}$ initiates sharp/instantaneous trends with a higher frequency and a longer cool-down period (d_3).

Figure 4.4.1 shows the simulated price process, volatility-free signal process, individual and aggregated jump processes. A total of 500 data points (with 54 jumps) are generated on an irregular time-grid with a maximum time interval of 1.5s. With the chosen hyperparameter settings, it can be seen that the simulated process possesses certain features that are often observed from a real-world LOB market including the superposition of long-short term trends and price volatility. The symmetric jumps resemble the impacts of submissions and cancellations of limit orders at both sides of the order book, which wears off exponentially.

Regarding the Bayesian inference of the model, I present results obtained using three different configurations of the RBPF: (i) a generic (bootstrap) RBPF with importance resampling (IRS); (ii) a semi-deterministic RBPF with importance resampling (IRS); and (iii) a semi-deterministic RBPF with “optimal” stratified resampling. Using a moderate number of particles $N_p = 500$, **Figure 4.4.2** shows the posterior state inference results on the synthetic dataset. Since the observation noise σ_{obs} is set to a small value for simulation, the posterior mean of x_t can be expected to be very close to the observations y_t , which will not reveal much information. Instead, I present the posterior mean and 95% confidence interval (CI) of the clear signal process $z_{1,t}$, and the aggregated velocity term ($z_{w2,t} + z_{3,t} + z_{4,t}$). Furthermore, the posterior jump probabilities are computed for time bins delineated by the input timestamps $\{t_n\}_{n=0}^N$. **Figure 4.4.3** shows the posterior jump-time probabilities of each jump process inferred using the three RBPF configurations. Visually from **Figure 4.4.3**, it can be observed that both RBPFs with semi-deterministic filtering scheme generally outperform the generic RBPF in terms of the jump accuracy. The importance resampling shows poor performance in handling the trajectory (time) degeneracy as can be observed from both **Figure 4.4.2** and **4.4.3**. On the other hand, the stratified resampler preserves more diverse particle trajectories by partitioning the particle collection into different strata [97], which provides more reasonable quantification of uncertainty e.g. more bins with high or moderate probabilities around the true jump time.

Table 4.4.2 quantitatively reports the inference performance using different metrics: the l_2 -norm computes the root-mean-squared error between the posterior mean and the truth of the state vectors; the average jump hit rate is the average probability of successfully detecting a jump within an interval e.g. $(t_{n-1}, t_n]$ as well as correctly identifying the

Table 4.4.3: LOB inference hyperparameters

Process	Hyperparameters
$z_{2,t}$	$\alpha_1 = 60.0$, $\beta_1 = 1.0$, $d_1 = 10.0$, $\theta_1 = -0.02$, $\mu_1 = 0$, $\sigma_1 = 0.2$
$z_{3,t}$	$\alpha_2 = 20.0$, $\beta_2 = 1.0$, $d_2 = 1.0$, $\theta_2 = -0.3$, $\mu_2 = 0$, $\sigma_2 = 0.4$
$z_{4,t}$	$\alpha_3 = 10.0$, $\beta_3 = 1.0$, $d_3 = 10.0$, $\theta_3 = -0.9$, $\mu_3 = 0$, $\sigma_3 = 0.5$
x_t	$\theta = -0.7$, $\sigma_v = 0.15$, $\sigma_{\text{obs}} = 0.1$

jump type i.e. from the k th jump process; while the average jump detection rate does not include the classification of jump types; the value $\hat{\mathcal{L}}(Y)$ represents the log of estimated model evidence (marginal likelihood) $\hat{p}(Y)$ which is computed as the product of the sums of the unnormalised weights in PF [64]:

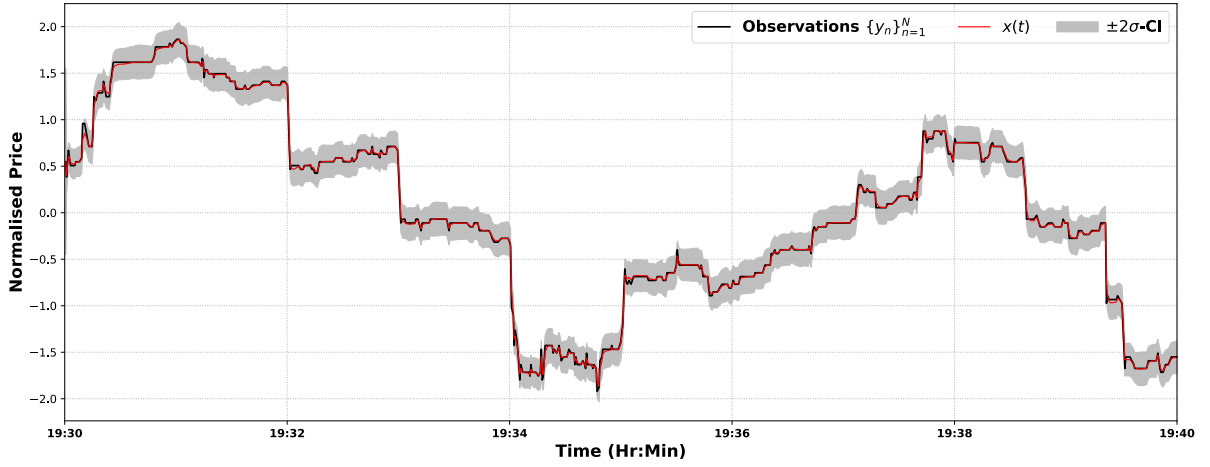
$$\hat{p}(Y) = \begin{cases} \prod_{n=1}^N \left[\sum_{p=1}^{N_p} \tilde{w}_n^p \right] & \text{if std. RBPF} \\ \prod_{n=1}^N \left[\sum_{p=1}^{N_p} \sum_{j=1}^{2^K} \tilde{w}_n^{p,j} \right] & \text{if semi-det. RBPF} \end{cases} \quad (4.29)$$

The values of $\hat{\mathcal{L}}(Y)$ reported in the table are obtained by averaging across 10 random runs of each algorithm. As the exact value of the model evidence cannot be obtained, the experiment compares the standard deviations of $\hat{\mathcal{L}}(Y)$ and focuses on the variance reduction aspect of these Monte Carlo algorithms. The table has also included an additional set of results obtained with a standard RBPF using adaptive IRS. It is clear that the semi-deterministic RBPF with stratified resampling outperforms other configurations in all metrics.

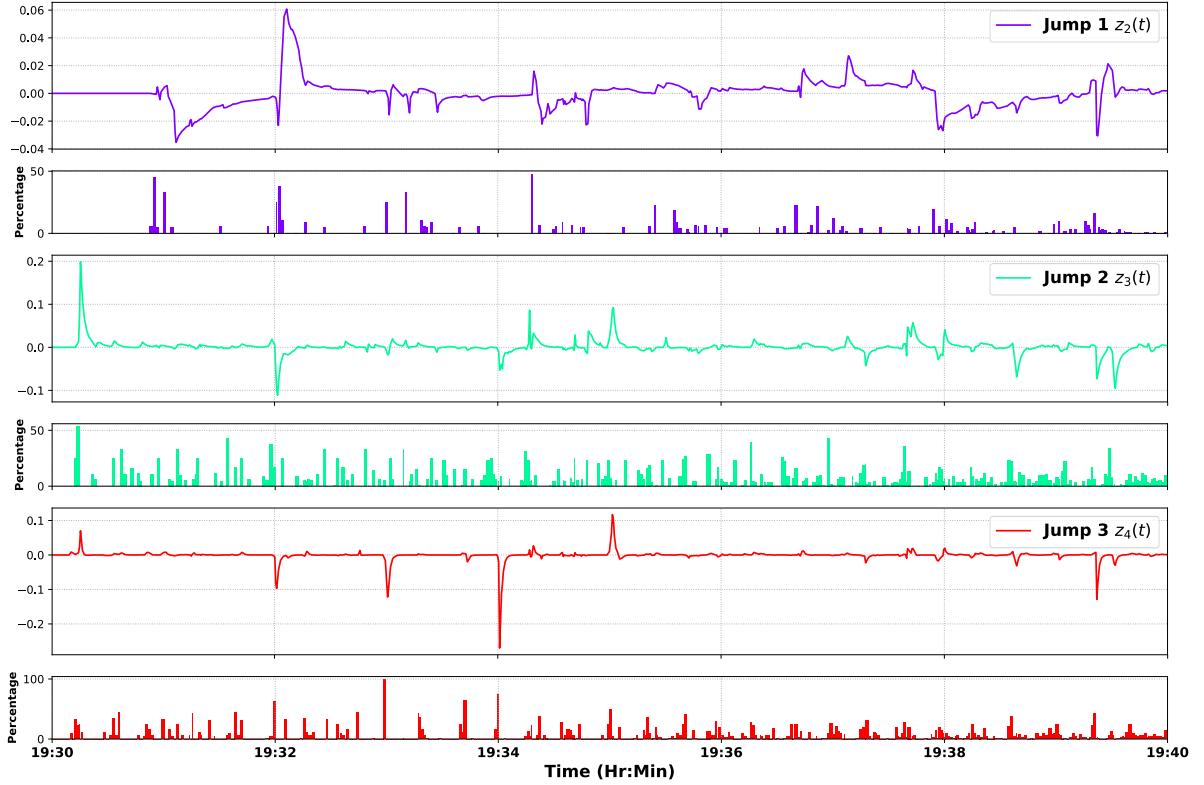
Based on the simulation results, the proposed model possesses the attractive feature of being conditionally tractable and Markovian while allowing trends of different time-scales to be expressed by multiple jump processes. Furthermore, empirical results suggest that the semi-deterministic filtering scheme and the stratified resampler are able to improve the posterior approximation accuracy on this difficult jump inference task.

4.4.2 *FOREX market price*

The high liquidity and the small *bid-ask spread* of FOREX market make it one of the most leveraged markets in finance. Hence, FOREX markets are often targeted and dominated by institutional orders of large volume that lead to significant impacts on the price dy-

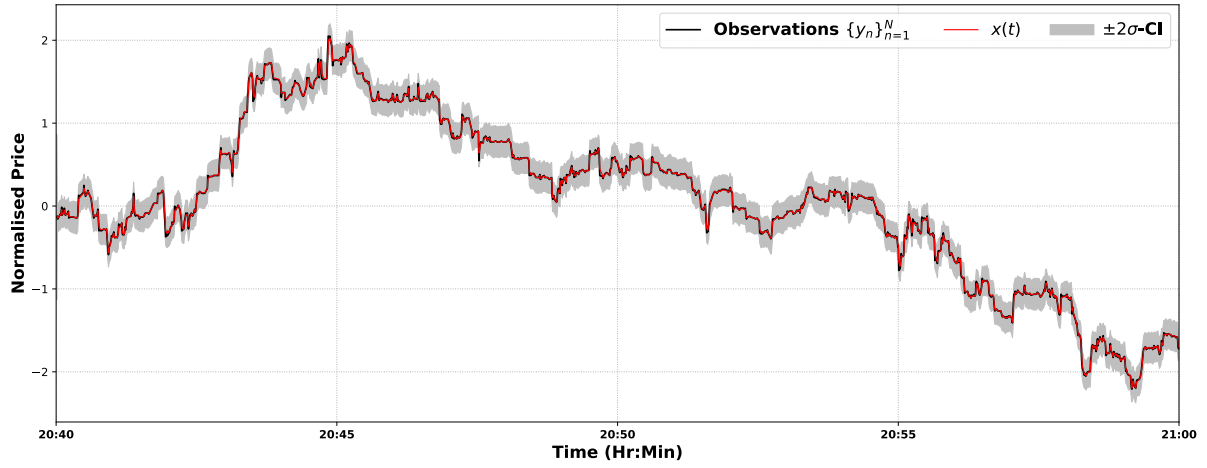


(a) Observations and posterior fair price process

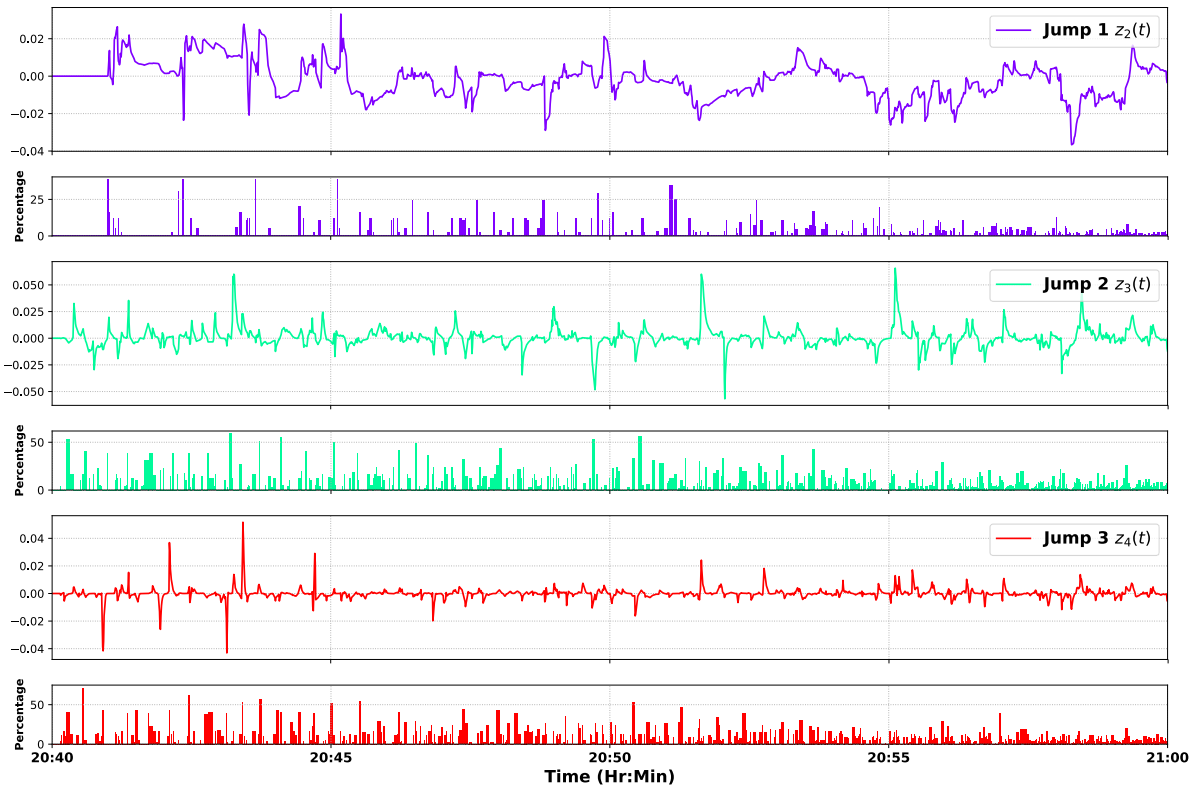


(b) Posterior jump trends and probability

Figure 4.4.4: Posterior inference results of the multi-jump diffusion model on the EUR-USD mid-price data between 19:30 and 19:40 ET on 2nd of September, 2015. (a) shows the observed mid-price (in black), posterior mean of the fair price x_t (in red) and its 95%-CI. (b) shows the posterior trend mean and probability of each jump process.



(a) Observations and posterior fair price process



(b) Posterior jump trends and probability

Figure 4.4.5: Posterior inference results on the EUR-USD mid-price data between 20:40 and 21:00 ET on 2nd of September, 2015.

namics. In order to investigate the connection between order operations and price actions, the proposed multi-jump diffusion model is applied to two sets of EUR-USD mid-price data for analyses on the inference results.

The two sets of data used in this experiment are extracted from the EUR-USD LOB on the 2nd of September 2015 for periods of 10 minutes (19:30 – 19:40 ET) and 20 minutes (20:40 – 21:00 ET) respectively. The model is constructed with 3 jump processes and a set of heuristically tuned hyperparameters shown in **Table 4.4.3** to allow salient features to be captured. Inference is performed with 500 particles using semi-deterministic RBPF with the “optimal” stratified resampling scheme. Both datasets are normalised to have zero mean and unit variance.

Figure 4.4.4 shows the posterior inference results on the mid-price data collected between 19:30 and 19:40. The mid-price in this period experiences several sharp jumps which are well captured by trend terms $z_3(t)$ (green) and $z_4(t)$ (red). These jumps are often caused by large buying/selling orders submitted very close to the mid-price (e.g. market orders). On the other hand, one can also observe trends in longer time scales e.g. from around 19:36 to around 19:38, as a result of continuous imbalance between the buying and selling pressure. These more persistent trends are also recognised as jump trends by the process $z_2(t)$ (purple) in the proposed model. Furthermore, it may be noticed that certain periodicity of the jump times exists in the bottom panel; while the jump sizes inferred are relatively small compared to other jumps from the same process. Such a pattern can be an indication of “iceberg orders” placed by institutional traders to minimise the impact of a single large limit order to the market.

Similarly, **Figure 4.4.5** shows the posterior inference results on the data collected in a 20-minutes interval from a busier period of the trading day. It is clear from the figure that the price dynamics in this period involves several more short-term jumps as a result of increased market liquidity and order transaction rate. While the short-term jump trends $z_3(t)$, $z_4(t)$ are fairly noisy, the long-term trend $z_2(t)$ in this case provides more informative signals for investment decisions.

Experiments with real-world LOB data illustrate that the proposed model is able to perform sequential tracking of the market price with good accountability of the process uncertainty as shown in the 95%-CI. The posterior inference of the jump processes in the model provides reasonable insights towards the market trends of various time scales which allow trend-following/momentum-based trading strategies to be executed at different frequencies and time horizons.

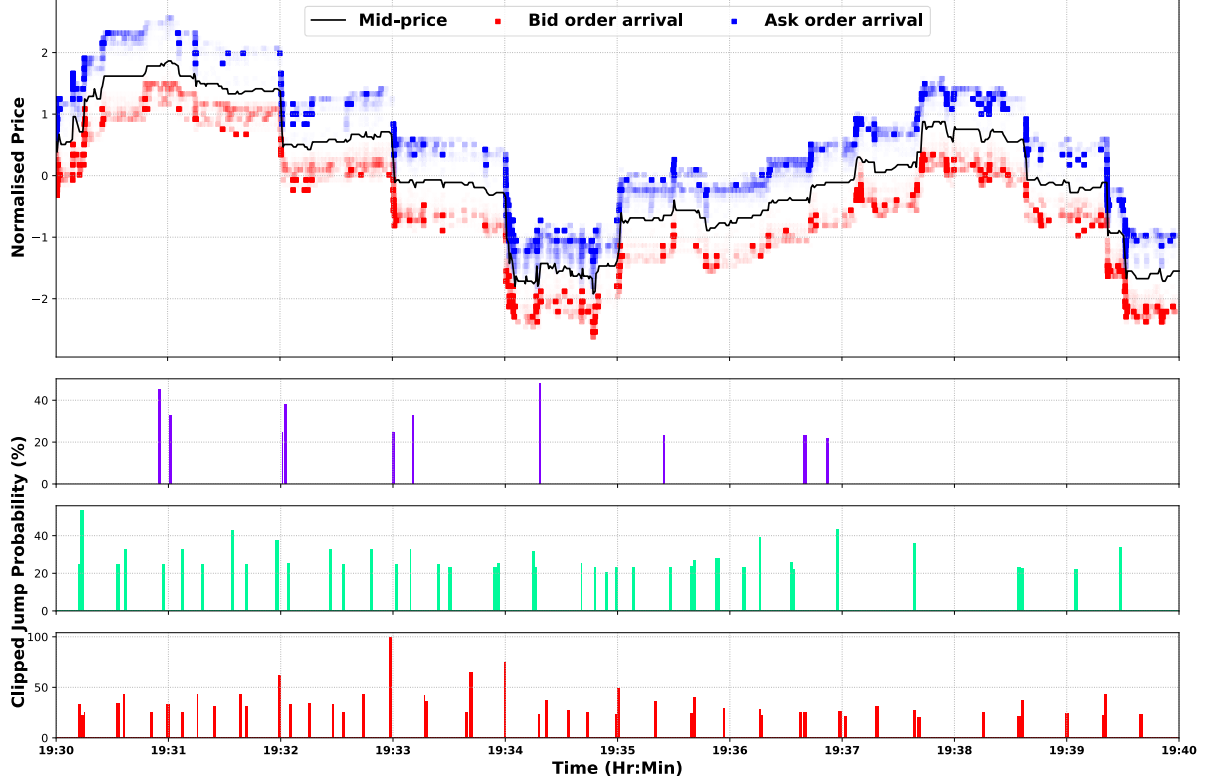


Figure 4.4.6: The top panel shows the arrivals of limit orders in scattered pixels during the period 19:30 – 19:40 ET. The opacity of the pixel is positively correlated to the volume of the order submitted. The bottom three panels re-display the posterior jump probabilities inferred in the proposed model with clipping applied at a threshold of 20%.

4.4.3 *Connections to the NHPP intensity inference model*

In **Chapter 2**, a model for sequential Bayesian inference on the intensity function of non-homogeneous Poisson processes (NHPP) is introduced. Its application on LOB arrival orders (i.e. submission) was also demonstrated to give good intuition of market behaviours. Therefore, it is possible to establish a connection between the multi-jump diffusion model proposed in this chapter to the S-LD intensity inference model.

Figure 4.4.6 shows the clipped jump probabilities computed in the first LOB experiment above using a (lower) threshold of 20%, together with a plot of arriving limit orders (i.e. submissions) in the top panel. Limit orders that submitted too far from the mid-price are filtered out and not shown in the plot. With the order volume positively correlated to the opacity of the scattered pixels, it can be roughly seen that there is a matching pattern between the inferred posterior jump times and the arrivals of large-volume orders (opaquer pixels). The pattern is clearer at the sharp price jumps (e.g. for $z_3(t)$ and $z_4(t)$) where the market also experiences large influx of limit orders from both bid and ask sides. This helps justify the intuition used when developing the proposed model: apart from outside factors, the jumps of high-frequency LOB market may also be initiated by operations of large institutional orders.

Therefore, a promising extension upon the current multi-jump diffusion model would be to integrate with the S-LD model proposed in **Chapter 2** which targets the intensities of large-volume order operations. The sequentially inferred NHPP intensities could thus replace the prior-based jump proposals and provide an improved predictive ability to the current model by incorporating real-time order book information.

4.5 Chapter summary and future work

In this chapter, I proposed a novel jump-diffusion model for price inference in the high-frequency LOB market. Taking special account of typical features of a liquid LOB market, the new model consists of multiple jump-reversion processes and a varying-mean OU process to accommodate multi-level trends of price dynamics as well as market’s inherent volatility. The model is constructed with a single SSM that preserves the (conditional) analytical tractability while allowing salient features to be incorporated.

Regarding the inference, I proposed a semi-deterministic filtering scheme beyond the RBPF algorithm in [13]. The proposed method performs more thorough exploration of the latent sample space and has been shown to achieve improved inference accuracy compared to the standard RBPF algorithm at the cost of extra computations. The model has also demonstrated good performance on real FOREX datasets by capturing trends at different time scales and providing plausible interpretations to the market activities.

For future work, the current model can be extended to have time-varying parameters/hyperparameters such as stochastic volatility $\sigma_{v,t}$ and mean-reversion $(\theta_t, \{\theta_{k,t}\}_k)$. Such an extension may provide additional accountability to the ever-changing nature of the financial markets.

Time-varying jump proposals may also be achieved by introducing non-homogeneous Poisson processes as jump processes with sequentially inferred intensities (e.g. with the S-LD model) as discussed in the previous section.

Recall that with the volumetric model introduced in **Chapter 3**, the price trend in the model takes the aggregated imbalance signal summarised from the outstanding limit order volumes on the LOB. Hence, another future research area of interest would be to consider both impulsive effect from instantaneous order operations and persistent imbalance pressure from the outstanding limit order volumes in the modelling of price dynamics, which may provide informative disaggregation of market signals and sophisticated trading strategies to be executed. This would typically require data with much finer details and higher temporal resolution e.g. *ticks data* and impose challenges in algorithm complexity.

Chapter 5

State-space regime-switching model with infinite mixture dynamics

So far in this thesis, I have covered multiple models that are built around the general framework of SSM. SSMs have demonstrated their particular strength in various aspects including simplicity in the design of system dynamics, good model interpretability and reasonable complexity for inference. While retrospective analyses of financial markets are important, the modelling of financial data stresses strongly the requirement to make accurate predictions with quantifiable uncertainties. Although the Bayesian structure of the SSM allows uncertainties to be readily quantified, its predictive power is heavily dependent on the prior design of dynamics, or the tuned set of hyperparameters. The process of tuning hyperparameters is typically time-consuming and prone to overfitting.

In this chapter, motivated by these limitations and inspired by the multi-modal diffusion behaviours of the price process in high-frequency financial markets, I propose a novel sequential model that accommodates multi-regime diffusion behaviours in a system requiring minimal prior assumptions and knowledge. The model is then inferred with an iterative inference algorithm that achieves both online sequential state inference and offline parameter learning.

The remainder of the chapter is organised as follows. Section 5.1 reviews the existing literature on switching SSMs and introduces the motivation in the context of financial price modelling. Two basic models which form important components of the full proposed model are introduced in Section 5.2. The detailed definition of the proposed model and its corresponding inference algorithm are presented in Section 5.3 and 5.4 respectively. In Section 5.5, I evaluate the performance of the model and the inference algorithm and present the experimental results obtained on both synthetic and real datasets. Finally, Section 5.6 explores the possibility of model generalisation using a hierarchical construction of the prior and present an example.

5.1 Introduction and motivations

The SSM, as a powerful generative modelling framework, has allowed many real-world systems to be probabilistically modelled using empirically estimated dynamics or theoretically derived differential equations. The development of the particle filter (PF) further allows richer prior knowledge about the modelled system to be incorporated in the SSM in a non-linear, non-Gaussian fashion, which could significantly improve model behaviour.

Diffusion in real-world processes rarely follows a fixed dynamics, which means that a single set of SSM hyperparameters can not capture all movement patterns exhibited by the object. To accommodate this, interacting multiple models (IMM) [98, 99, 100] and switching state-space models (SSSM) [15] were developed, in which the system dynamics or inputs transition discretely in a Markovian manner from one regime to another. Such models often require a substantial amount of prior knowledge about the dynamic system and careful tuning of the model hyperparameters. In order to better track manoeuvring targets and accommodate abruptly changes in diffusion parameters, the authors of [101] (with preliminary studies in [102, 103]) proposed a novel framework which incorporates the change-point analysis [104, 105] in the particle learning (PL) algorithm [106] to achieve online estimation of piecewise time-varying/constant parameters. While the change-point construction provides convenience for model inference, it does not perform clustering of the changes (i.e. each change is independent) and also gives up a certain amount of predictive power compared to the Markovian switching dynamics. More importantly, all these switching models rely on a more fundamental assumption – the exact number of regimes in the modelled system can be identified *a priori*.

In a high-frequency financial market, the price process is always subject to multiple known or unknown sources of innovations and disturbances. Different combinations of innovations are likely to result in different dynamics which the price process follows for a certain duration of time before switching into another. This limits the amount of prior knowledge that can be assumed or incorporated during the construction of a price model and imposes two major challenges for the SSM applications: (1) identification of the number of dynamic regimes; and (2) tuning/learning of hyperparameters for each regime. Unlike some other systems or processes where the number of dynamics can sometimes be estimated or known *a priori* by linking to their corresponding physical interpretations, market interactions make it difficult to anticipate the number of price regimes in the future market or to even identify the regimes from existing price data. On the other hand, heuristic tuning of SSM hyperparameters for financial models is more challenging than applications in other fields due to both the lack of process interpretability and the extra precautions needed to prevent catastrophic overfitting [107].

I propose a novel model which aims to tackle the challenging task where the dynamics of the diffusion object are unknown. This includes the number of movement regimes, the dynamics-controlling parameters associated with each regime and the transitional behaviours between regimes. Based on a standard linear Gaussian SSM, the proposed model allows a potentially *infinite* number of dynamic regimes to exist in the modelled system. The transition of regimes is designed to follow a continuous-time hidden semi-Markov Model (HsMM) where duration can be explicitly modelled. Furthermore, I present a complete inference algorithm that not only infers the latent states (i.e. regime, position, velocity etc.) but also performs Bayesian learning of the regime parameter and effectively mitigates the unwanted overfitting. This algorithm elegantly combines RBPF, particle-MCMC (PMCMC) [64] and blocked Metropolis-within-Gibbs (BMwG) samplers to achieve accurate posterior inference with a reasonable computational cost. The proposed model stands out not only in its minimal requirement for prior assumptions, but also in its ability to accommodate abrupt changes in diffusion behaviours and automatically cluster the changes for retrospective analyses.

5.2 Background and review

The hierarchical structure of Bayesian modelling has allowed us to model complex systems or data with novel integration of sophisticated models. In this section, I will review the two key components that realise the infinite regime-switching feature desired in the SSM: the HsMM and the Dirichlet process model (DPM).

Applications of the Dirichlet process (DP) as a non-parametric prior on switching models have been seen in several papers. The hierarchical DP (HDP), as a Bayesian-extended variant of the DP, was initially combined with a standard hidden Markov model (HMM) in [108] to provide richer transition dynamics for the discrete hidden states. Building on this work, a Bayesian non-parametric version of the HMM in which the HDP defines a prior distribution on the transition matrices over countably infinite state spaces was presented in [109], termed the HDP-HMM. Promising results have been obtained by the HDP-HMM in various applied problems such as visual scene recognition [110], music synthesis [111] and speaker diarisation [112]. The authors of [113] later improved the standard model with the more powerful HsMM and used Gibbs sampling for full Bayesian inference.

Extending the application of DP further to dynamical models, the DPM was employed in [114] to serve as general priors for the unknown process noise and observation noise in the SSM, which allows more interesting noise features, such as multimodality and heavy-tailedness, to be learned. The authors of [115] reviewed a number of HMM-governed discrete-time dynamical models including the SSM and the autoregressive (AR) model. However, the inference algorithm for these models and how inference can be achieved for

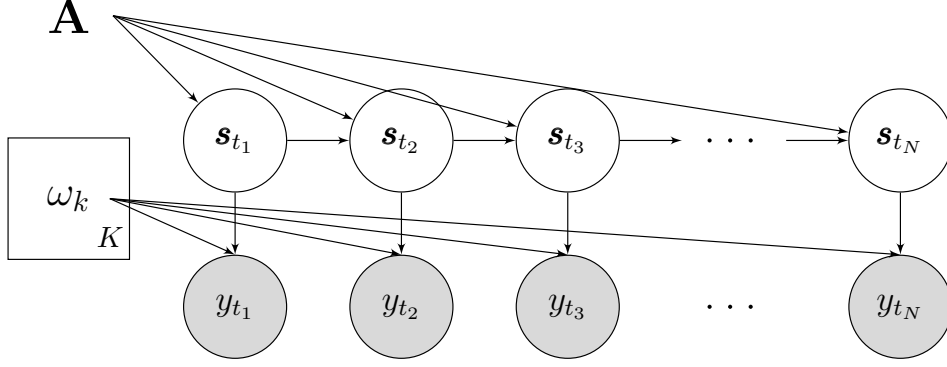


Figure 5.2.1: Graphical model of a standard HMM

non-conjugate cases are omitted in the paper. Another study in [116] aimed to model a linear-Gaussian dynamical system with unknown switching input regimes via the combination of the SSM and the HDP-HMM, and hence tackles a similar task as that described earlier in this chapter. Nevertheless, the proposed model in this chapter further achieves explicit modelling of regime duration under a non-linear continuous-time setting.

The remainder of this section will briefly review the two fundamental components that allow the building of the proposed model.

5.2.1 *Hidden semi-Markov model*

As has been seen in many applications, the standard HMM is a generative model consisting of two layers: a layer of hidden states $\{\mathbf{s}_n\}_{n=1}^N$; and a layer of state-dependent observations $\{y_n\}_{n=1}^N$ as shown in the graphical model of **Figure 5.2.1**. The hidden states $\{\mathbf{s}_n\}_{n=1}^N$ are a sequence of time-indexed discrete random variables, each of which represents one hidden class out of the total K classes that generates the observation y_n . Denote the hidden states using a sequence of “one-hot” vectors of finite dimension K , i.e. for the state \mathbf{s}_n to be assigned to class k , the k th element of the vector $\mathbf{s}_n^k = 1$ and all other elements $\mathbf{s}_n^{-k} = 0$. The transition of these hidden states is governed by a time-invariant transition matrix $\mathbf{A}_{K \times K}$ forming a Markov chain. Denoting A_{ij} as the element at i th row and j th column of matrix \mathbf{A} , we have $A_{ij} = p(\mathbf{s}_n^i = 1 \mid \mathbf{s}_{n-1}^j = 1)$ and $\sum_i A_{ij} = 1$. The transition dynamics can thus be summarised with a simple expression:

$$p(\mathbf{s}_n \mid \mathbf{s}_{n-1}) = \text{Categorical}(\mathbf{A}\mathbf{s}_{n-1}) \quad (5.1)$$

Further denote the emission distribution or likelihood for observation y_n conditioned on hidden state \mathbf{s}_n using *1-of- K* coding:

$$p(y_n \mid \mathbf{s}_n, \{\omega_k\}_{k=1}^K) = \prod_{k=1}^K p(y_n \mid \omega_k)^{\mathbf{s}_n^k} \quad (5.2)$$

where $\{\omega_k\}_{k=1}^K$ are the emission parameters for each class. Priors can also be placed on both emission parameters and transition matrix \mathbf{A} to enable full Bayesian inference for the model.

The simplicity of the standard HMM allows it to be conveniently integrated into different time-series models including the SSM. Despite its popularity, one of the major limitations of the HMM appears to be the lack of ability to explicitly model the runtime of each class (i.e. the duration of continuously staying in one class). The inherent nature of Markovian transitions prohibits any other modelling options for runtime except geometric distributions. This therefore motivates the development of HsMM as a variant of the standard HMM by allowing a random duration variable d_n to be drawn from a class-specific distribution $p(d_n|\mathbf{s}_n)$. Once a transition occurs, the state vector \mathbf{s}_n will remain unchanged until the duration d_n expires, at which point another Markovian transition will take place to produce a new (and different) hidden state. It is worth noting that although the transition at the end of a duration is Markovian, conditional independency does not generally exist as the name “semi-Markov” suggests.

While most HMMs and HsMMs operate under a discrete-time framework (i.e. the transitions occur at integer n of a regularly spaced time-grid), the use of a continuous-time HsMM is proposed here to allow further generalisation to irregular data updates and better adaptability to the continuous-time SSM used for the complete model. A continuous probability density is thus assigned to the class-specific duration distribution with support $[0, +\infty)$ e.g. a truncated normal distribution with mean and variance (μ_k, v_k) :

$$p(d_n | \mathbf{s}_n^k = 1) = \mathcal{N}^+(d_n | \mu_k, v_k) \quad (5.3)$$

Figure 5.2.2 shows the graphical model of a continuous-time explicit-duration HsMM. In such a model, let us allow the hidden states and observations to be indexed by both the data index n and the timing t_n . The duration d_l is drawn from the class-dependent distribution as in Eq. (5.3) and the hidden state transition dynamics are defined as follows:

$$\mathbf{s}_{t_l} = \mathbf{s}_{t_{l+1}} = \dots = \mathbf{s}_{t_n}, \quad n = \max_i \{i | t_i \leq t_l + d_l\} \quad (5.4)$$

$$p(\mathbf{s}_{t_{n+1}} | \mathbf{s}_{t_n}) = \text{Categorical}(\bar{\mathbf{A}} \mathbf{s}_{t_n}) \quad (5.5)$$

where the transition matrix $\bar{\mathbf{A}}$ is a re-normalised version of \mathbf{A} matrix with 0 self-transition probability e.g. the entry at i th row, k th column $\bar{A}_{ik} := \frac{A_{ik}}{1 - A_{kk}}(1 - \delta_{i-k})$. Another duration is randomly sampled at t_{n+1} conditioned on the just-switched hidden state $\mathbf{s}_{t_{n+1}}$. There are other ways of constructing a graphical model for HsMM: one typical discrete-time example is introduced in [113] using the concept of “super-state” nodes that emit random-length segments of observations; a different representation in [117] uses the idea of “finish” nodes

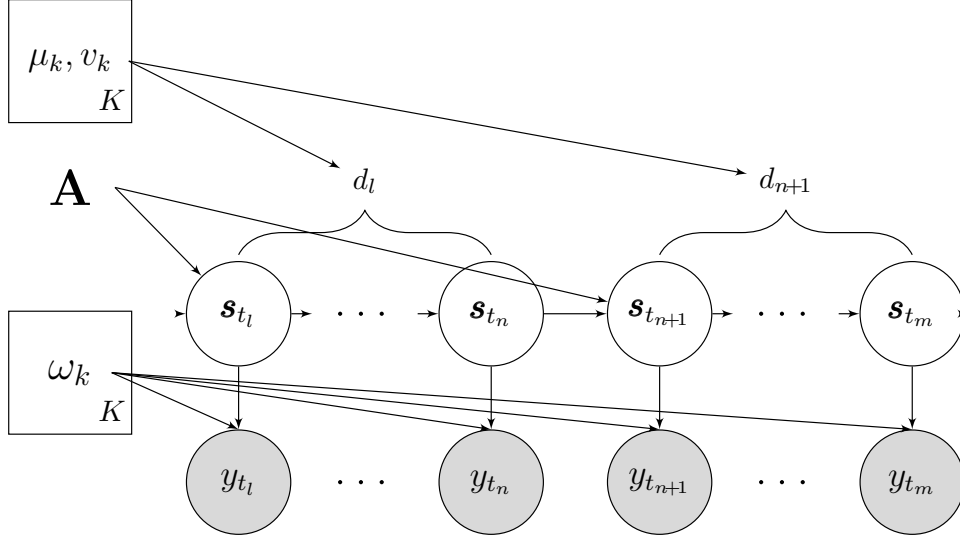


Figure 5.2.2: Graphical model for an explicit-duration HsMM

to signal state transitions. All these alternative graphical models demonstrate similar features as the model presented in **Figure 5.2.2**.

As far as the inference is concerned, while it is possible to perform the message-passing algorithm that is analogous to the forward-backward algorithm used for discrete-time HMM, the continuous-time generalisation as well as the integration with DPM and SSM presents challenges in terms of model tractability and algorithm complexity. Hence later in Section 5.3, a variant of this HsMM is presented, which restores the Markovian property with an accurate approximation of the duration model. The inference of the HsMM is handled in conjunction with the other two components in Section 5.4 once the complete model has been defined.

5.2.2 Dirichlet process model

In order to understand the “infinity” achieved by the DPM, the Dirichlet process (DP) should be introduced first. The DP is an infinite-dimensional generalisation of the Dirichlet distribution. Denoting \mathbb{P} as the random probability measure over (Ω, \mathcal{B}) , a DP is defined as $\mathbb{P}(\cdot) \sim \mathcal{DP}(\mathbb{P}_0, \alpha)$, which is parameterised by a base measure \mathbb{P}_0 and a positive scale factor α . Such a process satisfies that for any measurable partition B_1, \dots, B_K of the sample space Ω and any value of K , the probability measure of this partition follows:

$$\mathbb{P}(B_1), \dots, \mathbb{P}(B_K) \sim \text{Dirichlet}(\alpha \mathbb{P}_0(B_1), \dots, \alpha \mathbb{P}_0(B_K)) \quad (5.6)$$

Note that the sampled random probability measure \mathbb{P} essentially represents a special form of Bayesian histogram which only specifies that bin B_k is assigned with probability $\mathbb{P}(B_k)$ but does not specify how probability mass is distributed within the bin B_k and thus removes the sensitivity to the specific choice of bins/partitions. With such a formulation,

the DP prior allows the random probability measure $\mathbb{P}(\cdot)$ to be centered on a parametric model for the distribution of the data through the choice of \mathbb{P}_0 , while using α to control the degree of shrinkage of \mathbb{P} toward \mathbb{P}_0 .

An intuitive realisation of the DP is achieved via the *stick-breaking* construction. This representation allows us to induce $\mathbb{P} \sim \mathcal{DP}(\mathbb{P}_0, \alpha)$ by letting:

$$\mathbb{P}(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}(\cdot) \quad (5.7)$$

$$\pi_k = V_k \prod_{l < k} (1 - V_l) \quad (5.8)$$

$$V_k \sim \text{Beta}(1, \alpha), \quad \theta_k \sim \mathbb{P}_0 \quad (5.9)$$

where δ_θ is the Kronecker delta function that takes value 1 at θ and 0 elsewhere. It is worth noting that the finite number of K partitions in (5.6) have degenerated into an infinite number of atoms $\{\theta_k\}_{k=1}^{\infty}$ each associated with a weight π_k . This realisation process can be regarded as a process of breaking a unit-length stick (i.e. the total probability of 1) into an infinite number of pieces (i.e. atoms/samples) of length $\{\pi_k\}_{k=1}^{\infty}$ (i.e. probability masses). The length of the k th segment is determined by the length of the remaining stick $\prod_{l < k} (1 - V_l)$ and the random proportion V_k to break off from the remaining stick. The parameter α controls the relative distribution of the weights $\{\pi_k\}_{k=1}^{\infty}$ via the expected break-off portion $\mathbb{E}(V_k) = \frac{1}{1+\alpha}$.

Even though the stick-breaking process is intuitive, it is hardly useful as a model itself due to the degeneration from subsets to atoms. Especially in the modelling of continuous data, each atom should only be assigned with an infinitesimally small proportion of the “stick” meaning that $\alpha \rightarrow \infty$ and consequently $\mathbb{P}(\cdot) \rightarrow \mathbb{P}_0$.

However, with the stick-breaking DP as the prior, the DPM is developed by replacing the infinite number of samples θ_k drawn from \mathbb{P}_0 with conditional kernels to obtain:

$$p(y) = \sum_{k=1}^{\infty} \pi_k \mathcal{K}(y | \theta_k) \quad (5.10)$$

where π_k and θ_k are realisations of the DP; and $\mathcal{K}(\cdot | \theta_k)$ is the data kernel conditional on the parameter θ_k . With different kernels, the DPM acquires the ability of modelling various types of multi-modal data. The infinite sum in Eq. (5.10) does not imply that infinitely many clusters are *occupied* by the data y , but rather the model reserves the flexibility to introduce additional mixture component(s) when needed.

The posterior inference for the DPM is non-trivial as it involves the computation of a potentially infinite number of parameters; and the conjugacy between the base distribution \mathbb{P}_0 and the observational model in (5.10) is not guaranteed in most cases. However, the

marginalisation of the mixing measure $\mathbb{P}(\cdot)$ can be performed analytically to obtain a *Poly urn* representation for the induced prior which is commonly termed as a *Chinese restaurant process* [118, 119]: denoting $\theta^N = \{\theta_1, \dots, \theta_N\}$ as the data-specific parameters, the conditional predictive distribution for θ_h , $h \in \{1, \dots, N\}$ can thus be written as:

$$p(\theta_h | \theta_1, \dots, \theta_{h-1}) \sim \left(\frac{\alpha}{\alpha + h - 1} \right) \mathbb{P}_0(\theta_h) + \sum_{l=1}^{h-1} \left(\frac{1}{\alpha + h - 1} \right) \delta_{\theta_l} \quad (5.11)$$

The distribution above states that: in order to generate a new data point y_h based on θ_h given all previous parameters $\{\theta_l\}_{l=1}^{h-1}$, one can either draw a new value θ_h from \mathbb{P}_0 with probability $\alpha/(\alpha+h-1)$; or take the same value as one of the existing θ 's, each with a probability that is proportional to the number of data points already generated from using that (unique) value of θ .

This formulation of the parameter prior does not necessarily solve the problem of non-conjugacy but provides meaningful insight into how posterior inference for the DPM can be achieved via Gibbs sampling methods. Later in this chapter, I present an integrated inference algorithm that elegantly handles both non-conjugacy/intractability and infinity when the models are combined with feasible (e.g. not infinite or exploding) and bounded computation.

To help get an initial idea of how the DPM works and performs, I apply the DPM model on a synthesised set of multi-modal 2-D Gaussian samples. In this example, the DPM is constructed such that:

$$\begin{aligned} \mathcal{K}(\mathbf{y} | \theta_k) &= \mathcal{N}(\mathbf{y} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ \theta_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} &\sim \mathbb{P}_0(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \text{Normal-inverse-Wishart}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \end{aligned} \quad (5.12)$$

Inferred with a *collapsed Gibbs sampler* (thanks to conjugacy), **Figure 5.2.3** shows four identified posterior clusters. The locations of the crosses represent the clusters' means while the two lines of each cross show the eigenvalues and eigenvectors of the corresponding covariance matrix. In the figure, it can be seen that there are four occupied clusters in the DPM posterior with the top cluster and the bottom right cluster successfully identified and learned to the ground truth. The bottom left cluster pools the data points generated from two true Gaussian components into one posterior Gaussian component. This is reasonable as the simulated data points are very close together and the single Gaussian distribution learned by the DPM can provide a very good fit to these data. The unidentified true component (in the middle) generates only 1 data point out of the total 5000 points and is therefore not picked up by the DPM. With this example, it is fair to say that the DPM is able to assign an appropriate amount of clusters to multi-modal data without overfitting.

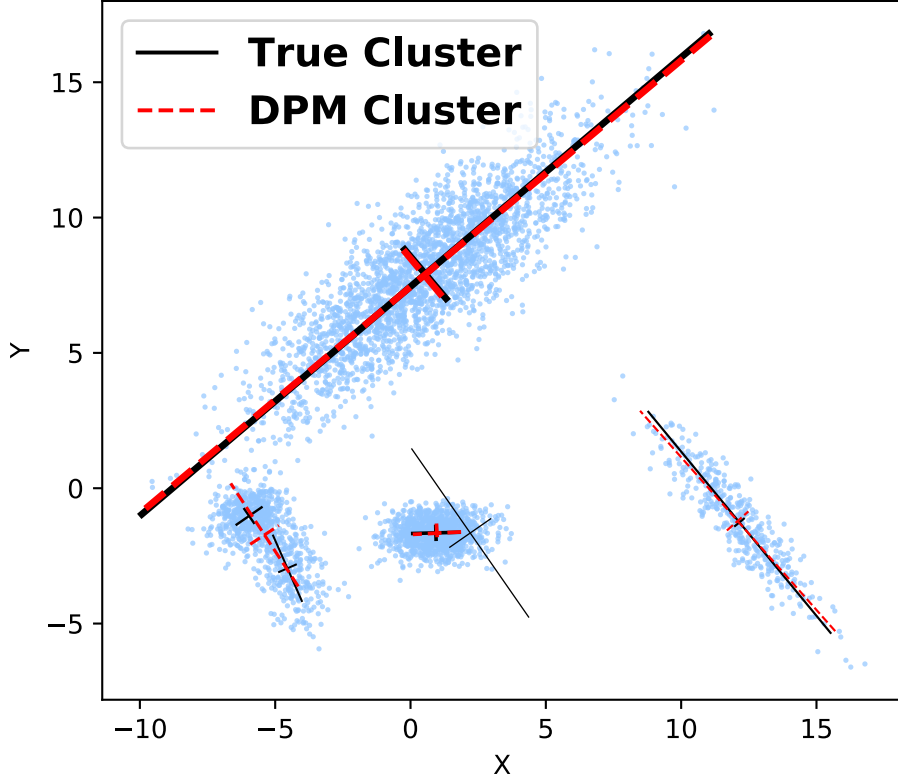


Figure 5.2.3: Inferred DPM posterior on 2D data simulated from a Gaussian mixture model (GMM). Crosses show the clusters' means and eigenvectors of the clusters' covariance.

5.3 Model

It is relatively clear that the features provided by the HsMM and the DPM introduced in Section 5.2 are capable of extending the standard SSM to provide a more powerful model for complex dynamic systems. In this section, I will introduce the proposed model in detail and demonstrate how individual models can be integrated together to provide the desirable features.

5.3.1 Continuous-time State-space Model

The continuous-time SSM is the core of the proposed model. The same Langevin dynamical model that has been seen in both **Chapter 3** and **4** for market price modelling is used here. With $x_{1,t}$ being the market fair price and $x_{2,t}$ being the price trend term, the SSM is constructed as:

$$d\mathbf{x}_t = \mathbf{F} \mathbf{x}_t dt + \mathbf{h} dW_t \quad (5.13)$$

where dW_t is the Wiener process and

$$\mathbf{x}_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 0 & 1 \\ 0 & \theta \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} 0 \\ \sigma \end{bmatrix} \quad (5.14)$$

Based on the derivations from previous chapters, it should be clear that the conditional transition/propagation density of the state vector from t_{n-1} to t_n can be readily computed as a linear-Gaussian distribution:

$$p(\mathbf{x}_{t_n} | \mathbf{x}_{t_{n-1}}, \theta, \sigma) = \mathcal{N}\left\{\mathbf{x}_{t_n} | \boldsymbol{\mu}(\mathbf{F}, \Delta t_n, \mathbf{x}_{t_{n-1}}), \boldsymbol{\Sigma}(\mathbf{F}, \Delta t_n, \mathbf{h})\right\} \quad (5.15)$$

where Δt_n is defined as the interval length $|t_n - t_{n-1}|$; and the parameters are computed as:

$$\boldsymbol{\mu}(\mathbf{F}, \Delta t_n, \mathbf{x}_{t_{n-1}}) = \mathbf{F}_n(\theta) \mathbf{x}_{t_{n-1}} \quad (5.16)$$

$$\boldsymbol{\Sigma}(\mathbf{F}, \Delta t_n, \mathbf{h}) = \mathbf{F}_n(\theta) \mathbf{K}_n(\theta, \sigma) \mathbf{F}_n(\theta)^T \quad (5.17)$$

$$\mathbf{F}_n(\theta) = e^{\mathbf{F} \Delta t_n} \quad (5.18)$$

$$\mathbf{K}_n(\theta, \sigma) = \int_0^{\Delta t_n} e^{-\mathbf{F}\tau} \mathbf{h} \mathbf{h}^T (e^{-\mathbf{F}\tau})^T d\tau \quad (5.19)$$

In this particular construction, the price dynamics are primarily controlled by two parameters θ and σ . Therefore in the complete model, it is aimed to allow different combinations of these two parameters so that the Langevin model is able to generate and accommodate various behaviours of the market price including long-term and short-term trends, fluctuations and even unstable processes. Naturally, these parameters are also learned in a Bayesian manner using the proposed inference algorithm.

Finally, define the observation model for price data y_{t_n} as:

$$p(y_{t_n} | \mathbf{x}_{t_n}) = \mathcal{N}\left(y_{t_n} | \mathbf{G} \mathbf{x}_{t_n}, \sigma_{\text{obs}}^2\right) \quad (5.20)$$

with $\mathbf{G} = [1 \ 0]$. Although it is possible to include the observation variance σ_{obs}^2 in the learning framework, this possibility is not considered here and σ_{obs}^2 is assumed to be fixed. The learning of observation variance is covered in **Chapter 6** including a heavy-tailed version of the PF.

5.3.2 Integration of models

As can be seen from the transition density (5.15), the conditional predictions of future states in an SSM depend completely on the exact prior dynamic construction and parameters chosen. Thus unlike other models where non-informative priors can be chosen, the prior dynamics for SSM significantly affect both prediction and posterior tracking per-

formance. Therefore, the proposed integration of the continuous-time SSM, the HsMM and the DPM can help add data-oriented multi-modal switching dynamics to the original SSM.

Semi-Markovian switching

To model the switching between discrete regimes in continuous time, let us first define a regime assignment random variable \mathbf{s}_n for each timestamp t_n indexed by n . The regime assignment \mathbf{s}_n is a one-hot-encoded vector of a finite dimension K , which specifies the exact dynamic regime that the state vector \mathbf{x}_n should follow at time t_n . With the assumption that regimes are of the same stochastic form (e.g. SDE) with different dynamics-controlling parameters, \mathbf{s}_n essentially determines which pair of parameters (θ, σ) should be used in the transition density of Eq. (5.15). Given K pairs of parameters $\{\theta_k, \sigma_k\}_{k=1}^K$, the state vector transition density can thus be re-written conditional on regime assignment using the 1-of- K representation as:

$$p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{s}_n, \{\theta_k, \sigma_k\}_{k=1}^K) = \prod_{k=1}^K \mathcal{N}\left\{\mathbf{x}_n | \boldsymbol{\mu}(\theta_k, \Delta t_n, \mathbf{x}_{n-1}), \boldsymbol{\Sigma}(\theta_k, \sigma_k, \Delta t_n)\right\}^{\mathbf{s}_n^k} \quad (5.21)$$

The next step is to allow the evolution of \mathbf{s}_n through time to be governed by the continuous-time semi-Markov transition dynamics so that the duration of each regime can be modelled explicitly.

Although it is possible to simply adopt the same probabilistic framework as defined earlier in the review of the HsMM, the inherent Markovian structure of the SSM can no longer be preserved, which will cause challenges to inference. Instead, the model introduces another random variable r_n as the runtime of the current regime at time t_n . The runtime states the amount of (continuous) time that the object has spent so far in the current regime and gets reset to zero once a switch occurs. Hence given a set of regime-specific duration parameters $\{\mu_k, v_k\}_{k=1}^K$ (e.g. for the truncated normal), r_n propagates as:

$$r_0 = 0 \quad (5.22)$$

$$p(r_n | r_{n-1}, \{\mu_k, v_k\}_{k=1}^K, \mathbf{s}_{n-1}) = \begin{cases} H_k(r_{n-1}, \Delta t_n) & \text{if } r_n = 0 \\ 1 - H_k(r_{n-1}, \Delta t_n) & \text{if } r_n = r_{n-1} + \Delta t_n \\ 0 & \text{otherwise} \end{cases} \quad (5.23)$$

The function $H(\cdot)$ is often referred to as the *hazard* function in discrete-time switching models [120]. Whilst in this model, it gives the probability of a switch occurring between t_{n-1} and t_n given that there has been no switch since r_{n-1} ago (i.e. the previous switch). If I define τ as the *exact* runtime which differs from the definition of r_n in that it is not

associated with an observation timestamp t_n and can be evaluated at any arbitrary time point, it is easy to prove that the maximum τ in a specific regime is just the regime duration modelled in Eq. (5.3). Thus the continuous-time hazard function can be calculated as:

$$\begin{aligned} H_k(r_{n-1}, \Delta t_n) &= P(r_{n-1} < \tau \leq (r_{n-1} + \Delta t_n) \mid \tau > r_{n-1}) \\ &= \frac{S_k(r_{n-1}) - S_k(r_{n-1} + \Delta t_n)}{S_k(r_{n-1})} \end{aligned} \quad (5.24)$$

where $S_k(\cdot)$ is the survival function of (5.3) with parameters (μ_k, v_k) of the specific regime k encoded in \mathbf{s}_{n-1} . This formulation of the semi-Markov process is attractive also because: (i) it restores the Markovian property; (ii) it automatically handles the positivity constraint of the duration distribution; and (iii) it fits naturally into the sequential framework of the SSM as well as the PF that will be used for inference.

It can be noted that r_n is a discrete approximation of τ with an error less than the update interval Δt_n . Hence the proposed switching model is also an approximation of the real-world regime-switching systems as it ignores the change of dynamical behaviours within the update interval (i.e. in-between observations) and constrains the switches to take place at the timestamps of the observations. In the modelling of high-frequency LOB markets, this approximation is very accurate due to the fine temporal resolution of the price updates. Additionally, the approximated runtime alleviates the burden on the PF inference by reducing the continuous sample space of all possible switching times to a discrete set of intervals.

Conditioned on the runtime r_n , the transition probability for regime assignment \mathbf{s}_n is:

$$p(\mathbf{s}_n \mid \mathbf{s}_{n-1}, r_n \neq 0) = \begin{cases} 1.0 & \mathbf{s}_n = \mathbf{s}_{n-1} \\ 0 & \text{otherwise} \end{cases} \quad (5.25)$$

$$p(\mathbf{s}_n \mid \mathbf{s}_{n-1}, r_n = 0) = \text{Categorical}(\bar{\mathbf{A}} \mathbf{s}_{n-1}) \quad (5.26)$$

where $\bar{\mathbf{A}}$ is the re-normalised Markovian transition matrix which prohibits self-transition.

Infinite regimes via the DPM

The model introduced so far in this section operates with a finite and pre-assigned K regimes. The incorporation of DPM helps relax this fundamental assumption and allows the model to work with a potentially infinite number of dynamical regimes.

Start by simplifying the notation of regime-specific parameters as $\beta_k = \{\theta_k, \sigma_k\}$ and duration parameters as $\psi_k = \{\mu_k, v_k\}$. The stick-breaking $\mathcal{DP}(\alpha, \mathbb{P}_0)$ is assigned as the

prior for both parameters β and ψ :

$$\begin{aligned}\mathbb{P}(\beta, \psi) &= \sum_{k=1}^{\infty} \pi_k \delta_{\beta_k, \psi_k}(\beta, \psi) \\ \pi_k &= V_k \prod_{l < k} (1 - V_l) \\ V_k &\sim \text{Beta}(1, \alpha), \quad (\beta_k, \psi_k) \stackrel{\text{i.i.d.}}{\sim} \mathbb{P}_0\end{aligned}\tag{5.27}$$

However instead of having a simple observation kernel as introduced in Section 5.2, the DPM employed in the proposed model is much more complex as it determines the transitions of all r_n , \mathbf{s}_n and \mathbf{x}_n . Imagine a transition from time t_{n-1} to t_n :

$$\begin{aligned}& p(\mathbf{x}_n, r_n, \mathbf{s}_n \mid \mathbf{x}_{n-1}, r_{n-1}, \mathbf{s}_{n-1}, \{\beta_k\}_{k=1}^{\infty}, \{\psi_k\}_{k=1}^{\infty}) \\ &= p(r_n \mid r_{n-1}, \{\psi_k\}_{k=1}^{\infty}, \mathbf{s}_{n-1}) \times p(\mathbf{s}_n \mid \mathbf{s}_{n-1}, r_n) \times p(\mathbf{x}_n \mid \mathbf{x}_{n-1}, \mathbf{s}_n, \{\beta_k\}_{k=1}^{\infty})\end{aligned}\tag{5.28}$$

This gives the DPM in the context of the proposed model as an infinite mixture of dynamic transition kernels. And the Markovian transition of regime assignment in (5.26) will be changed into the re-normalised stick-breaking proportion (length):

$$p(\mathbf{s}_n \mid \mathbf{s}_{n-1}, r_n = 0) = \bar{\pi}_{s_{n-1}}\tag{5.29}$$

where $\bar{\pi}_{s_{n-1}}$ is re-normalised after eliminating the self-transition probability of the regime specified in \mathbf{s}_{n-1} . The definition of the DPM component of the proposed model is completed by specifying the prior base distribution \mathbb{P}_0 for β and ψ :

$$\mathbb{P}_0 = \mathcal{N}(\theta \mid \mu_\theta, \sigma_\theta) \times \text{Gamma}(\sigma \mid \alpha_\sigma, \beta_\sigma) \times \mathcal{NIG}(\psi \mid m_\psi, \nu_\psi, \alpha_\psi, \beta_\psi)\tag{5.30}$$

where $\mathcal{NIG}(\cdot)$ is a normal inverse-Gamma distribution which is also the conjugate prior for (truncated) normal distribution with unknown mean and variance.

Figure 5.3.1 shows the Bayesian graphical model for the proposed model where the hidden semi-Markov transition is re-parameterised by runtime r_n . Thereby, given a set of N data points/observations $Y = \{y_n\}_{n=1}^N$, the corresponding timestamps $\{t_n\}_{n=1}^N$ and denoting $X = \{\mathbf{x}_n\}_{n=1}^N$, $S = \{\mathbf{s}_n\}_{n=1}^N$, the joint probability of all variables is:

$$\begin{aligned}& p(Y, X, S, r, \{\beta\}_{k=1}^{\infty}, \{\psi\}_{k=1}^{\infty}) \\ &= p(\mathbf{x}_0) p(r_0) p(\mathbf{s}_0) p(\{\beta\}_{k=1}^{\infty}, \{\psi\}_{k=1}^{\infty}) \times \prod_{n=1}^N \left\{ p(y_n \mid \mathbf{x}_n) p(\mathbf{x}_n \mid \mathbf{x}_{n-1}, \mathbf{s}_n, \{\beta\}_{k=1}^{\infty}) \right. \\ & \quad \left. \times p(\mathbf{s}_n \mid \mathbf{s}_{n-1}, r_n) p(r_n \mid r_{n-1}, \mathbf{s}_{n-1}, \{\psi\}_{k=1}^{\infty}) \right\}\end{aligned}\tag{5.31}$$

In this chapter, the model is constructed, inferred and tested under the assumption that

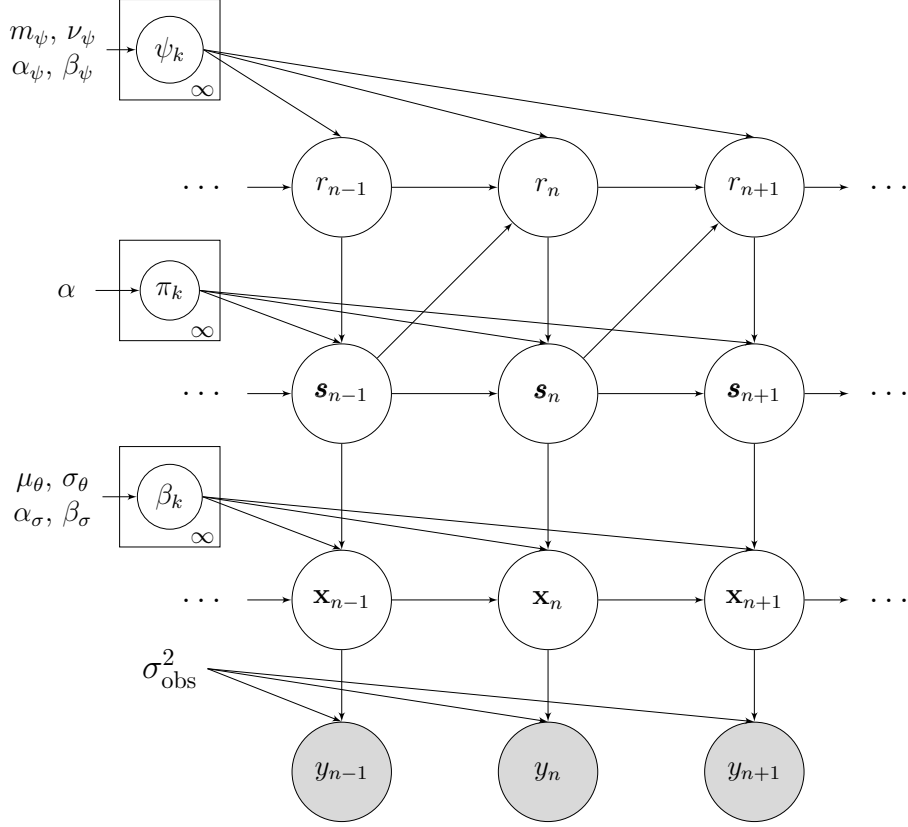


Figure 5.3.1: Graphical model for the proposed regime-switching model. The transition of regime allocation \mathbf{s}_n is governed by a HsMM. Given the dynamical regime at each timestamp, the state vector \mathbf{x}_n follows the diffusion of a pre-defined SSM with a set of regime-specific diffusion parameters. DPM prior provides non-parameteric modelling to both duration and diffusion parameters.

the diffusion follows a fixed functional form of SSM i.e. infinite mixture dynamics are achieved by different combinations of parameters under the same Langevin framework. However, in Section 5.6, I will demonstrate how this assumption can be relaxed so that diffusion models of different forms can also be included in the DPM via a hierarchical linkage.

5.4 Inference

Inference for the proposed model is non-trivial especially when it requires the learning of potentially infinite-dimensional regime parameters $\{\beta_k, \psi_k\}_{k=1}^{\infty}$ in conjunction with the state vectors $\{\mathbf{x}_n\}_{n=0}^N$ and regime assignment $\{\mathbf{s}_n\}_{n=0}^N$. The continuous-time SSM further prevents the closed-form inference of the dynamics-controlling parameters.

Inspired by the divide-and-conquer principle, the joint posterior derived from Eq. (5.31) is temporarily factorised into two parts using *mean-field* approximation: (i) the sequential part concerning state vectors, runtime and regime assignments; and (ii) the

static regime parameters in the DPM:

$$\begin{aligned}
& p(\{\beta_k\}_{k=1}^\infty, \{\psi_k\}_{k=1}^\infty, S, X, r | Y) \\
& \stackrel{\text{M-F}}{\approx} \underbrace{p(X, S, r | \{\beta_k\}_{k=1}^\infty, \{\psi_k\}_{k=1}^\infty, Y)}_{(i)} \times \underbrace{p(\{\beta_k\}_{k=1}^\infty, \{\psi_k\}_{k=1}^\infty | X, S, r, Y)}_{(ii)} \quad (5.32)
\end{aligned}$$

The above factorisation provides a useful starting point for a Gibbs-fashion alternating sampler for iterative inference of the two conditional posteriors. Two separately inference algorithms are hence employed, each targeting one conditional posterior of the factorisation. The outputs are joined together using the particle Gibbs (PG) algorithm under the PMCMC framework [64].

5.4.1 RBPF: time-series state inference

The state inference has always been a major task for time-series model including the SSM. In the application of LOB price modelling, the state vector contains the hidden market fair price and its trends while both runtime and regime assignments provide significant indications to investment decisions. Therefore in this part, a sequential inference algorithm is proposed, which targets the conditional posterior (i) in Eq. 5.32.

In this part of the inference, let us assume that the static regime parameters are known and write the conditional posterior in terms of its recursion as:

$$\begin{aligned}
& p(\mathbf{x}_{0:n}, \mathbf{s}_{0:n}, r_{0:n} | y_{1:n}, \{\beta_k\}_{k=1}^\infty, \{\psi_k\}_{k=1}^\infty) \\
& \propto p(\mathbf{x}_{0:n-1}, \mathbf{s}_{0:n-1}, r_{0:n-1} | y_{1:n-1}, \{\beta_k\}_{k=1}^\infty, \{\psi_k\}_{k=1}^\infty) p(y_n | \mathbf{x}_n) \times \\
& p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{s}_n, \{\beta_k\}_{k=1}^\infty) p(\mathbf{s}_n | \mathbf{s}_{n-1}, r_n) p(r_n | r_{n-1}, \mathbf{s}_{n-1}, \{\psi_k\}_{k=1}^\infty) \quad (5.33)
\end{aligned}$$

This recursion is again very similar to that used for the price models introduced in **Chapter 3**, except that now it has the conditional transition densities for both runtime r_n and regime assignment \mathbf{s}_n . However, this still allows us to use a particle filter or even a Rao-Blackwellised particle filter (RBPF) for inference.

Kalman filtering for state vectors

As can be seen from (5.15) and (5.21), the state vectors \mathbf{x}_n are linear-Gaussian conditioned on the regime assignments \mathbf{s}_n and a Gaussian initialisation for \mathbf{x}_0 . Therefore given a linear Gaussian observation model (5.20), this model feature can be fully capitalised by marginalising out the state vectors and make conditionally closed-form inference with a classic Kalman filter.

Given a known sequence of runtime and regime assignments $\{r_{0:n}^{(p)}, \mathbf{s}_{0:n}^{(p)}\}$ up till time t_n ,

the marginal predictive distribution of the state vector \mathbf{x}_n is:

$$\begin{aligned} & p(\mathbf{x}_n | y_{1:n-1}, \mathbf{s}_{0:n}^{(p)}, r_{0:n}^{(p)}, \{\beta_k\}_{k=1}^\infty) \\ &= \int p(\mathbf{x}_{n-1} | y_{1:n-1}, \mathbf{s}_{0:n-1}^{(p)}, r_{0:n-1}^{(p)}, \{\beta_k\}_{k=1}^\infty) p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{s}_n^{(p)}, \{\beta_k\}_{k=1}^\infty) d\mathbf{x}_{n-1} \end{aligned} \quad (5.34)$$

Suppose that the filtering posterior of \mathbf{x}_{n-1} at time t_{n-1} can also be expressed as a Gaussian distribution i.e.:

$$p(\mathbf{x}_{n-1} | y_{1:n-1}, \mathbf{s}_{0:n-1}^{(p)}, r_{0:n-1}^{(p)}, \{\beta_k\}_{k=1}^\infty) = \mathcal{N}(\mathbf{x}_{n-1} | \boldsymbol{\mu}_{n-1|0:n-1}^{(p)}, \boldsymbol{\Sigma}_{n-1|0:n-1}^{(p)}) \quad (5.35)$$

Then the marginal predictive distribution of (5.34) can be readily obtained as Gaussian from the Kalman predictive step:

$$p(\mathbf{x}_n | y_{1:n-1}, \mathbf{s}_{0:n}^{(p)}, r_{0:n}^{(p)}, \{\beta_k\}_{k=1}^\infty) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:n-1}^{(p)}, \boldsymbol{\Sigma}_{n|0:n-1}^{(p)}) \quad (5.36)$$

with predictive parameters:

$$\boldsymbol{\mu}_{n|0:n-1}^{(p)} = \mathbf{F}_n(\beta_{\mathbf{s}_n^{(p)}}) \boldsymbol{\mu}_{n-1|0:n-1}^{(p)} \quad (5.37)$$

$$\boldsymbol{\Sigma}_{n|0:n-1}^{(p)} = \mathbf{F}_n(\beta_{\mathbf{s}_n^{(p)}}) [\boldsymbol{\Sigma}_{n-1|0:n-1}^{(p)} + \mathbf{K}_n(\beta_{\mathbf{s}_n^{(p)}})] \mathbf{F}_n(\beta_{\mathbf{s}_n^{(p)}})^T \quad (5.38)$$

where $\mathbf{F}_n(\cdot)$ and $\mathbf{K}_n(\cdot)$ are calculated with formulae (5.18) and (5.19) respectively using the parameters of the regime indicated in assignment $\mathbf{s}_n^{(p)}$. In order to complete the recursive algorithm, Kalman filter updates the new filtering posterior at time t_n with the correction from observation y_n :

$$\mathbf{Q}_n = \boldsymbol{\Sigma}_{n|0:n-1}^{(p)} \mathbf{G}^T (\mathbf{G} \boldsymbol{\Sigma}_{n|0:n-1}^{(p)} \mathbf{G}^T + \sigma_{\text{obs}}^2)^{-1} \quad (5.39)$$

$$\boldsymbol{\mu}_{n|0:n}^{(p)} = \boldsymbol{\mu}_{n|0:n-1}^{(p)} + \mathbf{Q}_n (y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1}^{(p)}) \quad (5.40)$$

$$\boldsymbol{\Sigma}_{n|0:n}^{(p)} = (\mathbf{I} - \mathbf{Q}_n \mathbf{G}) \boldsymbol{\Sigma}_{n|0:n-1}^{(p)} \quad (5.41)$$

Particle filter for non-linear runtime and state assignments

With the state vectors marginalised and inferred (conditionally) in closed form, a particle filter is employed to target the new posterior recursion featuring only $r_{0:n}$ and $\mathbf{s}_{0:n}$:

$$\begin{aligned} & p(\mathbf{s}_{0:n}, r_{0:n} | y_{1:n}, \{\beta_k\}_{k=1}^\infty, \{\psi_k\}_{k=1}^\infty) \\ & \propto p(\mathbf{s}_{0:n-1}, r_{0:n-1} | y_{1:n-1}, \{\beta_k\}_{k=1}^\infty, \{\psi_k\}_{k=1}^\infty) p(\mathbf{s}_n | \mathbf{s}_{n-1}, r_n) \times \\ & p(r_n | r_{n-1}, \mathbf{s}_{n-1}, \{\psi_k\}_{k=1}^\infty) p(y_n | y_{1:n-1}, \mathbf{s}_{0:n}, \{\beta_k\}_{k=1}^\infty) \end{aligned} \quad (5.42)$$

This posterior density targets the entire trajectory/sequence of both runtime and regime assignments, and is commonly referred to as the posterior (online) smoothing density [60]

as opposed to the posterior filtering density e.g. (5.35). To begin the particle filter routine, suppose that the posterior smoothing density at time t_{n-1} can be accurately approximated by a large collection of N_p weighted particles:

$$\begin{aligned} & p(\mathbf{s}_{0:n-1}, r_{0:n-1} \mid y_{1:n-1}, \{\beta\}, \{\psi\}) \\ & \approx \sum_{p=1}^{N_p} w_{n-1}^{(p)} \delta_{\mathbf{s}_{0:n-1}, r_{0:n-1}}^{(p)}(s, r) \end{aligned} \quad (5.43)$$

with the weights satisfying $w_{n-1}^{(p)} \geq 0, \forall p$ and $\sum_p w_{n-1}^{(p)} = 1$. Substituting this empirical approximation into the posterior recursion of (5.42), a mixed discrete-continuous distribution can be expressed as:

$$\begin{aligned} & p(\mathbf{s}_{0:n}, r_{0:n} \mid y_{1:n}, \{\beta_k\}_{k=1}^{\infty}, \{\psi_k\}_{k=1}^{\infty}) \\ & \propto \sum_{p=1}^{N_p} \left\{ w_{n-1}^{(p)} \delta_{\mathbf{s}_{0:n-1}, r_{0:n-1}}^{(p)}(s, r) p(r_n \mid r_{n-1}^{(p)}, \mathbf{s}_{n-1}^{(p)}, \{\psi_k\}_{k=1}^{\infty}) \times \right. \\ & \quad \left. p(\mathbf{s}_n \mid \mathbf{s}_{n-1}^{(p)}, r_n) p(y_n \mid y_{1:n-1}, \mathbf{s}_{0:n-1}^{(p)}, \mathbf{s}_n, \{\beta_k\}_{k=1}^{\infty}) \right\} \end{aligned} \quad (5.44)$$

The distribution above contains the point masses of ‘ancestor’ trajectories $\{\mathbf{s}_{0:n-1}^{(p)}, r_{0:n-1}^{(p)}\}_{p=1}^{N_p}$ up to time t_{n-1} , the conditional transition probabilities for new variables \mathbf{s}_n , r_n , and the ‘likelihood’ in the form of Kalman *prediction error decomposition* (PED). The following bootstrap propagation can therefore be used as a single step of the PF to obtain the empirically approximated posterior at time t_n :

1. For a particular particle in the collection with index say $p = \tilde{p}$, sample the new runtime variable $r_n^{(\tilde{p})}$ randomly from Eq. (5.23) conditioned on the ‘ancestor’ runtime $r_{n-1}^{(\tilde{p})}$ and regime assignment $\mathbf{s}_{n-1}^{(\tilde{p})}$ of the previous timestamp.
2. Conditioned on newly sampled runtime $r_n^{(\tilde{p})}$ and the previous regime $\mathbf{s}_{n-1}^{(\tilde{p})}$, sample the new regime assignment vector $\mathbf{s}_n^{(\tilde{p})}$ from Eq. (5.25) or (5.26).
3. Append new variables to the ancestor trajectory giving $\{r_{0:n}^{(\tilde{p})}, \mathbf{s}_{0:n}^{(\tilde{p})}\}$. And re-weight the new trajectory with observation y_n :

$$\tilde{w}_n^{(\tilde{p})} = w_{n-1}^{(\tilde{p})} \times p(y_n \mid y_{1:n-1}, \mathbf{s}_{0:n}^{(\tilde{p})}, \{\beta_k\}_{k=1}^{\infty}) \quad (5.45)$$

The unnormalised weights $\tilde{w}_n^{(p)}$ are re-normalised after all particles have been propagated: $w_n^{(p)} = \tilde{w}_n^{(p)} / \sum_j \tilde{w}_n^{(j)}$. And the weight-correcting PED can be readily computed from the

conditional Kalman filter as:

$$\begin{aligned}
& p(y_n | y_{1:n-1}, s_{0:n}^{(p)}, \{\beta_k\}_{k=1}^\infty) \\
&= \iint p(\mathbf{x}_{n-1} | y_{1:n-1}, \mathbf{s}_{0:n-1}^{(p)}, r_{0:n-1}^{(p)}, \{\beta_k\}_{k=1}^\infty) \times \\
&\quad p(\mathbf{x}_n | \mathbf{x}_{n-1}, s_n^{(p)}, \{\beta_k\}_{k=1}^\infty) p(y_n | \mathbf{x}_n) d\mathbf{x}_n d\mathbf{x}_{n-1} \\
&= \mathcal{N}(y_n | \mu_{y_n}^{(p)}, \Sigma_{y_n}^{(p)})
\end{aligned} \tag{5.46}$$

with

$$\mu_{y_n}^{(p)} = \mathbf{G} \boldsymbol{\mu}_{n|0:n-1}^{(p)} \tag{5.47}$$

$$\Sigma_{y_n}^{(p)} = \mathbf{G} \Sigma_{n|0:n-1}^{(p)} \mathbf{G}^T + \sigma_{\text{obs}}^2 \tag{5.48}$$

As the last component to complete the RBPF algorithm, the content of a particle with index p at time t_n is defined to contain the following: a sequence of regime assignments $\mathbf{s}_{0:n}^{(p)}$; a sequence of runtime $r_{0:n}^{(p)}$; a set of posterior parameters $(\boldsymbol{\mu}_{n|0:n}^{(p)}, \Sigma_{n|0:n}^{(p)})$ for state vector \mathbf{x}_n ; and a normalised weight $w_n^{(p)} \in [0, 1]$. Denote the above content as:

$$P_n^{(p)} = \left\{ \mathbf{s}_{0:n}^{(p)}, r_{0:n}^{(p)}, \boldsymbol{\mu}_{n|0:n}^{(p)}, \Sigma_{n|0:n}^{(p)}, w_n^{(p)} \right\} \tag{5.49}$$

Assuming a finite number of K regimes, **Algorithm 9** summarises the bootstrap RBPF algorithm derived so far.

Unfortunately, the RBPF algorithm cannot yet be used for the state inference task for the proposed model as the issue of the infinite-dimensional regime parameters and assignment vectors has not been addressed yet. Naive application of the *Poly urn* representation described in Section 5.2 would require an additional step of considering whether a new regime needs to be added to the existing mixture to accommodate the diffusion behaviour seen at each timestamp t_n . This would impose a varying dimension of the regime parameters and assignments over the course of the RBPF, which can be troublesome for algorithm implementation. Moreover, the posterior probability of creating a new regime given observation requires a generally intractable integration over the base distribution \mathbb{P}_0 especially in the case of a continuous-time SSM where no conjugate priors can be found. Section 5.4.2 will focus on the solutions to these problems.

5.4.2 Blocked Metropolis-within-Gibbs: DPM inference

Having briefly mentioned the representation, let us now extend the *Poly urn* predictive rule to an **exchangeable** set of subjects i.e. with index $h = 1, \dots, N$. Continuing with the notation, the conditional prior distribution can be written in a non-sequential manner

Algorithm 9 (Bootstrap) Rao-Blackwellised particle filter

Input: Time-stamps $\{t_n\}_{n=1}^N$; observation $\{y_n\}_{n=1}^N$; DPM parameters $\{\beta_k, \psi_k\}_{k=1}^\infty$ and “stick-breaking” length $\{\pi_k\}_{k=1}^\infty$.

Output: A collection of weighted particles $\{P_N^{(p)}\}_{p=1}^{N_p}$

```

1: Initialisation: Create initial particle collection  $P_0^{(p)} = \{\emptyset, 0, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \frac{1}{N_p}\}$  for  $p = 1, \dots, N_p$ 
2: for  $n = 1, \dots, N$  do
3:   if  $n = 1$  then
4:     for  $p = 1, \dots, N_p$  do
5:        $r_1^{(p)} = r_0^{(p)} + (t_1 - t_0)$ 
6:       Sample  $\mathbf{s}_1^{(p)} \sim \text{Categorical}(\{\pi_k\}_{k=1}^K)$ 
7:       Compute  $\{\boldsymbol{\mu}_{1|0:0}^{(p)}, \boldsymbol{\Sigma}_{1|0:0}^{(p)} \mid \beta_{\mathbf{s}_1^{(p)}}\}$  from Eq. (5.37) and (5.38)
8:       Update weight  $\tilde{w}_1^{(p)} = w_0^{(p)} \times \text{PED}(y_1|\dots)$  with Kalman PED (5.46)
9:       Update posterior parameters  $\{\boldsymbol{\mu}_{1|0:1}^{(p)}, \boldsymbol{\Sigma}_{1|0:1}^{(p)} \mid y_1, \beta_{\mathbf{s}_1^{(p)}}\}$  from Eq. (5.40) and (5.41)
10:    end for
11:  else
12:    for  $p = 1, \dots, N_p$  do
13:      Compute hazard function  $H(r_{n-1}^{(p)}, \Delta t_n) \mid \psi_{\mathbf{s}_{n-1}^{(p)}}$  with (5.24)
14:      if  $\text{Uniform}(0, 1) < H(r_{n-1}^{(p)}, \Delta t_n)$  then ▷ Regime switch occurred
15:         $r_n^{(p)} = 0$ 
16:         $\mathbf{s}_n^{(p)} \sim \text{Categorical}(\bar{\boldsymbol{\pi}}_{\mathbf{s}_{n-1}^{(p)}})$  as of (5.29) ▷ Transition excluding self-transition
17:      else ▷ No regime switch
18:         $r_n^{(p)} = r_{n-1}^{(p)} + (t_n - t_{n-1})$ 
19:         $\mathbf{s}_n^{(p)} = \mathbf{s}_{n-1}^{(p)}$ 
20:      end if
21:      Compute  $\{\boldsymbol{\mu}_{n|0:n-1}^{(p)}, \boldsymbol{\Sigma}_{n|0:n-1}^{(p)} \mid \beta_{\mathbf{s}_n^{(p)}}\}$ 
22:      Update weight  $\tilde{w}_n^{(p)} = w_{n-1}^{(p)} \times \text{PED}(y_n|\dots)$ 
23:      Update posterior parameters  $\{\boldsymbol{\mu}_{n|0:n}^{(p)}, \boldsymbol{\Sigma}_{n|0:n}^{(p)} \mid y_n, \beta_{\mathbf{s}_n^{(p)}}\}$ 
24:    end for
25:  end if
26:  Re-normalise weights  $w_n^{(p)} = \tilde{w}_n^{(p)} / (\sum_j \tilde{w}_n^{(j)})$ 
27:  Resample if necessary
28: end for
29: Return:  $\{P_N^{(p)}\}_{p=1}^{N_p}$ 

```

for θ_h given $\theta_{-h} = \{\theta_j : \forall j \neq h\}$:

$$\theta_h \mid \theta_{-h} \sim \left(\frac{\alpha}{\alpha + N - 1} \right) \mathbb{P}_0(\theta_h) + \sum_{k=1}^{K^{(-h)}} \left(\frac{n_k^{(-h)}}{\alpha + N - 1} \right) \delta_{\theta_k^*} \quad (5.50)$$

where θ_k^* for $k = 1, \dots, K^{(-h)}$, are the unique values of θ_{-h} i.e. unique clusters/regimes; and $n_k^{(-h)} = \sum_{i \neq h} \mathbf{1}_{\theta_i = \theta_k^*}$ i.e. the number of other subjects that are assigned to this existing cluster k . Updating the full conditional prior described above with data, it is thus possible to perform a conditional posterior update of the cluster parameters one at a time given the data cluster allocation. The main intuition from the above extension is that posterior inference for the DPM need not follow the sequential order of the data as long as the exchangeability assumption can be satisfied.

However, this is apparently not the case for the SSM as the state vectors are temporally correlated. Furthermore, based on the definition of the DPM, the regime assignment \mathbf{s}_n at time t_n in the running of the RBPF should be sampled from a multinomial conditional posterior:

$$\Pr(\mathbf{s}_n^k = 1 \mid -) \propto \begin{cases} n_k^{(-n)} p(y_n \mid \mathbf{x}_n, r_n, \beta_k, \psi_k, \dots) & k = 1, \dots, K^{(-n)} \\ \alpha \int p(y_n \mid \mathbf{x}_n, r_n, \beta, \psi, \dots) d\mathbb{P}_0 & k = K^{(-n)} + 1 \end{cases} \quad (5.51)$$

where $K^{(-n)}$ is the number of unique regimes that have been created. While the probabilities for $k \in \{1, \dots, K^{(-n)}\}$ can be efficiently computed in the RBPF, the probability of creating a new regime requires an intractable integration over \mathbb{P}_0 . For the same reason, the posterior inference about the regime parameters cannot be performed in closed-form either in the case of a non-conjugate \mathbb{P}_0 .

In order to mitigate the difficulties addressed above, I propose a novel approach of the blocked Metropolis-within-Gibbs (BMwG) algorithm which targets the conditional posterior (ii) of Eq. (5.32) and use it iteratively with the RBPF algorithm.

By marginalising out the base measure \mathbb{P}_0 , the DPM encourages exploration over the prior sample space of the parameters for the possibility of creating a new regime/cluster to accommodate unseen data or diffusion behaviours in the proposed model's case. It is therefore possible to approximate this exploration process with a finite number of i.i.d. samples from \mathbb{P}_0 when the required marginalisation cannot be obtained analytically. Recall the “stick-breaking” construction that has been used throughout this chapter. As the prior weight (i.e. length) associated with each regime decreases stochastically with the increasing index k , it is reasonable to assume that for a sufficiently large K (depending on the value of α), the sum of weights $\sum_{k=K+1}^{\infty} \pi_k$, i.e. the length of the remaining stick, is negligible. Hence by choosing a relatively large K , the breaking process can be *blocked* at the K th piece by letting $V_K = 1$ (i.e. the K th piece takes the entire remaining stick) and obtain an accurate approximation of the DPM with a finite number of potential regimes. As a result, the inference algorithm can tractably perform exploration over the prior sample space and introduce new occupied regimes while fixing the dimension of \mathbf{s}_n (and consequently computation).

In order to learn the regime parameters in the DPM, the inference algorithm needs to target the following conditional posterior density:

$$p(\{\beta_k\}_{k=1}^K, \{\psi_k\}_{k=1}^K \mid X, S, r, Y) \quad (5.52)$$

Although the closed-form posterior cannot be obtained in general, given a sequence of regime assignments $\hat{S} = \hat{\mathbf{s}}_{0:N}$ and runtime $\hat{r} = \hat{r}_{0:N}$ sampled from the RBPF output, the

conditional evidence/likelihood can be readily computed with an efficient Kalman filter:

$$p(Y \mid \hat{S}, \hat{r}, \{\beta_k\}_{k=1}^K) \quad (5.53)$$

and so does the runtime probability:

$$p(\hat{r} \mid \hat{S}, \{\psi_k\}_{k=1}^K) \quad (5.54)$$

A standard Metropolis-Hastings (MH) algorithm can thus be adopted to acquire an empirically represented posterior of the DPM.

While it is possible to perform joint sampling of parameters of all regimes at once, the dimension of the posterior sample space can be extremely high given a large K , which consequently compromises the sampler's performance. I therefore propose to separately sample parameters for each regime i.e. (β_k, ψ_k) conditioned upon all other regimes' parameters $\{\beta_h, \psi_h\}_{h \neq k}$ so that the samplers operate in a ‘‘Gibbs’’ manner (i.e. MwG). For a particular regime k , the MH acceptance probability for the j th MH iteration is:

$$\rho_k = \min \left\{ 1, \frac{q(\beta_k^{j-1}, \psi_k^{j-1}) \mathbb{P}_0(\beta_k^*, \psi_k^*)}{q(\beta_k^*, \psi_k^*) \mathbb{P}_0(\beta_k^{j-1}, \psi_k^{j-1})} \times \frac{p(Y, \hat{r} \mid \hat{S}, \{\beta_h, \psi_h\}_{h \neq k}, \beta_k^*, \psi_k^*)}{p(Y, \hat{r} \mid \hat{S}, \{\beta_h, \psi_h\}_{h \neq k}, \beta_k^{j-1}, \psi_k^{j-1})} \right\} \quad (5.55)$$

where $q(\cdot)$ is the proposal density and is set to the model prior in this work. It is worth noting that with the specific choice of truncated normal and \mathcal{NIG} as duration distribution and parameters' prior respectively, the posterior of ψ can in fact be obtained as a closed-form \mathcal{NIG} distribution. In such a case, it is not necessary to include ψ in the MwG framework. However for generalisation purpose, the MCMC-style inference for ψ is still illustrated here so that the proposed algorithm also works for non-conjugate pairs of emission and prior.

The last step of the DPM update is the sampling of the stick-breaking weight V_k based on the number of transitions assigned to each regime. V_k is independent of other regime parameters and can be sampled analytically as:

$$V_k \sim \text{Beta} \left(1 + n_k, \alpha + \sum_{k'=k+1}^K n_{k'} \right) \quad \text{for } k = 1, \dots, K-1 \quad (5.56)$$

where n_k is the number of transitions assigned to regime k i.e. $n_k = \sum_{n=1}^N \mathbf{1}_{\hat{s}_n^k=1}$; and $V_K = 1$. Line 11 to 30 of **Algorithm 10** provide pseudo-code for the above BMwG algorithm.

For certain applications, it may be preferable to monitor the number of occupied clusters i.e. $\tilde{K} = \sum_{k=1}^K \delta_{[n_k > 0]}$ and increase the value of K once \tilde{K} is close to the blocked limit. This extra step can ensure the accuracy of the approximation. However, such a

step was not implemented in the experiments in this chapter and the empirical results also suggest that it is not necessary for the considered applications.

5.4.3 *Particle-MCMC: an iterative framework*

Although it seems that an iterative algorithm for both sequential state inference and posterior parameter learning is available, the two steps of the alternating algorithm each targets a mean-field factorised conditional posterior as described in Eq. (5.32) instead of the original joint posterior. However, simply replacing one step of the Gibbs sampler with the sampling from an PF approximation does not admit the correct joint posterior (5.32) as the invariant density [64].

In order to mitigate this potential drawback, I adopt the general framework of particle Gibbs (PG) algorithm first proposed in [64] as a type of PMCMC algorithm. Applications of PMCMC have arisen in various areas such as finance [121], biology [122] and epidemiology [123].

The PG algorithm targets an extended distribution of all random variables involved in the model which is the same as the joint posterior distribution in (5.32) prior to the factorisation. It allows iterative Gibbs-like updates while admitting an invariant target density by running the sequential inference step with a *conditional* PF. The conditional PF is similar to the standard PF but is different in that it additionally admits a pre-specified reference trajectory i.e. $\{\hat{S}, \hat{r}\}$ and guarantees its survival throughout all resampling steps, whereas the remaining $N_p - 1$ particles are generated and resampled as usual. With the assumption that the reference trajectory of the PG sampler does not dominate other particles during resampling, it has been proven in [124] that for a N_p large enough, the Markov kernel of PG is uniformly ergodic with the target density invariant.

Hence in the proposed model, one can simply replace the RBPF algorithm with a conditional RBPF which takes the sampled trajectory $\{\hat{S}(i-1), \hat{r}(i-1)\}$ from previous PG iteration as the reference trajectory and ensures the iterative algorithm converges towards the desired posterior distribution. **Algorithm 10** shows the complete iterative algorithm for the inference of the proposed state-space regime-switching model with infinite mixture dynamics.

One may notice that the algorithm contains a burn-in period for the MwG sampler in the regime update. The original PG algorithm of [64] assumes that the parameters can be sampled directly from the conditional posteriors (as would be the case for ψ in the proposed model). When using MH samplers, I suggest to include the burn-in period to ensure that the updated parameters are representative/converged samples of the target conditional posterior. It can certainly be argued that burn-in is not strictly necessary as long as enough iterations are allowed for the PG algorithm (e.g. a large enough N_{PG}). However with the conditional RBPF being the computational bottleneck, experimental

Algorithm 10 BMwG and RBPF in PG framework

Input Time-stamps $\{t_n\}_{n=1}^N$; and the corresponding scalar observation $\{y_n\}_{n=1}^N$

- 1: **Initialisation:** Initialise regimes and parameters $(\beta_k, \psi_k) \stackrel{\text{i.i.d.}}{\sim} \mathbb{P}_0$, $V_k \stackrel{\text{i.i.d.}}{\sim} \text{Beta}(1, \alpha)$, for $k = 1, \dots, K$; $V_K = 1$.
- 2: **for** $i = 1, \dots, N_{\text{PG}}$ **do**
- 3: Compute regime probability $\pi_k = V_k \prod_{l < k} (1 - V_l)$, for $k = 1, \dots, K$
- 4: **if** $i = 1$ **then**
- 5: Run RBPF with parameters $\{\beta_k, \psi_k\}_{k=1}^K$
- 6: **else**
- 7: Run conditional RBPF with parameters $\{\beta_k, \psi_k\}_{k=1}^K$ conditioned on $\{\hat{S}(i-1), \hat{r}(i-1)\}$
- 8: **end if**
- 9: Sample $\{\hat{S}(i), \hat{r}(i)\} \sim \text{Categorical}(\{w_N^{(p)}\}_{p=1}^{N_p})$
- 10: Calculate $n_k = \sum_{n=1}^N \mathbf{1}_{\hat{s}_n^k=1}$, for $k = 1, \dots, K$
- 11: **for** $k = 1, \dots, K$ **do**
- 12: Draw $V_k \sim \text{Beta}(1 + n_k, \alpha + \sum_{k'=k+1}^K n_{k'})$
- 13: **if** $n_k = 0$ **then** ▷ An “unoccupied” regime
- 14: Draw $(\beta_k^*, \psi_k^*) \sim \mathbb{P}_0$
- 15: Update the k th regime parameters: $(\beta_k, \psi_k) = (\beta_k^*, \psi_k^*)$
- 16: **else** ▷ For an “occupied” regime
- 17: **for** $j = 1, \dots, N_{\text{burn}}$ **do**
- 18: Propose a sample $(\beta_k^*, \psi_k^*) \sim q(\beta, \psi)$
- 19: Run KF conditioned on $\{\hat{S}(i), \hat{r}(i), \{\beta_h, \psi_h\}_{h \neq k}, \beta_k^*, \psi_k^*\}$
- 20: Compute MH acceptance ratio ρ_k from (5.55)
- 21: **if** $\text{Uniform}(0, 1) < \rho_k$ **then**
- 22: $(\beta_k^j, \psi_k^j) = (\beta_k^*, \psi_k^*)$
- 23: **else**
- 24: $(\beta_k^j, \psi_k^j) = (\beta_k^{j-1}, \psi_k^{j-1})$
- 25: **end if**
- 26: **end for**
- 27: Update the k th regime parameters: $(\beta_k, \psi_k) = (\beta_k^{N_{\text{burn}}}, \psi_k^{N_{\text{burn}}})$
- 28: **end if**
- 29: **end for**
- 30: Set $V_K = 1$
- 31: **end for**

results in Section 5.5 demonstrate that the PG convergence can be achieved with much fewer iterations (and thus fewer runs of the conditional RBPF) by allowing this local convergence of the DPM. This can be an interesting and important theoretical line of research, but detailed proof is not studied in this work.

5.4.4 Deterministic filtering and optimal resampling

Performance of the sequential inference algorithm (i.e. the RBPF) is usually critical to the overall performance of PMCMC. Different resampling schemes have been proposed in literature [124, 125] to tackle the inherent path degeneracy issue that is commonly regarded as a limitation of the PG algorithm. The (conditional) RBPF algorithm employed efficiently reduces the dimension of the sample space by marginalising the state vectors x_n . However with DPM as the prior for regime parameters and assignments, it is still possible to have a high-dimensional \mathbf{s}_n when a large value of K is chosen to block the

“stick-breaking” process. Moreover, unlike some other switching models where drastic changes are directly observable from the data, regime switches in diffusion dynamics are usually subtle based on a single observation and may only become distinguishable after a few more observations of diffusion. Hence to further improve the RBPF accuracy, the deterministic filtering scheme with an optimal resampling step originally proposed in [21] is adopted here.

However before going into details, I want to point out that the sequence of runtime $r_{0:n}$ can be deterministically calculated from a given sequence of regime allocations $\mathbf{s}_{0:n}$ (and timestamps $\{t_n\}_{n=1}^N$) as the proposed semi-Markov switching feature only tries to identify the intervals where regime changes occur. Thus, it is possible to take full advantage of the discrete nature of the regime assignments \mathbf{s}_n and perform deterministic filtering to achieve a thorough exploration of the discrete sample space at each propagation. Suppose there is a set of N_p particles $\{P_{n-1}^{(p)}\}_{p=1}^{N_p}$ from time t_{n-1} , the deterministic filtering scheme propagates the particles at time t_n as follows:

- (1) For each particle $P_{n-1}^{(p)}$ from t_{n-1} , generate a set of K new particles (*descendants*) $\{P_n^{*(pk)}\}_{k=1}^K$, one for each possible regime of diffusion where:

$$\begin{aligned} \mathbf{s}_n^{(pk)} &= [0 \dots 1_k \dots 0]^T \\ r_n^{(pk)} &= \begin{cases} r_{n-1}^{(p)} + (t_n - t_{n-1}) & \text{if } \mathbf{s}_n^{(pk)} = \mathbf{s}_{n-1}^{(p)} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

- (2) Compute the predictive parameters $(\mu_{n|0:n-1}^{(pk)}, \Sigma_{n|0:n-1}^{(pk)})$ of state vector x_n conditioned on $\mathbf{s}_n^{(pk)}$ from (5.37) and (5.38).
- (3) Compute the unnormalised weights of all KN_p particles. The weight of particle $P_n^{*(pk)}$, which is the descendant of $P_{n-1}^{(p)}$, is:

$$\begin{aligned} \tilde{q}_n^{(pk)} &= w_{n-1}^{(p)} \times p(r_n^{(pk)} | r_{n-1}^{(p)}, \{\psi_k\}_{k=1}^K, \mathbf{s}_{n-1}^{(p)}) \times \\ &\quad p(\mathbf{s}_n^{(pk)} | \mathbf{s}_{n-1}^{(p)}, r_n^{(pk)}) \times p(y_n | y_{1:n-1}, s_{0:n-1}^{(p)}, \mathbf{s}_n^{(pk)}, \{\beta_k\}_{k=1}^K) \end{aligned} \quad (5.57)$$

where all probabilities can be numerically evaluated using Eq. (5.23), (5.25), (5.29) and (5.46).

- (4) Perform Kalman update for posterior filtering parameters $(\mu_{n|0:n}^{(pk)}, \Sigma_{n|0:n}^{(pk)})$, and normalise weights w_n

(5) Normalised the weights of all KN_p particles:

$$q_n^{(pk)} = \frac{\tilde{q}_n^{(pk)}}{\sum_{p'=1}^{N_p} \sum_{k'=1}^K \tilde{q}_n^{(p'k')}} \quad (5.58)$$

By the end of stage (5), the algorithm would end up with a total of KN_p weighted particles. Resampling is thus necessary at each propagation to avoid an exponentially growing particle number and computational cost. In order to maintain a constant N_p particles at the start of each propagation, a resampling scheme is needed to approximate a discrete probability mass function with KN_p support by a *stochastic* probability mass function with a support no more than N_p points. An unbiased optimal scheme is outlined and proved in [21], which minimises the expected squared error between the discrete probability mass function and the stochastic approximation. **Algorithm 11** shows the pseudo-code for this optimal resampler.

The stratified sampler (used in **line 13** with details provided in **Appendix 5.A**) ensures that either none or one copy of each particle is resampled. As a result, this optimal resampling algorithm has the interesting feature of having only distinct particles (i.e. no multiple copies of the same particle) in the returned N_p collection if distinct particles were obtained at the previous timestamp i.e. t_{n-1} . This feature is particularly attractive under the settings where discrete proposals take place *deterministically*, as each particle will probe the future exhaustively and the same information can be obtained from a single particle with a higher weight compared to multiple identical particles with split weights. However in practice, this proposed inference algorithm still attempts to preserve some copies of particles in the resampled collection by using a relatively large N_p so that more diverse (low-weight) trajectories may be retained in the stratified sampling stage.

As proved in [124], the resampling schemes for the PG algorithm need to be marginally unbiased while ensuring the survival of the reference trajectory $\hat{S}(i-1)$. A simple adaptation of the optimal resampling scheme is hence made to keep at least one copy of the reference trajectory in the resampled collection. Since deterministic filtering explores all current switching possibility, as long as $\hat{s}_{0:n-1}(i-1)$ exists in the resampled collection Φ_{n-1} at time t_{n-1} , $\hat{s}_{0:n}(i-1)$ is guaranteed to exist in the KN_p samples before resampling at time t_n . The adaptation of the original resampling scheme to the PG algorithm is thus as follows: run a check on whether $\hat{s}_{0:n}(i-1) \in \Phi_n$ at the end of **Algorithm 11**, if not then set the last particle $P_n^{(N_p)} = \{\mathbf{s}_{0:n}^{(pk)}, r_{0:n}^{(pk)}, \boldsymbol{\mu}_{n|0:n}^{(pk)}, \boldsymbol{\Sigma}_{n|0:n}^{(pk)}, 1/c \mid \mathbf{s}_{0:n}^{(pk)} = \hat{\mathbf{s}}_{0:n}(i-1)\}$. As the particle with index N_p will always be sampled from the stratified sampling scheme and the scheme samples either none or one copy of each input particle, *marginal unbiasedness* of the optimal resampler remains intact.

Algorithm 11 Optimal Resampling

Input: A set of $M (= KN_p)$ particles before resampling at time t_n with normalised weights $\sum_{j=1}^M q_n^{(j)} = 1$.

Output: A collection of resampled N_p particles Φ_n with weights $\{w_n^{(p)}\}_{p=1}^{N_p}$.

```
1:  $\Omega \leftarrow \emptyset$ 
2: Calculate  $c$ , the unique solution to  $N_p = \sum_{j=1}^M \min(c w_n^{(j)}, 1)$ 
3: Set  $p = 1$ 
4: for  $j = 1, \dots, M$  do
5:   if  $q_n^{(j)} \geq 1/c$  then
6:     Admit particle  $j$  into the set  $\Phi_n$ 
7:      $w_n^{(p)} = q_n^{(j)}$ 
8:      $p = p + 1$ 
9:   else
10:    Admit particle  $j$  into the set  $\Omega$ 
11:   end if
12: end for
13: Perform stratified sampling algorithm [126] on  $\Omega$  to sample  $(N_p - |\Phi_n|)$  particles.
14: Admit the sampled  $(N_p - |\Phi_n|)$  particles to  $\Phi_n$  with identical weights  $w_n^{(p')} = 1/c$  for  $p' = p, \dots, N_p$ 
15: Return:  $\Phi_n, \{w_n^{(p)}\}_{p=1}^{N_p}$ 
```

5.5 Results and discussions

In this section, I present three sets of results obtained by applying the proposed model on three different datasets. The first dataset is synthesised with a matching dynamic model but with parameters unknown to the inference. The second dataset consists of GPS position recordings of movements of a wild animal (baboon), while the last set of data is the market mid-price of a high-frequency FOREX market where the actual physical dynamics of the system can not be clearly identified or defined.

All experiments are run with the same Langevin construction of the SSM as defined in (5.13) with unknown but static parameters. However, I stress again that the general framework of the proposed model is capable of adopting various forms of diffusion models that are best suited for the applications of interest. I show later in the next section that this generalisation may even be extended to combinations of diffusion models.

5.5.1 Synthetic data

In order to confirm whether the proposed model and the corresponding inference algorithm can perform as expected, the model is first tested on a set of 1-D synthetic data. The dataset is simulated using a matching Langevin SSM governed by three distinct diffusion regimes:

1. A long-lasting (average duration of 30s) dynamic for relatively stationary diffusion with $\theta_1 = -0.5$ and $\sigma_1 = 0.5$;
2. A short-lived (average duration of 15s) dynamic that is likely to have trend with

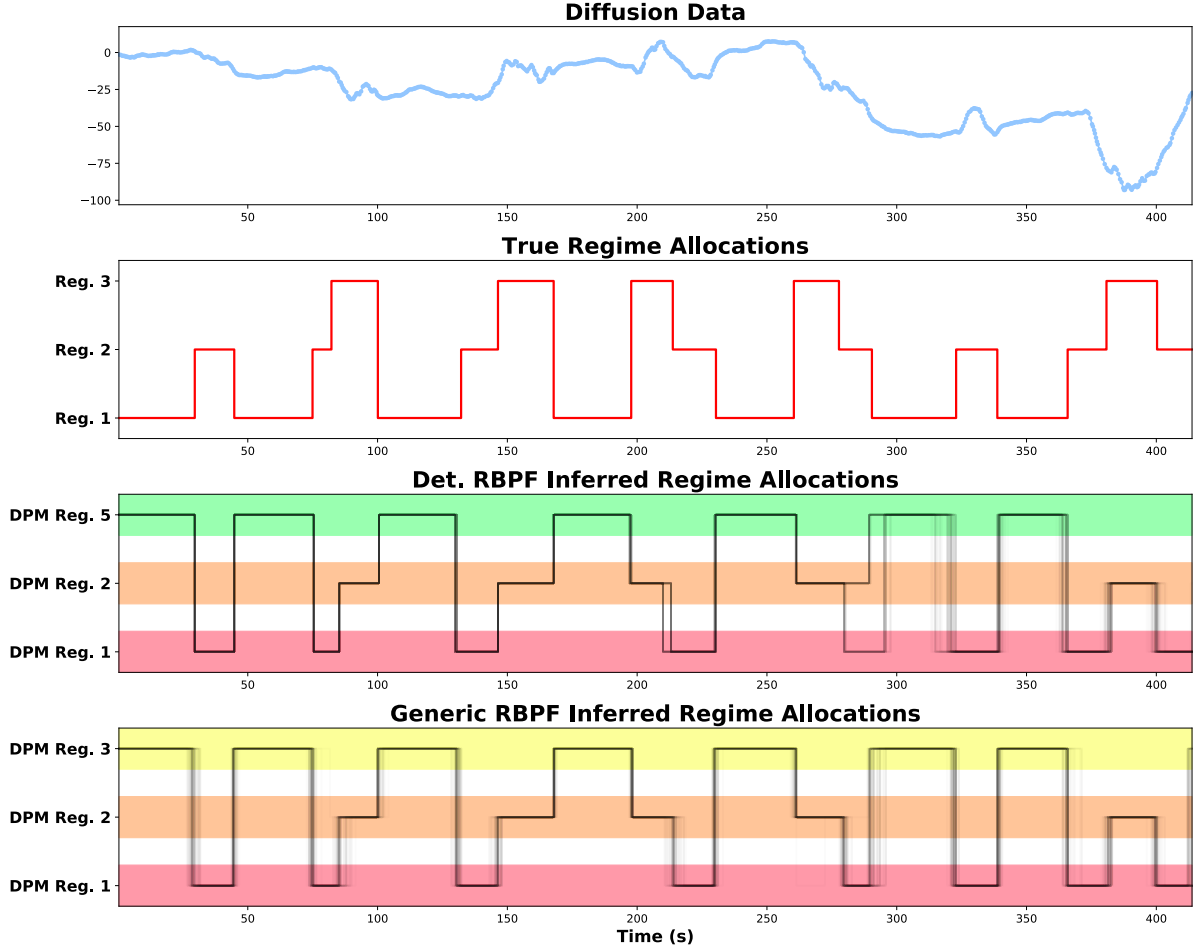


Figure 5.5.1: Regime allocation results obtained with both the deterministic RBPF (proposed) and the generic RBPF. The top plot shows the simulated diffusion data with true regime allocations that are indicated in red lines in the second plot. The bottom two plots show the inferred regime allocations after 20 PG iterations. Each allocation trajectory is plotted with the same transparency and thus the blurriness indicates the uncertainty of regime estimations.

$$\theta_2 = -0.2 \text{ and } \sigma_2 = 2.0;$$

3. And a medium-duration (average duration of 20s) fluctuating dynamics with $\theta_3 = -1.0$ and $\sigma_3 = 5.0$

Switching between regimes, 832 observation points with $\sigma_{\text{obs}} = 0.1$ spread across 20 diffusion intervals are synthesised on an irregular time grid. The top panel of **Figure 5.5.1** shows the synthesised observations and the corresponding true regime assignments are shown in the middle panel.

Fairly diffuse priors are used for the DPM base distribution \mathbb{P}_0 with hyperparameters:

- $\mu_\theta = -0.6$, $\sigma_\theta^2 = 1.0$
- $\alpha_\sigma = 1.5$, $\beta_\sigma = 0.5$
- $m_\psi = 20$, $\nu_\psi = 0.1$, $\alpha_\psi = 3.0$, $\beta_\psi = 20$
- stick-breaking $\alpha = 0.5$ and the maximum number of regimes $K = 10$

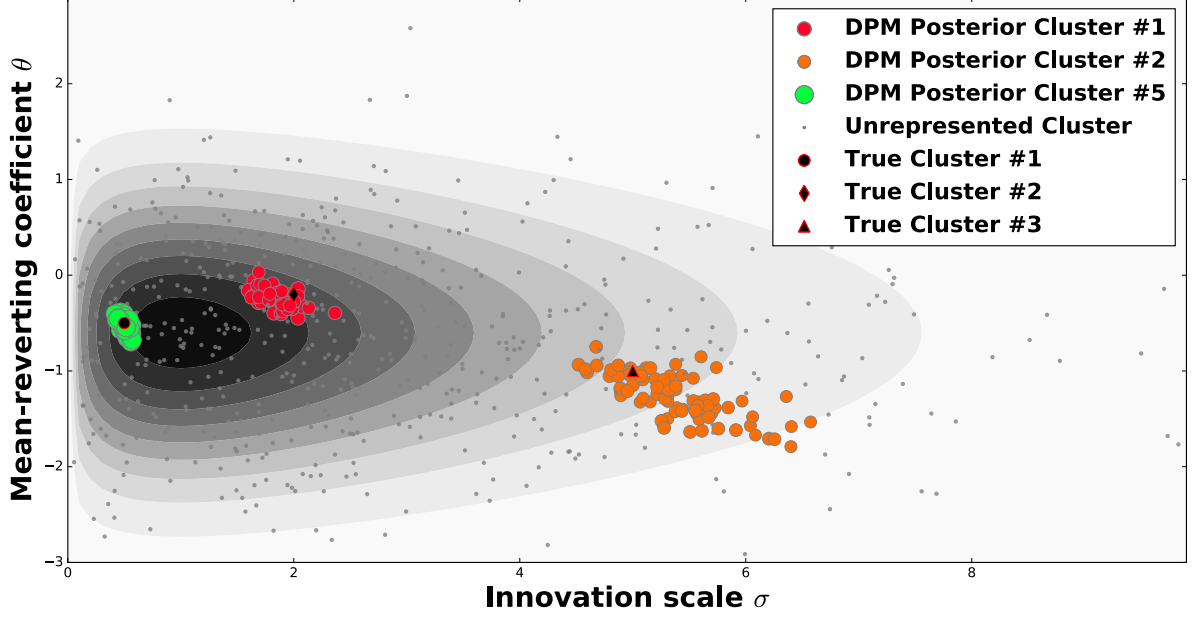


Figure 5.5.2: The figure shows the posterior samples of the dynamics-controlling regime parameters in each cluster of DPM. The sizes of the dots are proportional to the number of data points assigned to each regime with colour codes consistent with the regime plot. The unoccupied clusters are plotted in grey. Contour shows the prior.

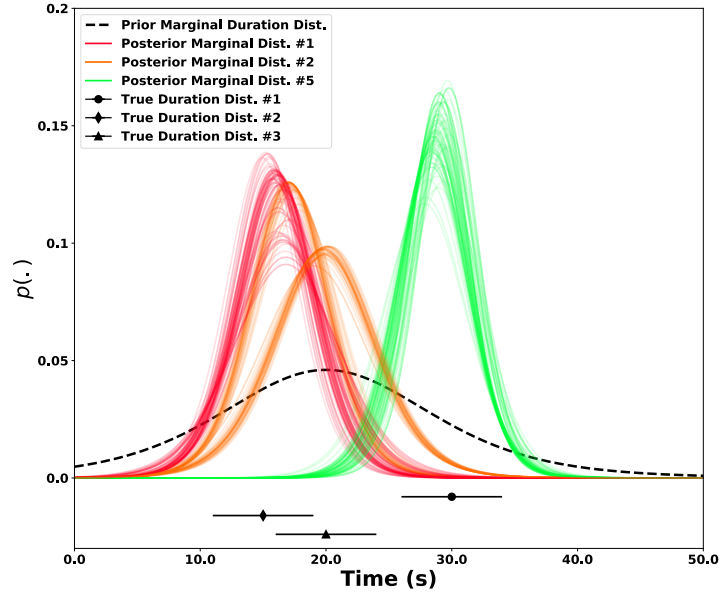


Figure 5.5.3: The figure shows the posterior (predictive) marginal duration distributions for represented regimes. The error bars indicate the mean and ± 2 std. of the true duration distribution.

The PG algorithm was run for 100 iterations ($N_{PG} = 100$) and with 100 particles ($N_p = 100$) and 100 MH burn-in ($N_{burn} = 100$). **Figure 5.5.1** shows the inferred regime assignments $\mathbf{s}_{0:N}$ obtained after 20 PG iterations from both the deterministic RBPF algorithm (with stratified resampling) and the generic RBPF (with IRS). Despite the exchange of regime orders, it is clear that the posterior switching patterns matches accurately with the true

allocations (in red in the second panel), except for a short period around 280s, where the diffusion exhibits a trend that is more likely to occur in regime 2 in spite of being in the fluctuating regime 3. In this case, the deterministic RBPF has shown uncertainty and recognised the possibility of both. Comparing between the two variants of the RBPF in the bottom two panels, the generic RBPF generally shows more uncertainties on the switching times as can be seen from the blurry vertical lines. On the other hand, the deterministic RBPF explores more thoroughly the discrete regime space and hence provides more accurate switching time estimations.

Figure 5.5.2 shows the posterior sample clusters of regime parameters $\{\theta_k, \sigma_k\}_{k=1}^K$ for the final 80 iterations using the same colour-code as the allocation plot. The sizes of the scatter points are proportional to the number of diffusion transitions assigned to each regime. Apparently, the three “occupied” regimes have all their posterior samples concentrating around the true values, whilst the unoccupied regimes have their samples (shown in grey) thoroughly exploring the prior parameter space of \mathbb{P}_0 (shown as the contour). An interesting observation is that as σ increases, the posterior cluster tends to become more diffuse. This may be caused by the growing stochasticity of the dynamic model with the increasing σ , which consequently increases the posterior variance.

As the posteriors of duration parameters $\{\psi\}_{k=1}^K$ can be obtained in closed-form conditioned on runtime, the inference results shown in **Figure 5.5.3** are the marginal *predictive* distributions for regime duration. The figure shows the (predictive) truncated *student-t* distributions of the three inferred DPM regimes over the final 80 iterations, the prior marginal distribution and the 95% confidence intervals of the three true distributions. Note that for the learning of ψ , each diffusion interval is treated as one data point. Hence with a total of only 20 intervals simulated in this dataset, the learning of duration parameters for each regime is likely to have uncertainties. However, a close fit between each marginal predictive distribution and true distribution can still be observed.

To provide a clear view of the PG convergence, the weighted log-PED and mean-squared error (MSE) are monitored across all 100 PG iterations for both algorithms with and without MH burn-in as discussed in Section 5.4.3. The results are plotted in **Figure 5.5.4**. Clearly with MH burn-in, the PG algorithm is able to converge quickly within 20 iterations. Whilst without burn-in, the convergence seems to be much slower and is possibly not reached with 100 iterations as slightly worse values of log-PED and MSE are achieved.

In summary, this experiment on synthetic data shows that the proposed model and inference algorithm are able to demonstrate the desirable and anticipated features. The posterior learning of DPM not only identifies the correct number of regimes based purely on diffusion behaviours in data, but also accurately estimates the regime parameters without overfitting. Furthermore, the proposed PG algorithm with burn-in BMwG has demonstrated fast convergence in evaluation metrics.

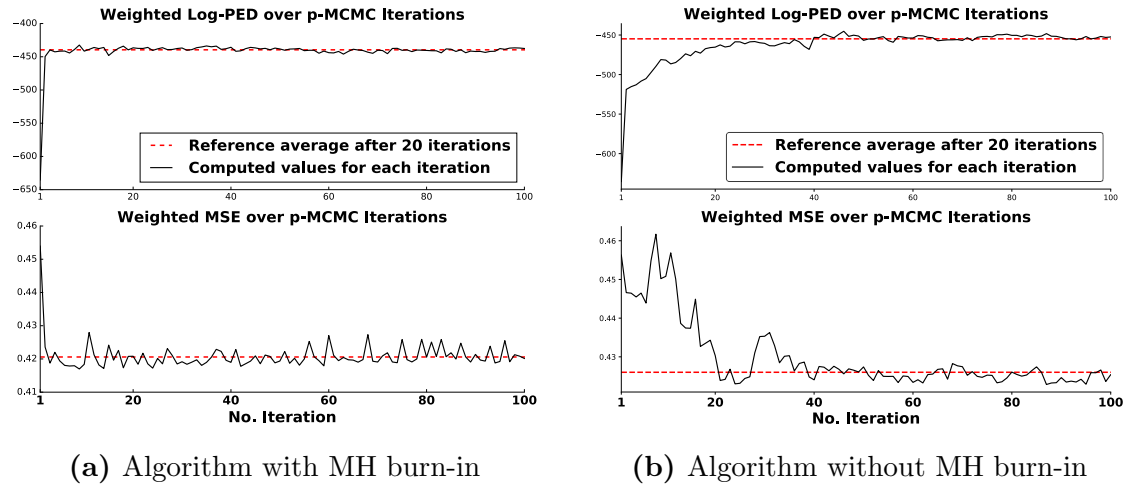


Figure 5.5.4: Figure shows the weighted log-PED and MSE evaluated across all 100 PG iterations for both algorithms with and without Metropolis-Hastings burn-in.

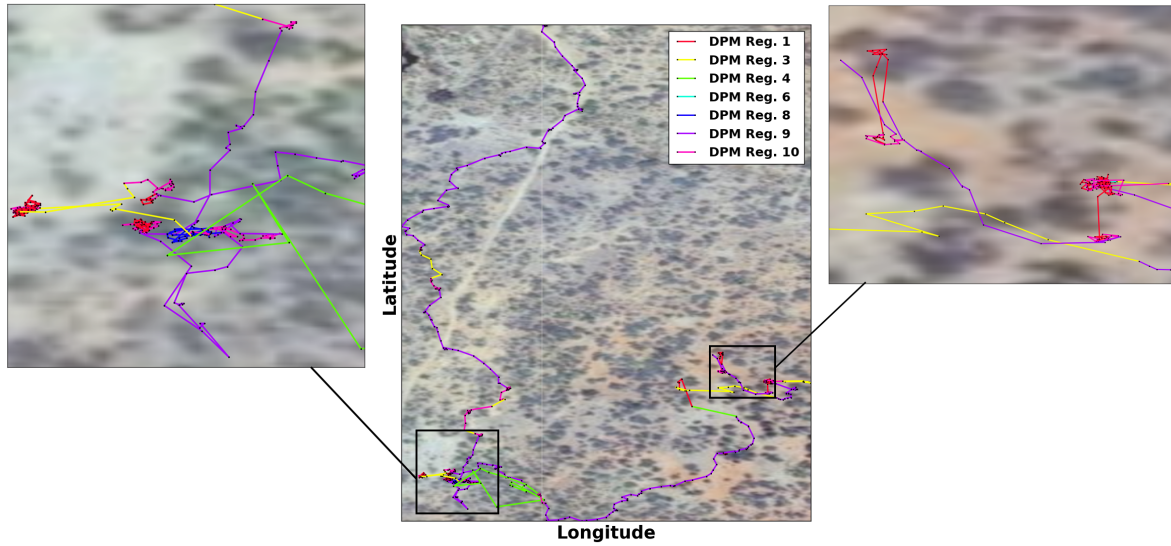
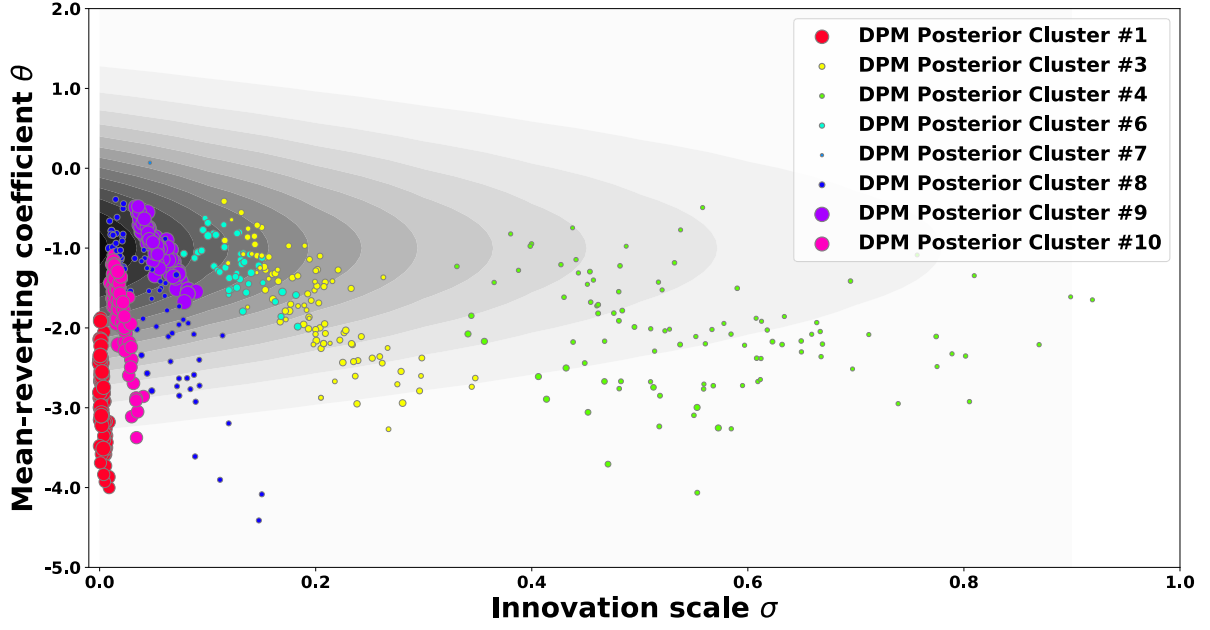


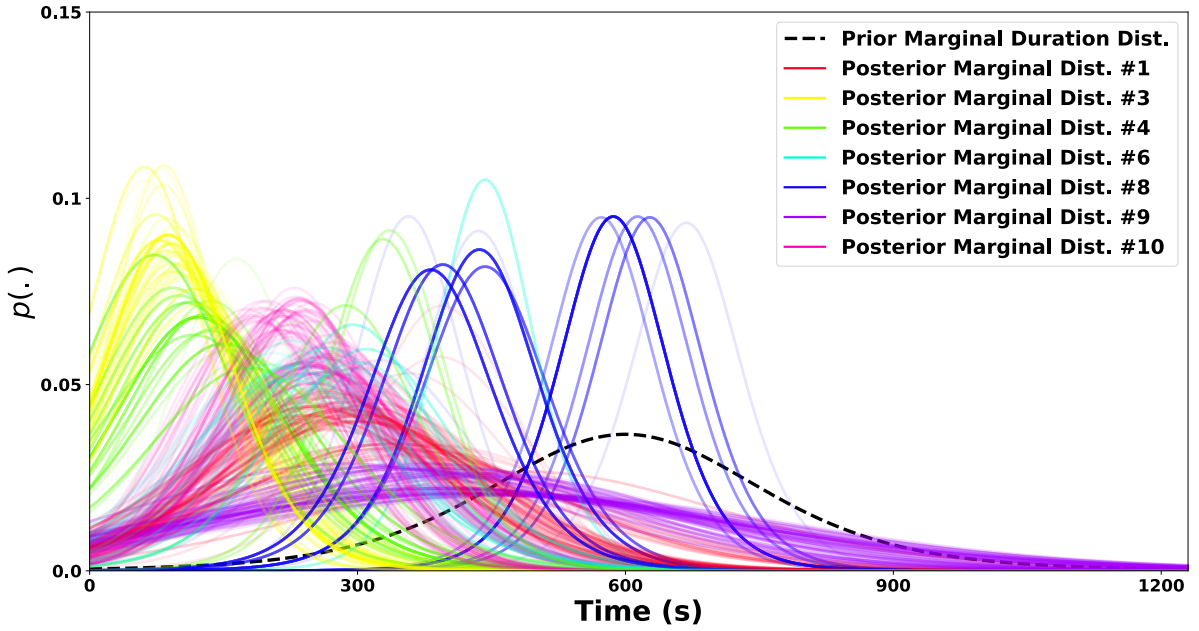
Figure 5.5.5: The figure shows the 2D GPS trail with two zoomed views of the tracked baboon for a duration of 4 hours. The trail starts from top left and finishes around bottom right. The colour of each segment is determined by the regime with highest posterior probability at each timestamp.

5.5.2 *Animal GPS data*

The SSM has been a popular choice for modelling real-life movements of physical objects such as manoeuvring ships [127, 87], animals [128] and humans [129]. Applications of SSM to these systems usually involves fairly uninformative prior dynamics with tuned hyperparameters so that various movement patterns of the tracked objects can be accommodated in the observational model. However, such prior dynamics can hardly provide any retrospective insights to the behaviours of the tracked object which can be important for the studies of these systems. Hence before experimenting on the LOB price data, I



(1) Posterior DPM clusters for (θ, σ)



(2) Predictive marginal duration distributions

Figure 5.5.6: Plots showing posterior learning results of DPM on baboon GPS data: (1) shows the posterior samples of (θ, σ) for each “represented” regime obtained in the final 100 PG iterations; (2) shows the predictive marginal distributions for duration computed for the same 100 iterations with the \mathcal{NIG} posteriors marginalised.

apply the proposed model to a set of baboon GPS data without changing the original model setup.

The dataset for this experiment contains the longitude (x) and latitude (y) GPS recordings of a single baboon taken with intervals of 15 seconds for 4 hours from 7:00am to 11:00am [130, 131]. However, for simplicity and consistency of the model, the sequential

inference and the parameters learning are only performed on the normalised (zero-mean, unit-variance) 1-D longitude (x) data (i.e. ignoring the latitude data/movements).

Inferred 200 PG iterations, the posterior probability of regime assignments $\{\mathbf{s}_n\}_{n=1}^N$ is calculated based on the final 100 converged samples. And **Figure 5.5.5** shows the 2D trail (i.e. data/observations) whose segments are marked with the regimes of the highest posterior probability at each timestamp. Correspondingly, **Figure 5.5.6** shows two plots of the inference results of the parameters for those “occupied/represented” regimes. Focusing firstly on cluster/regime #9 (purple), the diffusion parameters have shown moderate level of velocity innovation σ and relatively low intention to stop i.e. less negative θ , while the regime duration is able to take a wide range of values. Such a regime represents baboon activities that involve constant movements at medium speed e.g. migration, foraging. On the other hand, regime #1 (red) shows extremely low velocity innovation and high velocity resistance, which indicates that the tracked object was barely moving i.e. resting. Other notable regimes like #3 (yellow) and #4 (green) have shown highly “volatile” movements in a short period of time which may represent hunting or chasing behaviours of the baboon. The posterior samples of these two regimes are fairly scattered due to the infrequent appearances of similar activities in the 4 hours period.

This experiment on baboon GPS data has shown that the proposed model is able to provide reasonable identification of baboon behaviours based only on one-dimensional movements. It is reasonable to conjecture that the performance will be further improved with 2-D construction of SSM. Moreover, the learning of dynamics performed by the proposed model is data-oriented with minimal prior/human assumptions, which can provide additional insights to the study of animal behaviours.

5.5.3 FOREX market price

As opposed to diffusion in physical systems, different dynamic regimes of price diffusion in high-frequency financial markets are even harder to identify while inappropriate prior assumptions on models can easily lead to bad investment decisions and even more serious consequences for algorithmic trading.

I therefore apply the proposed model to price data in the EUR-USD FOREX market, which aims to learn the dynamics-controlling parameters purely based on diffusion data. The model is applied on two sets of price data each contains 1-D price diffusion with irregular updates collected on the 3rd of September, 2015. I run the complete model and inference algorithm on the first set of data (05:00AM – 05:10AM ET, 924 observations) for training and parameter learning. After that, the model with learned parameters is run in online prediction mode (i.e. non-iterative inference using only the RBPF) on the second set of data (05:10AM – 05:15AM ET, 427 observations) for validation. The data in both sets are normalised to have zero mean and unit standard deviation so that general priors can be used instead of the ones tuned to a specific market or a period of time.

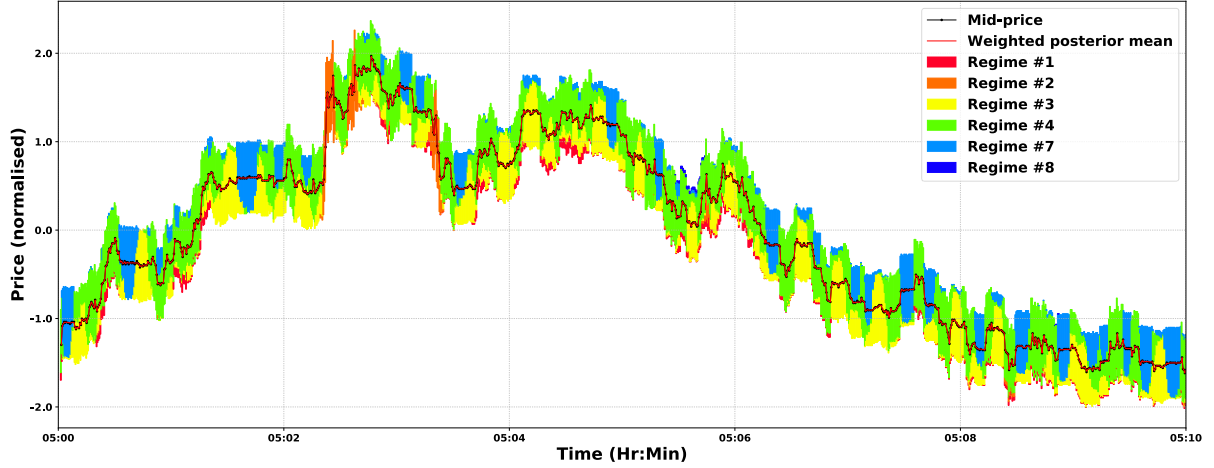


Figure 5.5.7: The figure shows the posterior weighted mean of $x_{1,t}$ and posterior regime allocation inferred by the proposed model on 10 minutes of mid-price (training) data from the FOREX market. The lengths of the colour bars are proportional to the posterior multinomial probabilities.

Running the PG algorithm in full, **Figure 5.5.7** shows the posterior regime assignment probabilities for each timestamp empirically estimated from the final 100 iterations of a total 200 PG iterations. The proportion of the colour bars at each timestamp represents the posterior multinomial probabilities. Meanwhile, the learned diffusion parameters (θ, σ) are shown in **Figure 5.5.8** with a comparison to the values learned for a single-regime model (i.e. a standard SSM). It is obvious that the posterior DPM has identified mainly four salient regimes in the 10-minute interval. Regime #7 (blue) shows the dynamics for stationary periods where the mid-price remains roughly at a constant level (low σ). On the other hand, regime #2 (orange) allows drastic changes/jumps in the price which are rare and short-lived. Trading around “orange” period is likely to have higher risk because of the high σ , but the relative small reversion (less negative θ) indicates that there may be momentum/trend to exploit. Diffusion in regime #4 is the most common in this FOREX dataset and exhibits typical stochastic behaviours showing trends as well as fluctuations. The yellow regime serves more like a transitional regime or middle ground between #4 and #7. In contrast, the parameters of a standard single-regime model (shown as black crosses in **Figure 5.5.8**) over-generalises different behaviours in the data and results an almost average of the identified regimes of the DPM.

In order to show that the proposed model is not overfitting the training data by introducing unnecessary regimes, validation is carried out on the 5-minute interval that immediately follows the training data. A prediction plot is shown in **Figure 5.5.9** for both multi-regime and single-regime models. Additionally, **Figure 5.5.10** quantitatively compares the prediction accuracy of the two models by computing the marginal joint

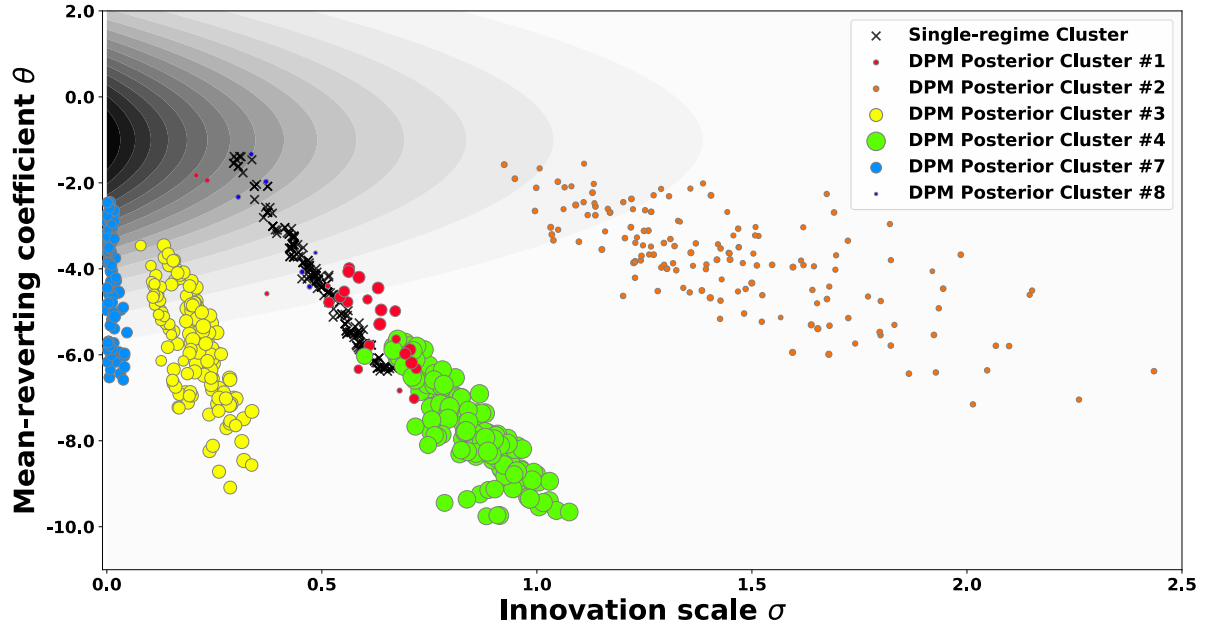


Figure 5.5.8: The figure shows the posterior parameters (θ, σ) learned from the 10-minutes mid-price data. The coloured circles are the posterior parameter samples from the DPM; whilst the black crosses are the posterior parameter samples learned on a single-regime model (i.e. a standard SSM).

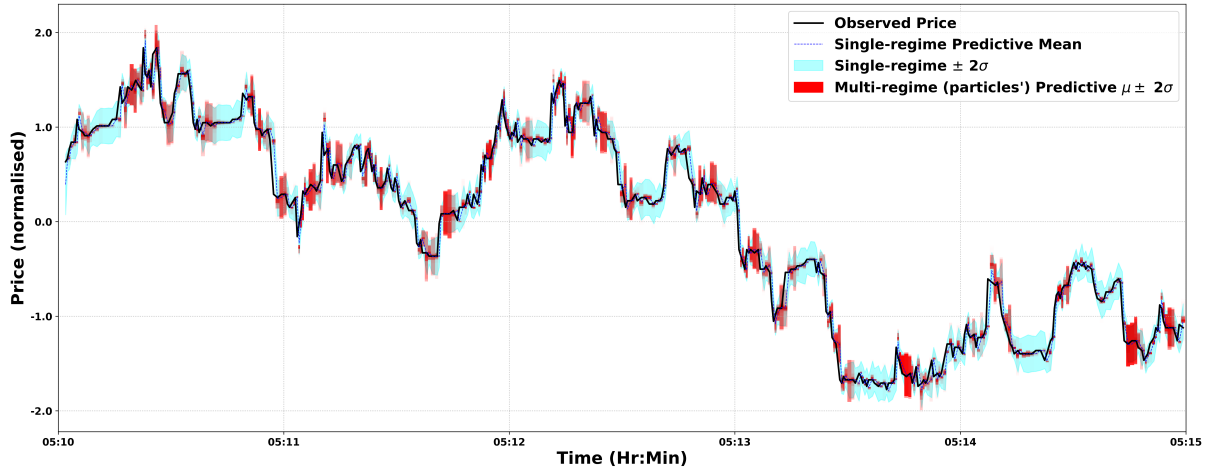


Figure 5.5.9: The figure shows predictive results of both the proposed multi-regime model and the single-regime model with one converged set (e.g. the final iteration) of learned parameters for a duration of 5 minutes that immediately follows the training data. The blue curves and the cyan region indicate the predictive mean and 95% ($\pm 2\sigma$) confidence interval of the single-regime model inferred by the Kalman filter. The red patches indicate the predictive mean and 95% confidence interval for each particle in the multi-regime model inferred by the deterministic RBPF; the opacity is proportional to the particle weight.

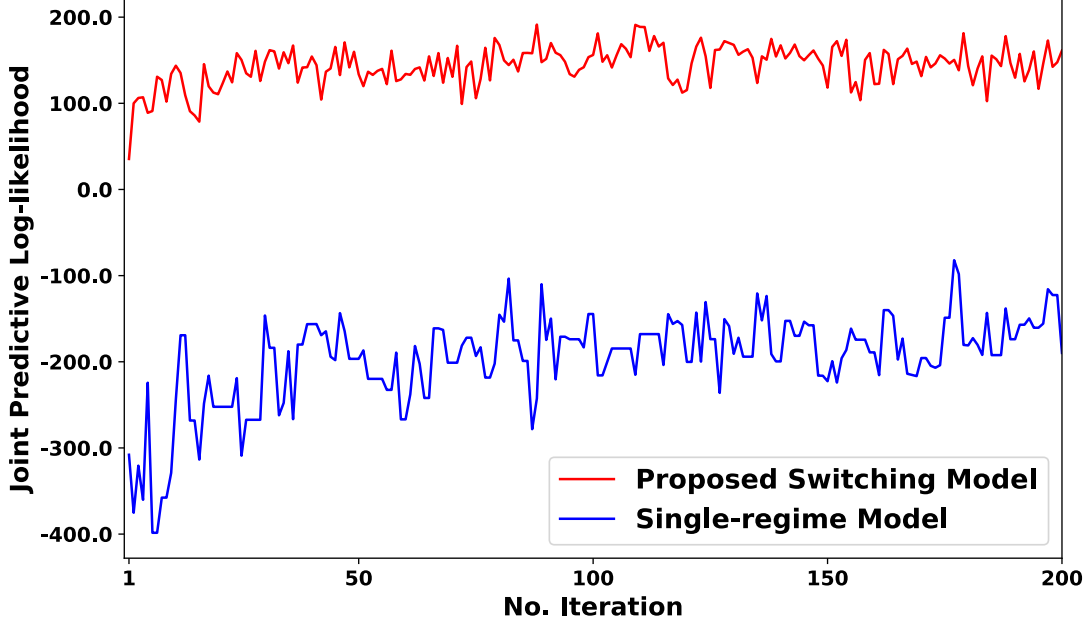


Figure 5.5.10: The figure shows the *log* of marginal joint PED for the 5-minutes validation dataset obtained from both single-regime and multi-regime models.

PEDs using posterior parameters obtained each PG iteration during training phase:

$$\begin{aligned}
 \hat{p}(Y) &= \prod_{n=1}^N \hat{p}(y_n | y_{0:n-1}) \\
 &= \prod_{n=1}^N \left\{ \sum_{p=1}^{N_p} w_{n-1}^{(p)} \mathcal{N}\left(y_n | \mu_{y_n}^{(p)}, \Sigma_{y_n}^{(p)}\right) \right\}
 \end{aligned} \tag{5.59}$$

It can be seen from **Figure 5.5.9** that the standard single-regime model assigns uniformly large uncertainties to all periods of the diffusion; while the proposed model is able to account stochasticity differently based on the detected regimes i.e. narrow confidence intervals for stationary periods and wide intervals for fluctuating periods. Moreover, judging from the “red-strips” (particles’ prediction intervals) with opacity proportion to particle weights, the proposed model is also capable of making skewed, heavy-tailed or even multi-modal predictions to the price. **Figure 5.5.10** also shows that numerically the proposed switching model exhibits much better model fit to the mid-price data compared to the single-regime model. Moreover, throughout all 200 PG iterations, the marginal joint PEDs only oscillate due to the inherent randomness of the sampling-based methods and shows no sign of overfitting e.g. decreasing $\hat{p}(Y)$.

In summary, the proposed model has demonstrated plausible posterior learning of regimes on real-world FOREX data. With very limited prior knowledge, the model reasonably identifies the price diffusion regime and learns the corresponding diffusion param-

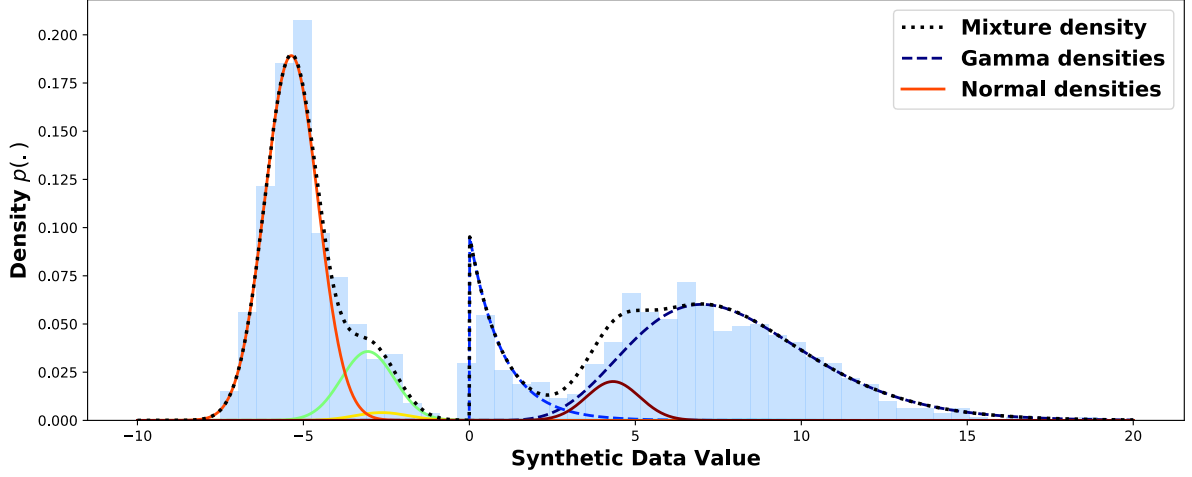


Figure 5.6.1: Histogram of example 1-D data randomly generated from the (weighted) mixture of 2 Gamma distributions and 4 normal distributions. One normal kernel (yellow) is significantly suppressed (due to its randomly assigned weight). Dotted line shows the (normalised) density of the mixture.

eters, which can potentially provide useful insights to retrospective analyses of the market and generation of investment decisions. Additionally, the proposed model is able to fully utilise its sequential construction and provide online price inference in the prediction mode with RBPF. The predictions obtained from the validation dataset show superior model fit and accountability for uncertainties compared to a standard single-regime SSM. However, in this experiment it has also been found that the posterior learning of the duration parameters failed to give meaningful results (not shown here). This is mainly due to the general short-lived nature of high-frequency market as well as the lack of observations as intervals.

5.6 Generalisation of diffusion models

In this section, I propose a hierarchical model that can potentially generalise the Langevin SSM in the proposed model to a combination of different continuous-time diffusion models such as Wiener process, OU process and Lévy process

However, I do not intend to go into great detail about the construction of these models. Instead, I illustrate my idea on a simple example of 1-D *exchangeable* dataset. **Figure 5.6.1** shows a histogram of 2000 1-D data generated from a mixture of Gamma distributions and normal distributions. It is clear that the general shape of the mixture density cannot be captured by (a mixture of) distributions of a single type (e.g. normal). This is analogous to the dynamical diffusion scenario where some diffusion behaviours simply cannot be accommodated by having different parameters with the same SSM setup.

A simple solution is provided to generalise the DPM by allowing an additional step of

model choice. Denote \mathcal{M} as the model choice random variable, so that the base measure \mathbb{P}_0 is re-defined as:

$$(\mathcal{M}, \theta) \stackrel{i.i.d.}{\sim} \mathbb{P}_0 = p(\mathcal{M}) p(\theta|\mathcal{M}) \quad (5.60)$$

where the parameters of a specific distribution are sampled conditioned on the chosen model, for example:

$$p(\mathcal{M}) = \begin{cases} 0.5 & \text{if } \mathcal{M} = \text{Normal} \\ 0.5 & \text{if } \mathcal{M} = \text{Gamma} \\ 0 & \text{otherwise} \end{cases} \quad (5.61)$$

$$p(\theta|\mathcal{M}) = \begin{cases} \mathcal{NIG}(\mu, \sigma^2) & \text{if } \mathcal{M} = \text{Normal} \\ \text{Gamma}(\alpha) \text{Gamma}(\beta) & \text{if } \mathcal{M} = \text{Gamma} \end{cases} \quad (5.62)$$

where θ can be different sets of parameters depending on \mathcal{M} . This gives the data kernel for a single observation y as:

$$p(y) = \sum_{k=1}^{\infty} \pi_k \mathcal{K}(y | \theta_k, \mathcal{M}_k) \quad (5.63)$$

where:

$$\mathcal{K}(y | \theta_k, \mathcal{M}_k) = \mathcal{N}(y | \theta_k) \delta_{\text{Normal}}(\mathcal{M}_k) + \text{Gamma}(y | \theta_k) \delta_{\text{Gamma}}(\mathcal{M}_k) \quad (5.64)$$

where $\delta_{\mathcal{M}'}(\mathcal{M}) = 1$ if $\mathcal{M} = \mathcal{M}'$ and 0 otherwise. As there is no inherent conjugacy between Gamma likelihood and its parameter prior, the *blocked* approximation of the “stick-breaking” from Section 5.4.2 is adopted again. The inference is performed with a blocked Gibbs sampler [119].

Figure 5.6.2 shows the inferred posterior clusters and their mixture density in comparison to the truth. For clarity of the figure, the mixture density is averaged across 50 converged iterations; whilst densities of individual clusters are shown only for the final iteration. It can be seen that there is an accurate fit of the posterior mixture density to both true mixture density and data histogram. It is clear that with model selection, DPM is now able to assign different distributions with different parameters to accommodate multi-modal and multi-model data.

In conclusion, this example of the extended DPM with model selection provides both theoretical support and empirical evidence to the generalisation of diffusion models. It is thus possible to achieve “infinity” in both parameters and models. However, it is worth noting that the dimension of the prior sample space would increase drastically if multiple parametric models are introduced to \mathbb{P}_0 . This may present challenges to inference.

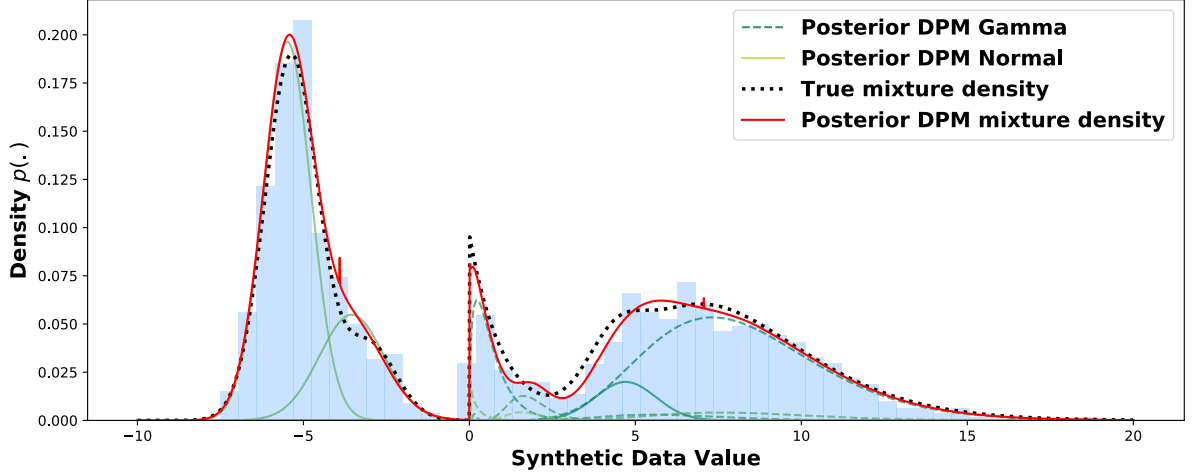


Figure 5.6.2: Inference results obtained from the DPM with model choice. The posterior mixture density is averaged across the converged 50 iterations; while the individual densities are shown only for the final iteration.

5.7 Conclusions

To summarise, in this chapter I have proposed a novel model that performs tracking, prediction, regime identification and parameter learning within a unified framework. Inspired originally by the diffusion of price in the LOB markets, the model employs an infinite-mixture DPM to account for unknown diffusion dynamics and a duration-explicit HsMM to account for switching features of the dynamic regimes. Re-parameterising the semi-Markovian duration with regime runtime, the complete model manages to maintain the Markovian property, which allows future predictions to be made online using offline-learned parameters.

In order to achieve full Bayesian inference of the model, I propose the combination of the RBPF algorithm and the blocked MwG sampler under the general iterative framework of the PG algorithm to achieve two-step iterative inference on the sequential states and the static regimes. The deterministic filtering scheme and the optimal resampling scheme are adopted to further improve the posterior inference performance.

Results obtained on synthetic data have shown good accuracy compared to the ground truth and fast convergence of the iterative algorithm. Experimenting on the two real datasets, the model has demonstrated reasonable identification of dynamic regimes and interpretable learning of regime parameters in real-world complex systems. Furthermore, the proposed model also showed better predictive performance compared to the standard single-regime model.

Lastly, a possible generalisation of the DPM via another hierarchy of model selection is proposed and briefly studied. Such a generalisation may provide even more flexible modelling of time-series diffusion data by accommodating different forms of diffusion models (i.e. SDEs). However, it also creates a higher-dimensional prior sample space (of

the base distribution \mathbb{P}_0) and thus requires a more efficient inference scheme. This could be an interesting topic for future work.

Appendix

5.A Stratified sampling algorithm

Stratified sampling algorithm, as a sub-routine for the optimal resampling scheme, can be implemented as follows. Denote the number of particles admitted to Φ_n as L ($= |\Phi_n|$) and w_i , $i = 1, \dots, M - L$, as the weights for the $M - L$ particles in set Ω . A total of $N_p - L$ particles need to be resampled in this stage. The stratified sampling algorithm [21] proceeds as follows:

Algorithm 12 Stratified resampling

```
1: Set  $K = \sum_{i=1}^{M-L} w_i / (N_p - L)$ 
2: Draw  $U \sim \text{Uniform}(0, K)$ 
3: for  $i = 1, \dots, M - L$  do
4:    $U = U - w_i$ 
5:   if  $U < 0$  then
6:     Resample the particle with index  $i$ 
7:      $U = U + K$ 
8:   end if
9: end for
```

Chapter 6

Marginal filters and variational parameter learning for state-space models

Previous chapters have covered several sequential model for prediction and inference of different aspects of high-frequency LOB markets. Whilst these models have demonstrated their strengths and unique features, each of them generally requires a good set of hyperparameters to achieve promising performance. In fact, the tuning of hyperparameters has always been the barrier between theoretical models and real-world applications. Due to the ever-changing nature of the high-frequency markets, the task of hyperparameters tuning becomes even more important and challenging in the case of financial modelling. Model parameter learning was first considered in this thesis in **Chapter 5**, and was achieved by the novel adaptation of the classic particle Gibbs (PG) algorithm. In this chapter, I utilise the hierarchical structure of Bayesian modelling and focus on developing efficient inference algorithms that can be applied generally to different state-space models for parameter learning purpose.

Time-series data modelling often establishes temporal correlations among time-indexed random variables, such as state vectors, to achieve accurate representations of real-world dynamics. However, the correlations among these random variables also present certain challenges to the learning of parameters especially for those that control diffusion dynamics. In the SSM, the dynamics-controlling parameters have correlations with state vectors at all timestamps. The change of dynamics-controlling parameters during the course of the sequential propagation will create a rippling effect to the states at all subsequent timestamps as well as inconsistency in diffusion dynamics to the states preceding the change. Furthermore, the large number of time-indexed random variables imposes a higher requirement on both accuracy and efficiency of the inference algorithm.

With the aim to overcome the challenges of parameter learning in sequential models, this chapter focuses on the development of novel algorithms for efficient inference of model parameters under both online and offline settings. The contributions of this chapter are as follows:

- A marginal Kalman filter is proposed to allow unknown model parameters to be propagated symbolically along with the Kalman mean and covariance. By employing the Rauch–Tung–Striebel (RTS) smoother [132] in the same time, the algorithm efficiently obtains both the posteriors of the parameters and the marginal posterior of the state vector in closed-form.
- Built upon the concept of the marginal Kalman filter, a marginal Rao-Blackwellised particle filter (RBPF) is developed to further accommodate non-linear and/or non-Gaussian SSMs. This novel variant of the RBPF enables particle weight computation based on online learned parameters’ posterior and summarises the marginal posterior of the parameters with a weighted mixture distribution.
- Finally, the chapter presents a novel integration of the PF and the variational Bayes (inference) algorithm, termed the particle filter variational inference (PF-VI) algorithm. The algorithm allows iterative Bayesian inference on both state vectors and model parameters for a wide range of SSMs. In comparison to the famous particle Markov chain Monte Carlo (PMCMC) algorithm [64], the PF-VI algorithm allows better utilisation of particle information and hence provides more stable posterior convergence.

Section 6.1 will introduce the marginal Kalman filter and its application on a synthetic linear-Gaussian diffusion example. In Section 6.2, the vision is extended to the development of algorithms associated with PFs to accommodate a wider range of SSMs. Both the marginal RBPF and the PF-VI algorithm are derived and introduced in this section. Last but not least, the experimental results obtained from the marginal RBPF and the PF-VI are displayed in Section 6.3 including a comparison with the PMCMC algorithm.

6.1 Marginal Kalman filter

Kalman filter (KF) [8], as the optimal inference algorithm for linear Gaussian SSMs, has been used extensively in many applications. The algorithm assumes Gaussian transition and observation models with known noise scales for both. In this section, we assume unknown but static noise scales and introduce a marginal Kalman filter (MKF) which allows unknown symbolic noise scales to be propagated with the time-indexed states and achieves accurate Bayesian parameters learning. The concept employed in the MKF differs from the Rao-Blackwellisation [81, 82] in the RBPF algorithm [83] the marginalisation

does not take place until the sequential propagation finishes and is taken with respect to the parameters' posterior. The MKF shares similarities in state posterior with the heavy-tailed student-t filters [133, 134] where transition and/or observation are formulated as student-t distributions.

6.1.1 Model setup

In order to setup a model for inference, a general continuous-time linear-Gaussian SSM is constructed as follows:

$$d\mathbf{x}_t = \mathbf{A} \mathbf{x}_t dt + \mathbf{h} dW_t \quad (6.1)$$

where \mathbf{h} is further written as $\mathbf{h} = \sigma_v \mathbf{b}$ with \mathbf{b} being a vector of zero(s) and one(s); dW_t is a unit-variance Wiener process. Further define the observation model for an observation y_t at time t as:

$$y_t = \mathbf{G} \mathbf{x}_t + \epsilon_t \quad (6.2)$$

where $\epsilon_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_w^2)$. The transition density and the likelihood can be written as:

$$p(\mathbf{x}_{t_n} | \mathbf{x}_{t_{n-1}}, \sigma_v^2) = \mathcal{N}\{\mathbf{x}_{t_n} | e^{\mathbf{A}\Delta t_n} \mathbf{x}_{t_{n-1}}, \Sigma(\Delta t_n, \sigma_v^2)\} \quad (6.3)$$

$$p(y_{t_n} | \mathbf{x}_{t_n}, \sigma_w^2) = \mathcal{N}(y_{t_n} | \mathbf{G} \mathbf{x}_{t_n}, \sigma_w^2) \quad (6.4)$$

where $\Delta t_n := t_n - t_{n-1}$ and $\Sigma(\Delta t_n, \sigma_v^2)$ can be readily computed as a function of time interval and innovation noise. From the above definition, one can see that this general model has two main parameters: the innovation/process noise variance σ_v^2 ; and the observation noise variance σ_w^2 . In certain applications, it is also possible to have matrix \mathbf{A} or element(s) of it as dynamics-controlling parameters (e.g. the regime-switching model proposed in **Chapter 5**). However, this will not be considered here for now.

Based on the results of previous chapters, the transition covariance matrix $\Sigma(\Delta t_n, \sigma_v^2)$ can be expanded and written as:

$$\begin{aligned} \Sigma(\Delta t_n, \sigma_v^2) &= e^{\mathbf{A}\Delta t_n} \left[\int_0^{\Delta t_n} e^{-\mathbf{A}\tau} \mathbf{h} \mathbf{h}^T (e^{-\mathbf{A}\tau})^T d\tau \right] (e^{\mathbf{A}\Delta t_n})^T \\ &= \sigma_v^2 e^{\mathbf{A}\Delta t_n} \left[\int_0^{\Delta t_n} e^{-\mathbf{A}\tau} \mathbf{b} \mathbf{b}^T (e^{-\mathbf{A}\tau})^T d\tau \right] (e^{\mathbf{A}\Delta t_n})^T \\ &= \sigma_v^2 \hat{\Sigma}_n \end{aligned} \quad (6.5)$$

The transition covariance can now be factorised into σ_v^2 and a term $\hat{\Sigma}_n$ that is a function of Δt_n . Furthermore, re-define the observation noise as a scaled version of the process noise:

$$\sigma_w^2 = \kappa_w \times \sigma_v^2 \quad (6.6)$$

where κ_w is a fixed constant. This proportional relation between the process noise and

the observation noise is usually much easier to tune or estimate based on specific applications. In order to perform Bayesian parameter learning for σ_v^2 , assign an inverse-Gamma distribution [135] as the prior of σ_v^2 :

$$p(\sigma_v^2) = \mathcal{IG}(\sigma_v^2 | \alpha, \beta) \quad (6.7)$$

The joint probability of the above model can thus be written, given a set of N observations $Y = \{y_{t_n}\}_{n=1}^N$ at (irregular) timestamps $\{t_n\}_{n=1}^N$ and an initial timestmap t_0 , as:

$$\begin{aligned} p(Y, \mathbf{X}, \sigma_v^2) &= p(\mathbf{x}_{t_0}) \prod_{n=1}^N \{p(\mathbf{x}_{t_n} | \mathbf{x}_{t_{n-1}}, \sigma_v^2) p(y_{t_n} | \mathbf{x}_{t_n}, \sigma_v^2)\} p(\sigma_v^2) \\ &= p(\mathbf{x}_{t_0}) \prod_{n=1}^N \left\{ \mathcal{N}(\mathbf{x}_{t_n} | e^{\mathbf{A}\Delta t_n} \mathbf{x}_{t_{n-1}}, \sigma_v^2 \hat{\Sigma}_n) \mathcal{N}(y_{t_n} | \mathbf{G} \mathbf{x}_{t_n}, \kappa_w \sigma_v^2) \right\} \\ &\quad \times \mathcal{IG}(\sigma_v^2 | \alpha, \beta) \end{aligned} \quad (6.8)$$

where $\mathbf{X} = \{\mathbf{x}_{t_n}\}_{n=0}^N$.

6.1.2 Marginal filtering

As for a standard KF [88], in order to begin marginal Kalman filtering it is necessary to define the initial condition of the state vector \mathbf{x}_{t_0} . This initialisation usually varies based on the exact application of the model and some applications simply assume a perfectly observed initial state. Without loss of generality, let us define a Gaussian initial state as:

$$p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0 | \boldsymbol{\mu}_{0|0}, \sigma_v^2 \mathbf{L}_{0|0}) \quad (6.9)$$

where again the covariance is constructed proportional to σ_v^2 . With a slight abuse of notation, let us allow state vectors and data to be indexed by their corresponding timestamps index n . Treating σ_v^2 as a static but unknown random variable, it can be kept in symbolic representation and the state vectors shall propagate as follows:

$$p(\mathbf{x}_n | y_{1:n-1}, \sigma_v^2) = \int p(\mathbf{x}_n | \mathbf{x}_{n-1}, \sigma_v^2) p(\mathbf{x}_{n-1} | y_{1:n-1}, \sigma_v^2) d\mathbf{x}_{n-1} \quad (6.10)$$

Assume that the posterior filtering density $p(\mathbf{x}_{n-1} | y_{1:n-1}, \sigma_v^2)$ can be obtained in a Gaussian form i.e.:

$$\begin{aligned} p(\mathbf{x}_{n-1} | y_{1:n-1}, \sigma_v^2) &= \mathcal{N}(\mathbf{x}_{n-1} | \boldsymbol{\mu}_{n-1|0:n-1}, \Sigma_{n-1|0:n-1}) \\ &= \mathcal{N}(\mathbf{x}_{n-1} | \boldsymbol{\mu}_{n-1|0:n-1}, \sigma_v^2 \mathbf{L}_{n-1|0:n-1}) \end{aligned} \quad (6.11)$$

The integration of (6.10) is readily computed as:

$$p(\mathbf{x}_n | y_{1:n-1}, \sigma_v^2) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:n-1}, \boldsymbol{\Sigma}_{n|0:n-1}) \quad (6.12)$$

with predictive parameters:

$$\boldsymbol{\mu}_{n|0:n-1} = e^{\mathbf{A}\Delta t_n} \boldsymbol{\mu}_{n-1|0:n-1} \quad (6.13)$$

$$\begin{aligned} \boldsymbol{\Sigma}_{n|0:n-1} &= e^{\mathbf{A}\Delta t_n} \left[\int_0^{\Delta t_n} e^{-\mathbf{A}\tau} \sigma_v^2 \mathbf{b}\mathbf{b}^T (e^{-\mathbf{A}\tau})^T d\tau + \sigma_v^2 \mathbf{L}_{n-1|0:n-1} \right] (e^{\mathbf{A}\Delta t_n})^T \\ &= \sigma_v^2 e^{\mathbf{A}\Delta t_n} \left[\int_0^{\Delta t_n} e^{-\mathbf{A}\tau} \mathbf{b}\mathbf{b}^T (e^{-\mathbf{A}\tau})^T d\tau + \mathbf{L}_{n-1|0:n-1} \right] (e^{\mathbf{A}\Delta t_n})^T \\ &= \sigma_v^2 \mathbf{L}_{n|0:n-1} \end{aligned} \quad (6.14)$$

The normal Kalman procedure allows us to update the predictive parameters based on the observation y_n , and obtain the posterior filtering density for state vector \mathbf{x}_n :

$$\begin{aligned} p(\mathbf{x}_n | y_{1:n}, \sigma_v^2) &= \frac{p(\mathbf{x}_n | y_{1:n-1}, \sigma_v^2) p(y_n | \mathbf{x}_n, \sigma_v^2)}{p(y_n | y_{1:n-1}, \sigma_v^2)} \\ &= \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:n}, \boldsymbol{\Sigma}_{n|0:n}) \end{aligned} \quad (6.15)$$

with

$$\begin{aligned} \mathbf{Q}_n &= \boldsymbol{\Sigma}_{n|0:n-1} \mathbf{G}^T (\mathbf{G} \boldsymbol{\Sigma}_{n|0:n-1} \mathbf{G}^T + \sigma_v^2 \kappa_w)^{-1} \\ &= \sigma_v^2 \mathbf{L}_{n|0:n-1} \mathbf{G}^T \left\{ \sigma_v^2 (\mathbf{G} \mathbf{L}_{n|0:n-1} \mathbf{G}^T + \kappa_w) \right\}^{-1} \\ &= \mathbf{L}_{t|0:t-1} \mathbf{G}^T (\mathbf{G} \mathbf{L}_{t|0:t-1} \mathbf{G}^T + \kappa_e)^{-1} \end{aligned} \quad (6.16)$$

$$\boldsymbol{\mu}_{n|0:n} = \boldsymbol{\mu}_{n|0:n-1} + \mathbf{Q}_n (y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1}) \quad (6.17)$$

$$\begin{aligned} \boldsymbol{\Sigma}_{n|0:n} &= (\mathbf{I} - \mathbf{Q}_n \mathbf{G}) \boldsymbol{\Sigma}_{n|0:n-1} \\ &= \sigma_v^2 (\mathbf{I} - \mathbf{Q}_n \mathbf{G}) \mathbf{L}_{n|0:n-1} \\ &= \sigma_v^2 \mathbf{L}_{n|0:n} \end{aligned} \quad (6.18)$$

Note that both Kalman gain \mathbf{Q}_n and posterior mean $\boldsymbol{\mu}_{n|0:n}$ are not functions of σ_v^2 while $\boldsymbol{\Sigma}_{n|0:n}$ can again be factorised with respect to σ_v^2 . This allows a “marginal” Kalman recursion to proceed with σ_v^2 unknown and numerical values of $\boldsymbol{\Sigma}_{n|0:n-1}$, $\boldsymbol{\Sigma}_{n|0:n}$ un-computed. The process of marginal Kalman filtering is essentially propagating the coefficients of the symbolic σ_v^2 at both prediction and filtering stages of a standard KF while keeping the propagation of state mean unchanged.

6.1.3 Posteriors, backward smoothing and marginal state inference

Conditioned on a symbolic σ_v^2 , the filtering distributions of the state vectors are readily available but unknown. The posterior inference of σ_v^2 should hence be performed by first

writing the Kalman *prediction error decomposition* (PED) at time t_n :

$$\begin{aligned}
p(y_n | y_{1:n-1}, \sigma_v^2) &= \int p(y_n | \mathbf{x}_n, \sigma_v^2) p(\mathbf{x}_n | y_{1:n-1}, \sigma_v^2) d\mathbf{x}_n \\
&= \mathcal{N}\left(y_n | \mathbf{G} \boldsymbol{\mu}_{n|0:n-1}, \mathbf{G} \boldsymbol{\Sigma}_{n|0:n-1} \mathbf{G}^T + \kappa_w \sigma_v^2\right) \\
&= \frac{1}{\sqrt{2\pi} \sigma_v^2 (\mathbf{G} \mathbf{L}_{n|0:n-1} \mathbf{G}^T + \kappa_w)} \exp\left\{-\frac{(y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1})^2}{2\sigma_v^2 (\mathbf{G} \mathbf{L}_{n|0:n-1} \mathbf{G}^T + \kappa_w)}\right\}
\end{aligned} \tag{6.19}$$

And thus, the conditional evidence of the model conditioned on σ_v^2 is:

$$p(Y | \sigma_v^2) = p(y_1 | \sigma_v^2) \prod_{n=2}^N p(y_n | y_{1:n-1}, \sigma_v^2) \tag{6.20}$$

Clearly, the closed-form posterior of σ_v^2 can be obtained easily with the above conditional evidence. For simplicity, this is derived in *log*-form:

$$\begin{aligned}
&\log\{p(\sigma_v^2 | Y)\} \\
&= \log\{p(Y | \sigma_v^2)\} + \log\{p(\sigma_v^2)\} + \text{const.} \\
&= -\frac{1}{2} \sum_{n=1}^N \left\{ \log(\sigma_v^2) + \frac{(y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1})^2}{\sigma_v^2 (\mathbf{G} \mathbf{L}_{n|0:n-1} \mathbf{G}^T + \kappa_w)} \right\} + (-\alpha - 1) \log(\sigma_v^2) - \frac{\beta}{\sigma_v^2} + \text{const.} \\
&= (-\alpha - \frac{N}{2} - 1) \log(\sigma_v^2) - \frac{1}{\sigma_v^2} \left(\beta + \sum_{n=1}^N \frac{(y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1})^2}{2(\mathbf{G} \mathbf{L}_{n|0:n-1} \mathbf{G}^T + \kappa_w)} \right) + \text{const.}
\end{aligned} \tag{6.21}$$

where the constant terms contain the values that are not functions of σ_v^2 . This gives the posterior hyperparameters of $p(\sigma_v^2 | Y) = \mathcal{IG}(\hat{\alpha}, \hat{\beta})$ as:

$$\hat{\alpha} = \alpha + \frac{N}{2} \quad , \quad \hat{\beta} = \beta + \sum_{n=1}^N \frac{(y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1})^2}{2(\mathbf{G} \mathbf{L}_{n|0:n-1} \mathbf{G}^T + \kappa_w)} \tag{6.22}$$

With the posterior of σ_v^2 obtained, it is now possible to perform posterior inference on the sequential state vectors. Recall the posterior filtering density for state vector \mathbf{x}_n during the marginal Kalman filtering:

$$p(\mathbf{x}_n | y_{1:n}, \sigma_v^2) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:n}, \sigma_v^2 \mathbf{L}_{n|0:n}) \tag{6.23}$$

It is clear that the marginal of this filtering posterior cannot be obtained by simply taking the expectation of σ_v^2 with respect to its posterior as this filtering distribution only has the information from observations up to time t_n . Whilst the posterior of σ_v^2 contains the information of the entire data sequence $\{y_n\}_{n=1}^N$.

In order to mitigate this and obtain the exact posterior of the state vector, the Rauch–Tung–Striebel (RTS) smoother is adopted as an efficient backwards smoothing

Algorithm 13 RTS smoother

Input: Filtering parameters $\{\boldsymbol{\mu}_{n|0:n}, \boldsymbol{\Sigma}_{n|0:n}\}_{n=0}^N$; predictive parameters $\{\boldsymbol{\mu}_{n|0:n-1}, \boldsymbol{\Sigma}_{n|0:n-1}\}_{n=1}^N$; and transition matrices $\{e^{\mathbf{A}\Delta t_n}\}_{n=1}^N$

Output: Posterior smoothing parameters $\{\boldsymbol{\mu}_{n|0:N}, \boldsymbol{\Sigma}_{n|0:N}\}_{n=0}^N$

```

1: for  $n = N - 1, N - 2, \dots, 0$  do
2:    $\mathbf{C}_n = \boldsymbol{\Sigma}_{n|0:n} (e^{\mathbf{A}\Delta t_{n+1}})^{-1} \boldsymbol{\Sigma}_{n+1|0:n}^{-1}$ 
3:    $\boldsymbol{\mu}_{n|0:N} = \boldsymbol{\mu}_{n|0:n} + \mathbf{C}_n (\boldsymbol{\mu}_{n|0:N} - \boldsymbol{\mu}_{n+1|0:n})$ 
4:    $\boldsymbol{\Sigma}_{n|0:N} = \boldsymbol{\Sigma}_{n|0:n} + \mathbf{C}_n (\boldsymbol{\Sigma}_{n+1|0:N} - \boldsymbol{\Sigma}_{n+1|0:n}) \mathbf{C}_n^T$ 
5: end for
6: Return:  $\{\boldsymbol{\mu}_{n|0:N}, \boldsymbol{\Sigma}_{n|0:N}\}_{n=0}^N$ 

```

algorithm in a linear-Gaussian model [132]. The target density of the RTS smoother is the smoothing posterior i.e.:

$$p(\mathbf{x}_n | Y, \sigma_v^2) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:N}, \boldsymbol{\Sigma}_{n|0:N}) \quad (6.24)$$

which is readily available for $n = N$. Hence, the algorithm starts at the last timestamp t_N and propagates the parameters backwards towards t_0 . **Algorithm 13** shows the scheme of a RTS smoother which takes parameters and values generated in MKF as its input. Clearly, both \mathbf{C}_n and $\boldsymbol{\mu}_{n|0:N}$ are not functions of σ_v^2 , and $\boldsymbol{\Sigma}_{n|0:N}$ can be factorised as:

$$\boldsymbol{\Sigma}_{n|0:N} = \sigma_v^2 \mathbf{L}_{n|0:N} \quad (6.25)$$

which allows us to still keep σ_v^2 as unknown throughout the course of the RTS smoother.

Finally, the marginal posterior of the state vector can be computed by integrating the unknown process noise, which has been kept symbolic throughout the entire inference process, with respect to its posterior distribution. This yields a multivariate *student-t* distribution as the standard result. The full derivation is shown below:

$$\begin{aligned}
p(\mathbf{x}_n | Y) &= \int p(\mathbf{x}_n | Y, \sigma_v^2) p(\sigma_v^2 | Y) d\sigma_v^2 \\
&= \int \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:N}, \boldsymbol{\Sigma}_{n|0:N}) \mathcal{IG}(\sigma_v^2 | \hat{\alpha}, \hat{\beta}) d\sigma_v^2 \\
&= \int \frac{1}{(2\pi)^{\frac{D}{2}} (\sigma_v^2)^{\frac{D}{2}} |\mathbf{L}_{n|0:N}|^{\frac{1}{2}}} \times \frac{\hat{\beta}^{\hat{\alpha}}}{\Gamma(\hat{\alpha})} (\sigma_v^2)^{-\hat{\alpha}-1} \times \\
&\quad \exp\left\{ -\frac{1}{2\sigma_v^2} (\mathbf{x}_n - \boldsymbol{\mu}_{n|0:N})^T \mathbf{L}_{n|0:N}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_{n|0:N}) - \frac{\hat{\beta}}{\sigma_v^2} \right\} d\sigma_v^2
\end{aligned} \quad (6.26)$$

where $\Gamma(\cdot)$ is the Gamma function. For simplicity of the expression, I further denote

$\mathbf{M} = (\mathbf{x}_n - \boldsymbol{\mu}_{n|0:N})^T \mathbf{L}_{n|0:N}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_{n|0:N})$ and continue the derivation as:

$$\begin{aligned}
p(\mathbf{x}_n | Y) &= \frac{\hat{\beta}^{\hat{\alpha}}}{(2\pi)^{\frac{D}{2}} |\mathbf{L}_{n|0:N}|^{\frac{1}{2}} \Gamma(\hat{\alpha})} \times \frac{\Gamma(\hat{\alpha} + \frac{D}{2})}{(\hat{\beta} + \frac{\mathbf{M}}{2})^{\hat{\alpha} + \frac{D}{2}}} \times \\
&\quad \underbrace{\int \frac{(\hat{\beta} + \frac{\mathbf{M}}{2})^{\hat{\alpha} + \frac{D}{2}}}{\Gamma(\hat{\alpha} + \frac{D}{2})} (\sigma_v^2)^{-(\hat{\alpha} + \frac{D}{2}) - 1} \exp\left\{-\frac{\hat{\beta} + \frac{\mathbf{M}}{2}}{\sigma_v^2}\right\} d\sigma_v^2}_{\text{integrates to 1}} \\
&= \frac{\Gamma(\hat{\alpha} + \frac{D}{2})}{\Gamma(\hat{\alpha}) (2\hat{\beta})^{D/2} \pi^{D/2} |\mathbf{L}_{n|0:N}|^{1/2}} \times \left[1 + \frac{\mathbf{M}}{2\hat{\beta}}\right]^{-(\hat{\alpha} + \frac{D}{2})} \\
&= \frac{\Gamma(\hat{\alpha} + \frac{D}{2})}{\Gamma(\hat{\alpha}) (2\hat{\alpha})^{D/2} \pi^{D/2} |\frac{\hat{\beta}}{\hat{\alpha}} \mathbf{L}_{n|0:N}|^{1/2}} \times \left[1 + \frac{(\mathbf{x}_n - \boldsymbol{\mu}_{n|0:N})^T (\frac{\hat{\beta}}{\hat{\alpha}} \mathbf{L}_{n|0:N})^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_{n|0:N})}{2\hat{\alpha}}\right]^{-(\hat{\alpha} + \frac{D}{2})}
\end{aligned} \tag{6.27}$$

where the first line of the above equation is obtained by simultaneously multiplying and dividing a constant. The last line essentially gives a generalised multivariate student-t distribution with degrees of freedom $\nu = 2\hat{\alpha}$, location $\boldsymbol{\mu} = \boldsymbol{\mu}_{n|0:N}$ and covariance $\boldsymbol{\Sigma} = \frac{\hat{\beta}}{\hat{\alpha}} \mathbf{L}_{n|0:N}$ i.e.:

$$p(\mathbf{x}_n | Y) = t_{2\hat{\alpha}}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:N}, \frac{\hat{\beta}}{\hat{\alpha}} \mathbf{L}_{n|0:N}) \tag{6.28}$$

And naturally, the posterior of the observation noise σ_w^2 can be obtained from its connection to the process noise:

$$p(\sigma_w^2 | Y) = \mathcal{IG}(\sigma_w^2 | \hat{\alpha}, \kappa_w \hat{\beta}) \tag{6.29}$$

6.1.4 Simulation results

I apply the MKF algorithm on a set of synthetic data simulated from a linear-Gaussian Langevin model. More specifically, 100 data points are generated on an irregular time-grid with:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -0.5 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \sigma_v^2 = 10, \sigma_w^2 = 20, \mu_{0|0} = \mathbf{x}_0, L_{0|0} = \mathbf{0} \tag{6.30}$$

The proposed MKF employs a diffuse inverse-Gamma prior with $\alpha = \beta = 0.01$. The posterior inference results for state vectors are compared to three Kalman filters with true, underestimated and overestimated σ_v^2 values. **Figure 6.1.1** shows the inference results obtained using both KFs with fixed parametric values and MKF with a diffuse prior. In this experiment, the hyperparameter $\kappa_w = 2$ is assumed to be known. It is clear that the MKF is able to achieve a confidence interval that is closest to the KF results obtained with the true parameter values. On the other hand, both KFs with 50% over and underestimation of σ_v^2 show worse results in terms of the confidence interval compared to the MKF. **Figure 6.1.2** shows the posterior obtained from the MKF.

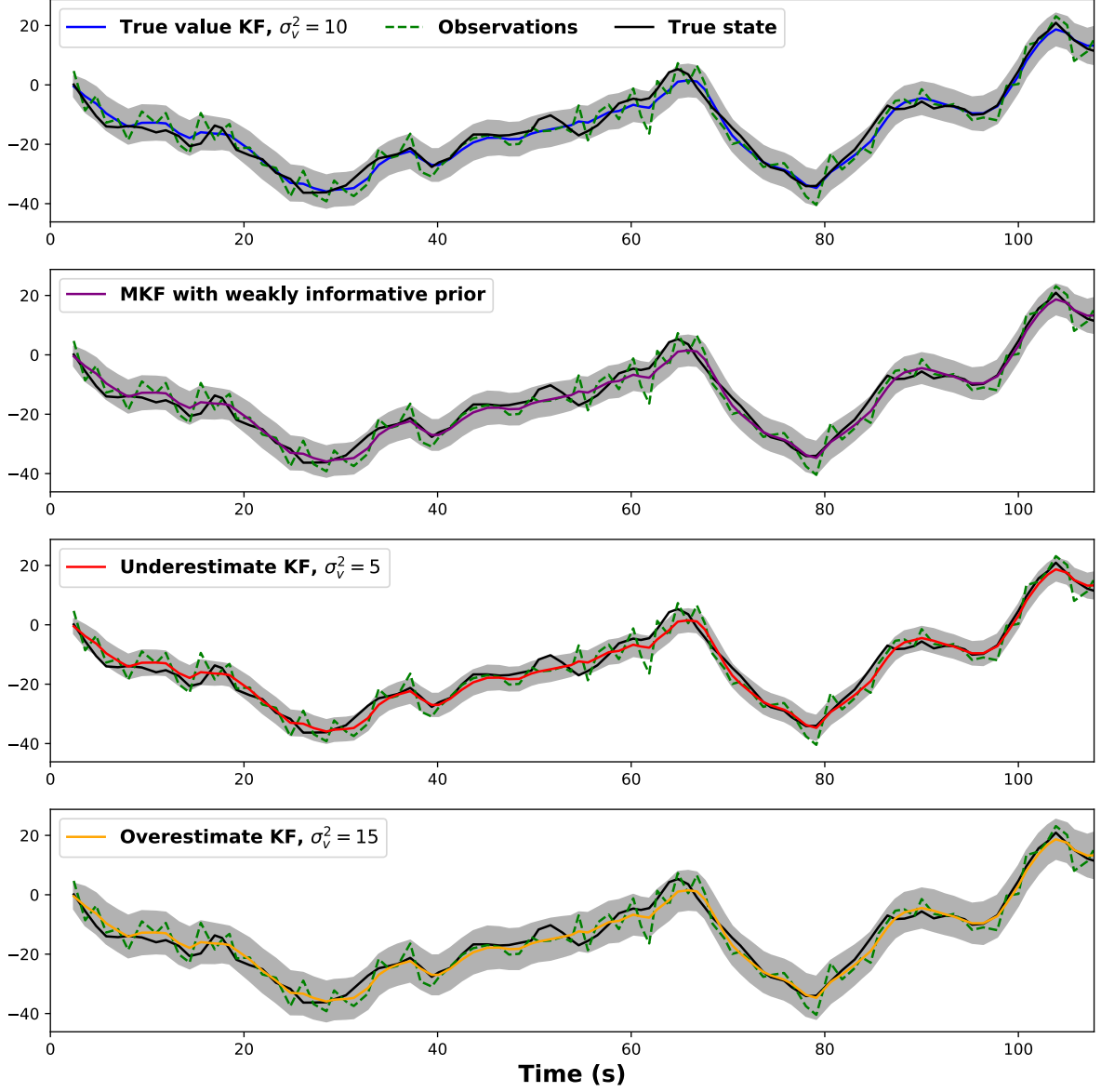


Figure 6.1.1: Posterior inference results for state vectors for different algorithms and parameter settings. The black curves are the observations. The colored lines are the posterior means of the state vectors while the shaded regions indicate the $\pm 2\sigma$ confidence intervals.

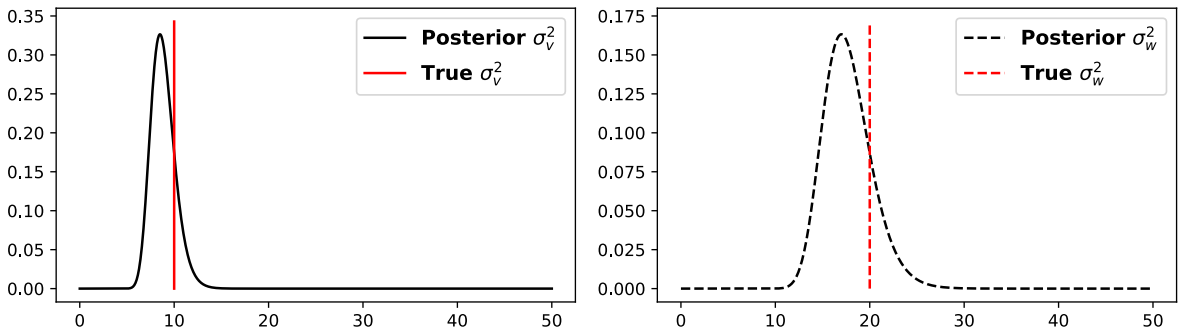
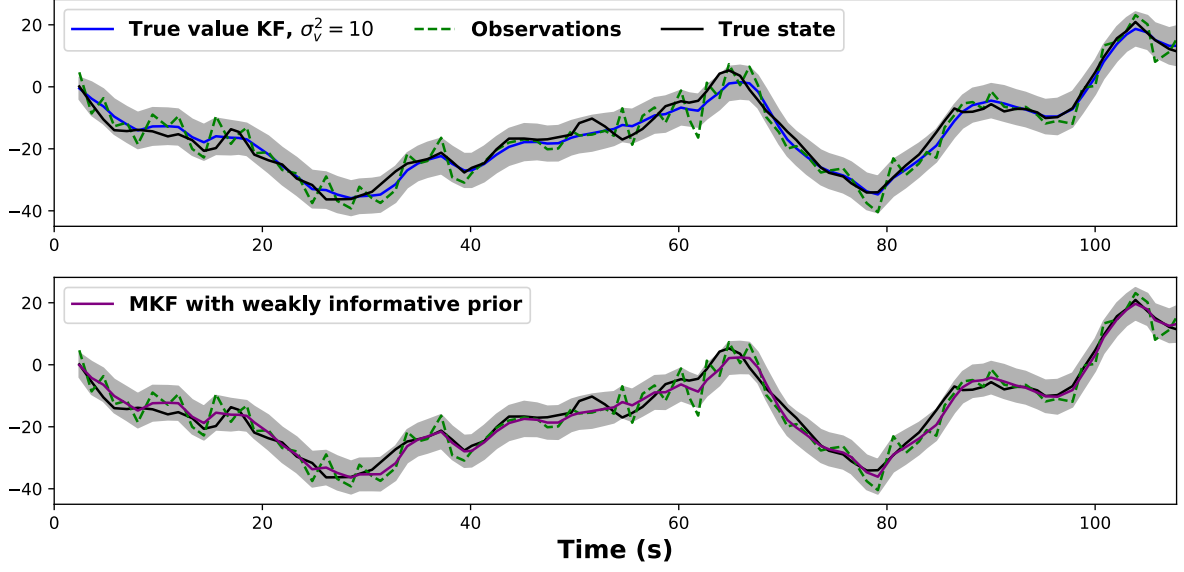
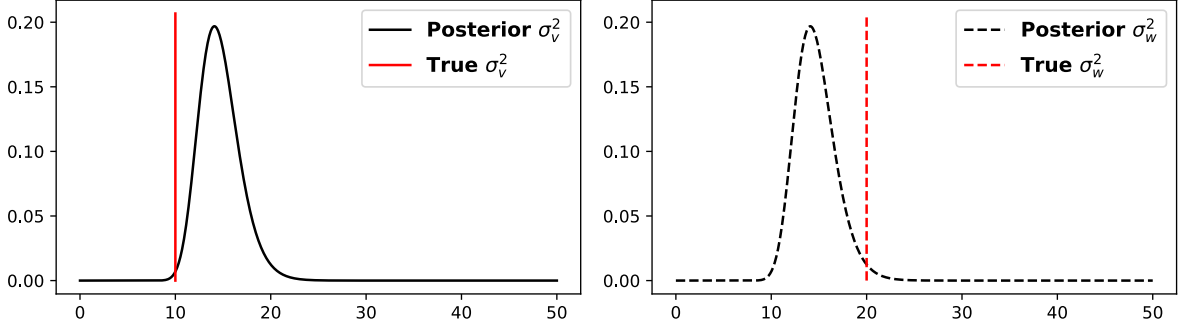


Figure 6.1.2: Posteriors for σ_v^2 and σ_w^2 obtained with MKF. Assume κ_w is known and accurate.



(a) State vector posteriors with means and $\pm 2\sigma$ confidence interval.



(b) Posteriors for σ_v^2 and σ_w^2 .

Figure 6.1.3: Two plots show the posteriors of state vectors and noise parameters respectively. The MKF was run with incorrect κ_w value of 1.

An additional experiment was carried out on MKF with incorrect values of κ_w , which evaluates the algorithm under a realistic scenario. **Figure 6.1.3** shows the posterior results obtained with an incorrect $\kappa_w = 1.0$. Apparently even with a wrong κ_w , the MKF algorithm is still able to achieve posterior results that are very close to those obtained from the KF with true parameters. As can be seen from panel (b), the learned posteriors has balanced out the effect of an incorrect κ_w by overestimating the process noise σ_v^2 .

Based on the experiments, it is fair to conclude that the proposed MKF algorithm is able to perform accurate posterior inference on both state vectors and noise parameters given a well-tuned κ_w . Even in the case where κ_w can not be estimated accurately, MKF is still able to perform robust posterior inference on state vectors by compensating the negative effect of an incorrect κ_w with a learned σ_v^2 from the data.

6.2 Parameter learning in the particle filter

Despite the good performance achieved by the MKF, its application is severely limited to only linear-Gaussian dynamical models just like the generic KF. And real-world systems always involve complex dynamics that cannot be simply described with linear-Gaussian models. While non-linear dynamics can be accommodated in some variants of the KF such as the extended Kalman filter [136], their performance relies heavily on the accuracy of approximated linearisation. Using Monte Carlo approximation, the particle filter (PF) was developed and has been widely adopted in the inference of highly non-linear and/or non-Gaussian dynamical models. Hence in this section, I study parameter learning algorithms for dynamical models that can be readily fitted in the general framework of the PF.

The authors of [137] reviewed a number of parameter estimation methods developed for SSMs incorporating the PF. This includes an offline non-Bayesian maximum likelihood (ML) method originally introduced in [138]. One typical difficulty of the ML approach is the discontinuous evaluation of likelihood function caused by the PF resampling. This issue was elegantly bypassed in [139] by introducing a “continuous” resampling step. Although the online ML learning of parameters has been shown to converge under regularity conditions for HMMs [140], the convergence for general SSM remains unproven.

Particle Markov chain Monte Carlo (PMCMC) [64], as an offline Bayesian parameter learning algorithm, has provided good results in **Chapter 5** via its particle Gibbs (PG) variant. Another algorithm that falls into the PMCMC framework is the particle marginal Metropolis-Hastings (PMMH) algorithm. The algorithm was initially proposed in [141] as a heuristic to sample from a posterior distribution $p(\theta | y_{1:N})$ with θ being model parameter(s). The authors of [64] later established a remarkable feature of the algorithm which admits the extended target posterior $p(\mathbf{x}_{1:N}, \theta | y_{1:N})$ as its invariant distribution. Online Bayesian methods, on the other hand, target the sequential posterior of $p(\mathbf{x}_{1:n}, \theta | y_{1:n})$, for $n = 1 \dots N$. A naive solution to this problem is by augmenting the state \mathbf{x}_n to include parameter(s) θ in the particle set so that both are inferred with the PF. However, a static θ does not possess any forgetting property and is bound to degenerate as the PF progresses. Important developments in this direction include two popular algorithms: the *Storvik’s* filter [142] where the algorithm assumes the online posterior $p(\theta | \mathbf{x}_{1:n}, y_{1:n})$ can be learned from a set of low-dimensional sufficient statistics of the states $\mathbf{x}_{1:n}$ and observations $y_{1:n}$ that can be recursively updated; and the *Liu and West’s* filter [143] which utilises the resample-propagate framework of the auxiliary particle filter (APF) [144] and aims to solve the degeneracy problem by sampling θ at each propagation from the online posterior approximated by a mixture of Gaussians. The authors of [106, 145] proposed a novel particle learning (PL) which proves to be a competitor to the MCMC algorithms. The PL combines the resample-propagate framework and the conditional sufficient statistics representation, and has demonstrated promising results in a recent financial application

[146]. An alternative workaround of the degeneracy problem involves introducing dynamics to the parameter θ and making it time-varying i.e. $\{\theta_n\}_{n=1}^N$. This concept has been applied in the volumetric model in **Chapter 3** and also in [147]. However in certain cases, this means the introduction of an unnecessary artificial dynamical model. The SMC² algorithm introduced simultaneously in [148] and [149] can be considered as an online particle equivalent of the PMCMC algorithm where a different set of N_θ particles is used to explore the distribution $p(\theta | y_{1:n})$ at each timestamp t_n . The θ -particles are re-weighted at time t_{n+1} according to the ratio of empirically estimated partial likelihoods $\hat{p}_{\theta^i}(y_{1:n+1})/\hat{p}_{\theta^i}(y_{1:n})$ for each of $\{\theta^i\}_{i=1}^{N_\theta}$. However, the SMC² algorithm often demands an extensive amount of computation and is thus generally impractical to be used under real online settings.

In the remainder of this section, two novel algorithms for Bayesian parameter learning in dynamical models are introduced. Furthermore, the proposed algorithms are tested on synthetic datasets with their relative performance evaluated against some existing methods.

6.2.1 Marginal Rao-Blackwellised particle filter

With the success achieved with the MKF, it is intuitive to generalise its application to a Rao-Blackwellised version of the particle filter where state vectors are marginalised and inferred with a conditional Kalman filter. This leads the marginal RBPF as a direct extension of MKF. The authors of [150] briefly introduced a type of marginal Monte Carlo filter which is similar to the proposed marginal RBPF. Meanwhile, the marginal RBPF also shares certain similarities with the *Storvik's* filter [142] and the PL [106] algorithm in that a set of recursively updated sufficient statistics is maintained for each particle during the PF propagation. However, instead of sampling parameters from an approximated online posterior or distribution, the marginal RBPF employs a conjugate prior structure and allows the unknown parameters to be marginalised mitigating the particle degeneracy problem.

Model setup

As the original RBPF algorithm can be applied in a range of SSMs where state vectors are either fully or partially “Rao-Blackwellised”, without loss of generality, the marginal RBPF algorithm is demonstrated on the jump-diffusion model proposed in [13] and reviewed in **Chapter 3**. Recall the continuous-time jump-diffusion SSM which follows the

Langevin dynamics:

$$d\mathbf{x}_t = \mathbf{A}\mathbf{x}_t dt + \sigma_v \mathbf{b} dW_t + \mathbf{c} dJ_t$$

$$d \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \theta \end{bmatrix} \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} dt + \sigma_v \begin{bmatrix} 0 \\ 1 \end{bmatrix} dW_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} dJ_t \quad (6.31)$$

where θ is assumed as a known hyperparameter; and the process dJ_t is typically a non-Gaussian jump process. Here, consider a Gauss-Poisson process with jump times $\{\tau_k\}_k$ (where k is the jump index) following a homogeneous Poisson process and a normally distributed jump size \mathcal{J}_k :

$$\tau_k - \tau_{k-1} \sim \text{Exponential}(\lambda_J) \quad (6.32)$$

$$\mathcal{J}_k \sim \mathcal{N}(\mu_J, \sigma_J^2) \quad (6.33)$$

In such a case, the state transition density of \mathbf{x}_n has been derived to be Gaussian conditioned on both \mathbf{x}_{n-1} and the jumps that occur within the time interval $(t_{n-1}, t_n]$:

$$p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}) = \mathcal{N}(\mathbf{x}_n | \tilde{\boldsymbol{\mu}}_n, \tilde{\boldsymbol{\Sigma}}_n) \quad (6.34)$$

where $\{\tau\}_{n-1:n}$ denotes the jumps in $(t_{n-1}, t_n]$; and the Gaussian mean and covariance are:

$$\tilde{\boldsymbol{\mu}}_n = e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1} + \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c} \mu_J \quad (6.35)$$

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_n &= \boldsymbol{\Sigma}(\Delta t, \sigma_v^2) + \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c} \mathbf{c}^T \sigma_J^2 (e^{\mathbf{A}(t_n - \tau_k)})^T \\ &= \sigma_v^2 \hat{\boldsymbol{\Sigma}}_n + \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c} \mathbf{c}^T \sigma_J^2 (e^{\mathbf{A}(t_n - \tau_k)})^T \end{aligned} \quad (6.36)$$

where $\hat{\boldsymbol{\Sigma}}_n$ is defined in Eq. (6.5) as the factorised coefficient of the transition covariance. The same intuition is adopted from the MKF algorithm by simply assuming a proportional relationship between jump variance and the process variance i.e. $\sigma_J^2 = \kappa_J \sigma_v^2$; and thus obtain:

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_n &= \sigma_v^2 \hat{\boldsymbol{\Sigma}}_n + \sigma_v^2 \kappa_J \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c} \mathbf{c}^T (e^{\mathbf{A}(t_n - \tau_k)})^T \\ &= \sigma_v^2 \hat{\boldsymbol{\Sigma}}_{\tau,n} \end{aligned} \quad (6.37)$$

In the case of symmetric jumps (i.e. $\mu_J = 0$), such a formulation of the jump variance allows an easy factorisation of the transition covariance and the likelihood variance. However, I take a step further and aim to infer the jump mean μ_J and the process variance σ_v^2 simultaneously by having a “seemingly” conjugate normal-inverse-Gamma prior jointly

on the two random variables:

$$\begin{aligned} p(\mu_J, \sigma_v^2) &= \mathcal{NIG}(\mu_J, \sigma_v^2 \mid m, \nu, \alpha, \beta) \\ &= \mathcal{N}(\mu_J \mid m, \frac{\sigma_v^2}{\nu}) \mathcal{IG}(\sigma_v^2 \mid \alpha, \beta) \end{aligned} \quad (6.38)$$

The observational model is the same as that in (6.2) with observation matrix $\mathbf{G} = [1 \ 0]$ for the Langevin model.

Conditional MKF

As the regular RBPF algorithm for this jump-diffusion application has been thoroughly introduced in **Chapter 3**, it will not be derived from scratch again here. Instead, let us focus on how the analogous symbolic representation (of the unknown parameters) used in the MKF can also be adopted in the marginal RBPF, or more precisely in the conditional MKF. Suppose that by conditioning on a (sampled) trajectory of jump times $\{\tau\}_{0:n}$ from t_0 to t_n , the posterior filtering distribution of the state vector \mathbf{x}_{n-1} is a Gaussian distribution:

$$p(\mathbf{x}_{n-1} \mid y_{1:n-1}, \{\tau\}_{0:n-1}) = \mathcal{N}(\mathbf{x}_{n-1} \mid \boldsymbol{\mu}_{n-1|0:n-1}, \boldsymbol{\Sigma}_{n-1|0:n-1}) \quad (6.39)$$

with posterior (filtering) parameters:

$$\boldsymbol{\mu}_{n-1|0:n-1} = \boldsymbol{\mu}_{n-1|0:n-1}^c + \mathbf{d}_{n-1|0:n-1} \mu_J \quad (6.40)$$

$$\boldsymbol{\Sigma}_{n-1|0:n-1} = \sigma_v^2 \mathbf{L}_{n-1|0:n-1} \quad (6.41)$$

where $\boldsymbol{\mu}_{n-1|0:n-1}^c$ is a constant term from the linear Gaussian diffusion; while $\mathbf{d}_{n-1|0:n-1}$ is the coefficient for jump-induced drifts. Combined with the state vector transition density in (6.34), one can obtain the marginal predictive distribution of \mathbf{x}_n as:

$$\begin{aligned} p(\mathbf{x}_n \mid y_{1:n}, \{\tau\}_{0:n}) &= \int p(\mathbf{x}_{n-1} \mid y_{1:n-1}, \{\tau\}_{0:n-1}) p(\mathbf{x}_n \mid \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}) d\mathbf{x}_{n-1} \\ &= \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_n|0:n-1, \boldsymbol{\Sigma}_n|0:n-1) \end{aligned} \quad (6.42)$$

with predictive parameters:

$$\begin{aligned}
\boldsymbol{\mu}_{n|0:n-1} &= e^{\mathbf{A}\Delta t_n} \boldsymbol{\mu}_{n-1|0:n-1} + \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c} \mu_J \\
&= e^{\mathbf{A}\Delta t_n} \boldsymbol{\mu}_{n-1|0:n-1}^c + \left(e^{\mathbf{A}\Delta t_n} \mathbf{d}_{n-1|0:n-1} + \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c} \right) \mu_J \quad (6.43)
\end{aligned}$$

$$\begin{aligned}
&= \boldsymbol{\mu}_{n|0:n-1}^c + \mathbf{d}_{n|0:n-1} \mu_J \\
\boldsymbol{\Sigma}_{n|0:n-1} &= e^{\mathbf{A}\Delta t_n} \boldsymbol{\Sigma}_{n-1|0:n-1} (e^{\mathbf{A}\Delta t_n})^T + \tilde{\boldsymbol{\Sigma}}_n \\
&= \left[e^{\mathbf{A}\Delta t_n} \mathbf{L}_{n-1|0:n-1} (e^{\mathbf{A}\Delta t_n})^T + \hat{\boldsymbol{\Sigma}}_{\tau,n} \right] \sigma_v^2 \\
&= \sigma_v^2 \mathbf{L}_{n|0:n-1} \quad (6.44)
\end{aligned}$$

where both $\tilde{\boldsymbol{\Sigma}}_n$ and $\hat{\boldsymbol{\Sigma}}_{\tau,n}$ are defined in (6.37). Again use the Kalman update formulae to obtain the posterior parameters of \mathbf{x}_n after being “corrected” by the observation y_n :

$$\mathbf{Q}_n = \mathbf{L}_{n|0:n-1} \mathbf{G}^T (\mathbf{G} \mathbf{L}_{n|0:n-1} \mathbf{G}^T + \kappa_n)^{-1} \quad (6.45)$$

$$\begin{aligned}
\boldsymbol{\mu}_{n|0:n} &= \boldsymbol{\mu}_{n|0:n-1} + \mathbf{Q}_n (y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1}) \\
&= \boldsymbol{\mu}_{n|0:n-1}^c + \mathbf{d}_{n|0:n-1} \mu_J + \mathbf{Q}_n [y_n - \mathbf{G} (\boldsymbol{\mu}_{n|0:n-1}^c + \mathbf{d}_{n|0:n-1} \mu_J)] \\
&= [\mathbf{Q}_n y_n + (\mathbf{I} - \mathbf{Q}_n \mathbf{G}) \boldsymbol{\mu}_{n|0:n-1}^c] + (\mathbf{I} - \mathbf{Q}_n \mathbf{G}) \mathbf{d}_{n|0:n-1} \mu_J \\
&= \boldsymbol{\mu}_{n|0:n}^c + \mathbf{d}_{n|0:n} \mu_J \quad (6.46)
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_{n|0:n} &= (\mathbf{I} - \mathbf{Q}_n \mathbf{G}) \boldsymbol{\Sigma}_{n|0:n-1} \\
&= \sigma_v^2 (\mathbf{I} - \mathbf{Q}_n \mathbf{G}) \mathbf{L}_{n|0:n-1} \\
&= \sigma_v^2 \mathbf{L}_{n|0:n} \quad (6.47)
\end{aligned}$$

With μ_J and σ_v^2 in their symbolic form, the posterior parameters for x_n is obtained in the same format as those assumed for \mathbf{x}_{n-1} . Hence, it can be easily proved by induction that this conditional MKF inside the marginal RBPF algorithm can propagate throughout timestamps without having to evaluate the unknown parameters. However, algorithmically in addition to the coefficients of σ_v^2 , this marginal RBPF also needs to keep separately track of both the constant term $\boldsymbol{\mu}^c$ and the coefficient term \mathbf{d} of the posterior mean.

Marginal VRPF

With the state vectors dealt with by the conditional MKF, the inference for the non-Gaussian jump times were handled elegantly by a VRPF in the original paper [13]. Similarly, the general framework of the VRPF algorithm can also be used in this case as the jump proposal only depends on the hyperparameter λ_J . However, one problem arises from the use of symbolic μ_J and σ_v^2 . As derived in **Chapter 3**, the sequence of proposed jump times $\{\tau\}_{0:n}^{(i)}$ of the i th particle should be re-weighted at time t_n according to its

conditional PED as:

$$\begin{aligned}
\tilde{w}_n^{(i)} &= w_{n-1}^{(i)} \times p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^{(i)}, \sigma_v^2, \mu_J) \\
&= w_{n-1}^{(i)} \times \mathcal{N}(y_n | \mathbf{G}\boldsymbol{\mu}_{n|0:n-1}^{(i)}, \mathbf{G}\boldsymbol{\Sigma}_{n|0:n-1}^{(i)}\mathbf{G}^T + \kappa_w \sigma_v^2) \\
&= w_{n-1}^{(i)} \times \mathcal{N}(y_n | \mathbf{G}(\boldsymbol{\mu}_{n|0:n-1}^{c(i)} + \mathbf{d}_{n|0:n-1}^{(i)} \mu_J), \sigma_v^2(\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)}\mathbf{G}^T + \kappa_w))
\end{aligned} \tag{6.48}$$

where $w_{n-1}^{(i)}$ is the normalised weight at t_{n-1} ; and both $\boldsymbol{\mu}_{n|0:n-1}^{(i)}$ and $\boldsymbol{\Sigma}_{n|0:n-1}^{(i)}$ are predictive parameters from Eq. (6.43) and (6.44) conditioned on the proposed jump sequence $\{\tau\}_{0:n}^{(i)}$. Apparently for weight evaluation, μ_J and σ_v^2 can no longer be kept as unknown symbols as numerical weights are required by the VRPF to perform resampling and avoid particle degeneracy. This issue is solved here by integrating out the parameters with respect to their online joint posterior learned with the data observed so far in the process i.e. $p(\mu_J, \sigma_v^2 | y_{1:n-1}, \{\tau\}_{0:n-1}^{(i)})$.

In order to perform the integration, the online posterior of (μ_J, σ_v^2) needs to be obtained first. Fortunately this online posterior can be inferred in closed-form with the conjugate normal-inverse-Gamma prior:

$$\begin{aligned}
&p(\mu_J, \sigma_v^2 | y_{1:n-1}, \{\tau\}_{0:n-1}^{(i)}) \\
&= p(\mu_J, \sigma_v^2) p(y_1 | \mu_J, \sigma_v^2, \{\tau\}_{0:1}^{(i)}) \prod_{n'=2}^{n-1} p(y_{n'} | y_{1:n'-1}, \mu_J, \sigma_v^2, \{\tau\}_{0:n'}^{(i)}) \\
&= \mathcal{NIG}(\mu_J, \sigma_v^2 | \hat{m}_{n-1}^{(i)}, \hat{\nu}_{n-1}^{(i)}, \hat{\alpha}_{n-1}^{(i)}, \hat{\beta}_{n-1}^{(i)})
\end{aligned} \tag{6.49}$$

with hyperparameters:

$$\hat{\alpha}_{n-1}^{(i)} = \alpha + \frac{n-1}{2} \tag{6.50}$$

$$\hat{\nu}_{n-1}^{(i)} = \nu + \sum_{n'=1}^{n-1} \frac{(\mathbf{G}\mathbf{d}_{n'|0:n'-1}^{(i)})^2}{\mathbf{G}\mathbf{L}_{n'|0:n'-1}^{(i)}\mathbf{G}^T + \kappa_w} \tag{6.51}$$

$$\hat{m}_{n-1}^{(i)} = \frac{1}{\hat{\nu}_{n-1}^{(i)}} \left(\nu m + \sum_{n'=1}^{n-1} \frac{(y_{n'} - \mathbf{G}\boldsymbol{\mu}_{n'|0:n'-1}^{c(i)})\mathbf{G}\mathbf{d}_{n'|0:n'-1}^{(i)}}{\mathbf{G}^{(i)}\mathbf{L}_{n'|0:n'-1}^{(i)}\mathbf{G}^T + \kappa_w} \right) \tag{6.52}$$

$$\hat{\beta}_{n-1}^{(i)} = \beta + \frac{1}{2} \left(\nu m^2 - \hat{\nu}_{n-1}^{(i)} (\hat{m}_{n-1}^{(i)})^2 + \sum_{n'=1}^{n-1} \frac{(y_{n'} - \mathbf{G}\boldsymbol{\mu}_{n'|0:n'-1}^{c(i)})^2}{\mathbf{G}\mathbf{L}_{n'|0:n'-1}^{(i)}\mathbf{G}^T + \kappa_w} \right) \tag{6.53}$$

Alternatively, one can also construct recursive formulae and update the hyperparameters

obtained from the previous timestamp with the new observation i.e.:

$$\hat{\alpha}_n^{(i)} = \hat{\alpha}_{n-1}^{(i)} + \frac{1}{2} \quad (6.54)$$

$$\hat{\nu}_n^{(i)} = \hat{\nu}_{n-1}^{(i)} + \frac{(\mathbf{G}\mathbf{d}_{n|0:n-1}^{(i)})^2}{\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)}\mathbf{G}^T + \kappa_w} \quad (6.55)$$

$$\hat{m}_n^{(i)} = \frac{1}{\hat{\nu}_n^{(i)}} \left[\hat{\nu}_{n-1}^{(i)} \hat{m}_{n-1}^{(i)} + \frac{(y_n - \mathbf{G}\boldsymbol{\mu}_{n|0:n-1}^{c(i)}) \mathbf{G}\mathbf{d}_{n|0:n-1}^{(i)}}{\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)}\mathbf{G}^T + \kappa_w} \right] \quad (6.56)$$

$$\hat{\beta}_n^{(i)} = \hat{\beta}_{n-1}^{(i)} + \frac{1}{2} \left(\hat{\nu}_{n-1}^{(i)} (\hat{m}_{n-1}^{(i)})^2 - \hat{\nu}_n^{(i)} (\hat{m}_n^{(i)})^2 + \frac{(y_n - \mathbf{G}\boldsymbol{\mu}_{n|0:n-1}^{c(i)})^2}{\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)}\mathbf{G}^T + \kappa_w} \right) \quad (6.57)$$

Thus compute the marginal PED as a generalised student-t distribution for weight evaluation:

$$\begin{aligned} & p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^{(i)}, \hat{m}_{n-1}^{(i)}, \hat{\nu}_{n-1}^{(i)}, \hat{\alpha}_{n-1}^{(i)}, \hat{\beta}_{n-1}^{(i)}) \\ &= \iint p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^{(i)}, \sigma_v^2, \mu_J) p(\mu_J, \sigma_v^2 | y_{1:n-1}, \{\tau\}_{0:n-1}^{(i)}) d\sigma_v^2 d\mu_J \\ &= t_{2\hat{\alpha}_{n-1}^{(i)}} \left\{ y_n | \mathbf{G}(\boldsymbol{\mu}_{n|0:n-1}^{c(i)} + \mathbf{d}_{n|0:n-1}^{(i)} \hat{m}_{n-1}^{(i)}), \frac{\hat{\beta}_{n-1}^{(i)}}{\hat{\alpha}_{n-1}^{(i)}} (\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)}\mathbf{G}^T + \kappa_w + \frac{(\mathbf{G}\mathbf{d}_{n|0:n-1}^{(i)})^2}{\hat{\nu}_{n-1}^{(i)}}) \right\} \end{aligned} \quad (6.58)$$

With such formulation, the VRPF can proceed with the standard routine and obtain an empirical approximation of the (marginal) posterior of the jump times in $(t_0, t_N]$ at the end of this marginal VRPF:

$$p(\{\tau\}_{0:N} | Y) \approx \sum_{i=1}^{N_p} w_N^{(i)} \delta_{\{\tau\}_{0:N}^{(i)}}(\{\tau\}_{0:N}) \quad (6.59)$$

One should note that the learning of posterior hyperparamters of (μ_J, σ_v^2) concerns the entire continuous trajectory of $\boldsymbol{\mu}_{n|0:n-1}^{c(i)}$, $\mathbf{d}_{n|0:n-1}^{(i)}$ and $L_{n|0:n-1}^{(i)}$ from t_0 . Thus it is crucial to keep a consistent lineage of these values in addition to the jump times during the resampling stage of the PF. Therefore, the content of a single particle that should be propagated in this marginal RBPF algorithm (at time t_n) is defined as: (1) the full sequence of jump times up to the current timestamp t_n ; (2) the full trajectory of relevant terms in both predictive and posterior parameters; and (3) the particle's normalised weight. Write this as:

$$P_n^{(i)} = \left\{ \{\tau\}_{0:n}^{(i)}, \{\boldsymbol{\mu}_{k|0:k-1}^{c(i)}, \boldsymbol{\mu}_{k|0:k}^{c(i)}, \mathbf{d}_{k|0:k-1}^{(i)}, \mathbf{d}_{k|0:k}^{(i)}, \mathbf{L}_{k|0:k-1}^{(i)}, \mathbf{L}_{k|0:k}^{(i)}\}_{k=1}^n, w_n^{(i)} \right\} \quad (6.60)$$

Posterior of (μ_J, σ_v^2)

With the update formulae for online posterior described early on, the full posterior conditioned on a particular jump sequence can be readily obtained at the end of the marginal RBPF. The marginal posterior for (μ_J, σ_v^2) can therefore be acquired by incorporating the empirical posterior in (6.59) and integrating out the jump times:

$$\begin{aligned} p(\mu_J, \sigma_v^2 | Y) &= \int p(\mu_J, \sigma_v^2 | Y, \{\tau\}_{0:N}) p(\{\tau\}_{0:N} | Y) d\{\tau\}_{0:N} \\ &\approx \sum_{i=1}^{N_p} w_N^{(i)} \mathcal{NIG}(\mu_J, \sigma_v^2 | \hat{m}_N^{(i)}, \hat{\nu}_N^{(i)}, \hat{\alpha}_N^{(i)}, \hat{\beta}_N^{(i)}) \end{aligned} \quad (6.61)$$

which gives a (weighted) mixture of N_p normal-inverse-Gamma distributions.

Posterior state inference

The posterior state inference in the marginal RBPF can be performed with various approaches. One simple solution is presented here. Similar to the MKF, the posterior filtering parameters $\boldsymbol{\mu}_{n|0:n}$ and $\boldsymbol{\Sigma}_{n|0:n}$ normally cannot be used directly to perform the posterior state inference as it only contains information up till time t_n . However, the full trajectories of these parameters are kept over the course of the marginal RBPF, which is considered as a type of simple (online) smoothing operation [60] i.e. current and future observations can impact past distributions/parameters. It is therefore intuitive to simply take the expectation of these posterior parameters $\{\boldsymbol{\mu}_{n|0:n}^{(i)}\}_{n=1}^N$ and $\{\boldsymbol{\Sigma}_{n|0:n}^{(i)}\}_{n=1}^N$ for each particle with respect to its corresponding conditional posterior $p(\mu_J, \sigma_v^2 | Y, \{\tau\}_{0:N}^{(i)})$:

$$\begin{aligned} &p(\mathbf{x}_n | Y, \{\tau\}_{0:N}^{(i)}) \\ &= \iint p(\mathbf{x}_n | Y, \mu_J, \sigma_v^2, \{\tau\}_{0:N}^{(i)}) p(\mu_J, \sigma_v^2 | Y, \{\tau\}_{0:N}^{(i)}) d\mu_J d\sigma_v^2 \\ &= \iint \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{n|0:n}^{c(i)} + \mathbf{d}_{n|0:n}^{(i)} \mu_J, \sigma_v^2 \mathbf{L}_{n|0:n}^{(i)}) \mathcal{NIG}(\mu_J, \sigma_v^2 | \hat{m}_N^{(i)}, \hat{\nu}_N^{(i)}, \hat{\alpha}_N^{(i)}, \hat{\beta}_N^{(i)}) d\mu_J d\sigma_v^2 \\ &= t_{2\hat{\alpha}} \left\{ \mathbf{x}_n | \boldsymbol{\mu}_{n|0:n}^{c(i)} + \mathbf{d}_{n|0:n}^{(i)} \hat{m}_N^{(i)}, \frac{\hat{\beta}_N^{(i)}}{\hat{\alpha}_N^{(i)}} (\mathbf{L}_{n|0:n}^{(i)} + \frac{\mathbf{d}_{n|0:n}^{(i)} (\mathbf{d}_{n|0:n}^{(i)})^T}{\hat{\nu}_N^{(i)}}) \right\} \end{aligned} \quad (6.62)$$

where the filtering parameters (e.g. $\boldsymbol{\mu}_{n|0:n}^{c(i)}$) are taken from the continuous trajectory of the particle in (6.60). Further taking expectations with respect to empirical jump posterior (6.59), the marginal posterior distribution of the state vector \mathbf{x}_n can be expressed as a weighted mixture of N_p multivariate student-t distributions:

$$p(\mathbf{x}_n | Y) = \sum_{i=1}^{N_p} w_N^{(i)} t_{2\hat{\alpha}} \left\{ \mathbf{x}_n | \boldsymbol{\mu}_{n|0:n}^{c(i)} + \mathbf{d}_{n|0:n}^{(i)} \hat{m}_N^{(i)}, \frac{\hat{\beta}_N^{(i)}}{\hat{\alpha}_N^{(i)}} (\mathbf{L}_{n|0:n}^{(i)} + \frac{\mathbf{d}_{n|0:n}^{(i)} (\mathbf{d}_{n|0:n}^{(i)})^T}{\hat{\nu}_N^{(i)}}) \right\} \quad (6.63)$$

Alternative smoothing algorithms such as fix-lag approximation and forward filtering-backward smoothing (FFBS) algorithm [89] can also be used in obtaining the smoothing distribution of the state vectors, which may result in improved performance.

Comparison to heavy-tailed filters

It may come to attention that the marginal RBPF has likelihood (i.e. weight-correcting PED) and posterior distributions of the state vectors both as student-t distributions. This provides a close connection to the heavy-tailed filter proposed in [134] where student-t distributions are used for both model likelihood and prior state transition density. Student-t distributions compared to Gaussian distributions are able to model more diverse noise types and thus have better accountability for heavy-tailed outliers. Such a feature is sometimes desirable in particular fields such as financial modelling where outliers can be detrimental to model-generated investment decisions.

Recall the integration performed in the marginal RBPF, the expectations are taken with respect to the online posterior $p(\mu_J, \sigma_v^2 | y_{1:n-1}, \{\tau\}_{0:n-1}^{(i)})$ of the parameters (μ_J, σ_v^2) . The heavy-tailed filter can thus be regarded as a special form of marginal RBPF where marginalisation (i.e. expectation with respect to the prior) of the parameters is performed instead of expectation with respect to the online posterior e.g.:

$$p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^{(i)}) = \iint p(y_n | y_{1:n-1}, \{\tau\}_{0:n}^{(i)}, \mu_J, \sigma_v^2) p(\mu_J, \sigma_v^2) d\mu_J d\sigma_v^2 \quad (6.64)$$

where $p(\mu_J, \sigma_v^2)$ is the parameter prior and note the difference compared to Eq. (6.58). With an appropriate choice of this prior, the heavy-tailed filter introduced in [134] can be reconstructed from this alternative formulation of the model. However, it is also clear that with marginalisation, the heavy-tailed filter gives up the ability of parameter learning and its performance can potentially be deteriorated with bad choices of hyperparameters (i.e. a non-informative prior for (μ_J, σ_v^2)).

6.2.2 Particle filter based variational inference

Unlike the MKF, the marginal RBPF does not preserve the symbolic form of the unknown parameters all the way throughout the sequential inference. This means that the propagation of marginal RBPF for the initial few timestamps is very similar to that of a heavy-tailed filter. Hence, there is still possibility that the marginal RBPF may overall perform unsatisfactorily because of a few bad proposals and weightings caused by a badly chosen prior at the early stage of the algorithm.

This lack of prior knowledge may be overcome by iteratively enhancing a “guess”

at the parameters. The PG algorithm proposed in [64] uses an iterative scheme where the “guess” for the parameters is progressively improved by learning from the sequential inference results obtained with parameter values sampled from the previous iteration (i.e. previous guess). However at each iteration of the PG algorithm, the learning of parameters is inefficient as it only admits information from a single particle trajectory importance-sampled at the end of PF and ignores all other particles.

The variational Bayes (VB) or variational inference (VI) algorithm [151, 152] refers to an algorithmic framework for approximating the (joint) posterior of a model. Similar to the loopy belief propagation [153] and the expectation propagation algorithm [154], the VI algorithm also falls into the class of optimisation-based approximate Bayesian inference approaches [155]. The algorithm has been employed extensively in the field of machine learning as a type of message-passing algorithm [119, 156] due to both its scalability [157] and promising convergence performance. With increasing popularity, a number of variants of the generic VI algorithm have been developed in the past decade including VI with different approximation methods, VI with alternative objectives and some more recent applications of the VI algorithm in deep and reinforcement learning, see the review [158] and references therein. Algorithms that combine the VI and the PF have also been studied in two recent papers [159] and [160]. However, both papers consider the integration from an optimisation perspective aiming to improve the inference of sequential states by optimising a PF-estimated objective function rather than focusing on the Bayesian parameters learning.

In this subsection, I propose a novel integration of variational inference (VI) and the particle filters to perform accurate posterior inference on parameters as well as sequential states with full usage of information from all particles. Furthermore, the proposed PF-based VI (termed PF-VI) algorithm can also be applied to a wider range of SSMs with non-linear and/or non-Gaussian dynamics. Some content of this subsection has been put into a paper that was recently accepted for the conference IEEE ICASSP 2021.

Introduction to VI

In order to achieve approximate posterior inference of all random variables, say $\theta = \{\theta_j\}_{j=1}^J$, given observations $Y = \{y_n\}_{n=1}^N$, the VI algorithm often starts by postulating a family/class of distributions $q(\theta)$ to approximate the true joint posterior $p(\theta | Y)$ in a closed form, i.e.:

$$p(\theta | Y) \approx q(\theta)$$

The algorithm then optimise iteratively the approximated posterior by minimising the Kullback-Leibler (KL) divergence [161, 162] between the target posterior and the postu-

lated family:

$$\text{KL}(q||p) = -\mathbb{E}_q\left(\log\left(\frac{p(\theta|Y)}{q(\theta)}\right)\right) = -\int \log\left(\frac{p(\theta|Y)}{q(\theta)}\right) q(\theta) d\theta \quad (6.65)$$

It is clear that the minimum of this divergence is 0, which is attained only when $q(\theta) \equiv p(\theta|Y)$. Hence, the VI algorithm is typically applied in models where the target posterior can neither be obtained analytically nor be sampled from easily. The algorithm then approximates the complicated (joint) posterior with distribution(s) $q(\theta)$ from a simpler (and restricted) class that are easier to infer in closed-form.

A standard and popular approach, called the coordinate ascent variational inference (CAVI) [155], for formulating the approximated posterior $q(\theta)$ is by using the *mean-field* approximation to independently factorise the joint posterior for each component θ_j , i.e.:

$$q(\theta | \Omega) = \prod_{j=1}^J q_j(\theta_j | \Omega_j) \quad (6.66)$$

where Ω_j denotes the hyperparameters of the approximated posteriors. The iterative algorithm of VI then proceeds as follows: cycling through each approximated posterior of the component θ_j , the algorithm updates the hyperparameters Ω_j so that:

$$\log q_j(\theta_j | \Omega_j) = \int \log p(\theta | Y) q_{-j}(\theta_{-j} | \Omega_{-j}) d\theta_{-j} \quad (6.67)$$

where the subscript $(-j)$ denotes all other components except j . With certain classes of distributions for $q(\theta)$, it is not necessary to compute the exact integration or expectation with respect to other approximated posteriors, but it only needs to match the hyperparameters with the terms that associate with θ_j for a functional form of the approximating distribution $q_j(\theta_j | \Omega_j)$.

It is not trivial to see how such operations would minimise the KL-divergence of (6.65). I will prove it here as it also provides important insight into the later extension with the PF. Start by re-writing the KL-divergence in an alternative form using the Bayes' rule:

$$\begin{aligned} \text{KL}(q||p) &= -\int \log\left(\frac{p(\theta|Y)}{q(\theta)}\right) q(\theta) d\theta \\ &= -\int \log\left(\frac{p(\theta, Y)}{q(\theta)}\right) q(\theta) d\theta + \int \log p(y) q(\theta) d\theta \\ &= -\mathbb{E}_q\left(\log\left(\frac{p(\theta, Y)}{q(\theta)}\right)\right) + \underbrace{\log p(y)}_{\text{const.}} \end{aligned} \quad (6.68)$$

The expectation in the last line is also commonly known as the *variational lower bound* or the *evidence lower bound* (ELBO) [163, 155]. Clearly, this new form of KL-divergence

is simply the negative ELBO subject to an additive constant. Suppose the VI algorithm is updating the approximated posterior for a specific component θ_j , we can factorise the joint distribution $q(\theta) = q_j(\theta_j) q_{-j}(\theta_{-j})$ and decompose the negative ELBO as:

$$\begin{aligned}
-\mathbb{E}_q\left(\log\left(\frac{p(\theta, Y)}{q(\theta)}\right)\right) &= \iint q_j(\theta_j) q_{-j}(\theta_{-j}) \left[-\log p(\theta, Y) + \log q_j(\theta_j) + \log q_{-j}(\theta_{-j}) \right] d\theta_j d\theta_{-j} \\
&= - \int q_j(\theta_j) \underbrace{\left(\int q_{-j}(\theta_{-j}) \log p(\theta, Y) d\theta_{-j} \right)}_{\textcircled{1}} d\theta_j + \\
&\quad \int q_j(\theta_j) \log q_j(\theta_j) d\theta_j + \int q_{-j}(\theta_{-j}) \log q_{-j}(\theta_{-j}) d\theta_{-j}
\end{aligned} \tag{6.69}$$

As the “current” iteration of the VI algorithm only updates $q_j(\theta_j)$, the last term of the above expression (i.e. the entropy of $q_{-j}(\theta_{-j})$) can be considered as a constant as it does not depend on $q_j(\theta_j)$. Furthermore, notice that $\textcircled{1}$ has integrated out all other component θ_{-j} and is a function of solely θ_j and Y . It can thus be re-written into the log of normalised density of θ_j and a constant term:

$$\textcircled{1} = \log \tilde{p}(\theta_j) + \text{const.} \tag{6.70}$$

and hence the KL-divergence becomes:

$$\begin{aligned}
\text{KL}(q||p) &= - \int q_j(\theta_j) \log \left(\frac{\tilde{p}(\theta_j)}{q(\theta_j)} \right) d\theta_j + \text{const.} \\
&= \text{KL}(q_j(\theta_j) || \tilde{p}(\theta_j)) + \text{const.}
\end{aligned} \tag{6.71}$$

which is minimised when $q_j(\theta_j) \equiv \tilde{p}(\theta_j)$. It is clear now how every iteration of the VI algorithm is able to reduce the overall KL-divergence by updating the hyperparameters Ω_j to equate:

$$\log q_j(\theta_j) = \langle \log p(\theta|Y) \rangle_{q_{-j}(\theta_{-j})} = \langle \log p(\theta, Y) \rangle_{q_{-j}(\theta_{-j})} + \text{const.} \tag{6.72}$$

where $\langle \cdot \rangle$ is the expectation operator. Despite that the first equality is the formal definition of the VI algorithm, in practice it is much more feasible to operate on the second equality using joint probability.

VI-PF integration

Going back to the jump-diffusion model discussed, the VI algorithm is employed by first approximating the joint posterior of jumps, state vectors and parameters with a factorisation:

$$p(\mathbf{x}_{0:N}, \{\tau\}_{0:N}, \mu_J, \sigma_v^2 | Y) \approx q(\mu_J, \sigma_v^2) q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \tag{6.73}$$

Clearly, the random variables of the model are separated into a set of static parameters and a set of time-indexed sequential variables. And the VI algorithm will perform iterative updates on the approximated posteriors of these two sets of random variables. While it is intuitive to choose the functional form of $q(\mu_J, \sigma_v^2)$ as a normal-inverse-Gamma distribution, it is generally hard to comprehend the exact type of distribution for $q(\mathbf{x}_{0:N}, \{\tau\}_{0:N})$ due to the non-linearity and non-Gaussianity of the SSM. Regardless, follow the standard VI procedure:

$$\begin{aligned}
& \log q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \\
& \equiv \langle \log p(\mathbf{x}_{0:N}, \{\tau\}_{0:N}, \mu_J, \sigma_v^2, Y) \rangle_{q(\mu_J, \sigma_v^2)} + \text{const.} \\
& = \langle \log \left\{ p(\mu_J, \sigma_v^2) p(\mathbf{x}_0 | \mu_J, \sigma_v^2) \prod_{n=1}^N [p(\{\tau\}_{n-1:n}) p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \right. \\
& \quad \left. \times p(y_n | \mathbf{x}_n, \sigma_v^2)] \right\} \rangle_{q(\mu_J, \sigma_v^2)} + \text{const.} \\
& = \langle \log p(\mathbf{x}_0 | \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \sum_{n=1}^N \left\{ \log p(\{\tau\}_{n-1:n}) + \right. \\
& \quad \left. \langle \log p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \langle \log p(y_n | \mathbf{x}_n, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \right\} + \text{const.}
\end{aligned} \tag{6.74}$$

Apparently, this approximated posterior cannot be obtained in a closed form even given the freedom to choose which functional form it can take. Hence instead, we try to obtain an empirical representation of this approximated posterior using the particle filter. A similar recursive construction of the approximated posterior is derived for the particle filtering to take place. The derivation is as follows:

$$\begin{aligned}
& \log q(\mathbf{x}_0) = \langle \log p(\mathbf{x}_0 | \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \langle \log p(\mathbf{x}_1 | \mathbf{x}_0, \{\tau\}_{0:1}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \text{const.} \\
& \log q(\mathbf{x}_{0:1}, \{\tau\}_{0:1}) = \log q(\mathbf{x}_0) + \log p(\{\tau\}_{0:1}) + \langle \log p(y_1 | \mathbf{x}_1, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \\
& \quad \langle \log p(\mathbf{x}_2 | \mathbf{x}_1, \{\tau\}_{1:2}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \text{const.} \\
& \log q(\mathbf{x}_{0:2}, \{\tau\}_{0:2}) = \log q(\mathbf{x}_{0:1}, \{\tau\}_{0:1}) + \log p(\{\tau\}_{1:2}) + \langle \log p(y_2 | \mathbf{x}_2, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \\
& \quad \langle \log p(\mathbf{x}_3 | \mathbf{x}_2, \{\tau\}_{2:3}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \text{const.} \\
& \quad \vdots \quad = \quad \quad \quad \vdots
\end{aligned}$$

Denote intermediate distributions $\tilde{q}(\mathbf{x}_{0:n}, \{\tau\}_{0:n})$ for $n = 1, \dots, N$, such that:

$$\begin{aligned}
\log \tilde{q}(\mathbf{x}_{0:n}, \{\tau\}_{0:n}) &= \log q(\mathbf{x}_{0:n}, \{\tau\}_{0:n}) - \langle \log p(\mathbf{x}_{n+1} | \mathbf{x}_n, \{\tau\}_{n:n+1}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \\
&= \log q(\mathbf{x}_{0:n-1}, \{\tau\}_{0:n-1}) + \log p(\{\tau\}_{n-1:n}) + \\
&\quad \langle \log p(y_n | \mathbf{x}_n, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \text{const.} \\
&= \log \tilde{q}(\mathbf{x}_{0:n-1}, \{\tau\}_{0:n-1}) + \log p(\{\tau\}_{n-1:n}) + \langle \log p(y_n | \mathbf{x}_n, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \\
&\quad + \langle \log p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \text{const.}
\end{aligned} \tag{6.75}$$

From the above, it can be seen that the recursion of the intermediate distributions with an initial condition $\log \tilde{q}(\mathbf{x}_0, \emptyset) = \langle \log p(\mathbf{x}_0 | \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)}$. Furthermore, the end condition can be obtained by inspection:

$$\log \tilde{q}(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \equiv \log q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \tag{6.76}$$

Hence, it is possible to employ a standard PF to obtain sequentially the particle representation of the intermediate distributions and consequently obtain the desired distribution $q(\mathbf{x}_{0:N}, \{\tau\}_{0:N})$ as long as the PF reaches the end of the recursion.

With the expectation taken over the parameters (μ_J, σ_v^2) , we still need to identify the proposal(s) and the weight-correction likelihood in the context of the VI algorithm. Taking an arbitrary propagation at time t_n , the sequential target distribution can be expressed as:

$$\begin{aligned}
\tilde{q}(\mathbf{x}_{0:n}, \{\tau\}_{0:n}) &= \tilde{q}(\mathbf{x}_{0:n-1}) p(\{\tau\}_{n-1:n}) \exp\{\langle \log p(y_n | \mathbf{x}_n, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)}\} \times \\
&\quad \exp\{\langle \log p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)}\} \times \text{const.}
\end{aligned} \tag{6.77}$$

First look at the expected transition density term:

$$\begin{aligned}
&\exp\{\langle \log p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)}\} \\
&= \exp\left\{-\frac{1}{2}\left(\log 2\pi + D \langle \log \sigma_v^2 \rangle_{q(\mu_J, \sigma_v^2)}\right) + \log |\hat{\Sigma}_{\tau,n}| + \right. \\
&\quad \left. \langle (\mathbf{x}_n - \tilde{\boldsymbol{\mu}}_n)^T (\sigma_v^2 \hat{\Sigma}_{\tau,n})^{-1} (\mathbf{x}_n - \tilde{\boldsymbol{\mu}}_n) \rangle_{q(\mu_J, \sigma_v^2)}\right\}
\end{aligned}$$

where both $\tilde{\boldsymbol{\mu}}_n$ and $\hat{\Sigma}_{\tau,n}$ are defined in (6.35) and (6.37) respectively; and D is the dimension of state vector. The closed-form solution to the expected transition density is non-trivial and the complete derivation is shown in **Appendix 6.A**. Assuming the approximated posterior $q(\mu_J, \sigma_v^2) = \mathcal{NIG}(\mu_J, \sigma_v^2 | \hat{m}, \hat{\nu}, \hat{\alpha}, \hat{\beta})$, the obtained solution is shown

directly here:

$$\begin{aligned} & \exp\{\langle \log p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)}\} \\ &= \mathcal{N}\left(\mathbf{x}_n | e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1} + \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c} \hat{m}, \frac{\hat{\beta}}{\hat{\alpha}} \hat{\Sigma}_{\tau,n}\right) \times \text{const.} \end{aligned} \quad (6.78)$$

It is clear that a Gaussian transition density can still be maintained with expectations taken with respect to the parameters. Analogously, the expected likelihood term (with full derivation in **Appendix 6.A**) can be expressed as:

$$\exp\{\langle \log p(y_n | \mathbf{x}_n, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)}\} = \mathcal{N}(y_n | \mathbf{G} \mathbf{x}_n, \kappa_w \frac{\hat{\beta}}{\hat{\alpha}}) \times \text{const.} \quad (6.79)$$

Eq. (6.77), (6.78) and (6.79) fully define a SSM after \mathbf{x} and τ and hence the target $q(\mathbf{x}_{0:N}, \{\tau\}_{0:N})$ may be simulated with a PF having the same transition density and observation likelihood. As a result, one may obtain the approximated posterior in the form of a large collection of weighted particles:

$$q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \approx \sum_{i=1}^{N_p} w_N^{(i)} \delta_{\mathbf{x}_{0:N}, \{\tau\}_{0:N}}^{(i)}(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \quad (6.80)$$

With the inference for the sequential states completed, we proceed to the other phase of the algorithm and update the approximated posterior of parameters (μ_J, σ_v^2) by:

$$\begin{aligned} & \log q(\mu_J, \sigma_v^2) \\ & \equiv \langle \log p(\mathbf{x}_{0:N}, \{\tau\}_{0:N}, \mu_J, \sigma_v^2, Y) \rangle_{q(\mathbf{x}_{0:N}, \{\tau\}_{0:N})} + \text{const.} \\ &= \log p(\mu_J, \sigma_v^2) + \langle \log p(\mathbf{x}_0 | \mu_J, \sigma_v^2) + \sum_{n=1}^N \{ \log p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \\ & \quad + \log p(y_n | \mathbf{x}_n, \sigma_v^2) \} \rangle_{q(\mathbf{x}_{0:N}, \{\tau\}_{0:N})} + \text{const.} \\ &= \log p(\mu_J, \sigma_v^2) + \sum_{i=1}^{N_p} w_N^{(i)} \log p(\mathbf{x}_0^{(i)} | \mu_J, \sigma_v^2) + \sum_{i=1}^{N_p} w_N^{(i)} \sum_{n=1}^N \left\{ \log p(\mathbf{x}_n^{(i)} | \mathbf{x}_{n-1}^{(i)}, \{\tau\}_{n-1:n}^{(i)}, \mu_J, \sigma_v^2) \right. \\ & \quad \left. + \log p(y_n | \mathbf{x}_n^{(i)}, \sigma_v^2) \right\} + \text{const.} \end{aligned} \quad (6.81)$$

where the final line of the above equation is obtained by substituting in the particle representation of $q(\mathbf{x}_{0:N}, \{\tau\}_{0:N})$. Recall the transition density of (6.34):

$$p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) = \mathcal{N}(\mathbf{x}_n | e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1} + \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c} \mu_J, \sigma_v^2 \hat{\Sigma}_{\tau,n})$$

If we further denote $\mathbf{a}_n^{(i)} = \mathbf{x}_n^{(i)} - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1}^{(i)}$ and $\mathbf{u}_n^{(i)} = \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c}$ for simplicity,

the updates of the hyperparameters of $q(\mu_J, \sigma_v^2) = \mathcal{NIG}(\hat{m}, \hat{\nu}, \hat{\alpha}, \hat{\beta})$ should be performed as follows (assuming the same \mathcal{NIG} prior as (6.38)):

$$\hat{\alpha} = \alpha + \sum_{i=1}^{N_p} w_N^{(i)} \frac{N}{2} = \alpha + \frac{N}{2} \quad (6.82)$$

$$\hat{\nu} = \nu + \sum_{i=1}^{N_p} w_N^{(i)} \left\{ \sum_{n=1}^N (\mathbf{u}_n^{(i)})^T (\hat{\Sigma}_{\tau,n}^{(i)})^{-1} \mathbf{u}_n^{(i)} \right\} \quad (6.83)$$

$$\hat{m} = \frac{1}{\hat{\nu}} \left(\nu m + \sum_{i=1}^{N_p} w_N^{(i)} \left\{ \sum_{n=1}^N (\mathbf{a}_n^{(i)})^T (\hat{\Sigma}_{\tau,n}^{(i)})^{-1} \mathbf{u}_n^{(i)} \right\} \right) \quad (6.84)$$

$$\hat{\beta} = \beta + \frac{1}{2} \left(\nu m^2 - \hat{\nu} \hat{m}^2 + \sum_{i=1}^{N_p} w_N^{(i)} \left\{ \sum_{n=1}^N (\mathbf{a}_n^{(i)})^T (\hat{\Sigma}_{\tau,n}^{(i)})^{-1} \mathbf{u}_n^{(i)} + \frac{(y_n - \mathbf{G} \mathbf{x}_n^{(i)})^2}{\kappa_w} \right\} \right) \quad (6.85)$$

Note that the effect of the initial state \mathbf{x}_0 have been omitted as the initialisation may vary across different applications and may even be deterministic. Clearly with the help of VI, the learning of parameters is now taking weighted contribution from all particles obtained during the sequential inference stage and thus potentially avoids the wasted information in the PMCMC. The experiment results in later section will show that this full utilisation of particle information is able to provide faster convergence compared to the PMCMC algorithm, where only a single particle is used at each iteration.

Furthermore, one may notice that the state vectors in the sequential inference phase are still maintained as linear-Gaussian conditioned on the jump sequence proposed. This means that a RBPF can easily be fitted into the general framework of PF-VI to improve the sequential inference performance. In such a case, the approximated posterior of the VI algorithm becomes $q(\{\tau\}_{0:N}) q(\mu_J, \sigma_v^2)$ and the sequential state inference will end up with a weighted particle collection for jump times only i.e.:

$$q(\{\tau\}_{0:N}) = \sum_{i=1}^{N_p} w_N^{(i)} \delta_{\{\tau\}_{0:N}^{(i)}}(\{\tau\}_{0:N}) \quad (6.86)$$

This then leads to fairly similar update formulae for the hyperparameters of $q(\mu_J, \sigma_v^2)$ as

those in the marginal RBPF in Subsection 6.2.1. The results are provided directly here:

$$\hat{\alpha} = \alpha + \frac{N}{2} \quad (6.87)$$

$$\hat{\nu} = \nu + \sum_{i=1}^{N_p} w_N^{(i)} \left\{ \sum_{n=1}^N \frac{(\mathbf{G} \mathbf{d}_{n|0:n-1}^{(i)})^2}{\mathbf{G} \mathbf{L}_{n|0:n-1}^{(i)} \mathbf{G}^T + \kappa_w} \right\} \quad (6.88)$$

$$\hat{m} = \frac{1}{\hat{\nu}} \left(\nu m + \sum_{i=1}^{N_p} w_N^{(i)} \left\{ \sum_{n=1}^N \frac{(y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1}^{c(i)}) \mathbf{G} \mathbf{d}_{n|0:n-1}^{(i)}}{\mathbf{G} \mathbf{L}_{n|0:n-1}^{(i)} \mathbf{G}^T + \kappa_w} \right\} \right) \quad (6.89)$$

$$\hat{\beta} = \beta + \frac{1}{2} \left(\nu m^2 - \hat{\nu} \hat{m}^2 + \sum_{i=1}^{N_p} w_N^{(i)} \left\{ \sum_{n=1}^N \frac{(y_n - \mathbf{G} \boldsymbol{\mu}_{n|0:n-1}^{c(i)})^2}{\mathbf{G} \mathbf{L}_{n|0:n-1}^{(i)} \mathbf{G}^T + \kappa_w} \right\} \right) \quad (6.90)$$

However in contrast to the marginal RBPF, the approximated parameter posterior here is a single \mathcal{NIG} distribution taking weighted contribution from all particles instead of a weighted mixture of N_p particle-specific \mathcal{NIG} distributions.

A link to marginal filters and heavy-tailed filters

Both the marginal RBPF and the heavy-tailed filter introduced in this chapter involve integration of Gaussian distributions with respect to parameters that follow conjugate distributions. Whether the integration is a marginalisation or not, it can always be regarded as an expectation operation on the parameters. In this subsection, we establish a link between the PF-based VI algorithm and the filters that require integration of the parameters.

In the sequential inference phase of the PF-VI algorithm, the targeted KL-divergence is minimised by updating the approximated posterior $q(\mathbf{x}_{0:N}, \{\tau\}_{0:N})$ so that the following equality holds:

$$\log q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \equiv \langle \log p(\mathbf{x}_{0:N}, \{\tau\}_{0:N}, \mu_J, \sigma_v^2, Y) \rangle_{q(\mu_J, \sigma_v^2)} + \text{const.}$$

Recall that this is derived from the expanded version of the KL-divergence in Eq. (6.68). Fitting into the jump diffusion model, this can be re-written as:

$$\begin{aligned}
\text{KL}(q||p) &= - \iint q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \left\{ \langle \log p(\mathbf{x}_{0:N}, \{\tau\}_{0:N}, \mu_J, \sigma_v^2, Y) \rangle_{q(\mu_J, \sigma_v^2)} \right. \\
&\quad \left. - \log q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \right\} d\mathbf{x}_{0:N} d\{\tau\}_{0:N} + \text{const.} \\
&= - \iint q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \left\{ \langle \log p(\mathbf{x}_0 | \mu_J, \sigma_v^2) + \sum_{n=1}^N \left[\log p(\{\tau\}_{n-1:n}) + \right. \right. \\
&\quad \left. \left. \log p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) + \log p(y_n | \mathbf{x}_n, \mu_J, \sigma_v^2) \right] \rangle_{q(\mu_J, \sigma_v^2)} \right. \\
&\quad \left. - \log q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \right\} d\mathbf{x}_{0:N} d\{\tau\}_{0:N} + \text{const.} \\
&\stackrel{\text{Jensen}}{\leq} - \iint q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \left\{ \log \langle p(\mathbf{x}_0 | \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \sum_{n=1}^N \left[\log p(\{\tau\}_{n-1:n}) + \right. \right. \\
&\quad \left. \left. \log \langle p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \log \langle p(y_n | \mathbf{x}_n, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \right] \right. \\
&\quad \left. - \log q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) \right\} d\mathbf{x}_{0:N} d\{\tau\}_{0:N} + \text{const.}
\end{aligned}$$

With *Jensen's inequality*, we may push the expectation inside the log as shown in the final line of the above equation and obtain an upper-bound on the targeted KL-divergence. In order to minimise this upper bound, an update formula for the alternative approximated posterior can be obtained following the final line of the equation above:

$$\begin{aligned}
\log q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) &\equiv \log \langle p(\mathbf{x}_0 | \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \sum_{n=1}^N \left[\log p(\{\tau\}_{n-1:n}) + \right. \\
&\quad \left. \log \langle p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} + \right. \\
&\quad \left. \log \langle p(y_n | \mathbf{x}_n, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \right] + \text{const.}
\end{aligned}$$

Exponentiate both sides:

$$\begin{aligned}
q(\mathbf{x}_{0:N}, \{\tau\}_{0:N}) &\propto \langle p(\mathbf{x}_0 | \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \prod_{n=1}^N \left[p(\{\tau\}_{n-1:n}) \times \right. \\
&\quad \left. \langle p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \langle p(y_n | \mathbf{x}_n, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \right] \\
&\hspace{15cm} (6.91)
\end{aligned}$$

Clearly, this alternative form of the approximated posterior cannot be obtained in closed-form, but instead can be inferred with a PF. Furthermore, the expectations taken with respect to the (approximated) parameter posterior resemble a heavy-tailed particle filter where both transition density of the state vector and observational likelihood follow student-t distributions. However, in contrast to the generic heavy-tailed filter where expectations are taken with respect to prior and the marginal RBPF where the expectation is taken with respect to an online posterior, this heavy-tailed filter of the VI algorithm

uses an improving but static (within an iteration) posterior $q(\mu_J, \sigma_v^2)$ at each iteration.

As a result, this variant of the PF-based VI algorithm iteratively optimises the KL-divergence during the parameter-learning stage and optimises the upper-bound of KL-divergence during the PF stage, which can still achieve convergence towards the approximated posteriors. Compared with the standard PF-VI algorithm introduced earlier, this variant can be sub-optimal due to its different objectives at the two stages of optimisation, but the modified heavy-tailed filter may provide additional accountability for outliers caused by model mismatch and better risk/uncertainty quantification (in financial models), although it is not investigated in this thesis.

6.2.3 Hybrid PF-VI

So far in this chapter, we have always assumed that the mean-reverting coefficient θ is a known hyperparameter. In many cases, the value of θ can also significantly influence the dynamic process and hence should be learnt from data. Due to the continuous-time construction of the SSM, θ is deeply involved in many operations of the transition density including matrix exponential, which makes it impossible to attain its posterior in closed-form or to compute its expectation in the VI algorithm. In order to handle this common issue in SSMs, I propose an algorithm which works as a hybrid of the PF-VI algorithm and the classic *Expectation-Maximisation* (EM) [164] algorithm.

To begin with, we reiterate the objective of the generic VI algorithm: to minimise the KL-divergence between the approximated posterior and the target true posterior. As shown in the derivation of generic VI algorithm, this objective can be readily translated into the maximisation of the *evidence lower bound* (ELBO) $\mathcal{O}(\theta)$:

$$\mathcal{O}(\theta) = \mathbb{E}_q\left(\log\left(\frac{p(\theta, Y)}{q(\theta)}\right)\right) = -\text{KL}(q||p) + \text{const.}$$

Note that θ here contains/represents all random variables in the model instead of the mean-reverting coefficient in Langevin dynamics. The ELBO, as an important performance measure, should be carefully monitored at each iteration of the VI algorithm for convergence purposes. In the jump-diffusion model of this section, the ELBO can be

computed as follows with the state vectors “Rao-Blackwellised” (i.e. using RBPF):

$$\begin{aligned}
& \mathcal{O}(Y, \{\tau\}_{0:N}, \mu_J, \sigma_v^2) \\
&= \mathbb{E}_{q(\mu_J, \sigma_v^2) q(\{\tau\}_{0:N})} \left\{ \log p(\mu_J, \sigma_v^2) + \sum_{n=1}^N [\log p(\{\tau\}_{n-1:n}) + \right. \\
& \quad \left. \log p(y_n | y_{0:n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2)] \right\} - \mathbb{E}_{q(\mu_J, \sigma_v^2)} (\log q(\mu_J, \sigma_v^2)) - \mathbb{E}_{q(\{\tau\}_{0:N})} (q(\{\tau\}_{0:N})) \\
&= \mathbb{E}_{q(\mu_J, \sigma_v^2) q(\{\tau\}_{0:N})} \left\{ \sum_{n=1}^N \log p(y_n | y_{0:n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \right\} - \underbrace{H(q(\mu_J, \sigma_v^2); p(\mu_J, \sigma_v^2))}_{\text{(i)}} \\
& \quad + \underbrace{H(q(\{\tau\}_{0:N}); p(\{\tau\}_{0:N}))}_{\text{(ii)}} + \underbrace{H(q(\mu_J, \sigma_v^2))}_{\text{(iii)}} + \underbrace{H(q(\{\tau\}_{0:N}))}_{\text{(iv)}} + \text{const.}
\end{aligned} \tag{6.92}$$

where **(i)** and **(ii)** are each the cross entropy between the approximated posterior and the corresponding prior; and both **(ii)** and **(iii)** are the entropies of the approximated posteriors. Fortunately, all the terms in (6.92) can be evaluated tractably including the constant term (although it remains the same throughout and is generally not of concern). The computations of the terms are however non-trivial as they involve the entropy of a \mathcal{NIG} distribution **(iii)**, the entropy of a weighted particle collection **(iv)**, the cross entropy between two \mathcal{NIG} distributions **(i)** and the cross entropy between an empirical and a closed-form distribution **(ii)**. The complete formula for the ELBO of the jump-diffusion model is derived in **Appendix 6.B**.

As long as the objective function can be obtained, it is possible to simply introduce the mean-reverting coefficient θ as an additional variable in this optimisation task and optimise θ in a non-Bayesian manner with gradient ascent (or descent of KL) algorithm at each iteration, e.g.:

$$\theta_{l+1} = \theta_l - \gamma \frac{\partial \mathcal{O}}{\partial \theta} \Big|_{\theta=\theta_l}$$

where γ is the learning rate/step size. This then gives a new sequence of iterative steps for this hybrid PF-VI algorithm:

1. Obtaining the particle representation of $q(\{\tau\}_{0:N})$ conditioned on $q(\mu_J, \sigma_v^2)$ using RBPF.
2. Closed-form inference of $q(\mu_J, \sigma_v^2)$ with particles obtained in Step 1.
3. Evaluate the ELBO and perform gradient ascent on parameter θ conditioned on both $q(\{\tau\}_{0:N})$ and $q(\mu_J, \sigma_v^2)$, and then go back to Step 1.

It is worth noting that the third step can also be performed with both $q(\mu_J, \sigma_v^2)$ and $q(\{\tau\}_{0:N})$ treated as functions of θ . However, this would incur even more complex relationship between θ and \mathcal{O} , which further prevents a tractable derivative from being

obtained. Hence, the optimisation of θ is performed in a similar manner as the EM algorithm.

Although the partial derivative of the ELBO may be attained via an analytic formula in certain models, the stochastic integration in the continuous-time jump-diffusion model did not make the evaluation of this partial derivative simple. Hence, this specific problem is tackled by employing a technique from computer science called *automatic differentiation* (AD) [165]. The use of AD has raised drastically in recent years thanks to the bloom of machine learning and neural networks [166]. AD first constructs a computational graph of (computer) functions which describes the functions' dependencies. By applying chain rule repeatedly to these functions/operations, AD is able to obtain total or partial derivatives of arbitrary order accurately to working precision. Therefore, it is most useful in systems where variables and functions are deeply entangled such as neural networks and in our case SSMs with (marginalised) sequential states.

Since a large proportion of the computational cost of AD is allocated to the construction of a computational graph, it may be wise to speed up the learning of θ by further utilising the second-order derivative (i.e. Hessian) of the objective function:

$$\theta_{l+1} = \theta_l - \gamma \left\{ \left[\frac{\partial^2 \mathcal{O}}{\partial \theta^2} \right]^{-1} \frac{\partial \mathcal{O}}{\partial \theta} \right\} \Big|_{\theta=\theta_l}$$

6.3 Results and discussions

In this section, I present experimental results obtained with both the marginal RBPF algorithm and the PF-VI algorithm on two sets of synthetic data. I demonstrate the parameter-learning ability of both algorithms on the jump-diffusion example used in this chapter. With an analysis on algorithms' costs and convergence behaviours, a conclusion is drawn on the relative merits and limitations of each algorithm.

Furthermore, the PF-VI algorithm is tested on a non-linear SSM and its performance is compared against two variants of the PMCMC algorithm proposed in [64]. The results demonstrate that the proposed PF-VI algorithm is able to achieve a much faster convergence speed with accurate approximation of the true posterior compared to the PMCMC algorithms.

6.3.1 Jump-diffusion example

I first apply both the marginal RBPF and the PF-VI algorithm on a synthesised jump-diffusion process. A total of 200 data points are simulated on an irregular time grid using

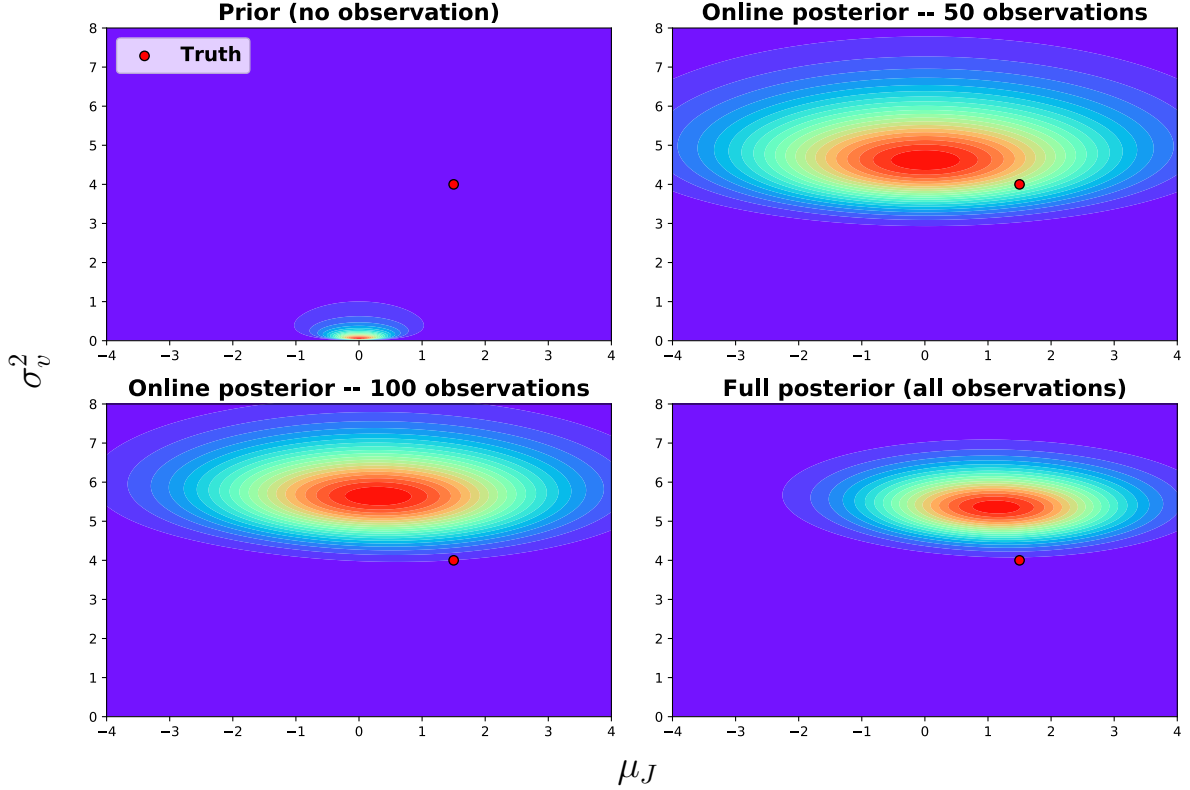


Figure 6.3.1: Marginal RBPF parameters' posteriors learned with increasing number of observations. The contour indicates the posterior distribution and the red dot indicates the true parametric values used in simulation.

the following parameters:

$$\begin{aligned} \theta &= -0.5 \quad , \quad \sigma_v^2 = 4.0 \quad , \quad \sigma_w^2 = 2.0, \\ \mu_J &= 1.5 \quad , \quad \sigma_J^2 = 40.0 \quad , \quad \lambda_J = 0.1 \end{aligned}$$

Inference-wise, $N_p = 2000$ particles are used for the PFs in both algorithms with a fairly uninformative \mathcal{NIG} prior:

$$p(\mu_J, \sigma_v^2) = \mathcal{NIG}(\mu_J, \sigma_v^2 | m = 0, \nu = 1, \alpha = 0.1, \beta = 0.1) \quad (6.93)$$

Note an assumption is made that the hyperparameters (i.e. $\theta, \lambda_J, \kappa_w, \kappa_J$) can be obtained accurately. **Figure 6.3.1** shows the online learning of parameters' posterior as the number of sequential observations increases. The top left panel shows the prior (i.e. no observation) whilst the bottom right panel shows the full posterior obtained at the end of the marginal RBPF. Note that the posteriors shown in **Figure 6.3.1** are the marginal posteriors computed as Eq. (6.61) with the jump times integrated out. It can be seen that the learning of parameters generally improves (in mean position and confidence interval) with more observations included. The final posterior learned with all observations is fairly accurate with a slight overestimation of σ_v^2 . It is also worth noting that despite the posi-

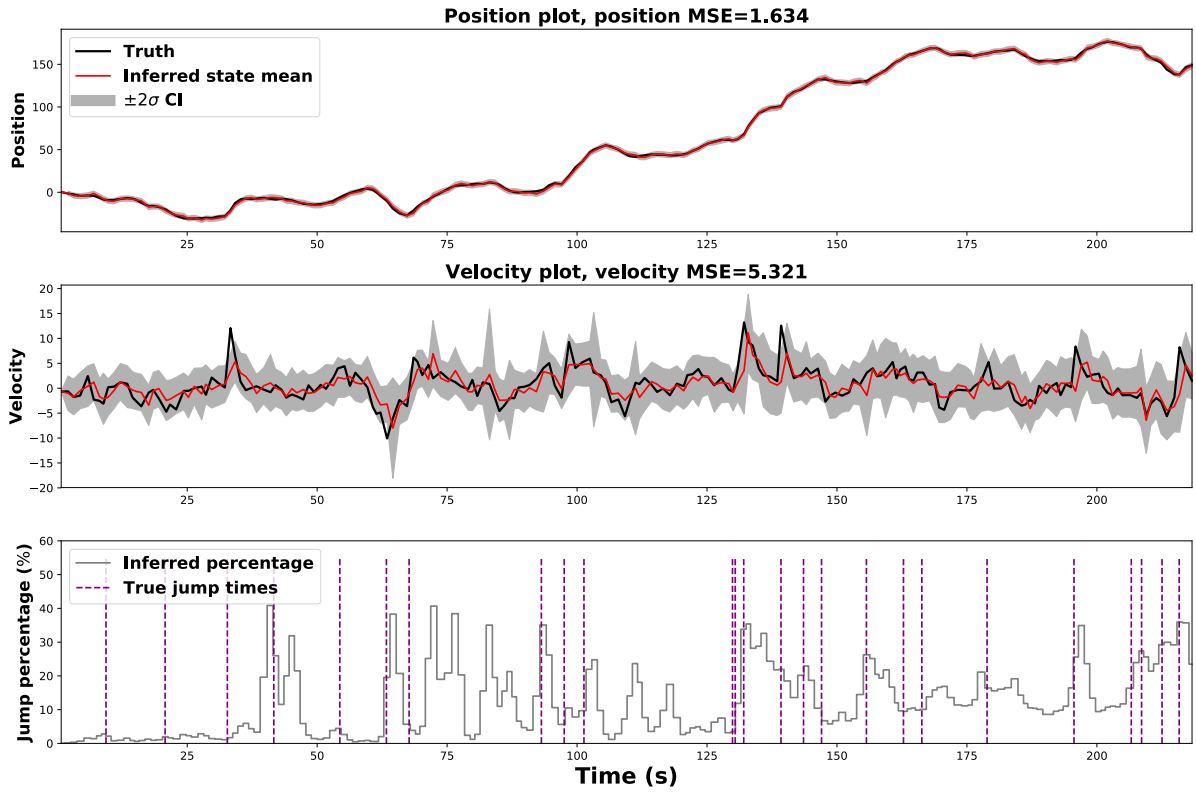


Figure 6.3.2: Marginal RBPF state inference results. The (weighted) posterior mean and 95% confidence interval are plotted with the ground truth. MSEs are shown in the titles. The bottom panel shows jump-time probability with the truth indicated as purple dashed lines.

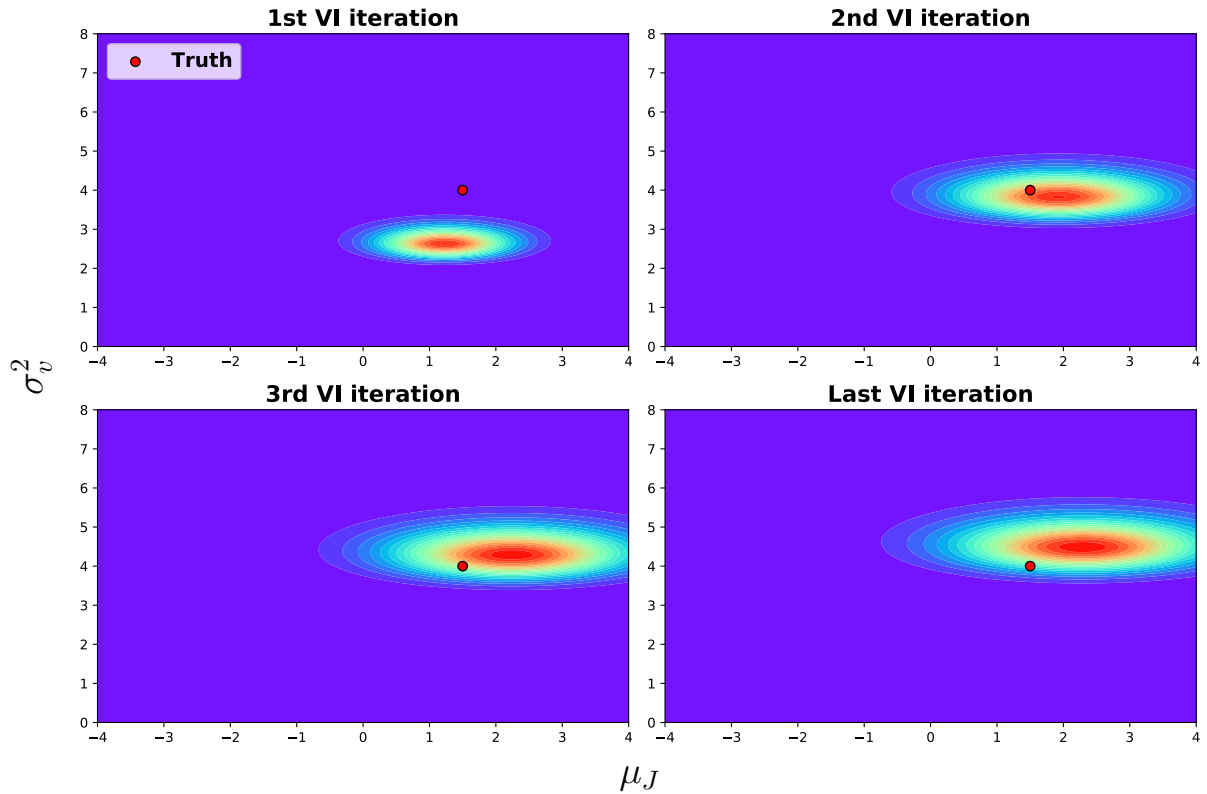


Figure 6.3.3: Approximated posteriors learned by PF-VI algorithm at different iterations.

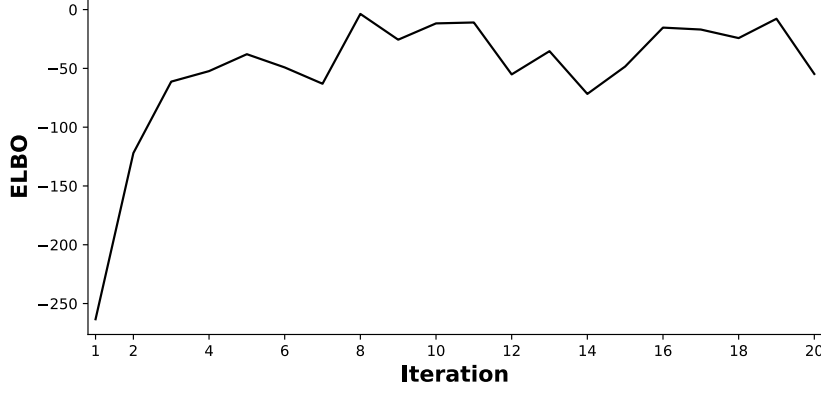


Figure 6.3.4: Evidence lower bound (ELBO) computed at each iteration of the PF-VI algorithm.

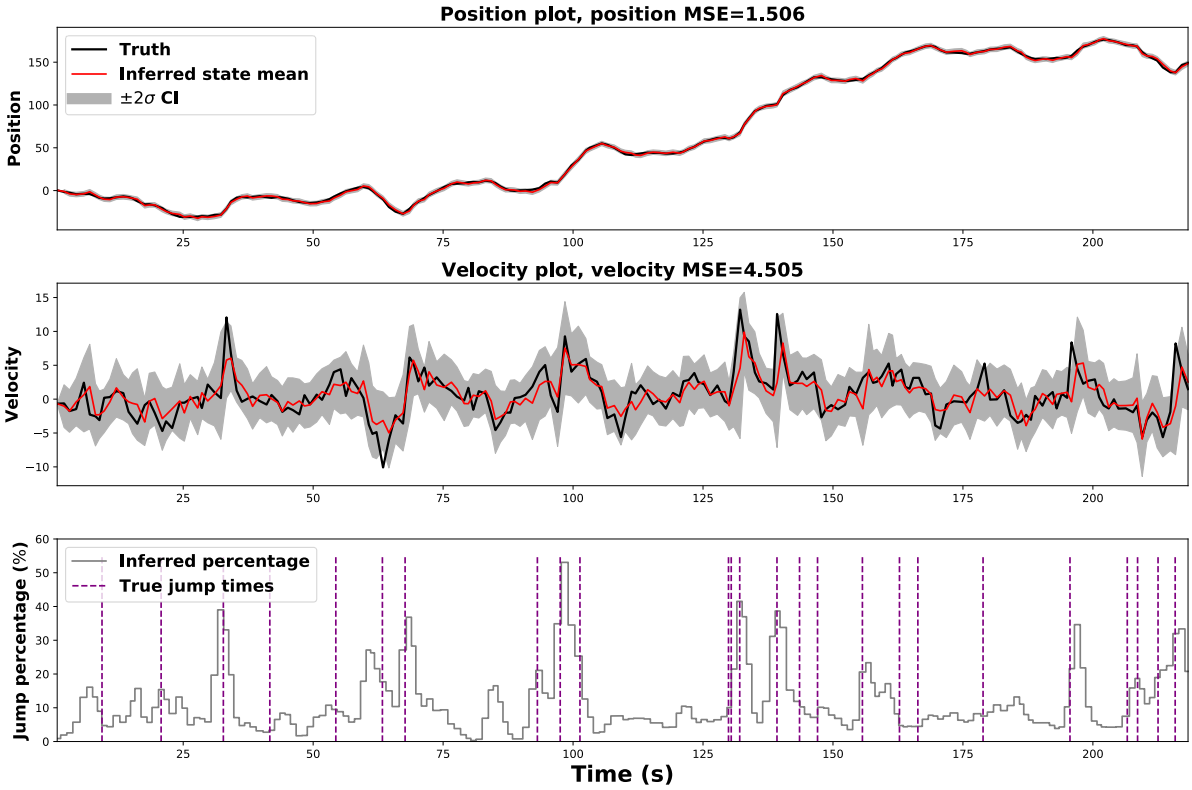


Figure 6.3.5: State and jump-time inference results obtained with PF-VI algorithm.

tive mean, the posterior variance of μ_J is very large. This is mainly caused by the large σ_J^2 used in the simulation. **Figure 6.3.2** shows the posterior inference results for both state vectors and jump times with corresponding mean-squared-errors (MSEs) indicated in the titles. The sequential inference accuracy, especially for jump times, is very poor for the initial few timestamps because of the integration with respect to an inaccurate online posterior in particle re-weightings. This performance gradually improves as the algorithm proceeds further with more data points being observed.

With the same N_p and prior, PF-VI algorithm is also applied to this synthetic jump-diffusion dataset with 20 VI iterations. **Figure 6.3.3** shows the approximated posteriors

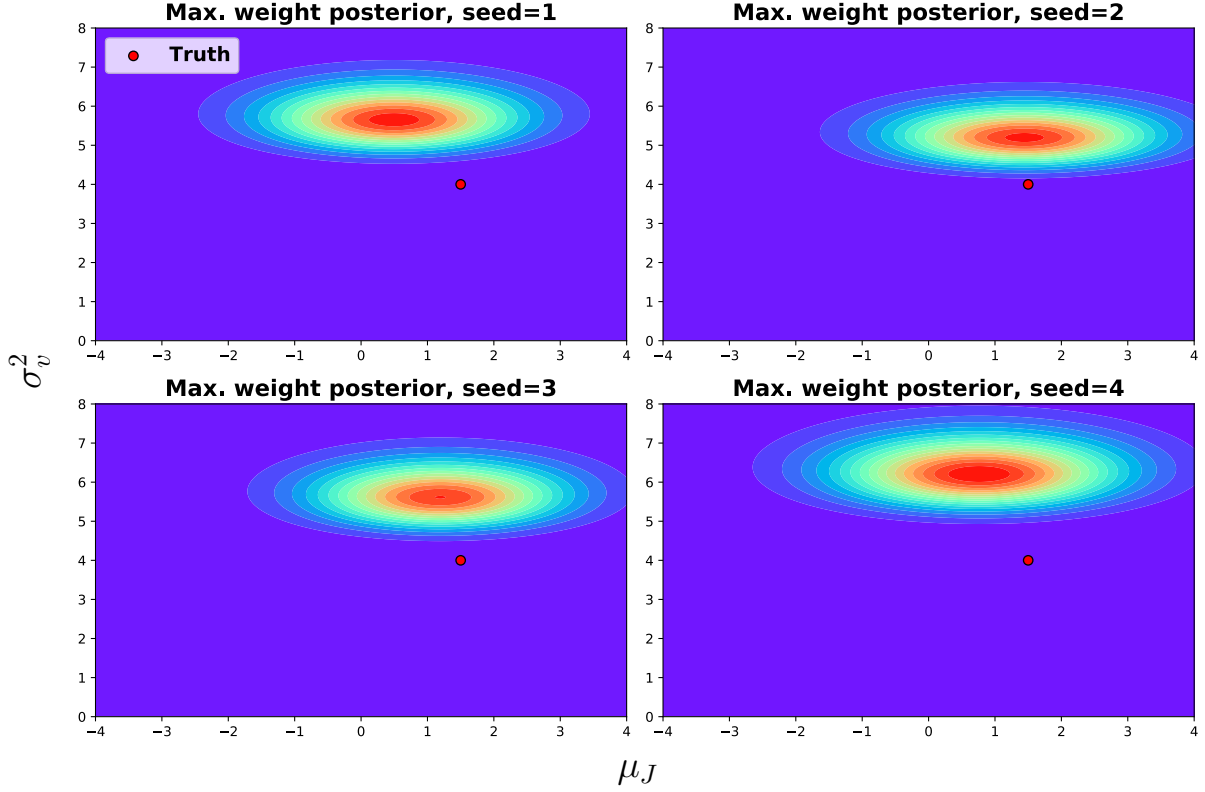


Figure 6.3.6: Conditional posteriors of the particle with the maximum weight $w_N^{(i)}$ in different random trials of the marginal RBPF.

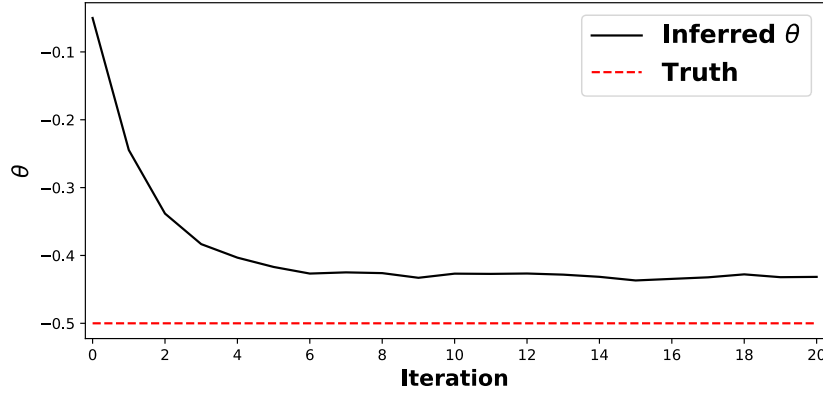


Figure 6.3.7: Learning curve of mean-reversion coefficient θ in the hybrid PF-VI algorithm.

inferred with the PF-VI algorithm at the 1st, 2nd, 3rd and last iterations. From the figure, the algorithm converges quickly as the 3rd iteration posterior is already almost identical to the posterior obtained in the final iteration. The fast convergence can also be observed from the ELBO computed at each iteration in **Figure 6.3.4**. Comparing to the posterior obtained via the marginal RBPF, the PF-VI achieves better mean accuracy and smaller uncertainty (i.e. variances) in the posterior. **Figure 6.3.5** shows the state inference results obtained in the last iteration of the PF-VI algorithm. In addition to the lower MSEs for both position and velocity comparing to the marginal RBPF, the PF-VI

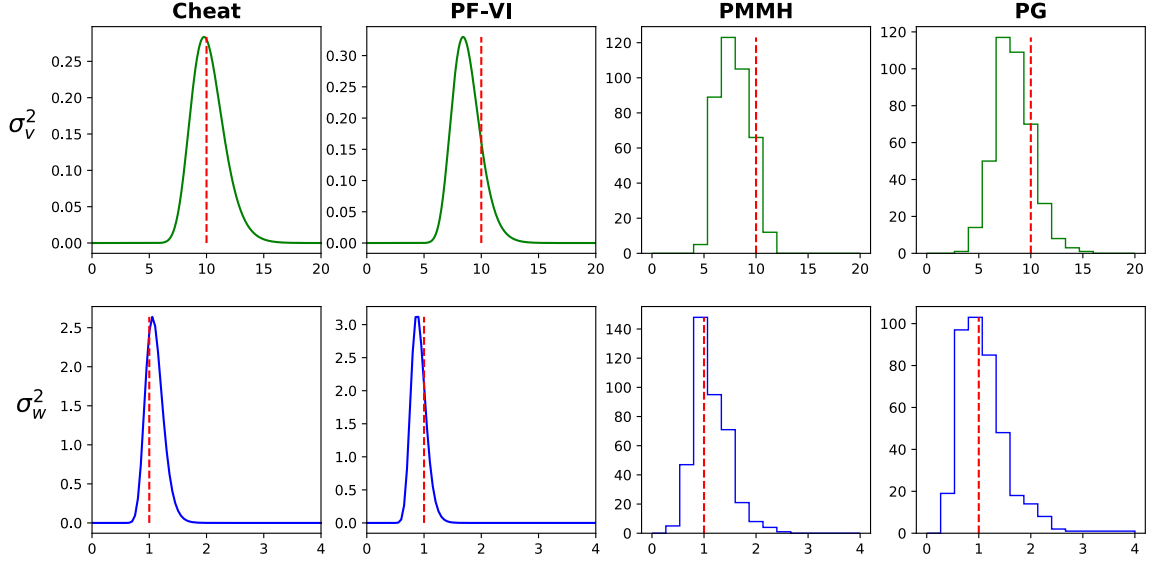
also shows a clear improvement in the jump-time inference accuracy.

Comparing further between the two proposed methods, it can be noted that the marginal RBPF requires less computational power as it only runs the PF once across the entire dataset with a complexity $\mathcal{O}(N \times N_p)$; whilst the PF-VI algorithm runs the PF for whatever number of times that ensures the algorithm’s convergence (despite the fast convergence). This makes the marginal RBPF more favourable for the task of online learning/adaptation of model parameters. On the other hand, the PF-VI has demonstrated more accurate posterior inference results by running multiple iterations across the dataset, which serves better as an offline learning scheme.

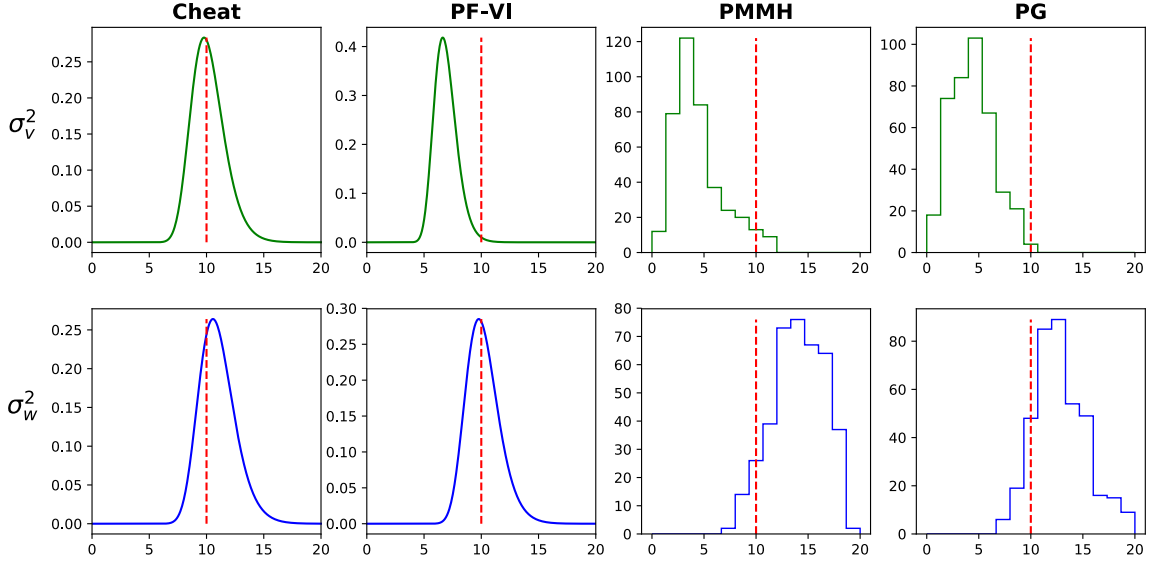
The inherent randomness of the PF always raises the concern of convergence. For marginal RBPF, this can be observed from **Figure 6.3.6**. The figure shows the conditional (conditioned on jump times) posteriors of the particles with the maximum weight at each trial of the marginal RBPF using a specific random seed. Clearly, this conditional posterior can have a fair amount of variability across different trials. However, empirical results have suggested that we can always obtain a consistent marginal posterior (i.e. the \mathcal{NIG} mixture) once the jump times are integrated out (not shown here). In generic VI algorithms, the ELBO at each VI iteration should be monotonically increasing. However, we see from **Figure 6.3.4** that the randomness of PF has removed this monotonicity of the ELBO in the PF-VI algorithm. Again based on empirical results, the convergence of the PF-VI algorithm can generally be guaranteed with a moderate amount of particles. While monotonicity can be achieved asymptotically as $N_p \rightarrow \infty$, it is also possible to achieve it by employing a more practical method of *backtracking* to accept the iterations that only increase the ELBO.

Figure 6.3.7 shows the learning of mean-reversion coefficient θ in the jump-diffusion model using the hybrid PF-VI algorithm. With an initial value $\theta_{\text{init}} = -0.05$ on the same synthetic dataset, θ is learned fairly successfully with a converged value that is close to the truth. The learning of θ may be improved by having smaller observation noise.

To summarise, both the marginal RBPF and the PF-VI algorithms have shown good performance in the posterior inference of the state vectors and parameters. The online framework and low computational cost of the marginal RBPF make it favourable for applications where parameters need to be learned based on a continuous stream of data. For good online inference accuracy, the marginal RBPF requires either a certain period of “data burn-in” or an informative prior to ensure appropriate weight evaluations using the online posterior. The PF-VI algorithm generally requires more computation than the marginal RBPF. However, it converges quickly and provides better inference accuracy by iteratively improving the posterior inference. Both algorithms have shown plausible



(a) $\sigma_v^2 = 10, \sigma_w^2 = 1$



(b) $\sigma_v^2 = 10, \sigma_w^2 = 10$

Figure 6.3.8: Posterior distributions and sample histograms obtained after 100 burn-in iterations. The true parametric values are indicated with red dashed lines. (a) is for the dataset with small observation noise; and (b) is for the dataset with large observation noise.

empirical convergence.

6.3.2 *Gordon-Kitagawa example*

In this experiment, a popular non-linear toy SSM [76, 60], which cannot be inferred with the marginal RBPF, is considered. The authors of [64] also employed this model to evaluate the performance of the PMCMC algorithms proposed in the paper. In this subsection, I present the results obtained with the PF-VI, the PG and the PMMH algorithms and

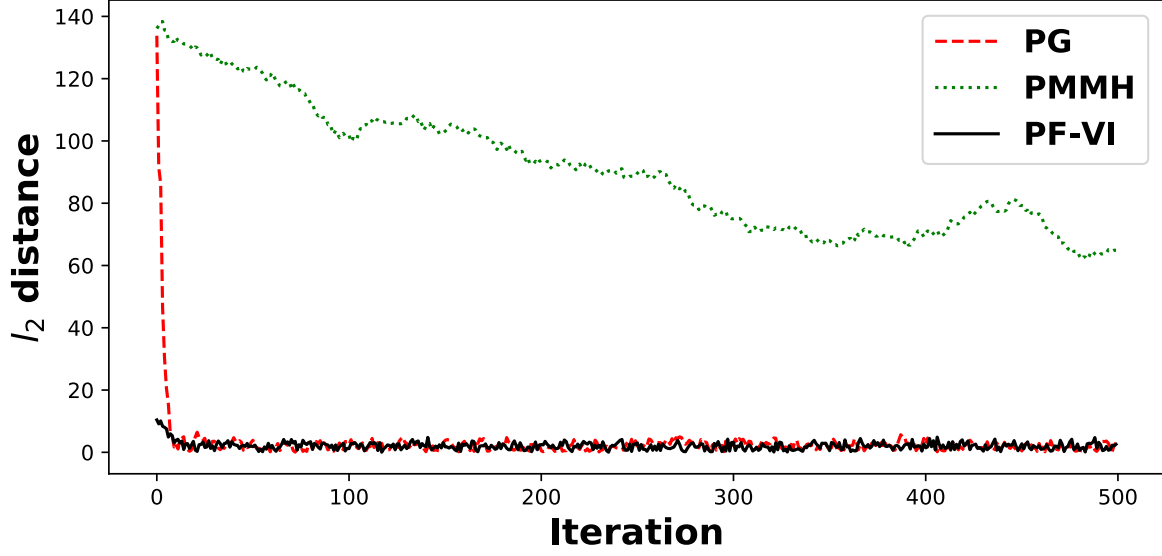


Figure 6.3.9: l_2 -distance between posterior samples and the truth. A fair initialisation with $\sigma_v^2(0) = \sigma_w^2(0) = 100$ was given to the PMMH and the PG algorithms. $N_p = 2000$ particles are used in the PF.

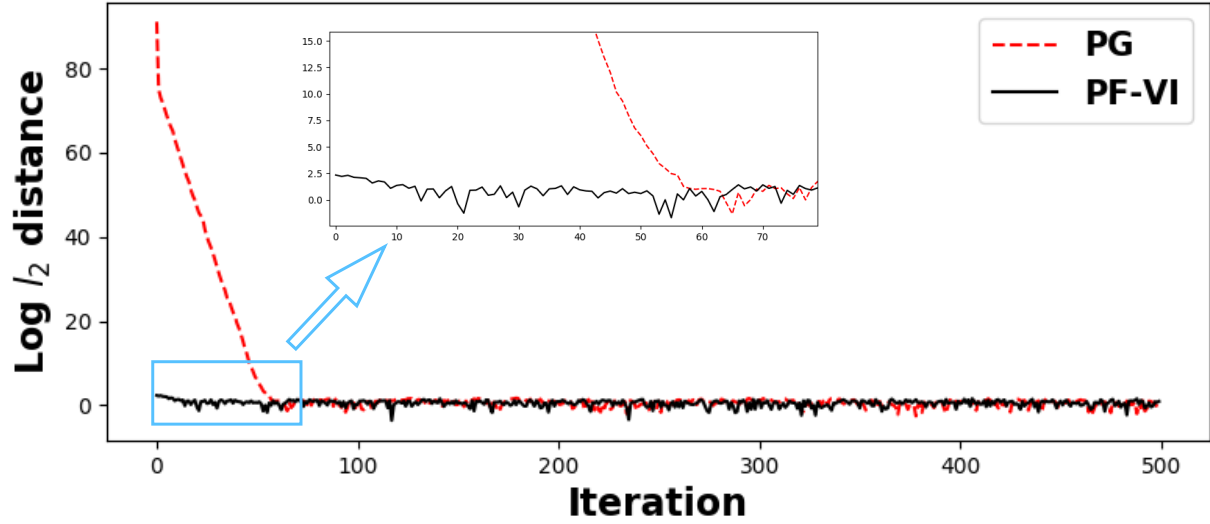


Figure 6.3.10: $\text{Log } l_2$ -distance between posterior samples and the truth. A very poor initialisation from the inverse-Gamma prior is provided to the PG algorithm. The PMMH results are not shown as the posterior samples did not converge at all.

compare the relative performance of these three iterative parameter-learning algorithms.

Consider the non-linear SSM with a transition function:

$$x_n = \frac{x_{n-1}}{2} + 25 \frac{x_{n-1}}{1 + x_{n-1}^2} + 8 \cos(1.2n) + v_n \quad (6.94)$$

and an observation model:

$$y_n = \frac{x_n^2}{20} + w_n \quad (6.95)$$

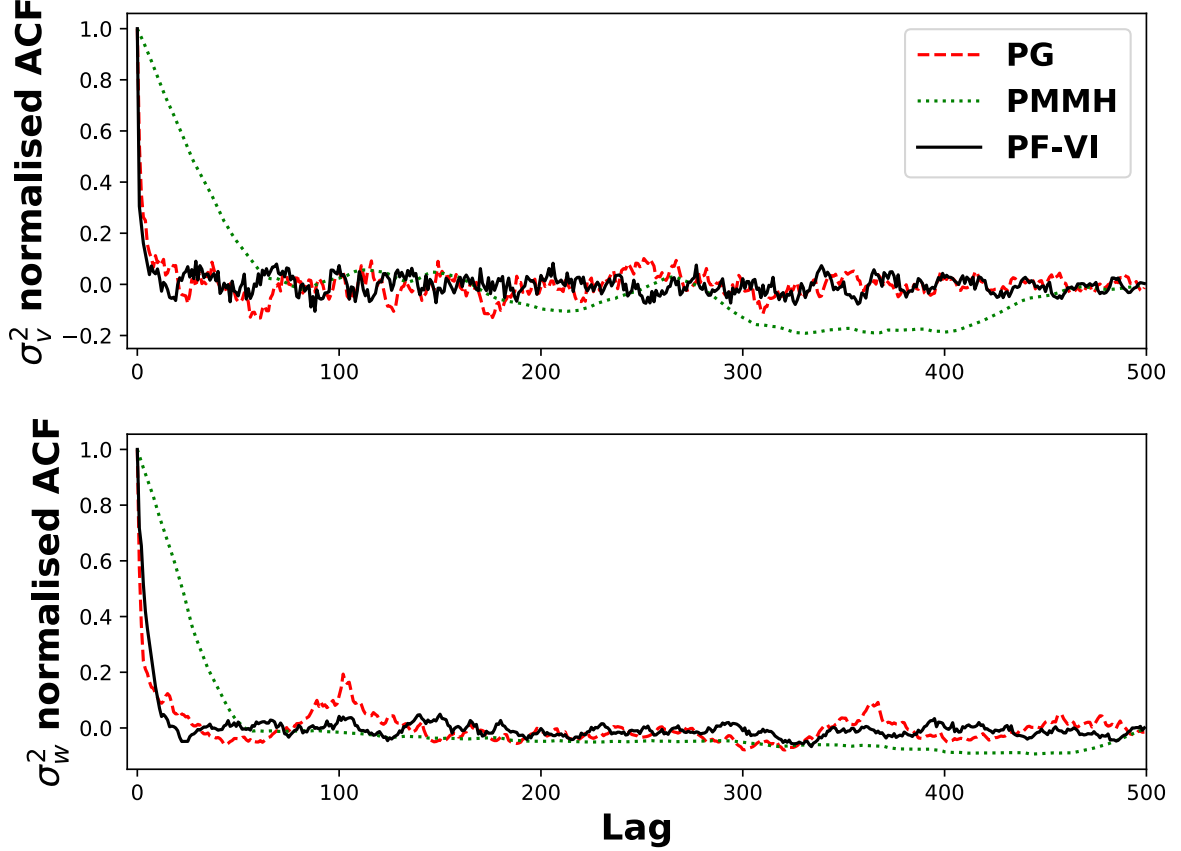


Figure 6.3.11: ACFs of the parameters obtained on the small observation noise dataset with good initialisation. $N_p = 2000$ particles are used in the PF.

where $x_1 \sim \mathcal{N}(0, 5)$, $v_n \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_v^2)$ and $w_n \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_w^2)$. It is obvious that in this model, there are two “priorly” independent parameters to infer, which denote here as $\theta = \{\sigma_v^2, \sigma_w^2\}$. This formulation of the SSM is often used to test the performance of the PF methods. Its conditional posterior of the states $p(x_{1:N} | y_{1:N}, \theta)$ is highly multi-modal as the sign of x_n cannot be observed in y_n due to its square.

Two sets of data are generated for testing of the inference algorithms: (1) 100 observations $\{y_n\}_{n=1}^{100}$ with relatively small observation noise $\sigma_w^2 = 1$ and $\sigma_v^2 = 10$; and (2) 100 observations with large observation noise $\sigma_w^2 = \sigma_v^2 = 10$. Two identical diffuse inverse-Gamma priors $\mathcal{IG}(\alpha=0.01, \beta=0.01)$ are adopted for both σ_v^2 and σ_w^2 . The PMMH sampler uses a 2D Gaussian random-walk proposal with diagonal covariance. A good (i.e. close-to-truth) initialisation with $\sigma_v^2(0) = \sigma_w^2(0) = 20$ is provided this time for both the PG and the PMMH algorithms; while the PF-VI algorithm assumes $q(\sigma_v^2) = q(\sigma_w^2) = \mathcal{IG}(0.01, 0.01)$ (i.e. prior) for the first iteration. Running all three inference algorithms for 500 iterations with a bootstrap PF and 2000 particles, **Figure 6.3.8** shows the posterior results for inferred parameters on both datasets. For comparison, I have also included the results obtained with “cheat” inference where the true hidden states $\{x_n\}_{n=0}^N$ are given. For the low σ_w^2 case, all three algorithms have achieved reasonable posterior results for both σ_w^2

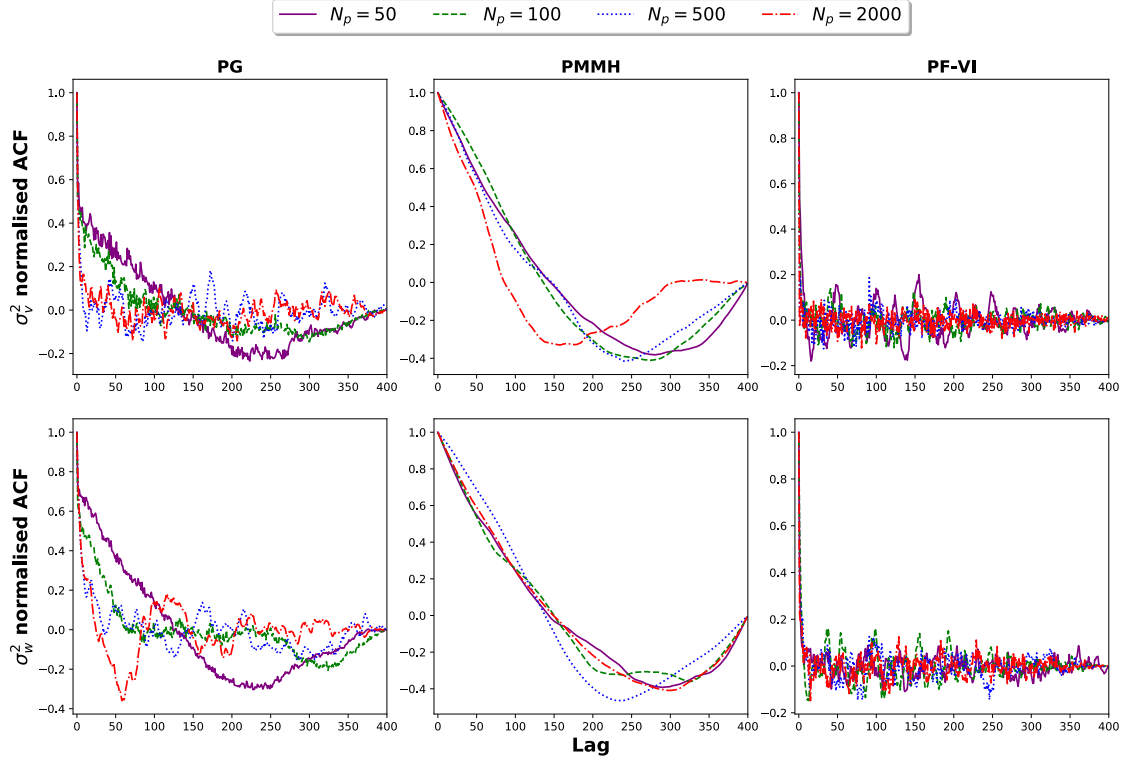


Figure 6.3.12: ACFs of the parameters obtained on the small observation noise with fair initialisation $\sigma_v^2(0) = \sigma_w^2(0) = 100$ and varying particle numbers N_p .

and σ_v^2 . We can also observe a general match in the shapes of the PF-VI posterior and sample histograms of the PMMH and the PG. For the dataset with large observation noise, σ_v^2 is significantly underestimated by all three methods. The PF-VI algorithm gives an accurate posterior estimation on σ_w^2 . Again similarities in results can be observed between the PF-VI and the PG. The PMMH algorithm performs rather poorly in the high σ_w^2 case.

Both the PMMH and the PG algorithms require initialisation of the parameters (σ_v^2, σ_w^2) as a design choice of the inference method. This initialisation can significantly affect the convergence performance of these algorithms. With a fair (i.e. less informative) initialisation $\sigma_v^2(0) = \sigma_w^2(0) = 100$ and true parameter values $\sigma_v^2 = 10$, $\sigma_w^2 = 1$, **Figure 6.3.9** shows the l_2 -distance between the posterior samples and the truth. Note that the posterior samples of the PF-VI are randomly sampled from the approximated (closed-form) posteriors at each iteration. Clearly from the figure, the PMMH algorithm fails to converge within 500 iterations while the PG is slightly affected but still converges quickly at around the 20th iteration. The convergence of the PG is further delayed with a random initialisation from the diffuse prior as indicated in **Figure 6.3.10**. Each iteration of any of the three algorithms requires the PF to run across the entire dataset and demands the same amount of computation. Hence, the PG and the PMMH algorithms would need reasonable initialisation and a good design of proposal densities to match the performance of

the PF-VI algorithm. Whilst the PF-VI algorithm requires no extra design input except for the PF that is shared among all three algorithms.

I present another analysis on the convergence of the three algorithms. **Figure 6.3.11** shows the (normalised) auto-correlation functions (ACFs) for the parameters samples (σ_v^2, σ_w^2) in the low observation noise dataset with good initialisation i.e. $\sigma_v^2(0) = \sigma_w^2(0) = 20$. For this case, all three algorithms appear to mix well as the ACFs decrease relatively quickly to zeros with an increasing lag. The PMMH algorithm performs comparatively worse than the other two in the convergence speed, which agrees with previous results.

Experiments have also been performed with varying numbers of particles N_p in the PF. I use the same low-noise dataset and adopt again the fair initialisation $\sigma_v^2(0) = \sigma_w^2(0) = 100$ to better illustrate the impact of the PF performance on the overall performance of the three algorithms. **Figure 6.3.12** shows the ACFs obtained with different values of N_p using all three inference algorithms. Comparing within one algorithm, it is clear that the increasing particle number improves the inference performance by allowing faster decay and smaller vibrations of the ACF. It has also been found (not shown here) that the improvement generally saturates for N_p greater than 2000 in this example. The PMMH algorithm fails to converge within 500 iterations regardless of the values of N_p . Focusing on the comparison between the PG and the PF-VI algorithm, we see that the ACFs of the PG algorithm are much more sensitive to the change of particle number; while the PF-VI algorithm is only slightly affected. Considering the similarities in algorithm structure between PG and PF-VI, the latter infers the parameters posterior with the full information of all particles obtained at the end of the PF and is thus less influenced by the lack of exploration of the sample space of the hidden states $\{x_n\}_{n=1}^N$ due to a low particle number. On the hand, the PG algorithm uses a single importance-sampled particle trajectory for parameter learning and is more prone to unsatisfactory performance of the PF.

In this experiment, I have demonstrated that the proposed PF-VI algorithm can be widely applied to non-linear SSMs where state vectors cannot be ‘‘Rao-Blackwellised’’. The PF-VI algorithm tallies with the well-established PG algorithm in posterior accuracy and generally outperforms the PMMH algorithm. In addition to the fast convergence, the PF-VI algorithm provides more robust posterior inference compared to the PG and the PMMH algorithms as it requires neither initialisation nor designed proposal densities. The full usage of particle information provides the PF-VI algorithm with extra robustness to the amount of particles used in the PF, which enables the trade-off between inference accuracy and computational cost.

6.4 Conclusions and future work

In this chapter, I have presented three parameter-learning algorithms that can be applied in various SSMs proposed in previous chapters. The MKF algorithm performs posterior inference on linear-Gaussian SSMs by preserving the symbolic representation of the unknown parameters throughout the entire forward Kalman filtering and the backward RTS smoothing processes. Experimental results have shown that the algorithm provides accurate closed-form posterior inference on parameters, as well as robust inference on sequential state vectors.

The marginal RBPF, as a direct extension of the MKF algorithm, is designed for the inference of non-linear SSMs where state vectors can be “Rao-Blackwellised” such as the non-linear jump-diffusion models proposed in **Chapter 3**. By occasionally integrating out the unknown parameters during the weight-evaluation stage of the PF, the marginal RBPF not only allows the posterior to be obtained at the end of the PF but also provides an efficient online parameter-learning framework for applications that require inference based on a continuous stream of data.

Last but not least, I have also proposed a novel PF-VI algorithm which integrates the particle filter with the VI (VB) framework. In addition to the jump-diffusion model, this algorithm can be widely applied to non-linear and/or non-Gaussian models. In contrast to the PG algorithm introduced in [64] and adopted in the regime-switching model in **Chapter 5**, the PF-VI algorithm fully utilises the information of all particles obtained in the PF. Empirical results obtained on synthetic datasets have shown that the PF-VI algorithm achieves more accurate, more efficient (faster convergence) and more robust inference of the parameter posteriors comparing to both the PMMH and the PG algorithms. As an iterative method, the PF-VI algorithm provides accurate closed-form approximations to the model posteriors and serves as a powerful offline inference algorithm. Moreover, the hybrid concept of the PF-VI algorithm has been briefly introduced for the learning of parameters that cannot be inferred in closed-form. This is achieved by optimising the parameters to improve the ELBO at each iteration with gradient ascent approaches.

For future work, one possible extension on the marginal RBPF is to refine the online inference framework. For a continuous stream of time-series data, the (parameter) posterior of the current marginal RBPF algorithm is likely to be “trapped” as the number of observations increases, which prohibits future adaptation of the parameters and leads to model overfitting to old data. Therefore, it is preferable for online applications to have a sliding learning window of the observations or a decaying weighting function on earlier observations.

One concern of the PF-VI algorithm is that it targets the approximated posterior instead of the true posterior. Hence, another promising future research topic is to use

the PF-VI posteriors as the adaptive proposals for the PMMH sampler. This could potentially give not only fast convergence of the algorithm but also convergence towards the true posterior. However, extra care is required to ensure ergodicity of the Markov chains while using such adaptive proposals [167].

Appendix

6.A Expected transition density and likelihood

Denote the term of expectation transition density as an unnormalised density of \mathbf{x}_n :

$$\begin{aligned}\tilde{g}(\mathbf{x}_n) &= \exp\{\langle \log p(\mathbf{x}_n | \mathbf{x}_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)}\} \\ &= \exp\left\{-\frac{1}{2}\left(\log 2\pi + D \langle \log \sigma_v^2 \rangle_{q(\mu_J, \sigma_v^2)}\right) + \log |\hat{\Sigma}_{\tau,n}| + \right. \\ &\quad \left. \langle (\mathbf{x}_n - \tilde{\boldsymbol{\mu}}_n)^T (\sigma_v^2 \hat{\Sigma}_{\tau,n})^{-1} (\mathbf{x}_n - \tilde{\boldsymbol{\mu}}_n) \rangle_{q(\mu_J, \sigma_v^2)}\right\}\end{aligned}$$

We can thus focus solely on the quadratic term as other terms eventually become normalising constant. Denoting $\mathbf{u}_n = \sum_{\tau_k \in \{\tau\}_{n-1:n}} e^{\mathbf{A}(t_n - \tau_k)} \mathbf{c}$, expand the quadratic term as:

$$\begin{aligned}& \langle (\mathbf{x}_n - \tilde{\boldsymbol{\mu}}_n)^T (\sigma_v^2 \hat{\Sigma}_{\tau,n})^{-1} (\mathbf{x}_n - \tilde{\boldsymbol{\mu}}_n) \rangle_{q(\mu_J, \sigma_v^2)} \\ &= \left\langle \left\{ (\mathbf{x}_n - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1})^T \hat{\Sigma}_{\tau,n}^{-1} (\mathbf{x}_n - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1}) \frac{1}{\sigma_v^2} - 2(\mathbf{x}_n - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1})^T \hat{\Sigma}_{\tau,n}^{-1} \mathbf{u}_n \frac{\mu_J}{\sigma_v^2} \right. \right. \\ &\quad \left. \left. + \mathbf{u}_n^T \hat{\Sigma}_{\tau,n}^{-1} \mathbf{u}_n \frac{\mu_J^2}{\sigma_v^2} \right\} \right\rangle_{q(\mu_J, \sigma_v^2)} \\ &= (\mathbf{x}_n - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1})^T \hat{\Sigma}_{\tau,n}^{-1} (\mathbf{x}_n - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1}) \left\langle \frac{1}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} - \\ &\quad 2(\mathbf{x}_n - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1})^T \hat{\Sigma}_{\tau,n}^{-1} \mathbf{u}_n \left\langle \frac{\mu_J}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} + \mathbf{u}_n^T \hat{\Sigma}_{\tau,n}^{-1} \mathbf{u}_n \left\langle \frac{\mu_J^2}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)}\end{aligned}$$

where with (approximated) posterior $q(\mu_J, \sigma_v^2) = \mathcal{NIG}(\mu_J, \sigma_v^2 | \hat{m}, \hat{\nu}, \hat{\alpha}, \hat{\beta})$, we have:

$$\left\langle \frac{1}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} = \frac{\hat{\alpha}}{\hat{\beta}} \quad , \quad \left\langle \frac{\mu_J}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} = \frac{\hat{\alpha}}{\hat{\beta}} \hat{m} \quad , \quad \left\langle \frac{\mu_J^2}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} = \frac{\hat{\alpha}}{\hat{\beta}} \hat{m}^2 + \frac{1}{\hat{\nu}} \quad (6.96)$$

Substituting in the expectations, the unnormalised density $\tilde{g}(\mathbf{x}_n)$ can thus be written as:

$$\begin{aligned}\tilde{g}(\mathbf{x}_n) &= \exp\left\{-\frac{1}{2}\left((\mathbf{x}_n - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1} - \mathbf{u}_n \hat{m})^T \left(\frac{\hat{\beta}}{\hat{\alpha}} \hat{\Sigma}_{\tau,n}\right)^{-1} (\mathbf{x}_n - e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1} - \mathbf{u}_n \hat{m}) + \text{const.}\right)\right\} \\ &= \mathcal{N}\left(\mathbf{x}_n | e^{\mathbf{A}\Delta t_n} \mathbf{x}_{n-1} - \mathbf{u}_n \hat{m}, \frac{\hat{\beta}}{\hat{\alpha}} \hat{\Sigma}_{\tau,n}\right) \times \text{const.}\end{aligned}$$

as required. Samples of \mathbf{x}_n can be drawn readily from the conditional normal density specified above.

Concerning the expected likelihood, we have:

$$\begin{aligned}
& \exp\{\langle \log p(y_n | \mathbf{x}_n, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)}\} \\
&= \exp\left\{-\frac{1}{2}\left\langle \log 2\pi\kappa_w + \log \sigma_v^2 + \frac{(y_n - \mathbf{G}\mathbf{x}_n)^2}{\kappa_w\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)}\right\} \\
&= \exp\left\{-\frac{(y_n - \mathbf{G}\mathbf{x}_n)^2}{2\kappa_w \frac{\hat{\beta}}{\hat{\alpha}}} + \text{const.}\right\} \\
&= \mathcal{N}(y_n | \mathbf{G}\mathbf{x}_n, \kappa_w \frac{\hat{\beta}}{\hat{\alpha}}) \times \text{const.}
\end{aligned}$$

which allows the proposed particles to be re-weighted according to observations.

6.B ELBO evaluation for jump-diffusion model

With the expression for ELBO derived in (6.92):

$$\begin{aligned}
& \mathcal{O}(Y, \{\tau\}_{0:N}, \mu_J, \sigma_v^2) \\
&= \mathbb{E}_{q(\mu_J, \sigma_v^2) q(\{\tau\}_{0:N})} \left\{ \sum_{n=1}^N \log p(y_n | y_{0:n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \right\} - \underbrace{H(q(\mu_J, \sigma_v^2); p(\mu_J, \sigma_v^2))}_{(i)} \\
&+ \underbrace{H(q(\{\tau\}_{0:N}); p(\{\tau\}_{0:N}))}_{(ii)} + \underbrace{H(q(\mu_J, \sigma_v^2))}_{(iii)} + \underbrace{H(q(\{\tau\}_{0:N}))}_{(iv)} + \text{const.}
\end{aligned}$$

Expand and compute each term individually. The first term is essentially the expectation of Kalman PED, which can be expanded as:

$$\begin{aligned}
& \sum_{n=1}^N \langle \log p(y_n | y_{n-1}, \{\tau\}_{n-1:n}, \mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2) q(\{\tau\}_{0:N})} \\
&= \sum_{i=1}^{N_p} w_N^{(i)} \sum_{n=1}^N \left\{ -\frac{1}{2} \left(\log[2\pi(\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)} \mathbf{G}^T + \kappa_w)] + \langle \log \sigma_v^2 \rangle_{q(\mu_J, \sigma_v^2)} \right. \right. \\
&+ \frac{(y_n - \mathbf{G}\boldsymbol{\mu}_{n|0:n-1}^{c(i)})^2}{\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)} \mathbf{G}^T + \kappa_w} \left\langle \frac{1}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} - \frac{2(y_n - \mathbf{G}\boldsymbol{\mu}_{n|0:n-1}^{c(i)}) \mathbf{G}\mathbf{d}_{n|0:n-1}^{(i)}}{\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)} \mathbf{G}^T + \kappa_w} \left\langle \frac{\mu_J}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} \\
&\left. \left. + \frac{(\mathbf{G}\mathbf{d}_{n|0:n-1}^{(i)})^2}{\mathbf{G}\mathbf{L}_{n|0:n-1}^{(i)} \mathbf{G}^T + \kappa_w} \left\langle \frac{\mu_J^2}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} \right) \right\} \quad (6.97)
\end{aligned}$$

In addition to the expectations in Eq. (6.96), we also have:

$$\langle \log \sigma_v^2 \rangle_{q(\mu_J, \sigma_v^2)} = \log(\hat{\beta}) - \psi(\hat{\alpha}) \quad (6.98)$$

where $\psi(\cdot)$ is the digamma function and $\log(\cdot)$ here denotes the natural log by default. This allows us to compute the first term numerically with all known values.

The entropy of a \mathcal{NIG} distribution **(iii)** and the cross entropy between two \mathcal{NIG} distributions **(i)** can be computed in almost the same way. Assuming the prior is $p(\mu_J, \sigma_v^2) = \mathcal{NIG}(\mu_J, \sigma_v^2 | m, \nu, \alpha, \beta)$, the cross entropy term **(i)** is:

$$\begin{aligned} & H(q(\mu_J, \sigma_v^2); p(\mu_J, \sigma_v^2)) \\ &= - \langle \log p(\mu_J, \sigma_v^2) \rangle_{q(\mu_J, \sigma_v^2)} \\ &= - \left\langle \frac{1}{2} \log\left(\frac{\nu}{2\pi}\right) - \frac{1}{2} \log \sigma_v^2 + \alpha \log \beta - \log[\Gamma(\alpha)] - (\alpha + 1) \log \sigma_v^2 - \frac{\beta}{\sigma_v^2} - \frac{\nu(\mu_J - m)^2}{2\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} \\ &= \frac{1}{2} \log\left(\frac{2\pi}{\nu}\right) + \left(\alpha + \frac{3}{2}\right) \langle \log \sigma_v^2 \rangle_{q(\mu_J, \sigma_v^2)} - \alpha \log \beta + \log[\Gamma(\alpha)] + \beta \left\langle \frac{1}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} + \frac{\nu}{2} \left\langle \frac{\mu_J^2}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} \\ &\quad - \nu m \left\langle \frac{\mu_J}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} + \frac{\nu m^2}{2} \left\langle \frac{1}{\sigma_v^2} \right\rangle_{q(\mu_J, \sigma_v^2)} \\ &= \frac{1}{2} \log\left(\frac{2\pi}{\nu}\right) + \left(\alpha + \frac{3}{2}\right) [\log(\hat{\beta}) - \psi(\hat{\alpha})] - \alpha \log \beta + \log[\Gamma(\alpha)] + \frac{\nu}{2} \left[\frac{\hat{\alpha}}{\hat{\beta}} \hat{m}^2 + \frac{1}{\hat{\nu}} \right] \\ &\quad - \nu m \left(\frac{\hat{\alpha}}{\hat{\beta}} \hat{m} \right) + \frac{\nu m^2 + 2\beta}{2} \left(\frac{\hat{\alpha}}{\hat{\beta}} \right) \end{aligned} \quad (6.99)$$

which again allows tractable evaluation. Replacing the prior $p(\mu_J, \sigma_v^2)$ in the cross entropy expression above with the approximated posterior $q(\mu_J, \sigma_v^2)$, the entropy can be obtained as:

$$H(q(\mu_J, \sigma_v^2)) = \frac{1}{2} \log\left(\frac{2\pi e}{\hat{\nu}}\right) + \frac{3}{2} \log(\hat{\beta}) + \log[\Gamma(\hat{\alpha})] - \left(\hat{\alpha} + \frac{3}{2}\right) \psi(\hat{\alpha}) + \hat{\alpha} \quad (6.100)$$

Both terms **(ii)** and **(iv)** involves an empirical distribution $q(\{\tau\}_{0:N})$ as a weighted particle collection. These can be computed from the generic definitions of entropy and cross entropy for discrete variables. However in particle filter, it is possible to have duplicates of the same particle (i.e. information) which need to be merged before any calculation. After merging, the cross entropy term **(ii)** can be expressed as:

$$\begin{aligned} H(q(\{\tau\}_{0:N}); p(\{\tau\}_{0:N})) &= \mathbb{E} \left[-\log p(\{\tau\}_{0:N}) \right] \Big|_{q(\{\tau\}_{0:N})} \\ &= - \sum_i^{N_u} \hat{w}_N^{(i)} \log p(\{\tau\}_{0:N}^{(i)}) \end{aligned} \quad (6.101)$$

where $p(\{\tau\}_{0:N}^{(i)})$ is the readily available prior probability under the homogeneous Poisson process assumption and the entropy term **(iv)** is:

$$H(q(\{\tau\}_{0:N})) = - \sum_i^{N_u} \hat{w}_N^{(i)} \log(\hat{w}_N^{(i)}) \quad (6.102)$$

where N_u ($\leq N_p$) is the number of unique particles after merging and $\hat{w}_N^{(i)}$ is the merged particle weight for the unique particle i . The merged weight is calculated as follows:

$$\hat{w}_N^{(i)} = \sum_j^{N_p} w_N^{(j)} \delta_{\{\tau\}_{0:N}^{(i)}}(\{\tau\}_{0:N}^{(j)}) \quad (6.103)$$

where $\delta_i(j)$ takes value 1 if $j = i$ and 0 otherwise.

Chapter 7

Summary

This thesis has presented five research projects, each of which tackles the modelling and the inference of the LOBs from a different perspective. The proposed models, inference algorithms and techniques have demonstrated good performance while being generalisable to other applications and transferable to other areas and disciplines of research.

In **Chapter 2**, I introduce a novel approach of modelling the intensity function of the NHPP. By modelling the NHPP intensity function with a transformed state-space stochastic diffusion, the approach not only mitigates the inherent intractability of the NHPP likelihood but also allows the sequential framework of the SSM to be fully utilised for possible online Bayesian inference. In addition to the generic SMC MC algorithm used for sequential inference, the MwG refinement scheme and the sequential batch scheme are proposed to further to improve the inference accuracy and control the computational cost. The model provides a plausible approach of transforming the noisy high-frequency ticks data of order operations into informative intensities that can be used for analysis and prediction of the LOB markets.

In **Chapter 3**, I study the LOBs by modelling the market's fair price process. Inspired by real-world market intuitions, the baseline jump-diffusion SSM in [13] is extended with additional model features. The volumetric model considers the contribution of dynamic LOB volume imbalance to the innovation of price trend; while the extended trend model introduces an additional state variable to capture trends at different time scales. The trend resetting mechanism provides an alternative insight to the jumps occurring in the high-frequency LOB market. The proposed features maintain the conditional tractability which allows the models to be efficiently inferred with the RBPF algorithm. Empirical results obtained on real FOREX data demonstrate that the proposed features help improve mean prediction accuracy and model fit. The retrospective analyses on the full model's posteriors also provide insightful interpretations for various market behaviours.

The research on the modelling of LOB price dynamics is continued in **Chapter 4**. By considering the jumps of different types (e.g. induced by different order operations) as the main innovation for price trends, I propose a continuous-time SSM which integrates a

multi-jump reversion process and a “varying-mean” OU process to account for the characteristic trends and the constant volatility in the price dynamics respectively. In order to handle the increased dimension of the latent variables, a semi-deterministic particle filtering algorithm is proposed, which provides more thorough exploration over the latent sample space. Experimental results obtained on simulated data show that the proposed semi-deterministic filtering scheme outperforms the standard bootstrap RBPF algorithm. The posteriors of the model obtained on FOREX price data demonstrate reasonable disaggregation of the price trend, which allows trend-following trading strategies to be executed at different scales and time horizons. Moreover, I demonstrate the correlation between the inferred jump times and the large-volume submissions of limit orders. This not only supports our intuition that impulsive jumps can be one of the main innovations to price trends, but also establishes a link between the proposed model and the NHPP intensity model introduced in **Chapter 2**.

In order to achieve more realistic modelling of the price dynamics in the high-frequency LOB market, in **Chapter 5** I study the multi-regime diffusion that commonly exists in complex dynamical systems including financial markets and introduces the state-space regime-switching model with infinite mixture dynamics. In addition to the semi-Markovian regime-switching feature, the proposed model takes a step further by allowing the number of regimes and the regime parameters to be unknown and modelled by the DPM. Under the general framework of the PMCMC algorithm, I adopt the deterministic particle filtering, the stratified resampling and the blocked-MwG sampling to achieve Bayesian inference for both sequential states and SSM parameters. Experimenting the model on different datasets, I show that the proposed model performs accurate regime identification as well as posterior learning of the regime parameters.

With various sequential models proposed in the earlier chapters, it is important to have an efficient and robust approach of setting the hyperparameters of these models. Thus in **Chapter 6**, I focus on the development of parameter learning algorithms that can be readily applied to the SSMs. Two online parameter-learning algorithms are proposed: the marginal Kalman filter and the marginal RBPF. The marginal Kalman filter provides efficient closed-form parameter learning for the linear-Gaussian SSMs; while the marginal RBPF allows certain non-linearity in the dynamic model and achieves “quasi-closed-form” inference of the parameter posteriors by marginalising (i.e. summing) the particles. Moreover, a novel offline inference algorithm is proposed by combining the particle filter with the variational Bayes (inference) algorithm. By using all particle information and an initialisation-free setup, the PF-VI algorithm achieves faster convergence and more robust inference than the competing PMCMC algorithms. I also introduce a hybrid PF-VI algorithm which handles the intractability of certain parameters in the SSM and allows an even broader class of the SSMs to be accommodated.

The study of the LOB in high-frequency finance is an area of great research potential with objectives ranging from deriving profitable trading strategies to market structure analysis to anomaly detection and prediction of market crises (e.g. flash crash). The topics, models and algorithms covered in this thesis constitute a small part of the grand atlas of various researches on high-frequency finance. However, the academic research in this area also faces many practical challenges. Both proponents and opponents of the efficient market hypothesis (EMH) agree on the idea that any market participants that deviate from the passive market portfolio or trade at an “unfair” market price are playing a zero-sum game. This highly competitive environment leads to the discreteness in the sharing of research resources and technical advancements especially in the financial industry. For a similar reason, researches in academia are often short of high-quality data and testbeds in spite of the enormous amount of trading happening every single day. This separation of research community between the academia and the industry (and companies in the industry) may have severely limited the development of novel models and algorithms for the study of the LOB in high-frequency finance.

This thesis reports the research achievements throughout the course of my PhD study. I believe the models and algorithms proposed here are able to contribute towards the advancement of the research on high-frequency LOB markets and I hope that the insights and directions for future work provided in this thesis can also shed some light onto even more areas and disciplines of research.

Bibliography

- [1] A. Bigiotti and A. Navarra. “Optimizing Automated Trading Systems”. In: *Proceedings of the 2018 International Conference on Digital Science*. Springer, 2018, pp. 254–261.
- [2] *Algorithmic Trading Market Report*. <https://www.marketsandmarkets.com/Market-Reports/algorithmic-trading-market-179361860.html>.
- [3] Arash M. *Knight Capital glitch loss hits \$461m*. URL: <https://www.ft.com/content/928a1528-1859-11e2-80e9-00144feabdc0>.
- [4] Á. Cartea, S. Jaimungal, and J. Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.
- [5] F. Abergel, M. Anane, A. Chakraborti, A. Jedidi, and I. Toke. *Limit order books*. Cambridge University Press, 2016.
- [6] Investopedia. *Guide to Trade Order Types*. URL: <https://www.investopedia.com/terms/o/order.asp>.
- [7] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [8] R. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. URL: <https://doi.org/10.1115/1.3662552>.
- [9] B. Friedland. *Control system design: an introduction to state-space methods*. Courier Corporation, 2012.
- [10] J. Vermaak, S. Godsill, and P. Perez. “Monte Carlo filtering for multi target tracking and data association”. In: *IEEE Transactions on Aerospace and Electronic systems* 41.1 (2005), pp. 309–332.
- [11] R. Roesser. “A discrete state-space model for linear image processing”. In: *IEEE Transactions on Automatic Control* 20.1 (1975), pp. 1–10.
- [12] J. Stock and M. Watson. “Dynamic factor models, factor-augmented vector autoregressions, and structural vector autoregressions in macroeconomics”. In: *Handbook of macroeconomics*. Vol. 2. Elsevier, 2016, pp. 415–525.

- [13] H. Christensen, J. Murphy, and S. Godsill. “Forecasting high-frequency futures returns using online Langevin dynamics”. In: *IEEE Journal of Selected Topics in Signal Processing* 6.4 (2012), pp. 366–380.
- [14] L. Paninski, Y. Ahmadian, D. Ferreira, S. Koyama, K. Rad, M. Vidne, J. Vogelstein, and W. Wu. “A new look at state-space models for neural data”. In: *Journal of computational neuroscience* 29.1-2 (2010), pp. 107–126.
- [15] Z. Ghahramani and G. Hinton. “Variational learning for switching state-space models”. In: *Neural computation* 12.4 (2000), pp. 831–864.
- [16] J. Liu and R. Chen. “Sequential Monte Carlo methods for dynamic systems”. In: *Journal of the American statistical association* 93.443 (1998), pp. 1032–1044.
- [17] N. Gordon, D. Salmond, and A. Smith. “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”. In: *IEE proceedings F (radar and signal processing)*. Vol. 140. 2. IET. 1993, pp. 107–113.
- [18] O. Cappé, S. Godsill, and E. Moulines. “An overview of existing methods and recent advances in sequential Monte Carlo”. In: *Proceedings of the IEEE* 95.5 (2007), pp. 899–924.
- [19] R. Douc and O. Cappé. “Comparison of resampling schemes for particle filtering”. In: *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005*. IEEE. 2005, pp. 64–69.
- [20] S. Godsill. “Particle filtering: the first 25 years and beyond”. In: *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7760–7764.
- [21] P. Fearnhead and P. Clifford. “On-line inference for hidden Markov models via particle filters”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 65.4 (2003), pp. 887–899.
- [22] C. Li and S. Godsill. “Sequential inference methods for non-homogeneous Poisson processes with state-space prior”. In: *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. 2018. 2018, pp. 2856–2860.
- [23] C. Li and S. Godsill. “Sequential inference methods for non-homogeneous Poisson processes with state-space prior”. In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 1154–1168.
- [24] D. Perkel, G. Gerstein, and G. Moore. “Neuronal spike trains and stochastic point processes: II. Simultaneous spike trains”. In: *Biophysical journal* 7.4 (1967), pp. 419–440.

- [25] J. Gardner and L. Knopoff. “Is the sequence of earthquakes in Southern California, with aftershocks removed, Poissonian?” In: *Bulletin of the Seismological Society of America* 64.5 (1974), pp. 1363–1367.
- [26] V. Iversen. “Teletraffic engineering handbook”. In: *ITU-D SG 2* (2005), p. 16.
- [27] S. Godsill and J. Vermaak. “Variable rate particle filters for tracking applications”. In: *IEEE/SP Proceedings of the 13th Workshop on Statistical Signal Processing, 2005*. IEEE. 2005, pp. 1280–1285.
- [28] R. Mahler. “Multitarget Bayes filtering via first-order multitarget moments”. In: *IEEE Transactions on Aerospace and Electronic systems* 39.4 (2003), pp. 1152–1178.
- [29] D. Cox and V. Isham. *Point processes*. Vol. 12. CRC Press, 1980.
- [30] P. Lewis and G. Shedler. “Simulation of nonhomogeneous Poisson processes by thinning”. In: *Naval Research Logistics (NRL)* 26.3 (1979), pp. 403–413.
- [31] S. Chiu, D. Stoyan, W. Kendall, and J. Mecke. *Stochastic geometry and its applications*. John Wiley & Sons, 2013.
- [32] P. Diggle. “A kernel method for smoothing point process data”. In: *Applied statistics* (1985), pp. 138–147.
- [33] W. Massey, G. Parker, and W. Whitt. “Estimating the parameters of a nonhomogeneous Poisson process with linear rate”. In: *Telecommunication Systems* 5.2 (1996), pp. 361–388.
- [34] D. Nicol and L. Leemis. “A continuous piecewise-linear NHPP intensity function estimator”. In: *Proceedings of the Winter Simulation Conference 2014*. IEEE. 2014, pp. 498–509.
- [35] Z. Zheng and P. Glynn. “Fitting continuous piecewise linear poisson intensities via maximum likelihood and least squares”. In: *Proceedings of the 2017 Winter Simulation Conference (WSC)*. IEEE. 2017, pp. 1740–1749.
- [36] R. Streit and L. Stone. “Bayes derivation of multitarget intensity filters”. In: *Proceedings of the 2008 11th International Conference on Information Fusion*. IEEE. 2008, pp. 1–8.
- [37] R. Streit. “Multisensor multitarget intensity filter”. In: *Proceedings of the 2008 11th International Conference on Information Fusion*. IEEE. 2008, pp. 1–8.
- [38] R. Streit. “JPDA intensity filter for tracking multiple extended objects in clutter”. In: *Proceedings of the 2016 19th International Conference on Information Fusion*. IEEE. 2016, pp. 1477–1484.
- [39] A. Raftery and V. Akman. “Bayesian analysis of a Poisson process with a change-point”. In: *Biometrika* (1986), pp. 85–89.

- [40] T. Herberts and U. Jensen. “Optimal detection of a change point in a Poisson process for different observation schemes”. In: *Scandinavian Journal of Statistics* 31.3 (2004), pp. 347–366.
- [41] M. Brown. “Bayesian detection of changes of a Poisson process monitored at discrete time points where the arrival rates are unknown”. In: *Sequential Analysis* 27.1 (2008), pp. 68–77.
- [42] X. Zhao and P. Chu. “Bayesian multiple changepoint analysis of hurricane activity in the eastern North Pacific: A Markov chain Monte Carlo approach”. In: *Journal of climate* 19.4 (2006), pp. 564–578.
- [43] S. Gugushvili, F. van der Meulen, M. Schauer, and P. Spreij. “Fast and scalable non-parametric Bayesian inference for Poisson point processes”. In: *arXiv preprint arXiv:1804.03616* (2018).
- [44] C. Rasmussen and C. Williams. *Gaussian processes for machine learning*. Vol. 1. MIT press Cambridge, 2006.
- [45] S. Rathbun and N. Cressie. “Asymptotic properties of estimators for the parameters of spatial inhomogeneous Poisson point processes”. In: *Advances in Applied Probability* (1994), pp. 122–154.
- [46] J. Møller, A. Syversveen, and R. Waagepetersen. “Log Gaussian Cox processes”. In: *Scandinavian journal of statistics* 25.3 (1998), pp. 451–482.
- [47] R. Adams, I. Murray, and D. MacKay. “Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, pp. 9–16.
- [48] T. Gunter, C. Lloyd, M. Osborne, and S. Roberts. “Efficient Bayesian Nonparametric Modelling of Structured Point Processes”. In: *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI’14)*. 2014, pp. 310–319.
- [49] C. Lloyd, T. Gunter, M. Osborne, and S. Roberts. “Variational inference for Gaussian process modulated Poisson processes”. In: *Proceedings of the International Conference on Machine Learning*. 2015, pp. 1814–1822.
- [50] S. Särkkä. *Recursive Bayesian inference on stochastic differential equations*. Helsinki University of Technology, 2006.
- [51] S. Godsill. “Particle filters for continuous-time jump models in tracking applications”. In: *ESAIM: Proceedings*. Vol. 19. EDP Sciences. 2007, pp. 39–52.
- [52] C. Berzuini and W. Gilks. “RESAMPLE-MOVE filtering with cross-model jumps”. In: *Sequential Monte Carlo Methods in Practice*. Springer, 2001, pp. 117–138.
- [53] A. Golightly and D. Wilkinson. “Bayesian sequential inference for nonlinear multivariate diffusions”. In: *Statistics and Computing* 16.4 (2006), pp. 323–338.

- [54] S. K. Pang, S. Godsill, J. Li, F. Septier, and S. Hill. “Sequential inference for dynamically evolving groups of objects”. English. In: *Bayesian Time Series Models*. 2011. Chap. 12, pp. 245–276. URL: www.scopus.com.
- [55] F. Septier and G. Peters. “Langevin and Hamiltonian based sequential MCMC for efficient Bayesian filtering in high-dimensional spaces”. In: *IEEE Journal of Selected Topics in Signal Processing* 10.2 (2016), pp. 312–327.
- [56] A. Finke, A. Doucet, and A. Johansen. “Limit theorems for sequential MCMC methods”. In: *Advances in Applied Probability* 52.2 (2020), pp. 377–403.
- [57] P. Green. “Reversible jump Markov chain Monte Carlo computation and Bayesian model determination”. In: *Biometrika* 82.4 (1995), pp. 711–732.
- [58] M. Girolami and B. Calderhead. “Riemann manifold Langevin and Hamiltonian Monte Carlo methods”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214.
- [59] D. Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.
- [60] G. Kitagawa. “Monte Carlo filter and smoother for non-Gaussian nonlinear state space models”. In: *Journal of computational and graphical statistics* 5.1 (1996), pp. 1–25.
- [61] M. Gould, M. Porter, S. Williams, M. McDonald, D. Fenn, and S. Howison. “Limit order books”. In: *Quantitative Finance* 13.11 (2013), pp. 1709–1742.
- [62] L. Menkhoff, L. Sarno, M. Schmeling, and A. Schrimpf. “Currency momentum strategies”. In: *Journal of Financial Economics* 106.3 (2012), pp. 660–684.
- [63] A. Sokal. “Monte Carlo methods in statistical mechanics: foundations and new algorithms”. In: *Functional integration*. Springer, 1997, pp. 131–192.
- [64] C. Andrieu, A. Doucet, and R. Holenstein. “Particle Markov chain Monte Carlo methods”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3 (2010), pp. 269–342.
- [65] W. Gilks and P. Wild. “Adaptive rejection sampling for Gibbs sampling”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 41.2 (1992), pp. 337–348.
- [66] D. Görür and Y. W. Teh. “Concave-convex adaptive rejection sampling”. In: *Journal of Computational and Graphical Statistics* 20.3 (2011), pp. 670–691.
- [67] S. K. Pang, J. Li, and S. Godsill. “Detection and tracking of coordinated groups”. In: *IEEE Transactions on Aerospace and Electronic Systems* 47.1 (2011), pp. 472–502.

- [68] H. Luckock. “A steady-state model of the continuous double auction”. In: *Quantitative Finance* 3.5 (2003), pp. 385–404.
- [69] P. Wang, L. Li, and S. Godsill. “Particle filtering and inference for limit order books in high frequency finance”. In: *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4264–4268.
- [70] R. Cont, S. Stoikov, and R. Talreja. “A stochastic model for order book dynamics”. In: *Operations research* 58.3 (2010), pp. 549–563.
- [71] E. Panayi and G. Peters. “Stochastic simulation framework for the limit order book using liquidity-motivated agents”. In: *International Journal of Financial Engineering* 2.02 (2015), p. 1550013.
- [72] H. Christensen, R. Turner, S. Hill, and S. Godsill. “Rebuilding the limit order book: sequential Bayesian inference on hidden states”. In: *Quantitative Finance* 13.11 (2013), pp. 1779–1799.
- [73] M. Avellaneda and S. Stoikov. “High-frequency trading in a limit order book”. In: *Quantitative Finance* 8.3 (2008), pp. 217–224.
- [74] R. Cont. “Statistical modeling of high-frequency financial data”. In: *IEEE Signal Processing Magazine* 28.5 (2011), pp. 16–25.
- [75] W. Bao, J. Yue, and Y. Rao. “A deep learning framework for financial time series using stacked autoencoders and long-short term memory”. In: *PLoS ONE* 12.7 (2017). ISSN: 19326203. DOI: 10.1371/journal.pone.0180944.
- [76] N. Gordon, D. Salmond, and A. Smith. “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”. In: *IEE Proceedings F Radar and Signal Processing* 140.2 (1993), pp. 107–113. ISSN: 0956375X. DOI: 10.1049/ip-f-2.1993.0015. arXiv: 9241F(E5).
- [77] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”. In: *IEEE Transactions on signal processing* 50.2 (2002), pp. 174–188.
- [78] J. Murphy. “Bayesian methods for high frequency financial time series analysis”. In: *PhD First Year Report, Cambridge University Department of Engineering* (2010).
- [79] H. Lopes and R. Tsay. “Particle filters and Bayesian inference in financial econometrics”. In: *Journal of Forecasting* 30.1 (2011), pp. 168–209.
- [80] R. Carmona, P. Del Moral, P. Hu, and N. Oudjane. “An introduction to particle methods with financial applications”. In: *Numerical Methods in Finance*. Springer, 2012, pp. 3–49.

- [81] C.R. Rao. “Information and accuracy attainable in the estimation of statistical parameters”. In: *Bulletin of the Calcutta Mathematical Society* 37 (1945), pp. 81–91.
- [82] D. Blackwell. “Conditional Expectation and Unbiased Sequential Estimation”. In: *The Annals of Mathematical Statistics* 18.1 (Mar. 1947), pp. 105–110. ISSN: 0003-4851. DOI: 10.1214/aoms/1177730497.
- [83] A. Doucet, S. Godsill, and C. Andrieu. “On sequential Monte Carlo sampling methods for Bayesian filtering”. In: *Statistics and computing* 10.3 (2000), pp. 197–208.
- [84] R. Cont and A. De Larrard. “Price dynamics in a Markovian limit order market”. In: *SIAM Journal on Financial Mathematics* 4.1 (2013), pp. 1–25.
- [85] R. Schöbel and J. Zhu. “Stochastic volatility with an Ornstein–Uhlenbeck process: an extension”. In: *Review of Finance* 3.1 (1999), pp. 23–46.
- [86] F. Daum and J. Huang. “Curse of dimensionality and particle filters”. In: *Proceedings of the 2003 IEEE Aerospace Conference (Cat. No. 03TH8652)*. Vol. 4. IEEE. 2003, 4_1979–4_1993.
- [87] S. Godsill, J. Vermaak, W. Ng, and J. Li. “Models and algorithms for tracking of maneuvering objects using variable rate particle filters”. In: *Proceedings of the IEEE* 95.5 (2007), pp. 925–952.
- [88] A. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.
- [89] A. Doucet and A. Johansen. “A tutorial on particle filtering and smoothing: Fifteen years later”. In: *Handbook of nonlinear filtering* 12.656-704 (2009), p. 3.
- [90] R. Merton. “Option pricing when underlying stock returns are discontinuous”. In: *Journal of financial economics* 3.1-2 (1976), pp. 125–144.
- [91] T. Andersen, L. Benzoni, and J. Lund. “Estimating jump-diffusions for equity returns”. In: *Journal of Finance* 57.3 (2002), pp. 1239–1284.
- [92] S. Kou. “A jump-diffusion model for option pricing”. In: *Management science* 48.8 (2002), pp. 1086–1101.
- [93] Z. Huang and S. Kou. “First passage times and analytical solutions for options on two assets with jump risk”. In: *Preprint, Columbia University* (2006).
- [94] D. Duffie, J. Pan, and K. Singleton. “Transform analysis and asset pricing for affine jump-diffusions”. In: *Econometrica* 68.6 (2000), pp. 1343–1376.
- [95] S. Kou. “Jump-diffusion models for asset pricing in financial engineering”. In: *Handbooks in operations research and management science* 15 (2007), pp. 73–116.

- [96] A. Golightly. “Bayesian filtering for jump-diffusions with application to stochastic volatility”. In: *Journal of Computational and Graphical Statistics* 18.2 (2009), pp. 384–400.
- [97] J. Hol, T. Schon, and F. Gustafsson. “On resampling algorithms for particle filters”. In: *Proceedings of the 2006 IEEE nonlinear statistical signal processing workshop*. IEEE. 2006, pp. 79–82.
- [98] H. Blom and Y. Bar-Shalom. “The interacting multiple model algorithm for systems with Markovian switching coefficients”. In: *IEEE transactions on Automatic Control* 33.8 (1988), pp. 780–783.
- [99] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. “Interacting multiple model methods in target tracking: a survey”. In: *IEEE Transactions on aerospace and electronic systems* 34.1 (1998), pp. 103–123.
- [100] Y. Boers and J. Driessen. “Interacting multiple model particle filter”. In: *IEEE Proceedings-Radar, Sonar and Navigation* 150.5 (2003), pp. 344–349.
- [101] C. Nemeth, P. Fearnhead, and L. Mihaylova. “Sequential Monte Carlo methods for state and parameter estimation in abruptly changing environments”. In: *IEEE Transactions on Signal Processing* 62.5 (2013), pp. 1245–1255.
- [102] C. Nemeth, P. Fearnhead, L. Mihaylova, and D. Vorley. “Bearings-only tracking with particle filtering for joint parameter learning and state estimation”. In: *Proceedings of the 2012 15th International Conference on Information Fusion*. IEEE. 2012, pp. 824–831.
- [103] Christopher Nemeth, Paul Fearnhead, Lyudmila Mihaylova, and Dave Vorley. “Particle learning methods for state and parameter estimation”. In: *Proceedings of the 9th IET Data Fusion Target Tracking Conference (DF TT 2012): Algorithms Applications*. 2012, pp. 1–6. DOI: 10.1049/cp.2012.0412.
- [104] P. Fearnhead and Z. Liu. “On-line inference for multiple changepoint problems”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.4 (2007), pp. 589–605.
- [105] S. Yildirim, S. Singh, and A. Doucet. “An online Expectation–Maximization algorithm for changepoint models”. In: *Journal of Computational and Graphical Statistics* 22.4 (2013), pp. 906–926.
- [106] C. Carvalho, M. Johannes, H. Lopes, and N. Polson. “Particle learning and smoothing”. In: *Statistical Science* 25.1 (2010), pp. 88–106.

- [107] D. Bailey, J. Borwein, M. Lopez de Prado, and Q. Zhu. “Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance”. In: *Notices of the American Mathematical Society* 61.5 (2014), pp. 458–471.
- [108] M. Beal, Z. Ghahramani, and C. Rasmussen. “The infinite hidden Markov model”. In: *Proceedings of the Advances in Neural Information Processing Systems*. 2002, pp. 577–584.
- [109] Y. Teh, M. Jordan, M. Beal, and D. Blei. “Hierarchical Dirichlet processes”. In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1566–1581.
- [110] J. Kivinen, E. Sudderth, and M. Jordan. “Learning multiscale representations of natural scenes using Dirichlet processes”. In: *Proceedings of the 2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.
- [111] M. Hoffman, P. Cook, and D. Blei. “Data-driven recomposition using the hierarchical Dirichlet process hidden Markov model”. In: *Proceedings of ICMC*. Citeseer. 2008.
- [112] E. Fox, E. Sudderth, M. Jordan, and A. Willsky. “A sticky HDP-HMM with application to speaker diarization”. In: *The Annals of Applied Statistics* (2011), pp. 1020–1056.
- [113] M. Johnson and A. Willsky. “Bayesian nonparametric hidden semi-Markov models”. In: *Journal of Machine Learning Research* 14.Feb (2013), pp. 673–701.
- [114] F. Caron, M. Davy, A. Doucet, E. Duflos, and P. Vanheeghe. “Bayesian inference for linear dynamic models with Dirichlet process mixtures”. In: *IEEE Transactions on Signal Processing* 56.1 (2007), pp. 71–84.
- [115] E. Fox, E. Sudderth, M. Jordan, and A. Willsky. “Bayesian nonparametric methods for learning Markov switching processes”. In: *IEEE Signal Processing Magazine* 27.6 (2010), pp. 43–54.
- [116] E. Fox, E. Sudderth, and A. Willsky. “Hierarchical Dirichlet processes for tracking maneuvering targets”. In: *Proceedings of the 2007 10th international conference on information fusion*. IEEE. 2007, pp. 1–8.
- [117] K. Murphy. “Hidden semi-Markov models (HSMMs)”. In: *unpublished notes* (2002).
- [118] D. Aldous. “Exchangeability and related topics”. In: *École d’Été de Probabilités de Saint-Flour XIII—1983*. Springer, 1985, pp. 1–198.
- [119] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.
- [120] R. Adams and D. MacKay. “Bayesian online changepoint detection”. In: *arXiv preprint arXiv:0710.3742* (2007).

- [121] M. Pitt, R. dos Santos Silva, P. Giordani, and R. Kohn. “On some properties of Markov chain Monte Carlo simulation methods based on the particle filter”. In: *Journal of Econometrics* 171.2 (2012), pp. 134–151.
- [122] A. Golightly and D. Wilkinson. “Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo”. In: *Interface focus* 1.6 (2011), pp. 807–820.
- [123] D. Rasmussen, O. Ratmann, and K. Koelle. “Inference for nonlinear epidemiological models using genealogies and time series”. In: *PLoS computational biology* 7.8 (2011).
- [124] N. Chopin and S. Singh. “On particle Gibbs sampling”. In: *Bernoulli* 21.3 (2015), pp. 1855–1883.
- [125] F. Lindsten, M. Jordan, and T. Schön. “Particle Gibbs with ancestor sampling”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 2145–2184.
- [126] J. Carpenter, P. Clifford, and P. Fearnhead. “Improved particle filter for nonlinear problems”. In: *IEE Proceedings-Radar, Sonar and Navigation* 146.1 (1999), pp. 2–7.
- [127] T. Fossen. “A nonlinear unified state-space model for ship maneuvering and control in a seaway”. In: *International Journal of Bifurcation and Chaos* 15.09 (2005), pp. 2717–2746.
- [128] T. Patterson, L. Thomas, C. Wilcox, O. Ovaskainen, and J. Matthiopoulos. “State-space models of individual animal movement”. In: *Trends in ecology & evolution* 23.2 (2008), pp. 87–94.
- [129] A. Paul and E. Wan. “RSSI-based indoor localization and tracking using sigma-point Kalman smoothers”. In: *IEEE Journal of selected topics in signal processing* 3.5 (2009), pp. 860–873.
- [130] A. Strandburg-Peshkin, D. Farine, I. Couzin, and M. Crofoot. “Shared decision-making drives collective movement in wild baboons”. In: *Science* 348.6241 (2015), pp. 1358–1361.
- [131] M. Crofoot, R. Kays, and M. Wikelski. *Data from: Shared decision-making drives collective movement in wild baboons*. 2015. DOI: doi:10.5441/001/1.kn0816jn. URL: <http://dx.doi.org/10.5441/001/1.kn0816jn>.
- [132] H. Rauch, F. Tung, and C. Striebel. “Maximum likelihood estimates of linear dynamic systems”. In: *AIAA journal* 3.8 (1965), pp. 1445–1450.
- [133] C. Masreliez and R. Martin. “Robust Bayesian estimation for the linear model and robustifying the Kalman filter”. In: *IEEE transactions on Automatic Control* 22.3 (1977), pp. 361–371.

- [134] M. Roth, E. Özkan, and F. Gustafsson. “A Student’s t filter for heavy tailed process and measurement noise”. In: *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 5770–5774.
- [135] K. Murphy. “Conjugate Bayesian analysis of the Gaussian distribution”. In: *Technical Report* (2007).
- [136] G. Einicke and L. White. “Robust extended Kalman filtering”. In: *IEEE Transactions on Signal Processing* 47.9 (1999), pp. 2596–2599.
- [137] N. Kantas, A. Doucet, S. Singh, J. Maciejowski, N. Chopin, et al. “On particle methods for parameter estimation in state-space models”. In: *Statistical science* 30.3 (2015), pp. 328–351.
- [138] M. Hürzeler and H. Künsch. “Approximating and maximising the likelihood for a general state-space model”. In: *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 159–175.
- [139] S. Malik and M. Pitt. “Particle filters for continuous likelihood evaluation and maximisation”. In: *Journal of Econometrics* 165.2 (2011), pp. 190–209.
- [140] F. LeGland and L. Mevel. “Recursive estimation in hidden Markov models”. In: *Proceedings of the 36th IEEE Conference on Decision and Control*. Vol. 4. IEEE. 1997, pp. 3468–3473.
- [141] P. Fearnhead, D. Wyncoll, and J. Tawn. “A sequential smoothing algorithm with linear computational cost”. In: *Biometrika* 97.2 (2010), pp. 447–464.
- [142] G. Storvik. “Particle filters for state-space models with the presence of unknown static parameters”. In: *IEEE Transactions on signal Processing* 50.2 (2002), pp. 281–289.
- [143] J. Liu and M. West. “Combined parameter and state estimation in simulation-based filtering”. In: *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 197–223.
- [144] M. Pitt and N. Shephard. “Filtering via simulation: Auxiliary particle filters”. In: *Journal of the American statistical association* 94.446 (1999), pp. 590–599.
- [145] J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West. “Particle learning for sequential Bayesian computation”. In: *Bayesian Statistics 9* 9 (2011), p. 317.
- [146] A. Virbickaitė, H. Lopes, C. Ausin, and P. Galeano. “Particle learning for Bayesian semi-parametric stochastic volatility model”. In: *Econometric Reviews* (2019).
- [147] G. Kitagawa. “A self-organizing state-space model”. In: *Journal of the American Statistical Association* (1998), pp. 1203–1215.

- [148] N. Chopin, P. Jacob, and O. Papaspiliopoulos. “SMC2: an efficient algorithm for sequential analysis of state space models”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75.3 (2013), pp. 397–426.
- [149] A. Fulop and J. Li. “Efficient learning via simulation: A marginalized resample-move approach”. In: *Journal of Econometrics* 176.2 (2013), pp. 146–161.
- [150] S. Godsill, M. Riabiz, and I. Kontoyiannis. “The Lévy State Space Model”. In: *Proceedings of the 2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE. 2019, pp. 487–494.
- [151] H. Attias. “A variational bayesian framework for graphical models”. In: *Proceedings of the Advances in Neural Information Processing Systems*. 2000, pp. 209–215.
- [152] J. Winn and C. Bishop. “Variational message passing”. In: *Journal of Machine Learning Research* 6.Apr (2005), pp. 661–694.
- [153] K. Murphy, Y. Weiss, and M. Jordan. “Loopy belief propagation for approximate inference: An empirical study”. In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI1999)*. 1999.
- [154] Tom M. “Expectation Propagation for approximate Bayesian inference”. In: *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI2001)*. 2001.
- [155] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.
- [156] D. Blei, A. Kucukelbir, and J. McAuliffe. “Variational inference: A review for statisticians”. In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.
- [157] M. Hoffman, D. Blei, C. Wang, and J. Paisley. “Stochastic variational inference.” In: *Journal of Machine Learning Research* 14.5 (2013).
- [158] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. “Advances in variational inference”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.8 (2018), pp. 2008–2026.
- [159] C. Maddison, D. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh. “Filtering variational objectives”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6576–6586.
- [160] C. Naesseth, S. Linderman, R. Ranganath, and D. Blei. “Variational sequential Monte Carlo”. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 968–977.
- [161] S. Kullback and R. Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

- [162] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.
- [163] T. Minka et al. *Divergence measures and message passing*. Tech. rep. Technical report, Microsoft Research, 2005.
- [164] A. Dempster, N. Laird, and D. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.
- [165] Abraham L. *ad: a Python package for first- and second-order automatic differentiation*. URL: <http://pythonhosted.org/ad/>.
- [166] A. Baydin, B. Pearlmutter, A. Radul, and J. Siskind. “Automatic differentiation in machine learning: a survey”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 5595–5637.
- [167] J. Rosenthal and G. Roberts. “Coupling and ergodicity of adaptive MCMC”. In: *Journal of Applied Probability* 44 (2007), pp. 458–475.