# Inference and Characterization of Planar Trajectories

Zhanglong Cao

a thesis submitted for the degree of

Doctor of Philosophy

at the University of Otago, Dunedin,

New Zealand.

All knowledge is, in the final analysis, history.
All science are, in the abstract, mathematics.
All judgments are, in their rationale, statistics.

– C. Radhakrishna Rao.

**Abstract**

This is the abstract of this thesis.

**Acknowledgements**

First of all, I am so grateful to my main supervisor Dr. Matthew Parry. He offered me this golden opportunity to join this joint project with Physics department and work with him and other talent researchers. His extensive knowledge at Statistics and Physics lead me on the right way and encourage me throughout this difficult project. His passionate at exploring unknown world of science motivated me and will be motivating me to go further and deeper in the area of developing and implementing statistical models.

Secondly, I would like to appreciate my co-supervisor Professor David Brant for his great help in my research process. He gave me extraordinary supports in the first six months of my second year when Matthew was away. His guidance and expert advice inspired me in finding a new method and lighted my way into darkness.

A special gratitude goes out to the Ministry of Business, Innovation and Employment for supporting my study and this research. This project would have been impossible without their generous fundings. Additionally, many thanks to TracMap company for providing all the research GPS data, which is the base stone of all the work.

Moreover, I want to thank all the support staff at Mathematics and Statistics Department for their great job, including Lenette, Marguerite, Leanne, Chris and Greg, and give many thanks to my officemates and friends, including but not limited to Paula, Chuen Yan, Johannes, Vivian, Juan, Bob and his family, Freddy and his family, for their friendship and accompanying me during the three years. We had enjoyable and unforgeable experiences together.

And finally, last but by no means least, I will thank my parents for their wise counsel and encouragements to me for studying overseas, and my beloved wife Yingying for all her love and support, and my daughter Anyang, who was born in my second year of study. She is the greatest accomplishment of my life.

Thanks for all your encouragements!

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Problem Statement

The Global Positioning System (abbreviated as GPS) is a space-based navigation system consisting 24 satellites running around the earth orbits at the altitude 202,00 km. These satellites cover 98% of the earth surface and at least 4 satellites are available at anytime from anywhere on the earth or near the earth orbit. With four or more satellites, a GPS receiver can provide geographic information to them and triangulate its location on the ground, such as longitude, latitude and elevation. These receivers are using passive locating technology, by which they can receive signals without transmitting any data. Therefore, GPS is used in plenty applications in military and general public, including aircraft tracking, vehicle navigation, surveying, astronomy and so on. Tractors, working on an orchard or vineyard, are often mounted with GPS units that provide position and velocity information to orchardist and landlords. With this information, an orchardist is able to follow the trajectory and motion patterns of tractors and monitor its working status.

Researchers are wondering how to get a more accurate estimation for the position of a moving vehicle and to track its moving trajectory instantly. That is the problem, which has been confusing researchers for a few years and I am exploring in this paper.

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete data linear filtering problem. The Kalman filter is a set of mathematical equations that provides an efficient computational means to estimate the state of a process in a recursive way, that minimizes the mean of the squared error Greg Welch (2006). Fernando Tusell gave a review of some R packages, which could be used to fit data with Kalman Filter methods Tusell (2011). However, limited to its property,

Kalman Filter is tied up for a dynamic system, where the parameters and noise variances are unknown. In some dynamic systems, the variances are obtained based on the system identification algorithm, correlation method and least squares fusion criterion. To solve this issue, a self-tuning weighted measurement fusion Kalman filter is presented in Chenjian RAN (2010). Also, a new adaptive Kalman filter will be a good choice Oussalah and De Schutter (2001).

## 1.2 Spline Reconstruction

Kalman Filter is a good choice. And there probably a more simple and better way to fit our data – the splines. Hastie introduces several kinds of splines that we could use in his book Hastie *et al.* (2008). The core idea of splines is to augment the vector of inputs $X$ with additional variables, then use linear models in this space of derived input features. Adding constraints to construct basis functions $h_i(x), i = 1, 2, \ldots, n$, a linear basis expansion in $X$ could be represented as

$$f(x) = \sum_{i=1}^{n} \theta_i h_i(x)$$

Once the basis functions $h_i(x)$ have been determined, the models are linear in the variables space.

Smoothing spline is a basic spline and could be used in two-dimensions. Mehdi Zamani proved that in two-dimensional spaces, splines have the advantages of simplicity and less computational operations Zamani (2010). The most important part of this method is to find the parameter $\lambda$. We have a process (function) to calculate the $RSS$ (Residual Sum of Square), where the smallest $RSS$ is, the best $\lambda$ would be

$$RSS(f, \lambda) = \sum_{n=1}^{n} \{(y_i) - g(x_i)\}^2 + \lambda \int g''(x_i)^2$$

The function $g(x)$ that minimizes $RSS$ is a natural cubic spline with knots at $x_1, ..., x_n$ Hastie *et al.* (2008).

Some ways are give to estimate the parameter Wood (2000), Kim and Gu (2004). However, we hope that $\lambda$ could be a piecewise parameter, rather than a constant number, then we will have piecewise penalty functions. Grace Wahba introduced adaptive splines and a method to calculate piecewise parameters Donoho *et al.* (1995) and Ziyue Liu and Wensheng Guo improved the method Liu and Guo (2010). But to get these parameters, we have to compromise to use adaptive smoothing splines.

Using splines to fit data has been studies by many researchers. Some simple examples are given in the book Hastie *et al.* (2008). Moreover, Fujiichi Yoshimoto offers a method can treat not only data with a smooth underlying function, but also data with an underlying function having discontinuous points and/or cusps Yoshimoto *et al.* (2003). Dan Simon discussed a new type of algebraic spline, which is used to derive a filter for smoothing or interpolating discrete data points in eighth-order Simon (2004).

Based on these methods, Matthew decided to create a new spline. Temporarily, we name the new spline "Tractor Spline". This spline has some properties. This spline has two linear segments outside the knots and $n-1$ internal cubics. The spline needs $4n = 4(n-1)+2\times 2$ constraints per component. These are provided by specifying the values and first derivatives at the knots. This means $2n$ constraints per component but they count twice since they constrain the spline on both sides of the knot. After that, once we know the functions of tractor spline, the position of tractor at time $t$ will be known. Moreover, we hope to integrate the details of orchard to improve predictions and begin to characterise the type of tractor trajectories at a higher level.

The observed position data set $y_i$ always comes with some errors $\varepsilon_i$, which are assumed independent Gaussian distribution with variance $\sigma_n^2$. A popular method for finding $f(x)$ that fits these data is to augment/replace the vector of inputs $\mathbf{X}$ with additional variables, which are transformations of $\mathbf{X}$, and then use linear models in this new space of derived input features.Trevor Hastie (2009)

In regression problem, linear regression, linear discriminant analysis, logistic regression and separating hyperplanes all rely on a linear model. With the good property of linear model, easy to be interpreted and first order Taylor approximation to $f(t)$, it is more convenient to represent $f(t)$ by linear model. However, the true function $f(t)$ is unlikely to be an actual linear function in space $\mathbb{R}$. Researchers found some methods for moving beyond linearity. One of them is replacing the vector of inputs $\mathbf{T}$ with its transformations as new variables, and then use linear models in this new space of derived input features.

**Temporally put here** Due to the noise generated from observation units, one can use regression methods to find the best reconstruction returning the smallest sum square errors among all the sequences. Consider a regression model $y_i = f(t_i) + \epsilon_i$, where $a \leq t_1 < \cdots < t_n \leq b$ and $f \in C^2[a,b]$ is an unknown smooth function, $(\epsilon_i)_{i=1}^n \sim N(0,\sigma^2)$ are random errors. In a classical parametric regression, $f$ is assumed having the form $f(x,\beta)$, which is known up to the data estimated parameters $\beta$ Kim and Gu (2004). When $f(x,\beta)$ is linear in $\beta$, we will have a standard linear model.

Almost every technique found in the scientific literature on the trajectory planning problem is based on the optimization of some parameter or some objective function Gasparetto and Zanotto (2007). The most significant optimality criteria are:

- (1) minimum execution time,
- (2) minimum energy (or actuator effort),
- (3) minimum jerk.

Besides the aforementioned approaches, some hybrid optimality criteria have also been proposed (e.g. time energy optimal trajectory planning).

Denote by $h_m(t) : \mathbb{R} \mapsto \mathbb{R}$ the $m$th transformation of $t$, $m = 1, \cdots, M$. We then model

$$f(t) = \sum_{m=1}^{M} \beta_m h_m(t). \tag{1.1}$$

a linear basis expansion of $\mathbf{t}$ in $\mathbb{R}$, where $h_m(t)$ are named basis functions, $\beta_m$ are coefficients. Once the basis functions $h_m$ have been determined, the models are linear in these new variables, and the fitting proceeds as before.

Suppose we are given observed data $t_1, t_2, \cdots, t_n$ on interval $[0, 1]$, satisfying $0 \leq t_1 < t_2 < \cdots < t_n \leq 1$. A piecewise polynomial function $f(t)$ can be obtained by dividing the interval into contiguous intervals $(t_1, t_2), \cdots, (t_{n-1}, t_n)$, and representing $f$ by a separate polynomial in each interval. The points $t_i$ are called knots. For example,

$$f(t) = d_i(t - t_i)^3 + c_i(t - t_i)^2 + b_i(t - t_i) + a_i, \tag{1.2}$$

for given coefficients $d_i, c_i, b_i$ and $a_i$, where $t_i \leq t \leq t_{i+1}$, $i = 1, 2, \cdots, n$. $f$ is a cubic spline on $[0, 1]$ if (1) on each intervals $f$ is a polynomial; (2) the polynomial pieces fit together at knots $t_i$ in such a way that $f$ itself and its first and second derivatives are continuous at each $t_i$. If the second and third derivatives of $f$ are zero at 0 and 1, $f$ is said to be a natural cubic spline. These conditions are called natural boundary conditions.

Over all spline functions $f(t)$ with two continuous derivatives fitting these observed data, the curve estimate $\hat{f}(t)$ will be defined to be the minimizer the following penalized residual sum of squares, **[edited expressions]**

$$\text{MSE}(f, \lambda) = \frac{1}{n} \sum_{i=1}^{n} (f(t_i) - y_i)^2 + \lambda \int_0^1 (f''(t))^2 dt \tag{1.3}$$

where $\lambda$ is a fixed smoothing parameter, $(t_i, y_i)$, $i = 1, \cdots, n$ are observed data and $0 \leq t_1 < t_2 < \cdots < t_n \leq 1$. In equation (3.2), the smoothing parameter $\lambda$ controls the trade-off between over-fitting and bias,

$$\begin{cases} \lambda = 0: & f \text{ can be any function that interpolates the data,} \\ \lambda = \infty: & \text{the simple least squares line fit since no second derivative can be tolerated.} \end{cases} \tag{1.4}$$

In our case, the velocity data set $v_i$ with some independent Gaussian distributed errors $\varepsilon_i \sim N(0, \frac{\sigma_n^2}{\gamma})$ are used to estimate $f(t)$ simultaneously. $f$ is a linear combination of basis functions, as shown in equation (3.1), in the meantime, $f'$ is a linear combination of the first derivative of these basis functions

$$f'(t) = \sum_{m=1}^{M} \alpha_m h'_m(t). \tag{1.5}$$

The velocity information is incorporated into MSE equation (3.2) by the addition of velocity term $(f'(t_i) - v_i)^2$. Then it becomes

$$\text{MSE}(f, \lambda, \gamma) = \frac{1}{n}\sum_{i=1}^{n}(f(t_i) - y_i)^2 + \frac{\gamma}{n}\sum_{i=1}^{n}(f'(t_i) - v_i)^2 + \lambda \int_0^1 (f''(t))^2 dt, \tag{1.6}$$

and $\hat{f}$ is the minimizer of the MSE equation (1.6).

In the model $y = f(t) + \varepsilon$, it is reasonable to assume that the observed data $y_i$ is Gaussian distribution with mean $f(t_i)$ and variance $\sigma_n^2$. In a similar way, the velocity is estimated as $v = f'(t) + \frac{\varepsilon}{\gamma}$, where $v_i$ is Gaussian distribution with mean $f'(t_i)$ and variance $\frac{\sigma_n^2}{\gamma}$. Then the joint distribution of $\mathbf{y}, \mathbf{v}, f(t)$ and $f'(t)$ is normal with zero mean and a covariance matrix, which can be estimated through Gaussian Process Regression.

Given a series of such fixes, I want to construct the most likely path taken by the tractor with some smart modelling of the movement. I then want to develop higher level characterisations of the trajectories and apply this to more general problems in curve and object recognition.

### 1.2.1 Cross Validation

### 1.2.2 K-Fold Cross Validation

Based on the procedure given by Wahba and Wold (1975), we follow the improved steps to calculate a K-fold cross validation.

Step 1. Remove the first data $t_1$ and last date $t_n$ from the dataset.

Step 2. Divide dataset into k groups:

$$\begin{aligned}
&\text{Group 1} : t_2, t_{2+k}, \cdots \\
&\text{Group 2} : t_3, t_{3+k}, \cdots \\
&\vdots \\
&\text{Group k} : t_{k+1}, t_{2k+1}, \cdots
\end{aligned}$$

Step 3. Guess values of $\lambda_{down}, \lambda_{up}$ and $\gamma$.

Step 4. Delete the first group of data. Fit a smoothing spline to the first data, the rest groups of dataset and the last data, with $\lambda_{down}, \lambda_{up}$ and $\gamma$ in step 3. Compute the sum of squared deviations of this smoothing spline from the deleted data points.

Step 5. Delete instead the second group of data. Fit a smoothing spline to the remaining data with $\lambda_{down}, \lambda_{up}$ and $\gamma$. Compute the sum of squared deviations of the spline from deleted data points.

Step 6. Repeat Step 5 for the 3rd, 4th, $\cdots$, $k$th group of data.

Step 7. Add the sums of squared deviations from steps 4 to 6 and divide by $k$. This is the cross validation score of three parameters $\lambda_{down}, \lambda_{up}$ and $\gamma$.

Step 8. Vary $\lambda_{down}, \lambda_{up}$ and $\gamma$ systematically and repeat steps 4-7 until CV shows a minimum.

## 1.3 Gaussian Process Regression

A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution, Rasmussen and Williams (2006).

A GP is fully defined by its mean $m(t)$ and covariance $K(s,t)$ functions as

$$m(t) = \mathbb{E}[f(t)] \tag{1.7}$$

$$K(s,t) = \mathbb{E}[(f(s) - m(s))(f(t) - m(t))], \tag{1.8}$$

where $s$ and $t$ are two variables, and a function $f$ distributed as such is denoted in form of

$$f \sim GP(m(t), K(s,t)). \tag{1.9}$$

Usually the mean function is assumed to be zero everywhere.

Given a set of input variables $\mathbf{T}$ for function $f(t)$ and the output $\mathbf{y} = f(\mathbf{T}) + \varepsilon$ with independent identically distributed Gaussian noise $\varepsilon$ with variance $\sigma_n^2$, we can use the

above definition to predict the value of the function $f_* = f(t_*)$ at a particular input $t_*$. As the noisy observations becoming

$$\text{cov}(y_p, y_q) = K(t_p, t_q) + \sigma_n^2 \delta_{pq} \tag{1.10}$$

where $\delta_{pq}$ is a Kronecker delta which is one iff $p = q$ and zero otherwise, the joint distribution of the observed outputs $\mathbf{y}$ and the estimated output $f_*$ according to prior is

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(\mathbf{T}, \mathbf{T}) + \sigma_n^2 I & K(\mathbf{T}, t_*) \\ K(t_*, \mathbf{T}) & K(t_*, t_*) \end{bmatrix}\right). \tag{1.11}$$

The posterior distribution over the predicted value is obtained by conditioning on the observed data

$$f_* | \mathbf{y}, \mathbf{T}, t_* \sim N(\bar{f}_*, \text{cov}(f_*)) \tag{1.12}$$

where

$$\bar{f}_* = \mathbb{E}[f_* | \mathbf{y}, \mathbf{T}, t_*] = K(t_*, \mathbf{T})[K(\mathbf{T}, \mathbf{T}) + \sigma_n^2 I]^{-1} \mathbf{y}, \tag{1.13}$$

$$\text{cov}(f_*) = K(t_*, t_*) - K(t_*, \mathbf{T})[K(\mathbf{T}, \mathbf{T}) + \sigma_n^2 I]^{-1} K(\mathbf{T}, t_*). \tag{1.14}$$

We now add velocity information $\mathbf{v} = f'(\mathbf{T}) + \varepsilon'$, where $\varepsilon'$ is independent distributed Gaussian noise with variance $\frac{\sigma_n^2}{\gamma}$.

It is expected that a position point $y_i$ and velocity point $v_i$ are all effected by other points $\mathbf{y}$ and $\mathbf{v}$. So the covariance matrix for $\mathbf{y}$ and $\mathbf{v}$ is

$$\Sigma(\mathbf{y}, \mathbf{v}) = \begin{bmatrix} \text{cov}(\mathbf{y}, \mathbf{y}) & \text{cov}(\mathbf{y}, \mathbf{v}) \\ \text{cov}(\mathbf{v}, \mathbf{y}) & \text{cov}(\mathbf{v}, \mathbf{v}) \end{bmatrix}, \tag{1.15}$$

where obviously $\text{cov}(\mathbf{y}, \mathbf{v}) = \text{cov}(\mathbf{v}, \mathbf{y})$. Then the joint distribution is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix} \sim N(\mu_{y,v}, \Sigma_{y,v}). \tag{1.16}$$

Define $f_*$ and $f'_*$ the estimated position and velocity values at point $t_*$. From equation (1.15) and using similar idea, it is easily to get the covariance matrices

$$\begin{aligned} \Sigma(f_*, \mathbf{v}) &= \begin{bmatrix} \text{cov}(f_*, f_*) & \text{cov}(f_*, \mathbf{v}) \\ \text{cov}(\mathbf{v}, f_*) & \text{cov}(\mathbf{v}, \mathbf{v}) \end{bmatrix}, \\ \Sigma(\mathbf{y}, f'_*) &= \begin{bmatrix} \text{cov}(\mathbf{y}, \mathbf{y}) & \text{cov}(\mathbf{y}, f'_*) \\ \text{cov}(f'_*, \mathbf{y}) & \text{cov}(f'_*, f'_*) \end{bmatrix}, \\ \Sigma(f_*, f'_*) &= \begin{bmatrix} \text{cov}(f_*, f_*) & \text{cov}(f_*, f'_*) \\ \text{cov}(f'_*, f_*) & \text{cov}(f'_*, f'_*) \end{bmatrix}, \end{aligned} \tag{1.17}$$

**[will need to give the form of these covariances at some point. in an appendix? i think you need discussion of how $f'$ is related to $f$ for a GP]**

### 1.3.1 A Reproducing Kernel in Space $\mathbb{H}$

A Reproducing Kernel in Space $\mathbb{H}$.

## 1.4 Filtering Methods

## 1.5 Sequential Monte Carlo Markov Chain Algorithm

# Chapter 2

# Adaptive Smoothing Tractor Spline for Trajectory Reconstruction

## 2.1 Introduction

GPS devices are widely used on tracking individuals and vehicles position. Objects move frequently and continuously with their position and moving status being reported to database server or recorded by GPS units or other devices. Use of GPS receivers for obtaining geographic information of trajectories has been carried out with different aims. The Kansas Department of Transportation has used GPS data to assist with the collection of highway attributes of the state highway system Ben-Arieh *et al.* (2004). A specific vehicle can be used to collect data for making maps for highway navigation systems Atkinson (2004). It can be used in studying the traffic congestion as well, and combing with Geographic Information System Taylor *et al.* (2000).

Given a sequence of position vectors in a trace system, the simplest way of construct the complete trajectory of a moving-object is by connecting positions with a sequence of lines (line-based trajectory representation) Agarwal *et al.* (2003). Vehicles with an omni-directional drive or a differential drive can actually follow such a path in a drive-and-turn fashion, though it is highly inefficient Gloderer and Hertle (2010) and this kind of non-smooth motions can cause slippage and over-actuation Magid *et al.* (2006). By contrast, most natural moving objects, such as cars and robots, typically return smooth trajectories without sharp turns.

Several methods have been invested to solve this issue. One of them is using the minimal length of path, which is a continuously differentiable curve consisting of not more than three pieces, between two postures in the plane via line segments or arcs of

circles Dubins (1957). This method is Dubins curve, which has been extended to other more-complex vehicle models but is still limited to line segments and arcs of circles Yang and Sukkarieh (2010). Additionally, discontinuities still exist in a Dubins curve and cause tracking errors.

Luckily, spline methods have been developed to overcome these issues and to construct smoothing trajectories. In 2006, Magid *et al.* (2006) proposed a path planning algorithm based on splines. The main objective of the method is the smoothness of the path not a shortest or minimum-time path. A curved-base method uses a parametric cubic function $P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ to obtain a spline that passes through any given sequence of joint position-velocity paired points $(x_1, v_1), (x_2, v_2), \ldots, (x_n, v_n)$ Yu *et al.* (2004). More generally, B-spline have a closed-form expression of positions and allows continuity of order two between the curve segments and goes through the points smoothly with ignoring the outliers, see e.g. Komoriya and Tanie (1989), Ben-Arieh *et al.* (2004). It is flexible and has minimal support with respect to a given degree, smoothness, and domain partition. Gasparetto and Zanotto (2007) is using fifth-order B-spline to compose the overall trajectory. In that paper, the author allows one to set kinematic constraints on the motion, expressed as velocity, acceleration and jerk. In computer (or computerized) numerical control (CNC), Altintas and Erkorkmaz Erkorkmaz and Altintas (2001) presented a quintic spline trajectory generation algorithm connecting a series of reference knots that produces continuous position, velocity and acceleration profiles. Yang and Sukkarieh (2010) proposed an efficient and analytical continuous curvature path-smoothing algorithm based on parametric cubic Bézier curves. Their method can fit ordered sequential points smoothly.

However, a parametric approach only captures features contained in the preconceived class of functions Yao *et al.* (2005) and increases model bias. To avoid this, nonparametric methods have been developed. Rather than giving specified parameters, it is desired to reconstruct $f$ from the data $y(t_i) \equiv y_i$, $i = 1, \ldots, n$ Craven and Wahba (1978). Smoothing spline estimates of the $f$ function appear as a solution to the following minimization problem: find $\hat{f} \in C^2[a, b]$ that minimizes the penalized residual sum of squares:

$$\text{RSS} = \sum_{j=1}^{n} (y_j - f(t_j))^2 + \lambda \int_a^b f''(t)^2 dt \tag{2.1}$$

for pre-specified value $\lambda > 0$ Aydin and Tuzemen (2012). In equation (2.1), the first part is residual sum square and it penalizes the lack of fit. The second part is roughness penalty term weighted by a smoothing parameter $\lambda$, which varies from 0 to $+\infty$ and

establishes a trade-off between interpolation and straight line. The motivation of the roughness penalty term is from a formalization of a mechanical device: if a thin piece of flexible wood, called a spline, is bent to the shape of the curve $g$, then the leading term in the strain energy is proportional to $\int f''^2$; see e.g. Green and Silverman (1993). The cost of equation (2.1) is determined not only by its goodness-of-fit to the data quantified by the residual sum of squares, but also by its roughness Schwarz (2012). For a given $\lambda$, minimizing equation (2.1) will give the best compromise between smoothness and goodness-of-fit. Notice that the first term in equation (2.1) depends only on the values of $f$ at knots $t_i, i = 1, \ldots, n$. In the book, the authors Green and Silverman (1993) show that the function that minimizes the roughness penalty for fixed values of $f(t_i)$ is a cubic spline: an interpolation of points via a continuous piecewise cubic function, with continuous first and second derivatives. The continuity requirements uniquely determine the interpolating spline, except at the boundaries Sealfon *et al.* (2005).

Zhang *et al.* (2013) proposed their method using Hermite interpolation on each intervals to fit position, velocity and acceleration with kinematic constrains. Their trajectory formulation is a combination of several cubic splines on every intervals or, in an alternative way, can be a single function found by minimizing

$$p \sum_{j=1}^{n} |y_j - f(x_j)|^2 + (1-p) \int |D^2 f(t)|^2 dt, \tag{2.2}$$

where $n$ is the number of values of $x$, $D^2$ represents the second derivative of the function f(t) and $p$ is a smoothing parameter Castro *et al.* (2006). residual sum square error objective with penalty termCastro *et al.* (2006). in which a parameter $1-p$ is used to control the curvature of the splines.

A conventional smoothing spline is controlled by one single parameter, which controls the smoothness of a spline on the whole domain. A natural extension is to allow the smoothing parameter to vary as a penalty function of the independent variable, adapting to the change of roughness in different domains Silverman (1985), Donoho *et al.* (1995). In this way, a new objective function is formulated in the form of

$$\sum_{j=1}^{n} (y_j - f(x_j))^2 + \int_T \lambda(t) f''(t)^2 dt, \tag{2.3}$$

by minimizing which, the best estimation $\hat{f}$ can be found. This approach makes adaptive smoothing as a minimization problem with a new penalty term.

Similar to the conventional smoothing spline problem, one have to to choose the penalty function $\lambda(t)$. The fundamental idea of nonparametric smoothing is to let

the data choose the amount of smoothness, which consequently decides the model complexity Gu (1998). Most methods focus on data driven criteria, such as cross validation (CV), generalized cross validation (GCV) Craven and Wahba (1978) and generalized maximum likelihood (GML) Wahba (1985). A new challenge is posed that the smoothing parameter becomes a function and is varying in domains. The structure of this penalty function controls the complexity on each domain and the whole final model. Liu and Guo proposed to approximate the penalty function with an indicator and extended the generalized likelihood to the adaptive smoothing spline Liu and Guo (2010).

In this chapter, we propose an adaptive smoothing spline method based on Hermite spline basis functions to obtain a reconstruction of $f$ and $f'$ from noisy data $\mathbf{y}$ and $\mathbf{v}$. Rather than only using residuals of $f(t_i) - y_i$, we include residuals of $f'(t_i) - v_i$ with a new parameter $\gamma$ in our objective function. In this way, the spline keeps a balance between position and velocity. By using new basis functions, we reconstruct a smoothing spline on the whole interval $[a, b]$. Derived from the new objective function, an advanced cross validation formula for $f(t)$ and $f'(t)$ is given. We show that it performs very well on simulated signal data *Blocks*, *Bumps*, *HeaviSine* and *Doppler* Donoho and Johnstone (1994). After that, we implement our algorithm on real data application and obtain a better reconstruction. This method can be used in either getting true signal from noisy data or reconstructing the trajectory of a moving object.

## 2.2 Tractor Spline

### 2.2.1 Objective Function

In a 2D curve nonparametric regression, consider $n$ time points $t_{1:n}$, such that $a \leq t_1 < t_2 < \cdots < t_n \leq b$. Let $z_i = (x_i, y_i)$ and $w_i = (u_i, v_i)$ for $i = 1, \ldots, n$. We define a positive piecewise constant function $\lambda(t)$ :

$$\lambda(t) = \lambda_i \geq 0, \tag{2.4}$$

where $t_i \leq t < t_{i+1}, t_0 = a, t_{n+1} = b$, that will control the curvature penalty in each interval. For a function $f : [a, b] \longrightarrow \mathbb{R}^2$ and $\gamma > 0$, define the objective function

$$J[f] = \frac{1}{n}\sum_{i=1}^{n}\|f(t_i) - z_i\|_2^2 + \frac{\gamma}{n}\sum_{i=1}^{n}\|f'(t_i) - w_i\|_2^2 + \sum_{i=0}^{n}\lambda_i \int_{t_i}^{t_{i+1}} \|f''(t)\|_2^2 dt, \tag{2.5}$$

where $\gamma$ is the parameter that weights the residuals between $\mathbf{f}'$ and $\mathbf{v}$, and $\lambda(t)$ is the smoothing parameter function.

**Theorem 1.** *For $n \geq 2$, the objective function $J[f]$ is minimized by a cubic spline that is linear outside the knots.*

The solution to the objective function (2.5) is called Tractor Spline.

In the following, we split the 2D function $f(x, y)$ into two sub functions $f_x(t)$ on $x$-axis and $f_y(t)$ on $y$-axis with respect to time $t$. Compared with other parameters, choosing time $t$ to be the parameter has some advantages: 1. The expressions of all the constraints are simpler Zhang *et al.* (2013); 2. It can be simply applied from 2-dimension to 3-dimension by adding an extra $z$-axis.

### 2.2.2 Basis Functions

Suppose we have a time series sequence of observed dataset $a = t_1 < t_2 < \cdots < t_n = b$. The function $f(t)$ (stands for $f_y(t)$) defined on this interval $[t_1, t_n]$ is called Tractor Spline, if it is the solution to the objective function (2.5). Then it has the following property: on each interior interval $(t_i, t_{i+1})$, $i = 2, \ldots, n-2$, $f(t)$ is a cubic polynomial, but on interval $(t_1, t_2)$ and $(t_{n-1}, t_n)$ can be linear; $f(t)$ fits together at each point $t_i$ in such a way that $f(t)$ itself and its first derivatives are continuous at each $t_i$, $i = 2, \ldots, n-2$.

Using Hermite interpolation on an arbitrary interval $[t_i, t_{i+1}]$, the cubic spline basis functions can be constructed as follows

$$h_{00}^{(i)}(t) = \begin{cases} 2(\frac{t-t_i}{t_{i+1}-t_i})^3 - 3(\frac{t-t_i}{t_{i+1}-t_i})^2 + 1 & t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{2.6}$$

$$h_{10}^{(i)}(t) = \begin{cases} \frac{(t-t_i)^3}{(t_{i+1}-t_i)^2} - 2\frac{(t-t_i)^2}{t_{i+1}-t_i} + (t-t_i) & t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{2.7}$$

$$h_{01}^{(i)}(t) = \begin{cases} -2(\frac{t-t_i}{t_{i+1}-t_i})^3 + 3(\frac{t-t_i}{t_{i+1}-t_i})^2 & t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{2.8}$$

$$h_{11}^{(i)}(t) = \begin{cases} \frac{(t-t_i)^3}{(t_{i+1}-t_i)^2} - \frac{(t-t_i)^2}{t_{i+1}-t_i} & t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}. \tag{2.9}$$

Then a Hermite spline $f^{(i)}(t)$ on interval $[t_i, t_{i+1})$ with points $p_i = \{y_i, v_i\}$ and $p_{i+1} = \{y_{i+1}, v_{i+1}\}$ can be expressed as

$$f^{(i)}(t) = h_{00}^{(i)}(t)y_i + h_{10}^{(i)}(t)v_i + h_{01}^{(i)}(t)y_{i+1} + h_{11}^{(i)}(t)v_{i+1}. \tag{2.10}$$

To construct a Tractor Spline on the entire interval $[t_1, t_n]$, the new basis functions are defined in such way, that $N_1 = h_{00}^{(1)}$, $N_2 = h_{10}^{(1)}$, and for all $k = 1, 2, \ldots, n-2$,

$$N_{2k+1} = \begin{cases} h_{01}^{(k)} + h_{00}^{(k+1)} & \text{if } t < t_n \\ 2(\frac{t-t_{n-1}}{t_n - t_{n-1}})^3 - 3(\frac{t-t_{n-1}}{t_n - t_{n-1s}})^2 + 1 & \text{if } t = t_n \end{cases}, \tag{2.11}$$

$$N_{2k+2} = \begin{cases} h_{11}^{(k)} + h_{10}^{(k+1)} & \text{if } t < t_n \\ \frac{(t-t_{n-1})^3}{(t_n - t_{n-1})^2} - 2\frac{(t-t_{n-1})^2}{t_n - t_{n-1}} + (t - t_{n-1}) & \text{if } t = t_n \end{cases}, \tag{2.12}$$

and

$$N_{2n-1} = \begin{cases} h_{01}^{(n-1)} & \text{if } t < t_n \\ -2(\frac{t-t_{n-1}}{t_n - t_{n-1}})^3 + 3(\frac{t-t_{n-1}}{t_n - t_{n-1}})^2 & \text{if } t = t_n \end{cases}, \tag{2.13}$$

$$N_{2n} = \begin{cases} h_{11}^{(n-1)} & \text{if } t < t_n \\ \frac{(t-t_{n-1})^3}{(t_n - t_{n-1})^2} - \frac{(t-t_{n-1})^2}{t_n - t_{n-1}} & \text{if } t = t_n \end{cases}. \tag{2.14}$$

**Theorem 2.** *On $[t_1, t_n]$, the functions $N_1, \ldots, N_{2n}$ provide a basis for the set of functions which are continuous, have continuous first derivatives and are cubic on each open interval $(t_i, t_{i+1})$, where $i = 1, \ldots, n-1$.*

As independent basis functions, $N_1(t), \ldots, N_{2n}(t)$ span a $2n$ dimensional function space $\mathbb{H}$. For any $f \in \mathbb{H}$, it can be represented in the form of

$$f = \sum_{k=1}^{2n} \theta_k N_k(t), \tag{2.15}$$

where $\{\theta_k\}_{k=1}^{2n}$ are parameters.

Figure (2.1) presents two basis functions on an arbitrary interval $[t_k, t_{k+2})$ where they are continuous and differential. At the interior joint knot $t_k$, basis functions in the previous and following interval share the same position $y_k$ and velocity $v_k$.

From the definition of basis functions, it can be seen that at a joint knot of two neighbor intervals, Hermite spline basis share the same $y_i$ and $v_i$. With this property, we construct Tractor Spline basis functions. There are 2 parameters on each joint knots and 4 on the start and end knots. Then the degrees of freedom for parameters is $2n - 4 + 4 = 2n$. However $2(n-2)$ constraints are added on the joint knots, on which the spline have continuous first and second derivatives. Additional two constraints are added to the starting and ending knots to keep their first derivatives continuous. Then the degrees of freedom for parameters is 2.

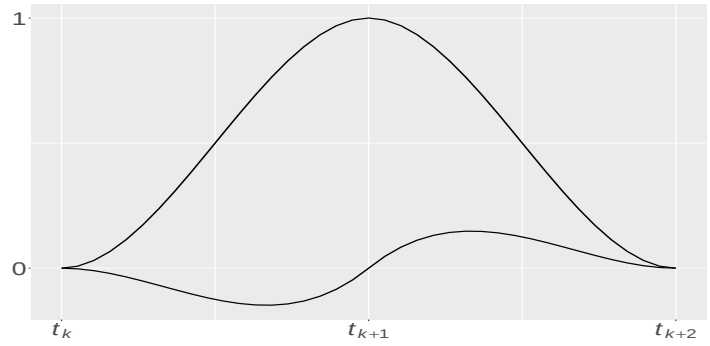



Figure 2.1: The two basis functions $N_{2k+1}$ and $N_{2k+2}$ on an arbitrary interval $[t_k, t_{k+2})$. It is apparently that these basis functions are continuous on this interval and have continuous first derivatives.

### 2.2.3 Solution to The Objective Function

Basis functions have been defined in the previous subsection, therefore the Tractor Spline $f(t)$ on $[a, b]$, where $a \leq t_1 < t_2 < \cdots < t_{n-1} < t_n \leq b$, can be found by minimizing objective function (2.5), which reduces to

$$\text{MSE}(\theta, \lambda, \gamma) = (\mathbf{y} - \mathbf{B}\theta)^\top (\mathbf{y} - \mathbf{B}\theta) + \gamma(\mathbf{v} - \mathbf{C}\theta)^\top (\mathbf{v} - \mathbf{C}\theta) + n\theta^\top \Omega_\lambda \theta, \tag{2.16}$$

where $\{\mathbf{B}\}_{ij} = N_j(t_i)$ , $\{\mathbf{C}\}_{ij} = N_j'(t_i)$ and $\{\Omega_{2n}^{(k)}\}_{jk} = \int_{t_k}^{t_{k+1}} \lambda_k N_j''(t) N_k''(t) dt$. After substituting the series observation $t_1, \ldots, t_n$ into basis functions, we get $N_1(t_1) = 1, N_1(t_2) = 0, \ldots, N_{2k-1}(t_k) = 1, N_{2k}(t_k) = 0, \ldots, N_{2n-1}(t_n) = 1, N_{2n}(t_n) = 0$; and into first derivative of basis functions, we get $N_1'(t_1) = 0, N_1'(t_2) = 1, \ldots, N_{2k-1}'(t_k) = 0, N_{2k}'(t_k) = 1, \ldots, N_{2n-1}'(t_n) = 0, N_{2n}'(t_n) = 1$. That means the matrices $\mathbf{B}$ and $\mathbf{C}$ in MSE equation (2.16) are $n \times 2n$ dimensional and the elements are

$$\mathbf{B} = \{B\}_{ij} = \begin{cases} 1, & j = 2i - 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.17}$$

$$\mathbf{C} = \{C\}_{ij} = \begin{cases} 1, & j = 2i \\ 0, & \text{otherwise} \end{cases} \tag{2.18}$$

where $i = 1, \ldots, n$. The $k$-th $\Omega_{\lambda_k}^{(k)}$ is a $2n \times 2n$ matrix and its details is in appendix. Then the penalty term is

$$\mathbf{\Omega}_\lambda = \sum_{k=1}^{n-1} \Omega_{\lambda_k}^{(k)}, \tag{2.19}$$

which is a bandwidth four matrix.

The solution to (2.16) is easily seen to be

$$\hat{\theta} = (\mathbf{B}^\top \mathbf{B} + \gamma \mathbf{C}^\top \mathbf{C} + n\Omega_\lambda)^{-1}(\mathbf{B}^\top \mathbf{y} + \gamma \mathbf{C}^\top \mathbf{v}) \tag{2.20}$$

a generalized ridge regression. Then the fitted smoothing spline is given by

$$\hat{f}(t) = \sum_{i=1}^{2n} N_i(t)\hat{\theta}_i \tag{2.21}$$

A smoothing spline with parameters $\lambda(t)$ and $\gamma$ is an example of a linear smoother Trevor Hastie (2009). This is because the estimated parameters in equation (2.20) are a linear combination of $y_i$ and $v_i$. Denote by $\hat{\mathbf{f}}$ and $\hat{\mathbf{f}}'$ the $2n$ vector of fitted values $\hat{f}(t_i)$ and $\hat{f}'(t_i)$ at the training points $t_i$. Then

$$\begin{aligned} \hat{\mathbf{f}} &= \mathbf{B}(\mathbf{B}^\top \mathbf{B} + \gamma \mathbf{C}^\top \mathbf{C} + n\Omega_\lambda)^{-1}(\mathbf{B}^\top \mathbf{y} + \gamma \mathbf{C}^\top \mathbf{v}) \\ &\triangleq \mathbf{S}_{\lambda,\gamma}\mathbf{y} + \gamma \mathbf{T}_{\lambda,\gamma}\mathbf{v} \end{aligned} \tag{2.22}$$

$$\begin{aligned} \hat{\mathbf{f}}' &= \mathbf{C}(\mathbf{B}^\top \mathbf{B} + \gamma \mathbf{C}^\top \mathbf{C} + n\Omega_\lambda)^{-1}(\mathbf{B}^\top \mathbf{y} + \gamma \mathbf{C}^\top \mathbf{v}) \\ &\triangleq \mathbf{U}_{\lambda,\gamma}\mathbf{y} + \gamma \mathbf{V}_{\lambda,\gamma}\mathbf{v} \end{aligned} \tag{2.23}$$

The fitted $\hat{\mathbf{f}}$ and $\hat{\mathbf{f}}'$ are linear in $\mathbf{y}$ and $\mathbf{v}$, and the finite linear operators $\mathbf{S}_{\lambda,\gamma}, \mathbf{T}_{\lambda,\gamma}, \mathbf{U}_{\lambda,\gamma}$ and $\mathbf{V}_{\lambda,\gamma}$ are known as the smoother matrices. One consequence of this linearity is that the recipe for producing $\hat{\mathbf{f}}$ and $\hat{\mathbf{f}}'$ from $\mathbf{y}$ and $\mathbf{v}$, do not depend on $\mathbf{y}$ and $\mathbf{v}$ themselves; $\mathbf{S}_{\lambda,\gamma}, \mathbf{T}_{\lambda,\gamma}, \mathbf{U}_{\lambda,\gamma}$ and $\mathbf{V}_{\lambda,\gamma}$ depend only on $t_i, \lambda(t)$ and $\gamma$.

Suppose in a traditional least squares fitting, $\mathbf{B}_\xi$ is $N \times M$ matrix of $M$ cubic-spline basis functions evaluated at the $N$ training points $x_i$, with knot sequence $\xi$ and $M \ll N$. Then the vector of fitted spline values is given by

$$\hat{\mathbf{f}} = \mathbf{B}_\xi(\mathbf{B}_\xi^\top \mathbf{B}_\xi)^{-1}\mathbf{B}_\xi \mathbf{y} = \mathbf{H}_\xi \mathbf{y} \tag{2.24}$$

Here the linear operator $\mathbf{H}_\xi$ is a symmetric, positive semidefinite matrices, and $\mathbf{H}_\xi \mathbf{H}_\xi = \mathbf{H}_\xi$ (idempotent) Trevor Hastie (2009). In our case, it is easily seen that $\mathbf{S}_{\lambda,\gamma}, \mathbf{T}_{\lambda,\gamma}, \mathbf{U}_{\lambda,\gamma}$ and $\mathbf{V}_{\lambda,\gamma}$ are symmetric, positive semidefinite matrices as well. Additionally, by Cholesky decomposition

$$(\mathbf{B}^\top \mathbf{B} + \gamma \mathbf{C}^\top \mathbf{C} + n\Omega_\lambda)^{-1} = \mathbf{R}\mathbf{R}^\top, \tag{2.25}$$

it is easily to prove that $\mathbf{T}_{\lambda,\gamma} = \mathbf{B}\mathbf{R}\mathbf{R}^\top \mathbf{C}^\top$ and $\mathbf{U}_{\lambda,\gamma} = \mathbf{C}\mathbf{R}\mathbf{R}^\top \mathbf{B}^\top$, then we will have $\mathbf{T}_{\lambda,\gamma} = \mathbf{U}_{\lambda,\gamma}^\top$. When $\lambda = \gamma = 0$, the matrix $\mathbf{S}_{\lambda_0,\gamma_0} = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1}\mathbf{B}^\top$ is idempotent.

For sufficient cases of $\lambda(t)$ and $\gamma$, Tractor Spline has the following property:

- if $\lambda(t)$ is a piecewise constant and $\gamma \neq 0$, then $f$ and $f'$ are continuous, $f''$ is piecewise linear but not continuous at knots;

- if $\lambda(t)$ is a piecewise constant and $\gamma = 0$, the same as above;

- if $\lambda(t) = \lambda$ is a constant and $\gamma \neq 0$, the same as above;

- if $\lambda(t) = \lambda$ is a constant and $\gamma = 0$, then $f$, $f'$ are continuous, $f''$ is piecewise linear and continuous at knots.

### 2.2.4 Adjusted Penalty Term and Parameter Function

To get the reconstructed trajectory in a multi-dimensional space, one can use the Tractor Spline to find the trajectory in each dimension separately and then combine them together for a higher dimension, such as 2D and 3D. Typically, the data are not regularly sampled in time. Due to the property of Hermite spline, the combination of a multi-dimensional reconstruction for irregular time difference dataset will bring some issues. Image the situation in which a vehicle is moving along the $x$-axis, but stays unchanged on its $y$ position. By fitting $\mathbf{x}$ and $\mathbf{u}$, the Tractor Spline $f_x(t)$ will give us a best fit which returns smallest errors to the objective function. While with the same parameter $\lambda(t)$ and $\gamma$, $f_y(t)$ returns a cubic curve, where it should give us a straight line as we expected. Moreover, in some circumstances, with time increasing $\mathbf{f}$ and $\mathbf{f}'$ remain the same, or change slightly. In this situation, the Hermite spline will return wiggles in each dimension and curves in combined two dimensions.

To get a reliable reconstruction, we introduce an adjusted penalty term $\frac{(\Delta t_i)^\alpha}{(\Delta d_i)^\beta}$, where $\alpha \geq 0$ and $\beta \geq 0$, to the penalty function $\lambda(t)$, in which the Tractor Spline is penalized by its real difference of $\Delta d_i$ and $\Delta t_i$ for each interval $[t_i, t_{i+1}]$. With this term, when either $\mathbf{u}$ for $x$ or $\mathbf{v}$ for $y$ goes down or equals to 0, it will make sure that the penalty function will be large enough and returns a straight line rather than a curve on this domain. Because of the unit of the penalty term is $m^2/t^3$, to keep the same scale, $\alpha$ and $\beta$ in the adjusted penalty term are chosen as 3 and 2 respectively. From the physical point of view, the term is the reciprocal of the product of velocity and acceleration. Either velocity or acceleration goes to zero, the moving object should either stop, which returns a straight line along time on $x$ or $y$ axises and a dot on higher dimension, or keep moving with the same speed, which returns a linear instead of a curve path.

Consequently, the final form of the penalty function is

$$\lambda(t) = \frac{(\Delta t_i)^3}{(\Delta d_i)^2}\lambda, \tag{2.26}$$

where $t_i \leq t < t_{i+1}$. Eventually, in objective function, there is one parameter $\lambda$ controlling the curvature of Tractor Spline on different domains, and another one $\gamma$ controlling the residuals of velocity.

## 2.3 Parameter Selection and Cross Validation

The problem of choosing the smoothing parameter is ubiquitous in curve estimation, and there are two different philosophical approaches to this question. The first one is to regard the free choice of smoothing parameter as an advantageous feature of the procedure. The other one is to find the parameter automatically by the data Green and Silverman (1993). We prefer the latter one, use data to train our model and find the best parameters. The most well known method is cross-validation.

Assuming that the random errors have zero mean, the true regression curve $f(t)$ has the property that, if an observation $y$ is taken at a point $t$, the value $f(t)$ is the best predictor of $y$ in terms of returning a small value of $(y - f(t))^2$.

Now we focus on an observation $y_i$ at point $t_i$ as being a new observation by omitting it from the set of data, which are used to estimate $\hat{f}$. Denote by $\hat{f}^{(-i)}(t, \lambda)$ the estimated function from the remaining data, where $\lambda$ is the smoothing parameter. Then $\hat{f}^{(-i)}(t, \lambda)$ minimizes

$$\frac{1}{n}\sum_{j \neq i}(y_j - f(t_j))^2 + \lambda \int f''^2 dt \tag{2.27}$$

and $\lambda$ can be quantified by cross-validation score function

$$\mathrm{CV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}^{(-i)}(t_i, \lambda)\right)^2. \tag{2.28}$$

The basis idea of cross-validation is to choose the value of $\lambda$ that minimizes $\mathrm{CV}(\lambda)$.

An efficient way to calculate cross validation score is given by Green and Silverman (1993). Through the equation (2.24), we know that the value of the smoothing spline $\hat{f}$ depend linearly on the data $y_i$. Define the matrix $A(\lambda)$, which is a map vector of observed values $y_i$ to predicted values $\hat{f}(t_i)$. Then we have

$$\hat{\mathbf{f}} = A(\lambda)\mathbf{y} \tag{2.29}$$

and the following lemma.

**Lemma 1.** *The cross validation score satisfies*

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(t_i)}{1 - A_{ii}(\lambda)}\right)^2 \tag{2.30}$$

*where $\hat{f}$ is the spline smoother calculated from the full data set $\{(t_i, y_i)\}$ with smoothing paramter $\lambda$.*

For a Tractor Spline and its MSE function, there are two parameters need to be estimated $\lambda$ and $\gamma$. Then the objective function (2.27) becomes

$$\frac{1}{n}\sum_{j\neq i}\|y_j - f(t_j)\|_2^2 + \frac{\gamma}{n}\sum_{j\neq i}\|v_j - f'(t_j)\|_2^2 + \int \lambda(t)\|f''\|_2^2 dt, \tag{2.31}$$

and the cross-validation score function is

$$\mathrm{CV}(\lambda, \gamma) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}^{(-i)}(t_i, \lambda, \gamma)\right)^2. \tag{2.32}$$

For a Tractor Spline, the parameter $\hat{\theta} = (B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}(B^\top \mathbf{y} + \gamma C^\top \mathbf{v})$ gives us

$$\begin{aligned}\hat{\mathbf{f}} &= B\hat{\theta} = B(B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}B^\top \mathbf{y} + B(B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}C^\top \mathbf{v} \\ &= S\mathbf{y} + \gamma T\mathbf{v},\end{aligned} \tag{2.33}$$

$$\begin{aligned}\hat{\mathbf{f}}' &= C\hat{\theta} = C(B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}B^\top \mathbf{y} + C(B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}C^\top \mathbf{v} \\ &= U\mathbf{y} + \gamma V\mathbf{v}.\end{aligned} \tag{2.34}$$

From lemma 1, we can prove the following theorem.

**Theorem 3.** *The cross validation score of a Tractor Spline satisfies*

$$CV(\lambda, \gamma) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{f}(t_i) - y_i + \gamma\frac{T_{ii}}{1-\gamma V_{ii}}(\hat{f}'(t_i) - v_i)}{1 - S_{ii} - \gamma\frac{T_{ii}}{1-\gamma V_{ii}}U_{ii}}\right)^2 \tag{2.35}$$

*where $\hat{f}$ is the Tractor Spline smoother calculated from the full data set $\{(t_i, y_i, v_i)\}$ with smoothing parameter $\lambda$ and $\gamma$.*

The proof of Theorem 3 follows immediately from a lemma, and gives an expression for the deleted residuals $y_i - \hat{f}^{(-i)}(t_i)$ and $v_i - \hat{f}'^{(-i)}(t_i)$ in terms of $y_i - \hat{f}(t_i)$ and $v_i - \hat{f}'(t_i)$ respectively.

**Lemma 2.** *For fixed $\lambda, \gamma$ and $i$, denote $\mathbf{f}^{(-i)}$ by the vector with components $f_j^{(-i)} = \hat{f}^{(-i)}(t_j, \lambda, \gamma)$, $\mathbf{f}'^{(-i)}$ by the vector with components $f_j'^{(-i)} = \hat{f}'^{(-i)}(t_j, \lambda, \gamma)$, and define vectors $\mathbf{y}^*$ and $\mathbf{v}^*$ by*

$$\begin{cases} y_j^* = y_j & j \neq i \\ y_i^* = \hat{f}^{(-i)}(t_i) & otherwise \end{cases}, \tag{2.36}$$

$$\begin{cases} v_j^* = v_j & j \neq i \\ v_i^* = \hat{f}'^{(-i)}(t_i) & otherwise \end{cases}. \tag{2.37}$$

*Then*

$$\hat{\mathbf{f}}^{(-i)} = S\mathbf{y}^* + \gamma T\mathbf{v}^* \tag{2.38}$$

$$\hat{\mathbf{f}}'^{(-i)} = U\mathbf{y}^* + \gamma V\mathbf{v}^* \tag{2.39}$$

## 2.4 Simulation Study

### 2.4.1 Numerical Examples

In this section, we examine the visual quality of the proposed method with four functions: *Blocks*, *Bumps*, *HeaviSine* and *Doppler*, which have been used in Donoho and Johnstone (1994), Donoho and Johnstone (1995) and Abramovich *et al.* (1998) because of their caricature features in imaging, spectroscopy and other scientific signal processing. However it is unfair for Tractor Spline fitting "jump" position in *Blocks* and *Bumps* function, because it fits position and velocity simultaneously and these points imply infinite first derivative in original functions, which are impossible for vehicles or individuals. In terms of this issue, we treat these functions as velocity, and use noise free points to generate accurate position data, then add noises back to them.

For calculating consideration, we use $n = 1024$ Nason (2010). Because all noises are randomly generated, for convenience of reinitialization and repetition of comparing, we set random seed as 2016. The noises are independent Gaussian distribution $\epsilon \sim N(0, 1)$ and signal-to-noise ratio (SNR) is 7. These data are treated as velocity (first derivative). By setting initial position $y_0 = 0$, acceleration $a_0 = 0$ and using the following formula to calculate position

$$y_{i+1} = y_i + (v_i + v_{i+1})\frac{t_{i+1} - ti}{2}, \tag{2.40}$$

we can easily generate position data. Then we add some noises, which are independent Gaussian distribution $\epsilon \sim N(0, 1)$ and SNR is 7. For wavelet transform reconstructions, we use the threshold policy of *sure* and *BayesThresh* with levels $j = 4, \ldots, 9$, see

Donoho and Johnstone (1995) and Abramovich *et al.* (1998). A semi-parametric regression model with spatially adaptive penalized splines (P-spline) is added in comparison, see Krivobokova *et al.* (2008) Ruppert *et al.* (2003).

For Tractor Spline, we have two parameters $\lambda$ and $\gamma$ to optimize. To evaluate the performance of the velocity term in objective function (2.5) and the adjusted penalty term in (2.26), the parameter $\gamma$ is set as 0 in one reconstruction of Tractor Spline, whose objective function and solution become

$$J[f]_{\gamma=0} = \frac{1}{n}\sum_{i=1}^{n}(f(t_i) - y_i)^2 + \sum_{i=1}^{n-1}\lambda_i \int_{t_i}^{t_{i+1}} f''(t)^2 dt, \tag{2.41}$$

and

$$\hat{\theta}_{\gamma=0} = (\mathbf{B}^\top\mathbf{B} + n\Omega_\lambda)^{-1}\mathbf{B}^\top\mathbf{y} \tag{2.42}$$

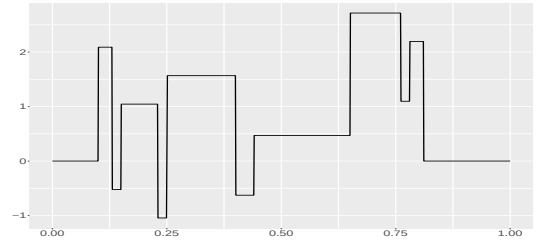
and the adjusted penalty term in (2.26) was removed from another reconstruction, noted as "Tractor Spline without APT". Figure (2.2) to (2.5) display the original (velocity), generated position, wavelet with two different threshold methods, P-spline and three kinds of Tractor Spline fitted functions. The parameters $\lambda$ and $\gamma$ of a Tractor Spline are automatically selected from formula (2.35) by *optim* function in *R* Nelder and Mead (1965).

By comparing, we can see that all these methods can rebuild up the skeleton of generated trajectory. *Wavelet(sure)* method have more wiggles in interior interval than *Wavelet(BayesThresh)*, and the latter one becomes fluctuation near boundary knots. *P-spline* gives much smoother fitting than wavelets, but the drawback is losing more specific details. Tractor Spline without velocity loses some information, as can be seen from *Blocks* and *Bumps* where there should be a straight line. Tractor Spline without adjusted penalty term get over fitting when the direction changes more frequently than normal, although it catches specific feature in *HeaviSine*. The proposed Tractor Spline performs much better than other methods and returns the near-true trajectory reconstructions.

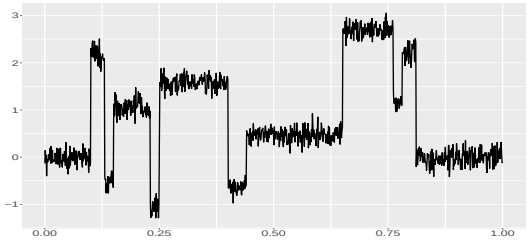
Figure 2.6 shows the estimated penalty function

$$\lambda(t) = \frac{(\Delta t)^3}{(\Delta d)^2}\lambda. \tag{2.43}$$

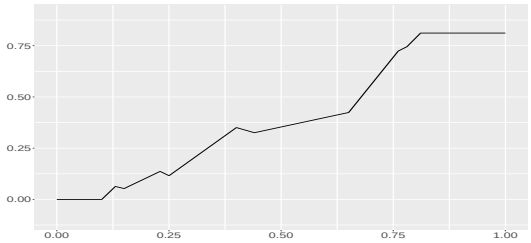
The left column illustrates the value of penalty function on different intervals and the right column is the projection on position. Bigger black dots present larger penalty values. It can be seen that $\lambda(t)$ adapts to the smoothness pattern of position and will be large where a long time gap may occur. The details of how this penalty function works will be explained in next subsection.



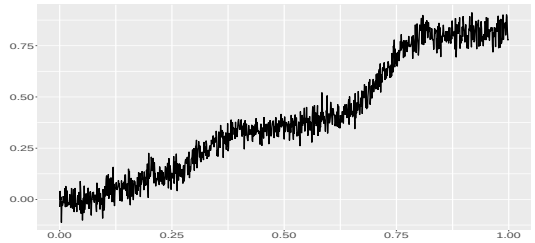
(a) The *Blocks* function.



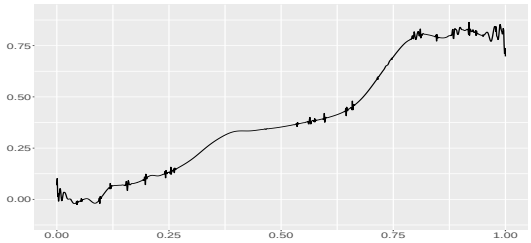
(b) Noisy *Blocks* at $SNR$=7.



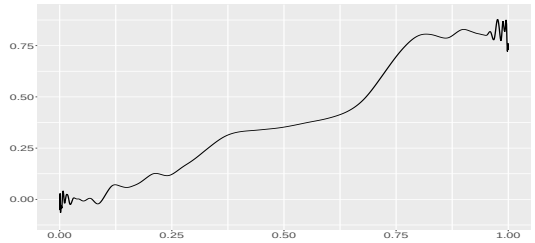
(c) Generated positions.



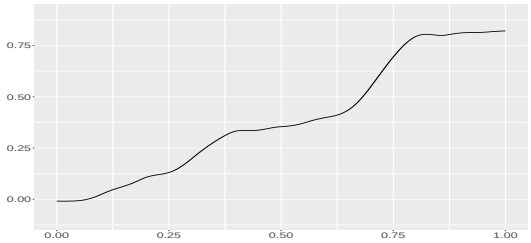
(d) Noisy position at $SNR$=7.



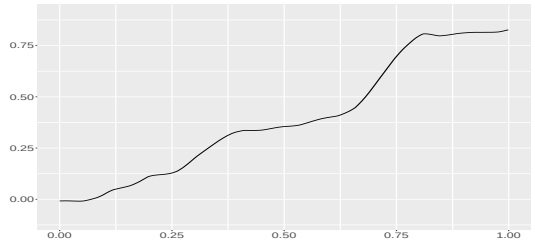
(e) Reconstruction from Wavelet by sure threshold.



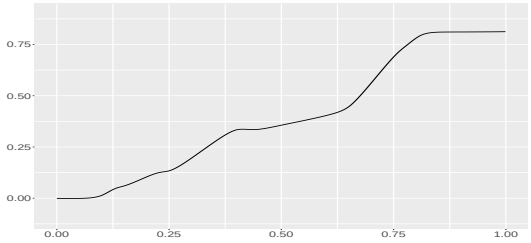
(f) Reconstruction from Wavelet by BayesThresh approach.



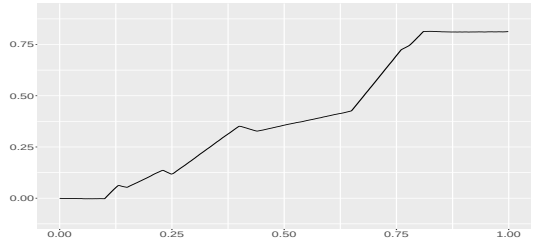
(g) Reconstruction by P-spline.



(h) Reconstruction by Tractor Spline setting $\gamma = 0$

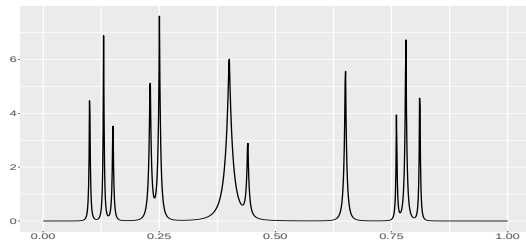


(i) Reconstruction by Tractor Spline with conventional penalty term.

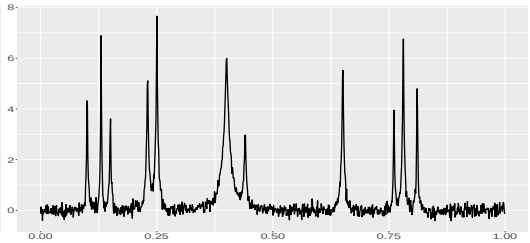


(j) Reconstruction by proposed Tractor Spline.

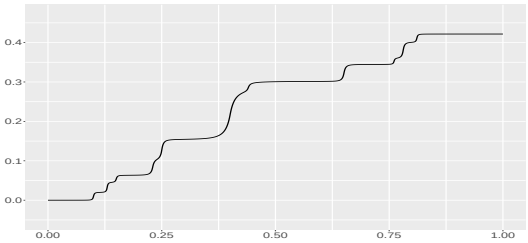
Figure 2.2: Numerical example: *Blocks*. Comparison of different reconstruction methods with simulated data.



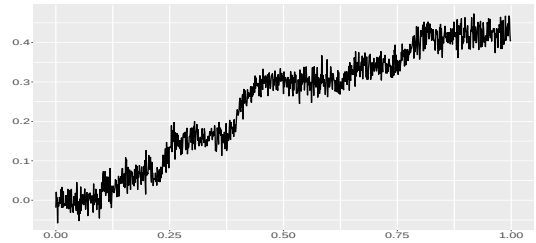
(a) True *Bumps* function.



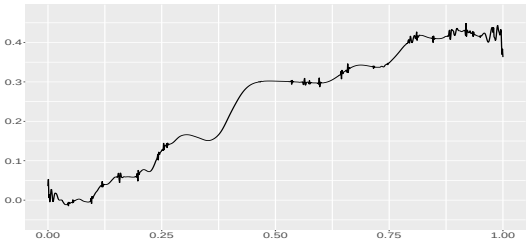
(b) Noisy *Bumps* at $SNR$=7.



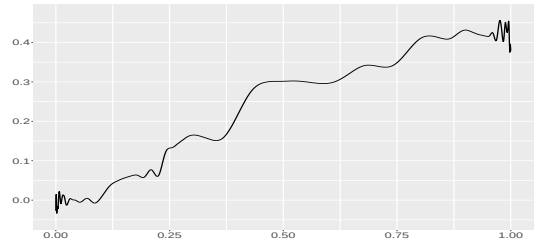
(c) Generated positions.



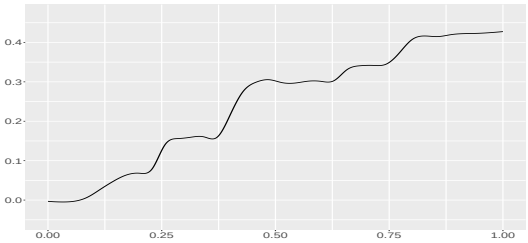
(d) Noisy position at $SNR$=7.



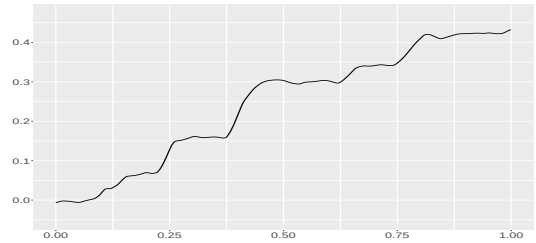
(e) Reconstruction from Wavelet by sure threshold.



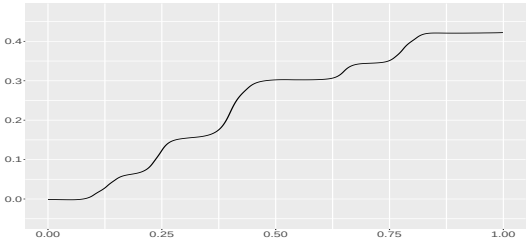
(f) Reconstruction from Wavelet by BayesThresh approach.



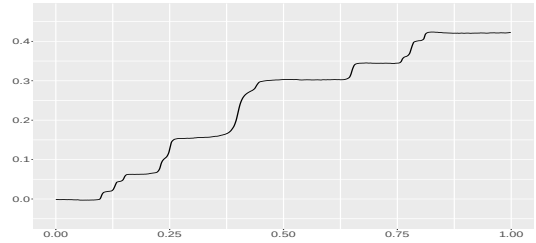
(g) Reconstruction by P-spline.



(h) Reconstruction by Tractor Spline setting $\gamma = 0$

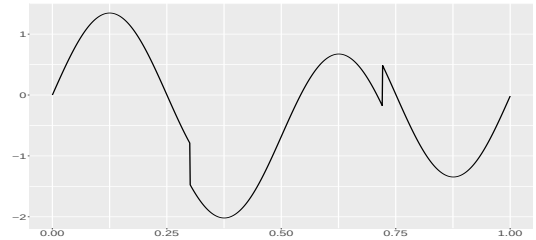


(i) Reconstruction by Tractor Spline with conventional penalty term.

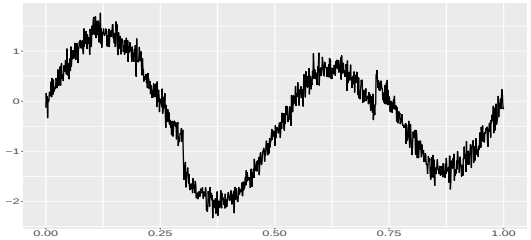


(j) Reconstruction by proposed Tractor Spline.

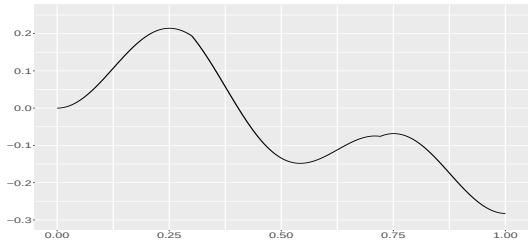
Figure 2.3: Numerical example: *Bumps*. Comparison of different reconstruction methods with simulated data.



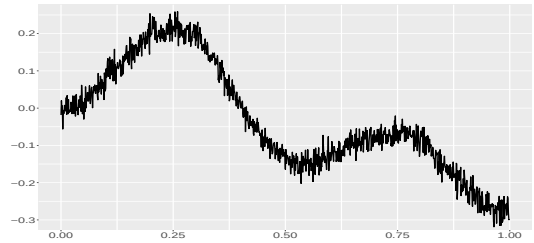
(a) True *HeaviSine* function.



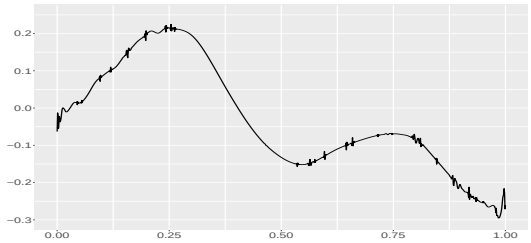
(b) Noisy *HeaviSine* at $SNR$=7.



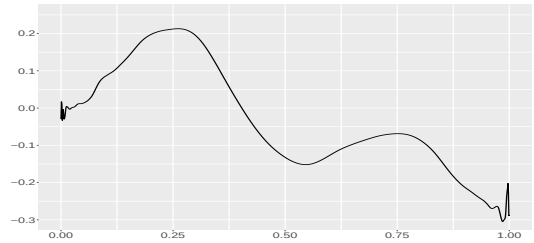
(c) Generated positions.



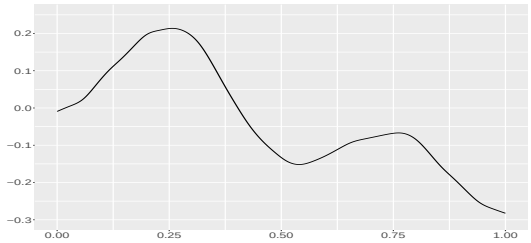
(d) Noisy position at $SNR$=7.



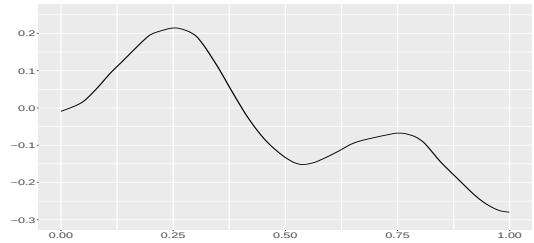
(e) Reconstruction from Wavelet by sure threshold.



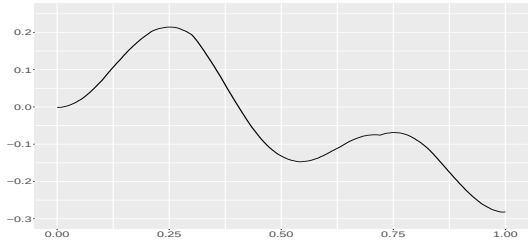
(f) Reconstruction from Wavelet by BayesThresh approach.



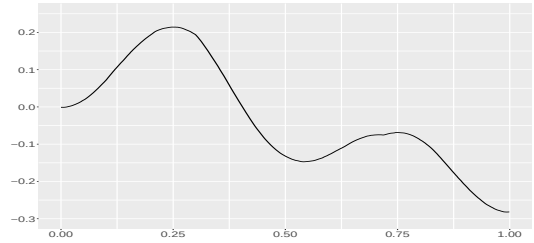
(g) Reconstruction by P-spline.



(h) Reconstruction by Tractor Spline setting $\gamma = 0$

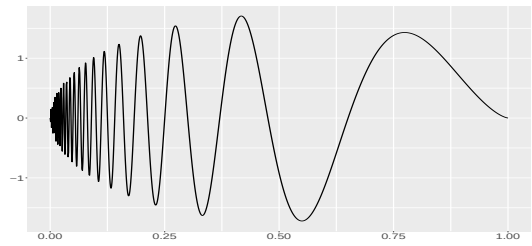


(i) Reconstruction by Tractor Spline with conventional penalty term.

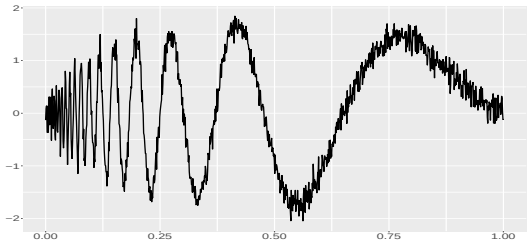


(j) Reconstruction by proposed Tractor Spline.

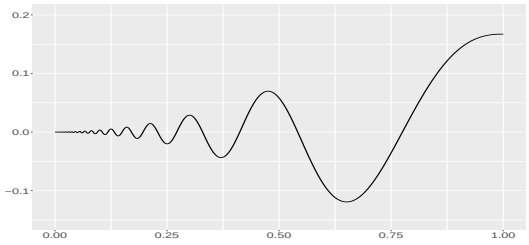
Figure 2.4: Numerical example: *HeaviSine*. Comparison of different reconstruction methods with simulated data.



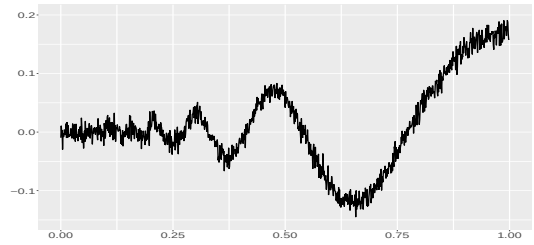
(a) True *Doppler* function.



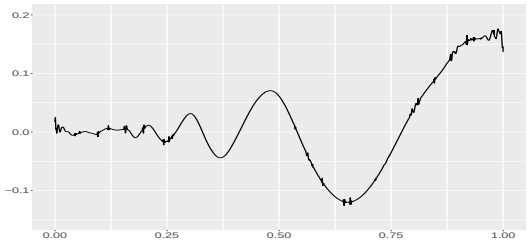
(b) Noisy *Doppler* at *SNR*=7.



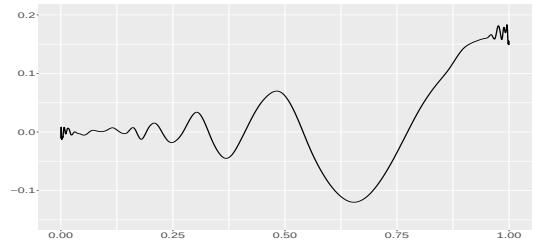
(c) Generated positions.



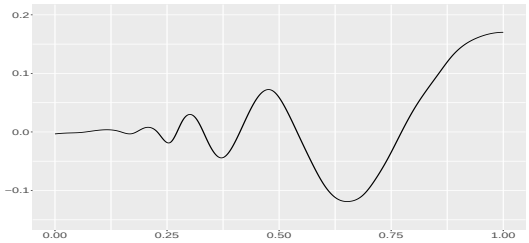
(d) Noisy position at *SNR*=7.



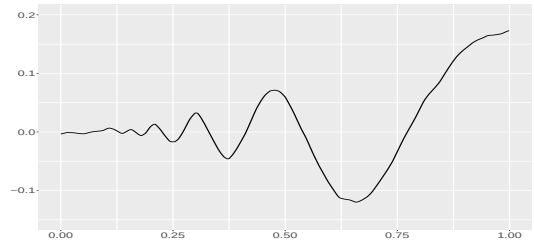
(e) Reconstruction from Wavelet by sure threshold.



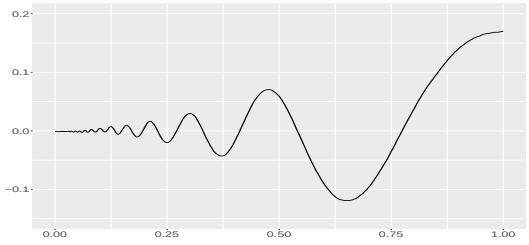
(f) Reconstruction from Wavelet by BayesThresh approach.



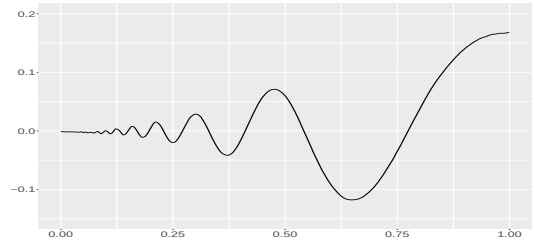
(g) Reconstruction by P-spline.



(h) Reconstruction by Tractor Spline setting $\gamma = 0$

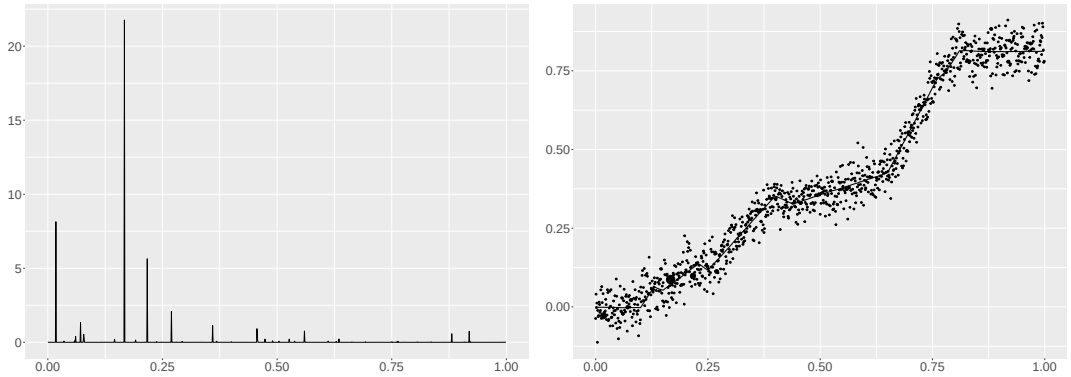


(i) Reconstruction by Tractor Spline with conventional penalty term.

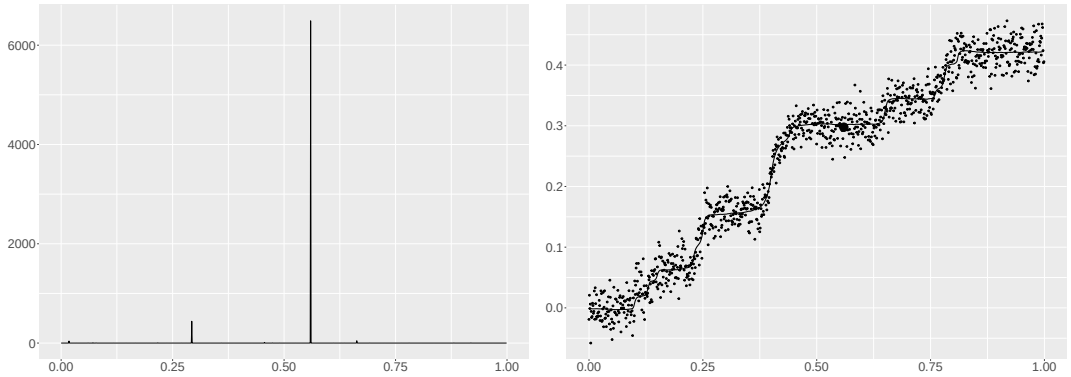


(j) Reconstruction by proposed Tractor Spline.

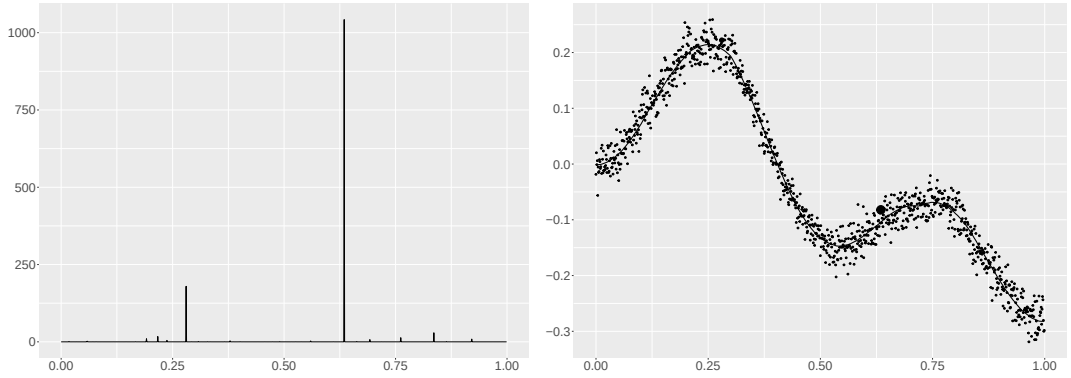
Figure 2.5: Numerical example: *Doppler*. Comparison of different reconstruction methods with simulated data.



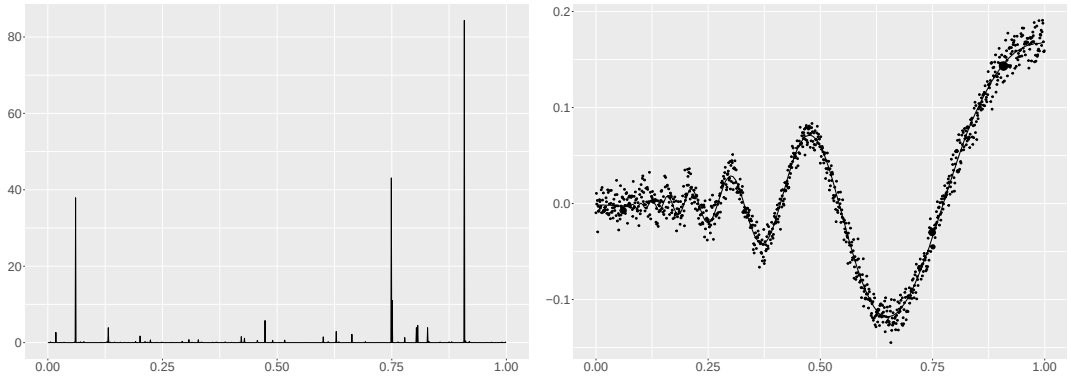

(a) Distribution of penalty values in reconstructed *Blocks* function.




(b) Distribution of penalty values in reconstructed *Bumps* function.

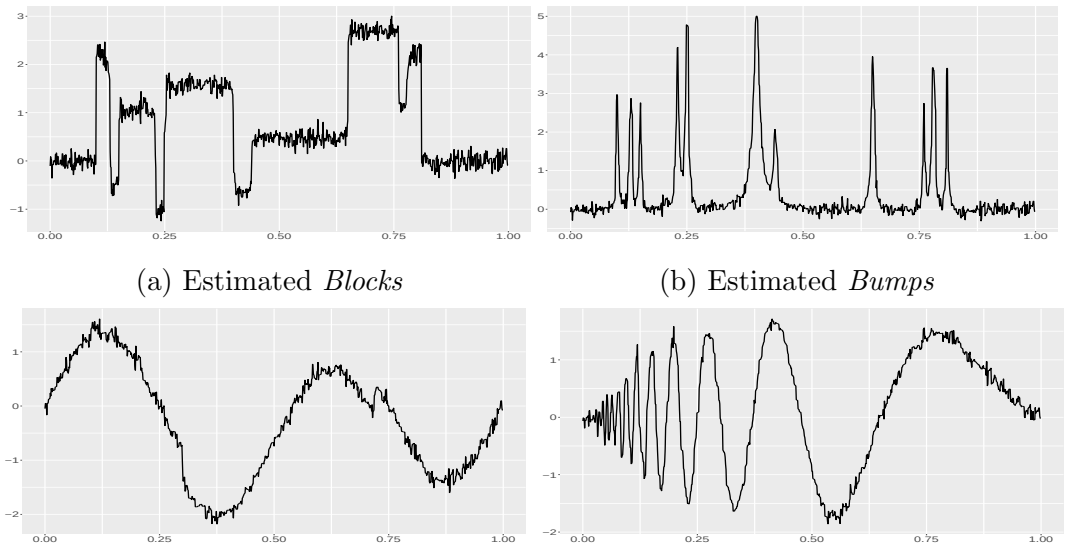



(c) Distribution of penalty values in reconstructed *HeaviSine* function.




(d) Distribution of penalty values in reconstructed *Doppler* function.

Figure 2.6: Distribution of penalty values in reconstructed Tractor Spline. Figures on left side indicate the values of $\lambda(t)$ varying in intervals. On right side, $\lambda(t)$ is projected into reconstructions. The bigger the blacks dots present, the larger the penalty values are.



(a) Estimated *Blocks* (b) Estimated *Bumps*

(c) Estimated *HeaviSine* (d) Estimated *Doppler*

Figure 2.7: Estimated velocity functions (original simulation functions) by Tractor Spline.

Figure 2.7 demonstrates the estimated velocity functions. By taking the first derivative of fitted tractor spine, it is easily to get the original four velocity functions. The fitting of velocity is not as smooth as that in position, because we only care about the smoothness of position rather than velocity in our cross-validation formula (3). However, velocity information do help us reconstruct the trajectory.

### 2.4.2 Evaluation

To examine the performance of Tractor Spline, we conducted a evaluation by comparing the mean square errors and true mean square errors, which are respectively calculated in

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_{\lambda,\gamma}(t_i))^2, \tag{2.44}$$

$$\text{TMSE} = \frac{1}{n}\sum_{i=1}^{n}(f(t_i) - \hat{f}_{\lambda,\gamma}(t_i))^2. \tag{2.45}$$

The results are shown in table 2.1 and 2.2. All of these methods have good performances in fitting noisy data. The differences of mean square error between these methods are not significant, as can be seen from table 2.1. The proposed method is not the best among these simulations according to MSE. However, from table 2.2, Tractor

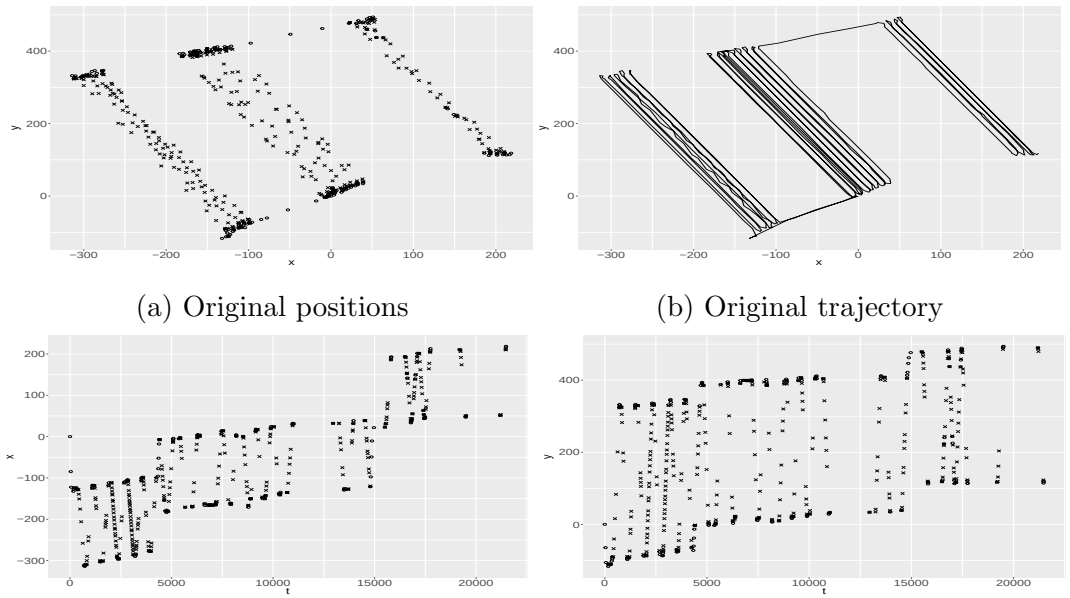
Table 2.1: MSE. Mean square errors of different methods. The numbers in bold indicate the smallest error among these methods under the same level. The difference is not significant.

| MSE ($10^{-4}$) | SNR | TS | $\text{TS}_{\gamma=0}$ | $\text{TS}_{APT=0}$ | P-spline | Wavelet(sure) | Wavelet(Bayes) |
|---|---|---|---|---|---|---|---|
| *Blocks* | 7 | 16.53 | 15.99 | 16.69 | 16.14 | **15.39** | 16.68 |
| *Blocks* | 3 | 89.79 | **87.64** | 89.94 | 88.27 | 98.35 | 90.24 |
| *Bumps* | 7 | 4.40 | 4.19 | 4.55 | 4.33 | **4.18** | 4.59 |
| *Bumps* | 3 | 23.93 | **23.19** | 24.10 | 23.55 | 26.23 | 23.74 |
| *HeaviSine* | 7 | 4.16 | 4.01 | 4.16 | 4.02 | **3.79** | 4.19 |
| *HeaviSine* | 3 | 22.63 | **22.19** | 22.65 | 22.02 | 23.53 | 22.07 |
| *Doppler* | 7 | 1.15 | **1.07** | 1.10 | 1.15 | **1.07** | 1.13 |
| *Doppler* | 3 | 6.27 | **5.94** | 6.28 | 6.05 | 6.85 | 6.29 |

Spline returns the smallest true mean square errors. The difference is significant, that means the reconstruction from Tractor Spline is closer to the true trajectory.

Table 2.2: TMSE. True mean square errors of different methods. The numbers in bold indicate the smallest error among these methods under the same level. The proposed Tractor Spline returns the smallest TMSE among all the methods under the same level except for *Doppler* with SNR=7. The differences are significant.

| TMSE ($10^{-6}$) | SNR | TS | $TS_{\gamma=0}$ | $TS_{APT=0}$ | P-spline | Wavelet(sure) | Wavelet(Bayes) |
|---|---|---|---|---|---|---|---|
| *Blocks* | 7 | **1.75** | 54.25 | 28.68 | 54.76 | 201.02 | 182.12 |
| *Blocks* | 3 | **16.44** | 152.5 | 30.76 | 171.59 | 1138.08 | 712.36 |
| *Bumps* | 7 | **1.64** | 23.44 | 21.10 | 24.21 | 71.71 | 69.26 |
| *Bumps* | 3 | **8.51** | 77.78 | 37.12 | 77.52 | 330.77 | 238.79 |
| *HeaviSine* | 7 | **1.53** | 7.80 | 1.56 | 9.54 | 55.37 | 44.88 |
| *HeaviSine* | 3 | **8.21** | 33.56 | 8.49 | 34.26 | 240.72 | 110.49 |
| *Doppler* | 7 | 1.51 | 6.67 | **1.08** | 8.26 | 14.87 | 12.01 |
| *Doppler* | 3 | **8.10** | 22.14 | 8.25 | 19.95 | 81.48 | 50.33 |




(a) Original positions

(b) Original trajectory

(c) Original positions on $x$ axis

(d) Original positions on $y$ axis

Figure 2.8: Original data points. (a) Original positions recorded by GPS units. Circle points means the boom is not working; cross points means it is working. (b) Original trajectory with line-based method: simply connect all the points sequentially with straight lines. (c) Original positions on $x$ axis. (d) Original positions on $y$ axis.

## 2.5 Application on Real Dataset

In this section, we apply the proposed method to real dataset, which are recorded by GPS units mounted on a tractor. The original dataset contains time marks, longitude, latitude, velocity, bearing (in degrees, heading to North) and boom status. First of all, we convert the longitude and latitude information from a 3D sphere to 2D surface by Universal Transverse Mercator coordinate system (UTM). After that, we split the velocity $v$ into $v_x$ and $v_y$ by

$$v_x = v \cdot \sin(\omega \frac{\pi}{180}), \tag{2.46}$$

$$v_y = v \cdot \cos(\omega \frac{\pi}{180}), \tag{2.47}$$

where $\omega$ is in degrees. Boom status is tagged as 0 if it is not working and 1 if it is. Time marks are transformed by subtract the first mark, in which way the time starts from 0. Time duplicated data, caused by errors, had been removed. In convenience of comparing with wavelet algorithm, we choose the first 512 out of 928 rows of data. The original data is plotted in figure 2.8.

To fit the real data, we bring the parameter $\lambda_d$ back to our model. Then we

Table 2.3: Mean square error. Tractor Spline returns smallest errors among all these methods. P-spline was unable to reconstruct the $y$ trajectory as the original dataset contains value-0 time differences.

| MSE | TS | $TS_{\gamma=0}$ | $TS_{APT=0}$ | P-spline | Wavelet(sure) | Wavelet(Bayes) |
|---|---|---|---|---|---|---|
| *x-axis* | 0.204582 | 0.283044 | 0.329834 | 2860.548 | 256.0494 | 6.295914 |
| *y-axis* | 0.002020 | 0.306154 | 0.311513 | *NA* | 1960.2220 | 19.332990 |

have three parameters $\lambda_d$ and $\lambda_u$ regarding to boom status and $\gamma$ controlling velocity residuals. The criteria of a good fitting is that it can catch more information, recognize time gaps between two points where tractor stops and return a smaller MSE.

### 2.5.1 1-Dimension Trajectory

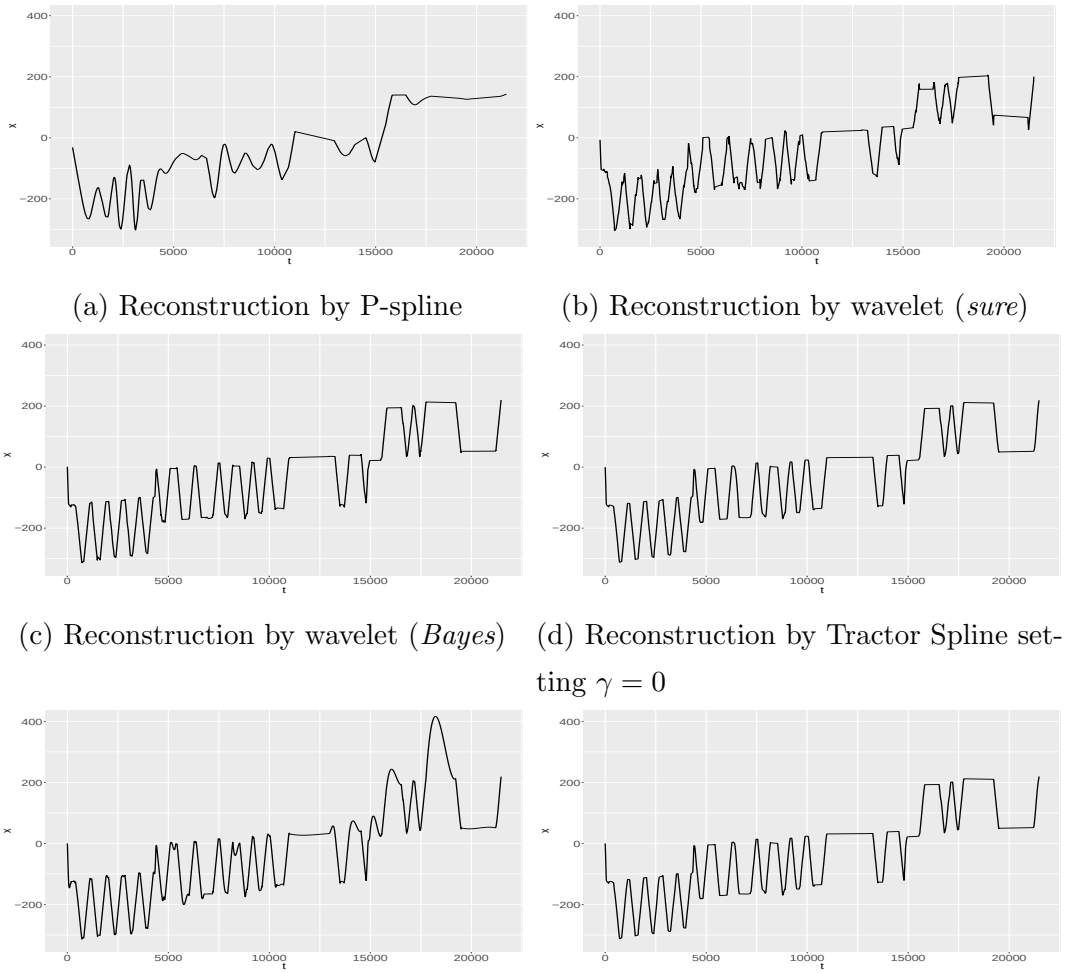
We treat $x$ and $y$ position separately and compare how velocity information in objective function (2.5) and the adjusted penalty term in equation (2.26) work in our model. All parameters in fitted Tractor Spline are automatically selected by cross validation in equation (2.35). Figure 2.9 and figure 2.10 compare the results of fitted methods on $x$ and $y$ axis. P-spline gives over fitting on $x$ axis reconstruction and not applicable on $y$ axis due to errors. Wavelet(sure) misses some key points at corners when a tractor tries to have a turn. Tractor Spline without adjusted penalty term presents less fitting at time gap knots, at which time marks keep increasing while position stays the same with velocity is 0. If we take the last knot $p_k$ before and the first knot $p_{k+1}$ after time gap, Hermite spline basis will use $y_k, v_k y_{k+1}$ and $v_{k+1}$ to build up a cubic spline, even though the velocity information is not useful. That is why we got a curve rather than a straight line. Wavelet(BayesThresh), Tractor Spline without velocity and proposed Tractor Spline give acceptable results.

Table 2.3 illustrates the MSE of all methods on both $x$ and $y$ axises. The proposed Tractor Spline returns the smallest errors among all methods.

The penalty function of the proposed Tractor Spline is

$$\lambda(t) = b\frac{\Delta t^3}{\Delta d^2} * \lambda_d + (1-b)\frac{\Delta t^3}{\Delta d^2} * \lambda_u, \text{where} \begin{cases} b = 1 & \text{if boom is working} \\ b = 0 & \text{if boom is not working} \end{cases}. \quad (2.48)$$

To tell the differences more clearly, we take $\log(\lambda(t))$ in our demonstration. Figure 2.11 indicates that at turning points and long time gap knots, the adjusted penalty




(a) Reconstruction by P-spline

(b) Reconstruction by wavelet (*sure*)

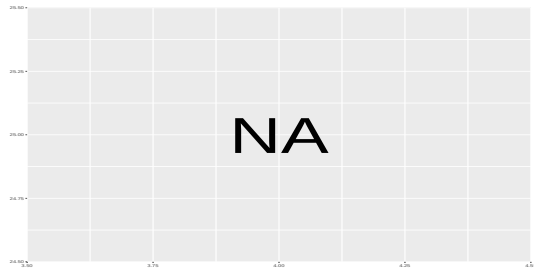
(c) Reconstruction by wavelet (*Bayes*)

(d) Reconstruction by Tractor Spline setting $\gamma = 0$

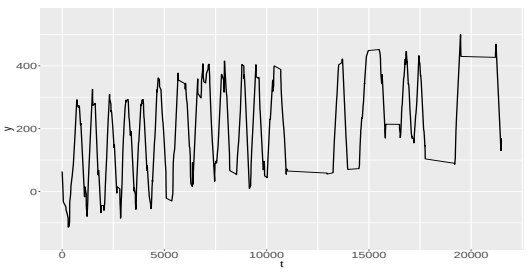
(e) Reconstruction by Tractor Spline setting with conventional penalty term

(f) Reconstruction by proposed Tractor Spline

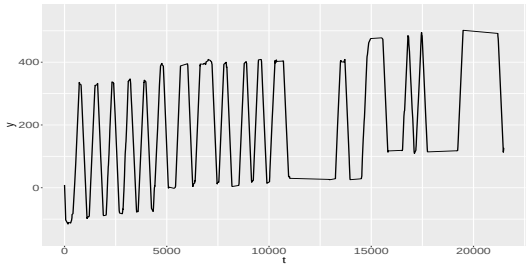
Figure 2.9: Fitted data points on $x$ axis. (a) Fitted by P-spline, which gives over-fitting on these points and misses some information. (b) Fitted by wavelet (*sure*) algorithm. At some turning points, it gives over-fitting. (c) Fitted by wavelet (*BayesThresh*) algorithm. It fits better than (*sure*) and the result is close to the proposed method. (d) Fitted by Tractor Spline without velocity information. The reconstruction is good to get the original trajectory. (e) Fitted by Tractor Spline without adjusted penalty term. It gives less fitting at boom-not-working points because of a large time gap. (f) Fitted by proposed method. It fits all data points in a good way.



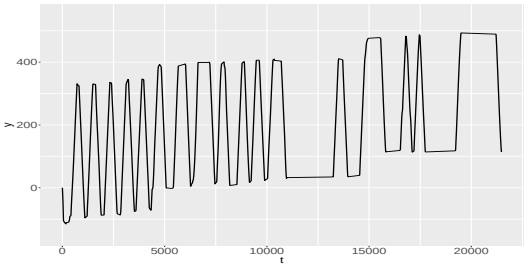

(a) Not available for P-spline



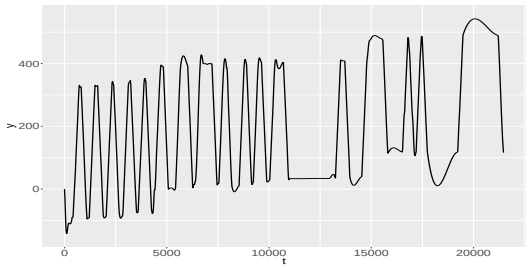

(b) Reconstruction by wavelet (*sure*)



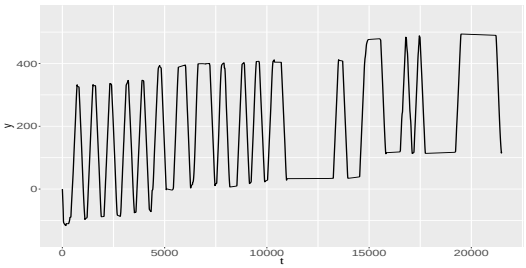

(c) Reconstruction by wavelet (*Bayes*)




(d) Reconstruction by Tractor Spline setting $\gamma = 0$




(e) Reconstruction by Tractor Spline setting with conventional penalty term




(f) Reconstruction by proposed Tractor Spline

Figure 2.10: Fitted data points on $y$ axis. (a) Fitted P-spline is not applicable on $y$ axis as the matrix is not invertible. (b) Fitted by wavelet (*sure*) algorithm. At some turning points, it gives over-fitting. (c) Fitted by wavelet (*BayesThresh*) algorithm is much better than wavelet (*sure*). (d) Fitted by Tractor Spline without velocity information. The reconstruction is good to get the original trajectory. (e) Fitted by Tractor Spline without adjusted penalty term. It gives less fitting at boom-not-working. (f) Fitted by proposed method. It fits all data points in a good way.

term will lead $\lambda(t)$ to large values, which forces the spline to be a straight line between two knots. It can be seen in figure (c) and (d) clearly. Histogram plots of $\log(\lambda(t))$ show that most of the penalty values are small, which allows the Tractor Spline goes as closer as it can to the observed points. Only a few of penalty values are large, so that Tractor Spline gives a straight line at tricky points.

### 2.5.2 2-Dimension Trajectory

In a 2-dimensional trajectory reconstruction, we use the same parameters $\lambda_d$, $\lambda_u$ and $\gamma$ on both $x$ and $y$ axises. The parameters will be found by cross-validation where it returns the smallest score of

$$\mathrm{CV} = \mathrm{CV}_x + \mathrm{CV}_y. \tag{2.49}$$

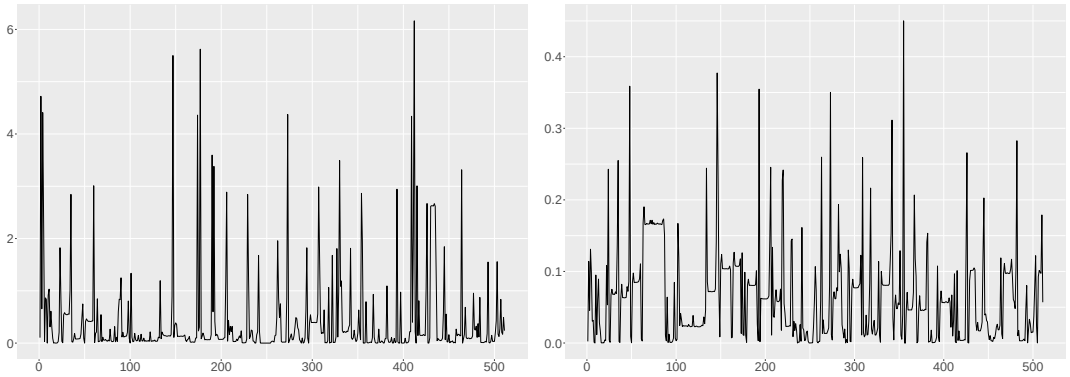
In a 2-dimensional reconstruction, $\Delta d$ in adjusted penalty term is the Euclidean distance $d(p_1, p_2) = \sqrt{(\Delta x)^2 + (\Delta y)^2}$ between two positions on 2D surface.

Similar results as that in 1 dimension reconstruction, the velocity information keeps trajectory in the right direction and the penalty term makes sure that the crazy curve will disappear between long time gap points. Figure 2.12 demonstrates the final overall 2D reconstruction.

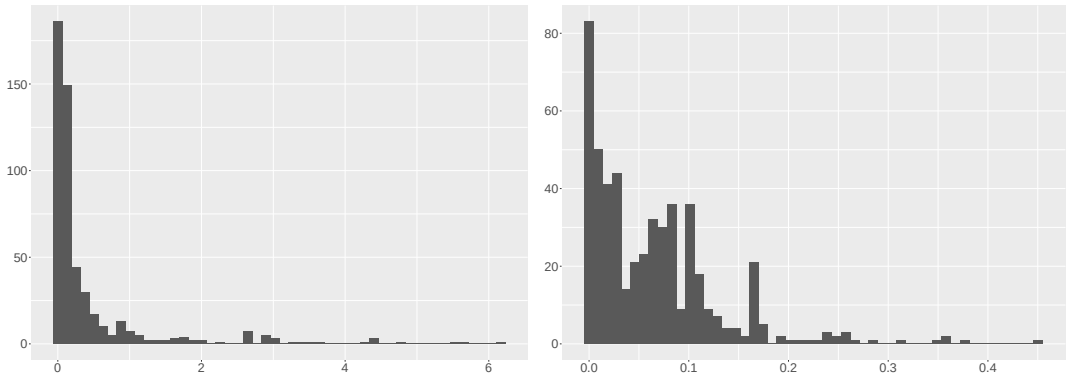
The penalty term of 2 dimension reconstruction will be the sum of each penalty on $x$ and $y$ axises. So the estimated penalty function is

$$\lambda(t) = \lambda(t)_x + \lambda(t)_y \tag{2.50}$$

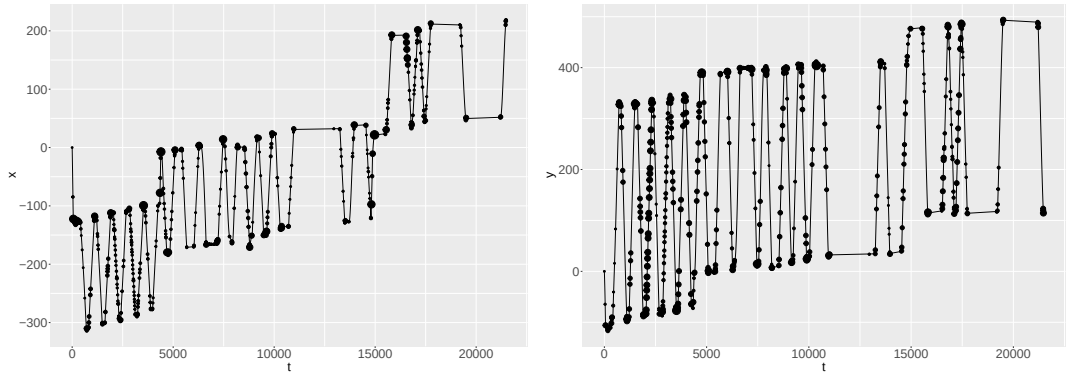
and presented in figure 2.13. As the similar results in 1 dimension reconstruction, most of large penalty values appear in long time gap knots and turning points. A histogram plot of penalty function shows that most of the values are small and only a couple of them are large.




(a) Distribution of penalty values on $x$ and $y$ axises.




(b) Histograms of penalty values on $x$ and $y$ axises.




(c) Reconstruction with penalty values on $x$ and $y$ axises separately.

Figure 2.11: Penalty function of Tractor Spline on $x$ and $y$ axises. The big black dots in plots (c) indicate large penalty values. It can be seen that most of large penalty values occur at turnings, where the tractor likely slows down and takes breaks.

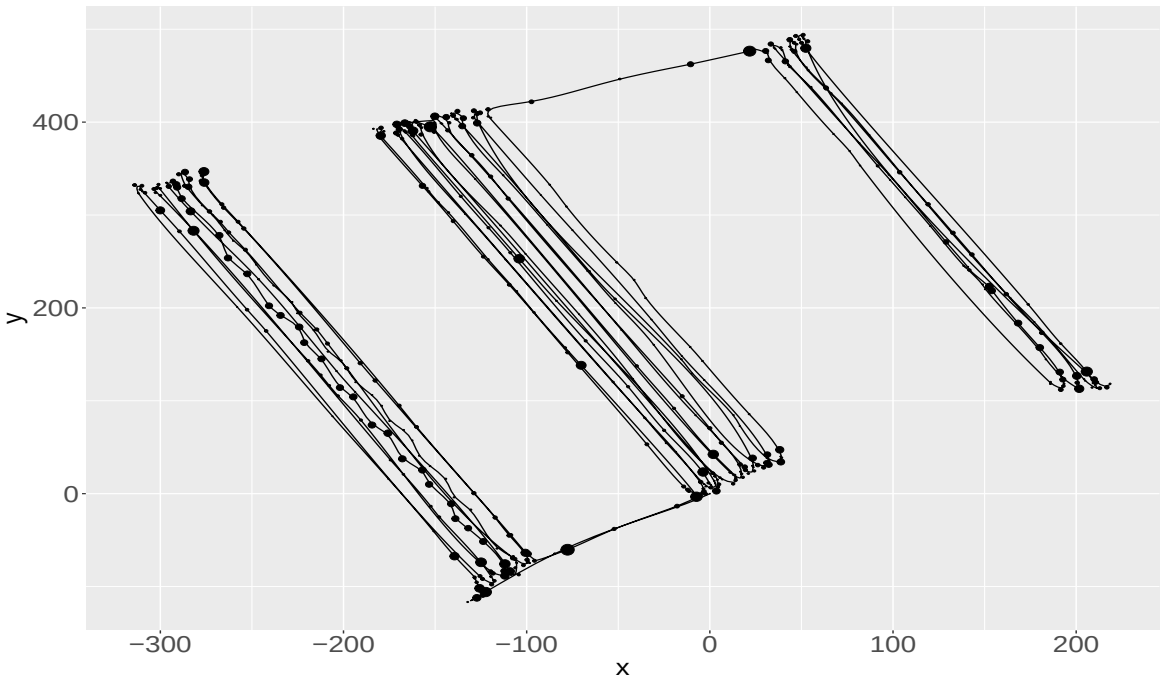



Figure 2.12: 2-Dimensional reconstruction with penalty function. Larger dots indicate bigger penalty values. The mean square errors (MSE) on $x$ and $y$ are 0.240734 and 0.478422 respectively. The mean distance error $\sqrt{(\hat{f}_x - x)^2 + (\hat{f}_y - y)^2}$ is 0.645830.

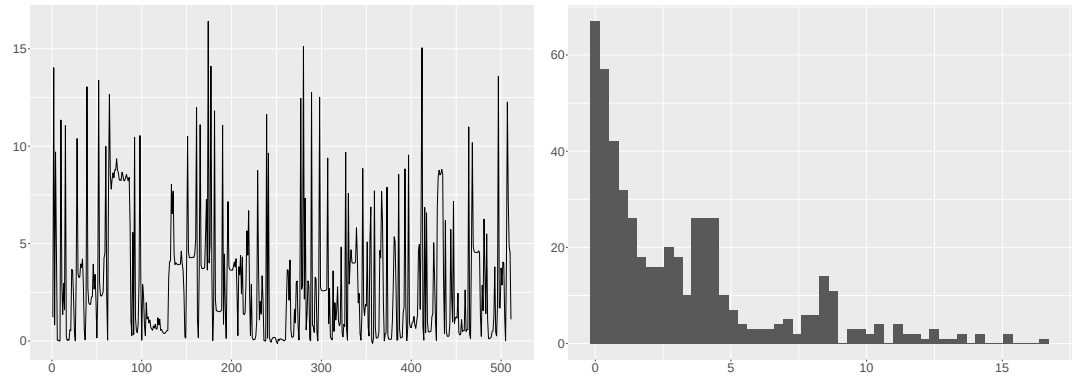



Figure 2.13: Penalty function of 2-Dimensional reconstruction.

## 2.6 Conclusion and Discussion

In this chapter, we propose a Tractor Spline model including the first derivative and adaptive penalty terms to reconstruct trajectory. This method performs better when we know **z** and **w** information than other methods. Additionally, the reconstruction of a Tractor Spline contains $4 \times (n-1)$ parameters if we have $n$ knots. By adding $2 \times (n-2)$ constraints, the original function and its first derivative are continuous on each interior knots, the degrees of freedom will be $4 \times (n-1) - 2 \times (n-2) = 2n$. Because there are $n$ location and $n$ velocity data, thus we don't need to specify more parameters or add more constrains on the model. Although the mean square error of Tractor Spline is not the smallest comparing with other methods, the true mean square error is the smallest most of the time. It means that the reconstruction is closer to the truth. In cross validation, we only focus on the errors of $f$ ignoring that in $f'$. So the reconstruction of $f'$ is not as smooth as that in $f$, which does not affect trajectory reconstruction. A drawback of Tractor Spline is the cost in finding local minimal parameters, where the cross validation algorithm returns a smaller score. So we optimized our code to make it runs as faster as it can.

# Chapter 3

# Tractor Spline and Gaussian Process Regression

## 3.1 Introduction

From Wikipedia, the free encyclopedia.

In probability theory and statistics, a Gaussian process is a particular kind of statistical model where observations occur in a continuous domain, e.g. time or space. In a Gaussian process, every point in some continuous input space is associated with a normally distributed random variable. Moreover, every finite collection of those random variables has a multivariate normal distribution, i.e. every finite linear combination of them is normally distributed. The distribution of a Gaussian process is the joint distribution of all those (infinitely many) random variables, and as such, it is a distribution over functions with a continuous domain, e.g. time or space.

Viewed as a machine learning algorithm, a Gaussian process uses lazy learning and a measure of the similarity between points (the kernel function) to predict the value for an unseen point from training data. The prediction is not just an estimate for that point, but also has uncertainty information it is a one-dimensional Gaussian distribution (which is the marginal distribution at that point).

For some kernel functions, matrix algebra can be used to calculate the predictions using the technique of Kriging. When a parameterized kernel is used, optimization software is typically used to fit a Gaussian process model.

The concept of Gaussian processes is named after Carl Friedrich Gauss because it is based on the notion of the Gaussian distribution (normal distribution). Gaussian processes can be seen as an infinite-dimensional generalization of multivariate normal

distributions.

Gaussian processes are useful in statistical modeling, benefiting from properties inherited from the normal. For example, if a random process is modeled as a Gaussian process, the distributions of various derived quantities can be obtained explicitly. Such quantities include the average value of the process over a range of times and the error in estimating the average using sample values at a small set of times.

### 3.1.1 Spline

In interpolation and curve fitting, piecewise linear approximation may not have the practical significance of cubic spline, or even higher order, approximation. These "broken lines" are neither very smooth nor very efficient approximation. Researchers can go to piecewise polynomial approximation with higher order pieces De Boor *et al.* (1978), which is called spline method. A spline is a numeric function that is piecewise-defined by polynomial functions, and which possesses a high degree of smoothness at the places where the polynomial pieces connect (known as knots) Judd (1998)Chen (2009). Suppose we are given observed data $t_1, t_2, \cdots, t_n$ on interval $[0, 1]$, satisfying $0 \leq t_1 < t_2 < \cdots < t_n \leq 1$. A piecewise polynomial function $f(t)$ can be obtained by dividing the interval into contiguous intervals $(t_1, t_2), \cdots, (t_{n-1}, t_n)$ and represented by a separate polynomial in each interval. For any continuous $f \in \mathbb{C}^{(m)}[0, 1]$, it can be represented in a linear combination of basis functions $h_m(t)$, just as every vector in a vector space can be represented as a linear combination of basis vectors. So we have

$$f(t) = \sum_{m=1}^{M} \beta_m h_m(t), \tag{3.1}$$

where $\beta_m$ are coefficients Ellis *et al.* (2009).

Suppose we were attempt to fit a model of $f(t)$ by least squares without any restrictions, the best fitting $\hat{f}(t)$ would go through every given data to reduce sum of squares to zero. Most of the time, the results are unsatisfactory as explanations of the given data. The roughness penalty approach is introduced to quantify the notion of a rapidly fluctuating curve and then to pose the estimation problem in a way that makes explicit the necessary compromise between varying rapidly fluctuation and slowly trend in curve estimation Green and Silverman (1993). By adding a penalty term $\int_0^1 (f^{(m)}(t))^2 dt$, the curve estimate $\hat{f}(t)$ exists and is unique over all spline functions $f(t)$ with $m - 1$ continuous derivatives fitting observed data in the space $\mathbb{C}^{(m)}[0, 1]$, and it can be found by

minimizing the following penalized mean residual sum squares

$$\text{MSE}(f, \lambda) = \frac{1}{n}\sum_{i=1}^{n}(f(t_i) - y_i)^2 + \lambda \int_0^1 (f^{(m)}(t))^2 dt, \tag{3.2}$$

where $\lambda$ is a fixed smoothing parameter, $(t_i, y_i)$, $i = 1, \cdots, n$ are observed data and $0 \leq t_1 < t_2 < \cdots < t_n \leq 1$. In equation (3.2), the smoothing parameter $\lambda$ controls the trade-off between over-fitting and bias. Smoothing spline provides a powerful tool to estimate nonparametric functions Hastie and Tibshirani (1990).

### 3.1.2 Gaussian Process Regression

A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution Rasmussen and Williams (2006). It is fully defined by its mean $m(t)$ and covariance $K(s, t)$ functions as

$$m(t) = \mathbb{E}[f(t)] \tag{3.3}$$

$$K(s, t) = \mathbb{E}[(f(s) - m(s))(f(t) - m(t))], \tag{3.4}$$

where $s$ and $t$ are two variables, and a function $f$ distributed as such is denoted in form of

$$f \sim GP(m(t), K(s, t)). \tag{3.5}$$

Usually the mean function is assumed to be zero everywhere.

Given a set of input variables $\mathbf{T}$ for function $f(t)$ and the output $\mathbf{y} = f(\mathbf{T}) + \varepsilon$ with independent identically distributed Gaussian noise $\varepsilon$ with variance $\sigma_n^2$, we can use the above definition to predict the value of the function $f_* = f(t_*)$ at a particular input $t_*$. As the noisy observations becoming

$$\text{cov}(y_p, y_q) = K(t_p, t_q) + \sigma_n^2 \delta_{pq} \tag{3.6}$$

where $\delta_{pq}$ is a Kronecker delta which is one iff $p = q$ and zero otherwise, the joint distribution of the observed outputs $\mathbf{y}$ and the estimated output $f_*$ according to prior is

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(\mathbf{T}, \mathbf{T}) + \sigma_n^2 I & K(\mathbf{T}, t_*) \\ K(t_*, \mathbf{T}) & K(t_*, t_*) \end{bmatrix}\right). \tag{3.7}$$

The posterior distribution over the predicted value is obtained by conditioning on the observed data

$$f_* \mid \mathbf{y}, \mathbf{T}, t_* \sim N(\bar{f}_*, \text{cov}(f_*)) \tag{3.8}$$

where

$$\bar{f}_* = \mathbb{E}[f_* \mid \mathbf{y}, \mathbf{T}, t_*] = K(t_*, \mathbf{T})[K(\mathbf{T}, \mathbf{T}) + \sigma_n^2 I]^{-1}\mathbf{y}, \tag{3.9}$$

$$\text{cov}(f_*) = K(t_*, t_*) - K(t_*, \mathbf{T})[K(\mathbf{T}, \mathbf{T}) + \sigma_n^2 I]^{-1}K(\mathbf{T}, t_*). \tag{3.10}$$

### 3.1.3 The Smoothing Spline as Bayes Estimates

It is possible to interpret the smoothing spline regression estimator as a Bayesian estimate when the mean function $r(.)$ is given an improper prior distribution. Berlinet and Thomas-Agnan (2011) Wahba (1990)

Consider the model

$$y_i = f(t_i) + \varepsilon_i, \tag{3.11}$$

where $i = 1, \ldots, n$, $\varepsilon_i$ are i.i.d. Gaussian distributed noise with variance $\sigma^2$. Assume $f \in \mathbb{H}^{(m)}[0, 1]$, where

$$\mathbb{H}^{(m)}[0,1] = \{f : f^{(\nu)} \text{absolutely continuous}, \nu = 0, \cdots, m-1, \int_0^1 (f^{(m)}(t)^2 dt < \infty\}. \tag{3.12}$$

A smoothing spline $\hat{f}_\lambda$ is the minimizer of objective function (3.2) Wang (1998). Equipped with an appropriate inner product

$$\langle f, g\rangle = \sum_{\nu=0}^{m-1} f^{(\nu)}(0)g^{(\nu)}(0) + \int_0^1 f^{(m)}g^{(m)}dt, \tag{3.13}$$

the space $\mathbb{H}^{(m)}[0, 1]$ becomes a reproducing kernel Hilbert space.

Let $\phi_\nu(t) = \frac{t^{\nu-1}}{(\nu-1)!}$ where $\nu = 1, \cdots, m$ and $R_1(s,t) = \int_0^1 \frac{(s-u)_+^{m-1}}{(m-1)!}\frac{(t-u)_+^{m-1}}{(m-1)!}du$. Denote $S = \{\phi_\nu(t_i)\}_{n\times m}$ where $i = 1, \cdots, n, \nu = 1, \cdots, m$ and $Q = \{R_1(t_i, t_j)\}_{n\times n}$ where $i = 1, \cdots, n, j = 1, \cdots, n$. Kimeldorf and Wahba (1971) and Kimeldorf and Wahba (1970) proved that $\hat{f}_\lambda$ has the form

$$\hat{f}(t) = \sum_{\nu=1}^{m} d_\nu\phi_\nu(t) + \sum_{i=1}^{n} c_i R_1(t, t_i). \tag{3.14}$$

By denoting $M = Q + n\lambda I$, Gu (2013) found that the coefficients will be given by

$$\mathbf{c} = (M^{-1} - M^{-1}S(S^\top M^{-1}S)^{-1}S^\top M^{-1})\mathbf{Y}, \tag{3.15}$$

$$\mathbf{d} = (S^\top M^{-1}S)^{-1}S^\top M^{-1}\mathbf{Y}. \tag{3.16}$$

## 3.2 A reproducing kernel on $\mathcal{C}^2_{p.w.}[0,1]$

The minimizer $f(x)$ of

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - f(x_i))^2 + \frac{\gamma}{n}\sum_{i=1}^{n}(v_i - f'(x_i))^2 + \lambda\int_0^1 f''^2 dx \tag{3.17}$$

in the space $\mathcal{C}^2_{p.w.}[0,1] = \{f : f, f' \text{ are continuous and } f'' \text{ is piecewise continuous on } [0,1]\}$ is a tractor spline. Equipped with an appropriate inner product

$$(f,g) = f(0)g(0) + f'(0)g'(0) + \int_0^1 f''g''dx, \tag{3.18}$$

the space $\mathcal{C}^2_{p.w.}[0,1]$ is made a reproducing kernel Hilbert space. In fact, the representer $R_x(\cdot)$ is

$$R_x(y) = 1 + xy + \int_0^1 (x-u)_+(y-u)_+ du. \tag{3.19}$$

It can be seen that $R_x(0) = 1, R'_x(0) = x$, and $R''_x(y) = (x-y)_+$.

The two terms of the reproducing kernel $R(x,y) = R_x(y) = R_0(x,y) + R_1(x,y)$, where

$$R_0(x,y) = 1 + xy \tag{3.20}$$

$$R_1(x,y) = \int_0^1 (x-u)_+(y-u)_+ du \tag{3.21}$$

are both non-negative definite themselves.

**Theorem 4.** *If the reproducing kernel $R$ of a space $\mathcal{H}$ on domain $X$ can be decomposed into $R = R_0 + R_1$, where $R_0$ and $R_1$ are both non-negative definite, $R_0(x,\cdot), R_1(x,\cdot) \in \mathcal{H}$, $\forall x \in X$, and $(R_0(x,), R_1(y,)) = 0$, $\forall x, y \in X$, then the spaces $\mathcal{H}_0$ and $\mathcal{H}_1$ corresponding respectively to $R_0$ and $R_1$ form a tensor sum decomposition of $\mathcal{H}$. Conversely, if $R_0$ and $R_1$ are both non-negative definite and $\mathcal{H}_0 \cap \mathcal{H}_1 = \{0\}$, then $\mathcal{H} = \mathcal{H}_0 \bigoplus \mathcal{H}_1$ has a reproducing kernel $R = R_0 + R_1$.*

According to Theorem 1, $R_0$ can correspond the space of polynomials $\mathcal{H}_0 = \{f : f'' = 0\}$ with an inner product $(f,g)_0 = f(0)g(0) + f'(0)g'(0)$, and $R_1$ corresponds the orthogonal complement of $\mathcal{H}_0$

$$\mathcal{H}_1 = \{f : f(0) = 0, f'(0) = 0, \int_0^1 f''^2 dx < \infty\}$$

with inner product $(f,g)_1 = \int_0^1 f''g''dx$. Thus, $\mathcal{H}_0$ and $\mathcal{H}_1$ are two subspaces of the $\mathcal{C}^2_{p.w.}[0,1]$, and the reproducing kernel is $R_x(\cdot) = R_0(x,\cdot) + R_1(x,\cdot)$.

Define a new notation $\dot{R}(x,y) = \frac{\partial R}{\partial x}(x,y) = \frac{\partial R_0}{\partial x}(x,y) + \frac{\partial R_1}{\partial x}(x,y) = y + \int_0^x (y-u)_+ du$. Obviously $\dot{R}_x(y) \in \mathcal{C}^2_{p.w.}[0,1]$. Additionally, we have $\dot{R}_x(0) = 0, \dot{R}'_x(0) = \frac{\partial \dot{R}_x}{\partial y}(0) = 1$, and $\dot{R}''_x(y) = \begin{cases} 0 & x \le y \\ 1 & x > y \end{cases}$. Then, for any $f \in \mathcal{C}^2_{p.w.}[0,1]$, it gives us

$$(\dot{R}_x, f) = \dot{R}_x(0)f(0) + \dot{R}'_x(0)f'(0) + \int_0^1 \dot{R}''_x f'' du = f'(0) + \int_0^y f'' du = f'(y).$$

It can be seen that the first term $\dot{R}_0 = y \in \mathcal{H}_0$, and the space spanned by the second term $\dot{R}_1 = \int_0^x (y-u)_+ du$, denoted as $\dot{\mathcal{H}}$, is not in $\mathcal{H}_1$, but $\dot{\mathcal{H}} \cap \mathcal{H}_1 \neq \emptyset$. Then we have a new space $\mathcal{H}_* = \dot{\mathcal{H}} \cup \mathcal{H}_1$. Thus the two new sub spaces in $\mathcal{C}^2_{p.w.}[0,1]$ are $\mathcal{H}_0$ and $\mathcal{H}_*$.

## 3.3 Computation of Polynomial Smoothing Splines

Given the sample points $x_j, j = 1, \cdots, n$ in equation (3.17) and noting that the space

$$\mathcal{A} = \{f : f = \sum_{j=1}^n \alpha_j R_1(x_j, \cdot) + \sum_{j=1}^n \beta_j \dot{R}_1(x_j, \cdot)\} \tag{3.22}$$

is a closed linear subspace of $\mathcal{H}_*$. Then $f \in \mathcal{C}^2_{p.w.}[0,1]$ can be written as

$$f(x) = d_1 + d_2 x + \sum_{j=1}^n c_j R_1(x_j, x) + \sum_{i=j}^n b_j \dot{R}_1(x_j, \cdot) + \rho(x) \tag{3.23}$$

where $\mathbf{d}, \mathbf{c}$ and $\mathbf{b}$ are coefficients, and $\rho(x) \in \mathcal{H}_* \ominus \mathcal{A}$.

The equation (3.17) can be written as

$$\begin{aligned} &\frac{1}{n}\sum_{i=1}^n \left( Y_i - d_1 - d_2 x - \sum_{j=1}^n c_j R_1(x_j, x_i) - \sum_{j=1}^n b_j \dot{R}_1(x_j, x_i) - \rho(x_i) \right)^2 \\ &\frac{\gamma}{n}\sum_{i=1}^n \left( V_i - d_2 - \sum_{j=1}^n c_j R'_1(x_j, x_i) - \sum_{j=1}^n b_j \dot{R}'_1(x_j, x_i) - \rho'(x_i) \right)^2 \\ &+\lambda \int_0^1 \left( \sum_{j=1}^n c_j R''_1(x_j, x) + \sum_{j=1}^n c_j \dot{R}''_1(x_j, x) + \rho''(x) \right)^2 dx \end{aligned}$$

By orthogonality, $\rho(x_i) = (R_1(x_i, \cdot), \rho) = 0$, $\rho'(x_i) = (\dot{R}_1(x_i, \cdot), \rho') = 0$, $i = 1, \cdots, n$. Denoting by

$$\begin{aligned} S = \{S_{ij}\}_{n\times 2} = \begin{bmatrix} 1 & x_i \end{bmatrix}, \quad Q = \{Q_{ij}\}_{n\times n} = R_1(x_j, x_i), \quad P = \{P_{ij}\}_{n\times n} = \dot{R}_1(x_j, x_i), \\ S' = \{S'_{ij}\}_{n\times 2} = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad Q' = \{Q'_{ij}\}_{n\times n} = R'_1(x_j, x_i), \quad P' = \{P'_{ij}\}_{n\times n} = \dot{R}'_1(x_j, x_i). \end{aligned}$$

and noting that $\int_0^1 R_1''(x_i,x)R_1''(x_j,x)dx = R_1(x_i,x_j)$, $\int_0^1 R_1''(x_i,x)\dot{R}_1''(x_j,x)dx = \int_0^v (x_i - x)dx = \dot{R}_1(x_j,x_i)$, and $\int_0^1 \dot{R}_1''(x_i,x)\dot{R}_1''(x_j,x)dx = \int_0^v 1dx = \dot{R}_1'(x_i,x_j)$, where $v = \min(x_i,x_j)$, the above equation can be written as

$$(\mathbf{Y} - S\mathbf{d} - Q\mathbf{c} - P\mathbf{b})^\top(\mathbf{Y} - S\mathbf{d} - Q\mathbf{c} - P\mathbf{b}) + \gamma(\mathbf{V} - S'\mathbf{d} - Q'\mathbf{c} - P'\mathbf{b})^\top(\mathbf{V} - S'\mathbf{d} - Q'\mathbf{c} - P'\mathbf{b}) + n\lambda(\mathbf{c}^\top Q\mathbf{c} + 2\mathbf{c}^\top P\mathbf{b} + \mathbf{b}^\top P'\mathbf{b}) + n\lambda(\rho,\rho). \tag{3.24}$$

Note that $\rho$ only appears in the third term in (3.24), which is minimised at $\rho = 0$. Hence, a polynomial smoothing spline resides in the space $\mathcal{H}_0 \oplus \mathcal{A}$ of finite dimension. Then the solution to (3.17) could be computed via minimization of the first two terms in (3.24) with respect to $\mathbf{d}$, $\mathbf{c}$ and $\mathbf{b}$.

## 3.4 Polynomial Smoothing Splines as Bayes Estimates

Now in the model $Y = f(x) + \epsilon$ and $V = f'(x) + \frac{\epsilon}{\gamma}$ where $\epsilon \sim N(0,\sigma^2)$, according to equation (3.23), for $f(x) \in \mathcal{C}^2_{p.w.}[0,1]$ and $x \in X$, we have

$$f(x) = (d_1 + d_2 x) + \sum_{i=1}^n c_i R_1(x_i,x) + \sum_{i=1}^n b_i \dot{R}_1(x_i,x). \tag{3.25}$$

The covariance functions for $Y, V$ and $f, f'$ are

$$\begin{aligned}
\mathbb{E}(f(x)f(y)) &= \tau^2 R_0(x,y) + \beta R_1(x,y) & \mathbb{E}(f(x)f'(y)) &= \tau^2 R_0'(x,y) + \beta R_1'(x,y)\\
\mathbb{E}(f'(x)f(y)) &= \tau^2 \dot{R}_0(x,y) + \beta \dot{R}_1(x,y) & \mathbb{E}(f'(x)f'(y)) &= \tau^2 \dot{R}_0'(x,y) + \beta \dot{R}_1'(x,y)\\
\mathbb{E}(y_i,y_j) &= \tau^2 R_0(x_i,x_j) + \beta R_1(x_i,x_j) + \sigma^2\delta_{ij} & \mathbb{E}(v_i,v_j) &= \tau^2 \dot{R}_0'(x_i,x_j) + \beta \dot{R}_1'(x_i,x_j) + \frac{\sigma^2}{\gamma}\delta_{ij}\\
\mathbb{E}(v_i,y_j) &= \tau^2 \dot{R}_0(x_i,x_j) + \beta \dot{R}_1(x_i,x_j) & \mathbb{E}(y_i,v_j) &= \tau^2 R_0'(x_i,x_j) + \beta R_1'(x_i,x_j)\\
\mathbb{E}(y_i,f(x)) &= \tau^2 R_0(x_i,x) + \beta R_1(x_i,x) & \mathbb{E}(y_i,f'(x)) &= \tau^2 R_0'(x_i,x) + \beta R_1'(x_i,x)\\
\mathbb{E}(v_i,f(x)) &= \tau^2 \dot{R}_0(x_i,x) + \beta \dot{R}_1(x_i,x) & \mathbb{E}(v_i,f'(x)) &= \tau^2 \dot{R}_0'(x_i,x) + \beta \dot{R}_1'(x_i,x)
\end{aligned}$$

where $R(x,y)$ is taken from (3.19).

Observing $Y_i \sim N(f(x_i),\sigma^2)$ and $V_i \sim N(f(x_i),\frac{\sigma^2}{\gamma})$, the joint distribution of $Y, V$ and $f(x)$ is normal with mean zero and a covariance matrix can be found. The posterior

mean of $f(x)$ is

$$\begin{aligned}
\mathbb{E}(f \mid Y, V) &= \begin{bmatrix}\operatorname{cov}(Y,f) & \operatorname{cov}(f,V)\end{bmatrix} \begin{bmatrix}\operatorname{var}(Y) & \operatorname{cov}(Y,V) \\ \operatorname{cov}(V,Y) & \operatorname{var}(V)\end{bmatrix}^{-1} \begin{bmatrix}Y \\ V\end{bmatrix} \\
&= \begin{bmatrix}\tau^2\phi^\top S^\top + \beta\xi^\top & \tau^2\phi^\top S'^\top + \beta\psi^\top\end{bmatrix} \begin{bmatrix}\tau^2 SS^\top + \beta Q + \sigma^2 I & \tau^2 SS'^\top + \beta P \\ \tau^2 S'S^\top + \beta Q' & \tau^2 S'S'^\top + \beta P' + \frac{\sigma^2}{\gamma} I\end{bmatrix}^{-1} \begin{bmatrix}Y \\ V\end{bmatrix} \\
&= \begin{bmatrix}\rho\phi^\top S^\top + \xi^\top & \rho\phi^\top S'^\top + \psi^\top\end{bmatrix} \begin{bmatrix}\rho SS^\top + Q + n\lambda I & \rho SS'^\top + P \\ \rho S'S^\top + Q' & \rho S'S'^\top + P' + \frac{n\lambda}{\gamma} I\end{bmatrix}^{-1} \begin{bmatrix}Y \\ V\end{bmatrix} \\
&= (\phi\top\rho \begin{bmatrix}S \\ S'\end{bmatrix}^\top + \begin{bmatrix}\xi^\top & \psi^\top\end{bmatrix}) \left(\rho \begin{bmatrix}S \\ S'\end{bmatrix}^\top \begin{bmatrix}S \\ S'\end{bmatrix} + \begin{bmatrix}Q + n\lambda I & P \\ Q' & P' + \frac{n\lambda}{\gamma} I\end{bmatrix}\right)^{-1} \begin{bmatrix}Y \\ V\end{bmatrix} \\
&\triangleq \phi\top\rho T^\top \left(\rho T^\top T + M\right)^{-1} \begin{bmatrix}Y \\ V\end{bmatrix} + \begin{bmatrix}\xi^\top & \psi^\top\end{bmatrix} \left(\rho T^\top T + M\right)^{-1} \begin{bmatrix}Y \\ V\end{bmatrix}
\end{aligned} \tag{3.26}$$

where $\phi$ is $2 \times 1$ matrix with entry 1 and $x$, $\xi$ is $n \times 1$ matrix with $i$th entry $R(x_i, x)$ and $\psi$ is $n \times 1$ matrix with $i$th entry $\dot{R}(x_i, x)$, $\rho = \tau^2/\beta$ and $n\lambda = \sigma^2/\beta$.

**Lemma 3.** *Suppose $M$ is symmetric and nonsingular and $S$ is of full column rank.*

$$\begin{aligned}
&\lim_{\rho\to\infty} (\rho TT^\top + M)^{-1} = M^{-1} - M^{-1}T(T^\top M^{-1}T)^{-1}T^\top M^{-1}, \\
&\lim_{\rho\to\infty} \rho T^\top (\rho TT^\top + M)^{-1} = (T^\top M^{-1}T)^{-1}T^\top M^{-1}.
\end{aligned}$$

Setting $\rho \to \infty$ in equation (3.26) and applying Lemma 3, the posterior mean $\mathbb{E}(f(x) \mid \mathbf{Y}, \mathbf{V})$ is of the form $f = \phi^\top \mathbf{d} + \xi^\top \mathbf{c} + \psi^\top \mathbf{b}$, with the coefficients given by

$$\begin{aligned}
\mathbf{d} &= (T^\top M^{-1}T)^{-1}T^\top M^{-1} \begin{bmatrix}Y \\ V\end{bmatrix}, \\
\begin{bmatrix}\mathbf{c} \\ \mathbf{b}\end{bmatrix} &= (M^{-1} - M^{-1}T(T^\top M^{-1}T)^{-1}T^\top M^{-1}) \begin{bmatrix}Y \\ V\end{bmatrix},
\end{aligned}$$

where $T = \begin{bmatrix}S \\ S'\end{bmatrix}$ and $M = \begin{bmatrix}Q + n\lambda I & P \\ Q' & P' + \frac{n\lambda}{\gamma} I\end{bmatrix}$.

It is easy to verify that $d, c, b$ are the solutions to

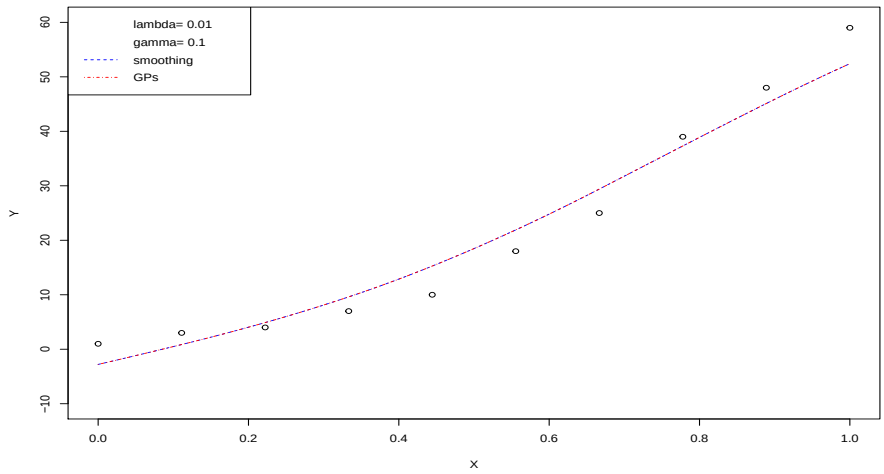
$$\begin{cases}
S^\top(Sd + Qc + Pb - Y) + \gamma S'^\top(S'd + P^\top c + S'^\top P'b - V) = 0, \\
Q(Sd + (Q + n\lambda I)c + Pb - Y) + P(\gamma S'd + \gamma P^\top c + (\gamma P' + n\lambda I)b - \gamma V) = 0, \\
P^\top(Sd + (Q + n\lambda I)c + Pb - Y) + P'(\gamma S'b + P^\top c + (\gamma P' + n\lambda I)b - \gamma V) = 0.
\end{cases}$$

## 3.5 Numeric Simulation of Smoothing Spline and GPR

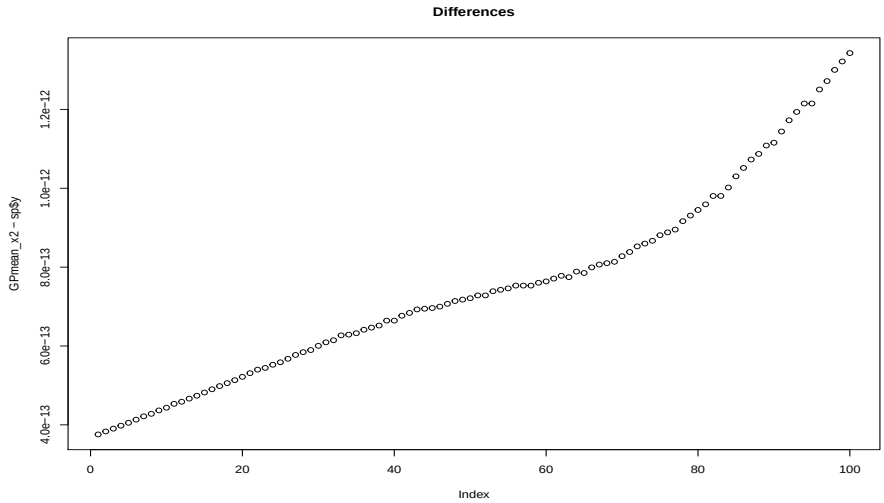
Given $\mathbf{X}$ a length-10 sequence from 0 to 1,

$$\mathbf{Y} = [1, 3, 4, 7, 10, 18, 25, 39, 48, 59],$$

$v_i = \begin{cases} \frac{y_{i+1}-y_i}{x_{i+1}-x_i} & \text{if } 1 \leq i \leq 9 \\ 0 & \text{if } i = 10 \end{cases}$. A simulated result is in figure 1.




(a)




(b)

Figure 3.1: (a) Comparing two methods under the same parameters $\lambda = 0.01$ and $\gamma = 0.1$. In this graph, the blue line is reconstruction from tractor spline, the red line is the mean of Gaussian Process, which is the posterior $\mathbb{E}(f(x) \mid \mathbf{Y}, \mathbf{V})$. (b) The differences between two methods under the same parameters.

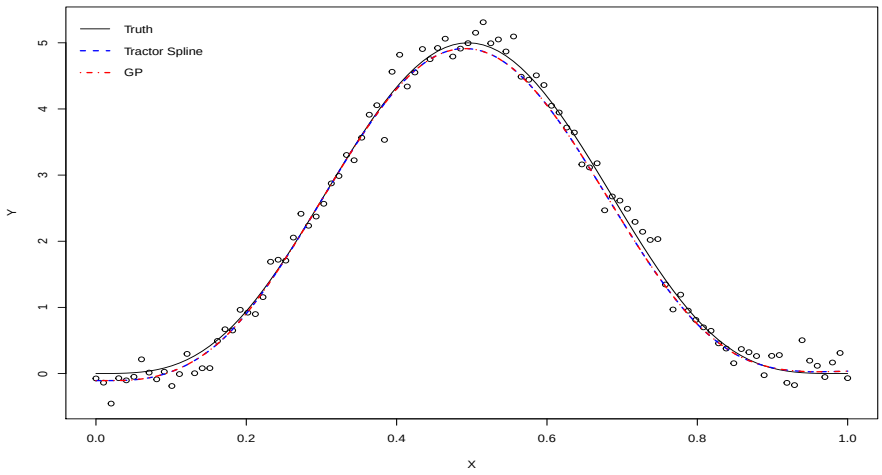



Figure 3.2: Best fitting simulation by TS and GPR.

# Chapter 4

# A Brief Overview of On-line State and Parameter Estimation

## 4.1 Introduction

As the development of technology of science and real life, the "big data" challenge becomes ubiquitous. Classical methods, such as Markov Chain Monte Carlo (MCMC), are normally suitable and good at handling a batch of data forecasting and analyzing. However, for big data and instant updating data stream, more robust and efficient methods are required.

Alternative approaches, such as Sequential Monte Carlo, for on-line updating and estimating are well studied in scientific literature and quite prevalent in academic research in the last decades. When it embraced with state space model, which is a very popular class of time series models that have found numerous of applications in fields as diverse as statistics, ecology, econometrics, engineering and environmental sciences Cappé *et al.* (2009) Arnaud Doucet (2011) Robert J Elliott (1995) Cargnoni *et al.* (1997), it allows us to establish complex linear and nonlinear Bayesian estimations in time series patterns Vieira and Wilkinson (2016).

### State Space Model

State space models are the natural form of system models relying on the general concept of state. If we describe a system as an operator mapping from the space of inputs to the space of outputs, then we may need the entire input-output history of the system together with the planned input in order to compute the future output

values Hangos *et al.* (2006). In an alternative way, by using new information at time $t$ containing all the past information up to the current state and initial conditions to get the current output is possible, that is known as a sequential method. A genetic state space model consists of two sets of equations: state equation and output equation. The state equation describes the evolution of the true input and state variables sequentially as a function and passes the variable one after one, generally, with some noises. The output equation catches the input values and interprets it out by an algebraic equation. A general state space model looks like the following form

$$\text{State equation } x_t = G_t(x_{t-1}) + w_t, \tag{4.1}$$

$$\text{Output equation } y_t = F_t(x_t) + \epsilon_t \tag{4.2}$$

with an initial state $x_0$, where $\epsilon_t$ and $w_t$ are noises passing through the process $G_t$ and $F_t$. $x_t$ are true status variables and $y_t$ are output values. Many researchers have been interested in this model and its application because of its good property. It can be used to model univariate or multivariate time series, also in the presence of non-stationarity, structural changes, and irregular patterns Petris *et al.* (2009).

The most simple and important system is given by Gaussian linear state space models, also known by dynamic linear models (DLM), which defines a very general class of non-stationary time series models. Firstly, the model is linear, which means $G_t$ and $F_t$ are linear processes and satisfying linearity property. Secondly, the it is specified by a normal prior distribution for the $p$-dimensional state vector at initial state $t = 0$,

$$x_0 \sim N_p(m_0, C_0)$$

and two independent zero mean normal distributed noises $\epsilon_t \sim N_p(0, V_t)$ and $w_t \sim N_p(0, W_t)$ Petris *et al.* (2009). The well known Kalman Filter is a particular algorithm that is used to solve state space models in the linear case. This was first derived by Kalman Kalman *et al.* (1960).

The assumption Markovian keeps the current state $x_t$ only depending on the previous one step $x_{t-1}$ and the observed $y_t$ depending on $x_t$. A state-space is shown in the diagram below:

$$\begin{array}{ccccccccccc}
\cdots & \rightarrow & x_{t-1} & \rightarrow & x_t & \rightarrow & x_{t+1} & \rightarrow & \cdots & & \text{truth} \\
\cdots & & \downarrow & & \downarrow & & \downarrow & & \cdots & & \\
\cdots & & y_{t-1} & & y_t & & y_{t+1} & & \cdots & & \text{observation}
\end{array}$$

In applications, the process function $G_t$ and $F_t$ contain unknown parameters to be estimated De Jong (1988) and the target is to estimate the true states on sequential observations $y_t, \cdots, y_t$. Then it becomes to estimate a joint density of $p(x_{1:t}, \theta \mid y_{1:t})$, where $x_{1:t} = \{x_1, x_2, \cdots, x_t\}$ are the hidden states and $y_{1:t} = \{y_1, y_2, \cdots, y_t\}$ are the observed outcomes and $\theta$ is a set of unknown parameters.

### Contents

In this chapter, I will give a brief review on existing methods for sequential state and parameter inference. In section 4.2, I'm introducing some concepts and popular algorithms on estimating states sequentially. These algorithms are the fundamental of advanced methods. In section 4.3, we will have a look at on-line algorithms that can estimate both unknown parameters and states simultaneously. In section 4.4, finally I will analyze these methods numerically with simulated data.

## 4.2 Filtering Problem and Estimation

### 4.2.1 Sequential Monte Carlo Method

The use of Monte Carlo methods for filtering can be traced back to the pioneering contributions of Handschin and Mayne (1969) Handschin and Mayne (1969) and Handschin(1970) Handschin (1970). These researchers tried to use an importance sampling paradigm to approximate the target distributions and. Later on, an importance sampling algorithms were implemented sequentially in the filtering context. This algorithm is called sequential importance sampling, often abbreviated SIS, and has been known since the early 1970s. Limited by the power of computers and suffering from sample impoverishment or weight degeneracy, the SIS didn't develop very well until 1993. Gordon used this a technique based on sampling and importance sampling methods to find the best state estimation Gordon *et al.* (1993). A particle filter algorithm was proposed to allow rejuvenation of the set of samples by duplicating the samples with high importance weights and, on the contrary, removing samples with low weights Cappé *et al.* (2009). Since then, sequential Monte Carlo (SMC) methods have been applied in many different fields including but not limited to computer vision, signal processing, control, econometrics, finance, robotics, and statistics Arnaud Doucet (2011) Ristic *et al.* (2004).

In the state space model, a generic particle filter estimates the posterior distribution

of the hidden states using the observation measurement process. The filtering problem is to estimate sequentially the values of the hidden states $x_t$ given the values of the observation process $y_{1:t}$ at any time $t$. In another word, it is to find the value of $p(x_t \mid y_{1:t})$. The process is divided into two steps: prediction and updating. In the prediction step, the assumption of Markov Chain is the current status $x_t$ only depends on the previous one $x_{t-1}$. Then we can calculate the probability of $x_t$ by

$$\begin{aligned} p(x_t \mid y_{1:t-1}) &= \int p(x_t, x_{k-1} \mid y_{1:t-1}) dx_{t-1} \\ &= \int p(x_t \mid x_{t-1}, y_{1:t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1} \\ &= \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1}. \end{aligned}$$

In the updating step, once $p(x_t \mid y_{1:t-1})$ is known, $p(x_t \mid y_{1:t})$ can be found by

$$\begin{aligned} p(x_t \mid y_{1:t}) =& \frac{p(y_t \mid x_t, y_{1:t-1}) p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})} \\ =& \frac{p(y_t \mid x_t) p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}, \end{aligned}$$

where the normalization $p(y_t \mid y_{1:t-1}) = \int p(y_t \mid x_t) p(x_t \mid y_{1:t-1}) dx_t$ Arulampalam *et al.* (2002).

Imagine that the state space is partitioned as many parts, in which the particles are filled according to some probability measure. The higher probability, the denser the particles are concentrated. Suppose the particles $x_k^{(1)}, \ldots, x_k^{(N)}$ at time $k$ are drawn from the target probability density function $p(x)$, then these particles are used to estimate the expectation and variance of $f(x)$ by

$$\begin{aligned} \mathrm{E}(f(x)) &= \int_a^b f(x) p(x) dx \\ \mathrm{Var}(f(x)) &= \mathrm{E}\left(f(x) - \mathrm{E}(f(x))\right)^2 p(x) dx. \end{aligned}$$

Back to our target, the posterior distribution or density is empirically represented by a weighted sum of samples $x_k^{(1)}, \ldots, x_k^{(N)}$

$$\hat{p}(x_k \mid y_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta(x_k - x_k^{(i)}) \approx p(x_k \mid y_{1:t}), \tag{4.3}$$

where $f(x) = \delta(x_k - x_k^{(i)})$ is Dirac delta function. Hence, a continuous variable is approximated by a discrete one with a random support. When $N$ is sufficiently large,

$\hat{p}(x_k \mid y_{1:t})$ was treated by particle filter as the true posterior $p(x_k \mid y_{1:t})$. By this approximation, the filtering problem becomes to get the expectation of current status

$$\begin{aligned}
\mathrm{E}(f(x_k)) &\approx \int f(x_k)\hat{p}(x_k \mid y_{1:t})dx_k \\
&= \frac{1}{N}\sum_{i=1}^{N}\int f(x_k)\delta(x_k - x_k^{(i)})dx_k \\
&= \frac{1}{N}\sum_{i=1}^{N} f(x_k^{(i)}).
\end{aligned}$$

The expectation is the mean of the status of all particles $x_k^{(1)}, \ldots, x_k^{(N)}$.

However, the posterior distribution is unknown and impossible to sample from the true posterior. To solve this issue, some sampling methods are introduced in the following sections.

### 4.2.2 Importance sampling

It is common to sample from an easy-to-implement distribution, the so-called proposal distribution $q(x \mid y)$, hence

$$\begin{aligned}
\mathrm{E}(f(x)) &= \int f(x_t)\frac{p(x_t \mid y_{1:t})}{q(x_t \mid y_{1:t})}q(x_t \mid y_{1:t})dx_x \\
&= \int f(x_t)\frac{p(x_t)p(y_{1:t} \mid x_t)}{p(y_{1:t})q(x_t \mid y_{1:t})}q(x_t \mid y_{1:t})dx_x \\
&= \int f(x_t)\frac{W_t(x_t)}{p(y_{1:t})}q(x_t \mid y_{1:t})dx_x,
\end{aligned}$$

where $W_t(x_t) = \frac{p(x_t)p(y_{1:t}|x_t)}{q(x_t|y_{1:t})} \propto \frac{p(x_t|y_{1:t})}{q(x_t|y_{1:t})}$. Because $p(y_{1:t}) = \int p(y_{1:t} \mid x_t)p(x_t)dx_t$, so the above equation can be rewritten as

$$\begin{aligned}
\mathrm{E}(f(x)) &= \frac{1}{p(y_{1:t})}\int f(x_t)W_t(x_t)q(x_t \mid y_{1:t})dx_t \\
&= \frac{\int f(x_t)W_t(x_t)q(x_t \mid y_{1:t})dx_t}{\int p(y_{1:t} \mid x_t)p(x_t)dx_t} \\
&= \frac{\int f(x_t)W_t(x_t)q(x_t \mid y_{1:t})dx_t}{\int W_t(x_t)q(x_t \mid y_{1:t})dx_t} \\
&= \frac{E_{q(x_t|y_{1:t})}[W_t(x_t)f(x_t)]}{E_{q(x_t|y_{1:t})}[W_t(x_t)]}.
\end{aligned}$$

To solve the above equation, we can use Monte Carlo method by drawing samples $\{x_t^{(i)}\}$ from $q(x_t \mid y_{1:t})$ and get their expectation, which is approximated by

$$\begin{aligned} \mathrm{E}(f(x_t)) &\approx \frac{\frac{1}{N}\sum_{i=1}^{N} W_t(x_t^{(i)}) f(x_t^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} W_t(x_t^{(i)})} \\ &= \sum_{i=1}^{N} \tilde{W}_t(x_t^{(i)}) f(x_t^{(i)}), \end{aligned} \tag{4.4}$$

where $\tilde{W}_t(x_t^{(i)}) = \frac{W_t(x_t^{(i)})}{\sum_{i=1}^{N} W_t(x_t^{(i)})}$ is factorized weight. Each particles has its own weighted value, so the expectation is a weighted mean. However, the drawback of this method is that the computation is quite expensive. A smarter way is to update $W_t^{(i)}$ recursively. Suppose the proposal distribution

$$q(x_{0:t} \mid y_{1:t}) = q(x_{0:t-1} \mid y_{1:t-1}) q(x_t \mid x_{0:t-1}, y_{1:t}),$$

then the recursive form of the posterior distribution is

$$\begin{aligned} p(x_{0:t} \mid y_{1:t}) &= \frac{p(y_t \mid x_{0:t}, y_{1:t-1}) p(x_{0:t} \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})} \\ &= \frac{p(y_t \mid x_{0:t}, y_{1:t-1}) p(x_t \mid x_{0:t-1}, y_{1:t-1}) p(x_{0:t-1} \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})} \\ &= \frac{p(y_t \mid x_t) p(x_t \mid x_{t-1}) p(x_{0:t-1} \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})} \\ &\propto p(y_t \mid x_t) p(x_t \mid x_{t-1}) p(x_{0:t-1} \mid y_{1:t-1}), \end{aligned}$$

the recursive form of the weights are

$$\begin{aligned} W_t^{(i)} &\propto \frac{p(x_{0:t}^{(i)} \mid y_{1:t})}{q(x_{0:t}^{(i)} \mid y_{1:t})} \\ &= \frac{p(y_{1:t} \mid x_{0:t}^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}) p(x_{0:t-1}^{(i)} \mid y_{1:t-1})}{q(x_t^{(i)} \mid x_{0:t-1}^{(i)}, y_t) q(x_{0:t-1}^{(i)} \mid y_{1:t-1})} \\ &= W_{t-1}^{(i)} \frac{p(y_{1:t} \mid x_{0:t}^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{q(x_t^{(i)} \mid x_{0:t-1}^{(i)}, y_t)}. \end{aligned}$$

### 4.2.3 Sequential Importance Sampling and Resampling

In practice, we are interested in the current filtered estimate $p(x_t \mid y_{1:t})$ instead of $p(x_{0:t} \mid y_{1:t})$. Provided

$$q(x_t \mid x_{0:t-1}, y_{1:t}) = q(x_t \mid x_{t-1}, y_t),$$

the importance weights $W_t^{(i)}$ can be updated recursively via

$$W_t^{(i)} \propto W_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{q(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t)}.$$

The problem of SIS filter is that the distribution of importance weights becomes more and more skewed as time increases. Hence, after some iterations, only very few particles have non-zero importance weights. This phenomenon is called *weight degeneracy* or *sample impoverishment* Arnaud Doucet (2011).

The effective sample size $N_{\mathit{eff}}$ is suggested to monitor how bad the degeneration is, which is

$$N_{\mathit{eff}} = \frac{N}{1 + \mathrm{Var}(w_t^{*(i)})},$$

where $w_t^{*(i)} = \frac{p(x_t^{(i)}|y_{1:t})}{q(x_t^{(i)}|x_{t-1}^{(i)}, y_{1:t})}$. The more different between the biggest weight and smallest weight, the worse the degeneration is. In practice, the effective sample size is approximated by

$$\hat{N}_{\mathit{eff}} \approx \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2}.$$

If the value of $N_{\mathit{eff}}$ is less than some threshold, some procedure should be used to avoid a worse degeneration. There are two ways one can do: choose an appropriate PDF for importance sampling, or use resampling after SIS.

The idea of resampling is keeping the same size of particles, replacing the low weights particles with new ones. As discussed before,

$$p(x_t \mid y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(x_t - x_t^{(i)}).$$

After resampling, it becomes

$$\tilde{p}(x_t \mid y_{1:t}) = \sum_{j=1}^{N} \frac{1}{N} \delta(x_t - x_t^{(j)}) = \sum_{i=1}^{N} \frac{n_i}{N} \delta(x_t - x_t^{(i)}),$$

where $n_i$ represents how many times the new particles $x_t^{(j)}$ were duplicated from$x_t^{(i)}$.

Then the process of SIS particle filter with resampling is:

- Initial particles when $t = 0$. For $i = 1, \ldots, N$, draw samples $\{x_0^{(i)}\}$ from $p(x_0)$.
- For $t = 1, 2, \ldots,$ run the process recursively

- Importance sampling: draw sample $\{\tilde{x}_t^{(i)}\}_{i=1}^N$ from $q(x_t \mid y_{1:t})$, calculate their weights $\tilde{w}_t^{(i)}$ and normalize them.
- Resampling: Resample $\{\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}\}$ and get a new set $\{x_t^{(i)}, \frac{1}{N}\}$.
- Output the status at time $t$: $\hat{x}_t = \sum_{i=1}^N \tilde{x}_t^{(i)} \tilde{w}_t^{(i)}$.

In SIR, if we choose

$$q(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t) = p(x_t^{(i)} \mid x_{t-1}^{(i)}),$$

the weights become

$$\begin{aligned} w_t^{(i)} &\propto w_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{q(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t)} \\ &\propto w_{t-1}^{(i)} p(y_t \mid x_t^{(i)}). \end{aligned}$$

Because $w_{t-1}^{(i)} = \frac{1}{N}$, thus we have $w_t^{(i)} \propto p(y_t \mid x_t^{(i)})$ and

$$w = \frac{1}{\sqrt{2\pi\Sigma}} \exp\left(-\frac{1}{2}(y_{\text{true}} - y)\Sigma^{-1}(y_{\text{true}} - y)\right).$$

However, SMC methods are suffering some drawbacks. At any time point $k(k < t)$, if $t-k$ is too large, the approximation to marginal $p(x_k \mid y_{1:t})$ is likely to be rather poor as the successive resampling steps deplete the number of distinct particle co-ordinates $x_k$ Andrieu *et al.* (2010), which is also the difficulty of approximating $p(\theta, x_{1:t} \mid y_{1:t})$ with SMC algorithms Andrieu *et al.* (1999) Fearnhead (2002) Storvik (2002).

### 4.2.4 Auxiliary Particle Filter

The auxiliary particle filter (APF) was first introduced in Pitt and Shephard (1999) as an extension of SIR to perform inference in state space model. The author uses the idea of stratification into particle filter to solve particle degeneracy by pre-selecting particles before propagation.

At each step, the algorithm draws a sample of the particle index $i$, which will be propagated from $t-1$ into the $t$, on the mixture in (4.4). These indexes are auxiliary variables only used as an intermediary step, hence the name of the algorithm Pitt and Shephard (1999). Thus, the task becomes to sample from the joint density $p(x_t, i \mid y_{1:t})$. Define

$$p(x_t, i \mid y_{1:t}) \propto p(y_t \mid x_t) p(x_t \mid x_{t-1}^{(i)}) w_{t-1}^{(i)}, \tag{4.5}$$

and define $\mu_t^{(i)}$ as some characterization of $x_t \mid x_{t-1}$, which suggested by the author could be mean, mode, a sample and so on, then the joint density can be approximated by

$$\pi(x_t, i \mid y_{1:t}) \propto p(y_t \mid \mu_t^{(i)}) p(x_t \mid x_{t-1}^{(i)}) w_{t-1}^{(i)}, \tag{4.6}$$

with weights

$$w_t^{(i)} \propto \frac{p(y_t \mid x_t^{(i)})}{p(y_t \mid \mu_t^{k(i)})}.$$

This auxiliary variable based SIR requires only the ability to propagate and evaluate the likelihood, just as the original SIR suggestion of Gordon *et al.* (1993).

The main idea behind the APF, that is modifying the original sequence of target distributions to guide particles in promising regions, can be extended outside the filtering framework Johansen and Doucet (2008). It is also recommended in the literature Liu (2008) that the particles can be re-sampled not according to the normalized weights $w_t^{\mathrm{SISR}}(x_{1:t}) = \frac{p(x_{1:t})}{p(x_{1:t-1}q(x_t|x_{1:t-1})}$ but according to a generic score function $w_t(x_{1:t}) > 0$ at time t

$$w_t(x_{1:t}) = g(w_t^{\mathrm{SISR}}(x_{1:t})),$$

where $g : \mathbb{R}^+ \to \mathbb{R}^+$ is a monotonously increasing function, such as $g(x) = x^\alpha$, where $0 < \alpha \leq 1$.

### 4.2.5 Sequential Particle Filter

SMC method is effective for exploring the sequence of posteriors distribution $\pi(x_t \mid \theta) = p(x_t \mid y_{1:t}, \theta)$, where the static parameters are treated as known. An inference about $\pi_{t-1}$ is used to draw an inference on $\pi_t$ by SIS and resampling. Its interest is focused on $x_t$ instead of the whole path $x_{0:t}$, that is the filtering problem. However, this algorithm evolves, weights and resamples a population of $N$ number of particles, $x_t^{(1)}, \ldots, x_t^{(N)}$, so that at each time $t$ they are a properly weighted sample from $\pi(x_t \mid \theta)$. Additionally, it is not practicable on huge datasets, because of numerous iterations in sampling process.

As a complementary solution, sequential particle filter method was proposed by Nicolas Chopin (2002) as the first part of his Ph.D. thesis. Instead, sequential particle filter is using preliminary explorations of partial distribution $\pi(\theta \mid y_{1:k})$ $(k < t)$. The concept is: an inference of $\pi(\theta)$ is drawn from the first $k$ observations, let's name it "learning phase", and it is then updated through importance sampling to incorporate the following $l$ observations, name it "updating phase", Chopin (2002). This method

is named as iterated batch importance sampling (IBIS) algorithm, which is used for the recursive exploration of the sequence of parameter posterior distributions $\pi(\theta)$. It updates a population of $N$ particles for $\theta$, $\theta^{(1)}, \ldots, \theta^{(N)}$, so that at each time $t$ they are a properly weighted sample from $\pi(\theta)$. The algorithm includes occasional MCMC steps for rejuvenating the current population of particles of $\theta$ to prevent the number of distinct from decreasing over time.

In a batch mode, we are assuming the parameter $\theta$ is static. When the first $k$ observations become available, we can find the posterior distribution is $\pi(\theta \mid y_{1:k})$. After that, a few $l(l < \infty)$ observations come into data stream, the posterior becomes $\pi(\theta \mid y_{1:k+l})$ and it is likely to be similar with $\pi(\theta \mid y_{1:k})$. Hence, a proper reweighting particles by the incremental weight is

$$\begin{aligned} w_{k,l}(\theta) &\propto \frac{\pi(\theta \mid y_{1:k+l})}{\pi(\theta \mid y_{1:k}} \\ &\propto \frac{p(y_{1:k+l} \mid \theta)}{p(y_{1:k} \mid \theta)} \\ &= p(y_{k+1:k+l} \mid y_{1:k}, \theta). \end{aligned}$$

Sequentially, the iterated batch importance sampling algorithm is in the following

- Initialization. General particles of $\theta_i$ and $w_i$, $i = 1, \ldots, N$.
- Reweighting. Update the weights by $w_i^* = w_i \times w_{k,l}$, where $w_{k,l}(\theta_i) \propto p(y_{k+1:k+l} \mid y_{1:k}, \theta_i)$, $i = 1, \ldots, N$.
- Resampling. Normalize $\theta_i$ and $w_i^*$ to $\theta_i^*$ and $\frac{1}{N}$ according to $p(\theta_i^* = \theta_i) = \frac{w_i^*}{\sum w_i^*}$, $i = 1, \ldots, N$.
- Move. Draw $\theta_i^m$ from $K_{k+l}(\theta_i^*)$, where $K_{k+l}$ is a predefined transition kernel function with stationary distribution $\pi_{k+l}$.
- Set $(\theta_i^m, \frac{1}{N})$ to $(\theta_i, w_i)$, $k + l$ to $k$ and return to reweighting step.

The algorithm stops when $k = t$, that is when the particle system targets the distribution of interest $\pi(\theta \mid y_{1:t})$.

### 4.2.6 Sequential MCMC

Markov Chain Monte Carlo (MCMC) methods are a set of powerful stochastic algorithms that allow us to solve most of these Bayesian computational problems when

the data are available in batches Tierney (1994). However, as data set becomes larger and larger, it requires numerous computing in the process. Sequential Monte Carlo approaches have become a powerful methodology to cope with large data set recursively. However, it is inefficient when applied to high dimensional problems Septier *et al.* (2009). A nature extension is whether there exists a sequential MCMC method to diversify the degenerate particle population thus improving the empirical approximation for multi-target tracking or high dimensional space. Luckily, sequential approaches using MCMC were proposed in Berzuini *et al.* (1997), in which the author combine MCMC with importance resampling to sequentially update the posterior distribution. Other discussions, such as Khan *et al.* (2005), Golightly and Wilkinson (2006) and Pang *et al.* (2008), are using either resampling nor importance sampling.

As we discussed before, the a filtering problem is to find the posterior distribution recursively, like

$$p(x_t \mid y_{1:t}) \propto \int p(y_t \mid x_t)p(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1})dx_{t-1}. \tag{4.7}$$

In particle filter, the posterior is approximated by particles $x_t^{(1)}, \ldots, x_t^{(N)}$ in equation (4.3). A MCMC procedure is designed using (4.3) as the target distribution with a proposal distribution of $q(x_t \mid x_t^{(i)})$. Therefore, like MCMC, the desired approximation $\hat{p}(x_t \mid y_{1:t})$ is obtained by storing every accepted samples after the initial burn-in iterations Septier *et al.* (2009). The drawback is excessive computation occurs as the number of particles increases at each iteration.

To avoid it, a MCMC-based particle algorithm in Pang *et al.* (2008) considers the joint posterior distribution of $x_t$ and $x_{t-1}$:

$$p(x_t, x_{t-1} \mid y_{1:t}) \propto p(y_t \mid x_t)p(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1}), \tag{4.8}$$

which becomes the new target distribution. Hence, the algorithm is summarized bellow:

- Initialize particles $x_0^{(1)}, \ldots, x_0^{(N)}$.
- Propose $\{x_k^*, x_{k-1}^*\} \sim q_1(x_k, x_{k-1} \mid x_k^{(i-1)}, x_{k-1}^{(i-1)})$.
- Accept $\{x_k^*, x_{k-1}^*\}$ with probability $\alpha_1 = \min\{1, \frac{p(x_k^*, x_{k-1}^* | y_{1:t})}{p(x_k^{(i-1)}, x_{k-1}^{(i-1)} | y_{1:t})} \frac{q_1(x_k^{(i-1)}, x_{k-1}^{(i-1)} | x_k^*, x_{k-1}^*)}{q_1(x_k^*, x_{k-1}^* | x_k^{(i-1)}, x_{k-1}^{(i-1)})}\}$
- Propose $x_{k-1}^* \sim q_2(x_{k-1} \mid x_k^{(i)}, x_{k-1}^{(i)})$
- Accept $x_{k-1}^*$ with probability $\alpha_2 = \min\{1, \frac{p(x_{k-1}^* | x_k^{(i)}, y_{1:t})}{p(x_{k-1}^{(i)} | x_k^{(i)}, y_{1:t})} \frac{q_2(x_{k-1}^{(i)} | x_{k-1}^*, x_k^{(i)})}{q_2(x_{k-1}^* | x_k^{(i)}, x_{k-1}^{(i)})}\}$

- Propose $x_k^* \sim q_3(x_k \mid x_k^{(i)}, x_{k-1}^{(i)})$
- Accept $x_k^*$ with probability $\alpha_3 = \min\{1, \frac{p(x_k^*|x_{k-1}^{(i)}, y_{1:t})}{p(x_k^{(i)}|x_{k-1}^{(i)}, y_{1:t})} \frac{q_3(x_k^{(i)}|x_k^*, x_{k-1}^{(i)})}{q_3(x_k^*|x_k^{(i)}, x_{k-1}^{(i)})}\}$
- After burn-in points, keep $x_k^{(j)}$ as new particles for approximating.
- Move to next particle $i + 1$
- Move to next state $k + 1$

In Septier *et al.* (2009), the authors discussed some attractive features of genetic algorithms and simulated annealing into the framework of MCMC based particle scheme.

## 4.3 On-line State and Parameter Estimation

The state transition density and the conditional likelihood function depend not only upon the dynamic state $x_t$, but also on a static parameter vector $\theta$, which will be stressed by use of the notations $f(x_t \mid x_{t-1}, \theta)$ and $g(y_t \mid x_t, \theta)$. Putting the algorithms on-line means to update the parameters and states instantly as new observations coming into the data stream. For Bayesian dynamic models, however, the most natural option consists in treating the unknown parameter $\theta$, using the state space representation, as a component of the state which has no dynamic evolution, also referred to as a static parameter Cappé *et al.* (2007). The standard SMC is deficiency for on-line parameter estimation. As a result of the successive resampling steps, after a certain time $t$, the approximation $\hat{p}(\theta \mid y_{1:t})$ will only contain a single unique value for $\theta$. In other words, SMC approximation of the marginalized parameter posterior distribution is represented by a single Dirac delta function. It also causes error accumulation in successive Monte Carlo (MC) steps grows exponentially or polynomially in time Kantas *et al.* (2009).

Therefore, in this section, we are introducing some methods that will estimate combined state and parameter by either jointly estimating of state and parameter or by marginalizing the parameter using sufficient statistics.

### 4.3.1 Artificial Dynamic Noise

Some methods are trying to solve the posterior distribution $p(\theta \mid y_{1:t})$ by

$$p(\theta \mid y_{1:t}) \propto p(y_{1:t} \mid \theta)p(\theta) \tag{4.9}$$

through maximize the likelihood function without introducing any bias or controlling the bias in states propagation. A pragmatic approach to reduce parameter sample degeneracy and error accumulation in successive MC approximations is to adding an artificial dynamic equation on $\theta$ Higuchi (2001) Kitagawa (1998), which gives

$$\theta_{n+1} = \theta_n + \varepsilon_{n+1}.$$

The artificial noise $varepsilon_{t+1} \sim N(0, W_{t+1})$ is specified by covariance matrix $W_{t+1}$. With this noise, SMC can now be applied to approximate $p(x_{1:t}, \theta \mid y_{1:t})$. A related kernel density estimation method proposes a kernel density estimate of the target Liu and West (2001)

$$\hat{p}(\theta \mid y_{1:t}) = \frac{1}{N} \sum M(\theta - \theta_n^{(i)}).$$

At time $t+1$, the samples obtain a new set of particles.

### 4.3.2 Practical Filtering

A fixed-lag approach to filtering and sequential parameter learning was proposed in Polson *et al.* (2008). Its key idea is to express the the filtering distribution as a mixture of lag-smoothing distributions and to implement it sequentially.

With a fixed-lag $l$, the state filtering and parameter learning require the sequence of the joint distribution $p(x_t, \theta \mid y_{1:t})$, which implies the desired filtering distribution $p(x_t \mid y_{1:t})$ being marginalized as

$$p(x_t \mid y_{1:t}) = \int p(x_{t-l+1:t} \mid y_{1:t}) dx_{t-l+1:t-1},$$

and the parameter posterior distribution $p(\theta \mid y_{1:t})$. Arguing that the approximation that draws from $p(x_{0:t-l} \mid y_{1:t-1})$ are approximate draws from $p(x_{0:t-l} \mid y_{1:t})$, the state filtering with static parameter $\theta$ is

$$\begin{aligned} p(x_{t-l+1:t}, \theta \mid y_{1:t}) &= \int p(x_{t-l+1,t}, \theta \mid x_{0:t-l}, y_{1:t}) dp(x_{0:t-l} \mid y_{1:t}) \\ &\approx \int p(x_{t-l+1,t}, \theta \mid x_{0:t-l}, y_{1:t}) dp(x_{0:t-l} \mid y_{1:t-1}). \end{aligned}$$

Therefore, we can firstly draw some samples $x_{0:t-l}^{(i)}$ from $p(x_{0:t-l} \mid y_{1:t-1})$, which is approximately the same as $p(x_{0:t-l} \mid y_{1:t})$ and $i = 1, \ldots, M$. Then, use these samples to estimate states and parameter by

$$\begin{aligned} x_{t-l+1} &\sim p(x_{t-l+1} \mid x_{0:t-l}^{(i)}, \theta, y_{t-l+1:t}), \\ \theta &\sim p(\theta \mid x_{0:t-l}^{(i)}, x_{t-l+1}, y_{t-l+1:t}), \end{aligned}$$

with two-step Gibbs sampler. The algorithm is summarized below:

- Step 1. Initialization. Set $\theta^{(i)} = \theta_0$ as initial values, $i = 1, \ldots, N$.
- Step 2. Burn in: For $k = 1, \ldots, l$, initialize $\theta = \theta^{(i)}$. Generate $x_{0:k} \sim p(x_{0:k} \mid \theta, y_{1:k})$ and $\theta \sim p(\theta \mid x_{0:k}, y_{1:k})$.
- Step 3. Achieve a set of $\left(x_{0:k}^{(i)}, \theta^{(i)}\right)$.
- Step 4. Sequential updating: For $k = l+1, \ldots, t$: initialize $\theta = \theta^{(i)}$. Generate $x_{k-l+1:k} \sim p(x_{k-l+1:k} \mid x_{k-l}^{(i)}, \theta, y_{k-l+1,k})$ and $\theta \sim p(\theta \mid x_{0:k-l}^{(i)}, x_{k-l+1:k}, y_{1:k})$.
- Step 5. Achieve a set of $\left(x_{k-l+1}^{(i)}, \theta^{(i)}\right)$ and leave $x_{0:k-l}^{(i)}$ unchanged.

The speed and accuracy of this algorithm depends on the choice of sample size $M$ and the lag $l$, which is difficult, and there is a non-vanishing bias which is difficult to quantify Polson *et al.* (2008) Kantas *et al.* (2009).

### 4.3.3 Liu and West's Filter

Particles degeneracy is inevitable in SMC. Method in section 4.3.1 reduce the degeneracy by adding artificial noise to the parameters, however, that will also lead to the variance of estimates. Liu and West in Liu and West (2001) used a kernel smoothing approximation combined with a neat shrinkage idea to kill over-dispersion.

At time $t$, suppose we have particles $\{x_t^{(i)}\}$ and associated weights $\{w_t^{(i)}\}$, $i = 1, \ldots, N$, Bayes' theorem tells us that approximation to the posterior distribution $p(x_{t+1} \mid y_{1:t+1})$ at time $t+1$ of the state is

$$p(x_{t+1} \mid y_{1:t+1}) \propto \sum_{i=1}^{N} w_t^{(i)} p(x_{t+1} \mid x_t^{(i)}) p(y_{t+1} \mid x_{t+1}).$$

However, variance increases through over $t$ by the Gaussian mixture. In West (1993), the author is a smooth kernel density

$$p(\theta \mid y_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} N(\theta \mid m_t^{(i)}, h^2 V_t) \tag{4.10}$$

to against the sample dispersion. Because $N(\cdot \mid m, C)$ is a multivariate normal density with mean $m$ and covariance matrix $C$, so the above density (4.10) is a mixture of $N(\theta \mid m_t^{(i)}, h^2 V_t)$ distribution weighted by the sample weights $w_t^{(i)}$. Without this shrinkage approach, the standard kernel locations would be $m_t^{(i)} = \theta_t^{(i)}$, by which there is an over

dispersed kernel density, because of $(1 = h^2)V_t$ is always large than $V_t$. $\theta_t$ indicates the samples are from the time $t$ posterior, not time-varying.

To correct it, the idea of shrinkage kernel is

$$m_t^{(i)} = \alpha\theta_t^{(i)} + (1-\alpha)\bar{\theta}_t, \tag{4.11}$$

where $\alpha = \sqrt{1-h^2}$ and $h > 0$ is the smoothing parameter and the covariance is $V_t = \sum_{i=1}^{N} \frac{(\theta_t^{(i)} - \bar{\theta}_t)(\theta_t^{(i)} - \bar{\theta}_t)^\top}{N}$. Consequently, the resulting normal mixture retains the mean $\bar{\theta}_t$ and now has the correct covariance $V_t$, hence the over dispersion is trivially corrected Liu and West (2001).

A general algorithm is summarized bellow:

- Step 1. Identify the prior estimation of $(x_t, \theta)$ by $(\mu_{t+1}^{(i)}, m_t^{(i)})$, where $\mu_{t+1}^{(i)} = \mathrm{E}(x_{t+1} \mid x_t^{(i)}, \theta_t^{(i)})$, and $m_t^{(i)} = \alpha\theta_t^{(i)} + (1-\alpha)\bar{\theta}_t$.
- Step 2. Sample an auxiliary integer index $k$ from $\{1, \ldots, N\}$ with probability proportional to $g_{t+1}^{(i)} \propto w_t^{(i)} p(y_{t+1} \mid \mu_{t+1}^{(i)}, m_t^{(i)})$.
- Step 3. Sample a new parameter vector $\theta_{t+1}^{(k)}$ from $N(\theta_{t+1} \mid m_t^{(k)}, h^2 V_t)$.
- Step 4. Sample current state vector $x_{t+1}^{(k)}$ from $p(x_{t+1} \mid x_t^{(k)}, \theta_{t+1}^{(k)})$.
- Step 5. Evaluate weight $w_{t+1}^{(k)} \propto \frac{p(y_{t+1} \mid x_{t+1}^{(k)}, \theta_{t+1}^{(k)})}{p(y_{t+1} \mid \mu_{t+1}^{(k)}, m_t^{(k)})}$ and normalize it.

### 4.3.4 Storvik Filter

Storvik Filter, presented in Storvik (2002), is assuming that the posterior $p(\theta \mid x_{0:t}, y_{1:t})$ depends on a low dimensional set of sufficient statistics $s_t$ with an associated recursive update via $s_t = S(s_{t-1}, x_t, y_t)$. This approach is based on marginalizing the static parameters out of the posterior distribution, in which only the state vector needs to be considered, and aiming at reducing the particle impoverishment. It can be thought of as an extension of particle filters with additional steps of updating sufficient statistics and sampling parameters sequentially Lopes and Tsay (2011). In particular, models for which the underlying process is Gaussian and linear in the parameters can be handled by this approach Storvik (2002). Furthermore, some other many observational distributions with unknown parameters can be handled by this approach but subject to unavailable sufficient statistics.

The Storvik filter is summarized bellow:

- Step 1. Sample $x_{t+1}^{(i)}$ from $p(x_{t+1} \mid x_t^{(i)}, y_{1:t+1}, \theta^{(i)})$

- Step 2. Calculate weights $w_{t+1} \propto p(y_{t+1} \mid x_{t+1}^{(i)}, \theta^{(i)})$ and normalize it by $w_{t+1}^{(i)} = \frac{w_{t+1}^{(i)}}{\sum w_{t+1}^{(i)}}$
- Step 3. Re-sample $\{\theta_{t+1}^{(i)}, x_{t+1}^{(i)}, s_{t+1}^{(i)}\}$ according to $w_{t+1}$
- Step 4. Update sufficient statistics $s_{t+1}^{(i)} = S(s_t^{(i)}, x_{t+1}, y_{t+1})$
- Step 5. Sample $\theta^{(i)}$ from $p(\theta \mid s_{t+1}^{(i)})$

### 4.3.5 Particle Learning

Particle Learning, proposed by Carvalho *et al.* (2010), uses the similar sufficient statistics as Storvik filter does, in which the set of sufficient statistics is used for parameters estimation only. As an extension to the mixture Kalman Filter Chen and Liu (2000), Particle Learning allows parameters learning through out the process and utilize a re-sample propagate framework together with a set of particles that includes a set of sufficient statistics (if it is available) for the states.

By denoting $s_t$ and $s_t^x$ the sufficient statistics for the parameter and state respectively, the updating rules are satisfied: $s_t = S(s_{t-1}, x_t, y_t)$ and $s_t^x = K(s_{t-1}^x, \theta, y_t)$, where $K(\cdot)$ is the Kalman filter recursions. In Particle Learning, the prior to sampling from the proposal distribution uses a predictive likelihood and takes $y_{t+1}$ into account Vieira and Wilkinson (2016). This algorithm is summarized as following:

- Step 1. Resample $\{\tilde{z}_t^{(i)}\}_{i=1}^N = (\tilde{s}_t^{x(i)}, \tilde{s}_t^{(i)}, \tilde{\theta}^{(i)})$ from $p(z_t \mid s_t^x, s_t, \theta)$ with weight $w \propto p(y_{t+1} \mid s_t^x, \theta)$.
- Step 2. Draw $x_{t+1}^{(i)}$ from $p(x_{t+1} \mid \tilde{s}_t^x, \tilde{\theta}, y_{1:t+1})$.
- Step 3. Update sufficient statistics $s_{t+1} = S(\tilde{s}_t, x_{t+1}, y_{t+1})$.
- Step 4. Sample $\theta^{(i)}$ from $p(\theta \mid s_{t+1})$.
- Step 5. Update $s_{t+1}^x = K(s_t^x, \theta, y_{t+1})$.

### 4.3.6 Adaptive Ensemble Kalman Filter

Storvik filter and Particle learning algorithms are efficient in some ways, however, the drawbacks are obvious. One of them is that the sufficient statistics are not always available, or hard to find, for complex models. They are trying to reduce the problem of particle impoverishment, although in practice they didn't solve the problem completely

Chopin *et al.* (2010). An ensemble Kalman filter method was proposed for sequential state and parameter estimation Stroud *et al.* (2016). It is fully Bayesian and propagates the joint posterior density of states and parameters through over the process.

The ensemble Kalman filter, which is an extension to the standard Kalman filter Kalman *et al.* (1960), is an approximate filtering method introduced in the geophysics literature by Evensen (1994). Instead of working with the entire distribution, the ensemble Kalman filter stores, propagates, and updates an ensemble of vectors that approximates the state distribution Katzfuss *et al.* (2016).

Recall that an on-line combined parameters and state estimation relies on the decomposition of the joint posterior distribution

$$p(x_{t+1}, \theta \mid y_{1:t+1}) \propto p(x_{t+1} \mid y_{1:t+1}, \theta) p(\theta \mid y_{1:t+1}).$$

To implement on-line sequential estimation, the first term on the right side of the above formula should be written in the following recursive form

$$p(x_{t+1} \mid \theta, y_{1:t+1}) \propto p(y_{t+1} \mid x_{t+1}, \theta) \int p(x_{t+1} \mid x_t, \theta) p(x_t \mid \theta, y_{1:t}) dx_t, \tag{4.12}$$

and the second term is in the recursive form of

$$\begin{aligned} p(\theta \mid y_{1:t+1}) &\propto p(y_{1:t+1} \mid \theta) p(\theta) \\ &= p(y_{t+1} \mid \theta, y_{1:t}) p(\theta \mid y_{1:t}). \end{aligned} \tag{4.13}$$

The ensemble Kalman filter is used to find (4.12), which is the state inference. The estimated Kalman gain is

$$\hat{K}_{t+1}(\theta) = H_{t+1}(\theta) \hat{P}^f_{t+1}(\theta) H_{t+1}(\theta)^\top \hat{\Sigma}_{t+1}(\theta)^{-1}.$$

The posterior ensemble at time $t+1$ based on parameter $\theta$ is

$$x^i_{t+1} = x^{fi}_{t+1} + \hat{K}_{t+1}(\theta)(y_{t+1} + v^i_{t+1} + H_{t+1}(\theta) x^{fi}_{t+1}). \tag{4.14}$$

For the second term (4.13), the author Stroud *et al.* (2016) proposed a feasible likelihood approximation by a multivariate Gaussian distribution Mitchell and Houtekamer (2000) for high-dimensional states:

$$p(y_{t+1} \mid \theta, y_{1:t}) \propto \left| \hat{\Sigma}_t(\theta) \right|^{-\frac{1}{2}} \exp\left( -\frac{1}{2} \hat{e}_{t+1}(\theta)^\top \hat{\Sigma}_t(\theta)^{-1} \hat{e}_{t+1}(\theta) \right). \tag{4.15}$$

To find $p(\theta \mid Y_{1:t})$, a generic way is using normal approximation, where the posterior density is given by

$$p(\theta \mid y_{1:t}) \propto \exp\left( -\frac{1}{2} (\theta - m_t)^\top C_t^{-1} (\theta - m_t \right).$$

A grid-based representation is writing the posterior in $p(\theta \mid y_{1:t}) \propto p(y_t \mid \theta, y_{1:t-1})p(\theta \mid y_{1:t-1})$ and update the recursion weights by $\pi_{t,k} \propto p(y_t \mid \theta, y_{1:t-1})\pi_{t,k-1}$.

This algorithm is summarized as

- Step 1. Initialize samples $\theta^{(i)} \sim p(\theta)$ and $x_1^{(i)} \sim N(x_0, P_0)$.
- Step 2. Propagate. $x_t^{(i)} = G(x_{t-1}^{(i)}, \theta)$.
- Step 3. Approximate likelihood function by (4.15).
- Step 4. Update. Draw $\theta$ either use normal approximation or grid-based approximation to find (4.13).
- Step 5. Draw $\theta^{(i)} \sim p(\theta \mid y_{1:t})$.
- Step 6. Generate forecast ensemble by $x_t^{fi} = x_t^{pi} + w_t$.
- Step 7. Draw posterior ensemble using (4.14)

### 4.3.7 On-line Pseudo-Likelihood Estimation

Bayesian estimation requires the posterior distribution of $p(\theta \mid y_{1:t})$, where the $\theta$ is treated as a random variable. By contrast, maximum likelihood estimation is looking for a value $\hat{\theta}$, which maximum the likelihood $p(y_{1:t} \mid \theta)$.

The classical expectation maximization (EM) algorithm Dempster *et al.* (1977) for maximizing $l(\theta)$ is a two step procedure:

- E-step: Compute $Q(\theta_k, \theta) = \int \ln p_\theta(x_{0:t}, y_{1:t}) p_{\theta_k}(x_{0:t} \mid y_{1:t}) dx_{0:t}$.
- M-step: Update the parameter $\theta_k$ by $\theta_{k+1} = \arg\max Q(\theta_k, \theta)$.

Then $\{l(\theta_k)_k\}$ generated by the EM is a non-decreasing sequence.

A straightforward on-line EM algorithm uses SMC method to maximize $l(\theta)$. However, it requires estimating sufficient statistics base on joint probability distributions whose dimension is increasing over time and has a computational load of $O(N^2)$ per time step Kantas *et al.* (2009). To circumvent this problem, Andrieu *et al.* (2005) proposed a pseudo-likelihood function for finite state space models.

Assuming that the process is stationary, give a time lag $L \geq 1$ and any $k \geq 1$, $x_{1:t}$ and $y_{1:t}$ are sliced to $X_k \triangleq x_{kL+1:(k+1)L}$ and $Y_k \triangleq y_{kL+1:(k+1)L}$. For example: $X_1 = x_{L+1:2L}$ consisting of $L$ data. Further, the joint distribution of $p(X_k, Y_k)$ is

$$p(X_k, Y_k) = \pi(x_{kL+1}) F(y_{kL+1} \mid x_{xL+1}) \Pi_{n=kL+2}^{(k+1)L} G(x_n \mid x_{n-1}) F(y_n \mid x_n). \tag{4.16}$$

The likelihood of a block $Y_k$ of observations is given by

$$p(Y_k) = \int p(X_k, Y_k) dX_k, \tag{4.17}$$

and the log pseudo-likelihood for $m$ slices is $\sum_{k=0}^{m-1} \ln p(Y_k)$ Andrieu *et al.* (2005).

The advantage of this algorithm is that it only requires an approximation of the fixed dimensional distribution $p(X_k \mid Y_k)$ and don't suffer degeneracy for small $L$ Kantas *et al.* (2009). The disadvantage is that it only applies for stationary distribution, and can be observed empirically that the algorithm might converge to incorrect values and even sometimes drift away from the correct values as $t$ increases Andrieu *et al.* (2010).

## 4.4 Simulation Study

In this section, we are comparing the performance of Liu and West's filter, Storvik filter, particle learning and adaptive ensemble Kalman filter by a simple dynamic linear model, see example Liu and West (2001). Explicitly, the model is

$$\begin{aligned} y_t &= F x_t + \epsilon_t, \\ x_t &= \phi x_{t-1} + w_t, \\ x_0 &\sim N(m_0, C_0), \end{aligned}$$

where $\epsilon_t \sim N(0, \sigma^2)$ and $w_t \sim N(0, \tau^2)$, $x_t$ are hidden status and $y_t$ are observations. Assuming that $F = 1$, $\sigma^2 = 1$ and $\tau^2 = 1$. The initial value $x_0 = 0$. $\theta = \phi$ a single static parameter without unobserved state variable.

A length 897 simulated data set was generated from this $AR(1)$ model at $\phi = 0.8$. First of all, we should find the sufficient statistics for Storvik filter and Particle Learning. For Particle Learning, the sufficient statistics $s_t$ and $s_t^x$ are satisfying the updating rules $s_t = S(s_{t-1}, x_t, y_t)$ and $s_t^x = K(s_{t-1}^x, \phi, y_t)$. Because of the assumption, the Kalman observation map is $H_k = 1$ and the variances are normal distributed. Thus, the Kalman gain $K = 1$. For details, the Particle Learning algorithm runs as :

- Step 1. Resample $\{\tilde{z}_t^{(i)}\}_{i=1}^N = (\tilde{s}_t^{x(i)}, \tilde{s}_t^{(i)}, \tilde{\phi}^{(i)})$ from $p(z_t \mid s_t^x, s_t, \phi)$ with weight $w \propto p(y_{t+1} \mid s_t^x, \phi)$. It is found that

$$p(y_{t+1} \mid s_t^x, \phi) \propto \exp\left(-\frac{1}{2}\frac{(y_{t+1} - \phi x_t)^2}{\sigma^2}\right).$$

- Step 2. Draw $x_{t+1}^{(i)}$ from $p(x_{t+1} \mid \tilde{s}_t^x, \tilde{\phi}, y_{1:t+1})$.

$$\begin{aligned} p(x_{t+1} \mid \tilde{z}_t^{(i)}, y_{1:t+1}) &= p(x_{t+1} \mid s_t^x, \phi, y_{1:t+1}) \propto p(x_{t+1}, y_{1:t+1} \mid s_t^x, \phi) \\ &\propto p(x_{t+1} \mid s_t^x, \phi) p(y_{t+1} \mid x_{t+1}, s_t^x, \phi) \\ &= N(x_{t+1} \mid \phi x_t, 1) N(y_{t+1} \mid x_{t+1}, 1) \\ &\sim N(\frac{1}{2}(y_{t+1} + \phi x_t), \frac{1}{\sqrt{2}}) \end{aligned}$$

- Step 3. Update sufficient statistics $s_{t+1} = S(\tilde{s}_t, x_{t+1}, y_{t+1})$.

$$\begin{aligned} s_{t+1,1} &= x_{t+1} \\ s_{t+1,2} &= x_t x_{t+1} + s_{t,2} = x_t s_{t,1} + s_{t,2} \\ s_{t+1,3} &= x_t^2 + s_{t,3} = s_{t,1}^2 + s_{t,3}. \end{aligned}$$

- Step 4. Sample $\phi$ from $p(\phi \mid s_{t+1})$.

$$\begin{aligned} p(\phi \mid x_{1:t+1}, y_{1:t+1}) &\propto p(x_{1:t+1}, y_{1:t+1} \mid \phi) p(\phi) \propto p(x_{1:t+1} \mid \phi) p(\phi) \\ &\sim N\left(\phi \mid \frac{s_{t+1,2}}{s_{t+1,3}}, \frac{1}{s_{t+1,3}}\right). \end{aligned}$$

- Step 5. Update $s_{t+1}^x = K(s_t^x, \phi, y_{t+1})$.

## 4.5 Conclusion

# Chapter 5

# Adaptive Sequential MCMC for On-line State and Parameter Estimation

## 5.1 Introduction

Data assimilation is a sequential process, by which the observations are incorporated with a numerical model describing the evolution of this system throughout the whole process. It is applied in many fields, particularly in weather forecasting and hydrology. The quality of the numerical model determines the accuracy of this system, which requires sequential combined state and parameters inferences. Enormous literature has been done on discussing pure state estimation, however, fewer research is talking about estimating combined state and parameters, particularly in a sequential updating way.

Sequential Monte Carlo method is well studied in the scientific literature and quite prevalent in academic research in the last decades. It allows us to specify complex, non-linear time series patterns and enables performing real time Bayesian estimations when it is coupled with Dynamic Generalized Linear Models Vieira and Wilkinson (2016). However, model's parameters are unknown in real world application and it is a limit for standard SMC. Extensions to this algorithm have been done by researchers. Kitagawa Kitagawa (1998) proposed a self-organizing filter and augmenting the state vector with unknown parameters. The state and parameters are estimated simultaneously by either a non-Gaussian filter or a particle filter. Liu and West Liu and West (2001) proposed an improved particle filter to kill degeneracy, which is a normal issue in static parameters

estimation. They are using a kernel smoothing approximation, with a correction factor to account for over-dispersion. Alternatively, Strovik Storvik (2002) proposed a new filter algorithm by assuming the posterior depends on a set of sufficient statistics, which can be updated recursively. However, this approach only applies to parameters with conjugate priors Stroud *et al.* (2016). Particle learning was first introduced in Carvalho *et al.* (2010). Unlike Strovik filter, it is using sufficient statistics solely to estimate parameters and promises to reduce particle impoverishment. These particle like methods are all using more or less sampling and resampling algorithms to update particles recursively. Jonathan proposed in Stroud *et al.* (2016) a SMC algorithm by using ensemble Kalman filter framework for high dimensional space models with observations. Their approach combines information about the parameters from data at different time points in a formal way using Bayesian updating. In Polson *et al.* (2008), the authors rely on a fixed-lag length of data approximation to filtering and sequential parameter learning in a general dynamic state space model. This approach allows for sequential parameter learning where importance sampling has difficulties and avoids degeneracies in particle filtering. A new adaptive Markov Chain Monte Carlo method yields a quick and flexible way for estimating posterior distribution in parameter estimation Haario *et al.* (1999). This new Adaptive Proposal method, depends on history data, is introduced to avoid the difficulties of tunning the proposal distribution in Metropolis-Hastings methods.

In this paper, I'm proposing an Adaptive Delayed-Acceptance Metropolis-Hastings algorithm to estimate the posterior distribution for combined states and parameters. To keep a higher running efficiency for the algorithm, we suggest cutting off history data but jut using a fixed length of data up to current state, like a sliding window. Once a new observation comes into the data stream, this window shift one step forward with the same length. The efficiency of this algorithm is discussed regarding to tune the best step size in learning process.

## 5.2 Bayesian Inference on Combined State and Parameters

In a general state-space model of the following form, either the forward map $F$ in hidden states or the observation transition matrix $G$ is linear or non-linear. We are

considering the model

$$\text{Observation:} \quad y_t = G(x_t, \theta), \tag{5.1}$$

$$\text{Hidden State:} \quad x_t = F(x_{t-1}, \theta), \tag{5.2}$$

where $G$ and $F$ are linear processes with Gaussian white noises $\epsilon \sim N(0, R(\theta))$ and $\epsilon' \sim N(0, Q(\theta))$. This model has an initial state $p(x_0 \mid \theta)$ and a prior distribution of the parameter $p(\theta)$ is known or can be estimated. Therefore, for a general Bayesian filtering problem with known static parameter $\theta$, it requires computing the posterior distribution of current state $p(x_t \mid y_{1:t})$ at each time $t = 1, \ldots, T$ by marginalizing the previous state

$$p(x_t \mid y_{1:t}) = \int p(x_t \mid x_{t-1}, y_{1:t}) p(x_{t-1} \mid y_{1:t}) dx_{t-1},$$

where $y_{1:t} = \{y_1, \ldots, y_t\}$ is the observation information up to time $t$. However, if $\theta$ is unknown, one has to marginalize the posterior distribution for parameter by

$$p(x_t \mid y_{1:t}) = \int p(x_t \mid y_{1:t}, \theta) p(\theta \mid y_{1:t}) d\theta. \tag{5.3}$$

The approach in equation (5.3) relies on the two terms : (i) a conditional posterior distribution for the states given parameters and observations; (ii) a marginal posterior distribution for parameter $\theta$. Several methods can be used in finding the second term, such as cross validation, Expectation Maximization algorithm, Gibbs sampling, Metropolis-Hastings algorithm and so on. A Monte Carlo method is popular in research area solving this problem. Monte Carlo method is an algorithm that relies on repeated random sampling to obtain numerical results. To compute an integration of $\int f(x)dx$, one has to sampling as many independent $x_i$ $(i = 1, \ldots, N)$ as possible and numerically to find $\frac{1}{N}\sum_i f(x_i)$ to approximate the target function. In the target function, we draw samples of $\theta$ and use a numerical way to calculate its posterior $p(\theta \mid y_{1:t})$.

Additionally, the marginal posterior distribution for the parameter can be written in two different ways:

$$p(\theta \mid y_{1:t}) \propto p(y_{1:t} \mid \theta) p(\theta), \tag{5.4}$$

$$p(\theta \mid y_{1:t}) \propto p(y_t \mid y_{1:t-1}, \theta) p(\theta \mid y_{1:t-1}). \tag{5.5}$$

The above formula (5.4) is a standard Bayesian inference requiring a prior distribution $p(\theta)$. It can be used in off-line methods, in which $\hat{\theta}$ is inferred by iterating over a fixed observation record $y_{1:t}$. In contrast, formula (5.5) is defined in a recursive way

over time depending on the previous posterior at time $t-1$, which is known as on-line method. $\hat{\theta}$ is estimated sequentially as a new observation $y_{t+1}$ becomes available.

Therefore, the question becomes finding an efficient way to sampling $\theta$, such as Importance sampling Hammersley and Handscomb (1964) Geweke (1989), Rejection sampling Casella *et al.* (2004) Martino and Míguez (2010), Gibbs sampling Geman and Geman (1984), Metropolis-Hastings method Metropolis *et al.* (1953) Hastings (1970) and so on.

### 5.2.1 Log-likelihood Function of Parameter Posterior

To sample $\theta$, firstly we should find its distribution function by starting from the joint covariance matrix of $x_{0:t}$ and $y_{1:t}$. With a given $\theta$, suppose the joint covariance matrix is in the form of

$$\begin{bmatrix} x_{1:t} \\ y_{1:t} \end{bmatrix} \theta \sim N\left(0, \Sigma_t\right), \tag{5.6}$$

where $x_{1:t}$ represents the hidden states $\{x_0, x_1, \ldots, x_t\}$, $y_{1:t}$ represents observed $\{y_1, \ldots, y_t\}$ and $\theta$ is the set of all known and unknown parameters. The inverse of the covariance matrix $\Sigma_t^{-1}$ is the procedure matrix. In fact, it is a block matrix in the form of

$$\Sigma_t^{-1} = \begin{bmatrix} A_t & -B_t \\ -B_t^\top & B_t \end{bmatrix},$$

where $A_t$ is a $t \times t$ matrix of forward map hidden states, $B_t$ is a $t \times t$ matrix of observation errors up to time $t$. The structure of the matrices, such as bandwidth, sparse density, depending on the structure of the model. Temporally, we are using $A$ and $B$ to represent the matrices $A_t$ and $B_t$ here. Then we may find the covariance matrix easily by calculating the inverse of the procedure matrix

$$\begin{aligned} \Sigma &= \begin{bmatrix} (A - B^\top B^{-1} B)^{-1} & -(A - B^\top B^{-1} B)^{-1} B^\top B^{-1} \\ -B^{-1} B (A - B^\top B^{-1} B)^{-1} & (B - B^\top A^{-1} B)^{-1} \end{bmatrix} \\ &= \begin{bmatrix} (A-B)^{-1} & (A-B)^{-1} \\ (A-B)^{-1} & (I - A^{-1}B)^{-1} B^{-1} \end{bmatrix} \\ &\triangleq \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}. \end{aligned}$$

Because of the covariance $\Sigma_{YY} = (I - A^{-1}B)^{-1} B^{-1}$, therefore the inverse is

$$\Sigma_{YY}^{-1} = B(I - A^{-1}B) = BA^{-1}\Sigma_{XX}^{-1}.$$

Given the Choleski decomposition $LL^\top = A$, we have

$$\begin{aligned}\Sigma_{YY}^{-1} &= BL^{-\top}L^{-1}\Sigma_{XX}^{-1} \\ &= (L^{-1}B)^\top(L^{-1}\Sigma_{XX}^{-1})\end{aligned}$$

More usefully, by given another Choleski decomposition $RR^\top = A - B = \Sigma_{XX}^{-1}$,

$$\begin{aligned}y_{1:t}^\top\Sigma_{YY}^{-1}y_{1:t} &= (L^{-1}By_{1:t})^\top(L^{-1}\Sigma_{XX}^{-1}y_{1:t}) \\ &\triangleq W^\top(L^{-1}\Sigma_{XX}^{-1}y_{1:t})\end{aligned} \tag{5.7}$$

$$\begin{aligned}\det\Sigma_{YY}^{-1} &= \det B\det L^{-\top}\det L^{-1}\det R\det R^\top \\ &= \det B(\det L^{-1})^2(\det R)^2.\end{aligned} \tag{5.8}$$

From the objective function (5.4), the posterior distribution of $\theta$ is

$$p(\theta \mid y_{1:t}) \propto p(y_{1:t} \mid \theta)p(\theta) \propto e^{-\frac{1}{2}y_{1:t}\Sigma_{YY}^{-1}y_{1:t}}\sqrt{\det\Sigma_{YY}^{-1}}p(\theta).$$

Then by taking natural logarithm on the posterior of $\theta$ and using the useful solutions in equations (5.7) and (5.8), we will have

$$\ln L(\theta) = -\frac{1}{2}y_{1:t}^\top\Sigma_{YY}^{-1}y_{1:t} + \frac{1}{2}\sum\ln\mathrm{tr}(B) - \sum\ln\mathrm{tr}(L) + \sum\ln\mathrm{tr}(R) + \ln p(\theta). \tag{5.9}$$

### 5.2.2 The Forecast Distribution

From equation (5.5), a sequential way for estimating the forecast distribution is needed. Suppose it is

$$p(y_t \mid y_{1:t-1}, \theta) \sim N\left(\bar{\mu}_t, \bar{\sigma}_t\right). \tag{5.10}$$

Look back to the covariance matrices of observations that we found in the previous section

$$\begin{aligned}p(y_{1:t-1}, \theta) &\sim N\left(0, \Sigma_{YY}^{(t-1)}\right), \\ p(y_t, y_{1:t-1}, \theta) &\sim N\left(0, \Sigma_{YY}^{(t)}\right),\end{aligned}$$

where the covariance matrix of the joint distribution is $\Sigma_{YY}^{(t)} = (I_t - A_t^{-1}B_t)^{-1}B_t^{-1}$, $I_t$ is a $t \times t$ identity matrix. Then, by taking its inverse, we will get

$$\begin{aligned}\Sigma_{YY}^{(t)(-1)} &= B_t(I_t - A_t^{-1}B_t) \\ &= B_t(B_t^{-1} - A_t^{-1})B_t \\ &\triangleq \begin{bmatrix}B_t & 0\\ 0 & B_1\end{bmatrix}\begin{bmatrix}Z_t & b_t\\ b_t^\top & K_t\end{bmatrix}\begin{bmatrix}B_t & 0\\ 0 & B_1\end{bmatrix}\end{aligned}$$

where $Z_t$ is a $t \times t$ matrix, $b_t$ is a $t \times 1$ matrix and $K_t$ is a $1 \times 1$ matrix. Thus by taking its inverse again, we will get

$$\Sigma_{YY}^{(t)} = \begin{bmatrix} B_t^{-1}(Z_t - b_t K_t^{-1} b_t^\top)^{-1} B_t^{-1} & -B_t^{-1} Z_t^{-1} b_t (K_t - b_t^\top Z_t^{-1} b_t)^{-1} B_1^{-1} \\ -B_1^{-1} K_t^{-1} b_t^\top (Z_t - b_t K_t^{-1} b_t^\top)^{-1} B_t^{-1} & B_1^{-1}(K_t - b_t^\top Z_t^{-1} b_t)^{-1} B_1^{-1} \end{bmatrix}.$$

So, from the above covariance matrix, we can find the mean and variance of $p(y_t \mid y_{1:t-1}, \theta)$ are

$$\bar{\mu}_t = B_1^{-1} K_t^{-1} b_t^\top B_{t-1}^{-1} y_{1:t-1}, \tag{5.11}$$

$$\bar{\sigma}_t = B_1^{-1} K_t B_1^{-1}. \tag{5.12}$$

### 5.2.3 The Estimation Distribution

From the joint distribution (5.6), one can find the best estimation with a given $\theta$ by

$$\begin{aligned} \hat{x}_{1:t} \mid y_{1:t}, \theta &\sim N\left(A_t^{-1} B_t y_{1:t}, A_t^{-1}\right) \\ &\sim N(L^{-\top} L^{-1} B_t y_{1:t-1}, L^{-\top} L^{-1}) \\ &\sim N(L^{-\top} W, L^{-\top} L^{-1}). \end{aligned}$$

Consequently

$$\hat{x}_{1:t} = L^{-\top}(W + Z),$$

where $Z \sim N(0, I(\epsilon))$ is independent and identically distributed and drawn from a zero-mean normal distribution with variance $I(\epsilon)$.

For sole $x_t$, its joint distribution with $y_{1:t}$ is

$$x_t, y_{1:t} \mid \theta \sim N\left(0, \begin{bmatrix} C_t^\top (A_t - B_t)^{-1} C_t & C_t^\top (A_t - B_t)^{-1} \\ (A_t - B_t)^{-1} C_t & (I - A_t^{-1} B_t)^{-1} B_t^{-1} \end{bmatrix}\right),$$

where $C_t^\top = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}$ helps to achieve the last element in the matrix. Thus the filtering distribution of the state is

$$p(x_t \mid y_{1:t}, \theta) \sim N\left(\mu_t^{(x)}, \mathrm{Var}(x_t)\right),$$

where, after simplifying, the mean and variance are

$$\mu_t^{(x)} = C_t^\top A_t^{-1} B_t y_{1:t}, \tag{5.13}$$

$$\mathrm{Var}(x_t) = C_t^\top A_t^{-1} C_t. \tag{5.14}$$

Generally, researchers would like to find the combined estimation for $x_t$ and $\theta$ at time $t$ by

$$p(x_t, \theta \mid y_{1:t}) = p(x_t \mid y_{1:t}, \theta) p(\theta \mid y_{1:t}).$$

Differently, from the target equation (5.3), the state inference containing $N$ samples is a mixture Gaussian distribution in the following form

$$p(x_t \mid y_{1:t}) = \int p(x_t \mid y_{1:t}, \theta) p(\theta \mid y_{1:t}) d\theta \doteq \frac{1}{N} \sum_{i=1}^{N} p(x_t \mid \theta^{(i)}, y_{1:t}). \tag{5.15}$$

Suppose $x_t \mid y_{1:t}, \theta_i \sim N\left(\mu_{ti}^{(x)}, \mathrm{Var}(x_t)_i\right)$ is found from equation (5.13) and (5.14) for each $\theta_i$, then its mean is

$$\mu_t^{(x)} = \frac{1}{N} \sum_i \mu_{ti}^{(x)} \tag{5.16}$$

and the unconditional variance of $x_t$, by law of total variance, is

$$\begin{aligned} \mathrm{Var}(x_t) &= \mathrm{E}(\mathrm{Var}(x_t \mid y_{1:t}, \theta)) + \mathrm{Var}(\mathrm{E}(x_t \mid y_{1:t}, \theta)) \\ &= \frac{1}{N} \sum_i \left(\mu_{ti}^{(x)} \mu_{ti}^{(x)\top} + \mathrm{Var}(x_t)_i\right) - \frac{1}{N^2} \left(\sum_i \mu_{ti}^{(x)}\right) \left(\sum_i \mu_{ti}^{(x)}\right)^{\top}. \end{aligned} \tag{5.17}$$

## 5.3 Random Walk Metropolis-Hastings algorithm

Metropolis-Hastings algorithm is an important class of Markov Chain Monte Carlo (MCMC) algorithms Smith and Roberts (1993) Tierney (1994) Gilks *et al.* (1995). This algorithm has been used extensively in physics but was little known to others until Müller Müller (1991) and Tierney Tierney (1994) expounded the value of this algorithm to statisticians. The algorithm is extremely powerful and versatile and has been included in a list of "The Top 10 Algorithms" with the greatest influence on the development and practice of science and engineering in the 20th century Dongarra and Sullivan (2000) Medova (2008).

Given essentially a probability distribution $\pi$ (the "target distribution"), MH algorithm provides a way to generate a Markov Chain $x_1, x_2, \ldots, x_t$, who has the target distribution as a stationary distribution, for the uncertain parameters $x$ requiring only that this density can be calculated at $x$. Suppose that we can evaluate $\pi(x)$ for any $x$. The transition probabilities should satisfy the detailed balance condition

$$\pi(x^{(t)}) q(x', x^{(t)}) = \pi(x') q(x^{(t)}, x'),$$

which means that the transition from the current state $\pi(x^{(t)})$ to the new state $\pi(x')$ has the same probability as that from $\pi(x')$ to $\pi(x^{(t)})$. In sampling method, drawing

$x_i$ first and then drawing $x_j$ should have the same probability as drawing $x_j$ and then drawing $x_i$. However, in most situations, the details balance condition is not satisfied. Therefore, we introduce a function $\alpha(x, y)$ satisfying

$$\pi(x')q(x', x^{(t)})\alpha(x', x^{(t)}) = \pi(x^{(t)})q(x^{(t)}, x')\alpha(x^{(t)}, x').$$

In this way, a tentative new state $x'$ is generated from the proposal density $q(x'; x^{(t)})$ and it is then accepted or rejected according to acceptance probability

$$\alpha = \frac{\pi(x')}{\pi(x^{(t)})}\frac{q(x^{(t)}, x')}{q(x', x^{(t)})}. \tag{5.18}$$

If $\alpha \geq 1$, then the new state is accepted. Otherwise, the new state is accepted with probability $\alpha$.

Here comes an issues of how to choose $q(\cdot \mid x^{(t)})$. The most widely used subclass of MCMC algorithms is based around the Random Walk Metropolis (RWM). The RWM updating scheme was first applied by Metropolis Metropolis *et al.* (1953) and proceeds as follows. Given a current value of the $d$-dimensional Markov chain $x^{(t)}$, a new value $x'$ is obtained by proposing a jump $\epsilon =\mid x' - x^{(t)} \mid$ from the pre-specified Lebesgue density

$$\tilde{\gamma}(\epsilon^\star; \lambda) = \frac{1}{\lambda^d}\gamma\left(\frac{\epsilon^\star}{\lambda}\right), \tag{5.19}$$

with $\gamma(\epsilon) = \gamma(-\epsilon)$ for all $\epsilon$. Here $\lambda > 0$ governs the overall size of the proposed jump and plays a crucial role in determining the efficiency of any algorithm. In a random walk, the proposal density function $q(\cdot)$ can be chosen for some suitable normal distribution, and hence $q(x' \mid x^{(t)}) = N(x' \mid x^{(t)}, \epsilon^2)$ and $q(x^{(t)} \mid x') = N(x^{(t)} \mid x', \epsilon^2)$ cancel in the above equation (5.18) Sherlock *et al.* (2016). Therefore, to decide whether to accept the new state, we compute the quantity

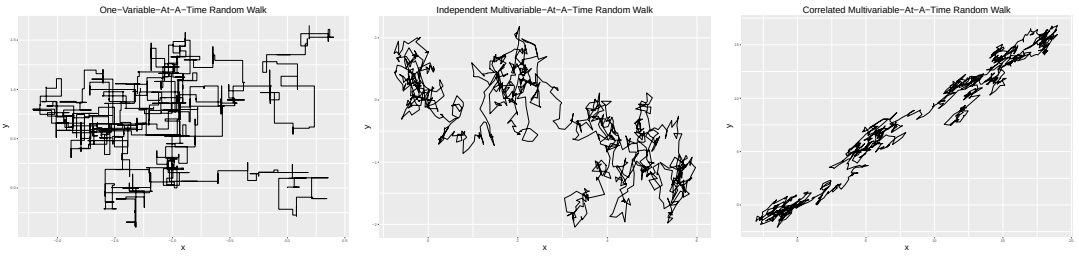
$$\alpha = \min\left\{1, \frac{\pi(x')q(x^{(t)} \mid x')}{\pi(x^{(t)})q(x' \mid x^{(t)})}\right\} = \min\left\{1, \frac{\pi(x')}{\pi(x^{(t)})}\right\}. \tag{5.20}$$

If the proposed value is accepted it becomes the next current value $x^{(t+1)} = x'$; otherwise the current value is left unchanged $x^{(t+1)} = x^{(t)}$ Sherlock *et al.* (2010).

### 5.3.1 Self-tuning Metropolis-Hastings Algorithm

In this section, we are proposing a Self-tuning MH algorithm with one-variable-at-a-time Random Walk, which can tune step sizes on its own to gain the target acceptance rates, to estimate the structure of parameters in a $d$-dimensional space. Supposing all

the parameters are independent, the idea of this algorithm is that in each iteration, only one parameter is proposed and the others are kept unchanged. After sampling, take $n$ samples out of the total amount of $N$ as new sequences. In figure (5.1), examples of different proposing methods are compared. To gain the target acceptance rates




(a) One-variable-at-a-time Random Walk.

(b) Independent Multi-variable-at-a-time Random Walk.

(c) Correlated Multi-variable-at-a-time Random Walk.

Figure 5.1: Examples of 2-Dimension Random Walk Metropolis-Hastings algorithm. Figure (a) is using one-variable-at-a-time proposal Random Walk. At each time, only one variable is changed and the other one stay constant. Figure (b) and (c) are using multi-variable-at-a-time Random Walk. The difference is in figure (b), every forward step are proposed independently, but in (c) are proposed according to the covariance matrix.

$\alpha_i (i = 1, \ldots, d)$, the step sizes $s_i$ for each parameter can be tuned automatically. The concept of the algorithm is if the proposal is accepted, then we have more confidence on the direction and step size that were made. In this scenario, the next movement should be further, that means the step size $s_{t+1}$ in the next step is bigger than $s_t$; otherwise, a conservative proposal is made with a shorter distance, which is $s_{t+1} \leq s_t$.

Supposing $a$ and $b$ are non-negative numbers indicating the distances of a forward movement, the new step size $s_{t+1}$ from current $s_t$ is

$$\ln s_{t+1} = \begin{cases} \ln s_t + a & \text{with probability } \alpha \\ \ln s_t - b & \text{with probability } 1 - \alpha \end{cases}, \tag{5.21}$$

where the logarithm guarantees the step size is positive. By taking its expectation

$$\mathrm{E}(\ln s_{t+1} \mid \ln s_t) = \alpha(\ln s_t + a) + (1 - \alpha)(\ln s_t - b),$$

and simplifying to

$$\mu = \alpha(\mu + a) + (1 - \alpha)(\mu - b),$$

we can find that

$$a = \frac{1-\alpha}{\alpha} b. \tag{5.22}$$

Thus, if the proposal is accepted, the step size $s_t$ is tuned to $s_{t+1} = s_t e^a$, otherwise $s_{t+1} = s_t / e^b$.

The complete one-variable-at-a-time MH is illustrated in the following table:

**Algorithm 1:** Self-tuning Random Walk Metropolis-Hastings Algorithm.

1 Initialization: Given an arbitrary positive step size $s_i^{(1)}$ for each parameter. Set up a value for $b$ and find $a$ by using formula (5.22). Set up a target acceptance rate $\alpha_i$ for each parameter, where $i = 1, \ldots, d$.

2 Run sampling algorithm: **for** *k from 1 to N* **do**

3 Randomly select a parameter $\theta_i^{(k)}$, propose a new one by $\theta_i' \sim N(\theta_i^{(k)}, \epsilon s_i^{(k)})$ and leave the rest unchanged.

4 Accept $\theta_i'$ with probability $\alpha = \min\left\{1, \frac{\pi(\theta')q(\theta^{(k)},\theta')}{\pi(\theta^{(k)})q(\theta',\theta^{(k)})}\right\}$.

5 If it is accepted, tune step size to $s_i^{(k+1)} = s_i^{(k)} e^a$, otherwise $s_i^{(k+1)} = s_i^{(k)} / e^b$.

6 Set $k = k + 1$ and move to step 3 until $N$.

7 **end**

8 Take $n$ samples out from $N$ with equal spaced index for each parameter being a new sequence.

The advantage of algorithm (1) is that it returns a more accurate estimation for $\theta$ and it is more reliable to learn the structure of parameter space. However, if $\pi(\cdot)$ is in an irregular structure, the algorithm is real time consuming and that cause a lower efficient. To accelerate the computation, we are introducing the Delayed Acceptance Metropolis-Hastings Algorithm.

### 5.3.2 Adaptive Delayed Acceptance Metropolis-Hastings Algorithm

The DA-MH algorithm proposed in Christen and Fox (2005) is a two-stage Metropolis-Hastings algorithm in which, typically, proposed parameter values are accepted or rejected at the first stage based on a computationally cheap surrogate $\hat{\pi}(x)$ for the likelihood $\pi(x)$. In stage one, the quantity $\alpha_1$ is found by a standard MH acceptance formula

$$\alpha_1 = \min\left\{1, \frac{\hat{\pi}(x')q(x^{(t)}, x')}{\hat{\pi}(x^{(t)})q(x', x^{(t)})}\right\},$$

where $\hat{\pi}(\cdot)$ is a cheap estimation for $x$ and a simple form is $\hat{\pi}(\cdot) = N(\cdot \mid \hat{x}, \epsilon)$. Once $\alpha_1$ is accepted, the process goes into stage two and the acceptance probability $\alpha_2$ is

$$\alpha_2 = \min\left\{1, \frac{\pi(x')\hat{\pi}(x^{(t)})}{\pi(x^{(t)})\hat{\pi}(x')}\right\}, \tag{5.23}$$

where the overall acceptance probability $\alpha_1\alpha_2$ ensures that detailed balance is satisfied with respect to $\pi(\cdot)$; however if a rejection occurs at stage one then the expensive evaluation of $\pi(x)$ at stage two is unnecessary.

For a symmetric proposal density kernel $q(x', x^{(t)})$ such as is used in the random walk MH algorithm, the acceptance probability in stage one is simplified to

$$\alpha_1 = \min\left\{1, \frac{\pi(x')}{\pi(x^{(t)})}\right\}. \tag{5.24}$$

If the true posterior is available then the delayed-acceptance Metropolis-Hastings algorithm is obtained by substituting this for the unbiased stochastic approximation in (5.23) Sherlock *et al.* (2015).

To accelerate the MH algorithm, Delayed-Acceptance MH requires a cheap approximate estimation $\hat{\pi}(\cdot)$ in formula (5.24). Intuitively, the approximation should be efficient with respect to time and accuracy to the true posterior $\pi(\cdot)$. A sensible option is assuming the parameter distribution at each time $t$ is following a normal distribution with mean $m_t$ and covariance $C_t$. So the posterior density is given by

$$\hat{\pi}(\theta \mid y_{1:t}) \propto \exp\left(-\frac{1}{2}(\theta - m_t)^\top C_t^{-1}(\theta - m_t)\right).$$

A lazy $C_t$ is using identity matrix, in which way all the parameters are independent. In terms of $m_t$, in most of circumstances, 0 is not an idea choice. To find an optimal or suboptimal $m_t$ and $C_t$, several algorithms have been discussed. In Stroud *et al.* (2016), the author is using a second-order expansion of $l(\theta)$ at the mode and the mean and covariance become $m_t = \arg\max l(\theta)$ and $C_t = -\left[\frac{\partial l(\theta)}{\partial\theta_i\partial\theta_j}\right]^{-1}_{\theta=m_t}$ respectively. The drawback of this estimation is a global optimum is not guaranteed. In Mathew *et al.* (2012), the author proposed a fast adaptive MCMC sampling algorithm, which is a consist of two phases. In the learning phase, they use hybrid Gibbs sampler to learn the covariance structure of the variance components. In phase two, the covariance structure is used to formulate an effective proposal distribution for a MH algorithm.

Likewise, we are suggesting that use a batch of data with length $L < t$ to learn the parameter space by using self-tuning random walk MH algorithm in the learning phase first. This algorithm tune each parameter at its own optimal step size and explore the

surface in different directions. When the process is done, we have a sense of Hypersurface of $\theta \approx \hat{\theta}$ and its mean $\hat{\mu} \approx m_L$ and covariance $\hat{\Sigma} \approx C_L$ can be estimated. Then we can move to the second phase: Delayed-Acceptance MH algorithm. The new $\theta'$ is proposed from $N(\theta^{(t)} \mid m_L, C_L)$, which is in the following form

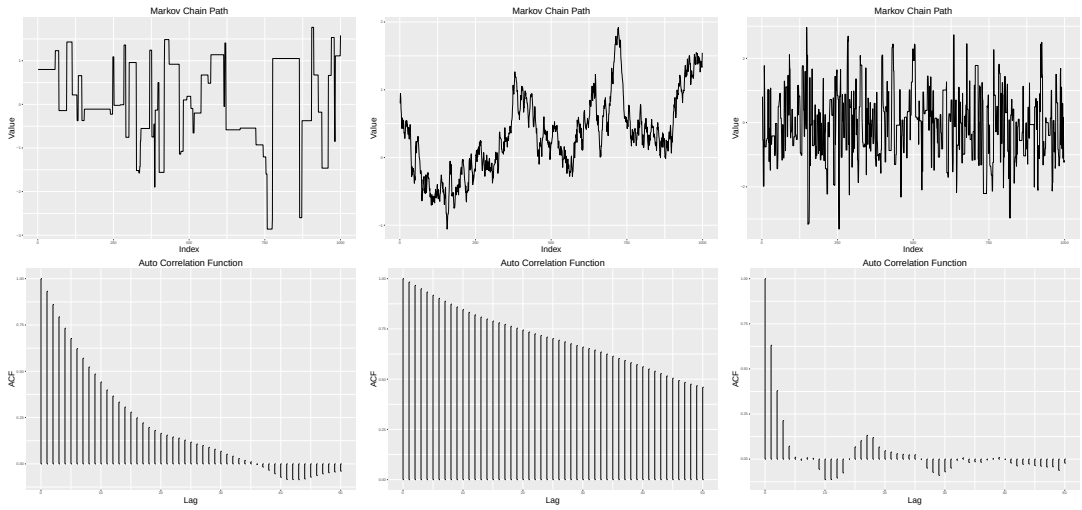
$$\theta' = \theta^{(t)} + R\epsilon z, \tag{5.25}$$

where $R^\top R = C_L$ is the Choleski decomposition, $\epsilon$ is the tuned step size and $z \sim N(0, 1)$ is Gaussian white noise. This proposing method decorrelates the impact of drawing $\theta'$ from a correlated space.

### 5.3.3 Efficiency of Metropolis-Hastings Algorithm

In equation (5.19), the jump size $\epsilon$ determines the efficiency of RWM algorithm. For a general RWM, it is intuitively clear that we can make the algorithm arbitrarily poor by making $\epsilon$ either very large or very small Sherlock *et al.* (2010). Assuming $\epsilon$ is extremely large, the proposal $x' \sim N(x^{(t)}, \epsilon)$, for example, is taken a further distance from current value $x^{(t)}$. Therefore, the algorithm will reject most of its proposed moves and stay where it was for a few iterations. On the other hand, if $\epsilon$ is extremely small, the algorithm will keep accepting the proposed $x'$ since $\alpha$ is always approximately be 1 because the continuity of $\pi(x)$ and $q(\cdot)$ Roberts *et al.* (2001). Thus, RWM takes a long time to explore the posterior space and converge to its stationary distribution. So, balance between these two extreme situations must exist. This appropriate step size $\hat{\epsilon}$ is an optimal, sometimes is suboptimal, solution to gain a Markov chain. Figure (5.2) illustrates the performances of RWM with different step size $\epsilon$. From these plots we may see that either too large or too small $\epsilon$ causes high correlation chains, indicating bad samples in sampling algorithm. An appropriate $\epsilon$ decorrelate samples and returns a stationary chain, which is said to be high efficiency.

Plenty of work has been done to determine the efficiency of Metropolis-Hastings algorithm in recent years. Gelman, Roberts, and Gilks Gelman *et al.* (1996) work with algorithms consisting of a single Metropolis move (not multi-variable-at-a-time), and obtain many interesting results for the $d$-dimensional spherical multivariate normal problem with a symmetric proposal distributions, including that the optimal scale is approximately $2.4/\sqrt{d}$ times the scale of target distribution, which implies optimal acceptance rates of 0.44 for $d = 1$ and 0.23 for $d \to \infty$ Gilks *et al.* (1995). Roberts and Rosenthal (2001) Roberts *et al.* (2001) evaluate scalings that are optimal (in the sense of integrated autocorrelation times) asymptotically in the number of components.




(a) With a large step size. (b) With a small step size. (c) With an appropriate step size.

Figure 5.2: Metropolis algorithm sampling for a single parameter with (a) a large step size, (b) a small step size, (c) an appropriate step size. The upper plots show the sample chain and lower plots indicate the autocorrelation for each case.

They find that an acceptance rate of 0.234 is optimal in many random walk Metropolis situations, but their studies are also restricted to algorithms that consist of only a single step in each iteration, and in any case they conclude that acceptance rates between 0.15 and 0.5 do not cost much efficiency. Other researchers Roberts *et al.* (1997) Bédard (2007), Beskos *et al.* (2009), Sherlock *et al.* (2009), Sherlock (2013) have been tackled for various shapes of target on choosing the optimal scale of the RWM proposal and led to the similar rule: choose the scale so that the acceptance rate is approximately 0.234. Although nearly all of the theoretical results are based upon limiting arguments in high dimension, the rule of thumb appears to be applicable even in relatively low dimensions Sherlock *et al.* (2010).

In terms of the step size $\epsilon$, it is pointed out that for a stochastic approximation procedure, its step size sequence $\{\epsilon_i\}$ should satisfy $\sum_{i=1}^{\infty} \epsilon_i = \infty$ and $\sum_{i=1}^{\infty} \epsilon_i^{1+\lambda} < \infty$ for some $\lambda > 0$. The former condition somehow ensure that any point of $X$ can eventually be reached, while the second condition ensures that the noise is contained and does not prevent convergence Andrieu and Thoms (2008). Sherlock, Fearnhead, and Roberts Sherlock *et al.* (2010) tune various algorithms to attain target acceptance rates, and their Algorithm 2 tunes step sizes of univariate updates to attain the optimal efficiency of Markov chain at the acceptance rates between 0.4 and 0.45. Additionally,

Graves in Graves (2011) mentioned that it is certainly that one could use the actual arctangent relationship to try to choose a good $\epsilon$: in the univariate example, if $\alpha$ is the desired acceptance rate, then $\epsilon = 2\sigma/\tan(\pi/2\alpha)$, where $\sigma$ is the posterior standard deviation, will be obtained. In fact, some explorations infer a linear relationship between acceptance rate and step size, which is $\text{logit}\alpha \approxeq 0.76 - 1.12 \log \epsilon/\sigma$, and the slope of the relationship is nearly equal to the constant -1.12 independently. However, in multi-variable-at-a-time RWM, one expects that the proper interpretation of $\sigma$ is not the posterior standard deviation but the average conditional standard deviation, which is presumably more difficult to estimate from a Metropolis algorithm. In a higher $d$-dimensional space, or propose multi-variable-at-a-time, suppose $\Sigma$ is known or could be estimated, then $X'$ can be proposed from $q \sim N(X, \epsilon^2\Sigma)$. Thus the optimal step size $\epsilon$ is required. A concessive way of RWM in high dimension is proposing one-variable-at-a-time and treating them as one dimension space individually. In any case, however, the behavior of RWM on a multi-variate normal distribution is governed by its covariance matrix $\Sigma$, and it is better than using a fixed $N(X, \epsilon^2 I_d)$ distributionRoberts *et al.* (2001).

To explore the efficiency of a MCMC process, we introduce some notions first. For an arbitrary square integrable function $g$, Gareth, Roberts and Jeffrey Roberts *et al.* (2001) define its integrated autocorrelation time by

$$\tau_g = 1 + 2\sum_{i=1}^{\infty} \text{Corr}\left(g(X_0), g(X_i)\right),$$

where $X_0$ is assumed to be distributed according to $\pi$. Because central limit theorem, the variance of the estimator $\bar{g} = \sum_{i=1}^{n} g(X_i)/n$ for estimating $\text{E}(g(X))$ is approximately $\text{Var}_\pi(g(X)) \times \tau_g/n$. The variance tells us the accuracy of the estimator $\bar{g}$. The smaller it is, the faster the chain converge. Therefore, they suggest that the efficiency of Markov chains can be found by comparing the reciprocal of their integrated autocorrelation time, which is

$$e_g(\sigma) \propto \left(\text{Var}_\pi(g(X))\tau_g\right)^{-1}.$$

However, the disadvantage of their method is that the measurement of efficiency is highly dependent on the function $g$. Instead, an alternative approach is using Effective Sample Size (ESS) Kass *et al.* (1998)Robert (2004). Given a Markov chain having $n$ iterations, the ESS measures the size of i.i.d. samples with the same standard error, which is defined in Gong and Flegal (2016) in the following form of

$$\text{ESS} = \frac{n}{1 + 2\sum_{k=1}^{\infty} \rho_k(X)} \approxeq \frac{n}{1 + 2\sum_{k=1}^{k_{\text{cut}}} \rho_k(X)} = \frac{n}{\tau},$$

where $n$ is the number of samples, $k_{\texttt{cut}}$ is lag of the first $\rho_k < 0.01$ or $0.05$ , and $\tau$ is the integrated autocorrelation time. Moreover, a wide support among both statisticians Geyer (1992) and physicists Sokal (1997) are using the following cost of an independent sample to evaluate the performance of MCMC, that is

$$\frac{n}{\text{ESS}} \times \text{cost per step} = \tau \times \text{cost per step}.$$

Being inspired by their researches, we now define the Efficiency in Unit Time (EUT) and ESS in Unit Time (ESSUT) as follows:

$$\text{EUT} = \frac{e_g}{T}, \tag{5.26}$$

$$\text{ESSUT} = \frac{ESS}{T}, \tag{5.27}$$

where $T$ represents the computation time, which is also known as running time. The computation time is the length of time, in minutes or hours, etc, required to perform a computational process. The best Markov chain with an appropriate step size $\epsilon$ should not only have a lower correlation, as illustrated in Figure (5.2), but also have less time consuming. The standard efficiency $e_g$ and ESS do not depend on the computation time, but EUT and ESSUT do. The best tuned step size gain the balance between the size of effective proposed samples and cost of time.

## 5.4 Simulation Studies

In this section, we consider the model in regular and irregular spaced time difference separately. For an one dimensional state space model, we consider the hidden state process $\{x_t, t \geq 1\}$ is a stationary and ergodic Markov process and transited by $F(x' \mid x)$. In this paper, we assume that the state of a system has an interpretation as the summary of the past one-step behavior of the system. The states are not observed directly but by another process $\{y_t, t \geq 1\}$, which is assumed depending on $\{x_t\}$ by the process $G(y \mid x)$ only and independent with each other. When observed on discrete time $T_1, \ldots, T_k$, the model is summarized on the directed acyclic following graph

$$\begin{array}{lccccccccc}
\text{State} & x_0 & \to & x_1 & \to \cdots & x_k & \to \cdots & x_t & \to \cdots \\
 & & & \downarrow & & \downarrow & & \downarrow & \\
\text{Observation} & & & y_1 & & y_k & & y_t & \\
 & & & \downarrow & & \downarrow & & \downarrow & \\
\text{Time} & & & T_1 & & T_k & & T_t &
\end{array}$$

We define $\Delta_k = T_k - T_{k-1}$. If $\Delta_t$ is a constant, we retrieve a standard *AR(1)* model process with regular spaced time steps; if $\Delta_t$ is not constant, then the model becomes more complicated with irregular spaced time steps.

### 5.4.1 Simulation on Regular Time Series Data

If the time steps are even spaced, the model can be written as a simple linear model in the following

$$\begin{aligned} y_t \mid x_t &\sim N(x_t, \sigma^2) \\ x_t \mid x_{t-1} &\sim N(\phi x_{t-1}, \tau^2), \end{aligned}$$

where $\sigma$ and $\tau$ are i.i.d errors occurring in processes and $\phi$ is a static process parameter in forward map. An initial value $x_0 \sim N(0, L)$ is known.

To get the joint distribution for $x_{0:t}$ and $y_{1:t}$

$$\left[\begin{array}{c} x \\ y \end{array}\middle|\theta\right] \sim N\left(0, \Sigma\right),$$

where $\theta = \{\phi, \sigma, \tau\}$, we should start from the procedure matrix $\Sigma^{-1}$, which looks like

$$\begin{bmatrix} \frac{1}{L^2} + \frac{\phi^2}{\tau^2} & \frac{-\phi}{\tau^2} & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \frac{-\phi}{\tau^2} & \frac{1+\phi^2}{\tau^2} + \frac{1}{\sigma^2} & \cdots & 0 & -\frac{1}{\sigma^2} & 0 & \cdots & 0 \\ 0 & \frac{-\phi}{\tau^2} & \cdots & 0 & 0 & -\frac{1}{\sigma^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\tau^2} + \frac{1}{\sigma^2} & 0 & 0 & \cdots & -\frac{1}{\sigma^2} \\ 0 & -\frac{1}{\sigma^2} & \cdots & 0 & \frac{1}{\sigma^2} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \frac{1}{\sigma^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -\frac{1}{\sigma^2} & 0 & 0 & \cdots & \frac{1}{\sigma^2} \end{bmatrix},$$

and denoted as $\Sigma^{-1} = \begin{bmatrix} A & -B \\ -B & B \end{bmatrix}$. Its inverse is the covariance matrix

$$\Sigma = \begin{bmatrix} (A-B)^{-1} & (A-B)^{-1} \\ (A-B)^{-1} & (I - A^{-1}B)^{-1}B^{-1} \end{bmatrix} \triangleq \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}, \tag{5.28}$$

where $B$ is a $t \times t$ diagonal matrix with elements $\frac{1}{\sigma^2}$. The covariance matrices $\Sigma_{XX} = (A-B)^{-1}$ and $\Sigma_{YY} = (I - A^{-1}B)^{-1}B^{-1}$ are easily found.

**Parameters Estimation**

In formula (5.4), the parameter posterior is estimated with observation data $y_{1:t}$. By using the algorithm (1), although it may take a longer time, we will achieve a precise estimation. Similarly with section (5.2.1), from the objective function, the posterior distribution of $\theta$ is

$$p(\theta \mid Y) \propto p(Y \mid \theta)p(\theta) \propto e^{-\frac{1}{2}Y\Sigma_{YY}^{-1}Y}\sqrt{\det \Sigma_{YY}^{-1}}p(\theta).$$

Then by taking natural logarithm on the posterior of $\theta$ and using the useful solutions in equations (5.7) and (5.8), we will have

$$\ln L(\theta) = -\frac{1}{2}Y^\top \Sigma_{YY}^{-1}Y + \frac{1}{2}\sum \ln \operatorname{tr}(B) - \sum \ln \operatorname{tr}(L) + \sum \ln \operatorname{tr}(R) + \ln p(\theta). \quad (5.29)$$

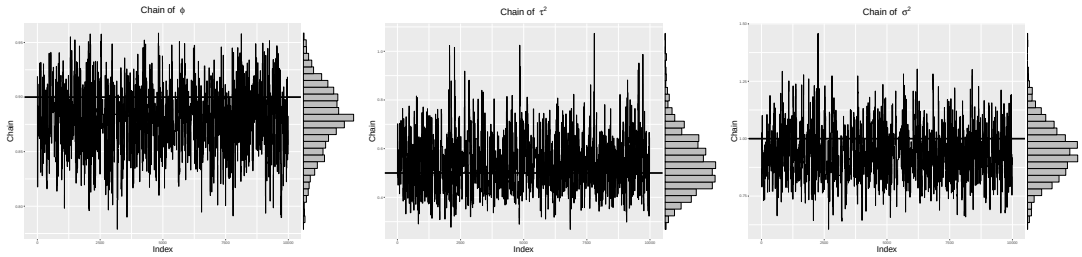
In a simple linear case, we are choosing the parameter $\theta = \{\phi = 0.9, \tau^2 = 0.5, \sigma^2 = 1\}$ as the author did in Lopes and Tsay (2011) and using $n = 500$ dataset, setting initial $L = 0$. Instead of inferring $\tau$ and $\sigma$, we are estimating $\nu_1 = \ln \tau^2$ and $\nu_2 = \ln \sigma^2$ in the RW-MH to avoid singular proposals. After the process, the parameters can be transformed back to original scale. Therefore, the new parameter $\theta^* = \{\phi, \nu_1, \nu_2\} = \{\phi, \ln \tau^2, \ln \sigma^2\}$.

Buy using algorithm (1) and aiming the optimal acceptance rate at 0.44, after 10 000 iterations we get the acceptance rates for each parameters are $\alpha_\phi = 0.4409, \alpha_{\nu_1} = 0.4289$ and $\alpha_{\nu_2} = 0.4505$, and the estimations are $\phi = 0.8794, \nu_1 = -0.6471$ and $\nu_2 = -0.0639$ respectively. Thus, we have the cheap surrogate $\hat{\pi}(\cdot)$. Keep going to the DA-MH with another 10 000 iterations, the algorithm returns the best estimation with $\alpha_1 = 0.1896$ and $\alpha_2 = 0.8782$. In figure (5.3), the trace plots illustrates that the Markov Chain of $\hat{\theta}$ is stably fluctuating around the true $\theta$.

**Recursive Forecast Distribution**

Calculating the log-posterior of parameters requires finding out the forecast distribution of $p(y_{1:t} \mid y_{1:t-1}, \theta)$. A general way is using the joint distribution of $y_t$ and $y_{1:t-1}$, which is $p(y_{1:t} \mid \theta) \sim N(0, \Sigma_{YY})$, and following the procedure in section 5.2.2 to work out the inverse matrix of a multivariate normal distribution. For example, one may find the inverse of the covariance matrix

$$\Sigma_{YY}^{-1} = B_t(I - A_t^{-1}B_t) = \frac{1}{\sigma^4}(\sigma^2 I_t - A_t^{-1}) \triangleq \frac{1}{\sigma^4}\begin{bmatrix} Z_t & b_t \\ b_t^\top & K_t \end{bmatrix}.$$




(a) Trace plot of $\phi$. (b) Trace plot of $\tau^2$. (c) Trace plot of $\sigma^2$.

Figure 5.3: Linear simulation with true parameter $\theta = \{\phi = 0.9, \tau^2 = 0.5, \sigma^2 = 1\}$. By transforming to original scale, the estimation is $\hat{\theta} = \{\phi = 0.8810, \tau^2 = 0.5247, \sigma^2 = 0.9416\}$.

Therefore, the original form of this covariance is

$$\Sigma_{YY} = \sigma^4 \begin{bmatrix} (Z_t - b_t K_t^{-1} b_t^\top)^{-1} & -Z_t^{-1} b_t (K_t - b_t^\top Z_t^{-1} b_t)^{-1} \\ -K_t^{-1} b_t^\top (Z_t - b_t K_t^{-1} b_t^\top)^{-1} & (K_t - b_t^\top Z_t^{-1} b_t)^{-1} \end{bmatrix}.$$

By denoting $C_t^\top = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}$ and post-multiplying $\Sigma_{YY}^{-1}$, we will have

$$\Sigma_{YY}^{-1} C_t = \frac{1}{\sigma^4}(\sigma^2 I - A^{-1}) C_t = \frac{1}{\sigma^4} \begin{bmatrix} b_t \\ K_t \end{bmatrix}. \tag{5.30}$$

A recursive way of calculating $b_t$ and $K_t$ is to use the Sherman-Morrison-Woodbury formula. In the late 1940s and the 1950s, Sherman and MorrisonSherman and Morrison (1950), Woodbury Woodbury (1950), Bartlett Bartlett (1951) and Bodewig **?** discovered the following result. The original Sherman-Morrison-Woodbury (for short SMW) formula has been used to consider the inverse of matrices Deng (2011). In this paper, we will consider the more generalized case.

Theorem 1.1 (Sherman-Morrison-Woodbury). Let $A \in B(H)$ and $G \in B(K)$ both be invertible, and $Y, Z \in B(K, H)$. Then $A + YGZ^*$ is invertible if and only if $G^{-1} + ZA^{-1}Y$ is invertible. In which case,

$$(A + YGZ^*)^{-1} = A^{-1} - A^{-1}Y(G^{-1} + ZA^{-1}Y)^{-1}ZA^{-1}. \tag{5.31}$$

A simple form of SMW formula is Sherman-Morrison formula represented in the following statement Bartlett (1951): Suppose $A \in R^{n \times n}$ is an invertible square matrix and $u, v \in R^n$ are column vectors. Then $A + uv\top$ is invertible $\iff 1 + u^\top A^{-1} v \neq 0$. If $A + uv\top$ is invertible, then its inverse is given by

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}. \tag{5.32}$$

By using the formula, one can find a recursive way to update $K_t$ and $b_{t-1}$, which is

$$K_t = \frac{\sigma^4}{\tau^2 + \sigma^2 + \phi^2(\sigma^2 - K_{t-1})}, \tag{5.33}$$

$$b_t = \begin{bmatrix} \frac{b_{t-1}\phi K_t}{\sigma^2} \\ \frac{K_t(\sigma^2+\tau^2)-\sigma^4}{\phi\sigma^2} \end{bmatrix}. \tag{5.34}$$

With the above formula, the recursive way of updating the mean and covariance is in the following formula:

$$\bar{\mu}_t = \frac{\phi}{\sigma^2} K_{t-1}\bar{\mu}_{t-1} + \phi(1 - \frac{K_{t-1}}{\sigma^2})y_{t-1}, \tag{5.35}$$

$$\bar{\Sigma}_t = \sigma^4 K_t^{-1}, \tag{5.36}$$

where $K_1 = \frac{\sigma^4}{\sigma^2+\tau^2+L^2\phi^2}$. For calculation details, we refer readers to appendices (B.1.1).

**The Estimation Distribution**

As introduced in section (5.2.3), from the joint distribution of $x_{1:t}$ and $y_{1:t}$, one can find the best estimation with a given $\theta$ by

$$\hat{x}_{1:t} \mid y_{1:t}, \theta \sim N(L^{-\top}W, L^{-\top}L^{-1}),$$

where $W = L^{-1}B_t y_{1:t-1}$. Consequently

$$\hat{x}_{1:t} = L^{-\top}(W + Z),$$

where $Z \sim N(0, I(\epsilon))$ is independent and identically distributed and drawn from a zero-mean normal distribution with variance $I(\epsilon)$. Moreover, the mixture Gaussian distribution $p(x_t \mid y_{1:t})$ can be found by

$$\mu_t^{(x)} = \frac{1}{N}\sum_i \mu_{ti}^{(x)} \tag{5.37}$$

$$\mathrm{Var}(x_t) = \frac{1}{N}\sum_i \left(\mu_{ti}^{(x)}\mu_{ti}^{(x)\top} + \mathrm{Var}(x_t)_i\right) - \frac{1}{N^2}\left(\sum_i \mu_{ti}^{(x)}\right)\left(\sum_i \mu_{ti}^{(x)}\right)^\top. \tag{5.38}$$

To find $\mu_{ti}^{(x)}$ and $\mathrm{Var}(x_t)_i$, we will use the joint distribution of $x_t$ and $y_{1:t}$, which is $p(x_t, y_{1:t} \mid \theta) \sim N(0, \Gamma)$ and

$$\Gamma = \begin{bmatrix} C_t^\top(A_t - B_t)^{-1}C_t & C_t^\top(A_t - B_t)^{-1} \\ (A_t - B_t)^{-1}C_t & (I_t - A_t^{-1}B_t)^{-1}B_t^{-1} \end{bmatrix}.$$

Because of

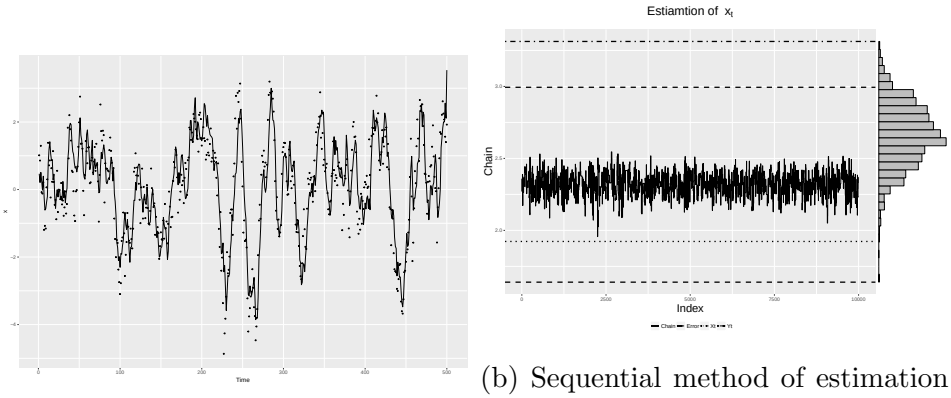
$$C_t^\top A_t^{-1} = \begin{bmatrix} -b_t^\top & \sigma^2 - K_t \end{bmatrix},$$

thus, for any given $\theta$, we have $\hat{x}_t \mid y_{1:t}, \theta \sim N\left(\mu_t^{(x)}, \mathrm{Var}(x_t)\right)$, where

$$\mu_t^{(x)} = \frac{K_t \bar{\mu}_t}{\sigma^2} + (1 - \frac{K_t}{\sigma^2}) y_t \tag{5.39}$$

$$\mathrm{Var}(x_t) = \sigma^2 - K_t. \tag{5.40}$$

By substituting them into the equation (5.37) and (5.38), the estimated $\hat{x}_t$ is easily got. For calculation details, we refer readers to appendices (B.1.1).




(a) Batch method of estimation $x_{1:t}$. $x_t$.

(b) Sequential method of estimation

Figure 5.4: Linear simulation of $x_{1:t}$ and sole $x_t$.In sub-figure (a), the dots is the true $x_{1:t}$ and the solid line is the estimation $\hat{x}_{1:t}$. In sub-figure (b), the chain in solid line is the estimation $\hat{x}_t$; dotted line is the true value of $x$; dot-dash line on top is the observed value of $y$; dashed lines are the estimated error.

### 5.4.2 Simulation on Irregular Time Series Data

Irregular sampled time series data is painful for scientists and researchers. In spatial data analysis, several satellites and buoy networks provide continuous observations of wind speed, sea surface temperature, ocean currents, etc. However, data was recorded with irregular time-step, with generally several data each day but also sometimes gaps of several days without any data. In Tandeo *et al.* (2011), the author adopt a continuous-time state-space model to analyze this kind of irregular time-step data, in which the state is supposed to be an Ornstein-Uhlenbeck process.

The OU process is an adaptation of Brownian Motion, which models the movement of a free particle through a liquid and was first developed by Albert Einstein Einstein (1956). By considering the velocity $u_t$ of a Brownian motion at time $t$, over a small time interval, two factors affect the change in velocity: the frictional resistance of the surrounding medium whose effect is proportional to $u_t$ and the random impact of neighboring particles whose effect can be represented by a standard Wiener process. Thus, because mass times velocity equals force, the process in a differential equation form is

$$mdu_t = -\omega u_t dt + dW_t,$$

where $\omega > 0$ is called the friction coefficient and $m > 0$ is the mass. If we define $\gamma = \omega/m$ and $\lambda = 1/m$, we obtain the OU process Vaughan (2015), which was first introduced with the following differential equation:

$$du_t = -\gamma u_t dt + \lambda dW_t.$$

The OU process is used to describe the velocity of a particle in a fluid and is encountered in statistical mechanics. It is the model of choice for random movement toward a concentration point. It is sometimes called a continuous-time Gauss Markov process, where a Gauss Markov process is a stochastic process that satisfies the requirements for both a Gaussian process and a Markov process. Because a Wiener process is both a Gaussian process and a Markov process, in addition to being a stationary independent increment process, it can be considered a Gauss-Markov process with independent increments Kijima (1997).

To apply OU process on irregular sampling data, we assume that the latent process $\{x_{1:t}\}$ is a simple OU process, that is a stationary solution of the following stochastic differential equation :

$$dx_t = -\gamma x_t dt + \lambda dW_t, \tag{5.41}$$

where $W_t$ is a standard Brownian motion, $\gamma > 0$ represents the slowly evolving transfer between two neighbor data and $\lambda$ is the forward transition variability. It is not hard to find the solution of equation (5.41) is

$$x_t = x_{t-1}e^{-\gamma t} + \int_0^t \lambda e^{-\gamma(t-s)} dW_s.$$

For any arbitrary time step $t$, the general form of the process satisfies

$$x_t = x_{t-1}e^{-\gamma \Delta_t} + \tau, \tag{5.42}$$

where $\Delta_t = T_t - T_{t-1}$ is the time difference between two consecutive data points, $\tau$ is a Gaussian white noise with mean zero and variances $\frac{\lambda^2}{2\gamma}\left(1 - e^{-2\gamma\Delta_t}\right)$.

The observed $y_{1:t}$ is measured by

$$y_t = Hx_t + \epsilon, \tag{5.43}$$

where $\epsilon \sim N(0, \sigma)$ is a Gaussian white noise.

To run simulations, we firstly generate irregular time lag sequence $\{\Delta_t\}$ from an *Inverse Gamma* distribution with parameters $\alpha = 2, \beta = 0.1$. Then the following parameters were chosen for the numerical simulation: $\gamma = 0.5$, $\lambda^2 = 0.1$, $\sigma^2 = 1$.

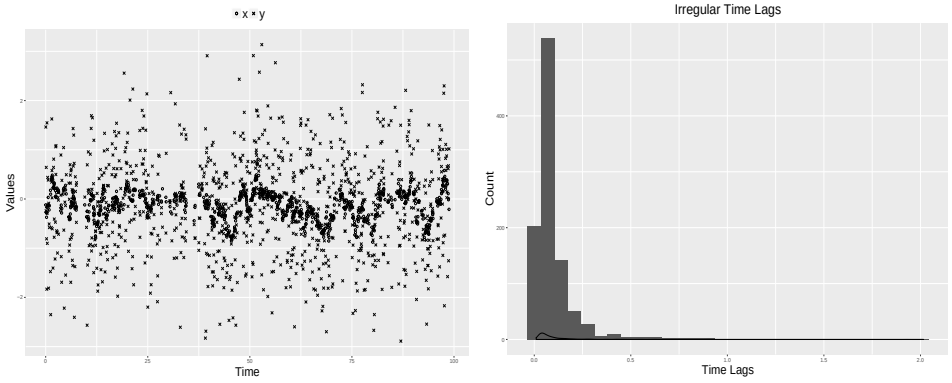



Figure 5.5: Simulated data. The circle dots are the true state $x_{1:t}$ and cross dots are observations $y_{1:t}$. Irregular time lag $\Delta_t$ are generated from *Inverse Gamma*(2,0.1) distribution.

Similarly, we can get the joint distribution for $x_{0:t}$ and $y_{1:t}$

$$\begin{bmatrix} x \\ y \end{bmatrix} \theta \sim N\left(0, \Sigma\right),$$

from the procedure matrix

$$\begin{bmatrix}
\frac{1}{L^2} + \frac{\phi_1^2}{\tau_1^2} & \frac{-\phi_1}{\tau_1^2} & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\frac{-\phi_1}{\tau_1^2} & \frac{1}{\tau_1^2} + \frac{\phi_2^2}{\tau_2^2} + \frac{1}{\sigma^2} & \cdots & 0 & -\frac{1}{\sigma^2} & 0 & \cdots & 0 \\
0 & \frac{-\phi_2}{\tau_2^2} & \cdots & 0 & 0 & -\frac{1}{\sigma^2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{1}{\tau_t^2} + \frac{1}{\sigma^2} & 0 & 0 & \cdots & -\frac{1}{\sigma^2} \\
0 & -\frac{1}{\sigma^2} & \cdots & 0 & \frac{1}{\sigma^2} & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & \frac{1}{\sigma^2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & -\frac{1}{\sigma^2} & 0 & 0 & \cdots & \frac{1}{\sigma^2}
\end{bmatrix},$$

where $\phi_t = e^{-\gamma\Delta_t}, \tau_t^2 = \frac{\lambda^2}{2\gamma}\left(1 - e^{-2\gamma\Delta_t}\right)$, $\theta$ represents unknown parameters. Denoted by $\Sigma^{-1} = \begin{bmatrix} A_t & -B_t \\ -B_t & B_t \end{bmatrix}$, covariance matrix is

$$\Sigma = \begin{bmatrix} (A_t - B_t)^{-1} & (A_t - B_t)^{-1} \\ (A_t - B_t)^{-1} & (I - A_t^{-1}B_t)^{-1}B_t^{-1} \end{bmatrix} \triangleq \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}, \tag{5.44}$$

where $B_t$ is a $t \times t$ diagonal matrix with elements $\frac{1}{\sigma^2}$. The covariance matrices $\Sigma_{XX} = (A_t - B_t)^{-1}$ and $\Sigma_{YY} = (I - A_t^{-1}B_t)^{-1}B_t^{-1}$.

**Parameters Estimation**

To use the algorithm (1), similarly with section (5.2.1), we firstly need to find the posterior distribution of $\theta$ with observations $y_{1:t}$, which in fact is

$$p(\theta \mid Y) \propto p(Y \mid \theta)p(\theta) \propto e^{-\frac{1}{2}Y\Sigma_{YY}^{-1}Y}\sqrt{\det \Sigma_{YY}^{-1}}p(\theta).$$

By taking natural logarithm on the posterior of $\theta$ and using the useful solutions in equations (5.7) and (5.8), we have

$$\ln L(\theta) = -\frac{1}{2}Y^\top\Sigma_{YY}^{-1}Y + \frac{1}{2}\sum \ln \operatorname{tr}(B) - \sum \ln \operatorname{tr}(L) + \sum \ln \operatorname{tr}(R) + \ln p(\theta). \tag{5.45}$$

Because of all parameters are positive, we are estimating $\nu_1 = \ln\lambda$, $\nu_2 = \ln\gamma^2$ and $\nu_3 = \ln\sigma^2$ instead. When the estimation process is done, we can transform them back to the original scale by taking exponential.

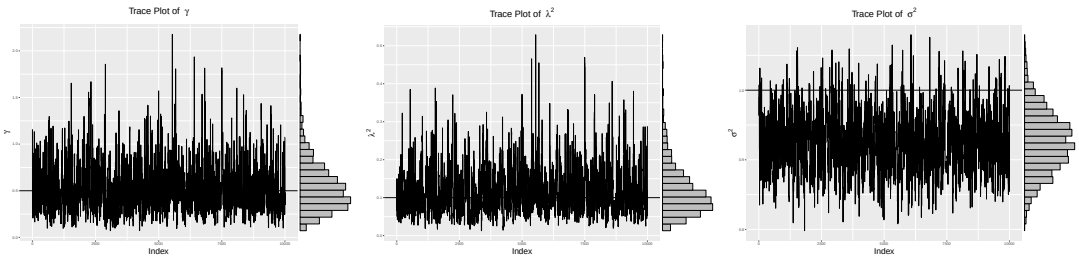
After running the whole process, it gives us the best estimation $\hat{\theta} = \{\gamma = 0.4841, \lambda^2 = 0.1032, \sigma^2 = 0.9276\}$. In figure (5.6), we can see that the $\theta$ chains are skew to the true value with tails.

**Recursive Calculation And State Estimation**

Follow the procedure in section 5.2.2 and do similar calculation with section 5.4.1, one can find a recursive way to update $K_t$ and $b_{t-1}$, which are

$$K_t = \frac{\sigma^4}{\tau_t^2 + \sigma^2 + \phi_t^2(\sigma^2 - K_{t-1})}, \tag{5.46}$$

$$b_t = \begin{bmatrix} \frac{b_{t-1}\phi_t K_t}{\sigma^2} \\ \frac{K_t(\sigma^2+\tau_t^2)-\sigma^4}{\phi_t\sigma^2} \end{bmatrix}. \tag{5.47}$$




(a) Trace plot of $\gamma$. (b) Trace plot of $\lambda^2$. (c) Trace plot of $\sigma^2$.

Figure 5.6: Irregular time step OU process simulation. The estimation of $\hat{\theta}$ is $\{\gamma = 0.4841, \lambda^2 = 0.1032, \sigma^2 = 0.9276\}$. In the plots, the horizontal dark lines are the true $\theta$.

With the above formula, the recursive way of updating the mean and covariance are

$$\bar{\mu}_t = \frac{\phi_t}{\sigma^2} K_{t-1} \bar{\mu}_{t-1} + \phi_t (1 - \frac{K_{t-1}}{\sigma^2}) y_{t-1}, \tag{5.48}$$

$$\bar{\Sigma}_t = \sigma^4 K_t^{-1}, \tag{5.49}$$

where $K_1 = \frac{\sigma^4}{\sigma^2 + \tau_1^2 + L^2 \phi_1^2}$.

Additionally, as introduced in section (5.2.3), the best estimation of $x_{1:t}$ with a given $\theta$ is

$$\hat{x}_{1:t} \mid y_{1:t}, \theta \sim N(L^{-\top} W, L^{-\top} L^{-1}),$$

where $W = L^{-1} B_t y_{1:t-1}$, and the mixture Gaussian distribution for $p(x_t \mid y_{1:t})$ is

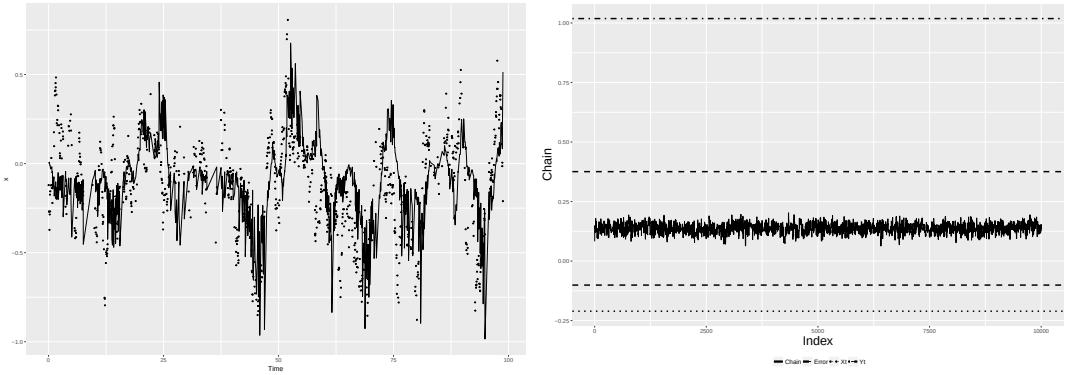
$$\mu_t^{(x)} = \frac{1}{N} \sum_i \mu_{ti}^{(x)} \tag{5.50}$$

$$\mathrm{Var}(x_t) = \frac{1}{N} \sum_i \left( \mu_{ti}^{(x)} \mu_{ti}^{(x)\top} + \mathrm{Var}(x_t)_i \right) - \frac{1}{N^2} \left( \sum_i \mu_{ti}^{(x)} \right) \left( \sum_i \mu_{ti}^{(x)} \right)^\top, \tag{5.51}$$

The same as we did in section 5.4.1, for any given $\theta$, we have $\hat{x}_t \mid y_{1:t}, \theta \sim N\left(\mu_t^{(x)}, \mathrm{Var}(x_t)\right)$, where

$$\mu_t^{(x)} = \frac{K_t \bar{\mu}_t}{\sigma^2} + (1 - \frac{K_t}{\sigma^2}) y_t$$
$$\mathrm{Var}(x_t) = \sigma^2 - K_t.$$

By substituting them into the equation (5.37) and (5.38), the estimated $\hat{x}_t$ is easily got. The difference at this time is the $\mu_t^{(x)}$ and $\mathrm{Var}(x_t)$ are dependent on time lag $\Delta_t$, that can be seen from formula (5.46) and (5.48).




(a) Batch method of estimation $x_{1:t}$. (b) Sequential method of estimation $x_t$.

Figure 5.7: Irregular time step OU process simulation of $x_{1:t}$ and sole $x_t$. In sub-figure (a), the dots is the true $x_{1:t}$ and the solid line is the estimation $\hat{x}_{1:t}$. In sub-figure (b), the chain in solid line is the estimation $\hat{x}_t$; dotted line is the true value of $x$; dot-dash line on top is the observed value of $y$; dashed lines are the estimated error.

## 5.5 High Dimensional OU-Process Application

Tractors moving on an orchard are mounted with GPS units, which are recording data and transfer to remote server. This data infers longitude, latitude, bearing, etc, with uneven spaced time mark. However, one dimensional OU process containing either only position or velocity is not enough to infer a complex movement.

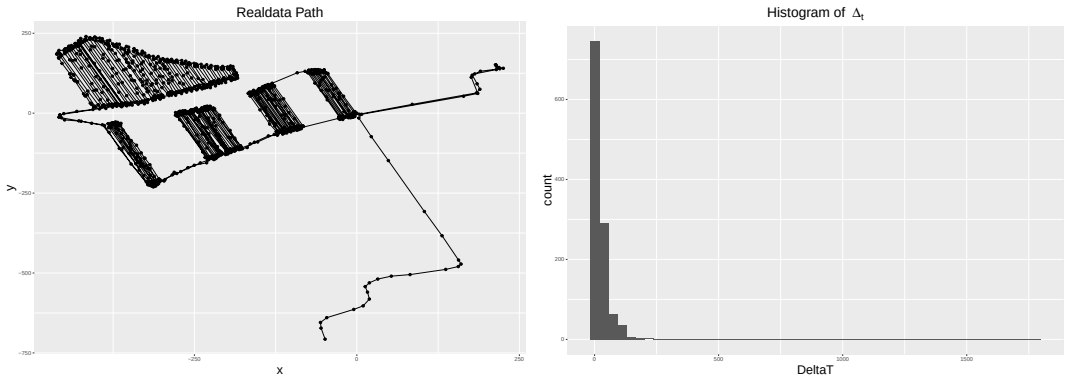



Figure 5.8: The trajectory of a moving tractor. The time lags (right side figure) obtained from GPS units are irregular.

Therefore, in this section, we are introducing an OU-process model combing both position and velocity with the following equations

$$\begin{cases} du_t = -\gamma u_t dt + \lambda dW_t, \\ dx_t = u_t dt + \xi dW'_t. \end{cases} \tag{5.52}$$

The solution can be found by integrating $dt$ out, that gives us

$$\begin{cases} u_t &= u_{t-1}e^{-\gamma t} + \int_0^t \lambda e^{-\gamma(t-s)} dW_s, \\ x_t &= x_{t-1} + \frac{u_{t-1}}{\gamma}(1-e^{-\gamma t}) + \int_0^t \frac{\lambda}{\gamma} e^{\gamma s}\left(1-e^{-\gamma t}\right) dW_s + \int_0^t \xi dW'_s. \end{cases} \tag{5.53}$$

As a result, the joint distribution is

$$\begin{bmatrix} x_t \\ u_t \end{bmatrix} \sim N\left( \begin{bmatrix} \mu_t^{(x)} \\ \mu_t^{(u)} \end{bmatrix}, \begin{bmatrix} \sigma_t^{(x)2} & \rho_t \sigma_t^{(x)} \sigma_t^{(u)} \\ \rho_t \sigma_t^{(x)} \sigma_t^{(u)} & \sigma_t^{(u)2} \end{bmatrix} \right), \tag{5.54}$$

where $\mu_t^{(x)}$ and $\mu_t^{(u)}$ are from the forward map process

$$\begin{bmatrix} \mu_t^{(x)} \\ \mu_t^{(u)} \end{bmatrix} = \begin{bmatrix} 1 & \frac{1-e^{-\gamma\Delta_t}}{\gamma} \\ 0 & e^{-\gamma\Delta_t} \end{bmatrix} \begin{bmatrix} x_{t-1}^{(x)} \\ u_{t-1} \end{bmatrix} \triangleq \Phi \begin{bmatrix} x_{t-1}^{(x)} \\ u_{t-1} \end{bmatrix}, \tag{5.55}$$

and

$$\begin{cases} \sigma_t^{(x)2} &= \frac{\lambda^2\left(e^{2\gamma\Delta_t}-1\right)\left(1-e^{-\gamma\Delta_t}\right)^2}{2\gamma^3} + \xi^2\Delta_t \\ \sigma_t^{(u)2} &= \frac{\lambda^2\left(1-e^{-2\gamma\Delta_t}\right)}{2\gamma} \\ \rho_t\sigma_t^{(x)}\sigma_t^{(u)} &= \frac{\lambda^2\left(e^{\gamma\Delta_t}-1\right)\left(1-e^{-2\gamma\Delta_t}\right)}{2\gamma^2} \end{cases}$$

In the above equations $\Delta_t = T_t - T_{t-1}$ and initial values are $\Delta_1 = 0$, $x_0 \sim N(0, L_x^2), u_0 \sim N(0, L_u^2)$, $\rho_t^2 = 1 - \frac{\xi^2\Delta_t}{\sigma_t^{(x)2}}$. To be useful, we are using $\frac{1}{1-\rho_t^2} = \frac{\sigma_t^{(x)2}}{\xi^2\Delta_t}$ instead in the calculation.

Furthermore, the independent observation process is

$$\begin{cases} y_t = x_t + \epsilon_t, \\ v_t = u_t + \epsilon'_t, \end{cases} \tag{5.56}$$

where $\epsilon_t \sim N(0,\sigma), \epsilon'_t \sim N(0,\tau)$ are normally distributed independent errors. Thus, the joint distribution of observations is

$$\begin{bmatrix} y_t \\ v_t \end{bmatrix} \sim N\left( \begin{bmatrix} x_t \\ u_t \end{bmatrix}, \begin{bmatrix} \sigma^2 & 0 \\ 0 & \tau^2 \end{bmatrix} \right). \tag{5.57}$$

Consequently, the parameter $\theta$ of an entire Ornstein-Uhlenbeck process is a set of five parameters from both hidden status and observation process, which is represented as $\theta = \{\gamma, \xi^2, \lambda^2, \sigma^2, \tau^2\}$.

Starting from the joint distribution of $x_{0:t}, u_{0:t}$ and $y_{1:t}, v_{1:t}$ by given $\theta$, it can be found that

$$\begin{bmatrix} \tilde{X} \\ \tilde{Y} \end{bmatrix} \Big| \theta \sim N\left(0, \tilde{\Sigma}\right), \tag{5.58}$$

where $\tilde{X}$ represents for the hidden statues $\{x, u\}$, $\tilde{Y}$ represents for observed $\{y, v\}$, $\theta$ is the set of five parameters. The inverse of the covariance matrix $\tilde{\Sigma}^{-1}$ is the procedure matrix in the form of

$$\tilde{\Sigma}^{-1} = \begin{bmatrix} Q_{xx} & Q_{xu} & -\frac{1}{\sigma^2}I & 0 \\ Q_{ux} & Q_{uu} & 0 & -\frac{1}{\tau^2}I \\ -\frac{1}{\sigma^2}I & 0 & \frac{1}{\sigma^2}I & 0 \\ 0 & -\frac{1}{\tau^2}I & 0 & \frac{1}{\tau^2}I \end{bmatrix}.$$

To make the covariance matrix a more beautiful form and convenient computing, $\tilde{X}$, $\tilde{Y}$ and $\tilde{\Sigma}$ can be rearranged in a time series order, that makes $X_{1:t} = \{x_1, u_1, x_2, u_2, \cdots, x_t, u_t\}$, $Y_{1:t} = \{y_1, v_1, y_2, v_2, \cdots, y_t, v_t\}$ and the new procedure matrix $\Sigma^{-1}$ looks like

$$\Sigma^{-1} = \begin{bmatrix} \sigma_{11}^{(x)2} + \frac{1}{\sigma^2} & \sigma_{11}^{(xu)2} & \cdots & \sigma_{1t}^{(x)2} & \sigma_{1t}^{(xu)2} & -\frac{1}{\sigma^2} & 0 & \cdots & 0 & 0 \\ \sigma_{11}^{(ux)2} & \sigma_{11}^{(u)2} + \frac{1}{\tau^2} & \cdots & \sigma_{1t}^{(ux)2} & \sigma_{1t}^{(x)2} & 0 & -\frac{1}{\tau^2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{t1}^{(x)2} & \sigma_{t1}^{(xu)2} & \cdots & \sigma_{tt}^{(x)2} + \frac{1}{\sigma^2} & \sigma_{tt}^{(xu)2} & 0 & 0 & \cdots & -\frac{1}{\sigma^2} & 0 \\ \sigma_{t1}^{(ux)2} & \sigma_{t1}^{(u)2} & \cdots & \sigma_{tt}^{(ux)2} & \sigma_{tt}^{(u)2} + \frac{1}{\tau^2} & 0 & 0 & \cdots & 0 & -\frac{1}{\tau^2} \\ -\frac{1}{\sigma^2} & 0 & \cdots & 0 & 0 & \frac{1}{\sigma^2} & 0 & \cdots & 0 & 0 \\ 0 & -\frac{1}{\tau^2} & \cdots & 0 & 0 & 0 & \frac{1}{\tau^2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -\frac{1}{\sigma^2} & 0 & 0 & 0 & \cdots & \frac{1}{\sigma^2} & 0 \\ 0 & 0 & \cdots & 0 & -\frac{1}{\tau^2} & 0 & 0 & \cdots & 0 & \frac{1}{\tau^2} \end{bmatrix} \triangleq \begin{bmatrix} A_t & - \\ -B_t^\top & \end{bmatrix}$$

where $B_t$ is a $2t \times 2t$ diagonal matrix of observation errors at time $t$ in the form of $\begin{bmatrix} \frac{1}{\sigma^2} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \frac{1}{\tau^2} & \cdot & \cdot & \cdot \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \cdot & \cdot & \cdot & \frac{1}{\sigma^2} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \frac{1}{\tau^2} \end{bmatrix}$. In fact, the matrix $A_t$ is a $2t \times 2t$ bandwidth six sparse matrix at time $t$ in the process. Temporally, we are using $A$ and $B$ to represent the matrices $A_t$ and $B_t$ here. Then we may find the covariance matrix by calculating the inverse of

the procedure matrix as

$$\begin{aligned}\Sigma &= \begin{bmatrix} (A - B^\top B^{-1} B)^{-1} & -(A - B^\top B^{-1} B)^{-1} B^\top B^{-1} \\ -B^{-1} B (A - B^\top B^{-1} B)^{-1} & (B - B^\top A^{-1} B)^{-1} \end{bmatrix} \\ &= \begin{bmatrix} (A-B)^{-1} & (A-B)^{-1} \\ (A-B)^{-1} & (I - A^{-1}B)^{-1} B^{-1} \end{bmatrix} \\ &\triangleq \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}.\end{aligned}$$

The detail structure of the covariance matrix $\Sigma_{XX}$ is in section (B.1.2).

### 5.5.1 Approximations of The Parameters Posterior

To find the log-posterior distribution of $X_{1:t}$ and $Y_{1:t}$, we shall start from the joint distribution. Similarly, the inverse of the covariance matrix is

$$\Sigma_{YY}^{-1} = B(I - A^{-1}B) = BA^{-1}\Sigma_{XX}^{-1}.$$

By using Choleski decomposition and similar technical solution, second term in the integrated objective function is

$$p(\theta \mid Y) \propto p(Y \mid \theta) p(\theta) \propto e^{-\frac{1}{2} Y \Sigma_{YY}^{-1} Y} \sqrt{\det \Sigma_{YY}^{-1}} P(\theta).$$

Then by taking natural logarithm on the posterior of $\theta$ and using the useful solutions in equations (5.7) and (5.8), we will have

$$\ln L(\theta) = -\frac{1}{2} Y^\top \Sigma_{YY}^{-1} Y + \frac{1}{2} \sum \ln \operatorname{tr}(B) - \sum \ln \operatorname{tr}(L) + \sum \ln \operatorname{tr}(R). \tag{5.59}$$

### 5.5.2 The Forecast Distribution

It is known that

$$\begin{aligned} p(Y_{1:t-1}, \theta) &\sim N\left(0, \Sigma_{YY}^{(t-1)}\right) \\ p(Y_t, Y_{1:t-1}, \theta) &\sim N\left(0, \Sigma_{YY}^{(t)}\right) \\ p(Y_t \mid Y_{1:t}, \theta) &\sim N\left(\bar{\mu}_t, \bar{\Sigma}_t\right) \end{aligned}$$

where the covariance matrix of the joint distribution is $\Sigma_{YY}^{(t)} = (I_t - A_t^{-1} B_t)^{-1} B_t^{-1}$. Then, by taking its inverse, we will get

$$\Sigma_{YY}^{(t)(-1)} = B_t (I_t - A_t^{-1} B_t).$$

To be clear, the matrix $B_t$ is short for the matrix $B_t(\sigma^2, \tau^2)$, which is $2t \times 2t$ diagonal matrix with elements $\frac{1}{\sigma^2}, \frac{1}{\tau^2}$ repeating for $t$ times on its diagonal. For instance, the very simple $B_1(\sigma^2, \tau^2) = \begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{\tau^2} \end{bmatrix}_{2\times 2}$ is a $2 \times 2$ matrix.

Because of $A_t$ is symmetric and invertible, $B_t$ is the diagonal matrix defined as above, then they have the following property

$$A_t B = A_t^\top B_t^\top = (B_t A_t)^\top,$$
$$A_t^{-1} B_t = A_t^{-\top} B_t^\top = (B_t A_t^{-1})^\top.$$

Followed up the form of $\Sigma_{YY}^{(t)(-1)}$, we can define that

$$\Sigma_{YY}^{(t)(-1)} \triangleq \begin{bmatrix} B_{t-1} & 0 \\ 0 & B_1 \end{bmatrix} \begin{bmatrix} Z_t & b_t \\ b_t^\top & K_t \end{bmatrix} \begin{bmatrix} B_{t-1} & 0 \\ 0 & B_1 \end{bmatrix}$$

where $Z_t$ is a $2t \times 2t$ matrix, $b_t$ is a $2t \times 2$ matrix and $K_t$ is a $2 \times 2$ matrix. Thus by taking its inverse again, we will get

$$\Sigma_{YY}^{(t)} = \begin{bmatrix} B_{t-1}^{-1}(Z_t - b_t K_t^{-1} b_t^\top)^{-1} B_{t-1}^{-1} & -B_{t-1}^{-1} Z_t^{-1} b_t (K_t - b_t^\top Z_t^{-1} b_t)^{-1} B_1^{-1} \\ -B_1^{-1} K_t^{-1} b_t^\top (Z_t - b_t K_t^{-1} b_t^\top)^{-1} B_{t-1}^{-1} & B_1^{-1}(K_t - b_t^\top Z_t^{-1} b_t)^{-1} B_1^{-1} \end{bmatrix}.$$

It is easy to find the relationship between $A_t$ and $A_t$ in the Sherman-Morrison-Woodbury form, which is

$$A_t = \begin{bmatrix} A_{t-1} & \cdot & \cdot \\ \cdot & \frac{1}{\sigma^2} & \cdot \\ \cdot & \cdot & \frac{1}{\tau^2} \end{bmatrix} + U_t U_t^\top \triangleq M_t + U_t U_t^\top,$$

where, in fact, $M_t = \begin{bmatrix} A_{t-1} & \cdot & \cdot \\ \cdot & \frac{1}{\sigma^2} & \cdot \\ \cdot & \cdot & \frac{1}{\tau^2} \end{bmatrix} = \begin{bmatrix} A_{t-1} & 0 \\ 0 & B_1 \end{bmatrix}$ and its inverse is $M_t^{-1} = \begin{bmatrix} A_{t-1}^{-1} & 0 \\ 0 & B_1^{-1} \end{bmatrix}$.
We may use Sherman-Morrison-Woodbury formula to find the inverse of $A_t$ in a recursive way, which is

$$A_t^{-1} = (M_t + U_t U_t^\top)^{-1} = M_t^{-1} - M_t^{-1} U_t (I + U_t^\top M_t^{-1} U_t)^{-1} U_t^\top M_t^{-1}. \tag{5.60}$$

Consequently, with some calculations, we will get

$$K_t = B_1^{-1} D_t (I + S_t^\top (B_1^{-1} - K_{t-1}) S_t + D_t^\top B_1^{-1} D_t)^{-1} D_t^\top B_1^{-1}, \tag{5.61}$$

and

$$b_t = \begin{bmatrix} -b_{t-1} \\ B_1^{-1} - K_{t-1} \end{bmatrix} S_t (I + S_t^\top (B_1^{-1} - K_{t-1}) S_t + D_t^\top B_1^{-1} D_t)^{-1} D_t^\top B_1^{-1},$$

that are updating in a recursive way. Therefore, one can achieve the recursive updating formula for the mean and covariance matrix, which are

$$\begin{cases} \bar{\mu}_t &= \Phi_t K_{t-1} B_1 \bar{\mu}_{t-1} + \Phi_t (I - K_{t-1} B_1) Y_{t-1} \\ \bar{\Sigma}_t &= (B_1 K_t B_1)^{-1} \end{cases}. \tag{5.62}$$

The matrix $K_t$ is updated via equation (5.61), or updating its inverse in the following form makes the computation faster, that is

$$\begin{cases} K_t^{-1} &= B_1 D_t^{-\top} D_t^{-1} B_1 + B_1 \Phi_t (B_1^{-1} - K_{t-1}) \Phi_t^\top B_1 + B_1, \\ \bar{\Sigma}_t &= D_t^{-\top} D_t^{-1} + \Phi_t (B_1^{-1} - K_{t-1}) \Phi_t^\top + B_1^{-1} \end{cases}$$

and $K_1 = B_1^{-1} - A_1^{-1} = \begin{bmatrix} \frac{\sigma^4}{\sigma^2 + L_x^2} & 0 \\ 0 & \frac{\tau^4}{\tau^2 + L_u^2} \end{bmatrix}$. For calculation details, readers can refer to section (B.1.2).

### 5.5.3 The Estimation Distribution

Because of the joint distribution (5.58), one can find the best estimation with a given $\theta$ by

$$X_{1:t} \mid Y_{1:t}, \theta \sim N(L^{-\top} W, L^{-\top} L^{-1}),$$

thus

$$\hat{X}_{1:t} = L^{-\top}(W + Z),$$

where $Z \sim N(0, I(\sigma, \tau))$.

For $X_t$, the joint distribution with $Y_{1:t}$ updated to time $t$ is

$$X_t, Y_{1:t} \mid \theta \sim N\left(0, \begin{bmatrix} C_t^\top (A_t - B_t)^{-1} C_t & C_t^\top (A_t - B_t)^{-1} \\ (A_t - B_t)^{-1} C_t & (I - A_t^{-1} B_t)^{-1} B_t^{-1} \end{bmatrix}\right),$$

where $C_t^\top = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$. Thus

$$X_t \mid Y_{1:t}, \theta \sim N(\mu_t^{(X)}, \Sigma_t^{(X)}),$$

where

$$\begin{aligned} \mu_t^{(X)} &= C_t^\top A^{-1} B Y = C_t^\top L^{-\top} W, \\ \Sigma_t^{(X)} &= C_t^\top A^{-1} C_t = U_t^\top U_t, \end{aligned}$$

and $U_t = L^{-1}C_t$. The recursive updating formula is

$$\mu_t^{(X)} = K_t B_1 \bar{\mu}_t + (I - B_1 K_t) Y_t \tag{5.63}$$

$$\Sigma_t^{(X)} = B_1^{-1} - K_t. \tag{5.64}$$

### 5.5.4 Prior Distribution for Parameters

The well known Hierarchical Linear Model, where the parameters vary at more than one level, was firstly introduced by Lindley and Smith in 1972 and 1973 Lindley and Smith (1972) Smith (1973). Hierarchical Model can be used on data with many levels, although 2-level models are the most common ones. The state space model in equations (5.1) and (5.2) is one of Hierarchical Linear Model if $G_t$ and $F_t$ are linear, and non-linear model if $G_t$ and $F_t$ are non-linear processes. Researchers have made a few discussions and works on these both linear and non-linear models. In this section, we only discuss on the prior for parameters in these models.

Various informative and non-informative prior distributions have been suggested for scale parameters in hierarchical models. Andrew Gelman gave a discussion on prior distributions for variance parameters in hierarchical models in 2006 Gelman *et al.* (2006). General considerations include using invariance Jeffries (1961), maximum entropy Jaynes (1983) and agreement with classical estimators Box and Tiao (2011). Regarding informative priors, Andrew suggests to distinguish them into three categories: The first one is traditional informative prior. A prior distribution giving numerical information is crucial to statistical modeling and it can be found from a literature review, an earlier data analysis or the property of the model itself. The second category is weakly informative prior. This genre prior is not supplying any controversial information but are strong enough to pull the data away from inappropriate inferences that are consistent with the likelihood. Some examples and brief discussions of weakly informative priors for logistic regression models are given in Gelman *et al.* (2008). The last one is uniform prior, which allows the information from the likelihood to be interpreted probabilistically.

Jonathan and Thomas in Stroud and Bengtsson (2007) have discussed a model, which is slightly different with a Gaussian state-space model from section one. The two errors $\omega_t$ and $\epsilon_t$ are assumed normally distributed as

$$\omega_t \sim N(0, \alpha Q),$$

$$\epsilon_t \sim N(0, \alpha R),$$

where the two matrices $R$ and $Q$ are known and $\alpha$ is an unknown scale factor to be estimated. (Note that a perfect model is obtained by setting $Q = 0$.) Therefore, the density of Gaussian state-space model is

$$p(y_t \mid x_t, \alpha) = N(F(x_t), \alpha R),$$
$$p(x_t \mid x_{t-1}, \alpha) = N(G(x_{t-1}), \alpha Q).$$

The parameter $\alpha$ is assumed *Inverse Gamma* distribution.

For the priors of all the parameters in OU-process, shown in equation (5.52) and (5.56), firstly we should understand what meanings of these parameters are standing for. The reciprocal of $\gamma$ is typical velocity falling in the reasonable range of 0.1 to 100 $m/s$. $\xi$ is the error occurs in transition process, $\sigma$ and $\tau$ are errors in the forward map for position and velocity respectively. Generally, the error is a positive finite number. Considering prior distributions for these parameters, before looking at the data, we have an idea of ranges where these parameters are falling in. Conversely, we don't have any assumptions about the true value of $\lambda$, which means it could be anywhere. According to this assumption, the prior distributions are

$$\gamma \sim IG(10, 0.5),$$
$$\xi^2 \sim IG(5, 2.5),$$
$$\sigma^2 \sim IG(5, 2.5),$$

where $IG(\alpha, \beta)$ represents the *Inverse Gamma* distribution with two parameters $\alpha$ and $\beta$.

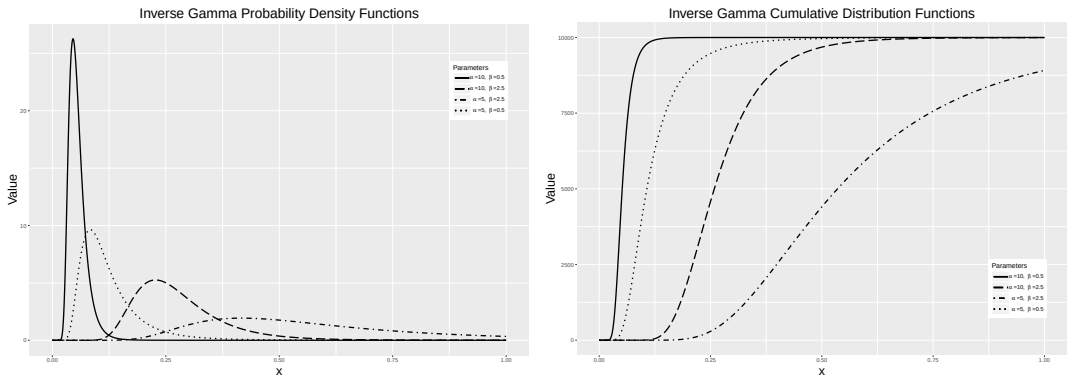



Figure 5.9: Probability density function and cumulative distribution function of *Inverse Gamma* with two parameters $\alpha$ and $\beta$.

### 5.5.5 Efficiency of DA-MH

We have discussed the efficiency of Delayed-Acceptance Metropolis-Hastings algorithm and how it is affected by the step size. To explain explicitly, here we give an example comparing Eff, EUT, ESS and ESSUT, which are calculated by using the same dataset and running 10 000 iterations of DA-MH. We are taking an 0.3-equal-spaced sequence $s = \{0.1, \ldots, 4\}$ from 0.1 to 4 and choosing each of them to calculate the criterion values. Table (5.1) and figure (5.10) show the results of this comparison.

The best step size found by Eff is 1, which is as the same as it found by ESS. By using $s = 1$ and running 1 000 iterations, the DA-MH takes 36.35 seconds to get the Markov chain for $\theta$ and the acceptance rates $\alpha_1$ for approximate $\hat{\pi}(\cdot)$ and $\alpha_2$ for posterior distribution $\pi(\cdot)$ are 0.3097 and 0.8324 respectively. By using EUT and ESSUT, the best step size is 2.5, which is bigger. The advantages of using this kind of step size is the computation time decreased to 5.10 seconds significantly. Because of the approximation $\hat{\pi}(\cdot)$ took bad proposals out and only approve good ones going to the next level, that can be seen from the lower rates $\alpha_1$ in table (5.1).

| | Values | Time (in seconds) | Step Size | $\alpha_1$ | $\alpha_2$ |
|---|---|---|---|---|---|
| Eff | 0.0515 | 36.35 | 1.0 | 0.3097 | 0.8324 |
| EUT | 0.0031 | 5.10 | 2.5 | 0.0360 | 0.78611 |
| ESS | 501.4248 | 36.35 | 1.0 | 0.3097 | 0.8324 |
| ESSUT | 29.8912 | 5.10 | 2.5 | 0.0360 | 0.78611 |

Table 5.1: An example of Eff, EUT, ESS and ESSUT found by running 10 000 iterations with same data.

On the surface, a bigger step size causes lower acceptance rates $\alpha_1$ and it might not be a smart choice. However, on the other hand, one should notice the less time cost. To make it sensible, we are running the Delayed-Acceptance MH with different step sizes, as presented in table (5.1), for the same (or similar) amount of time. Because of the bigger step size takes less time than smaller one, so we achieve a longer chain. To be more clear, we take 1 000 samples out from a longer chain, such as 8 500, and calculate Eff, EUT, ESS and ESSUT separately using the embedded function IAT and ESS in the package LaplacesDemon of $R$ and the above formulas. As we can see from the outcomes, by running the similar amount of time, the Markov chain using a bigger step size has a higher efficiency and effective sample size in unit time. More intuitively, the advantage of using larger step size is the sampling algorithm generates

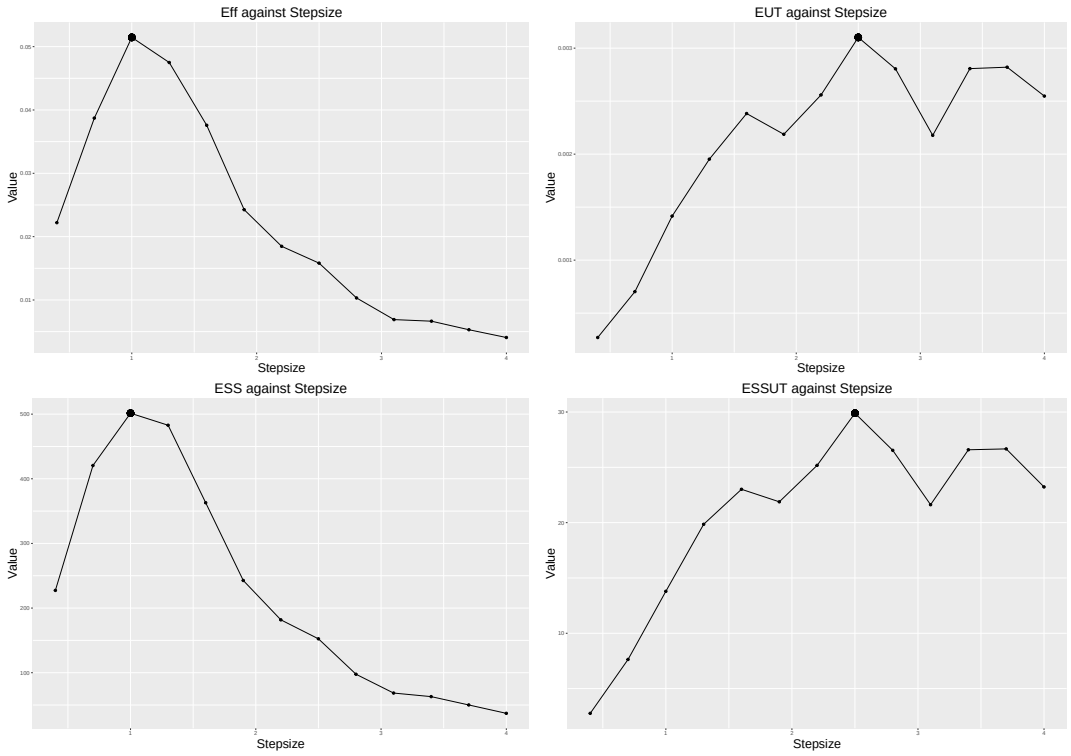



Figure 5.10: An example of Eff, EUT, ESS and ESSUT found by using the same data.

more representative samples per second. Figure (B.1) is comparing different $\theta$ chains found by using different step size but running the same amount of time. As we can see that $\theta$ with the optimal step size has a lower correlated relationship.

### 5.5.6 Sliding Window State and Parameter Estimation

The length of data used in the algorithm really affects the computation time. The forecast distribution $p(Y_t \mid Y_{1:t-1}, \theta)$ and estimation distribution $p(X_t \mid Y_{1:t}, \theta)$ require finding the inverse of the covariance $\Sigma_{YY}^{(t+1)}$, however, which is time consuming if the sample size is big to generate a large sparse matrix. For a moving vehicle, one is more willing to get the estimation and moving status instantly rather than being delayed. Therefore, a compromise solution is using fixed-length sliding window sequential filtering. A fixed-lag sequential parameter learning method was proposed in Polson *et al.* (2008) and named as *Practical Filtering*. The authors rely on the approximation of

$$p(x_{0:n-L}, \theta \mid y_{0:n-1}) \approx p(x_{0:n-L}, \theta \mid y_{0:n})$$

for large $L$. The new observations coming after the $n$th data has little influence on $x_{0:n-L}$.

| Step Size | Length of Data | Time (in seconds) | Eff | EUT | ESS | ESSUT |
|---|---|---|---|---|---|---|
| 1.0 | 1 000 | 3.48 | 0.0619 | 0.0178 | 69.4549 | 19.9583 |
| 1.3 | 1 400 | 3.40 | 0.0547 | 0.0161 | 75.37061 | 22.1678 |
| 1.3 | 1 000$^\star$ | 3.40 | 0.0813 | 0.0239 | 72.5370 | 21.3344 |
| 2.2 | 5 000 | 3.31 | 0.0201 | 0.0061 | 96.6623 | 29.2031 |
| 2.2 | 1 000$^\star$ | 3.31 | 0.0941 | 0.0284 | 94.2254 | 28.4669 |
| 2.5 | 7 000 | 3.62 | 0.0161 | 0.0044 | 112.3134 | 31.0258 |
| 2.5 | 1 000$^\star$ | 3.62 | **0.1095** | **0.0302** | **113.4063** | **31.3277** |

Table 5.2: Comparing Eff, EUT, ESS and ESSUT values using different step size. The 1000$^\star$ means taking 1 000 samples from a longer chain, like 1 000 out of 5 000 sample chain.

Being inspired, we are not using the first 0 to $n-1$ date and ignoring the latest $n$th, but using all the latest with truncating the first few history ones. Suppose we are given a fixed-length $L$, up to time $t$, which should be greater than $L$, we are estimating $x_t$ by using all the retrospective observations to the point at $t-L+1$. In another word, the estimation distribution for the current state is

$$p(X_t \mid Y_{t-L+1:t}, \theta), \tag{5.65}$$

where $t > L$. We name this method *Sliding Window Sequential Parameter Learning Filter*.

The next question is how to choose an appropriate $L$. The length of data used in MH and DA-MH algorithms has influence on the efficiency and accuracy of parameter learning and state estimation. Being tested on real data set, there is no doubt that the more data be in use, the more accurate the estimation is, and lower efficient is in computation. In table (B.1.3), one can see the pattern of parameters $\gamma, \xi, \tau$ follow the same trend with the choice of $L$ and $\sigma$ increases when $L$ decreases. Since estimation bias is inevitable, we are indeed to keep the bias as small as possible, and in the meantime, the higher efficiency and larger effective sample size are bonus items. In figure (5.11), we can see that the efficiency and effective sample size are not varying along the sample size used in sampling algorithm, but in unit time, they are decreasing rapidly as data size increasing. In addition, from a practical point of view, the observation error $\sigma$ should be kept at a reasonable level, let's say $50cm$, and the computation time should be as less as possible. To reach that level, $L = 100$ is an appropriate choice. For an one-dimensional linear model, $L$ can be chose larger and that doesn't change too much.

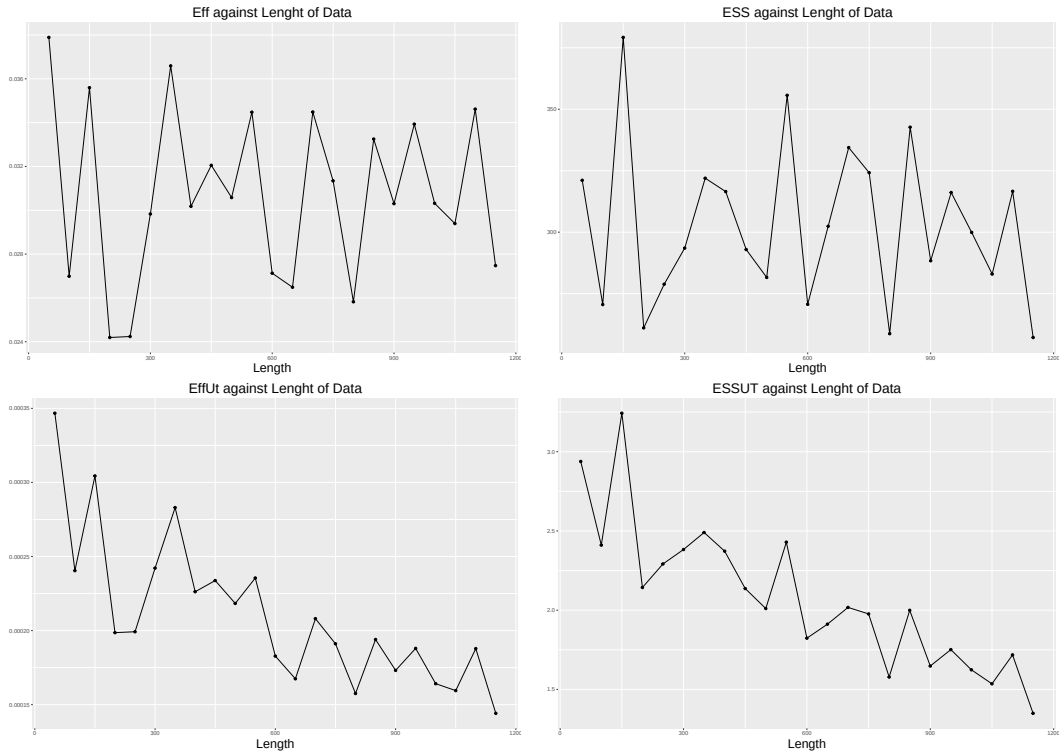



Figure 5.11: Comparison Eff, ESS, EffUT and ESSUT of different length of data.

If the data up to time $t$ is less than or equal to the chosen $L$, the whole data set is used in learning $\theta$ and estimating $X_t$.

For the true posterior, the algorithm requires a cheap estimation $\hat{\pi}(\cdot)$, which is found by one-variable-at-a-time Metropolis-Hastings algorithm. The advantage is getting a precise estimation of the parameter structure, and disadvantage is, obviously, lower efficiency. Luckily, we find that it is not necessary to run this MH every time when estimate a new state from $x_{t-1}$ to $x_t$. In fact, in DA-MH process, the cheap $\hat{\pi}$ doesn't vary too much in the filtering process with new data coming into the dataset. We may use this property in the algorithm. At first, we use all available data from 1 to $t$ with length up to $L$ to learn the structure of $\theta$ and find out the cheap approximation $\hat{\pi}$. Then, use DA-MH to estimate the true posterior $\pi$ for $\theta$ and $x_t$. After that, extend dataset to $1 : t + 1$ if $t \leq L$ or shift the data window to $2 : t + 1$ if $t > L$ and run DA-MH again to estimate $\theta$ and $x_{t+1}$. From figures (B.3) and (B.4), we can see that the main features and parameters in the estimating process between using batch and sliding window methods have not significant differences.

To avoid estimation bias in the algorithm, we are introducing *threshold* and *cut off* processes. *threshold* means when a bias occurs in the algorithm, the cheap $\hat{\pi}$ may not be appropriate and a new one is needed. Thus, we have to update $\hat{\pi}$ with a latest data

we have. A *cut off* process stops the algorithm when a large $\Delta_t$ happens. A large time gap indicates the vehicle stops at some time point and it causes irregularity and bias. A smart way is stopping the process and waiting for new data coming in. By running testings on real data, the *threshold* is chosen $\alpha_2 < 0.7$ and *cut off* is $\Delta_t \geq 300$ seconds. These two values are on researchers' choice. From figures (5.12) and ( 5.13), we can see that by using the *threshold*, we are efficiently avoiding bias and getting more effective samples.

So far, the complete algorithm is summarized as follows:

**Algorithm 2:** Sliding Window MCMC.

1 Initialization: Set up $L$, *threshold* and *cut off* criteria.

2 Learning process: Estimate $\theta$ with $p(\theta \mid Y_{1:\min\{t,L\}}) \propto p(Y_{1:\min\{t,L\}} \mid \theta)p(\theta)$ by one-variable-at-a-time Random Walk Metropolis-Hastings algorithm gaining the target acceptance rates and find out the structure of $\theta \sim N(\mu, \Sigma)$ and the approximation $\hat{\pi}(\cdot)$.

3 Estimate $X_{\max\{1,t-L+1\}:\min\{t,L\}}$ with $Y_{\max\{1,t-L+1\}:\min\{t,L\}}$: **for** *i from 1 to N* **do**

4 Propose $\theta_i^*$ from $N(\theta_i \mid \mu, \Sigma)$, accept it with probability $\alpha_1 = \min\left\{1, \frac{\hat{\pi}(\theta_i^*)q(\theta_i,\theta_i^*)}{\hat{\pi}(\theta_i)q(\theta_i^*,\theta_i)}\right\}$ and go to next step; otherwise go to step 4.

5 Accept $\theta_i^*$ with probability $\alpha_2 = \min\left\{1, \frac{\pi(\theta_i^*)\hat{\pi}(\theta_i)}{\pi(\theta_i)\hat{\pi}(\theta_i^*)}\right\}$ and go to next step; otherwise go to step 4.

6 Calculate $\mu_i^{(t)}, \Sigma_i^{(t)}$ for $X_t$ and $\mu_i^{(t+s)}, \Sigma_i^{(t+s)}$ for $X_{t+s}$.

7 **end**

8 Calculate $\mu_X^{(t)} = \frac{1}{N}\sum_i \mu_i^{(t)}$, $\texttt{Var}(X^{(t)}) = \frac{1}{N}\sum_i(\mu_i^{(t)}\mu_i^{(t)\top} + \Sigma_i) - \frac{1}{N^2}(\sum_i \mu_i^{(t)})(\sum_i \mu_i^{(t)})^\top$ and $\mu_X^{(t+s)}$, $\texttt{Var}(X^{(t+s)})$ in the same formula.

9 Check *threshold* and *cut off* criteria. **if** *threshold is TRUE* **then**

10 Update $\theta \sim N(\mu, \Sigma)$

11 **else if** *cut off is TRUE* **then**

12 Stop process.

13 **else**

14 Go to next step.

15 **end**

16 Shift the window by setting $t = t + 1$ and go back to step 3.

### 5.5.7 Implementation

To implement the algorithm (2), firstly we should get an idea of how the hyper parameter space looks like by running step 2 of the algorithm with some observed data. By setting $L = 100$ and running 5 000 iterations, we can find the whole $\theta$ samples in 59 seconds. For each parameter of $\theta$, we take 1 000 sub-samples out of 5 000 as new sequences. The new $\theta^*$ is representative for the hyper parameter space. Then the traces and correlation is derived from $\theta^*$. Meanwhile, the acceptance rates for each parameter are $\alpha_\gamma = 0.453, \alpha_{\xi^2} = 0.433, \alpha_{\lambda^2} = 0.435, \alpha_{\sigma^2} = 0.414, \alpha_{\tau^2} = 0.4490$ respectively. Hence, the structure of $\hat{\theta} \sim N(m_t, C_t)$ is achieved. That can be seen in figure (5.14).

Since a cheap surrogate $\hat{\pi}(\cdot)$ for the true $\pi(\cdot)$ is found in step 2, it is time to move to the next step. Algorithm (2) takes fixed $L$ length data from $Y_{1:L}$ to $Y_{t-L+1:t}$ until an irregular large time lag meets the *cut off* criterion. In the implementation, the first *cut off* occurs at $t = 648$th data point. The first estimated $\hat{X}_{1:L}$ was found by the batch method and $\hat{X}_{L+1}$ to $\hat{X}_t$ were found sequentially around 9 seconds with 10 000 iterations each time.

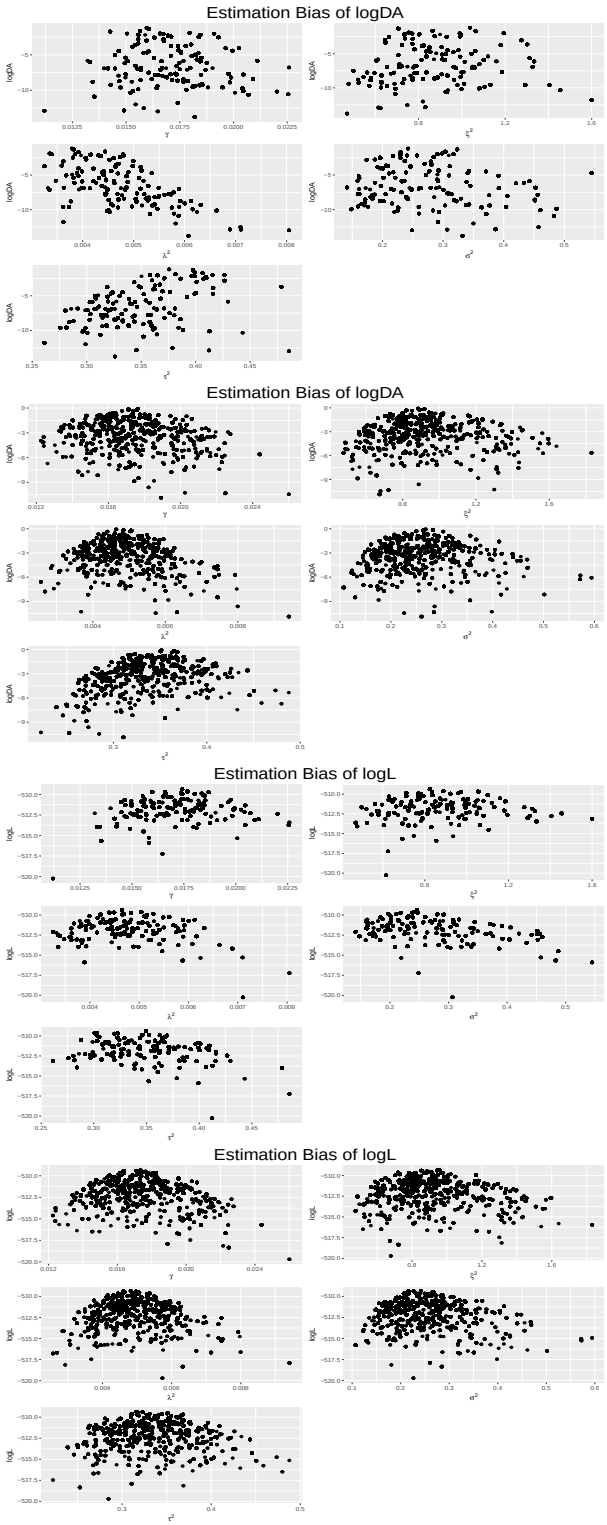



Figure 5.12: Comparison $\ln DA$ and $\ln L$ between not-updating and updating mean methods.

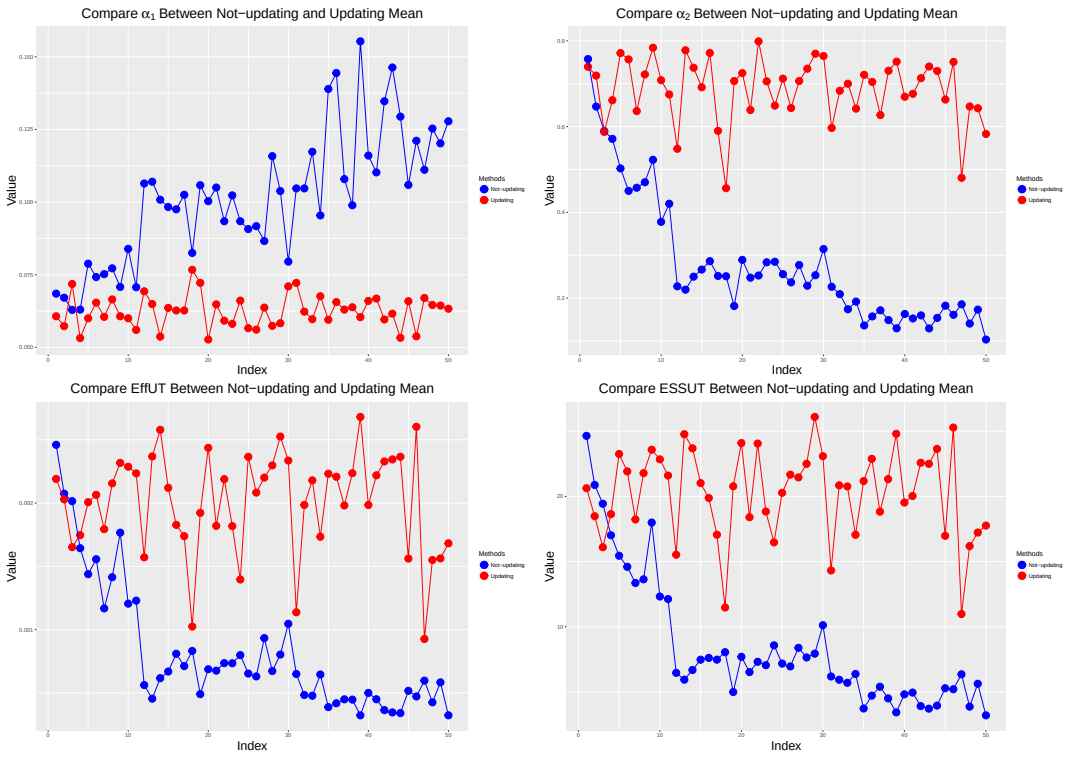



Figure 5.13: Comparison between not-updating and updating mean methods.

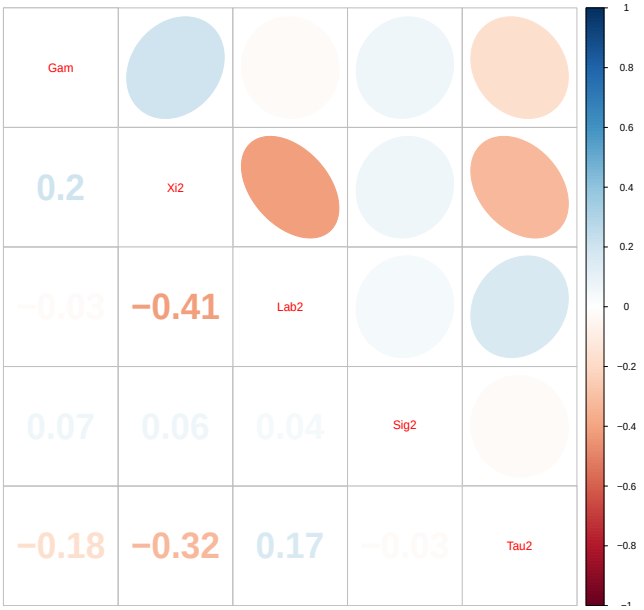



Figure 5.14: Visualization of correlation matrix of $\theta$, which is found in learning-surface process.

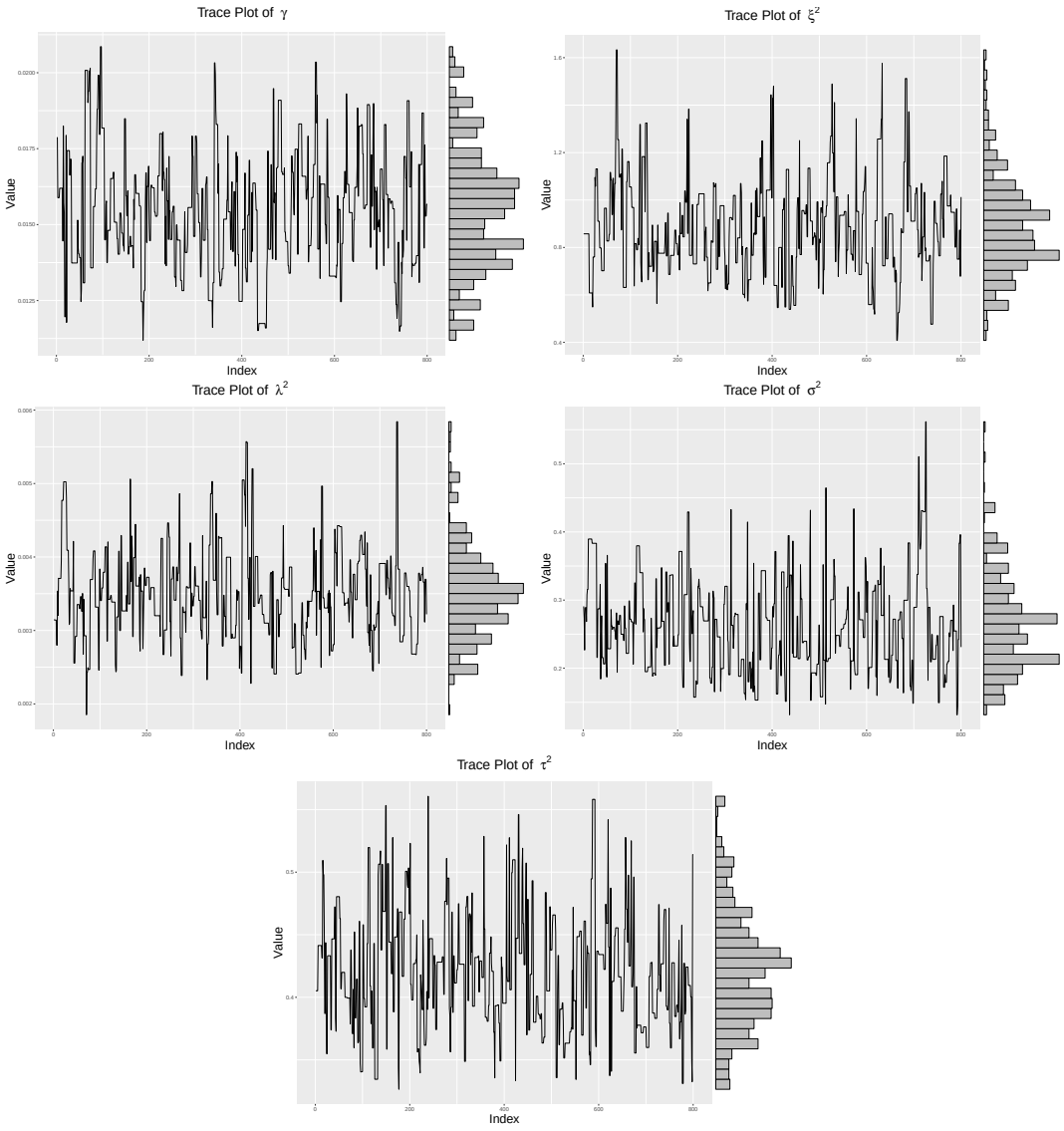



Figure 5.15: Trace plots of $\theta$ from learning surface process after taking 1 000 samples from 5 000.

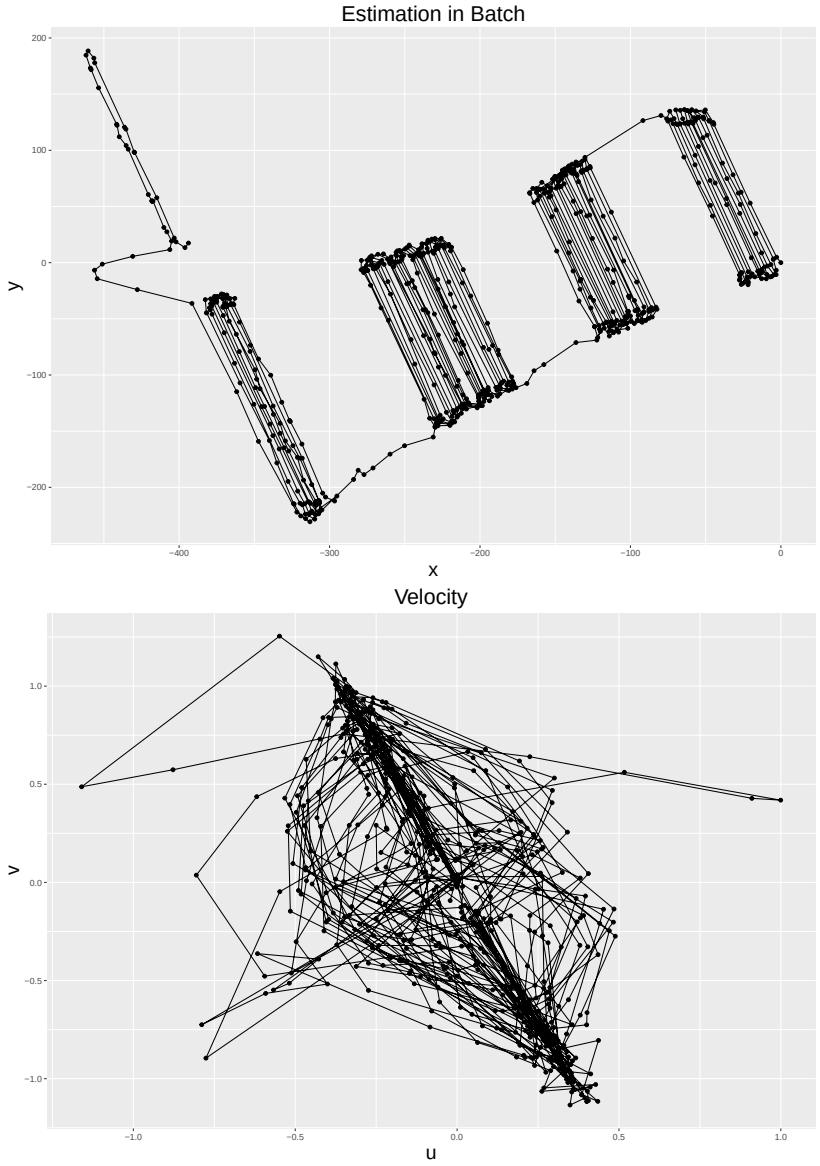



Figure 5.16: Position and velocity for $X$ and $Y$ found by combined batch and sequential methods.

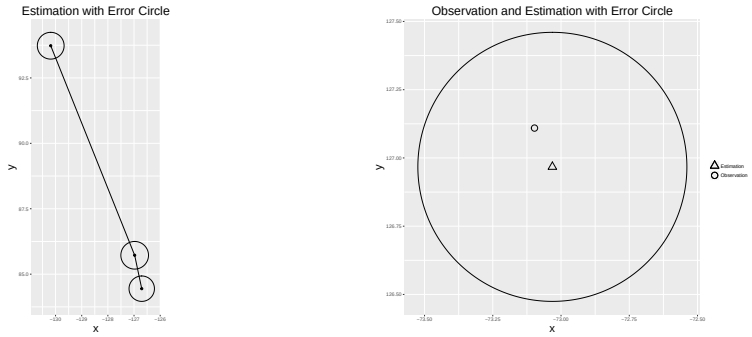



Figure 5.17: Zoom in on estimations. For each estimation $\hat{X}_i (i = 1, \ldots, t)$, there is a error circle around it.

## 5.6 Discussion and Future Work

In this paper, we are using the a self-tuning one-variable-at-a-time Metropolis-Hastings Random Walk to learn the parameter hyper space for a linear state space model. Starting from the joint covariance and distribution of $X$ and $Y$, we have a recursive way to update the mean and covariance sequentially. After getting the cheap approximation posterior distribution, Delayed-Acceptance Metropolis-Hastings algorithm accelerates the estimating process. The advantage of this algorithm is that it is easily to understand and implement in practice. Contrast, Particle Learning algorithm is high efficient but the sufficient statistics are not available all the time .

Some future work can be done on inferring state from precious movement with other kinetic information, not just with diffusive velocity. Besides, I am more interested in increase the efficiency and accuracy of MCMC method.

# Chapter 6

# Data Simplify

## 6.1 Introduction

GPS devices are widely used in orchard planting and maintenance. This location-based system allows orchardist to check trajectory of tractors.

Usually, GPS units record more data than necessary, and causes more errors due to the shelter from branches.

Local simplification algorithms. This algorithm focuses on a couple of particular consecutive points. By analyzing the relationship among these points, a decision is made that which point can be deleted or retained. Distance threshold algorithm is one of these algorithms. All points, for which the distance to the preceding track point is less than a predetermined threshold is deleted. Direction changing algorithm is another one. The point is retained if the change in direction is greater than a predetermined threshold Ivanov (2012).

Global simplification algorithms. These algorithms have an overview of all tracked points. After analyzing the relationships among these points, a decision will be made about which one or more points to delete or retain. The Douglas-Peucker algorithm is one of them Douglas and Peucker (1973).

It is apparently that global simplification algorithms can be used on off-line data analyses and local simplification algorithms will perform better on on-line or real-time track simplification.

However, a pertinent algorithm is required in our case.

## 6.2 Preliminary

In this section, we elaborate an algorithm used in simplifying global trajectories.

Trajectory is a connection by a time series successive positions recorded by GPS devices. A classical GPS device records skeleton information, including time stamp, latitude, longitude, number of available satellites, etc. Recently, researchers try to enrich trajectory (called Semantic Trajectory) by adding background geographic information to discover meaningful pattern Ying *et al.* (2011). A TS simplification method, represented in Chen *et al.* (2009), consider both the skeleton information and semantic meanings of a trajectory when performing simplification.

In our case, a GPS log is a sequence time series GPS points $p_i \in P$, $P = \{p_1, p_2, \cdots, p_n\}$. Each GPS point $p_i$ contains information of time stamp, latitude and longitude, and semantic information of velocity, heading direction and boom status, which can be written in form of

$$T = \{p_t = [x_t, y_t, v_t, \theta_t, b_t] | t \in \mathbb{R}\}. \tag{6.1}$$

Sequentially connect these points will give us a tractor trajectory.

For a tractor working on an orchard, there are two status of a boom, working and not working. These information is recorded by GPS units, and using $b = 1$ to represented for working and $b = 0$ for not working.

**Segment** A segment is a part of consecutive trajectory. Regarding to the status of boom, trajectory can be simply divided into two kinds of segment in our dataset, one is boom-working, the other is boom-not-working.

**Direction**. Direction $\theta$ denotes the heading direction of a tractor at a specific point location. This parameter uses north direction as a basis, in which way $0° \leq \theta < 360°$.

## 6.3 Track Simplification Algorithm

The first two steps are initial simplification to reduce some errors caused by misoperation and GPS units bugs.

- Step 1. Merging. If the length of a segment composed by consecutive boom working or not-working points is less than a threshold, merge this one into its backward segment.

- Step 2. Remove time stamp duplicated data.

Now only two types of segment points are left in GPS log, boom working and not-working, and the length of each segment is greater than the predetermined threshold.

The following algorithm is based on the relationship between a candidate point $p_i$ and it's neighboring points $p_{i-1}$ and $p_{i+1}$, and the importance of the $p_i$ in the segment where it belongs to.

- Rule 1. The candidate point $p_i$ is retained if it is not linear predictable or cannot be used for linear predicting. With the velocity information $v_{i-1}, v_i$ at point $p_{i-1}, p_i$ and time differences $\Delta t_{i-1} = |t_i - t_{i-1}|, \Delta t_i = |t_{i+1} - t_i|$, an estimated position can be calculated by $\hat{p}_i = \Delta t_{i-1} p_{i-1}$, $\hat{p}_{i+1} = \Delta t_i p_i$. If the distance $|\hat{p}_i - p_i|$ or $|\hat{p}_{i+1} - p_{i+1}|$ is less than a threshold, then the point $p_i$ is not linear predictable or cannot be used for linear predicting.

- Rule 2. Select a candidate point $p_i$. Retain this point if the distance between $p_i$ and $p_{i-1}$ is greater than the threshold $d$, where $d$ is the mean distances of these points $p_{i-1}, p_i, \cdots, p_{i+k}$ with same boom status $b_{i-1} = b_i = \cdots = b_{i+k}$.

- Rule 3. Neighbor Heading Changing. The candidate point $p_i$ belongs to the track if $|\theta_i - \theta_{i-1}| + |\theta_i - \theta_{i+1}| > \theta$, where $|\theta_i - \theta_{i-1}|$ and $|\theta_i - \theta_{i+1}|$ are the direction changes between points $p_i$ and $p_{i-1}$ and between points $p_i$ and $p_{i+1}$, $\theta$ is predefined threshold.

- Rule 4. The candidate point $p_i$ belongs to the track if the boom status $b_i \neq b_{i-1}$.

Finally, the point $p_i$ belongs to the track if Rule 1 = TRUE or Rule 2 = TRUE or Rule 3 = TRUE or Rule 4 = TRUE.

## 6.4 Evaluation

Errors are measured by Synchronized Euclidean Distance Lawson *et al.* (2011). SED measures the distances between the original and compressed trace at the same time. As shown in figure 1Lawson *et al.* (2011). $P_{t1}, \cdots, P_{t5}$ are original points. After simplification, the points $P_{t2}, P_{t3}$ and $P_{t4}$ were removed. The black curve is the original trajectory, in contrast, gray dash line is the simplified trajectory. The gray point $P'_{t2}$ on simplified trajectory has the same time difference as the point $P_{t2}$ on original trajectory does. Then the distance between $P_{t2}$ and $P'_{t2}$ is calculated.

Another way to calculate the difference between a GPS trace and its compressed version is to measure the perpendicular distance. This algorithms ignore the temporal

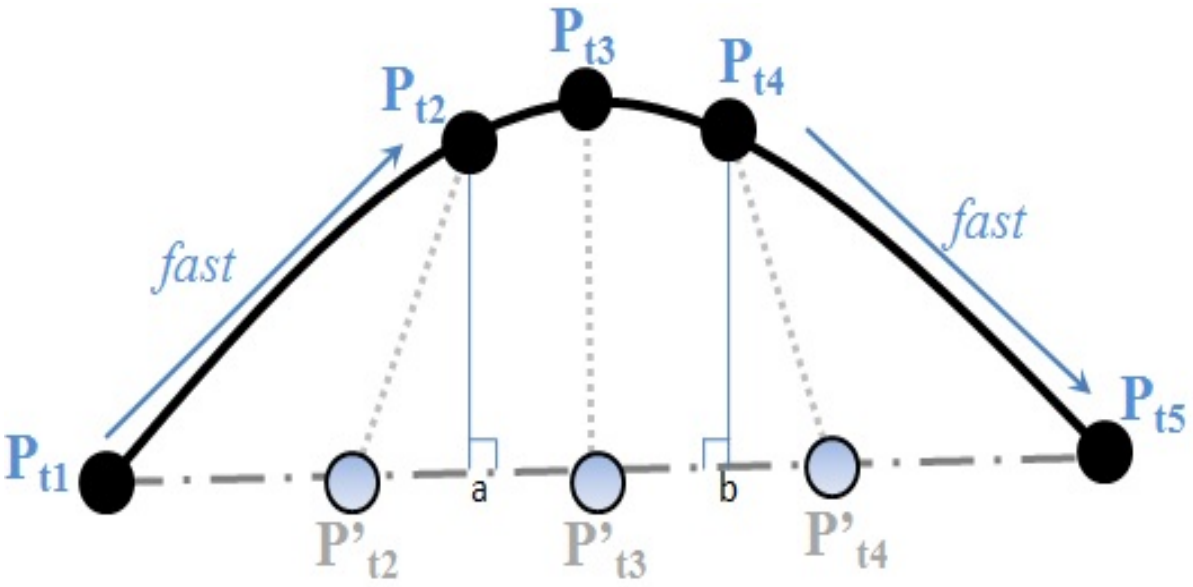



Figure 6.1: Synchronized Euclidean Distance

component and use simple perpendicular distance. The figure 2 Meratnia and Rolf (2004) expresses these difference clearly.

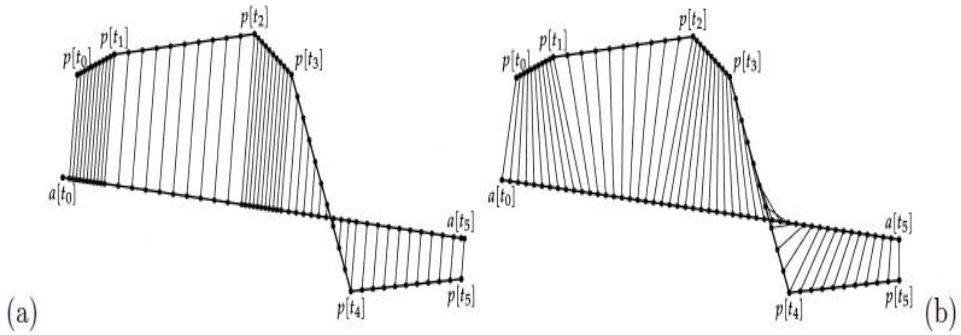



Figure 6.2: (a) error measured at fixed sampling rate as sum of perpendicular distance chords; (b) error measured at fixed sampling rates as sum of time-synchronous distance chords.

## 6.5 Adaptive Kalman Filter

Discrete Kalman Filter equations the describe the prediction step are

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$
$$P_k^- = AP_{k-1}A^\top + Q$$

where $\hat{x}_k^-$ is a priori state estimate, $\hat{x}_k$ is a posteriori state estimate, $A$ is status transition matrix, $P_k^-$ is a priori estimate for error covariance, $u_k$ is an input parameter and $Q$ is process noise covariance.

Discrete Kalman Filter equations the describe the correct step are

$$K_k = P_k^- H^\top (HP_k^- H^\top + R)^{-1}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$
$$P_k = (I - K_k H)P_k^-$$

where $K_k$ is the Kalman gain matrix, $z_k$ is the observed data.

## 6.6 Experimental results

Our original data set has 1021 rows, each of them contains latitude, longitude, velocity, heading direction and boom information. Douglas-Peucker Algorithm, with distance threshold $0.205m$, retained 847 points. Tractor Algorithm, given a predictable distance being $5m$ and heading direction changing being 30°, returned the same amount of points. Under the same circumstance, we calculated SED and other information.

Table 6.1: Error Comparing

| | **Original Data** | **DP Algorithm** | **Tractor Algorithm** |
|---|---|---|---|
| **Number of Points** | 1021 | 847 | 847 |
| **Tracked Distances**(m) | 74041.31 | 74038.33 | 74012.56 |
| **SED** (m) | NA | 1316.715 | 607.9587 |

Table (6.1) describe the results after simplifying with DP algorithm and Tractor Algorithm.

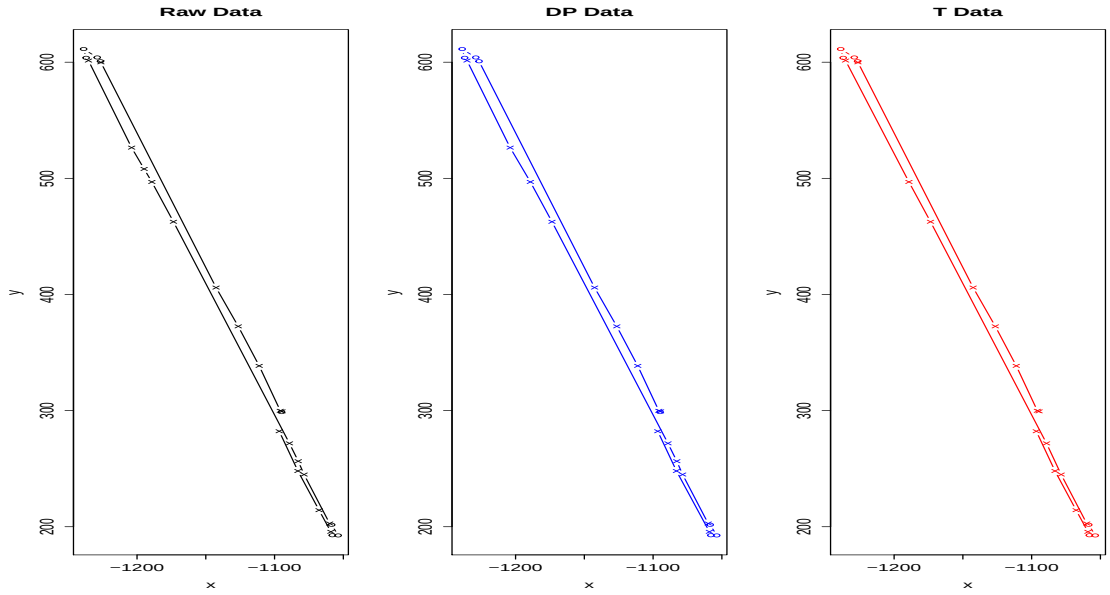



Figure 6.3: A segment start from time 2000 to 3000, recorded by GPS units. On the left side, it's the trajectory connected by raw data with 27 points. In the middle, it's the trajectory connected by simplified data with Douglas-Peucker Algorithm with 24 points. On the right side, it's the trajectory connected by simplified data with Tractor Simplification Algorithm with 23 points.

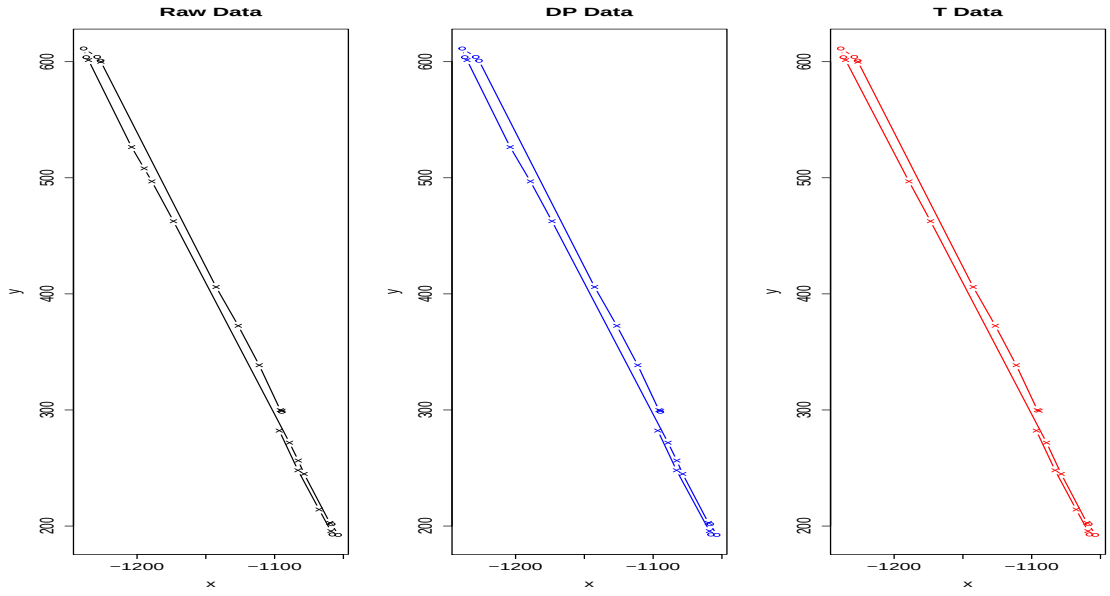



Figure 6.4: Trajectory fitted by Adaptive Kalman Filter. The errors of raw data, DP and tractor algorithm caused by AKF is 26.89217, 23.97877 and 23.97097 respectively.

# Chapter 7

# Future Work

# Chapter 8

# Summary

# Appendices

# Appendix A

# Spline Penalty

## A.1 Penalty Matrix in (2.16)

The $k$-th $\Omega^{(k)}$ is a $2n \times 2n$ matrix in the form of

$$\begin{aligned}
\Omega^{(k)}_{2k-1,2k-1} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k)}_{00}(t)}{dt^2} \frac{d^2 h^{(k)}_{00}(t)}{dt^2} dt = \frac{12}{\Delta_k^3} \\
\Omega^{(k)}_{2k-1,2k} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k)}_{00}(t)}{dt^2} \frac{d^2 h^{(k)}_{10}(t)}{dt^2} dt = \frac{6}{\Delta_k^2} \\
\Omega^{(k)}_{2k-1,2k+1} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k)}_{00}(t)}{dt^2} \frac{d^2 h^{(k+1)}_{01}(t)}{dt^2} dt = \frac{-12}{\Delta_k^3} \\
\Omega^{(k)}_{2k-1,2k+2} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k)}_{00}(t)}{dt^2} \frac{d^2 h^{(k+1)}_{11}(t)}{dt^2} dt = \frac{6}{\Delta_k^2} \\
\Omega^{(k)}_{2k,2k} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k)}_{10}(t)}{dt^2} \frac{d^2 h^{(k)}_{10}(t)}{dt^2} dt = \frac{4}{\Delta_k} \\
\Omega^{(k)}_{2k,2k+1} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k)}_{10}(t)}{dt^2} \frac{d^2 h^{(k+1)}_{01}(t)}{dt^2} dt = \frac{-6}{\Delta_k^2} \\
\Omega^{(k)}_{2k,2k+2} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k)}_{10}(t)}{dt^2} \frac{d^2 h^{(k+1)}_{11}(t)}{dt^2} dt = \frac{2}{\Delta_k} \\
\Omega^{(k)}_{2k+1,2k+1} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k+1)}_{01}(t)}{dt^2} \frac{d^2 h^{(k+1)}_{01}(t)}{dt^2} dt = \frac{12}{\Delta_k^3} \\
\Omega^{(k)}_{2k+1,2k+2} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k+1)}_{01}(t)}{dt^2} \frac{d^2 h^{(k+1)}_{11}(t)}{dt^2} dt = \frac{-6}{\Delta_k^2} \\
\Omega^{(k)}_{2k+2,2k+2} &= \int_{t_k}^{t_{k+1}} \frac{d^2 h^{(k+1)}_{11}(t)}{dt^2} \frac{d^2 h^{(k+1)}_{11}(t)}{dt^2} dt = \frac{4}{\Delta_k}
\end{aligned}$$

$k = 1, 2, \cdots, n-1$. It's a bandwidth four matrix. Then

$$\mathbf{\Omega} = \sum_{k=1}^{n-1} \Omega^{(k)}$$

## A.2 Proof of Theorem 2

*Proof.* It is obviously that every basis functions are continuous on subinterval $[t_k, t_{k+1}]$. We firstly prove that these basis functions are independent. We have $2n$ basis functions and $n$ knots. Then choose $t_1, \frac{t_1+t_2}{2}, t_2, \frac{t_2+t_3}{2}, \cdots, t_{n-1}, \frac{t_{n-1}+t_n}{3}, \frac{2(t_{n-1}+t_n)}{3}, t_n$ as new $2n$ knots, and denoted by $c_1, c_2, \cdots, c_{2n}$. Then the determinant is

$$\begin{aligned} D(c_1, c_2, \cdots, c_{2n}) &= \begin{vmatrix} N_1(c_1) & N_1(c_2) & \cdots & N_1(c_{2n}) \\ N_2(c_1) & N_2(c_2) & \cdots & N_2(c_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ N_{2n}(c_1) & N_{2n}(c_2) & \cdots & N_{2n}(c_{2n}) \end{vmatrix} \\ &= \begin{vmatrix} 1 & a_{12} & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_{22} & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_{32} & 1 & a_{34} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{2n-1,2n-1} & 1 \\ 0 & 0 & 0 & 0 & \cdots & a_{2n,2n-1} & 0 \end{vmatrix}, \end{aligned} \tag{A.1}$$

$$\text{where } D(c_1, c_2, \cdots, c_{2n}) = \begin{cases} N_1(t_1) = 1 & \\ N_1(\frac{t_1+t_2}{2}) = a_{12} & \\ N_2(t_1) = 0 & \\ N_2(\frac{t_1+t_2}{2}) = a_{22} & \\ N_{2k+1}(\frac{t_k+t_{k+1}}{2}) = a_{2k+1,2k} & k = 1, 2, \cdots, 2n \\ N_{2k+1}(t_{k+1}) = 1 & k = 1, 2, \cdots, 2n \\ N_{2k+1}(\frac{t_{k+1}+t_{k+2}}{2}) = a_{2k+1,2k+2} & k = 1, 2, \cdots, 2n \\ N_{2k+2}(\frac{t_k+t_{k+1}}{2}) = a_{2k+2,2k} & k = 1, 2, \cdots, 2n \\ N_{2k+2}(\frac{t_{k+1}+t_{k+2}}{2}) = a_{2k+2,2k+2} & k = 1, 2, \cdots, 2n, \\ N_{2n-1}(t_{2n-1}) = 0 & \\ N_{2n-1}(\frac{t_{2n-1}+t_{2n}}{3}) = a_{2n-1,2n-2} & \\ N_{2n-1}(\frac{2(t_{2n-1}+t_{2n})}{3}) = a_{2n-1,2n-1} & \\ N_{2n-1}(t_{2n}) = 1 & \\ N_{2n}(\frac{t_{2n-1}+t_{2n}}{3}) = a_{2n,2n-2} & \\ N_{2n}(\frac{2(t_{2n-1}+t_{2n})}{3}) = a_{2n,2n-1} & \\ N_{2n}(t_{2n}) = 0 & \\ 0 & otherwise \end{cases}$$

and $a_{ij} \neq 0$. After decomposing determinant $D$ in equation (A.1), gives

$$\begin{aligned} \det D &= \begin{vmatrix} a_{22} & 0 & 0 & \cdots & 0 & 0 \\ a_{32} & 1 & a_{34} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{2n-1,2n-1} & 1 \\ 0 & 0 & 0 & \cdots & a_{2n,2n-1} & 0 \end{vmatrix} = a_{22} \begin{vmatrix} 1 & a_{34} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{2n-1,2n-1} & 1 \\ 0 & 0 & \cdots & a_{2n,2n-1} & 0 \end{vmatrix} \\ &= \cdots = a_{22}a_{44}\cdots a_{2n-4,2n-4} \begin{vmatrix} a_{2n-2,2n-2} & a_{2n-2,2n-1} & 0 \\ a_{2n-1,2n-2} & a_{2n-1,2n-1} & 1 \\ a_{2n,2n-2} & a_{2n,2n-1} & 0 \end{vmatrix} \\ &= a_{22}a_{44}\cdots a_{2n-4,2n-4}(a_{2n-2,2n-1}a_{2n,2n-2} - a_{2n,2n-1}a_{2n-2,2n-2}) \neq 0. \end{aligned}$$

With the conclusion in Lemma 1, $N_1(t), \ldots, N_{2n}(t)$ are linearly independent on interval $[t_1, t_n]$. Secondly, we prove that basis functions represent any cubic function on each interval $[t_k, t_{k+1}]$. Due to the definition of cubic spline, on interval $[t_k, t_{k+1}]$, a cubic

spline $g(t)$ can be written in the form of

$$g(t) = d_k(t-t_k)^3 + c_k(t-t_k)^2 + b_k(t-t_k) + a_k, \text{for } t_k \le t \le t_{k+1} \tag{A.2}$$

For any $f(t)$ on $[t_1, t_n]$, it can be represented as $f(t) = \sum_{k=1}^{2n} \theta_k N_k(t)$. Then for $\forall t \in [t_k, t_{k+1}]$, we have

$$f(t) = \begin{cases} \theta_{2k-1}N_{2k-1}(t) + \theta_{2k}N_{2k}(t) + \theta_{2k+1}N_{2k+1}(t) + \theta_{2k+2}N_{2k+3}(t), & t_k \le t \le t_{k+1} \\ 0, & \text{otherwise} \end{cases},$$

thus

$$\begin{aligned} f(t) =& \theta_{2k-1}\{2(\frac{t-t_k}{t_{k+1}-t_k})^3 - 3(\frac{t-t_k}{t_{k+1}-t_k})^2 + 1\} \\ &+\theta_{2k}\{\frac{(t-t_k)^3}{(t_{k+1}-t_k)^2} - 2\frac{(t-t_k)^2}{t_{k+1}-t_k} + (t-t_k)\} \\ &+\theta_{2k+1}\{-2(\frac{t-t_k}{t_{k+1}-t_k})^3 + 3(\frac{t-t_k}{t_{k+1}-t_k})^2\} + \theta_{2k+2}\{\frac{(t-t_k)^3}{(t_{k+1}-t_k)^2} - \frac{(t-t_k)^2}{t_{k+1}-t_k}\}. \end{aligned}$$

After rearranging, we have

$$\begin{aligned} f(t) =& \{\frac{2\theta_{2k-1}}{(t_{k+1}-t_k)^3} + \frac{\theta_{2k}}{(t_{k+1}-t_k)^2} - \frac{2\theta_{2k+1}}{(t_{k+1}-t_k)^3} + \frac{\theta_{2k+2}}{(t_{k+1}-t_k)^2}\}(t-t_k)^3 \\ &+\{-\frac{3\theta_{2k-1}}{(t_{k+1}-t_k)^2} - \frac{2\theta_{2k}}{(t_{k+1}-t_k)} + \frac{3\theta_{2k+1}}{(t_{k+1}-t_k)^2} - \frac{\theta_{2k+2}}{(t_{k+1}-t_k)}\}(t-t_k)^2 \\ &+\theta_{2k}(t-t_k) + \theta_{2k-1} \end{aligned}$$

where coefficients are

$$\begin{cases} \theta_{2k-1} = a_k \\ \theta_{2k} = b_k \\ -\frac{3\theta_{2k-1}}{(t_{k+1}-t_k)^2} - \frac{2\theta_{2k}}{(t_{k+1}-t_k)} + \frac{3\theta_{2k+1}}{(t_{k+1}-t_k)^2} - \frac{\theta_{2k+2}}{(t_{k+1}-t_k)} = c_k \\ \frac{2\theta_{2k-1}}{(t_{k+1}-t_k)^3} + \frac{\theta_{2k}}{(t_{k+1}-t_k)^2} - \frac{2\theta_{2k+1}}{(t_{k+1}-t_k)^3} + \frac{\theta_{2k+2}}{(t_{k+1}-t_k)^2} = d_k \end{cases}$$

the resulting can always be solved for $\theta_{2k-1}, \theta_{2k}, \theta_{2k+1}, \theta_{2k+2}$ in terms of $a_k, b_k, c_k, d_k$ on interval $[t_k, t_{k+1}]$. So cubic spline on each interval can be represented by basis functions.

Finally, we will prove basis functions are continuous on $[t_1, t_n]$. For any knot $t_k$, where $t_1 < t_k < t_n$, it is known that $f(t_k) = \theta_{2k-1}$. Moreover,

$$\lim_{t \to t_k+} f(t) = \lim_{t \to t_k+} (\theta_{2k-1}N_{2k-1}(t) + \theta_{2k}N_{2k}(t) + \theta_{2k+1}N_{2k+1}(t) + \theta_{2k+2}N_{2k+3}(t)) = \theta_{2k-1},$$

$$\lim_{t \to t_k-} f(t) = \lim_{t \to t_k-} (\theta_{2k-1}N_{2k-1}(t) + \theta_{2k}N_{2k}(t) + \theta_{2k+1}N_{2k+1}(t) + \theta_{2k+2}N_{2k+3}(t)) = \theta_{2k-1}.$$

So

$$\lim_{t \to t_k +} f(t) = \lim_{t \to t_k -} f(t) = f(t),$$

$f(t)$ is continuous at knots, and then continuous on whole interval $[t_1, t_n]$. $f(t)$ is a continuous cubic spline on interval $[t_1, t_n]$, then $f(t)$ has continuous first and second derivatives. □

## A.3 Proof of Lemma 2

*Proof.* For any spline $f$,

$$\begin{aligned}
&\frac{1}{n}\sum_{j=1}^{n}(y_j^* - f(t_j))^2 + \frac{\gamma}{n}\sum_{j=1}^{n}(v_j^* - f'(t_j)) + \lambda\int f''^2 \\
\geq &\frac{1}{n}\sum_{j\neq i}^{n}(y_j^* - f(t_j))^2 + \frac{\gamma}{n}\sum_{j\neq i}^{n}(v_j^* - f'(t_j)) + \lambda\int f''^2 \\
\geq &\frac{1}{n}\sum_{j\neq i}^{n}(y_j^* - \hat{f}^{(-i)}(t_j))^2 + \frac{\gamma}{n}\sum_{j\neq i}^{n}(v_j^* - \hat{f}'^{(-i)}(t_j)) + \lambda\int \hat{f}^{(-i)''2} \\
= &\frac{1}{n}\sum_{j=1}^{n}(y_j^* - \hat{f}^{(-i)}(t_j))^2 + \frac{\gamma}{n}\sum_{j=1}^{n}(v_j^* - \hat{f}'^{(-i)}(t_j)) + \lambda\int \hat{f}^{(-i)''2}
\end{aligned} \tag{A.3}$$

by the definition of $\hat{f}^{(-i)}$, $\hat{f}'^{(-i)}$ and $y_i^* = \hat{f}^{(-i)}(t_i)$, $v_i^* = \hat{f}'^{(-i)}(t_i)$. It follows that $\hat{f}^{(-i)}$ is the minimizer of the MSE function (*****), so that

$$\hat{\mathbf{f}}^{(-i)} = S\mathbf{y}^* + \gamma T\mathbf{v}^* \tag{A.4}$$

$$\hat{\mathbf{f}}'^{(-i)} = U\mathbf{y}^* + \gamma V\mathbf{v}^* \tag{A.5}$$

□

## A.4 Proof of Theorem 3

*Proof.*

$$\begin{aligned}\hat{f}^{(-i)}(t_i) - y_i =& \sum_{j=1}^{n} S_{ij} y_j^* + \gamma \sum_{j=1}^{n} T_{ij} v_j^* - y_i^* \\ =& \sum_{j\neq i}^{n} S_{ij} y_j + \gamma \sum_{j\neq i}^{n} T_{ij} v_j + S_{ii}\hat{f}^{(-i)}(t_i) + \gamma T_{ii}\hat{f}'^{(-i)}(t_i) - y_i \\ =& \sum_{j=1}^{n} S_{ij} y_j + \gamma \sum_{j=1}^{n} T_{ij} v_j + S_{ii}(\hat{f}^{(-i)}(t_i) - y_i) + \gamma T_{ii}(\hat{f}'^{(-i)}(t_i) - v_i) - y_i \\ =& (\hat{f}(t_i) - y_i) + S_{ii}(\hat{f}^{(-i)}(t_i) - y_i) + \gamma T_{ii}(\hat{f}'^{(-i)}(t_i) - v_i).\end{aligned} \tag{A.6}$$

Additionally,

$$\begin{aligned}\hat{f}'^{(-i)}(t_i) - v_i =& \sum_{j=1}^{n} U_{ij} y_j^* + \gamma \sum_{j=1}^{n} V_{ij} v_j^* - v_i^* \\ =& \sum_{j\neq i}^{n} U_{ij} y_j + \gamma \sum_{j\neq i}^{n} V_{ij} v_j + U_{ii}\hat{f}^{(-i)}(t_i) + \gamma V_{ii}\hat{f}'^{(-i)}(t_i) - v_i \\ =& \sum_{j=1}^{n} U_{ij} y_j + \gamma \sum_{j=1}^{n} V_{ij} v_j + U_{ii}(\hat{f}^{(-i)}(t_i) - y_i) + \gamma V_{ii}(\hat{f}'^{(-i)}(t_i) - v_i) - v_i \\ =& (\hat{f}'(t_i) - v_i) + U_{ii}(\hat{f}^{(-i)}(t_i) - y_i) + \gamma V_{ii}(\hat{f}'^{(-i)}(t_i) - v_i).\end{aligned} \tag{A.7}$$

Then

$$\hat{f}'^{(-i)}(t_i) - v_i = \frac{\hat{f}'(t_i) - v_i}{1 - \gamma V_{ii}} + \frac{U_{ii}(\hat{f}^{(-i)}(t_i) - y_i)}{1 - \gamma V_{ii}}. \tag{A.8}$$

After substituting equation (A.8) into (A.6), we get

$$\hat{f}^{(-i)}(t_i) - y_i = \frac{\hat{f}(t_i) - y_i + \gamma \frac{T_{ii}}{1-\gamma V_{ii}}(\hat{f}'(t_i) - v_i)}{1 - S_{ii} - \gamma \frac{T_{ii}}{1-\gamma V_{ii}} U_{ii}}. \tag{A.9}$$

Then

$$CV(\lambda, \gamma) = \frac{1}{n}\sum_{i=1}^{n} \frac{\hat{f}(t_i) - y_i + \gamma \frac{T_{ii}}{1-\gamma V_{ii}}(\hat{f}'(t_i) - v_i)}{1 - S_{ii} - \gamma \frac{T_{ii}}{1-\gamma V_{ii}} U_{ii}}. \tag{A.10}$$

□

# Appendix B

# MCMC

## B.1 Appendices

### B.1.1 Linear Simulation Calculations

**Forecast**

Calculating the log-posterior of parameters requires finding out the forecast distribution of $p(y_{1:t} \mid y_{1:t-1}, \theta)$. A general way is using the joint distribution of $y_t$ and $y_{1:t-1}$, which is $p(y_{1:t} \mid \theta) \sim N(0, \Sigma_{YY})$ and following the procedure in section 5.2.2 to work out the inverse matrix of a multivariate normal distribution. For example, one may find the inverse of the covariance matrix

$$\Sigma_{YY}^{-1} = B(I - A^{-1}B) = \frac{1}{\sigma^4}(\sigma^2 I - A^{-1}) \triangleq \frac{1}{\sigma^4}\begin{bmatrix} Z_t & b_t \\ b_t^\top & K_t \end{bmatrix}.$$

Therefore, the original form of this covariance is

$$\Sigma_{YY} = \sigma^4 \begin{bmatrix} (Z - bK^{-1}b^\top)^{-1} & -Z^{-1}b(K - b^\top Z^{-1}b)^{-1} \\ -K^{-1}b^\top(Z - bK^{-1}b^\top)^{-1} & (K - b^\top Z^{-1}b)^{-1} \end{bmatrix}.$$

For sake of simplicity, here we are using $Z$ to represent the $t \times t$ matrix $Z_t$, $b$ to represent the $t \times 1$ vector $b_t$ and $K$ to represent the $1 \times 1$ constant $K_t$. By denoting $C_t^\top = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}$ and post-multiplying $\Sigma_{YY}^{-1}$, it gives us

$$\Sigma_{YY}^{-1} C_t = \frac{1}{\sigma^4}(\sigma^2 I - A^{-1})C_t = \frac{1}{\sigma^4}\begin{bmatrix} b_t \\ K_t \end{bmatrix}. \tag{B.1}$$

By using the formula, one can find a recursive way to update $K_t$ and $b_{t-1}$, which

are

$$K_t = \frac{\sigma^4}{\tau^2 + \sigma^2 + \phi^2(\sigma^2 - K_{t-1})}, \tag{B.2}$$

$$b_t = \begin{bmatrix} \frac{b_{t-1}\phi K_t}{\sigma^2} \\ \frac{K_t(\sigma^2+\tau^2)-\sigma^4}{\phi\sigma^2} \end{bmatrix}. \tag{B.3}$$

With the above formula, the recursive way of updating the mean and covariance are

$$\bar{\mu}_t = \frac{\phi}{\sigma^2} K_{t-1}\bar{\mu}_{t-1} + \phi(1 - \frac{K_{t-1}}{\sigma^2})y_{t-1}, \tag{B.4}$$

$$\bar{\Sigma}_t = \sigma^4 K_t^{-1}, \tag{B.5}$$

where $K_1 = \frac{\sigma^4}{\sigma^2+\tau^2+L^2\phi^2}$. For calculation details, readers can refer to appendices (B.1.1).

By using the formula, one term of equation (5.30) becomes

$$A_t^{-1}C_t = \left(I - \frac{M_t^{-1}u_t u_t^\top}{1 + u_t^\top M_t^{-1} u_t}\right) M_t^{-1} C_t, \tag{B.6}$$

in which

$$M_t^{-1}C_t = \begin{bmatrix} A_{t-1}^{-1} & 0 \\ 0 & \sigma^2 \end{bmatrix} C_t = \sigma^2 C_t,$$

$$u_t^\top C_t = \begin{bmatrix} 0 & \cdots & 0 & \frac{-\phi}{\tau} & \frac{1}{\tau} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \frac{1}{\tau}.$$

Then the above equation becomes

$$A_t^{-1}C_t = \sigma^2 C_t - \frac{M_t^{-1}u_t\frac{\sigma^2}{\tau}}{1 + u^\top M_t^{-1} u}. \tag{B.7}$$

Moreover,

$$M_t^{-1}u_t = \begin{bmatrix} A_{t-1}^{-1} & 0 \\ 0 & \sigma^2 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -\frac{\phi}{\tau} \\ \frac{1}{\tau} \end{bmatrix} = \begin{bmatrix} A_{t-1}^{-1} & 0 \\ 0 & \sigma^2 \end{bmatrix} \begin{bmatrix} -\frac{\phi}{\tau}C_{t-1} \\ \frac{1}{\tau} \end{bmatrix} = \begin{bmatrix} -\frac{\phi}{\tau}A_{t-1}^{-1}C_{t-1} \\ \frac{\sigma^2}{\tau} \end{bmatrix},$$

$$u^\top M_t^{-1}u = \begin{bmatrix} 0 & \cdots & 0 & -\frac{\phi}{\tau} & \frac{1}{\tau} \end{bmatrix} \begin{bmatrix} -\frac{\phi}{\tau}A_{t-1}^{-1}C_{t-1} \\ \frac{\sigma^2}{\tau} \end{bmatrix} = \begin{bmatrix} -\frac{\phi}{\tau}C_{t-1}^\top & \frac{1}{\tau} \end{bmatrix} \begin{bmatrix} -\frac{\phi}{\tau}A_{t-1}^{-1}C_{t-1} \\ \frac{\sigma^2}{\tau} \end{bmatrix}$$

$$= \frac{\phi^2}{\tau^2} C_{t-1}^\top A_{t-1}^{-1} C_{t-1} + \frac{\sigma^2}{\tau^2}.$$

Thus

$$
\begin{aligned}
A_t^{-1}C_t = \begin{bmatrix} -b_t \\ \sigma^2 - K_t \end{bmatrix} &= \sigma^2 C_t - \frac{1}{1 + \frac{\phi^2}{\tau^2} C_{t-1}^\top A_{t-1}^{-1} C_{t-1} + \frac{\sigma^2}{\tau^2}} \begin{bmatrix} -\frac{\phi\sigma^2}{\tau^2} A_{t-1}^{-1} C_{t-1} \\ \frac{\sigma^4}{\tau^2} \end{bmatrix} \\
&= \sigma^2 C_t - \frac{1}{\tau^2 + \phi^2 C_{t-1}^\top A_{t-1}^{-1} C_{t-1} + \sigma^2} \begin{bmatrix} -\phi\sigma^2 A_{t-1}^{-1} C_{t-1} \\ \sigma^4 \end{bmatrix}
\end{aligned} \tag{B.8}
$$

and

$$
\sigma^2 - K_t = \sigma^2 - \frac{\sigma^4}{\tau^2 + \phi^2 C_{t-1}^\top A_{t-1}^{-1} C_{t-1} + \sigma^2} = \sigma^2 - \frac{\sigma^4}{\tau^2 + \sigma^2 + \phi^2(\sigma^2 - K_{t-1})},
$$

therefore

$$
K_t = \frac{\sigma^4}{\tau^2 + \sigma^2 + \phi^2(\sigma^2 - K_{t-1})}, \tag{B.9}
$$

and

$$
b_t = \begin{bmatrix} \frac{b_{t-1}\phi K_t}{\sigma^2} \\ \frac{K_t(\sigma^2+\tau^2)-\sigma^4}{\phi\sigma^2} \end{bmatrix},
$$

$$
\begin{aligned}
\bar{\mu}_t &= 0 - \sigma^4 K_t^{-1} b_t^\top (Z - b_t K_t^{-1} b_t^\top)^{-1} \sigma^{-4} (Z - b_t K_t^{-1} b_t^\top) y_{1:t-1} \\
&= -K_t^{-1} b_t^\top y_{1:t-1} \\
&= \frac{\phi}{\sigma^2} K_{t-1} \bar{\mu}_{t-1} + \phi(1 - \frac{K_{t-1}}{\sigma^2}) y_{t-1}, \\
\bar{\Sigma}_t &= \sigma^4 (K_t - b_t^\top Z^{-1} b_t)^{-1} - \sigma^4 K_t^{-1} b_t^\top (Z_t - b_t K_t^{-1} b_t^\top)^{-1} (Z_t - b_t K_t^{-1} b_t^\top) Z_t^{-1} b_t (K_t - b_t^\top Z_t^{-1} b_t)^{-1} \\
&= \sigma^4 (I - K_t^{-1} b_t^\top Z_t^{-1} b_t)(K_t - b_t^\top Z_t^{-1} b_t)^{-1} \\
&= \sigma^4 K_t^{-1},
\end{aligned}
$$

where $K_1 = \frac{\sigma^4}{\sigma^2+\tau^2+L^2\phi^2}$.

**Estimation**

As introduced before, $p(x_t \mid y_{1:t})$ is a mixture Gaussian distribution with given $\theta$ and its mean and variance can be found by

$$
\mu_x^{(t)} = \frac{1}{N} \sum_i \mu_i \tag{B.10}
$$

$$
\mathrm{Var}(x)^{(t)} = \mathrm{E}(\mathrm{Var}(x \mid y, \theta)) + \mathrm{Var}(\mathrm{E}(x \mid y, \theta)) = \frac{1}{N} \sum_i (\mu_i \mu_i^\top + \Sigma_i) - \frac{1}{N^2} (\sum_i \mu_i)(\sum_i \mu_i)^\top. \tag{B.11}
$$

To find $\mu_i$ and $\Sigma_i$, we will use the joint distribution of $x_t$ and $y_{1:t}$, which is $p(x_t, y_{1:t} \mid \theta) \sim N(0, \Gamma)$ and

$$\Gamma = \begin{bmatrix} C_t^\top (A-B)^{-1} C_t & C_t^\top (A-B)^{-1} \\ (A-B)^{-1} C_t & (I - A^{-1}B)^{-1} B^{-1} \end{bmatrix}.$$

Because of

$$C_t^\top A_t^{-1} = \begin{bmatrix} -b_t^\top & \sigma^2 - K_t \end{bmatrix},$$

thus, for any given $\theta_i$, $x_t \mid y_{1:t}, \theta_i \sim N(\mu_t^{(x)}, \sigma_t^{(x)2})$, where

$$\begin{aligned}
\mu_i &= \phi \hat{x}_{t-1} + C_t^\top (A-B)^{-1} B (I - A^{-1}B) y_{1:t} \\
&= \phi \hat{x}_{t-1} + C_t^\top A^{-1} B y_{1:t} \\
&= \phi \hat{x}_{t-1} + \frac{1}{\sigma^2} C_t^\top A^{-1} y_{1:t} \\
&= 0 + \frac{1}{\sigma^2} \begin{bmatrix} -b_t^\top & \sigma^2 - K_t \end{bmatrix} \begin{bmatrix} y_{1:t-1} \\ y_t \end{bmatrix} \\
&= -\frac{1}{\sigma^2} b_{t-1}^\top y_{1:t-1} + (1 - \frac{K_t}{\sigma^2}) y_t \\
&= \frac{K_t \bar{\mu}_t}{\sigma^2} + (1 - \frac{K_t}{\sigma^2}) y_t \\
\Sigma_i &= C_t^\top (A-B)^{-1} C_t - C_t^\top (A-B)^{-1} B (I - A^{-1}B)(A-B)^{-1} C_t \\
&= C_t^\top (A-B)^{-1} C_t - C_t^\top A^{-1} B (A-B)^{-1} C_t \\
&= C_t^\top A^{-1} C_t \\
&= \sigma^2 - K_t.
\end{aligned}$$

By substituting them into the equation (5.37) and (5.38), the estimated $\hat{x}_t$ is easily got.

### B.1.2 OU process calculation

**Forecast**

We are now using the capital letter $Y$ to represent the joint $\{y, v\}$ and $Y_{1:t} = \{y_1, v_1, y_2, v_2, \cdots, y_t, v_t\}$, $Y_{t+1} = \{y_{t+1}, v_{t+1}\}$. It is known that

$$\begin{aligned}
p(Y_{1:t}, \theta) &\sim N\left(0, \Sigma_{YY}^{(t)}\right) \\
p(Y_{t+1}, Y_{1:t}, \theta) &\sim N\left(0, \Sigma_{YY}^{(t+1)}\right) \\
p(Y_{t+1} \mid Y_{1:t}, \theta) &\sim N\left(\bar{\mu}_{t+1}, \bar{\Sigma}_{t+1}\right)
\end{aligned}$$

where the covariance matrix of the joint distribution is $\Sigma_{YY}^{(t+1)} = (I_{t+1} - A_{t+1}^{-1}B_{t+1})^{-1}B_{t+1}^{-1}$. Then, by taking its inverse, we will get

$$\Sigma_{YY}^{(t+1)(-1)} = B_{t+1}(I_{t+1} - A_{t+1}^{-1}B_{t+1}).$$

To be clear, the matrix $B_t$ is short for the matrix $B_t(\sigma^2, \tau^2)$, which is $2t \times 2t$ diagonal matrix with elements $\frac{1}{\sigma^2}, \frac{1}{\tau^2}$ repeating for $t$ times on its diagonal. For instance, the very simple $B_1(\sigma^2, \tau^2) = \begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{\tau^2} \end{bmatrix}_{2\times 2}$ is a $2 \times 2$ matrix.

Because of $A$ is symmetric and invertible, $B$ is the diagonal matrix defined as above, then they have the following property

$$\begin{aligned} AB &= A^\top B^\top = (BA)^\top, \\ A^{-1}B &= A^{-\top}B^\top = (BA^{-1})^\top. \end{aligned}$$

Followed up the form of $\Sigma_{YY}^{(t+1)(-1)}$, we can find out that

$$\begin{aligned} \Sigma_{YY}^{(t+1)(-1)} &= B_{t+1}(I_{t+1} - A_{t+1}^{-1}B_{t+1}) \\ &= B_{t+1}(B_{t+1}^{-1} - A_{t+1}^{-1})B_{t+1} \\ &\triangleq \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} \begin{bmatrix} Z_{t+1} & b_{t+1} \\ b_{t+1}^\top & K_{t+1} \end{bmatrix} \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} \end{aligned}$$

where $Z_{t+1}$ is a $2t \times 2t$ matrix, $b_{t+1}$ is a $2t \times 2$ matrix and $K_{t+1}$ is a $2 \times 2$ matrix. Thus by taking its inverse again, we will get

$$\Sigma_{YY}^{(t+1)} = \begin{bmatrix} B_t^{-1}(Z_{t+1} - b_{t+1}K_{t+1}^{-1}b_{t+1}^\top)^{-1}B_t^{-1} & -B_t^{-1}Z_{t+1}^{-1}b_{t+1}(K_{t+1} - b_{t+1}^\top Z_{t+1}^{-1}b_{t+1})^{-1}B_1^{-1} \\ -B_1^{-1}K_{t+1}^{-1}b_{t+1}^\top(Z_{t+1} - b_{t+1}K_{t+1}^{-1}b_{t+1}^\top)^{-1}B_t^{-1} & B_1^{-1}(K_{t+1} - b_{t+1}^\top Z_{t+1}^{-1}b_{t+1})^{-1}B_1^{-1} \end{bmatrix}.$$

It is easy to find the relationship between $A_{t+1}$ and $A_t$ in the Sherman-Morrison-Woodbury form, which is

$$A_{t+1} = \begin{bmatrix} A_t & \cdot & \cdot \\ \cdot & \frac{1}{\sigma^2} & \cdot \\ \cdot & \cdot & \frac{1}{\tau^2} \end{bmatrix} + U_{t+1}U_{t+1}^\top \triangleq M_{t+1} + U_{t+1}U_{t+1}^\top,$$

where $M_{t+1} = \begin{bmatrix} A_t & \cdot & \cdot \\ \cdot & \frac{1}{\sigma^2} & \cdot \\ \cdot & \cdot & \frac{1}{\tau^2} \end{bmatrix} = \begin{bmatrix} A_t & 0 \\ 0 & B_1 \end{bmatrix}$ and its inverse is $M_{t+1}^{-1} = \begin{bmatrix} A_t^{-1} & 0 \\ 0 & B_1^{-1} \end{bmatrix}$.

Additionallly, $U$ is a $2t+2 \times 2$ matrix in the following form

$$U_{t+1} = \frac{1}{\sqrt{1-\rho_{t+1}^2}} \begin{bmatrix} \mathbf{0}_{2t-2} & \mathbf{0}_{2t-2} \\ \frac{1}{\sigma_{t+1}^{(x)}} & 0 \\ \frac{1-e^{-\gamma\Delta_{t+1}}}{\gamma\sigma_{t+1}^{(x)}} - \frac{\rho_{t+1}e^{-\gamma\Delta_{t+1}}}{\sigma_{t+1}^{(u)}} & \frac{\sqrt{1-\rho_{t+1}^2}e^{-\gamma\Delta_{t+1}}}{\sigma_{t+1}^{(u)}} \\ -\frac{1}{\sigma_{t+1}^{(x)}} & 0 \\ \frac{\rho_{t+1}}{\sigma_{t+1}^{(u)}} & -\frac{\sqrt{1-\rho_{t+1}^2}}{\sigma_{t+1}^{(u)}} \end{bmatrix} \triangleq \begin{bmatrix} C_t S_{t+1} \\ D_{t+1} \end{bmatrix},$$

where $S_{t+1} = \frac{1}{\sqrt{1-\rho_{t+1}^2}} \begin{bmatrix} \frac{1}{\sigma_{t+1}^{(x)}} & 0 \\ \frac{1-e^{-\gamma\Delta_{t+1}}}{\gamma\sigma_{t+1}^{(x)}} - \frac{\rho_{t+1}e^{-\gamma\Delta_{t+1}}}{\sigma_{t+1}^{(u)}} & \frac{\sqrt{1-\rho_{t+1}^2}e^{-\gamma\Delta_{t+1}}}{\sigma_{t+1}^{(u)}} \end{bmatrix}$,

$D_{t+1} = \frac{1}{\sqrt{1-\rho_{t+1}^2}} \begin{bmatrix} -\frac{1}{\sigma_{t+1}^{(x)}} & 0 \\ \frac{\rho_{t+1}}{\sigma_{t+1}^{(u)}} & -\frac{\sqrt{1-\rho_{t+1}^2}}{\sigma_{t+1}^{(u)}} \end{bmatrix}$ and $C_{t+1} = \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_t \\ I_2 \end{bmatrix}$.

By post-multiplying $\Sigma_{YY}^{(t+1)(-1)}$ with $C_{t+1}$, it gives us

$$\begin{aligned} \Sigma_{YY}^{(t+1)(-1)} C_{t+1} &= B_{t+1}(I_{t+1} - A_{t+1}^{-1}B_{t+1})C_{t+1} \\ &= \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} \left( \begin{bmatrix} B_t^{-1} & 0 \\ 0 & B_1^{-1} \end{bmatrix} - A_{t+1}^{-1} \right) \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} C_{t+1} \\ &= \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} \begin{bmatrix} Z_{t+1} & b_{t+1} \\ b_{t+1}^\top & K_{t+1} \end{bmatrix} \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} C_{t+1} \\ &= \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} \begin{bmatrix} b_{t+1}B_1 \\ K_{t+1}B_1 \end{bmatrix}. \end{aligned}$$

and the property of $A_{t+1}^{-1}$ is

$$A_{t+1}^{-1} C_{t+1} = \begin{bmatrix} -b_{t+1} \\ B_1^{-1} - K_{t+1} \end{bmatrix}.$$

Moreover, by pre-multiplying $C_{t+1}^\top$ on the left side of the above equation, we will have

$$C_{t+1}^\top A_{t+1}^{-1} C_{t+1} = B_1^{-1} - K_{t+1}, \tag{B.12}$$

$$K_{t+1} = B_1^{-1} - C_{t+1}^\top A_{t+1}^{-1} C_{t+1}. \tag{B.13}$$

We may use Sherman-Morrison-Woodbury formula to find the inverse of $A_{t+1}$ in a recursive way, which is

$$A_{t+1}^{-1} = (M_{t+1} + U_{t+1}U_{t+1}^\top)^{-1} = M_{t+1}^{-1} - M_{t+1}^{-1}U_{t+1}(I + U_{t+1}^\top M_{t+1}^{-1} U_{t+1})^{-1} U_{t+1}^\top M_{t+1}^{-1}. \tag{B.14}$$

Consequently, it is easy to find that $M_{t+1}^{-1}C_{t+1} = \begin{bmatrix} 0 \\ B_1^{-1} \end{bmatrix}$ and

$$\begin{aligned}
A_{t+1}^{-1}C_{t+1} &= \begin{bmatrix} 0 \\ B_1^{-1} \end{bmatrix} - \begin{bmatrix} A_t^{-1} & 0 \\ 0 & B_1^{-1} \end{bmatrix} \begin{bmatrix} C_t S_{t+1} \\ D \end{bmatrix} (I + U_{t+1}^\top M_{t+1}^{-1} U_{t+1})^{-1} \begin{bmatrix} S_{t+1}^\top C_t^\top & D_{t+1}^\top \end{bmatrix} \begin{bmatrix} 0 \\ B_1^{-1} \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ B_1^{-1} \end{bmatrix} - \begin{bmatrix} A_t^{-1} C_t S_{t+1} \\ B_1^{-1} D_{t+1} \end{bmatrix} (I + U_{t+1}^\top M_{t+1}^{-1} U_{t+1})^{-1} D_{t+1}^\top B_1^{-1} \\
&= \begin{bmatrix} 0 \\ B_1^{-1} \end{bmatrix} - \begin{bmatrix} A_t^{-1} C_t S_{t+1} \\ B_1^{-1} D_{t+1} \end{bmatrix} (I + S_{t+1}^\top C_t^\top A_t^{-1} C_t S_{t+1} + D_{t+1}^\top B_1^{-1} D_{t+1})^{-1} D_{t+1}^\top B_1^{-1} \\
&= \begin{bmatrix} 0 \\ B_1^{-1} \end{bmatrix} - \begin{bmatrix} A_t^{-1} C_t S_{t+1} \\ B_1^{-1} D_{t+1} \end{bmatrix} (I + S_{t+1}^\top (B_1^{-1} - K_t) S_{t+1} + D_{t+1}^\top B_1^{-1} D_{t+1})^{-1} D_{t+1}^\top B_1^{-1}.
\end{aligned}$$

Thus, by using the equation (B.13), we will get

$$K_{t+1} = B_1^{-1} D_{t+1} (I + S_{t+1}^\top (B_1^{-1} - K_t) S_{t+1} + D_{t+1}^\top B_1^{-1} D_{t+1})^{-1} D_{t+1}^\top B_1^{-1}, \qquad \text{(B.15)}$$

and

$$\begin{aligned}
b_{t+1} &= A_t^{-1} C_t S_{t+1} (I + S_{t+1}^\top (B_1^{-1} - K_t) S_{t+1} + D_{t+1}^\top B_1^{-1} D_{t+1})^{-1} D_{t+1}^\top B_1^{-1} \\
&= \begin{bmatrix} -b_t \\ B_1^{-1} - K_t \end{bmatrix} S_{t+1} (I + S_{t+1}^\top (B_1^{-1} - K_t) S_{t+1} + D_{t+1}^\top B_1^{-1} D_{t+1})^{-1} D_{t+1}^\top B_1^{-1}.
\end{aligned}$$

To achieve the recursive updating formula, firstly we need to find the form of $b_{t+1}^\top B_t^2 Y_{1:t}$.
In fact, it is

$$\begin{aligned}
b_{t+1}^\top B_t Y_{1:t} &= B_1^{-\top} D_{t+1} (I + S_{t+1}^\top (B_1^{-1} - K_t) S_{t+1} + D_{t+1}^\top B_1^{-1} D_{t+1})^{-\top} S_{t+1}^\top \begin{bmatrix} -b_t^\top & B_1^{-1} - K_t \end{bmatrix} B_t \begin{bmatrix} Y_{1:t-1} \\ Y_t \end{bmatrix} \\
&= B_1^{-\top} D_{t+1} (I + S_{t+1}^\top (B_1^{-1} - K_t) S_{t+1} + D_{t+1}^\top B_1^{-1} D_{t+1})^{-\top} S_{t+1}^\top \\
&\quad \left( -b_t^\top B_{t-1} Y_{1:t-1} + (B_1^{-1} - K_t) B_1 Y_t \right) \\
&= B_1^{-\top} D_{t+1} (I + S_{t+1}^\top (B_1^{-1} - K_t) S_{t+1} + D_{t+1}^\top B_1^{-1} D_{t+1})^{-\top} S_{t+1}^\top \left( K_t B_1 \bar{\mu}_t + (I - K_t B_1) Y_t \right),
\end{aligned}$$

By using equation (B.15) and simplifying the above equation, one can achieve a recursive updating form of the mean, which is

$$\begin{aligned}
\bar{\mu}_{t+1} &= -B_1 K_{t+1}^{-1} b_{t+1}^\top B_t Y_{1:t} \\
&= -D_{t+1}^{-\top} S_{t+1}^\top (K_t B_1 \bar{\mu}_t + (I - K_t B_1) Y_t) \\
&= -D_{t+1}^{-\top} S_{t+1}^\top (Y_t + K_t B_1 (\bar{\mu}_t - Y_t)),
\end{aligned}$$

where by simplifying $D^{-\top}S^{\top}$, one may find

$$D_{t+1}^{-\top}S_{t+1}^{\top} = \begin{bmatrix} -1 & -\frac{1-e^{-\gamma\Delta_{t+1}}}{\gamma} \\ 0 & -e^{-\gamma\Delta_{t+1}} \end{bmatrix} = -\Phi_{t+1},$$

which is the negative of forward process. Then the final form of recursive updating formula are

$$\begin{cases} \bar{\mu}_{t+1} &= \Phi_{t+1}K_tB_1\bar{\mu}_t + \Phi_{t+1}(I - K_tB_1)Y_t \\ \bar{\Sigma}_{t+1} &= (B_1K_{t+1}B_1)^{-1} \end{cases}. \tag{B.16}$$

The matrix $K_{t+1}$ is updated via

$$K_{t+1} = B_1^{-1}D_{t+1}(I + S_{t+1}^{\top}(B_1^{-1} - K_t)S_{t+1} + D_{t+1}^{\top}B_1^{-1}D_{t+1})^{-1}D_{t+1}^{\top}B_1^{-1}, \tag{B.17}$$

or updating its inverse in the following form makes the computation faster, that is

$$\begin{cases} K_{t+1}^{-1} &= B_1D_{t+1}^{-\top}D_{t+1}^{-1}B_1 + B_1\Phi_{t+1}(B_1^{-1} - K_t)\Phi_{t+1}^{\top}B_1 + B_1, \\ \bar{\Sigma}_{t+1} &= D_{t+1}^{-\top}D_{t+1}^{-1} + \Phi_{t+1}(B_1^{-1} - K_t)\Phi_{t+1}^{\top} + B_1^{-1} \end{cases}$$

and $K_1 = B_1^{-1} - A_1^{-1} = \begin{bmatrix} \frac{\sigma^4}{\sigma^2+L_x^2} & 0 \\ 0 & \frac{\tau^4}{\tau^2+L_u^2} \end{bmatrix}$.

**Estimation**

Because of the joint distribution (5.58), one can find the best estimation with a given $\theta$ by

$$\begin{aligned} X \mid Y, \theta &\sim N\left(A^{-1}BY, A^{-1}\right) \\ &\sim N(L^{-\top}L^{-1}BY, L^{-\top}L^{-1}) \\ &\sim N(L^{-\top}W, L^{-\top}L^{-1}), \end{aligned}$$

thus

$$\hat{X} = L^{-\top}(W + Z),$$

where $Z \sim N(0, I(\sigma, \tau))$.

For $X_{t+1}$, the joint distribution with $Y$ updated to stage $t+1$ is

$$X_{t+1}, Y \mid \theta \sim N\left(0, \begin{bmatrix} C_{t+1}^{\top}(A-B)^{-1}C_{t+1} & C_{t+1}^{\top}(A-B)^{-1} \\ (A-B)^{-1}C_{t+1} & (I - A^{-1}B)^{-1}B^{-1} \end{bmatrix}\right),$$

where $C_{t+1}^\top = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$ is a $2 \times 2(t+1)$ matriX. Thus

$$X_{t+1} \mid Y, \theta \sim N(\bar{\mu}_{t+1}^{(X)}, \bar{\Sigma}_{t+1}^{(X)}),$$

where

$$\begin{aligned} \bar{\mu}_{t+1}^{(X)} &= C_{t+1}^\top A^{-1} B Y = C_{t+1}^\top L^{-\top} W, \\ \bar{\Sigma}_{t+1}^{(X)} &= C_{t+1}^\top A^{-1} C_{t+1} = U_{t+1}^\top U_{t+1}, \end{aligned}$$

and $U_{t+1} = L^{-1} C_{t+1} = \text{solve}(L, C_{t+1})$.

The filtering distribution of the state given parameters is $p(X_t \mid Y_{1:t}, \theta)$. To find its form, one can use the joint distribution of $X_{t+1}$ and $Y_{1:t+1}$, which is $p(X_{t+1}, Y_{1:t+1} \mid \theta) \sim N(0, \Gamma)$, where

$$\Gamma = \begin{bmatrix} C_{t+1}^\top (A-B)^{-1} C_{t+1} & C_{t+1}^\top (A-B)^{-1} \\ (A-B)^{-1} C_{t+1} & (I - A^{-1}B)^{-1} B^{-1} \end{bmatrix}.$$

Because of

$$C_{t+1}^\top A_{t+1}^{-1} = \begin{bmatrix} -b_{t+1}^\top & B_1^{-1} - K_{t+1} \end{bmatrix},$$

then $X_{t+1} \mid Y_{1:t+1}, \theta \sim N(\bar{\mu}_{t+1}^{(X)}, \bar{\sigma}_{t+1}^{(X)2})$, where

$$\begin{aligned} \bar{\mu}_{t+1}^{(X)} &= \Phi \hat{x}_t + C_{t+1}^\top (A-B)^{-1} B (I - A^{-1}B) Y_{1:t+1} \\ &= \Phi \hat{x}_t + C_{t+1}^\top A^{-1} B Y_{1:t+1} \\ &= 0 + \begin{bmatrix} -b_{t+1}^\top & B_1^{-1} - K_{t+1} \end{bmatrix} \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} \begin{bmatrix} Y_{1:t} \\ Y_{y+1} \end{bmatrix} \\ &= -b^\top B_t Y_{1:t} + (I - B_1 K_{t+1}) Y_{t+1} \\ &= K_{t+1} B_1 \bar{\mu}_{t+1} + (I - B_1 K_{t+1}) Y_{t+1} \\ \bar{\sigma}_{t+1}^{(X)2} &= C_{t+1}^\top (A-B)^{-1} C_{t+1} - C_{t+1}^\top (A-B)^{-1} B (I - A^{-1}B)(A-B)^{-1} C_{t+1} \\ &= C_{t+1}^\top (A-B)^{-1} C_{t+1} - C_{t+1}^\top A^{-1} B (A-B)^{-1} C_{t+1} \\ &= C_{t+1}^\top A^{-1} C_{t+1} \\ &= B_1^{-1} - K_{t+1}. \end{aligned}$$

**Covariance Matrix in Details**

$$
\begin{aligned}
\Sigma_t &= \begin{bmatrix} \sigma_t^{(x)2} & \rho_t \sigma_t^{(x)} \sigma_t^{(u)} \\ \rho_t \sigma_t^{(x)} \sigma_t^{(u)} & \sigma_t^{(u)2} \end{bmatrix} \\
\Sigma_t^{-1} &= \frac{1}{1-\rho_t^2} \begin{bmatrix} \frac{1}{\sigma_t^{(x)2}} & -\frac{\rho_t}{\sigma_t^{(x)} \sigma_t^{(u)}} \\ -\frac{\rho_t}{\sigma_t^{(x)} \sigma_t^{(u)}} & \frac{1}{\sigma_t^{(u)2}} \end{bmatrix} \\
M_t^\top &= \begin{bmatrix} 1 & 0 \\ \frac{1-e^{-\gamma \Delta_t}}{\gamma} & e^{-\gamma \Delta_t} \end{bmatrix} \\
z_t &= \begin{bmatrix} x_t \\ u_t \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
M_t^\top \Sigma_t^{-1} &= \frac{1}{1-\rho_t^2} \begin{bmatrix} \frac{1}{\sigma_t^{(x)2}} & -\frac{\rho_t}{\sigma_t^{(x)} \sigma_t^{(u)}} \\ \frac{1-e^{-\gamma \Delta_t}}{\gamma \sigma_t^{(x)2}} - \frac{\rho_t e^{-\gamma \Delta_t}}{\sigma_t^{(x)} \sigma_t^{(u)}} & -\frac{\rho_t (1-e^{-\gamma \Delta_t})}{\gamma \sigma_t^{(x)} \sigma_t^{(u)}} + \frac{e^{-\gamma \Delta_t}}{\sigma_t^{(u)2}} \end{bmatrix} \\
M_t^\top \Sigma_t^{-1} M_t &= \frac{1}{1-\rho_t^2} \begin{bmatrix} \frac{1}{\sigma_t^{(x)2}} & \frac{1-e^{-\gamma \Delta_t}}{\gamma \sigma_t^{(x)2}} - \frac{\rho_t e^{-\gamma \Delta_t}}{\sigma_t^{(x)} \sigma_t^{(u)}} \\ \frac{1-e^{-\gamma \Delta_t}}{\gamma \sigma_t^{(x)2}} - \frac{\rho_t e^{-\gamma \Delta_t}}{\sigma_t^{(x)} \sigma_t^{(u)}} & \frac{(1-e^{-\gamma \Delta_t})^2}{\gamma^2 \sigma_t^{(x)2}} - \frac{2\rho_t e^{-\gamma \Delta_t} (1-e^{-\gamma \Delta_t})}{\gamma \sigma_t^{(x)} \sigma_t^{(u)}} + \frac{e^{-2\gamma \Delta_t}}{\sigma_t^{(u)2}} \end{bmatrix}
\end{aligned}
$$

$$\Sigma_4^{(X)-1} = \frac{1}{1-\rho_t^2}\begin{bmatrix}
\frac{1-\rho_t^2}{\sigma_1^{(x)2}} + \frac{1}{\sigma_2^{(x)2}} & -\frac{1}{\sigma_2^{(x)2}} & 0 & 0 & \frac{1-e^{-\gamma\Delta_2}}{\gamma\sigma_2^{(x)2}} - \frac{\rho_t e^{-\gamma\Delta_2}}{\sigma_2^{(x)}\sigma_2^{(u)}} & \frac{\rho_t}{\sigma_2^{(x)}\sigma_2^{(u)}} & 0 & 0 \\
-\frac{1}{\sigma_2^{(x)2}} & \frac{1}{\sigma_2^{(x)2}} + \frac{1}{\sigma_3^{(x)2}} & -\frac{1}{\sigma_3^{(x)2}} & 0 & -\frac{1-e^{-\gamma\Delta_2}}{\gamma\sigma_2^{(x)2}} + \frac{\rho_t e^{-\gamma\Delta_2}}{\sigma_2^{(x)}\sigma_2^{(u)}} & \frac{1-e^{-\gamma\Delta_3}}{\gamma\sigma_3^{(x)2}} - \frac{\rho_t e^{-\gamma\Delta_3}}{\sigma_3^{(x)}\sigma_3^{(u)}} - \frac{\rho_t}{\sigma_2^{(x)}\sigma_2^{(u)}} & \frac{\rho_t}{\sigma_3^{(x)}\sigma_3^{(u)}} & 0 \\
0 & -\frac{1}{\sigma_3^{(x)2}} & \frac{1}{\sigma_3^{(x)2}} + \frac{1}{\sigma_4^{(x)2}} & -\frac{1}{\sigma_4^{(x)2}} & 0 & -\frac{1-e^{-\gamma\Delta_3}}{\gamma\sigma_3^{(x)2}} + \frac{\rho_t e^{-\gamma\Delta_3}}{\sigma_3^{(x)}\sigma_3^{(u)}} & \frac{1-e^{-\gamma\Delta_4}}{\gamma\sigma_4^{(x)2}} - \frac{\rho_t e^{-\gamma\Delta_4}}{\sigma_4^{(x)}\sigma_4^{(u)}} - \frac{\rho_t}{\sigma_3^{(x)}\sigma_3^{(u)}} & \frac{\rho_t}{\sigma_4^{(x)}\sigma_4^{(u)}} \\
0 & 0 & -\frac{1}{\sigma_4^{(x)2}} & \frac{1}{\sigma_4^{(x)2}} & 0 & 0 & -\frac{1-e^{-\gamma\Delta_4}}{\gamma\sigma_4^{(x)2}} + \frac{\rho_t e^{-\gamma\Delta_4}}{\sigma_4^{(x)}\sigma_4^{(u)}} & -\frac{\rho_t}{\sigma_4^{(x)}\sigma_4^{(u)}} \\
\top & \top & 0 & 0 & \frac{1-\rho_t^2}{\sigma_1^{(u)2}} + C_2 & -\frac{e^{-\gamma\Delta_2}}{\sigma_2^{(u)2}} + \frac{\rho_t(1-e^{-\gamma\Delta_2})}{\gamma\sigma_2^{(x)}\sigma_2^{(u)}} & 0 & 0 \\
\top & \top & \top & 0 & -\frac{e^{-\gamma\Delta_2}}{\sigma_2^{(u)2}} + \frac{\rho_t(1-e^{-\gamma\Delta_2})}{\gamma\sigma_2^{(x)}\sigma_2^{(u)}} & \frac{1}{\sigma_2^{(u)2}} + C_3 & -\frac{e^{-\gamma\Delta_3}}{\sigma_3^{(u)2}} + \frac{\rho_t(1-e^{-\gamma\Delta_3})}{\gamma\sigma_3^{(x)2}\sigma_3^{(u)}} & 0 \\
0 & \top & \top & \top & 0 & -\frac{e^{-\gamma\Delta_3}}{\sigma_3^{(u)2}} + \frac{\rho_t(1-e^{-\gamma\Delta_3})}{\gamma\sigma_3^{(x)}\sigma_3^{(u)}} & \frac{1}{\sigma_3^{(u)2}} + C_4 & -\frac{e^{-\gamma\Delta_4}}{\sigma_4^{(u)2}} + \frac{\rho_t(1-e^{-\gamma\Delta_4})}{\gamma\sigma_4^{(x)}\sigma_4^{(u)}} \\
0 & 0 & \top & \top & 0 & 0 & -\frac{e^{-\gamma\Delta_4}}{\sigma_4^{(u)2}} + \frac{\rho_t(1-e^{-\gamma\Delta_4})}{\gamma\sigma_4^{(x)}\sigma_4^{(u)}} & \frac{1}{\sigma_4^{(u)2}}
\end{bmatrix}$$

where $C_t = \frac{(1-e^{-\gamma\Delta_t})^2}{\gamma^2\sigma_t^{(x)2}} + \frac{e^{-2\gamma\Delta_t}}{\sigma_t^{(u)2}} - \frac{2\rho_t e^{-\gamma\Delta_t}(1-e^{-\gamma\Delta_t})}{\gamma\sigma_t^{(x)}\sigma_t^{(u)}}$, $\Delta_t = T_t - T_{t-1}$.

### B.1.3 Real Data Implementation

#### Efficiency Plots

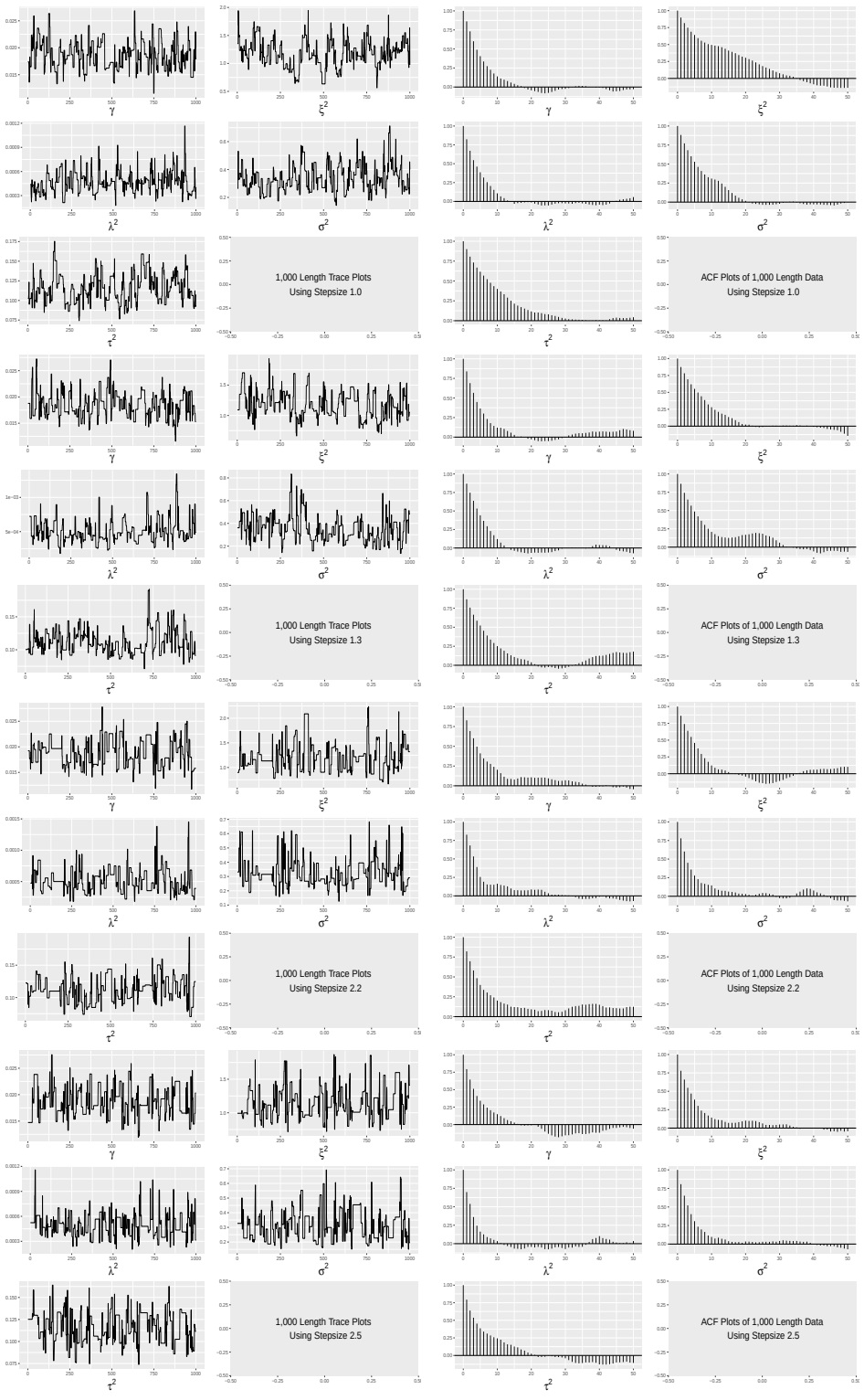



Figure B.1: Running the same amount of time and taking the same length of data, the step size $\epsilon = 2.5$ returns the highest ESSUT value and generates more effective samples with a lower correlation.

## Comparing Estimation with Different Length of Data

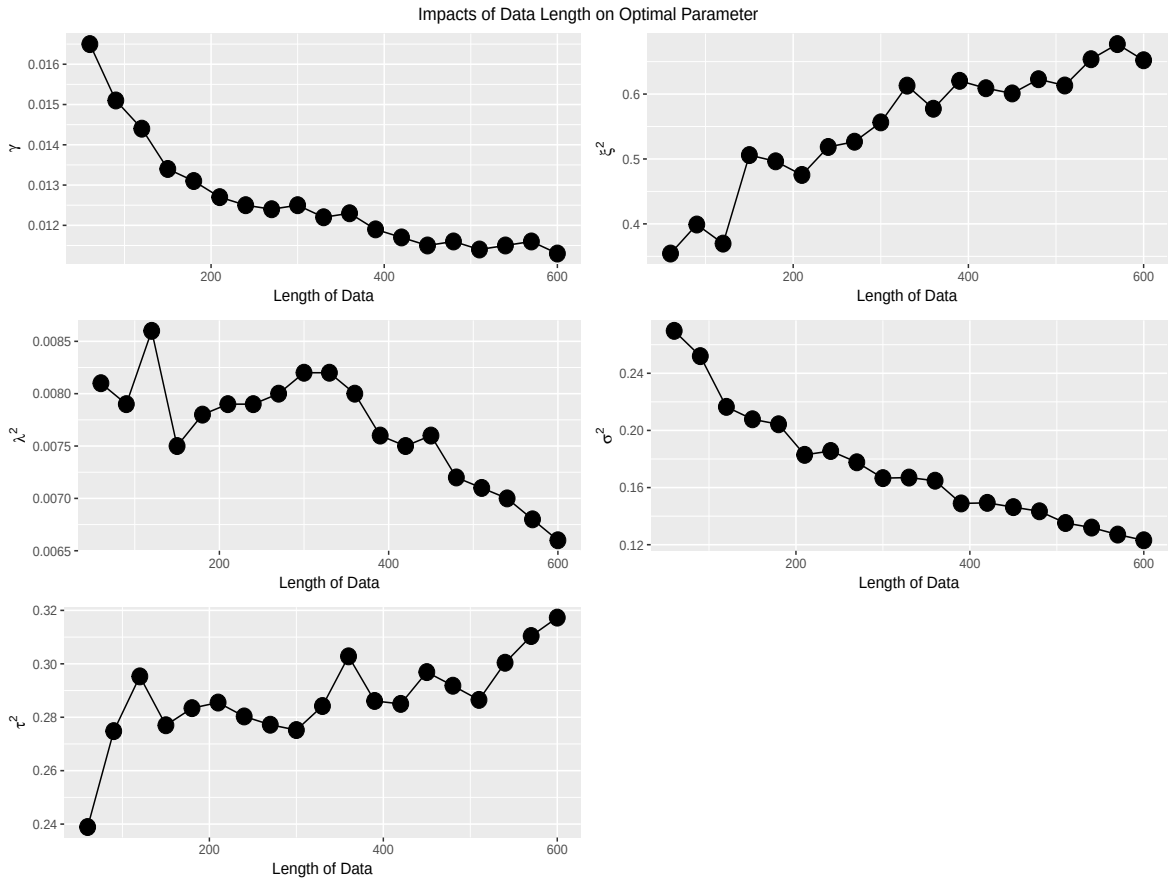



Figure B.2: Impacts of data length on optimal parameter.

Table B.1: Parameter estimation by running the whole surface learning and DA-MH processes with different length of data

| **Length** | **Time** | $\gamma$ | $\xi^2$ | $\lambda^2$ | $\sigma^2$ | $\tau^2$ | $\alpha_1$ | $\alpha_2$ | $x$ | $v_x$ | $y$ | $v_y$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Obs** | - | - | - | - | - | - | - | - | -339.0569 | 0.0413 | -100.2065 | 1.1825 |
| **600** | 85.96 | 0.0113 | 0.6521 | 0.0066 | 0.1231 | 0.3173 | 0.0536 | 0.7873 | -339.0868 | 0.4331 | -100.1498 | -0.7498 |
| **570** | 85.72 | 0.0116 | 0.6770 | 0.0068 | 0.1271 | 0.3104 | 0.0542 | 0.7638 | -339.0872 | 0.4292 | -100.1476 | -0.7356 |
| **540** | 84.25 | 0.0115 | 0.6537 | 0.0070 | 0.1320 | 0.3004 | 0.0662 | 0.7553 | -339.0889 | 0.4326 | -100.1435 | -0.7375 |
| **510** | 85.13 | 0.0114 | 0.6132 | 0.0071 | 0.1352 | 0.2865 | 0.0684 | 0.7310 | -339.0907 | 0.4376 | -100.1387 | -0.7425 |
| **480** | 81.23 | 0.0116 | 0.6229 | 0.0072 | 0.1434 | 0.2918 | 0.0534 | 0.8127 | -339.0921 | 0.4368 | -100.1359 | -0.7408 |
| **450** | 81.57 | 0.0115 | 0.6010 | 0.0076 | 0.1463 | 0.2969 | 0.0580 | 0.7931 | -339.0924 | 0.4432 | -100.1348 | -0.7521 |
| **420** | 80.31 | 0.0117 | 0.6090 | 0.0075 | 0.1493 | 0.2850 | 0.0626 | 0.7636 | -339.0938 | 0.4392 | -100.1310 | -0.7397 |
| **390** | 78.84 | 0.0119 | 0.6204 | 0.0076 | 0.1489 | 0.2861 | 0.0620 | 0.7581 | -339.0931 | 0.4373 | -100.1320 | -0.7354 |
| **360** | 76.66 | 0.0123 | 0.5774 | 0.0080 | 0.1648 | 0.3028 | 0.0554 | 0.7762 | -339.0971 | 0.4457 | -100.1248 | -0.7563 |
| **330** | 76.38 | 0.0122 | 0.6130 | 0.0082 | 0.1670 | 0.2842 | 0.0636 | 0.7830 | -339.0969 | 0.4403 | -100.1220 | -0.7336 |
| **300** | 73.27 | 0.0125 | 0.5564 | 0.0082 | 0.1666 | 0.2752 | 0.0548 | 0.8212 | -339.0989 | 0.4457 | -100.1174 | -0.7443 |
| **270** | 73.68 | 0.0124 | 0.5266 | 0.0080 | 0.1777 | 0.2772 | 0.0636 | 0.6698 | -339.1027 | 0.4489 | -100.1104 | -0.7546 |
| **240** | 71.85 | 0.0125 | 0.5185 | 0.0079 | 0.1856 | 0.2803 | 0.0548 | 0.7336 | -339.1050 | 0.4495 | -100.1067 | -0.7590 |
| **210** | 71.26 | 0.0127 | 0.4754 | 0.0079 | 0.1829 | 0.2855 | 0.0656 | 0.7561 | -339.1057 | 0.4559 | -100.1065 | -0.7754 |
| **180** | 70.25 | 0.0131 | 0.4964 | 0.0078 | 0.2043 | 0.2834 | 0.0566 | 0.7880 | -339.1107 | 0.4498 | -100.0955 | -0.7620 |
| **150** | 70.87 | 0.0134 | 0.5060 | 0.0075 | 0.2078 | 0.2770 | 0.0582 | 0.7801 | -339.1129 | 0.4436 | -100.0916 | -0.7507 |
| **120** | 68.38 | 0.0144 | 0.3696 | 0.0086 | 0.2165 | 0.2953 | 0.0570 | 0.7754 | -339.1168 | 0.4705 | -100.0825 | -0.8057 |
| **90** | 65.73 | 0.0151 | 0.3990 | 0.0079 | 0.2520 | 0.2748 | 0.0552 | 0.8188 | -339.1296 | 0.4550 | -100.0556 | -0.7740 |
| **60** | 68.81 | 0.0165 | 0.3543 | 0.0081 | 0.2697 | 0.2389 | 0.0694 | 0.7176 | -339.1412 | 0.4527 | -100.0204 | -0.7573 |

## Comparison Between Batch and Sliding Window Methods

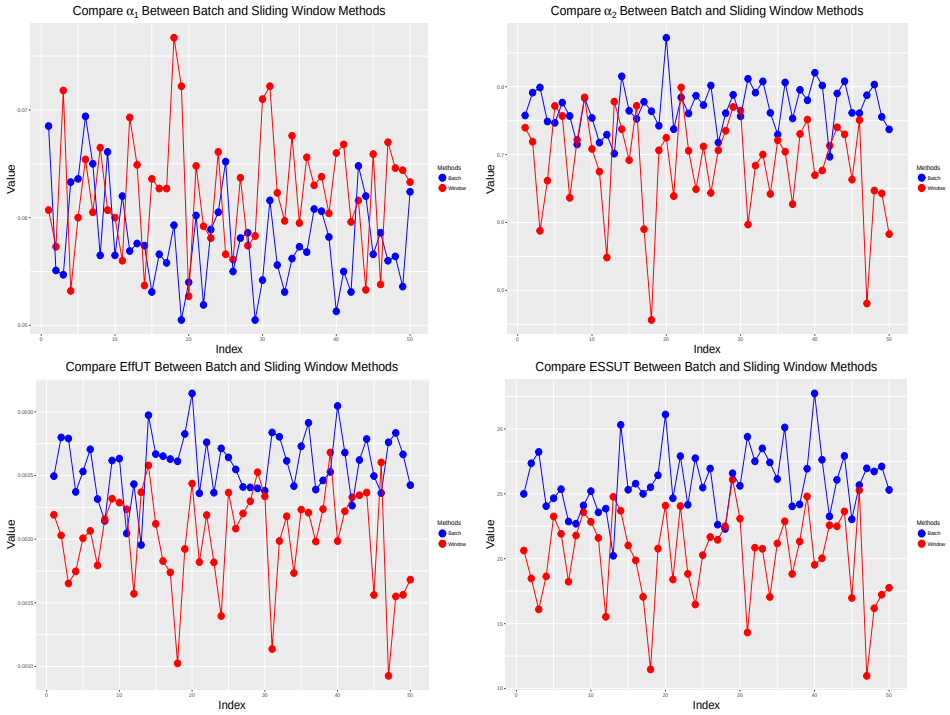



Figure B.3: Key features comparison.


Figure B.4: Parameter Comparison.

## Parameter Evolution Visualization

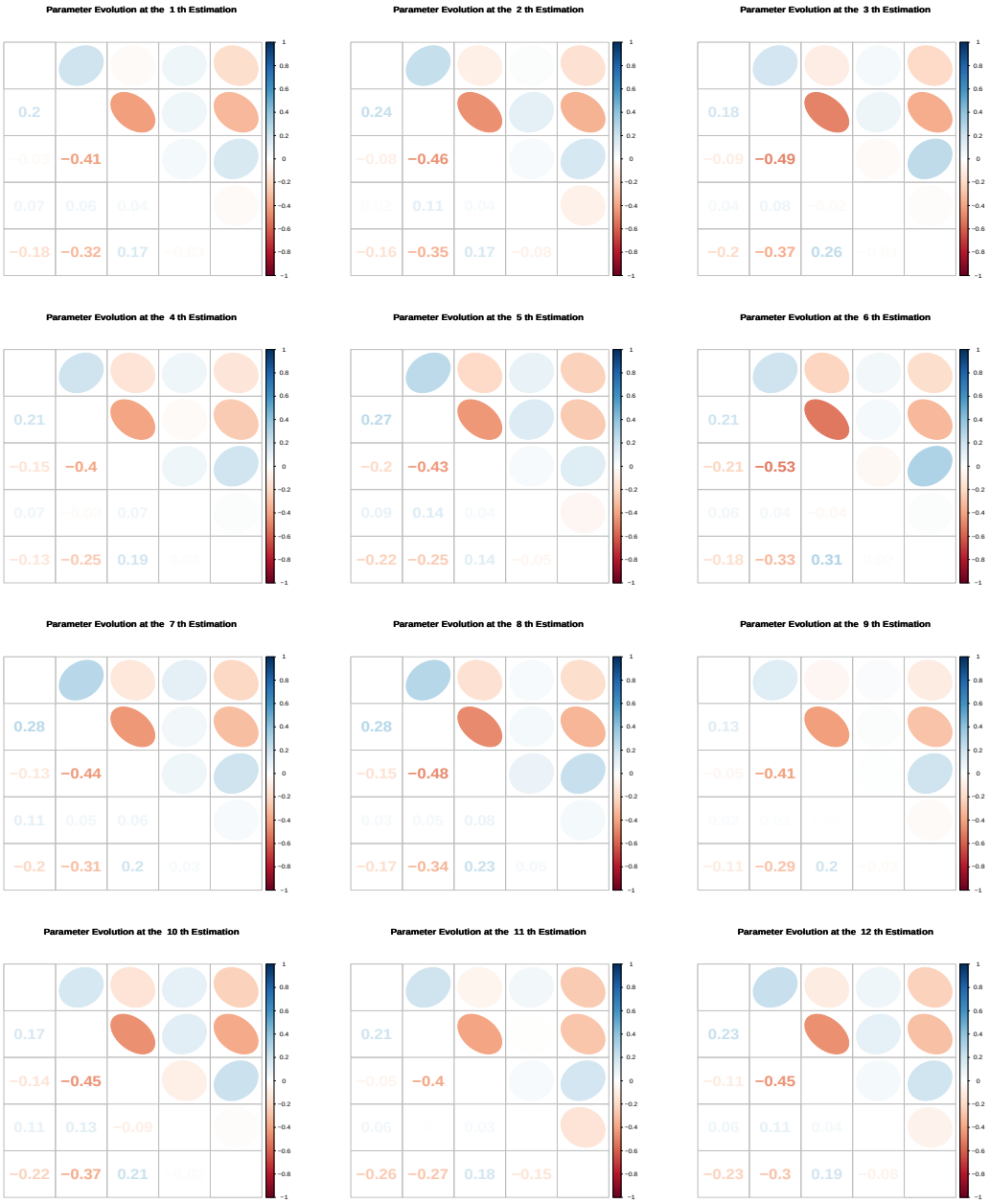



Figure B.5: Parameter Evolution Visualization.


Figure B.6: Parameter Evolution Visualization.


Figure B.7: Parameter Evolution Visualization.

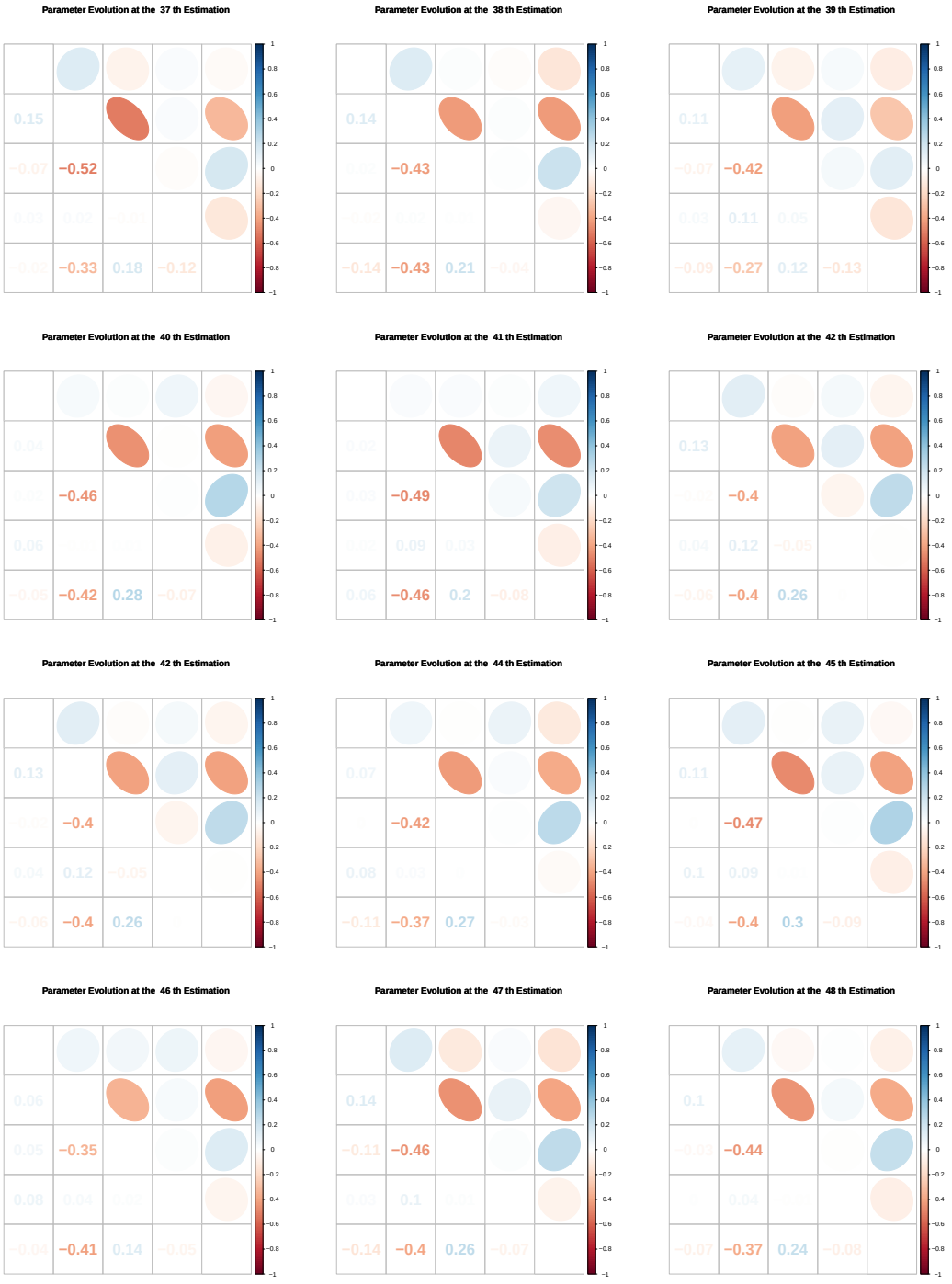



Figure B.8: Parameter Evolution Visualization.

# References


Abramovich, F., Sapatinas, T., and Silverman, B. W. (1998). Wavelet thresholding via a Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *60*(4), 725–749.

Agarwal, P. K., Arge, L., and Erickson, J. (2003). Indexing moving points. *Journal of Computer and System Sciences*, *66*(1), 207–243.

Andrieu, C., De Freitas, N., and Doucet, A. (1999). Sequential MCMC for Bayesian model selection. In *Higher-Order Statistics, 1999. Proceedings of the IEEE Signal Processing Workshop on*, 130–134. IEEE.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*(3), 269–342.

Andrieu, C., Doucet, A., and Tadic, V. B. (2005). On-line parameter estimation in general state-space models. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, 332–337. IEEE.

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and computing*, *18*(4), 343–373.

Arnaud Doucet, Nando de Freitas, N. G. (Ed.) (2011). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag New York.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing*, *50*(2), 174–188.

Atkinson, A.D., S. J. (2004). Propuesta para el levantamiento de vas de comunicacin para navegacin para mediante GPS. *Datum*, *XXI 9*, 29–32.

Aydin, D. and Tuzemen, M. S. (2012). Smoothing parameter selection problem in nonparametric regression based on smoothing spline: A simulation study. *Journal of Applied Sciences*, *12*(7), 636.

Bartlett, M. S. (1951). An inverse matrix adjustment arising in discriminant analysis. *The Annals of Mathematical Statistics*, *22*(1), 107–111.

Bédard, M. (2007). Weak convergence of Metropolis algorithms for non-iid target distributions. *The Annals of Applied Probability*, 1222–1244.

Ben-Arieh, D., Chang, S., Rys, M., and Zhang, G. (2004). Geometric modeling of highways using global positioning system data and B-spline approximation. *Journal of Transportation Engineering*, *130*(5), 632–636.

Berlinet, A. and Thomas-Agnan, C. (2011). *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media.

Berzuini, C., Best, N. G., Gilks, W. R., and Larizza, C. (1997). Dynamic conditional independence models and Markov chain Monte Carlo methods. *Journal of the American Statistical Association*, *92*(440), 1403–1412.

Beskos, A., Roberts, G., and Stuart, A. (2009). Optimal scalings for local Metropolis-Hastings chains on nonproduct targets in high dimensions. *The Annals of Applied Probability*, 863–898.

Box, G. E. and Tiao, G. C. (2011). *Bayesian inference in statistical analysis*, Volume 40. John Wiley & Sons.

Cappé, O., Godsill, S. J., and Moulines, E. (2007). An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, *95*(5), 899–924.

Cappé, O., Moulines, E., and Rydén, T. (2009). Inference in hidden markov models. In *Proceedings of EUSFLAT Conference*, 14–16.

Cargnoni, C., Müller, P., and West, M. (1997). Bayesian forecasting of multinomial time series through conditionally Gaussian dynamic models. *Journal of the American Statistical Association*, *92*(438), 640–647.

Carvalho, C. M., Johannes, M. S., Lopes, H. F., Polson, N. G., *et al.* (2010). Particle learning and smoothing. *Statistical Science*, *25*(1), 88–106.

Casella, G., Robert, C. P., and Wells, M. T. (2004). Generalized accept-reject sampling schemes. *Lecture Notes-Monograph Series*, 342–347.

Castro, M., Iglesias, L., Rodríguez-Solano, R., and Sánchez, J. A. (2006). Geometric modelling of highways using global positioning system (GPS) data and spline approximation. *Transportation Research Part C: Emerging Technologies*, *14*(4), 233–243.

Chen, R. and Liu, J. S. (2000). Mixture kalman filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *62*(3), 493–508.

Chen, W.-K. (2009). *Feedback, nonlinear, and distributed circuits*. CRC Press.

Chen, Y., Jiang, K., Zheng, Y., Li, C., and Yu, N. (2009). Trajectory simplification method for location-based social networking services. In *Proceedings of the 2009 International Workshop on Location Based Social Networks*, 33–40. ACM.

Chenjian RAN, Z. D. (2010). Self-tuning weighted measurement fusion Kalman filter and its convergence. *J Control Theory Application*, *4*, 435–440.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, *89*(3), 539–552.

Chopin, N., Iacobucci, A., Marin, J.-M., Mengersen, K., Robert, C. P., Ryder, R., and Schäfer, C. (2010). On particle learning. *arXiv preprint arXiv:1006.0554*.

Christen, J. A. and Fox, C. (2005). Markov chain Monte Carlo using an approximation. *Journal of Computational and Graphical statistics*, *14*(4), 795–810.

Craven, P. and Wahba, G. (1978). Smoothing noisy data with spline functions. *Numerische Mathematik*, *31*(4), 377–403.

De Boor, C., De Boor, C., Mathématicien, E.-U., De Boor, C., and De Boor, C. (1978). *A practical guide to splines*, Volume 27. Springer-Verlag New York.

De Jong, P. (1988). The likelihood for a state space model. *Biometrika*, 165–169.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1–38.

Deng, C. Y. (2011). A generalization of the Sherman–Morrison–Woodbury formula. *Applied Mathematics Letters*, *24*(9), 1561–1564.

Dongarra, J. and Sullivan, F. (2000). Guest editors introduction: The top 10 algorithms. *Computing in Science & Engineering*, *2*(1), 22–23.

Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, *90*(432), 1200–1224.

Donoho, D. L., Johnstone, I. M., and Kerkyacharian (1995). Wavelet shrinkage: asymptopia? *Journal of the Royal Statistical Society. Series B (Methodological)*, 301–369.

Donoho, D. L. and Johnstone, J. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *biometrika*, *81*(3), 425–455.

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *10*(2), 112–122.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, *79*(3), 497–516.

Einstein, A. (1956). Investigations on the theory of the Brownian Movement edited with notes by, R. Furth, translated by AD Cowper Dover Publications, Reprinted from (1905). *Ann. Phys*, *17*, 549–560.

Ellis, D., Sommerlade, E., and Reid, I. (2009). Modelling pedestrian trajectory patterns with gaussian processes. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, 1229–1234. IEEE.

Erkorkmaz, K. and Altintas, Y. (2001). High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation. *International Journal of machine tools and manufacture*, *41*(9), 1323–1345.

Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, *99*(C5), 10143–10162.

Fearnhead, P. (2002). Markov chain Monte Carlo, sufficient statistics, and particle filters. *Journal of Computational and Graphical Statistics*, *11*(4), 848–862.

Gasparetto, A. and Zanotto, V. (2007). A new method for smooth trajectory planning of robot manipulators. *Mechanism and machine theory*, *42*(4), 455–471.

Gelman, A. *et al.* (2006). Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian analysis*, *1*(3), 515–534.

Gelman, A., Jakulin, A., Pittau, M. G., and Su, Y.-S. (2008). A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 1360–1383.

Gelman, A., Roberts, G. O., Gilks, W. R., *et al.* (1996). Efficient Metropolis jumping rules. *Bayesian statistics*, *5*(599-608), 42.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 721–741.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica: Journal of the Econometric Society*, 1317–1339.

Geyer, C. J. (1992). Practical markov chain monte carlo. *Statistical science*, 473–483.

Gilks, W. R., Richardson, S., and Spiegelhalter, D. (1995). *Markov chain Monte Carlo in practice*. CRC press.

Gloderer, M. and Hertle, A. (2010). Spline-based Trajectory Optimization for Autonomous Vehicles with Ackerman drive.

Golightly, A. and Wilkinson, D. J. (2006). Bayesian sequential inference for nonlinear multivariate diffusions. *Statistics and Computing*, *16*(4), 323–338.

Gong, L. and Flegal, J. M. (2016). A practical sequential stopping rule for high-dimensional Markov Chain Monte Carlo. *Journal of Computational and Graphical Statistics*, *25*(3), 684–700.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, Volume 140, 107–113. IET.

Graves, T. L. (2011). Automatic step size selection in random walk Metropolis algorithms. *arXiv preprint arXiv:1103.5986*.

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

Greg Welch, G. B. (2006). An Introduction to the Kalman Filter. *UNC-Chapel Hill*, *TR 95-041*.

Gu, C. (1998). Model indexing and smoothing parameter selection in nonparametric function estimation. *Statistica Sinica*, 607–623.

Gu, C. (2013). *Smoothing Spline ANOVA Models Second Edition*. Springer New York Heidelberg Dordrecht London.

Haario, H., Saksman, E., and Tamminen, J. (1999). Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, *14*(3), 375–396.

Hammersley, J. M. and Handscomb, D. (1964). Percolation processes. In *Monte Carlo Methods*, 134–141. Springer.

Handschin, J. (1970). Monte Carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica*, *6*(4), 555–563.

Handschin, J. E. and Mayne, D. Q. (1969). Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International journal of control*, *9*(5), 547–559.

Hangos, K. M., Bokor, J., and Szederkényi, G. (2006). *Analysis and control of nonlinear process systems*. Springer Science & Business Media.

Hastie, T., Tibshirani, R., Friedman, J., and Franklin, J. (2008). *The elements of statistical learning: data mining, inference and prediction*. Springer.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized additive models*, Volume 43. CRC Press.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*(1), 97–109.

Higuchi, T. (2001). Self-organizing time series model. In *Sequential Monte Carlo Methods in Practice*, 429–444. Springer.

Ivanov, R. (2012). Real-time GPS track simplification algorithm for outdoor navigation of visually impaired. *Journal of Network and Computer Applications*, *35*(5), 1559–1567.

Jaynes, E. T. (1983). Papers On Probability. *Statistics and Statistical Physics*.

Jeffries, H. (1961). *Theory of probability*. Clarendon Press, Oxford.

Johansen, A. M. and Doucet, A. (2008). A note on auxiliary particle filters. *Statistics & Probability Letters*, *78*(12), 1498 – 1504.

Judd, K. L. (1998). *Numerical methods in economics*. MIT press.

Kalman, R. E. *et al.* (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, *82*(1), 35–45.

Kantas, N., Doucet, A., Singh, S., and Maciejowski, J. (2009). An overview of sequential Monte Carlo methods for parameter estimation. In *in General State-Space Models, in IFAC System Identification, no. Ml*. Citeseer.

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M. (1998). Markov chain Monte Carlo in practice: a roundtable discussion. *The American Statistician*, *52*(2), 93–100.

Katzfuss, M., Stroud, J. R., and Wikle, C. K. (2016). Understanding the ensemble Kalman filter. *The American Statistician*, *70*(4), 350–357.

Khan, Z., Balch, T., and Dellaert, F. (2005). MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE transactions on pattern analysis and machine intelligence*, *27*(11), 1805–1819.

Kijima, M. (1997). *Markov processes for stochastic modeling*, Volume 6. CRC Press.

Kim, Y.-J. and Gu, C. (2004). Smoothing spline Gaussian regression: more scalable computation via efficient approximation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *66*(2), 337–356.

Kimeldorf, G. and Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of mathematical analysis and applications*, *33*(1), 82–95.

Kimeldorf, G. S. and Wahba, G. (1970). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, *41*(2), 495–502.

Kitagawa, G. (1998). A self-organizing state-space model. *Journal of the American Statistical Association*, 1203–1215.

Komoriya, K. and Tanie, K. (1989). Trajectory design and control of a wheel-type mobile robot using B-spline curve. In *Intelligent Robots and Systems' 89. The Autonomous Mobile Robots and Its Applications. IROS'89. Proceedings., IEEE/RSJ International Workshop on*, 398–405. IEEE.

Krivobokova, T., Crainiceanu, C. M., and Kauermann, G. (2008). Fast adaptive penalized splines. *Journal of Computational and Graphical Statistics*, *17*(1), 1–20.

Lawson, C. T., Ravi, S., and Hwang, J.-H. (2011). Compression and Mining of GPS Trace Data: New Techniques and Applications. Technical report, Technical Report. Region II University Transportation Research Center.

Lindley, D. V. and Smith, A. F. (1972). Bayes estimates for the linear model. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1–41.

Liu, J. and West, M. (2001). Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo methods in practice*, 197–223. Springer.

Liu, J. S. (2008). *Monte Carlo strategies in scientific computing*. Springer Science & Business Media.

Liu, Z. and Guo, W. (2010). Data driven adaptive spline smoothing. *Statistica Sinica*, *20*(1143-1163).

Lopes, H. F. and Tsay, R. S. (2011). Particle filters and Bayesian inference in financial econometrics. *Journal of Forecasting*, *30*(1), 168–209.

Magid, E., Keren, D., Rivlin, E., and Yavneh, I. (2006). Spline-based robot navigation. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2296–2301. IEEE.

Martino, L. and Míguez, J. (2010). Generalized rejection sampling schemes and applications in signal processing. *Signal Processing*, *90*(11), 2981–2995.

Mathew, B., Bauer, A., Koistinen, P., Reetz, T., Léon, J., and Sillanpää, M. (2012). Bayesian adaptive Markov chain Monte Carlo estimation of genetic parameters. *Heredity*, *109*(4), 235.

Medova, E. (2008). *Bayesian Analysis and Markov Chain Monte Carlo Simulation*. Wiley Online Library.

Meratnia, N. and Rolf, A. (2004). Spatiotemporal compression techniques for moving point objects. In *Advances in Database Technology-EDBT 2004*, 765–782. Springer.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*(6), 1087–1092.

Mitchell, H. L. and Houtekamer, P. (2000). An adaptive ensemble Kalman filter. *Monthly Weather Review*, *128*(2), 416–433.

Müller, P. (1991). *A generic approach to posterior integration and Gibbs sampling*. Purdue University, Department of Statistics.

Nason, G. (2010). *Wavelet methods in statistics with R*. Springer Science & Business Media.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308–313.

Oussalah, M. and De Schutter, J. (2001). Adaptive Kalman filter for noise identification. *PROCEEDINGS OF THE INTERNATIONAL SEMINAR ON MODAL ANALYSIS*, *3*, 1225–1232.

Pang, S. K., Li, J., and Godsill, S. J. (2008). Models and algorithms for detection and tracking of coordinated groups. In *Aerospace Conference, 2008 IEEE*, 1–17. IEEE.

Petris, G., Petrone, S., and Campagnoli, P. (2009). Dynamic linear models. *Dynamic Linear Models with R*, 31–84.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, *94*(446), 590–599.

Polson, N. G., Stroud, J. R., and Müller, P. (2008). Practical filtering with sequential parameter learning. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *70*(2), 413–428.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). Beyond the Kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, *19*(7), 37–38.

Robert, C. P. (2004). *Monte carlo methods*. Wiley Online Library.

Robert J Elliott, Lakhdar Aggoun, J. B. M. (1995). *Estimation and control: Estimation and Control*, Volume 29 of *0172-4568*. Springer-Verlag New York.

Roberts, G. O., Gelman, A., Gilks, W. R., *et al.* (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The annals of applied probability*, *7*(1), 110–120.

Roberts, G. O., Rosenthal, J. S., *et al.* (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical science*, *16*(4), 351–367.

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003). *Semiparametric regression*. Number 12. Cambridge university press.

Schwarz, K.-P. (2012). *Geodesy Beyond 2000: The Challenges of the First Decade, IAG General Assembly Birmingham, July 19–30, 1999*, Volume 121. Springer Science & Business Media.

Sealfon, C., Verde, L., and Jimenez, R. (2005). Smoothing spline primordial power spectrum reconstruction. *Physical Review D*, *72*(10), 103520.

Septier, F., Carmi, A., Pang, S., and Godsill, S. (2009). Multiple object tracking using evolutionary and hybrid MCMC-based particle algorithms. In *15th IFAC Symposium on System Identification, 2009*, Volume 15. IFAC.

Septier, F., Pang, S. K., Carmi, A., and Godsill, S. (2009). On MCMC-based particle methods for Bayesian filtering: Application to multitarget tracking. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2009 3rd IEEE International Workshop on*, 360–363. IEEE.

Sherlock, C. (2013). Optimal scaling of the random walk Metropolis: general criteria for the 0.234 acceptance rule. *Journal of Applied Probability*, *50*(1), 1–15.

Sherlock, C., Fearnhead, P., and Roberts, G. O. (2010). The random walk Metropolis: linking theory and practice through a case study. *Statistical Science*, 172–190.

Sherlock, C., Golightly, A., and Henderson, D. A. (2016). Adaptive, delayed-acceptance MCMC for targets with expensive likelihoods. *Journal of Computational and Graphical Statistics*, (just-accepted).

Sherlock, C., Roberts, G., *et al.* (2009). Optimal scaling of the random walk Metropolis on elliptically symmetric unimodal targets. *Bernoulli*, *15*(3), 774–798.

Sherlock, C., Thiery, A., and Golightly, A. (2015). Efficiency of delayed-acceptance random walk Metropolis algorithms. *arXiv preprint arXiv:1506.08155*.

Sherman, J. and Morrison, W. J. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, *21*(1), 124–127.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1–52.

Simon, D. (2004). Data smoothing and interpolation using eighth-order algebraic splines. *Signal Processing, IEEE Transactions on*, *52*(4), 1136–1144.

Smith, A. F. (1973). A general Bayesian linear model. *Journal of the Royal Statistical Society. Series B (Methodological)*, 67–75.

Smith, A. F. and Roberts, G. O. (1993). Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, 3–23.

Sokal, A. (1997). Monte Carlo methods in statistical mechanics: foundations and new algorithms. In *Functional integration*, 131–192. Springer.

Storvik, G. (2002). Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on signal Processing*, *50*(2), 281–289.

Stroud, J. R. and Bengtsson, T. (2007). Sequential state and variance estimation within the ensemble Kalman filter. *Monthly Weather Review*, *135*(9), 3194–3208.

Stroud, J. R., Katzfuss, M., and Wikle, C. K. (2016). A Bayesian adaptive ensemble Kalman filter for sequential state and parameter estimation. *arXiv preprint arXiv:1611.03835*.

Tandeo, P., Ailliot, P., and Autret, E. (2011). Linear Gaussian state-space model with irregular sampling: application to sea surface temperature. *Stochastic Environmental Research and Risk Assessment*, *25*(6), 793–804.

Taylor, M. A., Woolley, J. E., and Zito, R. (2000). Integration of the global positioning system and geographical information systems for traffic congestion studies. *Transportation Research Part C: Emerging Technologies*, *8*(1), 257–285.

Tierney, L. (1994). Markov chains for exploring posterior distributions. *the Annals of Statistics*, 1701–1728.

Trevor Hastie, Robert Tibshirani, J. F. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition*. Springer-Verlag.

Tusell, F. (2011). Kalman Filtering in R. *Journal of Statistical Software*, *39*(2).

Vaughan, A. (2015). Goodness of Fit Test: Ornstein-Uhlenbeck Process.

Vieira, R. and Wilkinson, D. J. (2016). Online state and parameter estimation in Dynamic Generalised Linear Models. *arXiv preprint arXiv:1608.08666*.

Wahba, G. (1985). A comparison of GCV and GML for choosing the smoothing parameter in the generalized spline smoothing problem. *The Annals of Statistics*, 1378–1402.

Wahba, G. (1990). *Spline models for observational data*, Volume 59. Siam.

Wahba, G. and Wold, S. (1975). A completely automatic French curve: Fitting spline functions by cross validation. *Communications in Statistics-Theory and Methods*, *4*(1), 1–17.

Wang, Y. (1998). Smoothing spline models with correlated random errors. *Journal of the American Statistical Association*, *93*(441), 341–348.

West, M. (1993). Mixture models, Monte Carlo, Bayesian updating, and dynamic models. *Computing Science and Statistics*, 325–325.

Wood, S. N. (2000). Modelling and smoothing parameter estimation with multiple quadratic penalties. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *62*(2), 413–428.

Woodbury, M. A. (1950). Inverting modified matrices. *Memorandum report*, *42*(106), 336.

Yang, K. and Sukkarieh, S. (2010). An analytical continuous-curvature path-smoothing algorithm. *Robotics, IEEE Transactions on*, *26*(3), 561–568.

Yao, F., Müller, H.-G., Wang, J.-L., *et al.* (2005). Functional linear regression analysis for longitudinal data. *The Annals of Statistics*, *33*(6), 2873–2903.

Ying, J. J.-C., Lee, W.-C., Weng, T.-C., and Tseng, V. S. (2011). Semantic trajectory mining for location prediction. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 34–43. ACM.

Yoshimoto, F., Harada, T., and Yoshimoto, Y. (2003). Data fitting with a spline using a real-coded genetic algorithm. *Computer-Aided Design*, *35*(8), 751–760.

Yu, B., Kim, S. H., Bailey, T., and Gamboa, R. (2004). Curve-based representation of moving object trajectories. In *Database Engineering and Applications Symposium, 2004. IDEAS'04. Proceedings. International*, 419–425. IEEE.

Zamani, M. (2010). A simple 2-D interpolation model for analysis of nonlinear data. *Natural Science*, *2*(6), 641–645.

Zhang, K., Guo, J.-X., and Gao, X.-S. (2013). Cubic spline trajectory generation with axis jerk and tracking error constraints. *International Journal of Precision Engineering and Manufacturing*, *14*(7), 1141–1146.