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 sciences are, in the abstract, mathematics.
All judgments are, in their rationale, statistics.
– C. Radhakrishna Rao.

Abstract

Inference and characterization of planar trajectories have been problems that confusing researchers for years. Precise and efficient algorithms are highly demanded in all kinds of applications.

Given a set of time series GPS data from a moving object, connecting location points consequently will represent the trajectory of an individual or a vehicle. However, sparse points and data errors will give a trajectory with angels, which are unlike for a moving object. Smoothing spline methods can efficiently build up a more smooth trajectory. In conventional smoothing spline, the objective function tries to minimize errors of locations with a penalty term, who has a single parameter that controls the smoothness of reconstruction. Adaptive smoothing spline extends single parameter to a function varying in different domains and adapting the change of roughness. In Chapter 2, a new method is proposed and named Tractor spline that incorporates both location and velocity information but penalizes excessive accelerations. The penalty term is dependent on mechanic boom status. A new parameter, which controls the errors of velocity, and adjusted penalty terms, which adapts to a more complicated curvature status, are introduced to the new objective function. Additionally, an extended cross-validation technique is utilized to find all the smoothing parameters of interest. It can be seen from simulated studies that the Tractor spline has a higher accuracy than other methods. At the end of the chapter, a real data example is presented to demonstrate the effectiveness of Tractor spline.

It has been proved that the Bayesian estimates with improper priors are corresponding to smoothing splines. By constructing a reproducing kernel Hilbert space with an appropriate inner product on $[0, 1]$, the Bayesian

estimates calculated by Gaussian process regression is as the same as conventional smoothing splines. In Chapter 3, the Bayesian form for a trivial Tractor spline, whose penalty parameter is a fixed constant instead of a function, was introduced. In the particular reproducing kernel Hilbert space, in which the second derivatives are piecewise continuous, the Tractor spline is corresponding to the posterior mean of the Bayesian estimates, even though with correlated errors. As an extension to generalized cross-validation, a modified GCV formula is utilized to find the optimal parameters.

Chapter 4 takes a brief overview of existing filtering and estimation algorithms. It is well known that most algorithms for combined state and parameters estimation in state space models either estimate the states and parameters by incorporating the parameters in the state space, or by marginalizing out the parameters through sufficient statistics. Then in Chapter 5, an adaptive Markov Chain Monte Carlo (MCMC) algorithm is proposed. In the case of a linear state-space model and starting with a joint distribution over states, observations, and parameters, an MCMC sampler is implemented with two phases. In the learning phase, a self-tuning sampler is used to learn the parameter mean and covariance structure. In the estimation phase, the parameter mean and covariance structure informs the proposed mechanism and is also used in a delayed-acceptance algorithm, which greatly improves sampling efficiency. Information on the resulting state of the system is indicated by a Gaussian mixture. In the on-line mode, the algorithm is adaptive and uses a sliding window approach by cutting off historical data to accelerate sampling speed and to maintain applicable acceptance rates. This algorithm is applied to the joint state and parameters estimation in the case of irregularly sampled GPS time series data.

Acknowledgements

First of all, I am so grateful to my main supervisor Dr. Matthew Parry. He offered me this golden opportunity to take part in this joint project with the Physics department and work with him and other talented researchers. His extensive knowledge of statistics and physics leads me in the right way and encourages me throughout this difficult project. His passionate at exploring the unknown world of science motivated me and will motivate me to go further and deeper in the area of developing and implementing statistical models.

Secondly, I would want to appreciate my co-supervisor Professor David Bryant for his great help in my research progress. He offered me extraordinary supports during the time when Matthew was overseas. His guidance and expert advice inspired me in finding a different method and lighted my way into darkness.

A special gratitude goes out to the Ministry of Business, Innovation and Employment for supporting my research. This project would have been impossible without their generous fundings. Additionally, many thanks to TracMap company for providing GPS data set, 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 office mates 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 unforgettable experience together.

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. Being her father is the greatest accomplishment of my life.

Thanks for all your encouragements!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The Global Positioning System (abbreviated as GPS) is a space-based navigation system consisting of a network of 24 satellites placed in space in six different 12-hour orbital paths Agrawal and Zeng (2015), so that at least five of them are in view from every point on the globe Kaplan and Hegarty (2005) Bajaj *et al.* (2002). With these satellites, a GPS receiver sends 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. It is perhaps the most widely publicized location-sensing system providing an excellent lateration framework for determining geographic positions Hightower and Borriello (2001). Offered free of charge and accessible worldwide, GPS is used in plenty applications in military and general public, including aircraft tracking, vehicle navigation, robot localization, surveying, astronomy, and so on.

For a moving vehicle, it is often mounted with a GPS unit that communicates with these satellites and records position, speed, and direction of traveling information. With this kind of information, a maneuvering target tracking system becomes available and useful. This tracking system can be used to reduce the cost by knowing in real-time the current location of a vehicle, such as a truck or a bus Chadil *et al.* (2008), and also be useful for Intelligent Transportation System (ITS) McDonald (2006). For instance, it can be used in probe cars to measure real-time traffic data to identify the congesting area. In particular, an orchardist is able to follow the trajectory and motion patterns of tractors, which are working on an orchard or vineyard, and monitor their working status with the tracking system.

It is suggested that the smoothing spline fitting through the measured values reflecting the movements is an excellent approach, see Eubank (2004), Durbin and Koopman (2012). This algorithm can approximate data points throughout the whole process.

1.2 Smoothing Spline Based Reconstruction

Starting from interpolation, the simplest way of connecting a set of sequences $(t_1, y_1), \ldots, (t_n, y_n)$ is by straight line segments. Known as piecewise linear interpolation. Apparently, sharp turnings occur at joint knots. A smooth path is expected and more common in real life application. A single polynomial function goes through the entire interval, such as Bézier curve, is not as flexible as a combination of several polynomials, each of which is defined on subintervals and joint at certain knots. This kind of piecewise polynomial interpolation is called a spline.

The core idea of splines is to augment the vector of inputs T with additional variables, then use linear models in this space of derived input features. Adding constraints to construct basis functions $h_i(t), i = 1, 2, \ldots, n$, a linear basis expansion in T is represented as

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

The key step of a spline interpolation is the choice of basis functions. Once the $h_i(t)$ have been determined, the models are linear in the variables space.

Several kinds of splines were introduced by Trevor Hastie (2009). One of them is B-spline, short for basis spline, which is constructed from polynomial pieces and joint at knots. B-spline have a closed-form expression of positions and allows continuity 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. Once the knots are given, it is easy to compute B-spline recursively for any desired degree of the polynomial De Boor *et al.* (1978) Cox (1982). Basic simplicity of the idea is explained in Dierckx (1995) and Eilers and Marx (1996). The attractive feature of B-spline is its flexibility for univariate regression. For example, Gasparetto and Zanotto (2007) is using a fifth-order B-spline to compose the overall trajectory with a set of paired data. Roughly speaking, every spline can be represented by B-spline basis.

Another widely used spline is piecewise cubic spline, which is continuous on interval $[a, b]$ and its first and second derivatives are continues as well Wolberg (1988). For example, consider the model in the following form

$$f(t) = d_i(t - t_i)^3 + c_i(t - t_i)^2 + b_i(t - t_i) + a_i,$$

for given coefficients d_i, c_i, b_i and a_i, where $t_i \leq t \leq t_{i+1}$, $i = 1, 2, \dots, n$. f is a cubic spline on $[a, b]$ 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 Green and Silverman (1993).

To find the best estimation $\hat{f}(t)$ goes through observation y_i, $i = 1, \dots, n$, one can use regression methods returning the least sum square errors among all the sequences. Consider a model $y_i = f(t_i) + \varepsilon_i$ with random errors $\{\varepsilon_i\}_{i=1}^n \sim N(0, \sigma^2)$ in space $C^m[a, b]$, classical parametric regression assumes that f has the form $f(t, \beta)$, which is known up to the data estimated parameters β Kim and Gu (2004). When $f(t, \beta)$ is linear in β, we will have a standard linear model.

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

$$\text{RSS} = \sum_{i=1}^{n} (y_i - f(t_i))^2 + \lambda \int_a^b \left(f^{(m)}\right)^2 dt \tag{1.1}$$

for pre-specified value $\lambda > 0$ Aydin and Tuzemen (2012). In the above equation, the first term is the residual sum squares controlling the lack of fit. The second term is the roughness penalty weighted by a smoothing parameter λ, which varies from 0 to $+\infty$ and establishes a trade-off between interpolation and straight line in the following way:

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

Trevor Hastie (2009).

Hence, the cost of the equation (1.1) 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). 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 $\int f''^2$ Green and Silverman (1993).

1.3 Cross-Validation Parameter Selection

As discussed in the previous section, the determination of an optimum smoothing parameter λ in the interval $(0, +\infty)$ was found to be an underlying complication and the fundamental idea of nonparametric smoothing is to let the data choose the amount of smoothness, which consequently decides the model complexity Gu (1998). Various studies for selecting an appropriate smoothing parameter are developed and compared in literatures. 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) and recently developed methods, such as improved Akaike information criterion (AIC) Hurvich *et al.* (1998), exact risk approaches Wand and Gutierrez (1997) and so on. See e.g. Craven and Wahba (1978), Härdle *et al.* (1988), Härdle (1990), Wahba (1990), Green and Silverman (1993), Cantoni and Ronchetti (2001) Aydın *et al.* (2013).

A classical parameter selection method is called cross-validation (CV). The idea behind this method can be traced back to 1930s Larson (1931). Because in most applications, only a limited amount of data is available. Thus, an idea is to split this dataset into two subgroups, one of which is used for training the model and the other one is used to evaluating its statistical performance. The sample used in evaluation is considered as "new data" as long as data are i.i.d..

A single data split yields a validation estimate of the risk and averaging over several splits yields a cross-validation estimate Arlot *et al.* (2010). Because of the assumption that data are identically distributed, and training and validation samples are independent, CV methods are widely used in parameter selection and model evaluation.

For example, a k-fold CV splits the data into k roughly equal-sized parts. For the kth part, we fit the model to the other $k-1$ parts of the data, and calculate the prediction error of the fitted model when predicting the kth part of the data. A detailed procedure is given by Wahba and Wold (1975): suppose we have n paired data $(t_1, y_1), \ldots, (t_n, y_n)$. Run a k-fold CV according to the following algorithm 1.1. Mathematically, we denote the CV score as

Algorithm 1.1: k-fold cross-validation.

1 Initialization: Remove the first data t_1 and last date t_n from the dataset.
2 Split the rest data $t_2, \ldots, t_{n-1}$ into k groups by: Group k : $t_{k+1}, t_{2k+1}, \cdots$
3 Guess a value λ^*.
4 **while** *CV score is not optimized* **do**
5   **for** $i = 1, \ldots, k$ **do**
6   Delete the ith group of data. Fit a smoothing spline to the first data (t_1, y_1), the rest $k-1$ groups of dataset and the last data (t_n, y_n) with λ^*.
7   Compute the sum of squared deviations s_i of this smoothing spline from the deleted ith group data points.
8   **end**
9   Add the sums of squared deviations from steps 5 to 8 and divide it by k to achieve a cross-validation score of λ^* that is $s = \frac{1}{k}\sum_{i=1}^{k} s_i$.
10   Vary λ systematically and repeat steps 5 to 9 until CV shows a minimum.
11 **end**

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

where $\hat{f}^{(-k)}(t)$ denotes the fitted function computed with the kth part of the data removed. Typical choices for k are 5 and 10 Trevor Hastie (2009). The function $\mathrm{CV}(\hat{f}, \lambda)$ provides an estimate of the test error curve, and the tuning parameter λ that minimizes it will be the optimal solution.

A special case of k-fold CV is setting $k = n$, which is known as leave-one-out CV. In this scenario, the CV function takes each of the data out and calculate the errors of $\hat{f}^{(-i)}$ from the remaining $n-1$ points. In fact, the property that taking one point out does not affect the estimation, of smoothing splines fitting allows us to implement CV methods without hesitation.

As an improvement of CV, the GCV algorithm was proposed to calculate the trace of the estimation matrix $A(\lambda)$ instead of calculating individual elements for linear fitting under squared error loss, in which way it provides further computational savings. Suppose we have a solution $\hat{f} = A(\lambda)y$ with a given λ, for many linear fittings, the CV score is

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

where A_{ii} is the ith diagonal element of $A(\lambda)$. Then, the GCV approximation score is

$$\mathrm{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(t_i)}{1 - \mathrm{tr}(A)/n}\right)^2 .$$

In smoothing problems, GCV can also alleviate the tendency of cross-validation to under-smooth Trevor Hastie (2009).

Rather than λ being constant, a new challenge is posed that the smoothing parameter becomes a function $\lambda(t)$ and is varying in domains. The structure of this penalty function controls the complexity of each domain and the whole final model. Donoho *et al.* (1995) introduced adaptive splines and a method to calculate piecewise parameters and Liu and Guo (2010) give an improved formula for this method. They proposed an approximation to the penalty function with an indicator and extended the generalized likelihood to the adaptive smoothing spline. This will be another interesting research topic.

1.4 Bayesian Filtering

Smoothing spline algorithms have several advantages in inferring and characterizing planar trajectories, particularly in reconstructing. However, subject to the property that smoothing splines require the solution of a global problem that involves the entire set of points to be interpolated, it might not be suitable for on-line estimation Biagiotti and Melchiorri (2013). Therefore, it is time to use Bayesian filtering to implement instant/on-line estimation and prediction.

The word *filtering* refers to the methods for estimating the state of a time-varying system, which is indirectly observed through noisy measurements. A Bayes filter is a general probabilistic approach to infer an unknown probability density function recursively over time using incoming measurements and a mathematical process model. The concept *optimal estimation* refers to some criteria that measures the optimality in specific sense Anderson and Moore (1979). For example, a posterior estimation of $\hat{x}_t \sim p(x_t \mid y_{1:t})$ that minimizes the loss function $J_t = \mathrm{E}(x_t - \hat{x}_t)^2$ for incorrect estimates, or least mean squared errors, maximum likelihood approximation and so on Chen *et al.* (2003) Särkkä (2013). Hence, an optimal *Bayesian filtering* is using Bayesian way of formulating optimal filtering by meeting some statistical criteria.

It is no doubt that in a conventional target tracking system, the most common method is the standard Kalman filter, which is a recursive solution to the discrete data linear filtering problem.

Kalman Filter

In a discrete-time linear system, the optimal Bayesian solution coincides with the least squares solution. The successful optimal one was given by Kalman *et al.* (1960), the celebrated *Kalman filter*. It is a set of mathematical equations that provides an efficient computational means to estimate the state of a process in a recursive way by minimizing the mean of the squared errors Bishop and Welch (2001).

The Kalman filter recursively updates the estimated state by computing from the previous estimation and new input observation. In addition to eliminating the need for storing the entire past observed data, it is computationally more efficient than computing the estimate directly from the entire past observed data at each step of the filtering process Haykin *et al.* (2001).

A detailed Kalman filter and its may variants can be found in Chen *et al.* (2003), Rhodes (1971), Kailath (1981), Sorenson (1985). And Tusell *et al.* (2011) gives a review of some *R* packages, which are used to fit data with Kalman filter methods.

Consider the following model

$$\begin{array}{ccccccccc} \cdots & \rightarrow & x_{t-1} & \rightarrow & x_t & \rightarrow & x_{t+1} & \rightarrow & \cdots & \text{truth} \\ \cdots & & \downarrow & & \downarrow & & \downarrow & & \cdots & \\ \cdots & & y_{t-1} & & y_t & & y_{t+1} & & \cdots & \text{observation} \end{array} \tag{1.2}$$

in which $x_t = F(x_{t-1}) + \varepsilon_x$ is the true hidden state propagating through the transition matrix F and $y_t = G(x_t) + \varepsilon_y$ is the observation measured by the measurement matrix G of the system, where ε indicates i.i.d.noise in the model.

To estimate the filtering state x_t from $y_{1:t} = \{y_1, \ldots, y_t\}$, Bayesian wants to maximize the posterior $p(x_t \mid y_{1:t})$ by marginalizing out all the previous measurements. Given the joint distribution of $p(x_t, x_{t-1}, y_{1:t})$, Kalman filter supposes the expectation $\hat{x}_{t-1}$ and its variance S_{t-1} are known and passing through the system by $\hat{x}_t = F\hat{x}_{t-1}$ and $S_t = FS_{t-1}F^\top + Q_t$. Because of the log likelihood function is written as $\log p(x_t \mid y_{1:t}) \propto \frac{1}{2}(y_t - Gx_t)^2 - \frac{1}{2}(x_t - \hat{x}_{t-1})^2$, therefore, the solution is

$$\hat{x}_t = \left(G^\top R^{-1} G + S_{t-1}^{-1}\right)^{-1} \left(G^\top R_t^{-1} y_t + S_{t-1}^{-1}\hat{x}_{t-1}\right).$$

Additionally, by setting $S_t^{-1} = G^\top R_t^{-1} G + S_{t-1}^{-1}$, the recursive estimation of covariance matrix is $S_t = S_{t-1} - K_t G S_{t-1}$, and $K_t = S_{t-1}G^\top(R_t + GS_{t-1}G^\top)^{-1}$ is named Kalman gain matrix. Consequently, the recursive estimation is

$$\hat{x}_t = \hat{x}_{t-1} + K_t(y_t - G\hat{x}_{t-1}). \tag{1.3}$$

Furthermore, compared with filtering distribution $p(x_t \mid y_{1:t})$, the prediction distribution is trying to find an n-steps later distribution $p(x_{t+n} \mid y_{1:t})$ from current state, and the smoothing distribution is to find a specific state in the past $p(x_k \mid y_{1:t})$ for $k < t$.

Kalman Filter has a limitation that it does not apply to general non-linear model and non-Gaussian distributions. For a non-linear system, one can use the extended Kalman filter (EKF), which is widely used for solving nonlinear state estimation applications Gelb (1974), Bar-Shalom and Li (1993). The EKF uses Taylor expansion to construct linear approximations of nonlinear functions, therefore, the state transition f and observation g do not have to be linear but to be differentiable. However, in the EKF process, these approximations can introduce large errors in the true posterior mean and covariance of the transformed (Gaussian) random variable Wan and Van Der Merwe (2000).

Alternatively, the Unscented Kalman Filter (UKF) is a derivative-free method Julier and Uhlmann (1997), Wan and Van Der Merwe (2000), György *et al.* (2014). It is using the Kalman filter to create a normal distribution that approximates the result of a non-linear transformation numerically by seeing what happens to a few carefully chosen points. The unscented transform is used to recursively estimating the equation (1.3), where the state random variable is redefined as the concatenation of the original state and noise variables. In contrast, Kalman filter does not require numerical approximations.

The performance of EKF and UKF is compared in a few references regarding to different kinds of aspects, such as Chandrasekar *et al.* (2007), LaViola (2003), St-Pierre and Gingras (2004). There is not an overall conclusion that which one performs better.

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 Ran and Deng (2010). Likewise, a new adaptive Kalman filter will be another choice Oussalah and De Schutter (2001).

However, when the target maneuver occurs, Kalman filtering accuracy will be reduced or even diverged due to the model mismatch and noise characteristics that cannot be known exactly Liu *et al.* (2014). Additionally, Kalman filter based methods require the state vector contains pre-specified coefficients during the whole approximation procedure and are within the bounded definition range determined at the beginning Jauch *et al.* (2017).

A more generic algorithm is introduced in the following section.

Monte Carlo Filter

Monte Carlo filter, Chen *et al.* (2003), is a class of Monte Carlo approaches. The power of these approaches is that they can numerically and efficiently handle integration and optimization problems.

The important advantage of Monte Carlo is that a large number of posterior moments can be estimated at a reasonable computational effort and that estimates of the numerical accuracy of these results are obtained in a simple way Kloek and Van Dijk (1978). Sequential Monte Carlo method is using Monte Carlo approaches to estimate and to compute recursively. One of the attractive merits is in the fact that they allow on-line estimation by combining the powerful Monte Carlo sampling methods with Bayesian inference at an expense of reasonable computational cost Chen *et al.* (2003).

For example, consider the model (1.2) with parameter θ. The likelihood approximation is $p(y_t \mid y_{1:t-1}, \theta)$ and can be written by

$$p(y_t \mid y_{1:t-1}, \theta) = \int p(y_t \mid x_t, \theta) p(x_t \mid y_{1:t-1}, \theta) dx_t = \mathrm{E}\left[p(y_t \mid x_t, \theta)\right].$$

The standard Monte Carlo algorithm is trying to compute the integration by drawing N independent samples $x_t^{(i)}$ from $p(x_t \mid y_{1:t-1}, \theta)$ at first and then, by adding them up, to approximate the integration for large N in the following way

$$\mathrm{E}\left[p(y_t \mid x_t, \theta)\right] \approx \frac{1}{N} \sum_i p(y_t \mid x_t^{(i)}, \theta),$$

Kalos and Whitlock (2008).

In terms of getting good samples of $x_t^{(i)}$, which can be used for representing $p(y_t \mid x_t^{(i)}, \theta)$, an importance sampling method was devised. The idea of this method is by assigning weights $W_t^{(i)}$ to samples, the most important ones are evaluated for computing the integral. Further, sequential importance sampling (SIS) allows a sequential update of the importance weights by

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

with an appropriate chosen *proposal distribution* $q(x_t \mid x_{t-1}, y_t)$. It is also called *importance density* or *important function* Chen *et al.* (2003).

Nevertheless, the SIS makes samples skewed that only a few samples have proper weights as time increases and most of them have small but positive weights. This phenomenon is often called *weight degeneracy* or *sample impoverishment* Green (1995) Berzuini *et al.* (1997).

Besides the SIS processes, a resampling step, also called as selection step, is trying to eliminate the samples with small weights and duplicate the samples with large weights in a principled way Rubin (2004), Tanner and Wong (1987). This method is named SIR. Suppose samples with associated weights are $\{x_t^{(i)}, W_t^{(i)}\}$, a resampling step is executed by generating new samples $\tilde{x}_t^{(i)}$ according to normalized weights $\tilde{W}_t^{(i)}$. It is pointed out that the resampling step does not prevent weights degeneracy but improve further calculation.

The common feature of SIS and SIR is that both of these methods are based on importance sampling and updating samples weights recursively. The difference is that in SIR, the resampling step is always performed. Whereas in SIS, resampling is only taken when needed.

The most successful application of importance sampling with resampling algorithm is *Particle filter* (PF). PF randomly generates a cloud of points and push these points through the computation process. It is a recursive implementation of the Monte Carlo approaches Doucet and Johansen (2009).

A generic PF firstly generates N uniformly weighted random measurements $\{x_{t-1}^{(i)}, \frac{1}{N}\}$ at time $t-1$. Once a new observation y_t comes into the system, the weights will be updated recursively by involving the likelihood function $p(y_t \mid x_t^{(i)})$ and propagation function $p(x_t^{(i)} \mid x_{t-1}^{(i)})$. In fact, it is the SIS step. To monitor how bad is the weight degeneracy, a suggested measurement *effective sample size* was introduced in Kong *et al.* (1994). It is the reciprocal of sum squared weights in the form of

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N}\left(W_t^{(i)}\right)^2}.$$

If the N_{eff} is less than a predefined threshold, the resampling procedure is executed and the set of particles remains the same size N.

However, the PF sampling and re-sampling methods may cause practical problems. Such as high weighted particles have been selected many times and lead to the loss of diversity. This problem is known as sample impoverishment, in which way the particles are not representative. The improvements of particle filter's performance have been devoted by Carpenter *et al.* (1999), Godsill *et al.* (2001), Stavropoulos and Titterington (2001), Arnaud Doucet (2011).

Apparently, Bayesian filtering has become a broad topic involving many scientific areas that a comprehensive survey and detailed treatment seems crucial to cater the ever growing demands of understanding this important field for many novices, though it is noticed by the author that in the literature there exist a number of excellent tutorial papers on particle filters and Monte Carlo filters Chen *et al.* (2003) Doucet *et al.* (2000) Doucet *et al.* (2000) Chen *et al.* (2012).

1.5 Markov Chain Monte Carlo Methods

Schemes exist to counteract sample impoverishment, which incurs in particle filter Ristic *et al.* (2004). One approach is to consider the states for the particles to be predetermined by the forward filter and then to obtain the smoothed estimates by recalculating the particles' weights via a recursion from the final to the first time step Godsill *et al.* (2000). Another approach is to use a *Markov chain Monte Carlo* (MCMC) move step Carlin *et al.* (1992).

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 Andrieu *et al.* (1999), Green (1995), Andrieu *et al.* (2001). They are based on sampling from probability distributions based on a Markov chain. If samples are unable to be drawn directly from a distribution $\pi(x)$, we can construct a Markov chain of samples from another distribution $\hat{\pi}(x)$ that is equilibrium to $\pi(x)$. If the chain is long enough, these samples of the chain can be used as a basis for summarizing features of $\pi(x)$ of interest Smith and Roberts (1993). This is a crucial property. See e.g. Cappé *et al.* (2009) and Liu (2008) for details.

Metropolis-Hastings Algorithm

Metropolis-Hastings algorithm is an important class of MCMC algorithms Smith and Roberts (1993) Tierney (1994) Gilks *et al.* (1995). Given essentially a probability distribution π (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, a function $\alpha(x, y)$ is introduced for 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

$$\alpha = \frac{\pi(x')}{\pi(x^{(t)})} \frac{q(x^{(t)}, x')}{q(x', x^{(t)})}. \tag{1.4}$$

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

A simple mechanic proposing algorithm is *Random Walk Metropolis-Hastings* (RM MH). It is easy to implement and symmetric under the exchange of the initial and proposed points.

Besides, modified Metropolis-Hastings algorithms, such as the delayed-rejection MH, multiple-try MH and reversible-jump MH algorithms have been studied by Tierney and Mira (1999), Liu *et al.* (2000) and Green (1995).

Adaptive MCMC Algorithm

Metropolis-Hastings algorithm is widely used in statistical inference, to sample from complicated high-dimensional distributions. Typically, this algorithm has parameters that must be tuned in each new situation to obtain reasonable mixing times, such as the step size in a random walk Metropolis Mahendran *et al.* (2012). Tuning of associated parameters such as proposal variances is crucial to achieving efficient mixing, but can also be difficult.

Adaptive MCMC methods have been developed to automatically adjust these parameters. See e.g. Andrieu and Thoms (2008), Atchade *et al.* (2009), Roberts and Rosenthal (2009). One of the most successful adaptive MCMC algorithms was introduced by Haario *et al.* (2001), where, based on the random walk Metropolis algorithm, the covariance of the proposal distribution is calculated using all of the previous states. For instance, with an AM chain $x_0, x_1, \ldots, x_t$, the proposal x' is from $N(\cdot \mid x_t, R_t)$, where R_t is the covariance matrix determined by the spatial distribution of the state $x_0, x_1, \ldots, x_t$.

Even though the adaptive Metropolis algorithm is non-Markovian, the establishment was verified that the AM algorithm indeed has the correct ergodic properties.

A Bayesian optimization for adaptive MCMC was proposed by Mahendran *et al.* (2012). The author proposed adaptive strategy consists of two phases: adaptation and sampling. In the first phase, Bayesian optimization is used to construct a randomized policy. After that, in the second phase, a mixture of MCMC kernels selected according to the learned randomized policy is used to explore the target distribution.

Further investigation in the use of adaptive MCMC algorithms to automatically tune the Markov chain parameters can be found at Roberts and Rosenthal (2009).

Other Monte Carlo Algorithms

The *Hamiltonian Monte Carlo* (HMC), devised by Duane *et al.* (1987) as hybrid Monte Carlo, is using Hamiltonian dynamics to produce distant proposals for the Metropolis algorithm in order to avoid slow exploration of the state space that results from the diffusive behavior of simple random walk proposals Neal *et al.* (2011). In practice, the HMC sampler is more efficient for sampling in high-dimensional distributions than MH.

The key feature of HMC is the Hamiltonian system equation as follows:

$$H(x, v) = U(x) + K(v),$$

which is consisting of potential energy $U(x)$ with a d-dimensional momentum vector (position) x and kinetic energy function $K(v) = \frac{v^\top M^{-1} v}{2}$ with a d-dimensional momentum vector (velocity) v. To propose $\{x', v'\}$, HMC is using leapfrog method, which is based on Euler's method and modified Euler's method, to increase the proposing accuracy Betancourt (2017). It is accepted with the probability

$$\alpha = \min\{\exp\left(H(x, v) - H(x', v')\right), 1\}.$$

Compared with MH sampler, the HMC has a higher efficiency in most of the high-dimensional cases. It is incorporating not only with energy $U(x)$ but also with a gradient. In this way, HMC explores a larger area and converges to balance faster.

The *Zig-Zag Monte Carlo* is using a continuous-time piecewise zig-zag process to increase the sampling efficiency Bierkens *et al.* (2016b). It is an application of the Curie-Weiss model, Turitsyn *et al.* (2011), in high dimension and provides a practically efficient sampling scheme for sampling in the big data regime with some remarkable properties Bierkens and Duncan (2017).

Given a target density $\pi(\cdot)$, the Zig-Zag process $f(x, \theta)$ is defined in a $d \otimes 2$ space $\mathcal{E}$. x is in the d-dimensional topological subspace and θ is in a binary discrete $\{-1, 1\}_d$ subspace denoting the flipping statues. The switching rate $\lambda(x, \theta)$ agrees with the target distribution $\pi(\cdot)$ in a certain way and is defined as $\lambda(x, \theta) = \max\{0, \theta U'(x)\} + \gamma(x)$, where $U'(x) = \lambda(x, \theta) - \lambda(x, -\theta)$. Then, the Zig-Zag operator L is

$$Lf(x, \theta) = \theta \partial f_x + \lambda(x, \theta) \left(f(x, -\theta) - f(x, \theta) \right),$$

for all $(x, \theta) \in \mathcal{E}$.

Thereafter, the obtained sequence of the Zig-Zag process is used to approximate expectations with respect to $\pi(\cdot)$ according to the law of large numbers. The application of Zig-Zag process in big data scheme and some properties are given in Bierkens and Duncan (2017) and Bierkens *et al.* (2016a).

The t-walk given by Christen *et al.* (2010) is a self-adjusting MCMC algorithm that requires no tuning and has been shown to provide good results in many cases of up to 400 dimensions. Because of the t-walk is not adaptive, it does not require new restricting conditions but only the log of the posterior and two initial points Blaauw *et al.* (2011).

Given a posterior distribution $\pi(\cdot)$, the new objective function $f(x, x')$ is the product of $\pi(x)\pi(x')$ from $X \otimes X$. The new proposal (y, y') is given by

$$(y, y') = \begin{cases} (x, h(x', x)) & \text{with probability } 0.5 \\ (h(x', x), x') & \text{with probability } 0.5 \end{cases}$$

where $h(x, x')$ is a preselected proposing strategy. In each iteration, only one of the two chains x and x' moves according to a random walk. For example, suppose in the first step the x stays the same but y' is proposed from $q(\cdot \mid x, x')$, then the acceptance ratio is

$$\frac{\pi(y')}{\pi(x')} \frac{q(x' \mid y', x)}{q(y' \mid x', x)}.$$

After a few iterations, there are two dual and coupled chains obtained. Hence, the t-walk is a kind of multiple chain approach.

Four recommendations for the choices of $h(x, x')$ including a scaled random walk, referred to *the walk move*, *traverse move*, *hop moves* and *blow moves* are given in Christen *et al.* (2010). The t-walk is now available in a complete set of computer packages, including *R*, *Python*. It is convenient for researchers to go a deeper implementation.

1.6 Outline of Thesis

The structure of this thesis is organized in the following order.

In chapter 2, I propose an adaptive smoothing spline method based on *Hermite* spline basis functions to obtain a reconstruction of f and f' from noisy data $y_{1:n}$ and $v_{1:n}$. Instead of minimizing the residuals of $f(t_i) - y_i$ only, the residuals of $f'(t_i) - v_i$ with a new parameter γ are consisted in the new objective function. A modified leave-one-out cross-validation algorithm is used for find the optimal parameters. Numerical simulation and real data implementation are given after theoretical methodology.

In chapter 3, the Bayesian estimation form of the Tractor spline is given. It is proved that the Bayesian estimate is corresponding to a trivial Tractor spline in the reproducing kernel Hilbert space $\mathcal{C}^2_{\text{p.w.}}[0,1]$, where the second-derivative is piecewise-continuous. An extended GCV is used to find the optimal parameters for the Bayesian estimate.

In chapter 4, I give a brief overview of existing methods for sequential state and parameters inference. Basic concepts and popular algorithms on sequential state estimation are introduced firstly. Furthermore, the algorithms that can estimate combined unknown state and parameters are brought into a separate section. Numerical comparison studies are given at the end of this chapter.

In chapter 5, it is using a random walk Metropolis-Hastings algorithm in a learning phase to learn the mean of covariance of the parameters space. After that, the information is implemented in the estimation phase, where an adaptive Delayed-Acceptance Metropolis-Hastings algorithm is proposed for estimating the posterior distribution of combined states and parameters. To remain a high running efficiency, a sliding window approach, in which way historical data is cut off when new observations come into the data stream, is used to improve the sampling speed. This algorithm is applied to irregularly sampled time series data and implemented in real GPS data set.

Theorems proofs, equations calculations, and some simulation results are presented in appendices.

Chapter 2

Adaptive Smoothing Tractor Spline for Trajectory Reconstruction

2.1 Introduction

GPS devices are widely used for tracking individuals and vehicles position. Objects move frequently and continuously with their position and moving status being reported to a 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 through 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 omnidirectional 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 the 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 $(y_1, v_1), (y_2, v_2), \ldots, (y_n, v_n)$ Yu *et al.* (2004). More generally, B-spline has 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 the 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:

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - f(t_i))^2 + \lambda \int_a^b (f''(t))^2 \, dt \tag{2.1}$$

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 λ, 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 λ, 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

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

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 squared 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

$$\sum_{i=1}^{n} (y_i - f(t_i))^2 + \int_T \lambda(t) \left(f''(t)\right)^2 dt, \tag{2.3}$$

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 has 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 of each domain and the whole final model. Liu and Guo (2010) proposed to approximate the penalty function with an indicator and extended the generalized likelihood to the adaptive smoothing spline.

In this chapter, I propose an adaptive smoothing spline method based on Hermite spline basis functions to obtain a reconstruction of f and f' from n paired time series noisy data $\mathbf{y} = \{y_1, \ldots, y_n\}$ and $\mathbf{v} = \{v_1, \ldots, v_n\}$. Rather than only using residuals of $f(t_i) - y_i$, an extra residuals of $f'(t_i) - v_i$ with a new parameter γ in our objective function is included. In this way, the spline keeps a balance between position and velocity. By using new basis functions, the new smoothing spline is reconstructed on the whole interval $[a, b]$. Derived from the new objective function, an advanced cross-validation formula for both $f(t)$ and $f'(t)$ is given. It is shown that the new spline performs well on simulated signal data *Blocks*, *Bumps*, *HeaviSine* and *Doppler* Donoho and Johnstone (1994). At the end of this chapter, an application of this smoothing spline on a set of 2-dimensional real data is given. It obtains a better reconstruction. This method can be used in either getting the true signal from noisy data or reconstructing the trajectory of a moving object.

2.2 Tractor spline

In the nonparametric regression, consider n paired time series points $\{t_1, y_1, v_1\}$, …, $\{t_n, y_n, v_n\}$, such that $a \le t_1 < t_2 < \cdots < t_n \le b$, y is the position information and v indicates its velocity. We define a positive piecewise constant function $\lambda(t)$:

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

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

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

where γ 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 named Tractor spline. The proof of Theorem 1 is in appendix A.2.

2.2.1 Basis Functions

Suppose there is a time series sequence of observed dataset $a \leq t_1 < t_2 < \cdots < t_n \leq b$. The function $f(t)$ 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}), $i = 1, \ldots, n-1$, $f(t)$ is a cubic polynomial, but on interval (a, t_0) and (t_n, b) is 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, $i = 1, \ldots, n-1$.

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 $i = 1, 2, \ldots, n-2$,

$$N_{2i+1} = \begin{cases} h_{01}^{(i)} + h_{00}^{(i+1)} & \text{if } t < t_n \\ 0 & \text{if } t = t_n \end{cases},$$

$$N_{2i+2} = \begin{cases} h_{11}^{(i)} + h_{10}^{(i+1)} & \text{if } t < t_n \\ 0 & \text{if } t = t_n \end{cases},$$

and

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

$$N_{2n} = \begin{cases} h_{11}^{(n-1)} & \text{if } t < t_n \\ 0 & \text{if } t = t_n \end{cases}.$$

Theorem 2. *On $[t_1, t_n]$, the functions $N_1, \ldots, N_{2n}$ provide a set of basis functions for the set of functions which are cubic on each interval $[t_i, t_{i+1}]$, $i = 1, \ldots, n-1$, and continuous at joint knots, thus it is continuous on the entire interval $[t_1, t_n]$.*

The proof of theorem 2 is in appendix A.3. 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

$$f = \sum_{k=1}^{2n} \theta_k N_k(t), \tag{2.11}$$

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

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

From the definition of basis functions, it can be seen that at a joint knot of two neighbor intervals, Hermite spline basis functions share the same y_i and v_i. With this property, we construct Tractor spline basis functions. There are two parameters at each joint knots and 4 at the start and end knots. Hence, the degrees of freedom for parameters is $2n - 4 + 4 = 2n$. However $2(n-2)$ constraints are added to the joint knots, at 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. Consequently, parameters' the degrees of freedom is 2.

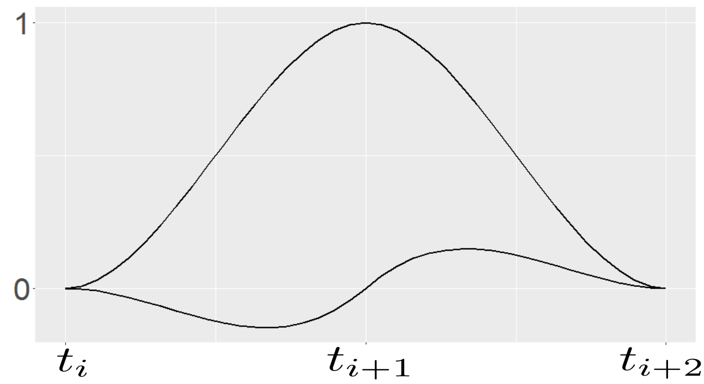


Figure 2.1: The two basis functions N_{2i+1} and N_{2i+2} on an arbitrary interval $[t_i, t_{i+2})$. It is apparently that these basis functions are continuous on this interval and have continuous first derivatives.

2.2.2 Solution to The Objective Function

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

$$\mathrm{MSE}(\theta, \lambda, \gamma) = (\mathbf{y} - \mathbf{B}\theta)^\top(\mathbf{y} - \mathbf{B}\theta) + \gamma(\mathbf{v} - \mathbf{C}\theta)^\top(\mathbf{v} - \mathbf{C}\theta) + n\theta^\top\mathbf{\Omega}_\lambda\theta, \tag{2.12}$$

where $\{\mathbf{B}\}_{ij} = N_j(t_i)$, $\{\mathbf{C}\}_{ij} = N_j'(t_i)$ and $\{\Omega_{2n}^{(i)}\}_{jk} = \int_{t_i}^{t_{i+1}} \lambda_i N_j''(t) N_k''(t) dt$. After substituting the series observation $t_1, \ldots, t_n$ into basis functions, we get $N_1(t_1) = 1, N_1(t_2) = 0, \ldots, N_{2i-1}(t_i) = 1, N_{2i}(t_i) = 0, \ldots, N_{2n-1}(t_n) = 1, N_{2n}(t_n) = 0$; and into first derivative of basis functions, we get $N_1'(t_1) = 0, N_1'(t_2) = 1, \ldots, N_{2i-1}'(t_i) = 0, N_{2i}'(t_i) = 1, \ldots, N_{2n-1}'(t_n) = 0, N_{2n}'(t_n) = 1$. That means the matrices $\mathbf{B}$ and $\mathbf{C}$ in MSE equation (2.12) are $n \times 2n$ dimensional and the elements are

$$\mathbf{B} = \{B\}_{ij} = \begin{cases} 1, & j = 2i - 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

$$\mathbf{C} = \{C\}_{ij} = \begin{cases} 1, & j = 2i \\ 0, & \text{otherwise} \end{cases} \tag{2.14}$$

where $i = 1, \ldots, n$, $j = 1, \ldots, 2n$ and $k = 1, \ldots, 2n$. The i-th $\Omega^{(i)}$ on the interval $[t_i, t_{i+1}]$ is a $2n \times 2n$ matrix and $\Omega^{(n)}$ does not exist. Its detail is in appendix A.1. As a result, the penalty term is

$$\mathbf{\Omega}_\lambda = \sum_{i=1}^{n-1} \lambda_i \Omega^{(i)}, \tag{2.15}$$

which is a bandwidth four matrix.

The solution to (2.12) is easily seen to be

$$\hat{\theta} = (\mathbf{B}^\top\mathbf{B} + \gamma\mathbf{C}^\top\mathbf{C} + n\mathbf{\Omega}_\lambda)^{-1}(\mathbf{B}^\top\mathbf{y} + \gamma\mathbf{C}^\top\mathbf{v}) \tag{2.16}$$

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

$$\hat{f}(t) = \sum_{k=1}^{2n} N_k(t)\hat{\theta}_k \tag{2.17}$$

A smoothing spline with parameters $\lambda(t)$ and γ is an example of a linear smoother Trevor Hastie (2009). This is because the estimated parameters in equation (2.16) 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

$$\begin{aligned}\hat{\mathbf{f}} &= \mathbf{B}(\mathbf{B}^\top\mathbf{B} + \gamma\mathbf{C}^\top\mathbf{C} + n\mathbf{\Omega}_\lambda)^{-1}(\mathbf{B}^\top\mathbf{y} + \gamma\mathbf{C}^\top\mathbf{v}) \\ &\triangleq \mathbf{S}_{\lambda,\gamma}\mathbf{y} + \gamma\mathbf{T}_{\lambda,\gamma}\mathbf{v}\end{aligned} \tag{2.18}$$

$$\begin{aligned}\hat{\mathbf{f}}' &= \mathbf{C}(\mathbf{B}^\top\mathbf{B} + \gamma\mathbf{C}^\top\mathbf{C} + n\mathbf{\Omega}_\lambda)^{-1}(\mathbf{B}^\top\mathbf{y} + \gamma\mathbf{C}^\top\mathbf{v}) \\ &\triangleq \mathbf{U}_{\lambda,\gamma}\mathbf{y} + \gamma\mathbf{V}_{\lambda,\gamma}\mathbf{v}\end{aligned} \tag{2.19}$$

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 γ.

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 ξ and $M \ll N$. Thus the vector of fitted spline values is given by

$$\hat{\mathbf{f}} = \mathbf{B}_\xi(\mathbf{B}_\xi^\top\mathbf{B}_\xi)^{-1}\mathbf{B}_\xi\mathbf{y} = \mathbf{H}_\xi\mathbf{y} \tag{2.20}$$

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

$$(\mathbf{B}^\top\mathbf{B} + \gamma\mathbf{C}^\top\mathbf{C} + n\mathbf{\Omega}_\lambda)^{-1} = \mathbf{R}\mathbf{R}^\top, \tag{2.21}$$

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.

Corollary 1. *If $f(t)$ is the Tractor spline on the entire interval $[t_1, t_n]$, for sufficient cases of $\lambda(t)$ and γ, $f(t)$ 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.*

The proof of Corollary 1 is in appendix A.4.

2.2.3 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 that 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 the best fit which returns smallest errors to the objective function. While with the same parameter $\lambda(t)$ and γ, $f_y(t)$ 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 Δd_i and Δt_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, α and β 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 through time on x or y axes and a dot on the higher dimension or keep moving with the same speed, which returns a linear instead of a curved path.

Consequently, the final form of the penalty function is

$$\lambda(t) = \frac{(\Delta t_i)^3}{(\Delta d_i)^2}\lambda, \tag{2.22}$$

where $t_i \leq t < t_{i+1}$. Eventually, in objective function, there is one parameter λ controlling the curvature of Tractor spline on different domains, and another one γ 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, 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 λ is the smoothing parameter. Then $\hat{f}^{(-i)}(t,\lambda)$ is the minimizer of

$$\frac{1}{n}\sum_{j\neq i}(y_j - f(t_j))^2 + \lambda\int (f'')^2 dt, \tag{2.23}$$

and can be quantified by the cross-validation score function

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

The basis idea of the cross-validation is to choose the value of λ that minimizes $\mathrm{CV}(\lambda)$ Green and Silverman (1993).

An efficient way to calculate the cross-validation score is introduced by Green and Silverman (1993). Through the equation (2.20), it is known 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

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

and the following lemma.

Lemma 1. *The cross-validation score satisfies*

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

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

For a Tractor spline and its MSE function, there are two parameters $\lambda(t)$ and γ to be estimated for. Therefore, $\hat{f}^{(-i)}(t,\lambda)$ is the minimizer of

$$\frac{1}{n}\sum_{j\neq i}(y_j - f(t_j))^2 + \frac{\gamma}{n}\sum_{j\neq i}(v_j - f'(t_j))^2 + \int \lambda(t)\left(f''\right)^2 dt, \tag{2.24}$$

and the cross-validation score function is

$$\mathrm{CV}\left(\lambda(t),\gamma\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}^{(-i)}\left(t_i,\lambda(t),\gamma\right)\right)^2. \tag{2.25}$$

Additionally, it is known that the parameter $\hat{\theta} = (B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}(B^\top\mathbf{y} + \gamma C^\top\mathbf{v})$ and will give us the following form

$$\begin{aligned}\hat{\mathbf{f}} &= B\hat{\theta} = B(B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}B^\top\mathbf{y} + B(B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}C^\top\mathbf{v} \\ &= S\mathbf{y} + \gamma T\mathbf{v},\end{aligned} \tag{2.26}$$

$$\begin{aligned}\hat{\mathbf{f}}' &= C\hat{\theta} = C(B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}B^\top\mathbf{y} + C(B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}C^\top\mathbf{v} \\ &= U\mathbf{y} + \gamma V\mathbf{v}.\end{aligned} \tag{2.27}$$

From Lemma 1, we can prove the following theorem:

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

$$CV(\lambda(t),\gamma) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{f}(t_i) - y_i + \gamma\frac{T_{ii}}{1-\gamma V_{ii}}(\hat{f}'(t_i) - v_i)}{1 - S_{ii} - \gamma\frac{T_{ii}}{1-\gamma V_{ii}}U_{ii}}\right)^2 \tag{2.28}$$

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

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 $\lambda(t), \gamma$ and i, denote $\mathbf{f}^{(-i)}$ by the vector with components $f_j^{(-i)} = \hat{f}^{(-i)}(t_j, \lambda(t), \gamma)$, $\mathbf{f}'^{(-i)}$ by the vector with components $f_j'^{(-i)} = \hat{f}'^{(-i)}(t_j, \lambda(t), \gamma)$, and define vectors $\mathbf{y}^*$ and $\mathbf{v}^*$ by*

$$\begin{cases} y_j^* = y_j & j \neq i \\ y_i^* = \hat{f}^{(-i)}(t_i) & \textit{otherwise} \end{cases} \tag{2.29}$$

$$\begin{cases} v_j^* = v_j & j \neq i \\ v_i^* = \hat{f}'^{(-i)}(t_i) & \textit{otherwise} \end{cases} \tag{2.30}$$

Then

$$\hat{\mathbf{f}}^{(-i)} = S\mathbf{y}^* + \gamma T \mathbf{v}^* \tag{2.31}$$

$$\hat{\mathbf{f}}'^{(-i)} = U\mathbf{y}^* + \gamma V \mathbf{v}^* \tag{2.32}$$

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 the convenience of reinitialization and repeatable comparison, we set random seed at 2016. The noises are i.i.d.zero-mean Gaussian distributed with standard deviation regarding to signal-to-noise ratio (SNR), which specifies the ratio of the standard deviation of the function to the standard deviation of the simulated errors. Explicitly, if the standard deviation of the true signal f is σ_f, the simulated data will be $f+\varepsilon$, where the simulated error $\varepsilon \sim N(0, \sigma_f/SNR)$.

Therefore, if the original function $g(t)$ is treated as velocity function $f'(t) = g(t)$, thus by setting initial position $y_0 = 0$, acceleration $a_0 = 0$ and using the following formula

$$f(t_{i+1}) = f(t_i) + (g(t_i) + g(t_{i+1}))\frac{t_{i+1} - t_i}{2} \tag{2.33}$$

for calculating position information, we can easily generate simulated data. Further, we add some i.i.d.zero-mean ε noises with SNR to them to get the measurements by

$$\begin{aligned} y_i &= f(t_i) + \varepsilon_f, \\ v_i &= g(t_i) + \varepsilon_g, \end{aligned} \tag{2.34}$$

where $\varepsilon_f \sim N(0, \sigma_f/SNR)$ and $\varepsilon_g \sim N(0, \sigma_g/SNR)$. The value of SNR can be chosen 7 or 3. For wavelet transform reconstructions, we use the threshold policy of *sure* and *BayesThresh* with levels $l = 4, \ldots, 9$, 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 λ and γ to optimize. To evaluate the performance of the velocity term in objective function (2.5) and the adjusted penalty term in (2.22), the parameter γ is set as 0 in one reconstruction of Tractor spline, whose objective function and solution become

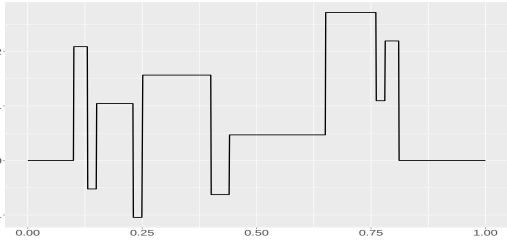
$$J[f]_{\gamma=0} = \frac{1}{n}\sum_{i=1}^{n}(f(t_i) - y_i)^2 + \sum_{i=1}^{n-1}\lambda_i \int_{t_i}^{t_{i+1}} (f'')^2 dt, \tag{2.35}$$

and

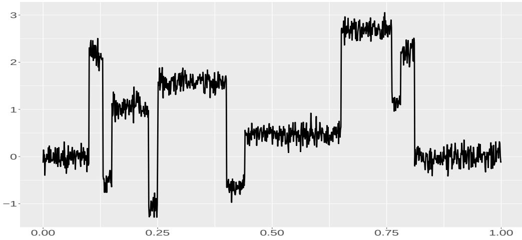
$$\hat{\theta}_{\gamma=0} = (\mathbf{B}^\top \mathbf{B} + n\Omega_\lambda)^{-1}\mathbf{B}^\top \mathbf{y} \tag{2.36}$$

and the adjusted penalty term in (2.22) 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 λ and γ of a Tractor spline are automatically selected from formula (2.28) by `optim` function in *R* Nelder and Mead (1965).

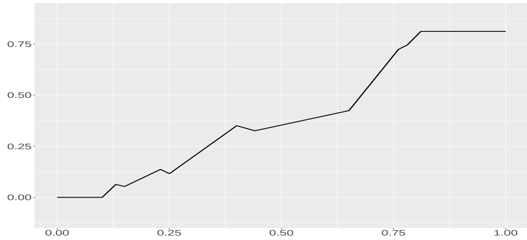
By comparing, we can see that all these methods can rebuild up the skeleton of generated trajectory. *Wavelet(sure)* method has more wiggles in interior interval than *Wavelet(BayesThresh)* does, and the latter one becomes fluctuation near boundary



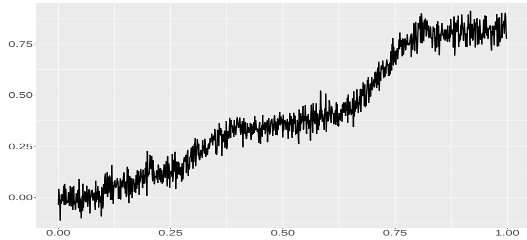
(a) The *Blocks* function



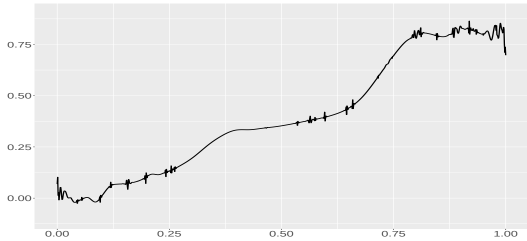
(b) Noisy *Blocks* at *SNR*=7



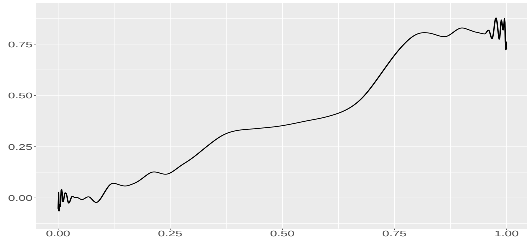
(c) Generated positions



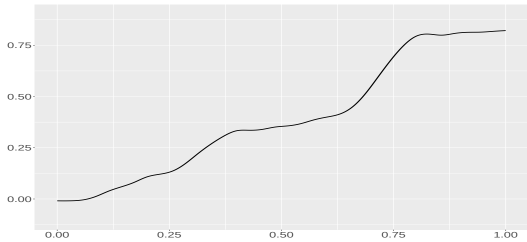
(d) Noisy position at *SNR*=7



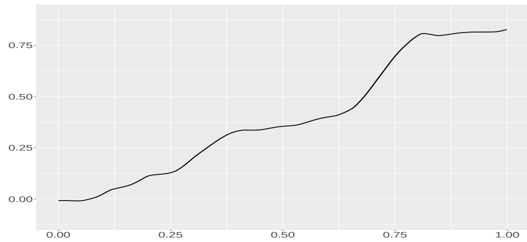
(e) Reconstruction from Wavelet by sure threshold



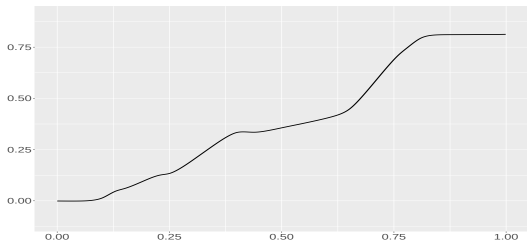
(f) Reconstruction from Wavelet by BayesThresh approach



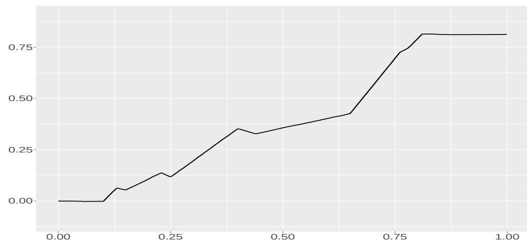
(g) Reconstruction by P-spline



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

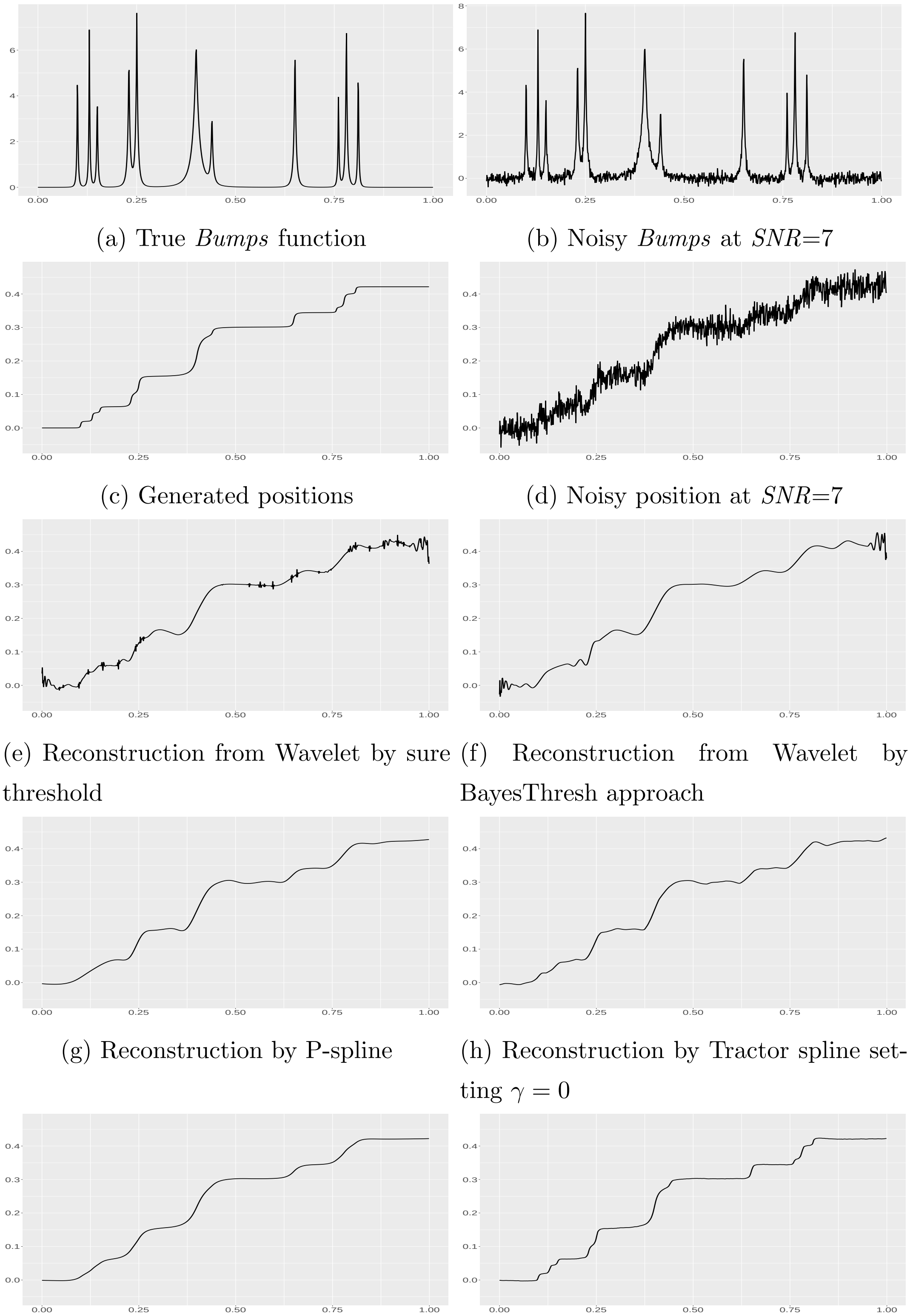


(i) Reconstruction by Tractor spline with conventional penalty term



(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

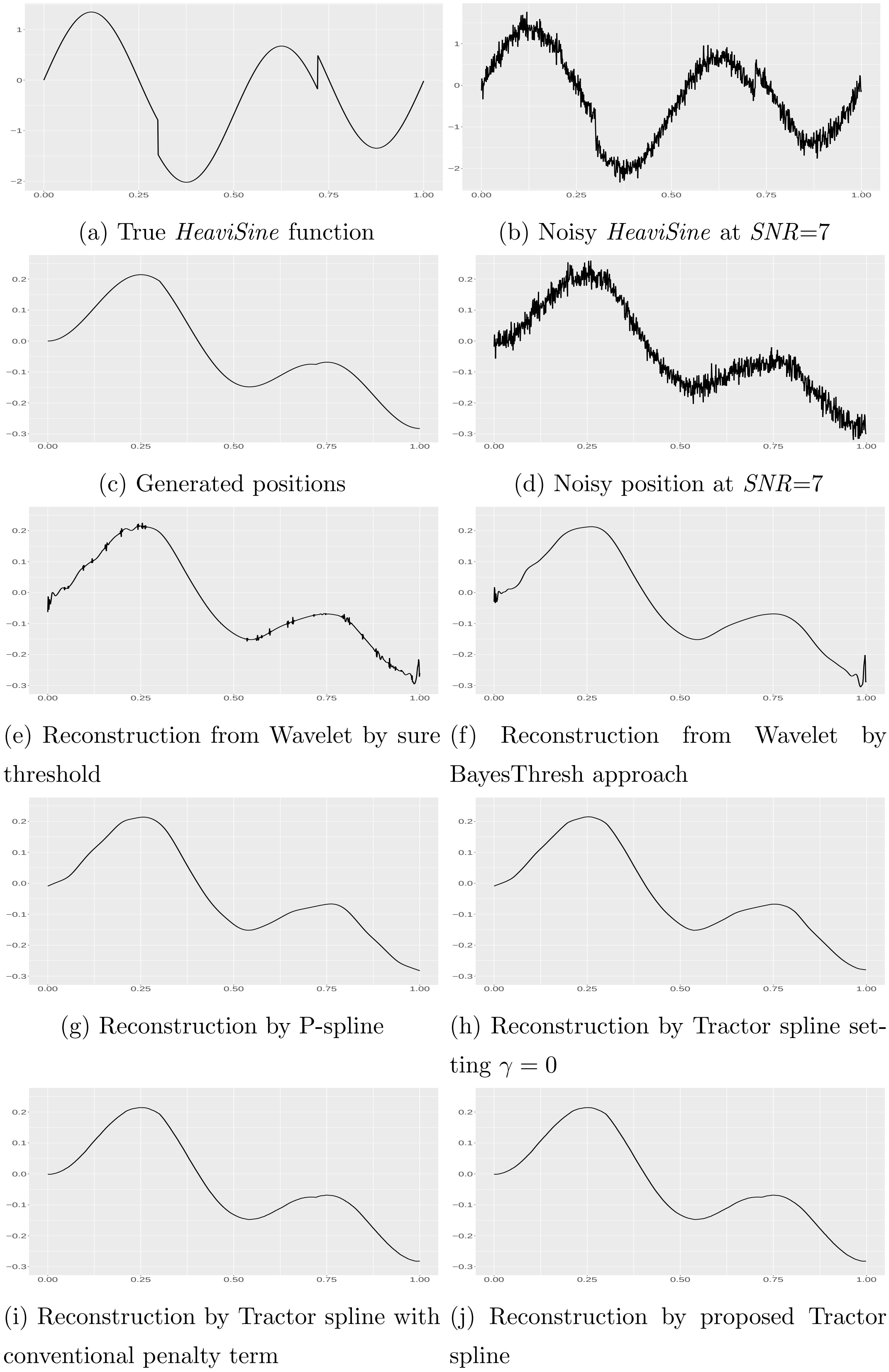
(g) Reconstruction by P-spline

(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

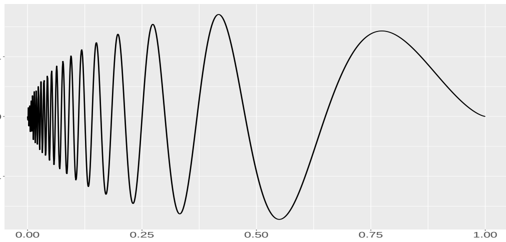
(g) Reconstruction by P-spline

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

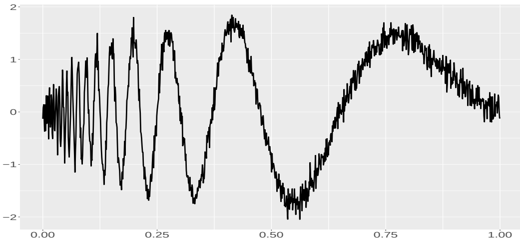
(i) Reconstruction by Tractor spline with conventional penalty term

(j) Reconstruction by proposed Tractor spline

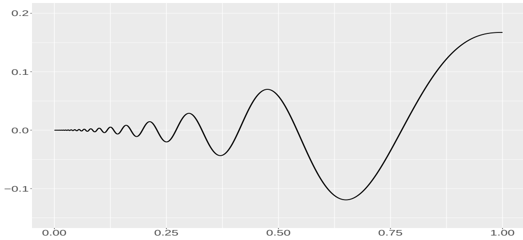
Figure 2.4: Numerical example: *HeaviSine*. Comparison of different reconstruction methods with simulated data.



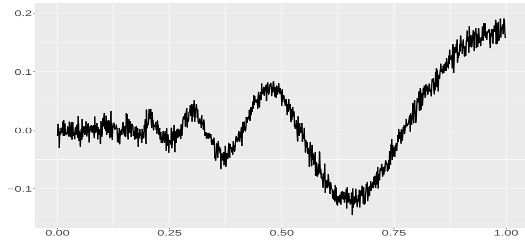
(a) True *Doppler* function



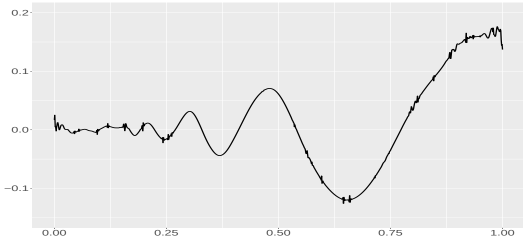
(b) Noisy *Doppler* at *SNR*=7



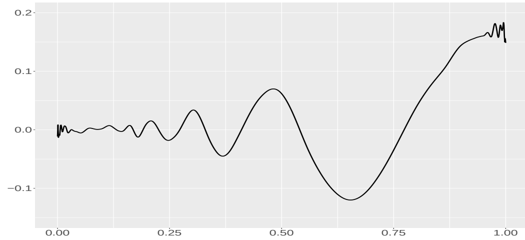
(c) Generated positions



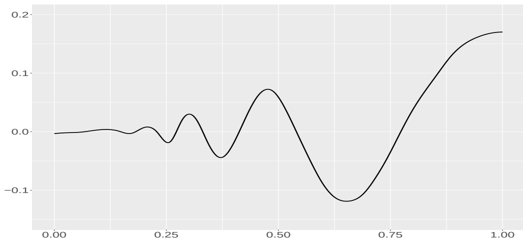
(d) Noisy position at *SNR*=7



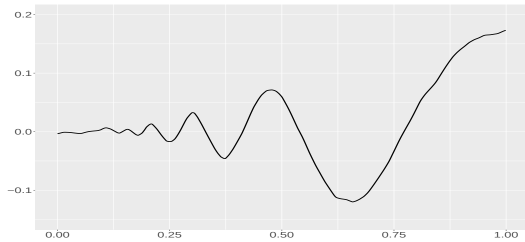
(e) Reconstruction from Wavelet by sure threshold



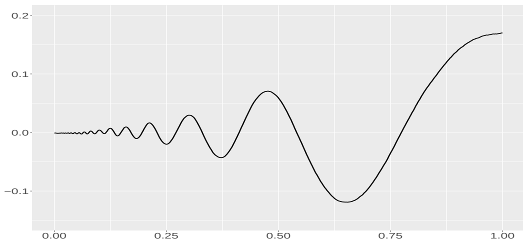
(f) Reconstruction from Wavelet by BayesThresh approach



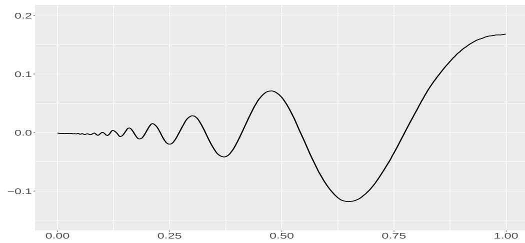
(g) Reconstruction by P-spline



(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

knots. *P-spline* gives a smoother fitting than wavelets, but the drawback is lack of 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 gets 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 values $\lambda(t) = \frac{(\Delta t)^3}{(\Delta d)^2}\lambda$ at SNR=7. The figures in the left column illustrate the values of the penalty term at different intervals, the figures in the right column are the observations and reconstructed trajectory. 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. Figure A.1 illustrates the reconstructions of Tractor spline at SNR=3.

Figure 2.7 demonstrates the estimated velocity functions. By taking the first derivative of fitted Tractor spine, it is simple to get the original four velocity functions. The fittings of velocity are not as smooth as that of position, because we only care about the smoothness of position rather than velocity in our cross-validation formula (2.28). However, velocity information does help us in reconstructing the trajectory.

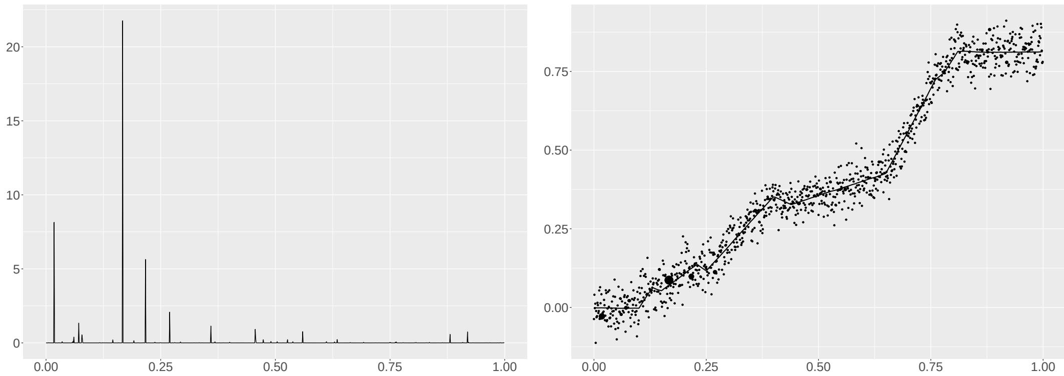
2.4.2 Evaluation

To examine the performance of the Tractor spline, we conduct a evaluation by comparing the mean squared errors and true mean squared errors, which are respectively calculated with the following formulas:

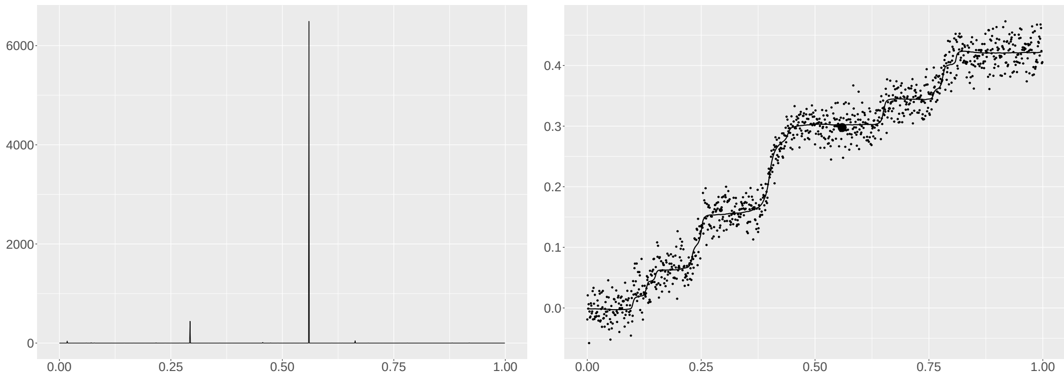
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}_{\lambda,\gamma}(t_i)\right)^2, \tag{2.37}$$

$$\mathrm{TMSE} = \frac{1}{n}\sum_{i=1}^{n}\left(f(t_i) - \hat{f}_{\lambda,\gamma}(t_i)\right)^2. \tag{2.38}$$

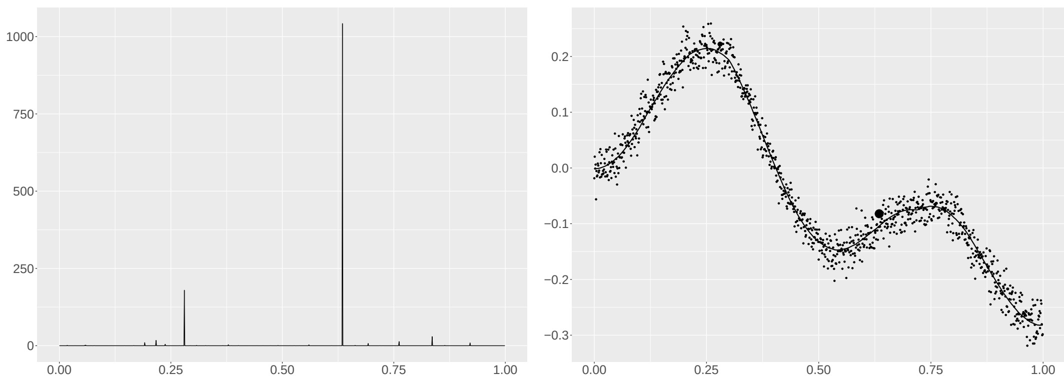
The results are shown in table 2.1 and 2.2. All of these methods have good performances in fitting noisy data. The differences of mean squared error between these methods are not significant, as can be seen from table 2.1. The proposed method is not the best among these simulations according to MSE. However, from table 2.2, Tractor spline returns the smallest true mean squared errors. The difference is significant, that means the reconstruction from Tractor spline is closer to the true trajectory.



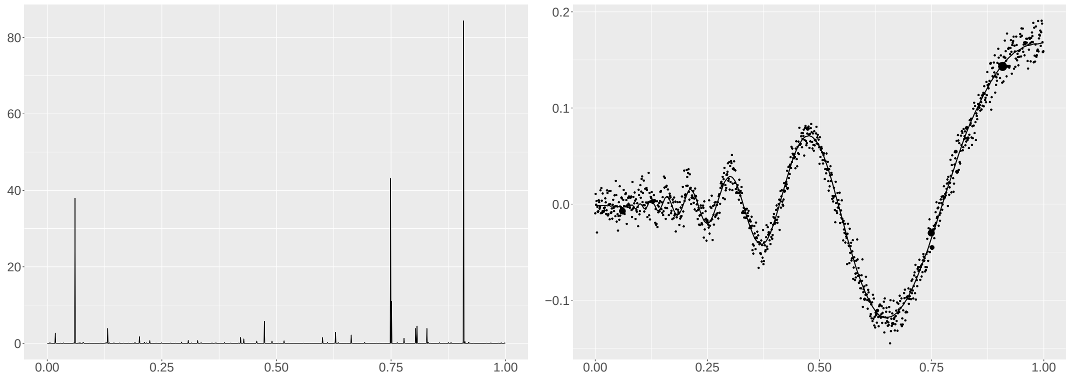
(a) Distribution of the penalty values in reconstructed *Blocks*



(b) Distribution of the penalty values in reconstructed *Bumps*

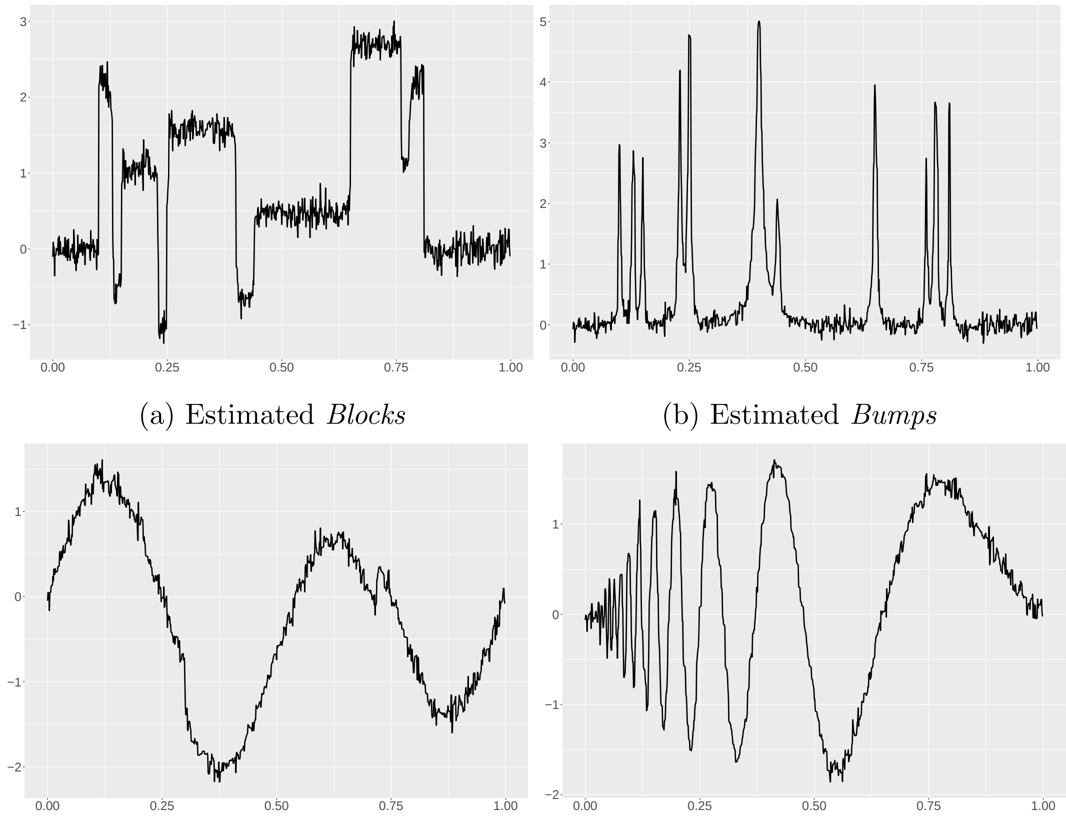


(c) Distribution of the penalty values in reconstructed *HeaviSine*



(d) Distribution of the penalty values in reconstructed *Doppler*

Figure 2.6: Distribution of the penalty values $\lambda(t)$ in Tractor spline. Figures on the left side indicate the values varying in intervals. On the right side, these values are 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 by Tractor spline. The velocity is generated from the original simulation functions by equation (2.33)

Table 2.1: MSE. Mean squared 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

Table 2.2: TMSE. True mean squared 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	$\text{TS}_{\gamma=0}$	$\text{TS}_{APT=0}$	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

Table 2.3: Retrieved SNR. Tractor spline effectively retrieves the SNR, which is calculated by $\sigma_{\hat{f}}/\sigma_{(\hat{f}-y)}$.

SNR	predefined value	from f	from Tractor spline $\hat{f}$
Blocks	7	6.9442	6.9485
Blocks	3	2.9761	2.9817
Bumps	7	6.9442	6.9548
Bumps	3	2.9761	2.9953
HeaviSine	7	6.9442	6.9207
HeaviSine	3	2.9761	2.9706
Doppler	7	6.9442	6.8757
Doppler	3	2.9761	2.9625

2.4.3 Residual Analysis

The simulated data is generated by equations (2.34) and the SNRs are set at 7 and 3 separately to compare the performances of different algorithms. All of the algorithms can reconstruct the true trajectory from noisy data and return acceptable MSE values, though Tractor spline returns the least TMSE in most of the circumstances.

Table 2.3 is comparing the capability of Tractor spline in retrieving the true SNR. The measurements are generated from f and g with predefined SNR. The Tractor spline reconstructs the true trajectory and retrieves the SNR value, both of which are close to the truth.

Further analysis in figures A.2 and A.3 shows that the residuals from Tractor splines are independent.

2.5 Application on Real Dataset

In this section, we apply the proposed Tractor spline to real dataset, which is recorded by a GPS unit mounted on a tractor. The original dataset contains the information about time marks, longitude, latitude, velocity, bearing (in degrees, heading to North) and boom status.

In a 2 or higher d-dimensional curve nonparametric regression, consider the general form of a length n time series data points $\{t_1, p_1, s_1\}, \ldots, \{t_n, p_n, s_n\}$, such that $a \leq t_1 < t_2 < \cdots < t_n \leq b$, p_i and s_i are d-dimensional vectors contain position and velocity information at time i respectively. The positive piecewise constant function $\lambda(t) = \lambda_i$

on each interval $t_i \leq t < t_{i+1}, t_0 = a, t_{n+1} = b$. Then the function $f : [a, b] \to \mathbb{R}^d$ with $\gamma > 0$ is a Tractor spline in the d-dimensional space if it is the solution to the generic form of the objective function:

$$J[f] = \frac{1}{n}\sum_{i=1}^{n}\|f(t_i) - p_i\|_d^2 + \frac{\gamma}{n}\sum_{i=1}^{n}\|f'(t_i) - s_i\|_d^2 + \sum_{i=0}^{n}\lambda_i \int_{t_i}^{t_{i+1}} \|f''(t)\|_d^2 dt. \tag{2.39}$$

Particularly, the GPS data is recorded in a 2-dimensional form, in which scenario $d = 2$. Hence, in the following application, we split the 2-dimensional 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. Without of generality, a dataset in a higher dimensional space can be projected into several sub-spaces, such as $p = \{x, y, z, \ldots\}$ and $s = \{u, v, w, \ldots\}$.

Thereafter, at first, we convert the longitude and latitude information from a 3D sphere to 2D surface by Universal Transverse Mercator coordinate system (UTM) and then project the velocity s into u and v on x-axis and y-axis respectively by

$$u = s \cdot \sin(\omega \frac{\pi}{180}), \tag{2.40}$$

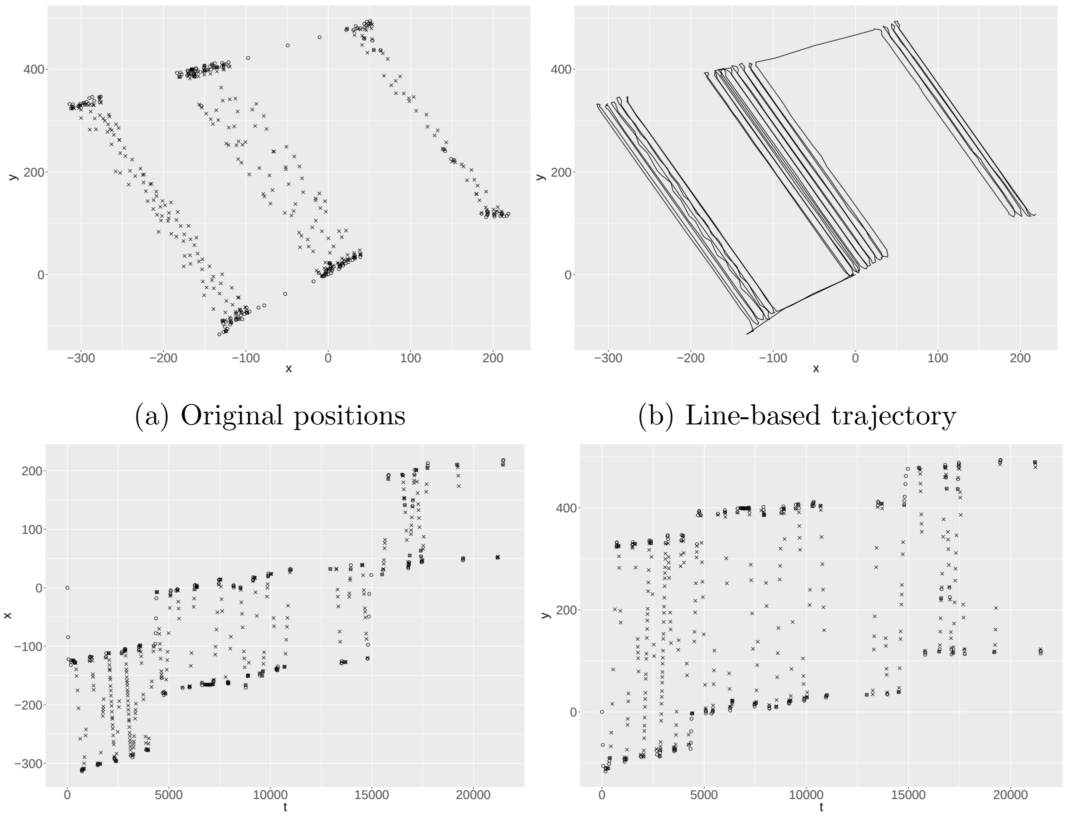
$$v = s \cdot \cos(\omega \frac{\pi}{180}), \tag{2.41}$$

where ω is the bearing in degrees. Boom status is tagged as 0 if it is not operating and 1 if it is. Time marks are transformed by subtracting the first mark, in which way the time starts from 0. Time duplicated data, caused by errors, had been removed from the dataset. In the 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 λ_d to our model. Then, we are now having three parameters λ_d and λ_u regarding boom status and γ controlling velocity residuals. The criteria of a good fitting are that it can catch more information, recognize time gaps between two points where tractor stops and return a smaller MSE.

2.5.1 1-Dimensional 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 (2.22) work in our model. All parameters in fitted Tractor spline are automatically selected by cross-validation by equation (2.28). Figure 2.9 and figure 2.10 compare the results of fitted



(a) Original positions

(b) Line-based trajectory

(c) Original positions on x-axis

(d) Original positions on y-axis

Figure 2.8: Original data points. Figure 2.8a is th original positions recorded by GPS units. Circle points means the boom is not operating; cross points means it is operating. Figure 2.8b is the line-based trajectory by simply connecting all points sequentially with straight lines. Figure 2.8c is the original x position. Figure 2.8d is the original y positions.

Table 2.4: Mean squared error. Tractor spline returns smallest errors among all these methods. P-spline was unable to reconstruct the y trajectory as the original dataset contains 0 Δ_y.

MSE	TS	$\text{TS}_{\gamma=0}$	$\text{TS}_{APT=0}$	P-spline	Wavelet(sure)	Wavelet(Bayes)
x	**0.2046**	0.2830	0.3298	2860.5480	256.0494	6.2959
y	**0.0020**	0.3062	0.3115	*NA*	1960.2220	19.3330

methods on x and y axes. 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 turn around. Tractor spline without adjusted penalty term presents less fitting at time gap knots, where time marks keep increasing while position stays the same and velocity is 0. If we take the last knot p_k before and the first knot p_{k+1} after the 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.

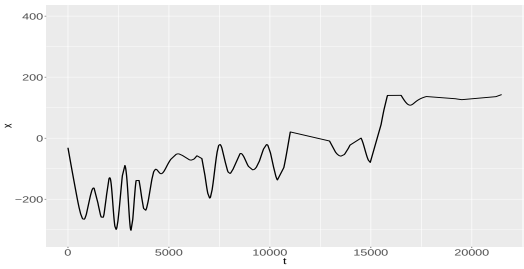
Table 2.4 illustrates the MSE of all methods on both x and y axes. The proposed Tractor spline returns the least errors among all methods.

The penalty function of the proposed Tractor spline is

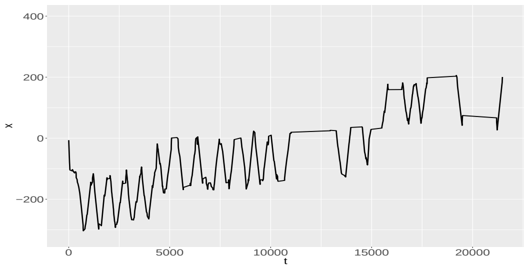
$$\lambda(t) = b\frac{\Delta t^3}{\Delta d^2}\lambda_d + (1-b)\frac{\Delta t^3}{\Delta d^2}\lambda_u, \text{ where } \begin{cases} b=1 & \text{if boom is operating} \\ b=0 & \text{if boom is not operating} \end{cases} \tag{2.42}$$

To tell the differences more clearly, we take $\lambda(t)$ in our demonstration. Figure 2.11 indicates that at turning points and long time gap knots, the adjusted penalty 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 2.11b clearly. Histogram plots of $\lambda(t)$ show that most of the penalty values are small, which allows the Tractor spline to go as closer as possible to the observed points. Only a few of penalty values are large, so that Tractor spline gives a straight line at tricky points.

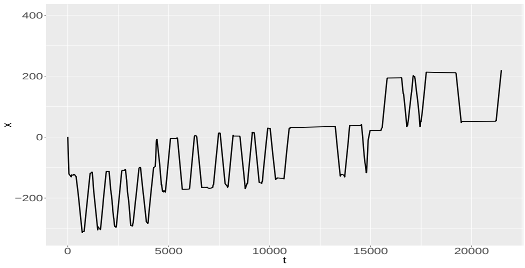
The 1-dimensional reconstruction gets the best fittings $\hat{f}_x$ and $\hat{f}_y$ on x and y axes separately using different penalty values, denoted as $\lambda_{d,x}$, $\lambda_{u,x}$, $\lambda_{d,y}$, $\lambda_{u,y}$, γ_x and γ_y. The final reconstruction is the combination of $\hat{f}_x$ and $\hat{f}_y$. It is shown in figure 2.12.



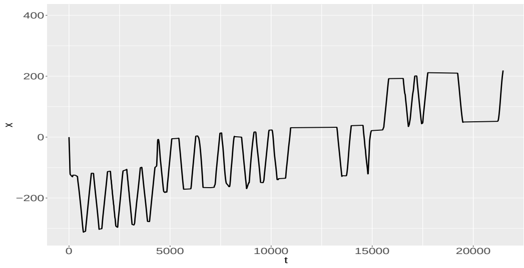
(a) Reconstruction by P-spline



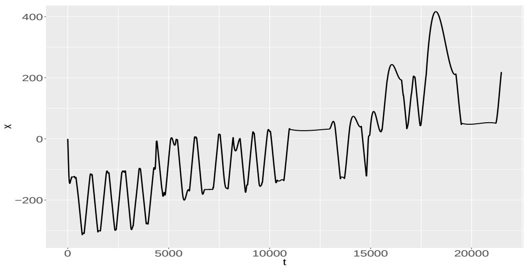
(b) Reconstruction by wavelet (*sure*)



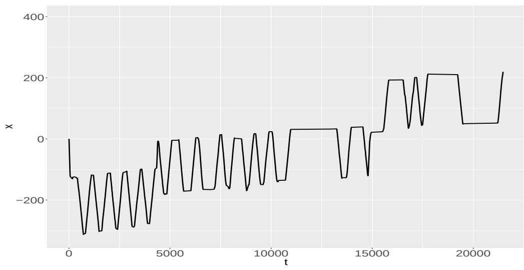
(c) Reconstruction by wavelet (*Bayes*)



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

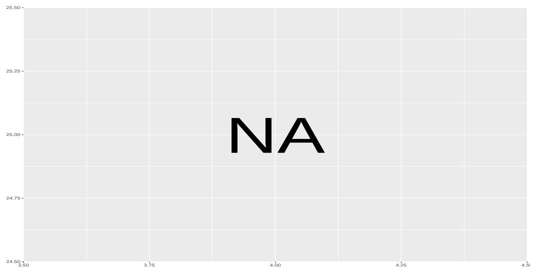


(e) Reconstruction by Tractor spline setting with conventional penalty term

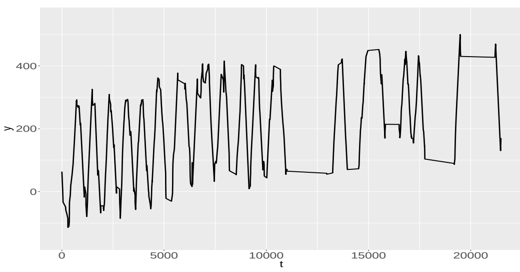


(f) Reconstruction by proposed Tractor spline

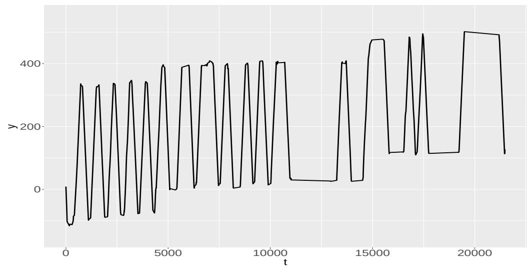
Figure 2.9: Fitted data points on x axis. Figure 2.9a Fitted by P-spline, which gives over-fitting on these points and misses some information. Figure 2.9b Fitted by wavelet (*sure*) algorithm. At some turning points, it gives over-fitting. Figure 2.9c Fitted by wavelet (*BayesThresh*) algorithm. It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



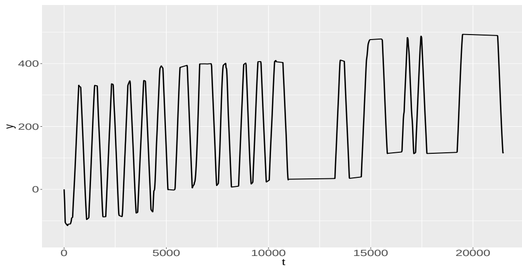
(a) Not available for P-spline



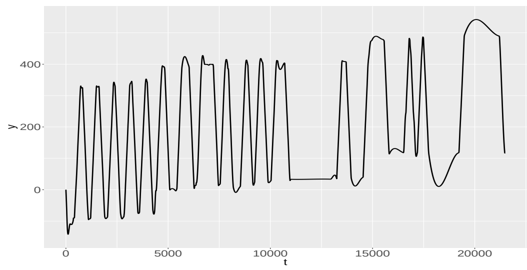
(b) Reconstruction by wavelet (*sure*)



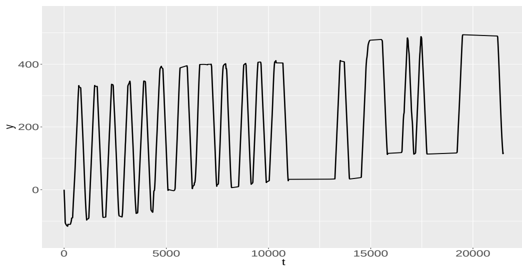
(c) Reconstruction by wavelet (*Bayes*)



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

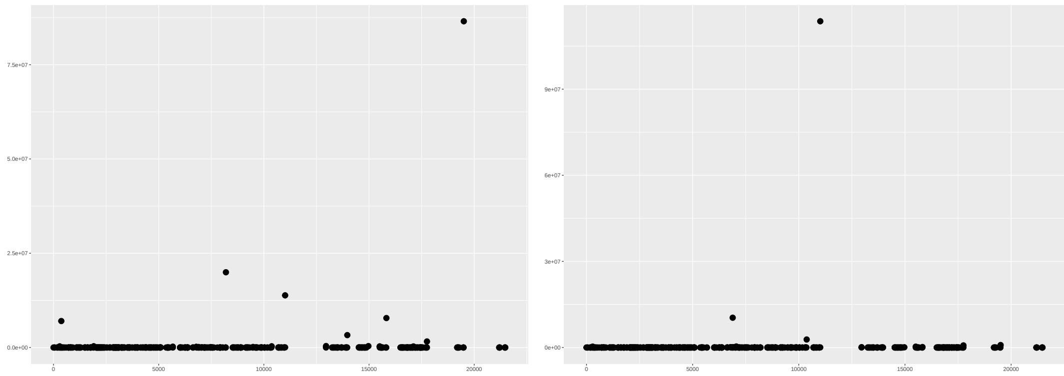


(e) Reconstruction by Tractor spline setting with conventional penalty term

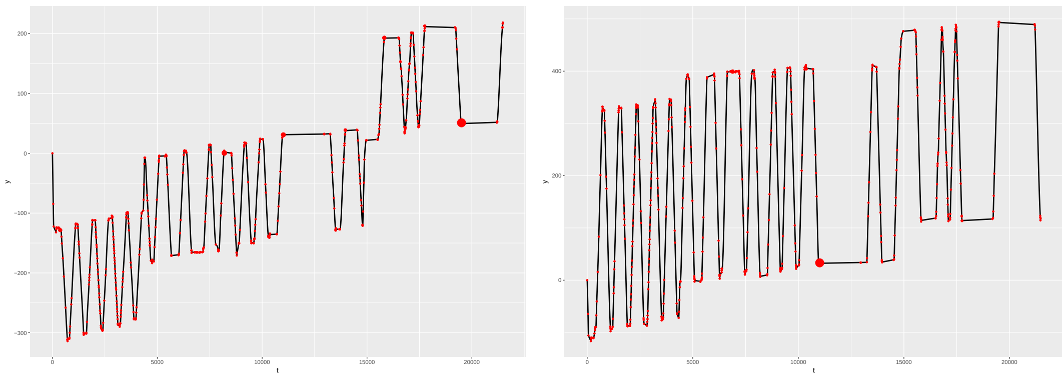


(f) Reconstruction by proposed Tractor spline

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



(a) Distribution of the penalty term on x and y



(b) Reconstruction on x and y

Figure 2.11: The penalty value $\lambda(t)$ of the Tractor spline on x and y axes. Red dots are the measurements $\mathbf{y}$. The bigger red dots in figure 2.11b indicate larger penalty values. It can be seen that most of large penalty values occur at turnings, where the tractor likely slows down and takes breaks.

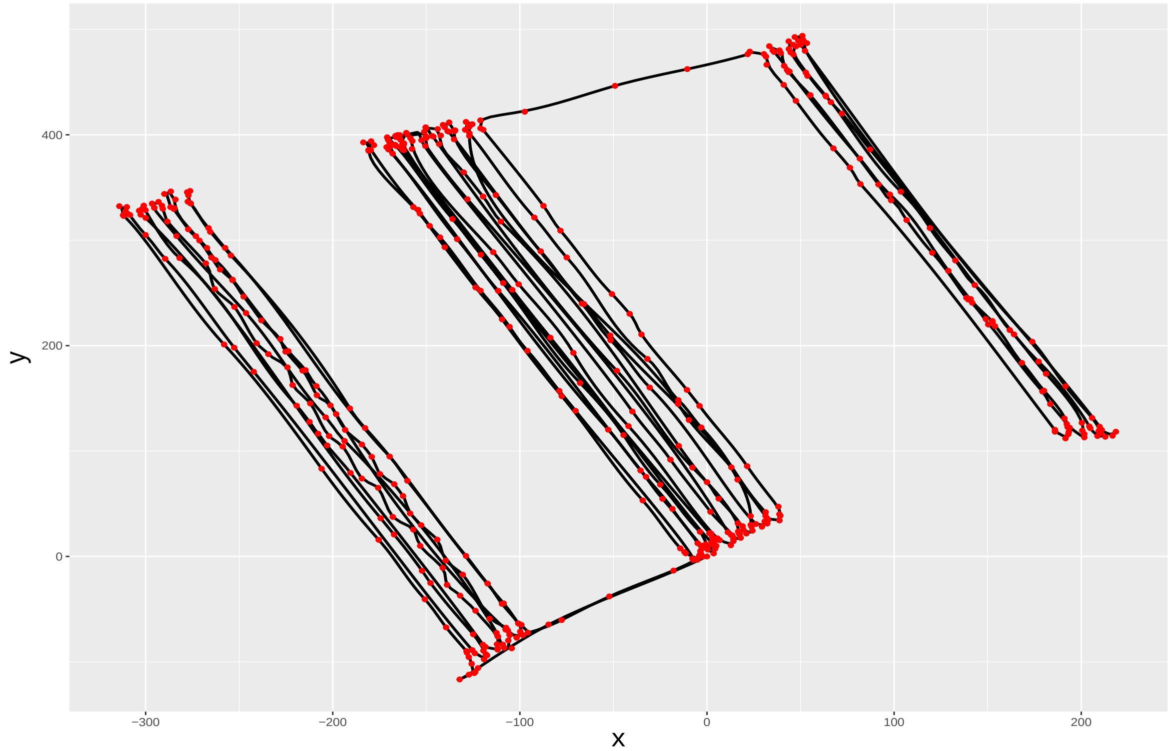


Figure 2.12: Combined reconstruction on x and y. Red dots are the measurements $\mathbf{y}$. The bigger size it is, the larger penalty value it indicates.

2.5.2 2-Dimensional Trajectory

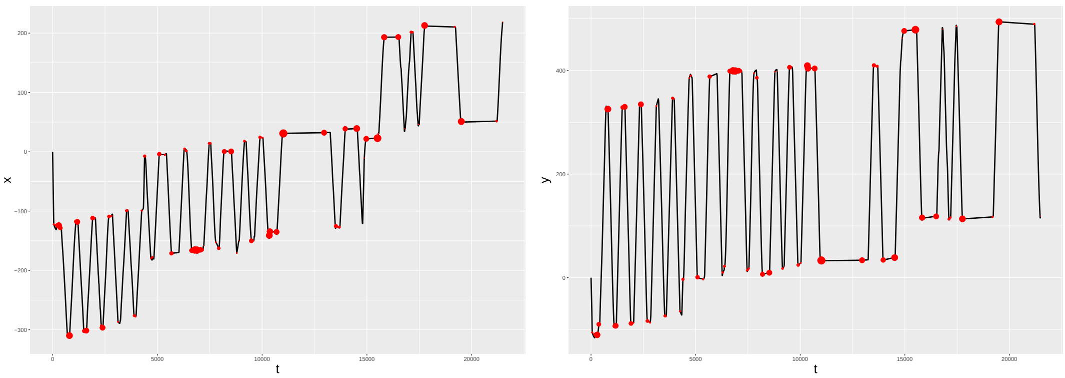
In a 2-dimensional trajectory reconstruction, different from combined 1-dimensional reconstruction, we are using the same parameters λ_d, λ_u and γ for both x and y axes. The overall best parameters return the least cross-validation score on all axes. Explicitly, it is calculated by the following formula

$$\mathrm{CV} = \mathrm{CV}_x + \mathrm{CV}_y. \tag{2.43}$$

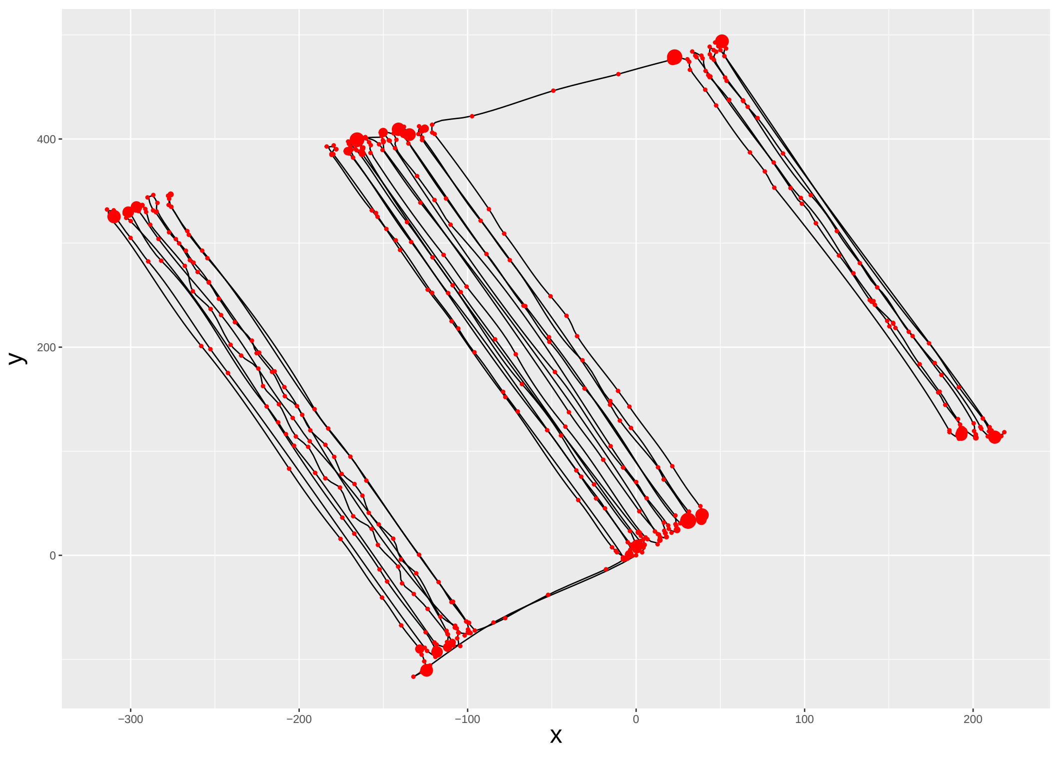
In the adjusted penalty term, Δd is the Euclidean distance $\Delta_d(p_1, p_2) = \sqrt{(\Delta x)^2 + (\Delta y)^2}$ between two positions on the 2D surface. Similarly in 1-dimensional reconstruction, the velocity information keeps trajectory in the right direction and the penalty term makes sure that the crazy curve will disappear between long-time-gap points. Figure 2.13 demonstrates the complete 2D reconstruction of the whole dataset.

The penalty function $\lambda(t)$ of a 2-dimensional reconstruction is shared by x and y axes and presented in figure 2.14. The complete penalty term is

$$n\theta_x^\top \Omega_{\lambda_d,\lambda_u} \theta_x + n\theta_y^\top \Omega_{\lambda_d,\lambda_u} \theta_y.$$



(a) 2-dimensional reconstruction on separate x and y



(b) 2-dimensional reconstruction

Figure 2.13: 2-dimensional reconstruction. Larger dots indicate bigger values of penalty function $\lambda(t)$.

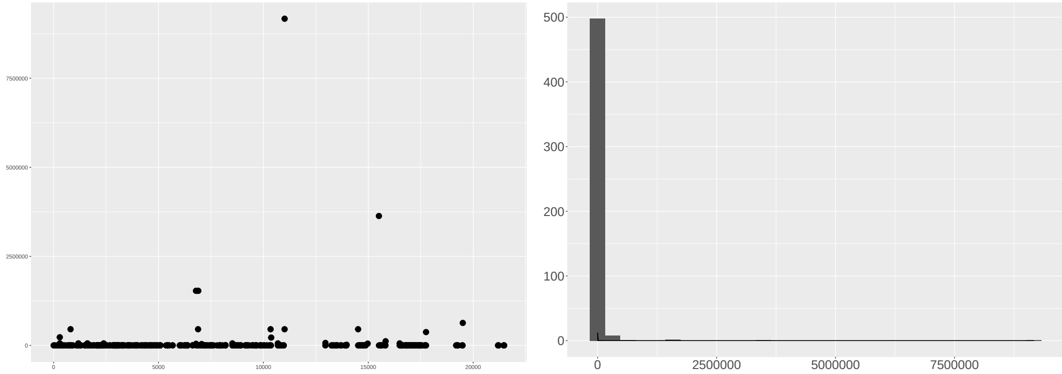


Figure 2.14: Penalty value of $\lambda(t)$ in 2-dimensional reconstruction.

Similarly, most of the large penalty values appear at 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.

The following figure 2.15 is a complete reconstruction from the whole observed dataset $\{x, u, y, v\}$. The overall reconstruction gives a smoothing path that goes through each measurement and avoids curvatures at turning points.

2.6 Conclusion and Discussion

In this chapter, I propose a Tractor spline model, which is the solution to the objective function consisting of both position and velocity information. The adjusted penalty function adapts to complicated curvatures. In a high d-dimensional space, Tractor spline can be projected into sub-spaces with respect to t and combined each solution together as a final. This method performs better when we know p and s 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 at each interior knots, the degrees of freedom will be $4 \times (n-1) - 2 \times (n-2) = 2n$. Because there are n position and n velocity points, thus we do not need to specify more parameters or add more constraints to the model.

Even though the mean squared errors of a Tractor spline is not the least comparing with other methods, the true mean squared errors are the least of among all the methods. That means the reconstruction is closer to the true path.

In parameter selection, the cross-validation only focuses on the errors of f ignoring that in f'. So the reconstruction of f' is not as smooth as that of f, which does not

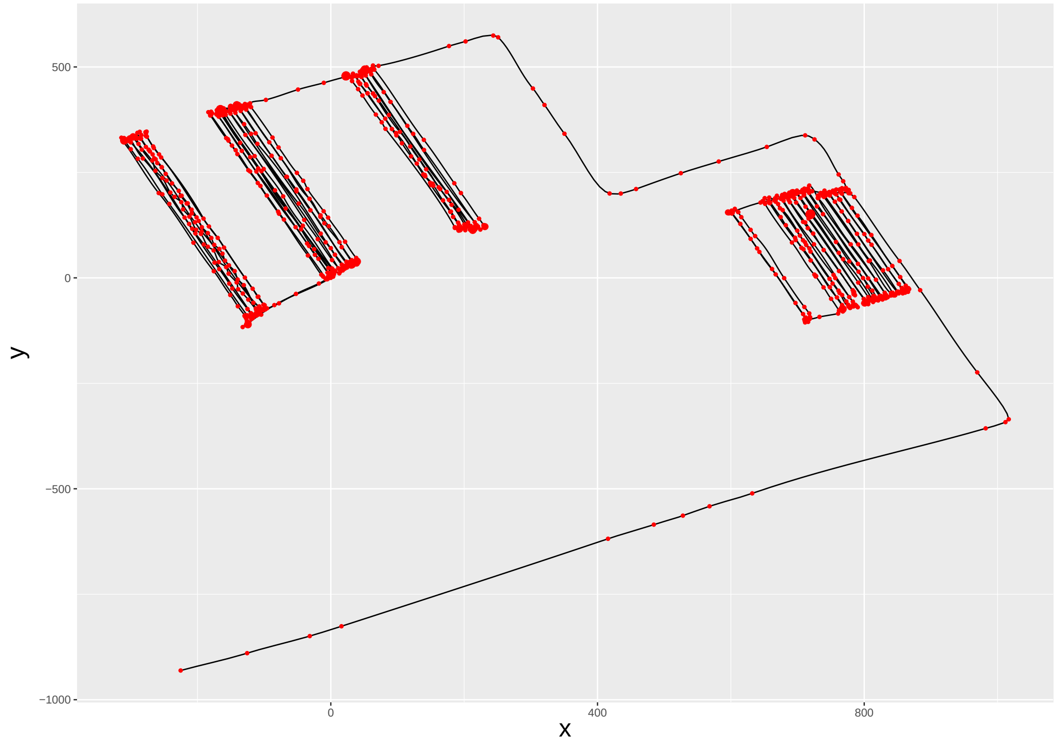


Figure 2.15: 2-dimensional reconstruction. Larger dots indicate bigger values of penalty function $\lambda(t)$.

affect trajectory reconstruction. A drawback of Tractor spline is that the computing time in finding local minimal CV score is higher than using B-spline. If there is an efficient way to compute matrix inverse, the calculation speed will be much faster. So in the simulation and application studies, we try to optimize our coding to make it run as faster as possible.

Another potential application of Tractor spline is on vessel monitoring system. This system is a fisheries surveillance that allows environmental and fisheries regulatory organization to track and monitor the activities of fishing vessels. The system calculates the unit's position and sends a data report to shore-side users. This information includes time and position in latitude and longitude. However, due to weak signals, the tracking system may lose some information. The Tractor spline can help to reconstruct the whole trajectory for a fishery vessel and use it to analyze its performance. For example, a larger penalty value indicates a stop on the sea inferring that the vessel is casting nets; a smaller penalty value indicates the vessel is moving normally.

After all, there is a wide range of applications for Tractor spline in real life. A future work is how to implement Tractor spline on-line to get instant estimations and how to make it run faster.

Chapter 3

Tractor Spline as Bayes Estimate

3.1 Introduction

A Hilbert space is a real or complex inner product space with respect to the distance function induced by the inner product Dieudonné (2013). Particularly, the Hilbert space $\mathcal{L}_2[0,1]$ is a set of square integrable functions $f(t):[0,1]\to\mathbb{R}$, where all functions satisfy

$$\mathcal{L}_2[0,1]=\{f:\int_0^1 f^2 dt<\infty\}$$

with an inner product $\langle f,g\rangle=\int_0^1 fg dt$.

Consider a regression problem with observations $y_i=f(t_i)+\varepsilon_i,\ i=1,\dots,n$, consisting i.i.d.normal noises $\varepsilon_i\sim N(0,\sigma^2)$ in the space $C^{(m)}[0,1]=\{f:f^{(m)}\in L_2[0,1]\}$. The classic nonparametric or semi-parametric regression is a function that minimizes the following penalized least square functional

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i-f(t_i)\right)^2+\lambda\int_0^1\left(f^{(m)}\right)^2 dt, \tag{3.1}$$

where the first term is the lack of fit of f to the data. In the second term, generally, λ is a fixed smoothing parameter controlling the trade-off between over-fitting and bias Trevor Hastie (2009). The minimizer f_λ of the above equation resides in an n-dimensional space and the computation in multivariate settings is generally of the order $\mathrm{O}\left(n^3\right)$ Kim and Gu (2004). It is shown that a piecewise polynomial smoothing spline of degree $2m-1$ is well known to provide an aesthetically satisfying method for estimating f if $\mathbf{y}$ cannot be interpolated exactly by some polynomial of degree less than m Schoenberg (1964). For instance, when $m=2$, a piecewise cubic smoothing spline provides a powerful tool to estimate the above nonparametric function, in which the penalty term is $\int f''^2 dt$ Hastie and Tibshirani (1990).

Further, Wahba (1978) showed that a Bayesian version of this problem is to take a Gaussian process prior $f(t_i) = a_0 + a_1 t_i + \cdots + a_{m-1} t_i^{m-1} + x_i$ on f with $x_i = X(t_i)$ being a zero-mean Gaussian process whose mth derivative is scaled white noise, $i = 1, \ldots, n$ Speckman and Sun (2003). The extended Bayes estimates f_λ with a "partially diffuse" prior is as exactly the same as spline solution. Some works have been done on discovering the relationship between nonparametric regression and Bayesian estimation. Heckman and Woodroofe (1991) shows that if f the regression function $\mathrm{E}(y \mid f)$ has unknown prior distribution $\mathbf{f} = (f(t_1), \ldots, f(t_n))^\top$ lying in a known class of Ω, then the maximum is taken over all priors in Ω and the minimum is taken over linear estimator of $\mathbf{f}$. Branson *et al.* (2017) propose a Gaussian process regression method that acts as a Bayesian analog to local linear regression for sharp regression discontinuity designs. It is no doubt that one of the attractive features of the Bayesian approach is that, in principle, one can solve virtually any statistical decision or inference problem. Particularly, one can provide an accuracy assessment for $\hat{f} = \mathrm{E}(f \mid \mathbf{y})$ using posterior probability regions Cox (1993).

Based on the correspondence, Craven and Wahba (1978) proposed an generalized cross-validation estimate for the minimizer f_λ. The estimate $\hat{\lambda}$ is the minimizer of the function where the trace of matrix $A(\lambda)$ in (2.3) is incorporated. It is also able to establish an optimal convergence property for their estimator when the number of observations in a fixed interval tends to infinity Wecker and Ansley (1983). A high efficient algorithm to optimize generalized cross-validation and generalized maximum likelihood scores with multiple smoothing parameters via the Newton method was proposed by Gu and Wahba (1991). This algorithm can also be applied to the maximum likelihood and the restricted maximum likelihood estimation. The behavior of the optimal regularization parameter in the method of regularization was investigated by Wahba and Wang (1990).

In this chapter, it is proved that the Tractor spline can be estimated by a Bayesian approach in second-derivative-piecewise-continuous Reproducing Kernel Hilbert Space. The GCV is used to find the optimal parameters for Tractor spline.

3.2 Polynomial Smoothing Splines on $[0, 1]$ as Bayes Estimates

A polynomial smoothing spline of degree $2m - 1$ is a piecewise polynomial of the same degree on each intervals $[t_i, t_{i+1}]$, $i = 1, \ldots, n-1$, and the first $2m-2$ derivatives are continuous at joint knots. For instance, when $m = 2$, a piecewise cubic smoothing spline is a special case of the polynomial smoothing spline providing a powerful tool to estimate the above nonparametric function (3.1) in the space $\mathcal{C}^{(2)}[0, 1]$, where the penalty term is $\int f''^2 dt$ Hastie and Tibshirani (1990) Wang (1998). If a general space $\mathcal{C}^{(m)}[0, 1]$ is equipped with an appropriate inner product, it can be made a reproducing kernel Hilbert space.

3.2.1 Polynomial Smoothing Spline

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 $(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)$ in the 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 $(t_1, t_2), \ldots, (t_{n-1}, t_n)$ and represented by a separate polynomial on each interval. For any continuous $f \in \mathcal{C}^{(m)}[0, 1]$, it can be represented in a linear combination of basis functions $h_m(t)$ as $f(t) = \sum_{m=1}^{M} \beta_m h_m(t)$, where β_m are coefficients Ellis *et al.* (2009). It is just like every vector in a vector space can be represented as a linear combination of basis vectors.

A smoothing polynomial spline is uniquely the smoothest function that achieves a given degree of fidelity to a particular data set Whittaker (1922). In deed, the minimizer of function (3.1) is the curve estimate $\hat{f}(t)$ over all spline functions $f(t)$ with $m-1$ continuous derivatives fitting observed data in the space $\mathcal{C}^{(m)}[0, 1]$. In fact, Kimeldorf and Wahba (1971), Kimeldorf and Wahba (1970) prove that the minimizer f_λ of function (3.1) has the form

$$f(t) = \sum_{\nu=1}^{m} d_\nu \phi_\nu(t) + \sum_{i=1}^{n} c_i R_1(t, t_i). \tag{3.2}$$

where $\{\phi_\nu(t)\}$ is a set of basis functions of space $\mathcal{H}_0$ and $R(\cdot, \cdot)$ is the reproducing kernel in $\mathcal{H}_1$.

3.2.2 Reproducing Kernel Hilbert Space $\mathcal{H}^{(m)}[0,1]$

For any $f \in \mathcal{C}^{(m)}[0,1]$, its standard Taylor expansion is

$$f(t) = \sum_{\nu=0}^{m-1} \frac{t^\nu}{\nu!} f^{(\nu)}(0) + \int_0^1 \frac{(t-u)_+^{m-1}}{(m-1)!} f^{(m)}(t)dt, \tag{3.3}$$

where $(\cdot)_+ = \max(0, \cdot)$. With an inner product

$$\langle f, g \rangle = \sum_{\nu=0}^{m-1} f^{(\nu)}(0) g^{(\nu)}(0) + \int_0^1 f^{(m)} g^{(m)} dt, \tag{3.4}$$

the representer is

$$R_s(t) = \sum_{\nu=0}^{m-1} \frac{s^\nu}{\nu!} \frac{t^\nu}{\nu!} + \int_0^1 \frac{(s-u)_+^{m-1}}{(m-1)!} \frac{(t-u)_+^{m-1}}{(m-1)!} du \triangleq R_0(s,t) + R_1(s,t). \tag{3.5}$$

It is easily to prove that $R(s,t)$ is non-negative and is reproducing kernel, by which $\langle R(s,t), f(t) \rangle = \langle R_s(t), f(t) \rangle = f(s)$. Additionally, $R_s^{(\nu)}(0) = s^\nu/\nu!$ for $\nu = 0, \ldots, m-1$.

Before moving on to further steps, we are now introducing the following two theorems.

Theorem 4. *Aronszajn (1950) Suppose R is a symmetric, positive definite kernel on a set X. Then, there is a unique Hilbert space of functions on X for which R is a reproducing kernel.*

Theorem 5. *Gu (2013) 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, $R_0(x,\cdot), R1(x,\cdot) \in \mathcal{H}$, for $\forall x \in X$, and $\langle R_0(x,\cdot), R_1(y,\cdot) \rangle = 0$, for $\forall x, y \in X$, 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 nonnegative definite and $\mathcal{H}_0 \cap \mathcal{H}_1 = \{0\}$, then $\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{H}_1$ has a reproducing kernel $R = R_0 + R_1$.*

According to theorem 4, the Hilbert space associated with R can be constructed as containing all finite linear combinations of the form $\sum a_i R(t_i, \cdot)$, and their limits under the norm induced by the inner product $\langle R(s,\cdot), R(t,\cdot) \rangle = R(s,t)$. As for theorem 5, it is easy to verify that R_0 corresponds to the space of polynomials $\mathcal{H}_0 = \{f : f^{(m)} = 0\}$ with an inner product $\langle f, g \rangle_0 = \sum_{\nu=0}^{m-1} f^{(\nu)}(0) g^{(\nu)}(0)$ and R_1 corresponds to the orthogonal complement of $\mathcal{H}_0$, that is $\mathcal{H}_1 = \{f : f^{(\nu)}(0) = 0, \nu = 0, \ldots, m-1, \int_0^1 (f^{(m)})^2 dt < \infty\}$ with an inner product $\langle f, g \rangle_1 = \int_0^1 f^{(m)} g^{(m)} dt$.

3.2.3 Polynomial Smoothing Spline as Bayes Estimates

Because it is possible to interpret the smoothing spline regression estimator as a Bayes estimate when the mean function $r(\cdot)$ is given an improper prior distribution Wahba (1990) Berlinet and Thomas-Agnan (2011), therefore, we will find that the posterior mean of $f = f_0 + f_1$ on $[0, 1]$ with proper priors is the polynomial smoothing spline of (3.1).

Indeed, it can be found that for each $f = f_0 + f_1$ on $[0, 1]$ with f_0 and f_1, it has independent Gaussian priors with zero means and covariances, which have the following property that

$$
\begin{aligned}
\mathrm{E}(f_0(s)f_0(t)) &= \tau^2 R_0(s,t) = \tau^2 \sum_{\nu=0}^{m-1} \frac{s^\nu}{\nu!}\frac{t^\nu}{\nu!},\\
\mathrm{E}(f_1(s)f_1(t)) &= bR_1(s,t) = b\int_0^1 \frac{(s-u)_+^{m-1}}{(m-1)!}\frac{(t-u)_+^{m-1}}{(m-1)!},
\end{aligned}
$$

where R_0 and R_1 are from (3.5). Because of the observations are normally distributed as $y_i \sim N(f(t_i), \sigma^2)$, then the joint distribution for $\mathbf{y} = \{y_1, \ldots, y_n\}$ and $f(t)$ is normal with zero mean and the following covariance matrix

$$
\mathrm{Cov}(\mathbf{f}, \mathbf{y}) = \begin{bmatrix} bQ + \tau SS^\top + \sigma^2 I & b\xi + \tau^2 S\phi \\ b\xi^\top + \tau^2\phi^\top S^\top & bR_1(t,t) + \tau^2\phi^\top\phi \end{bmatrix},
$$

where $\{Q_{i,j}\}_{n\times n} = R_1(t_i, t_j)$, $\{S_{i,\nu}\}_{n\times m} = t_i^{\nu-1}/(\nu-1)!$, $\{\xi_{i,1}\}_{n\times 1} = R_1(t_i, t)$ and $\{\phi_{\nu,1}\}_{m\times 1} = t^{\nu-1}/(\nu-1)!$. Consequently, the posterior is

$$
\begin{aligned}
\mathrm{E}(f(t) \mid \mathbf{y}) &= (b\xi^\top + \tau\phi^\top s^\top)(bQ + \tau^2 SS^\top + \sigma^2 I)^{-1}\mathbf{y} \\
&= \xi^\top(Q + \rho SS^\top + n\lambda I)^{-1}\mathbf{y} + \phi^\top\rho S^\top(Q + \rho SS^\top + n\lambda I)^{-1}\mathbf{y},
\end{aligned} \tag{3.6}
$$

where $\rho = \tau^2/b$ and $n\lambda = \sigma^2/b$. Furthermore, by denoting $M = Q + n\lambda I$, Gu (2013) gives that, when $\rho \to \infty$, the posterior mean is in the form $\mathrm{E}\left(f(t) \mid y_{1:n}\right) = \xi^\top\mathbf{c} + \phi^\top\mathbf{d}$ with coefficients

$$
\mathbf{c} = (M^{-1} - M^{-1}S(S^\top M^{-1}S)^{-1}S^\top M^{-1})\mathbf{y}, \tag{3.7}
$$

$$
\mathbf{d} = (S^\top M^{-1}S)^{-1}S^\top M^{-1}\mathbf{y}. \tag{3.8}
$$

Theorem 6. *Gu (2013) The polynomial smoothing spline of (3.1) is the posterior mean of $f = f_0 + f_1$, where f_0 diffuses in span $\{t^{\nu-1}, \nu = 1, \ldots, m\}$ and f_1 has a Gaussian process prior with mean zero and a covariance function*

$$
bR_1(s,t) = b\int_0^1 \frac{(s-u)_+^{m-1}}{(m-1)!}\frac{(t-u)_+^{m-1}}{(m-1)!},
$$

for $b = \sigma^2/n\lambda$.

3.2.4 Gaussian Process Regression

Gaussian processes are the extension of multivariate Gaussian to infinite-sized collections of real value variables, any finite number of which have a joint Gaussian distribution Rasmussen and Williams (2006). Gaussian process regression is a probability distribution over functions. It is fully defined by its mean $m(t)$ and covariance $K(s,t)$ function as

$$\begin{aligned} m(t) &= \mathrm{E}[f(t)] \\ K(s,t) &= \mathrm{E}[(f(s)-m(s))(f(t)-m(t))], \end{aligned}$$

where s and t are two variables. A function f distributed as such is denoted in form of

$$f \sim GP(m(t), K(s,t)).$$

Usually the mean function is assumed to be zero everywhere.

Given a set of input variables $\mathbf{t} = \{t_1, \ldots, t_n\}$ for function $f(t)$ and the output $\mathbf{y} = f(\mathbf{t}) + \varepsilon$ with i.i.d.Gaussian noise ε of variance σ_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

$$\mathrm{Cov}(y_p, y_q) = K(t_p, t_q) + \sigma_n^2 \delta_{pq}$$

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

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(\mathbf{t},\mathbf{t}) + \sigma_n^2 I & K(\mathbf{t}, t_*) \\ K(t_*, \mathbf{t}) & K(t_*, t_*) \end{bmatrix}\right). \tag{3.9}$$

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

$$f_* \mid \mathbf{y}, \mathbf{t}, t_* \sim N(\bar{f}_*, \mathrm{Cov}(f_*))$$

where

$$\bar{f}_* = \mathrm{E}[f_* \mid \mathbf{y}, \mathbf{t}, t_*] = K(t_*, \mathbf{t})[K(\mathbf{t},\mathbf{t}) + \sigma_n^2 I]^{-1}\mathbf{y}, \tag{3.10}$$

$$\mathrm{Cov}(f_*) = K(t_*, t_*) - K(t_*, \mathbf{t})[K(\mathbf{t},\mathbf{t}) + \sigma_n^2 I]^{-1} K(\mathbf{t}, t_*). \tag{3.11}$$

Therefore, it can seen that the Bayesian estimation of a smoothing spline is a special format of Gaussian process regression with diffuse prior and the covariance matrix $R(s,t)$.

3.3 Tractor Spline as Bayes Estimate

Recall the definition of Tractor spline introduced in Chapter 2.2. It is the solution to the objective function (2.5), where an extra term for $f'(t) - v$ and an extra parameter γ are incorporated. The penalty parameter $\lambda(t)$ is a function varying on different domains. If $\lambda(t) = \lambda$ is constant and $\gamma = 0$, the Tractor spline degenerate to a conventional cubic smoothing spline consisting of a set of given basis functions.

However, the Bayes estimate for a polynomial smoothing spline requires fixed interval on $[0, 1]$ and the penalty parameter is constant. For the first constraint, without loss of generality, an arbitrary interval $[a, b]$ can be transformed to $[0, 1]$. For the second constraint, it is assumed that $\lambda(t)$ stays the same constant in each subinterval of $[0, 1]$ and name the solution "trivial Tractor spline". In this section, I still call it the "Tractor spline" for sake of simplicity.

In the following, we are going to prove that this kind of trivial Tractor spline is corresponding to Bayes estimate in a particular reproducing kernel Hilbert space.

3.3.1 Reproducing Kernel Hilbert Space $\mathcal{C}^2_{p.w.}[0, 1]$

The space $C^{(m)}[0, 1] = \{f : f^{(m)} \in L_2[0, 1]\}$ is a set of functions f whose mth derivatives are square integrable on the domain $[0, 1]$. For a tractor spline, it only requires $m = 2$. In fact, its second derivative is piecewise linear but is not necessarily continuous at joint knots. Besides, only if $\lambda(t)$ is constant and $\gamma = 0$, the second derivative is piecewise linear and continuous at joint knots. Here we are introducing the space

$$\mathcal{C}^2_{p.w.}[0, 1] = \{f : f'' \in L_2[0, 1], f, f' \text{ are continuous and } f'' \text{ is piecewise linear}\},$$

in which the second derivative of any function f is not necessarily continuous.

Given a sequence of paired data $\{(t_1, y_1, v_1), \ldots, (t_n, y_n, v_n)\}$, the the minimizer of

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(t_i))^2 + \frac{\gamma}{n}\sum_{i=1}^{n}(v_i - f'(t_i))^2 + \lambda\int_0^1 f''^2 dt \tag{3.12}$$

in the space $\mathcal{C}^2_{p.w.}[0, 1]$ is a Tractor spline. Equipped with an appropriate inner product

$$\langle f, g\rangle = f(0)g(0) + f'(0)g'(0) + \int_0^1 f''g''dx, \tag{3.13}$$

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

$$R_s(t) = 1 + st + \int_0^1 (s-u)_+(t-u)_+du. \tag{3.14}$$

It can be seen that $R_s(0) = 1, R'_s(0) = s$, and $R''_s(t) = (s-t)_+$. The two terms of the reproducing kernel $R(s,t) = R_s(t) \triangleq R_0(s,t) + R_1(s,t)$, where

$$R_0(s,t) = 1 + st$$
$$R_1(s,t) = \int_0^1 (s-u)_+(t-u)_+ du$$

are both non-negative definite themselves.

According to Theorem 5, R_0 can correspond the space of polynomials $\mathcal{H}_0 = \{f : f'' = 0\}$ with an inner product $\langle f, g\rangle_0 = f(0)g(0) + f'(0)g'(0)$, and R_1 corresponds the orthogonal complement of $\mathcal{H}_0$

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

with inner product $\langle f, g\rangle_1 = \int_0^1 f''g''dt$. 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 $R_s(\cdot) = R_0(s,\cdot) + R_1(s,\cdot)$.

Define a new notation $\dot{R}(s,t) = \frac{\partial R}{\partial s}(s,t) = \frac{\partial R_0}{\partial s}(s,t) + \frac{\partial R_1}{\partial s}(s,t) = t + \int_0^s (t-u)_+ du$. Obviously $\dot{R}_s(t) \in \mathcal{C}^2_{p.w.}[0,1]$. Additionally, we have $\dot{R}_s(0) = 0, \dot{R}'_s(0) = \frac{\partial \dot{R}_s}{\partial t}(0) = 1$, and $\dot{R}''_s(t) = \begin{cases} 0 & s \le t \\ 1 & s > t \end{cases}$. Then, for any $f \in \mathcal{C}^2_{p.w.}[0,1]$, we have

$$\langle \dot{R}_s, f\rangle = \dot{R}_s(0)f(0) + \dot{R}'_s(0)f'(0) + \int_0^1 \dot{R}''_s f'' du = f'(0) + \int_0^t f'' du = f'(t).$$

It can be seen that the first term $\dot{R}_0 = t \in \mathcal{H}_0$, and the space spanned by the second term $\dot{R}_1 = \int_0^s (t-u)_+ du$, denoted as $\dot{\mathcal{H}}$, is a subspace of $\mathcal{H}_1$, and $\dot{\mathcal{H}} \ominus \mathcal{H}_1 \neq \emptyset$. Given the sample points $t_j, j = 1, \ldots, n$, in equation (3.12) and noting that the space

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

is a closed linear subspace of $\mathcal{H}_1$. Then, we have a new space $\mathcal{H}_* = \dot{\mathcal{H}} \cup \mathcal{A}$. Thus, the two new sub spaces in $\mathcal{C}^2_{p.w.}[0,1]$ are $\mathcal{H}_0$ and $\mathcal{H}_*$.

For any $f \in \mathcal{C}^2_{p.w.}[0,1]$, it can be written as

$$f(t) = d_1 + d_2 t + \sum_{j=1}^n c_j R_1(t_j, t) + \sum_{i=j}^n b_j \dot{R}_1(t_j, \cdot) + \rho(t) \tag{3.15}$$

where $\mathbf{d}, \mathbf{c}$ and $\mathbf{b}$ are coefficients, and $\rho(t) \in \mathcal{H}_1 \ominus \mathcal{H}_*$. Thus, by substituting to the equation (3.12), it can be written as

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\left(y_i - d_1 - d_2 t - \sum_{j=1}^{n} c_j R_1(t_j,t_i) - \sum_{j=1}^{n} b_j \dot{R}_1(t_j,t_i) - \rho(t_i)\right)^2 \\
&\frac{\gamma}{n}\sum_{i=1}^{n}\left(v_i - d_2 - \sum_{j=1}^{n} c_j R_1'(t_j,t_i) - \sum_{j=1}^{n} b_j \dot{R}_1'(t_j,t_i) - \rho'(t_i)\right)^2 \\
&+\lambda\int_0^1\left(\sum_{j=1}^{n} c_j R_1''(t_j,t) + \sum_{j=1}^{n} b_j \dot{R}_1''(t_j,t) + \rho''(t)\right)^2 dt
\end{aligned} \tag{3.16}$$

Because of orthogonality, $\rho(t_i) = (R_1(t_i,\cdot),\rho) = 0$, $\rho'(t_i) = (\dot{R}_1(t_i,\cdot),\rho') = 0$, $i = 1,\dots,n$. By denoting that

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

and noting that $\int_0^1 R_1''(t_i,t)R_1''(t_j,t)dt = R_1(t_i,t_j)$, $\int_0^1 R_1''(t_i,t)\dot{R}_1''(t_j,t)dt = \int_0^v (t_i - t)dt = \dot{R}_1(t_j,t_i)$, and $\int_0^1 \dot{R}_1''(t_i,t)\dot{R}_1''(t_j,t)dt = \int_0^v 1dt = \dot{R}_1'(t_i,t_j)$, where $v = \min(t_i,t_j)$, the above equation (3.16) can be written as

$$\begin{aligned}
&(\mathbf{y} - S\mathbf{d} - Q\mathbf{c} - P\mathbf{b})^\top(\mathbf{y} - S\mathbf{d} - Q\mathbf{c} - P\mathbf{b}) + \\
&\gamma(\mathbf{v} - S'\mathbf{d} - Q'\mathbf{c} - P'\mathbf{b})^\top(\mathbf{v} - S'\mathbf{d} - Q'\mathbf{c} - P'\mathbf{b}) \\
&+n\lambda(\mathbf{c}^\top Q\mathbf{c} + 2\mathbf{c}^\top P\mathbf{b} + \mathbf{b}^\top P'\mathbf{b}) + n\lambda(\rho,\rho).
\end{aligned} \tag{3.17}$$

Note that ρ only appears in the third term and is minimized at $\rho = 0$. Hence, a tractor spline resides in the space $\mathcal{H}_0 \oplus \mathcal{H}_*$ of finite dimension. Thus, the solution to (3.12) is computed via the minimization of the first three terms in (3.17) with respect to $\mathbf{d}$, $\mathbf{c}$ and $\mathbf{b}$.

3.3.2 Posterior of Bayes Estimates

Now in the model $y = f(t) + \varepsilon_1$, $\varepsilon_1 \sim N(0,\sigma^2)$, and $v = f'(t) + \varepsilon_2$, $\varepsilon_2 \sim N(0,\frac{\sigma^2}{\gamma})$, according to equation (3.15), for $f(t) \in \mathcal{C}^2_{p.w.}[0,1]$, we have

$$f(t) = (d_1 + d_2 t) + \sum_{i=1}^{n} c_i R_1(t_i,t) + \sum_{i=1}^{n} b_i \dot{R}_1(t_i,t). \tag{3.18}$$

The covariance functions for y, v and f, f' are

$$
\begin{aligned}
\mathrm{E}(f(s)f(t)) &= \tau^2 R_0(s,t) + \beta R_1(s,t) & \mathrm{E}(f(s)f'(t)) &= \tau^2 R_0'(s,t) + \beta R_1'(s,t)\\
\mathrm{E}(f'(s)f(t)) &= \tau^2 \dot{R}_0(s,t) + \beta \dot{R}_1(s,t) & \mathrm{E}(f'(s)f'(t)) &= \tau^2 \dot{R}_0'(s,t) + \beta \dot{R}_1'(s,t)\\
\mathrm{E}(y_i, y_j) &= \tau^2 R_0(s_i,s_j) + \beta R_1(s_i,s_j) + \sigma^2\delta_{ij} & \mathrm{E}(v_i, v_j) &= \tau^2 \dot{R}_0'(s_i,s_j) + \beta \dot{R}_1'(s_i,s_j) + \frac{\sigma^2}{\gamma}\delta_{ij}\\
\mathrm{E}(v_i, y_j) &= \tau^2 \dot{R}_0(s_i,s_j) + \beta \dot{R}_1(s_i,s_j) & \mathrm{E}(y_i, v_j) &= \tau^2 R_0'(s_i,s_j) + \beta R_1'(s_i,s_j)\\
\mathrm{E}(y_i, f(s)) &= \tau^2 R_0(s_i,s) + \beta R_1(s_i,s) & \mathrm{E}(y_i, f'(s)) &= \tau^2 R_0'(s_i,s) + \beta R_1'(s_i,s)\\
\mathrm{E}(v_i, f(s)) &= \tau^2 \dot{R}_0(s_i,s) + \beta \dot{R}_1(s_i,s) & \mathrm{E}(v_i, f'(s)) &= \tau^2 \dot{R}_0'(s_i,s) + \beta \dot{R}_1'(s_i,s)
\end{aligned}
$$

where $R(s,t)$ is taken from (3.14).

Observing $y_i \sim N(f(t_i), \sigma^2)$ and $v_i \sim N(f(t_i), \frac{\sigma^2}{\gamma})$, $i = 1, \ldots, n$, the joint distribution of $\mathbf{y}, \mathbf{v}$ and $f(t)$ is normal with mean zero and a covariance matrix can be found. The posterior mean of $f(t)$ is

$$
\begin{aligned}
\mathrm{E}(f \mid \mathbf{y}, \mathbf{v}) &= \begin{bmatrix} \mathrm{Cov}(\mathbf{y}, f) & \mathrm{Cov}(f, \mathbf{v}) \end{bmatrix} \begin{bmatrix} \mathrm{Var}(\mathbf{y}) & \mathrm{Cov}(\mathbf{y}, \mathbf{v}) \\ \mathrm{Cov}(\mathbf{v}, \mathbf{y}) & \mathrm{Var}(\mathbf{v}) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix}\\
&= \begin{bmatrix} \tau^2\phi^\top S^\top + \beta\xi^\top & \tau^2\phi^\top S'^\top + \beta\psi^\top \end{bmatrix} \begin{bmatrix} \tau^2 SS^\top + \beta Q + \sigma^2 I & \tau^2 SS'^\top + \beta P \\ \tau^2 S'S^\top + \beta Q' & \tau^2 S'S'^\top + \beta P' + \frac{\sigma^2}{\gamma} I \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix}\\
&= \begin{bmatrix} \rho\phi^\top S^\top + \xi^\top & \rho\phi^\top S'^\top + \psi^\top \end{bmatrix} \begin{bmatrix} \rho SS^\top + Q + n\lambda I & \rho SS'^\top + P \\ \rho S'S^\top + Q' & \rho S'S'^\top + P' + \frac{n\lambda}{\gamma} I \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix}\\
&= (\phi^\top \rho \begin{bmatrix} S \\ S' \end{bmatrix}^\top + \begin{bmatrix} \xi^\top & \psi^\top \end{bmatrix}) \left(\rho \begin{bmatrix} S \\ S' \end{bmatrix}^\top \begin{bmatrix} S \\ S' \end{bmatrix} + \begin{bmatrix} Q + n\lambda I & P \\ Q' & P' + \frac{n\lambda}{\gamma} I \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix}\\
&\triangleq \phi^\top \rho T^\top \left(\rho T^\top T + M\right)^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \xi^\top & \psi^\top \end{bmatrix} \left(\rho T^\top T + M\right)^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix}
\end{aligned}
\tag{3.19}
$$

where ϕ is 2×1 matrix with entry 1 and t, ξ is $n \times 1$ matrix with ith entry $R(t_i, t)$, $T^\top = \begin{bmatrix} S & S' \end{bmatrix}$ and ψ is $n \times 1$ matrix with ith entry $\dot{R}(t_i, t)$, $\rho = \tau^2/\beta$ and $n\lambda = \sigma^2/\beta$.

Lemma 3. *Suppose M is symmetric and nonsingular and T 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 (3.19) and applying Lemma 3, the posterior mean

$\mathrm{E}(f(x) \mid \mathbf{y}, \mathbf{v})$ is in the form $f = \phi^\top \mathbf{d} + \xi^\top \mathbf{c} + \psi^\top \mathbf{b}$, with the coefficients given by

$$\mathbf{d} = (T^\top M^{-1} T)^{-1} T^\top M^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix},$$

$$\begin{bmatrix} \mathbf{c} \\ \mathbf{b} \end{bmatrix} = (M^{-1} - M^{-1} T (T^\top M^{-1} T)^{-1} T^\top M^{-1}) \begin{bmatrix} \mathbf{y} \\ \mathbf{v} \end{bmatrix},$$

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 $\mathbf{d}, \mathbf{c}, \mathbf{b}$ are the solutions to

$$\begin{cases} S^\top (S\mathbf{d} + Q\mathbf{c} + P\mathbf{b} - \mathbf{y}) + \gamma S'^\top (S'\mathbf{b} + P^\top \mathbf{c} + S'^\top P' \mathbf{b} - \mathbf{v}) = 0, \\ Q(S\mathbf{d} + (Q + n\lambda I)\mathbf{c} + P\mathbf{b} - \mathbf{y}) + P(\gamma S'\mathbf{b} + \gamma P^\top \mathbf{c} + (\gamma P' + n\lambda I)\mathbf{b} - \gamma \mathbf{v}) = 0, \\ P^\top (S\mathbf{d} + (Q + n\lambda I)\mathbf{c} + P\mathbf{b} - \mathbf{y}) + P'(\gamma S'\mathbf{b} + P^\top \mathbf{c} + (\gamma P' + n\lambda I)\mathbf{b} - \gamma \mathbf{v}) = 0. \end{cases}$$

Finally we obtain the following theorem:

Theorem 7. *The Tractor smoothing spline of (3.12) is the posterior mean of $f = f_0 + f_1 + \dot{f}_1$, where f_0 diffuses in span $\{1, t\}$ and f_1 has a Gaussian process prior with mean zero and a covariance*

$$\mathrm{Cov}(f_1, f_1) = R_1(s,t) = \beta \int_0^1 (s-u)_+ (t-u)_+ du,$$
$$\mathrm{Cov}(\dot{f}_1, f_1) = \dot{R}(s,t) = \beta (t + \int_0^s (t-u)_+ du),$$

functions for $\beta = \sigma^2 / n\lambda$.

3.4 Tractor Spline with Correlated Random Errors

In most of the studies on polynomial smoothing splines, the random errors are assumed being independent. In contrast, in applications, observations are often correlated, such as time series data and spatial data. It is known that the correlation greatly affects the selection of smoothing parameters, which are critical to the performance of smoothing spline estimates Wang (1998). The parameter selection methods, such as generalized maximum likelihood (GML), generalized cross-validation (GCV), underestimate smoothing parameters when data are correlated.

Diggle and Hutchinson (1989) extended GCV for choosing the degree of smoothing spline to accommodate an autocorrelated error sequence, by which the smoothing parameter and autocorrelation parameters are estimated simultaneously. Kohn *et al.* (1992) proposed an algorithm to evaluate the cross-validation functions, whose autocorrelated errors are modeled by an autoregressive moving average. Wang (1998) extend GML and unbiased risk (UBR), other than GCV, to estimate the smoothing parameters and correlation parameters simultaneously. In this section, we are exploring the extended GCV for Tractor spline with correlated errors.

First of all, consider observations $y = f(t) + \varepsilon_1$ and $v = f'(t) + \varepsilon_2$, where $\varepsilon_1 \sim N(0, \sigma^2 W^{-1})$, $\varepsilon_2 \sim N(0, \frac{\sigma^2}{\gamma} U^{-1})$ with variance parameter σ^2. The Tractor spline $\hat{f}$ with correlated errors in space $\mathcal{C}^2_{p.w}[0,1]$ is the minimizer of

$$\frac{1}{n}(\mathbf{y} - \mathbf{f})^\top W(\mathbf{y} - \mathbf{f}) + \frac{\gamma}{n}(\mathbf{v} - \mathbf{f}')^\top U(\mathbf{v} - \mathbf{f}') + \lambda \int_0^1 (f'')^2 dt. \tag{3.20}$$

Because $f = \sum_{i=1}^{2n} \theta_i N_i(t)$ is a linear combination of basis functions, extended to the solution with covariance matrices, the coefficients is found by

$$\hat{\theta} = (B^\top W B + \gamma C^\top U C + n\Omega_\lambda)^{-1}(B^\top W \mathbf{y} + \gamma C^\top U \mathbf{v}). \tag{3.21}$$

Furthermore, in Gaussian process regression, the covariance matrix with correlated variances becomes $M = \begin{bmatrix} Q + n\lambda W & P \\ Q' & P' + \frac{n\lambda}{\gamma} U \end{bmatrix}$ and the rest remains the same.

Recall the leave-one-out cross-validation score of a Tractor spline is

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

Followed by the approximation $S_{ii} \approx \frac{1}{n}\text{tr}(S)$, $T_{ii} \approx \frac{1}{n}\text{tr}(T)$, $U_{ii} \approx \frac{1}{n}\text{tr}(U)$ and $V_{ii} \approx \frac{1}{n}\text{tr}(V)$ Syed (2011), the Tractor spline GCV will be

$$\text{GCV}(\lambda, \gamma) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{f}(t_i) - y_i + \frac{\gamma \text{tr}(T)/n}{1-\gamma \text{tr}(V)/n}(\hat{f}'(t_i) - v_i)}{1 - \text{tr}(S)/n - \frac{\gamma \text{tr}(T)/n}{1-\gamma \text{tr}(V)/n} \text{tr}(U)/n}\right)^2,$$

which may provide further computational savings since it requires finding the trace rather than the individual diagonal entries of the hat matrix. Hence, it can be written in the form of

$$\text{GCV}(\lambda, \gamma) = \frac{(\hat{\mathbf{f}} - \mathbf{y})^\top(\hat{\mathbf{f}} - \mathbf{y}) + \frac{2\text{tr}(\gamma T)}{\text{tr}(I-\gamma V)}(\hat{\mathbf{f}} - \mathbf{y})^\top(\hat{\mathbf{f}}' - \mathbf{v}) + \left(\frac{\text{tr}(\gamma T)}{\text{tr}(I-\gamma V)}\right)^2(\hat{\mathbf{f}}' - \mathbf{v})^\top(\hat{\mathbf{f}}' - \mathbf{v})}{\left(\text{tr}(I - S - \frac{\text{tr}(\gamma T)}{\text{tr}(I-\gamma V)} U)\right)^2}.$$

A natural extension to the above GCV for tractor spine with correlated errors is

$$\text{GCV}(\lambda, \gamma) = \frac{(\hat{\mathbf{f}} - \mathbf{y})^\top W(\hat{\mathbf{f}} - \mathbf{y}) + \frac{2\text{tr}(\gamma T)}{\text{tr}(I-\gamma V)}(\hat{\mathbf{f}} - \mathbf{y})^\top W^{1/2} U^{\top 1/2}(\hat{\mathbf{f}}' - \mathbf{v}) + \left(\frac{\text{tr}(\gamma T)}{\text{tr}(I-\gamma V)}\right)^2(\hat{\mathbf{f}}' - \mathbf{v})^\top U(\hat{\mathbf{f}}' - \mathbf{v})}{\left(\text{tr}(I - S - \frac{\text{tr}(\gamma T)}{\text{tr}(I-\gamma V)} U)\right)^2}. \tag{3.22}$$

The GCV is used for finding the unknown constant parameter λ, instead of a piecewise $\lambda(t)$ at different intervals, and the parameter γ. The structure of covariance matrices W and U is supposed to be known. If the errors are independent, in which way W and U become identity matrices, the solution $\hat{f}$ degenerate to a conventional Tractor spline with constant λ through over the entire interval $[0,1]$.

3.5 A Numeric Simulation

Consider a 100-length even spaced simulation data on $[0,1]$ and the model $f(t) = \sin(2\pi t)$, $f'(t) = \cos(2\pi t)$. Given parameters $\sigma^2 = 0.01$, $\gamma = 0.02$ and covariance matrices W, U, then the simulated observation is

$$\begin{cases} y_i = f(t_i) + \varepsilon_i^{(1)}, \\ v_i = f'(t_i) + \varepsilon_i^{(2)}, \end{cases} \tag{3.23}$$

where $\varepsilon^{(1)} \sim N(0, \sigma^2 W^{-1})$, $\varepsilon^{(2)} \sim N(0, \frac{\sigma^2}{\gamma} U^{-1})$, $0 \leq t_1 < \cdots < t_n \leq 1$.

With the same parameter found by GCV, Tractor spline and its Bayes estimate return the same reconstruction. However, because we are using constant λ rather than a piecewise $\lambda(t)$, the trivial Tractor spline might not be the optimal solution compared with the one consisting an adaptive penalty term in section 2.2.

3.6 Conclusion

In this Chapter, we take a review of the work that has been done in the last few decades for the relationship between polynomial smoothing spline and the Bayes estimates. With improper priors, the two methods are corresponding to each other. In fact, smoothing spline is a particular case of Gaussian process regression. By following the work done by Gu (2013), we find the Bayes estimate for Tractor spline with a constant penalty parameter λ. However, it is a particular scenario of Tractor spline whose $\lambda(t)$ varying on domains. So the trivial Tractor spline might not be an optimal solution. Additionally, we give the formula of GCV for Tractor spline with correlated errors on y and v.

A further work is to find the Bayes estimate for the real Tractor spline with a piecewise penalty term $\lambda(t)$. Besides GCV, other methods, such as EM and UBR are worth being invested in further research.

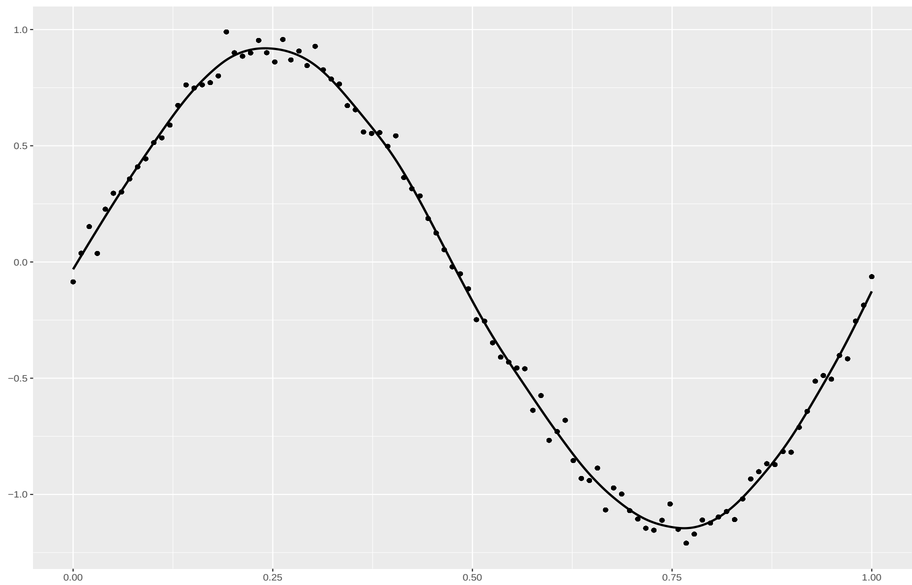


Figure 3.1: Comparing (trivial) Tractor spline and its Bayes estimate. Two methods are corresponding to each other. In this figure, black dots indicate observations and solid line stands for the reconstruction. The Tractor spline and its posterior $\mathrm{E}(f(t) \mid \mathbf{y}, \mathbf{v})$ of Bayes estimate give the same results.

Chapter 4

A Brief Overview of On-line State and Parameters Estimation

4.1 Introduction

As the development of the 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 prevalent in academic research in the last decades. When it embraced with state space model, which is a 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 models that rely on the concept of state variables. 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, which 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 has the form 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 ϵ_t and w_t are noise terms. 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 and can be applied to a system that exhibits 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, 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 celebrated 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:

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

In applications, the process function G_t and F_t contain unknown parameters to be estimated De Jong (1988) and the goal 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 θ is a set of unknown parameters.

Contents

In this chapter, I will give a brief overview of existing methods for sequential state and parameter inference. In section 4.2, I am 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 and compare them with a proposed algorithm 5.2 in section 5.5 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) and 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 named *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 did not develop 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 $x_k^{(1)}, \ldots, x_k^{(N)}$

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

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

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

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

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

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

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

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 its expectation, which is approximated by

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

where $\tilde{W}_t\left(x_t^{(i)}\right) = \frac{W_t\left(x_t^{(i)}\right)}{\sum_{i=1}^{N} W_t\left(x_t^{(i)}\right)}$ is factorized weight. Each particle has its own weighted value, so the overall expectation is a weighted mean. However, the drawback of this method is that the computation is 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 several iterations, only few particles have non-zero importance weights. This phenomenon is called *weight degeneracy* or *sample impoverishment* Arnaud Doucet (2011).

The effective sample size N_{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

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

If the value of N_{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 probability density function 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,

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

After resampling, it becomes

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

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 summarized in following Algorithm 4.1.

In SIR, if we choose

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

Algorithm 4.1: Sampling and Importance Sampling.

1 Initialization: Initial particles when $t = 0$. For $i = 1, \ldots, N$, draw samples $\{x_0^{(i)}\}$ from $p(x_0)$.

2 **for** $t = 1, 2, \ldots, n$ **do**

3 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.

4 Resampling: Resample $\{\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}\}$ and get a new set $\{x_t^{(i)}, \frac{1}{N}\}$.

5 Output the status at time t: $\hat{x}_t = \sum_{i=1}^N \tilde{x}_t^{(i)} \tilde{w}_t^{(i)}$.

6 **end**

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 by 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 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 $w_t^{\text{SISR}}(x_{1:t}) = \frac{p(x_{1:t})}{p(x_{1:t-1}q(x_t|x_{1:t-1})}$ but according to a generic score function $w_t(x_{1:t}) > 0$ at time t

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

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 π_{t-1} is used to draw an inference on π_t by SIS and resampling. Its interest is focusing on x_t instead of the whole path $x_{0:t}$, that is the filtering problem. However, this algorithm evolves, weights and re-samples 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 the sampling process.

As a complementary solution, *sequential particle filter* method was proposed by Nicolas Chopin (2002) in the first part of his doctorate 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, named as learning phase, and it is updated through importance sampling to incorporate the following l observations, named as updating phase, Chopin (2002). This method is the *iterated batch importance sampling* (IBIS) algorithm, which is used for the recursive exploration of the sequence of the parameter posterior distributions $\pi(\theta)$. It updates a population of N particles for θ, $\theta^{(1)}, \ldots, \theta^{(N)}$, 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 θ to prevent the number of distinct from decreasing over time.

In a batch mode, we are assuming that the parameter θ is static. When the first k observations become available, we can find the posterior distribution $\pi(\theta \mid y_{1:k})$. After that, with length of $l(<\infty)$ observations coming 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 set of proper re-weighted 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 Algorithm 4.2.

Algorithm 4.2: Sequential Particle Filter.

Initialization: General particles of θ_i and w_i, $i = 1, \ldots, N$.
while $k < t$ **do**
  Re-weighting. 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 θ_i and w_i^* to θ_i^* and $\frac{1}{N}$ according to $p(\theta_i^* = \theta_i) = \frac{w_i^*}{\sum w_i^*}$, $i = 1, \ldots, N$.
  Propagating. Draw θ_i^m from $K_{k+l}(\theta_i^*)$, where K_{k+l} is a predefined transition kernel function with stationary distribution π_{k+l}.
  Set $(\theta_i^m, \frac{1}{N})$ to (θ_i, w_i), $k + l$ to k.
end

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

4.2.6 MCMC-Based Particle Algorithm

A set of powerful stochastic algorithms that allow us to solve most of Bayesian computational problems is *Markov chain Monte Carlo* (MCMC) methods, which refer to constructing Markov chains that move in the space of the unobserved quantity of interest and produces a sequence of dependent samples from the posterior distribution. After the chain has been run long enough, the sequence of visited states may be used as an approximation to the posterior distribution Kokkala *et al.* (2016).

However, as dataset 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, but unfortunately are inefficient when apply to high dimensional problems Septier *et al.* (2009). A natural 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 combines 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 this issue, an 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. At the ith sampling iteration, the joint x_k and x_{k-1} was proposed in a Metropolis-Hastings sampling step. After that, a refinement

Metropolis-within-Gibbs step update x_k and x_{k-1} individually. Hence, the algorithm is summarized in algorithm 4.3:

Algorithm 4.3: MCMC-Based Particle Algorithm.

Initialization: Initialize particles $x_0^{(j)}, j = 1, \dots, N$.
for $k = 1, \dots, t$ **do**
  for $i = 1, \dots, M$ **do**
    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 $x_{k-1}^{(i)} = x_{k-1}^*$ 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 $x_k^{(i)} = x_k^*$ 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 $x_k^{(j)} = x_k^{(i)}$ as new particles for approximating $p(x_k \mid y_{1:k})$.
  end
end

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. One may refer to the paper for details.

4.3 On-line State and Parameters 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 θ, 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 θ, using the state space representation, as a component of the state which lacks dynamic evolution, also referred to as a static parameter Cappé *et al.* (2007). The standard SMC is deficient 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 θ. 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 θ 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 a 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 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 θ 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:

The speed and accuracy of this algorithm depend 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. A 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 (2001) use 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

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

Algorithm 4.4: Practical Filtering Algorithm.

Initialization: Set $\theta^{(i)} = \theta_0$ as initial values, $i = 1, \dots, N$.
Burn-In: **for** $k = 1, \dots, l$ **do**
  for $i = 1, \dots, N$ **do**
    Initialize $\theta = \theta^{(i)}$.
    Generate $x_{0:k} \sim p(x_{0:k} \mid \theta, y_{1:k})$ and $\theta \sim p(\theta \mid x_{0:k}, y_{1:k})$.
    After a few iterations, achieve a set of $\{\tilde{x}_{0:k}^{(i)}, \theta^{(i)}\}$.
  end
end
Sequential Updating: **for** $k = l+1, \dots, t$ **do**
  for $i = 1, \dots, N$ **do**
    initialize $\theta = \theta^{(i)}$.
    Generate $x_{k-l+1:k} \sim p(x_{k-l+1:k} \mid \tilde{x}_{k-l}^{(i)}, \theta, y_{k-l+1,k})$ and $\theta \sim p(\theta \mid \tilde{x}_{0:k-l}^{(i)}, x_{k-l+1:k}, y_{1:k})$.
    Achieve a set of $\{\tilde{x}_{k-l+1}^{(i)}, \theta^{(i)}\}$ and leave $\tilde{x}_{0:k-l}^{(i)}$ unchanged.
  end
end

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. θ_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:

Algorithm 4.5: Liu and West's Filter.

for $k = 1, \dots, t$ **do**

  for $i = 1, \dots, N$ **do**

    Identify the prior estimation of (x_k, θ) by $(\mu_{k+1}^{(i)}, m_k^{(i)})$, where $\mu_{k+1}^{(i)} = \mathrm{E}(x_{k+1} \mid x_k^{(i)}, \theta_k^{(i)})$, and $m_k^{(i)} = \alpha\theta_k^{(i)} + (1-\alpha)\bar{\theta}_k$.

    Sample an auxiliary integer index I from $\{1, \dots, N\}$ with probability proportional to $g_{k+1}^{(i)} \propto w_k^{(i)} p(y_{k+1} \mid \mu_{k+1}^{(i)}, m_k^{(i)})$.

    Sample a new parameter vector $\theta_{k+1}^{(I)}$ from $N(\theta_{k+1} \mid m_k^{(I)}, h^2 V_k)$.

    Sample current state vector $x_{k+1}^{(I)}$ from $p(x_{k+1} \mid x_k^{(I)}, \theta_{k+1}^{(I)})$.

    Evaluate weights $\tilde{w}_{k+1}^{(i)} \propto \frac{p(y_{k+1}|x_{k+1}^{(I)}, \theta_{k+1}^{(I)})}{p(y_{k+1}|\mu_{k+1}^{(I)}, m_k^{(I)})}$

  end

  Normalize weights: $w_{k+1}^{(i)} = \frac{\tilde{w}_{k+1}^{(i)}}{\sum \tilde{w}_{k+1}^{(i)}}$.

end

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 observational distributions with unknown parameters can be handled by this approach but subject to unavailable sufficient statistics.

The Storvik filter is summarized bellow:

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 throughout the process and utilize

Algorithm 4.6: Storvik Filter.

for $k = 1, \ldots, t$ **do**

  for $i = 1, \ldots, N$ **do**

    Sample $x_k^{(i)}$ from $p(x_k \mid x_{k-1}^{(i)}, y_{1:k}, \theta^{(i)})$.

    Calculate weights $\tilde{w}_k \propto p(y_k \mid x_k^{(i)}, \theta^{(i)})$ and normalize it by $w_k^{(i)} = \frac{\tilde{w}_k^{(i)}}{\sum \tilde{w}_k^{(i)}}$.

    Re-sample $\{\theta_k^{(i)}, x_k^{(i)}, s_k^{(i)}\}$ according to w_k.

    Update sufficient statistics $s_k^{(i)} = S(s_{k-1}^{(i)}, x_k, y_k)$.

    Sample $\theta^{(i)}$ from $p(\theta \mid s_k^{(i)})$.

  end

end

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:

Algorithm 4.7: Particle Learning Algorithm.

for $k = 1, \ldots, t$ **do**

  for $i = 1, \ldots, N$ **do**

    Re-sample $\tilde{z}_k^{(i)} = (\tilde{s}_k^{x(i)}, \tilde{s}_k^{(i)}, \tilde{\theta}^{(i)})$ from $p(z_k \mid s_k^x, s_k, \theta)$ with weight $w \propto p(y_{k+1} \mid s_k^x, \theta)$.

    Draw $x_{k+1}^{(i)}$ from $p(x_{k+1} \mid \tilde{s}_k^x, \tilde{\theta}, y_{1:k+1})$.

    Update sufficient statistics $s_{k+1} = S(\tilde{s}_k, x_{k+1}, y_{k+1})$.

    Sample $\theta^{(i)}$ from $p(\theta \mid s_{k+1})$.

    Update $s_{k+1}^x = K(s_k^x, \theta, y_{k+1})$.

  end

end

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 did not 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 as

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

and the second term in the recursive form is

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

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

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

where F_{t+1} is the observation map. The posterior ensemble at time $t+1$ based on parameter θ is

$$x^{(i)}_{t+1} = x^{f(i)}_{t+1} + \hat{K}_{t+1}(\theta)(y_{t+1} + v^{(i)}_{t+1} + F_{t+1}(\theta) x^{f(i)}_{t+1}), \tag{4.14}$$

where $\{x_t^{(i)}\}_1^N$ is an ensemble of states representing the filtering distribution at time t. $x_{t+1}^{f(i)}$ is the forecast ensemble from the forward map by $x_{t+1}^{f(i)} = x_{t+1}^{p(i)} + w_{t+1}^{(i)} = G(x_t^{(i)}) + w_{t+1}^{(i)}$. 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:

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

where $\hat{e}_{t+1}(\theta) = y_{t+1} - F_{t+1}(\theta)\hat{a}_{t+1}$, and $\hat{a}_{t+1} = \frac{1}{N}x_{t+1}^{p(i)}$

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 the way that $p(\theta \mid y_{1:t}) \propto p(y_t \mid \theta, y_{1:t-1})p(\theta \mid y_{1:t-1})$ and update the recursion weights by $\pi_{t,k} \propto p(y_t \mid \theta, y_{1:t-1})\pi_{t,k-1}$.

To summarize it up, the complete algorithm is in the following

Algorithm 4.8: Adaptive Ensemble Kalman Filter.

```
Initialize samples θ^(i) ~ p(θ) and x_1^(i) ~ N(x_0, P_0), i = 1, ..., N.
for k = 1, ..., t do
    for i = 1, ..., N do
        Propagate. x_k^{p(i)} = G(x_{k-1}^(i)).
        Approximate likelihood function by (4.15).
        Update. Draw θ either use normal approximation or grid-based
        approximation to find (4.13).
        Draw θ^(i) ~ p(θ | y_{1:k}).
        Generate forecast ensemble by x_k^{f(i)} = x_k^{p(i)} + w_k.
        Draw posterior ensemble using (4.14).
    end
end
```

This algorithm works well when θ is small and the parameter in the forward map $G(\cdot)$ is known. If the forward map parameter is not known and has a high correlation with the state, the author suggests that it can be combined with the state augmentation Anderson (2001) and this algorithm is still working.

4.3.7 On-line Pseudo-Likelihood Estimation

Bayesian estimation requires the posterior distribution of $p(\theta \mid y_{1:t})$, where the θ 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 θ_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 $X_k = x_{kL+1:(k+1)L}$ and $Y_k = y_{kL+1:(k+1)L}$. 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 to 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 (LW), Storvik filter (St), Particle Learning (PL) and the proposed sequential MCMC Algorithm 5.2 in Section 5.5 by a simple dynamic linear model, see example Liu and West

(2001). Explicitly, the model is

$$y_t = Fx_t + \epsilon_t,$$
$$x_t = \phi x_{t-1} + w_t,$$
$$x_0 \sim N(m_0, C_0),$$

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 of 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^x for state x and s_t for parameter ϕ are satisfying the updating rules $s_t^x = K(s_{t-1}^x, \phi, y_t)$ and $s_t = S(s_{t-1}, x_t, y_t)$ respectively. Because of the assumption, the Kalman observation map is $H_k = 1$ and the variances are normal distributed. Thus, the Kalman gain is $K = 1$. 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
$$p(y_{t+1} \mid s_t^x, \phi) \propto \exp\left(-\frac{1}{2}(y_{t+1} - \phi x_t)^2\right).$$
- Step 2. Draw $x_{t+1}^{(i)}$ from $p(x_{t+1} \mid \tilde{s}_t^x, \tilde{\phi}, y_{1:t+1})$.
$$\begin{aligned} p(x_{t+1} \mid \tilde{z}_t^{(i)}, y_{1:t+1}) &= p(x_{t+1} \mid s_t^x, \phi, y_{1:t+1}) \propto p(x_{t+1}, y_{1:t+1} \mid s_t^x, \phi) \\ &\propto p(x_{t+1} \mid s_t^x, \phi) p(y_{t+1} \mid x_{t+1}, s_t^x, \phi) \\ &= N(x_{t+1} \mid \phi x_t, 1) N(y_{t+1} \mid x_{t+1}, 1) \\ &= N\left(x_{t+1} \mid \frac{1}{2}(y_{t+1} + \phi x_t), \frac{1}{\sqrt{2}}\right) \end{aligned}$$
- 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. Sampling ϕ from $p(\phi \mid s_{t+1})$.
$$\begin{aligned} p(\phi \mid x_{1:t+1}, y_{1:t+1}) &\propto p(x_{1:t+1}, y_{1:t+1} \mid \phi) p(\phi) \propto p(x_{1:t+1} \mid \phi) p(\phi) \\ &= N\left(\phi \mid \frac{s_{t+1,2}}{s_{t+1,3}}, \frac{1}{s_{t+1,3}}\right). \end{aligned}$$

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

Notice that the proposed sliding window MCMC algorithm requires using a few data to learn the parameter's mean and variance in the learning phase. Besides, LW, St and PL do not converge at the first few data. Then, to be fair, we take the first 300 data out and use them in the learning phase of the proposed algorithm. In the estimation phase, we are using three different strategies: a fixed length of 100, a fixed length of 300 and an expanding length including all the historical data along time t. In the former three algorithms, we are using 5 000 particles to infer state and parameter. In the latter MCMC algorithm, we are taking 5 000 samples at each time t for $x_t^{(i)}$ and $\phi^{(i)}$, where $i = 1, \ldots, 5\,000$ and $t = 300, \ldots, 897$. Furthermore, we run the comparison for 50 times to check their stabilities.

From figure 4.1 it can be seen that the former three algorithms converge to the true parameter ϕ with similar speeds along time t. LW filter has a larger distance to the true parameter comparing with St and PL filters. The proposed MCMC with 100 length data in estimation phase has the biggest variance. By setting a longer length, the MCMC becomes more stable.

The estimations for x_t from $t = 300$ to 897 are very close and hard to tell visually. From table 4.2, we can see that, by repeatedly running 50 times, the proposed MCMC algorithm is more stable in estimating parameter ϕ as it has the lowest standard error (SE) among all the algorithms. The MSE of all the algorithms on estimating x is on the similar levels.

4.5 Conclusion

In this chapter, we take a brief overview of existing sequential state estimation and combined state and parameters estimation. The particle filter is an efficient method for state estimation, although the particles impoverishment is inevitable. Extended to this method, researchers are working on killing particles degeneracy and several accomplishments have been achieved. A new challenge is estimating the unknown parameters. In no doubt, for complex stochastic processes and dynamics, both Bayesian and Frequentist methods are working well. One may refer to Wakefield (2013) for further discussion between Bayesian and Frequentist methods and implementation.

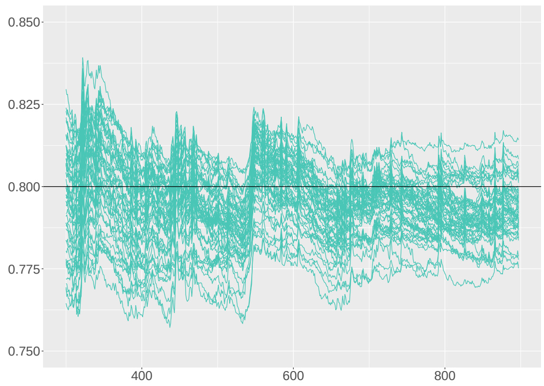
As a summary, most of the research issues for filtering and parameter estimation focused on

- Choices of resampling scheme to kill particles degeneracy, such as Liu and West's filter.

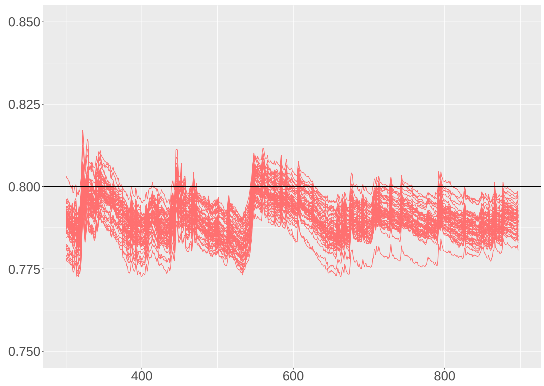
- Exploration of an efficient sampling algorithm, such as Metropolis-Hastings sampler and related sampling algorithms.

- Exploration of accurate estimation with low variances in higher dimensional space.

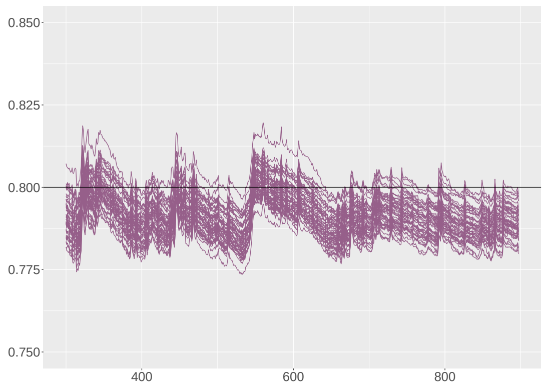
Subject to pages and time, some algorithms are not included in this chapter. For more information and interests, readers can refer to unscented Kalman filter Wan and Van Der Merwe (2000) and its related algorithms for non-linear estimation, particle MCMC Andrieu *et al.* (2010) for off-line Bayesian estimation and on-line gradient approach Poyiadjis *et al.* (2005) for parameter estimation.



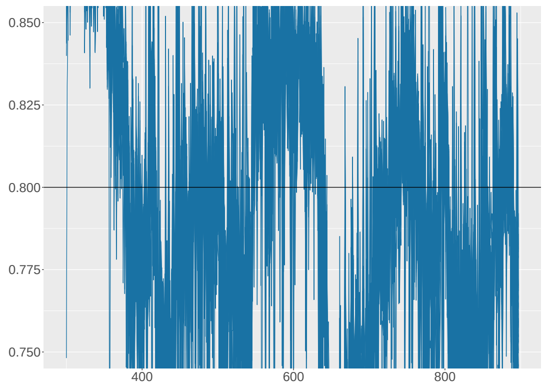
(a) Liu and West's filter



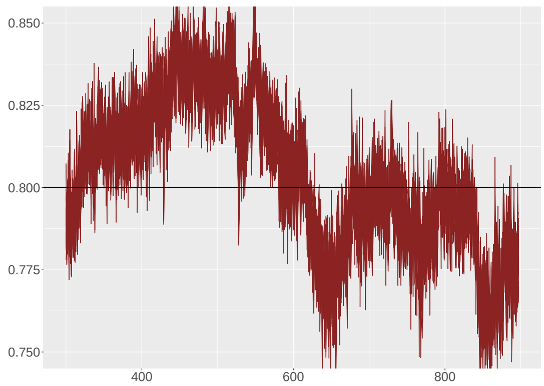
(b) Storvik Filter



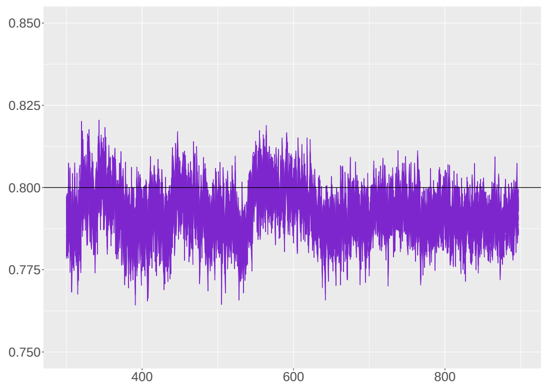
(c) Particle Learning



(d) MCMC with 100-length

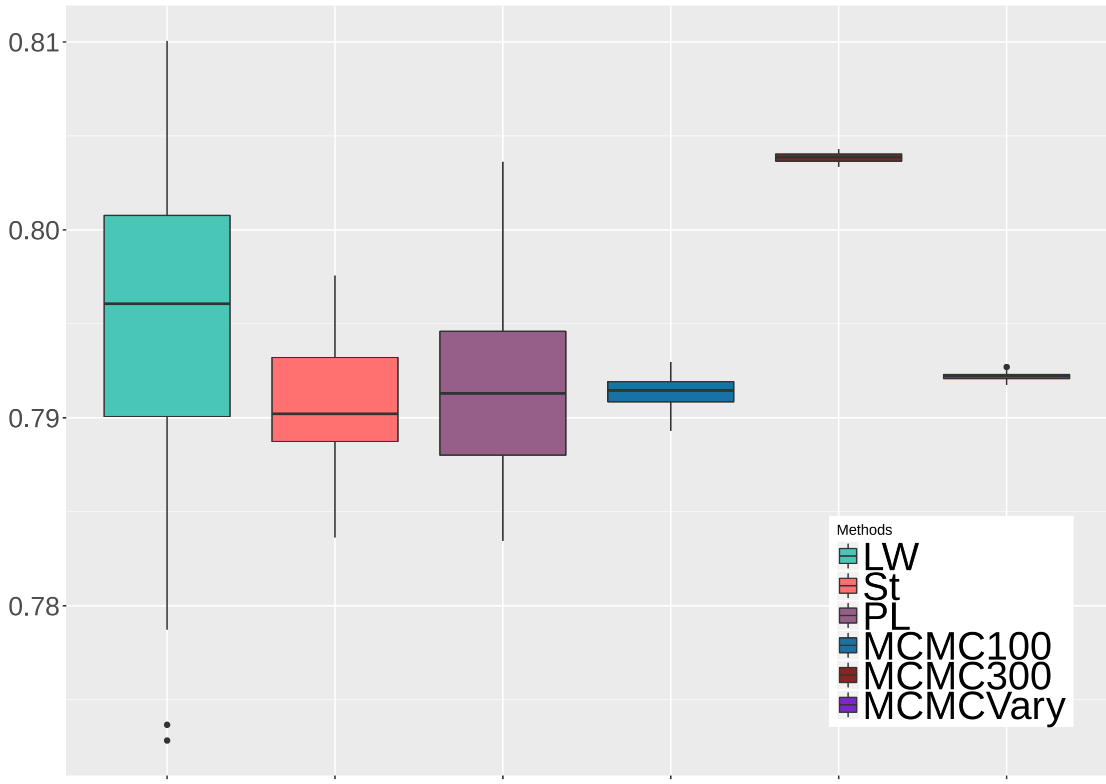


(e) MCMC with 300-length



(f) MCMC with varying-length

Figure 4.1: Trace plots. Repeatedly running 50 times. Cutting off the first **300** data. It is apparent that all these algorithms converge to the true parameter (black horizontal line) along time. St, PL and MCMC-vary have a narrower range. MCMC-100 has a higher variability and MCMC-vary has the least. The more data incorporated in the estimation phase the better approximation to be obtained.



	LW	St	PL	MCMC-100	MCMC-300	MCMC-vary
Mean of $\hat{\phi}$	0.7947	0.7908	0.7918	0.7914	0.8038	0.7922
Se of $\hat{\phi}$ $(\times 10^{-4})$	13.1000	4.8793	6.2877	1.1388	0.3275	0.27506
$\frac{1}{n}\sum_i(\hat{x}_i - x_i)^2$	0.5745	0.5739	0.5737	0.5875	0.5741	0.5740

Figure 4.2: Box-plots comparison of all the algorithms. The proposed MCMC algorithm is more stable than other filters.

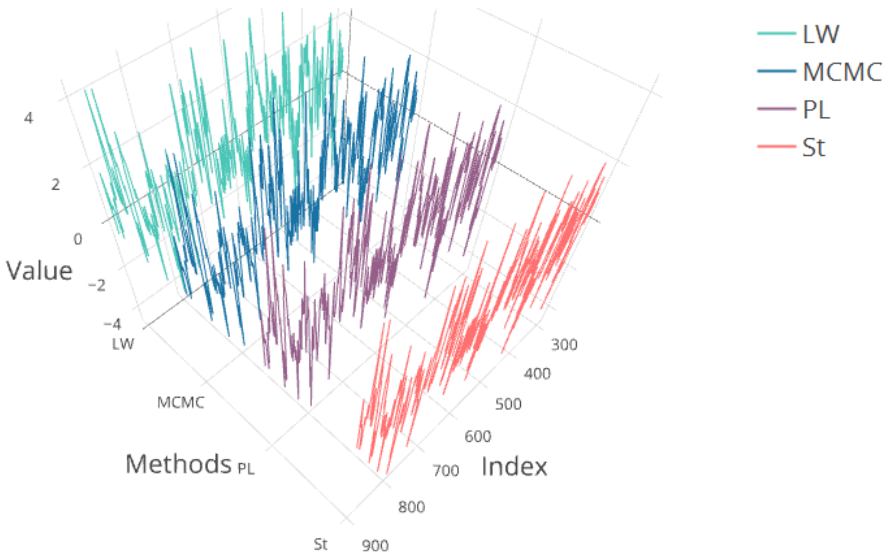


Figure 4.3: The filtering for $x_{300:897}$ is competitive. These algorithms return close estimations.

Chapter 5

Adaptive Sequential MCMC for On-line State and Parameters Estimation

5.1 Introduction

Data assimilation is a sequential process, by which the observations are incorporated into 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. An enormous literature has been done on discussing pure state estimation, however, less 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, Storvik 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 Storvik 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) an 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 historical data, is introduced to avoid the difficulties of tunning the proposal distribution in Metropolis-Hastings methods.

In this chapter, I am proposing an adaptive Delayed-Acceptance Metropolis-Hastings algorithm to estimate the posterior distribution for combined state and parameters with two phases. In the learning phase, a self-tuning random walk Metropolis-Hastings sampler is used to learn the parameter mean and covariance structure. In the estimation phase, the parameter mean and covariance structure informs the proposed mechanism and is also used in a delayed-acceptance algorithm, which greatly improves sampling efficiency. Information on the resulting state of the system is given by a Gaussian mixture. To keep the algorithm a higher computing efficiency for on-line estimation, it is suggested to cut off historical data and to use a fixed length of data up to the current state, like a window sliding along time. At the end of this chapter, an application of this algorithm on irregularly sampled GPS time series data is presented.

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 θ, it requires computing the posterior distribution of current state $p(x_t \mid y_{1:t})$ at each time $t = 1, \dots, T$ 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 $y_{1:t} = \{y_1, \dots, y_t\}$ is the observation information up to time t. However, if θ 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 θ. 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 $(i = 1, \dots, N)$ 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 θ 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 θ, 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 θ, 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 θ, suppose the joint covariance matrix is in the form of

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

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 θ is the set of all known and unknown parameters. The inverse of the covariance matrix Σ_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 θ 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 θ 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

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

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 1×1 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 θ 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 θ 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 θ_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 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 π (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 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 $\alpha \geq 1$, the new state is accepted. Otherwise, the new state is accepted with probability α.

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 $\epsilon = |x' - x^{(t)}|$ from the pre-specified Lebesgue density

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

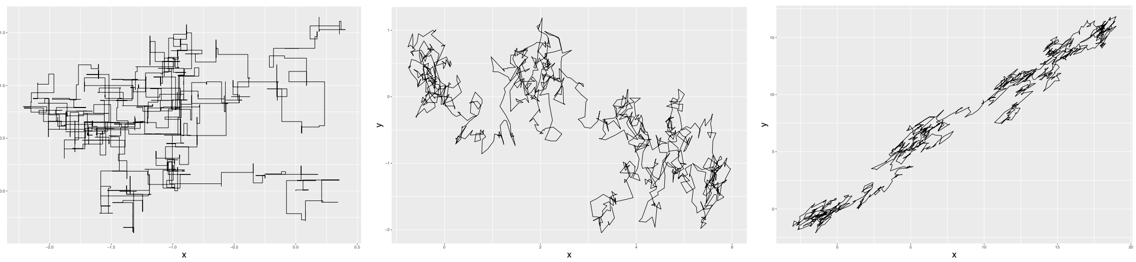
with $\gamma(\epsilon) = \gamma(-\epsilon)$ for all ϵ. Here, the positive λ governs the overall distance of the proposed jump and plays a crucial role in determining the efficiency of any algorithm. In a random walk, the proposal density function $q(\cdot)$ can be chosen for some suitable normal distribution, 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, I am 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 5.1a 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 5.1b and 5.1c are using multi-variable-at-a-time Random Walk. The difference is in figure 5.1b, every forward step are proposed independently, but in 5.1c 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, 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 5.1: 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 α_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 θ_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 Take n samples out from N with equal spaced index for each parameter being a new sequence.

The advantage of the algorithm (10) is that it returns a more accurate estimation for θ and it is more reliable to learn the structure of parameter space. However, if $\pi(\cdot)$ is in an irregular structure, the algorithm is really time-consuming and that cause a lower efficiency. 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 α_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 α_1 is accepted, the process goes into stage two and the acceptance probability α_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, 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, 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 tunes each parameter at its own optimal step size and explores 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 θ' is proposed from $N(\theta^{(t)} \mid m_L, C_L)$, which is in the following form

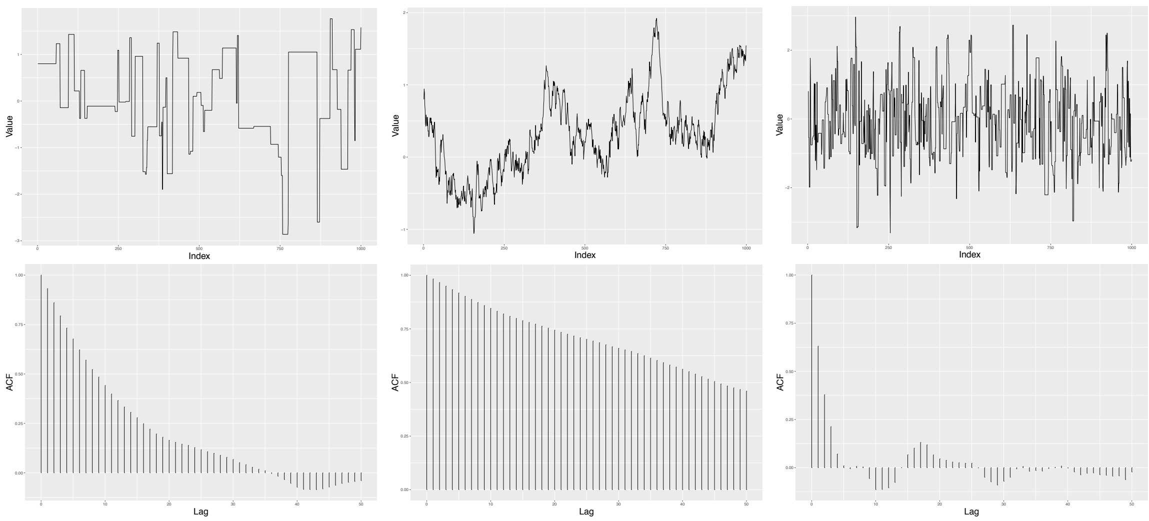
$$\theta' = \theta^{(t)} + R\epsilon z, \tag{5.25}$$

where $R^\top R = C_L$ is the Cholesky decomposition, ϵ is the tuned step size and $z \sim N(0, 1)$ is Gaussian white noise. This proposing method reduces the impact of drawing θ' from a correlation space.

5.3.3 Efficiency of Metropolis-Hastings Algorithm

In equation (5.19), the jump size ϵ determines the efficiency of RWM algorithm. For a general RWM, it is intuitively clear that we can make the algorithm arbitrarily poor by making ϵ either very large or very small Sherlock *et al.* (2010). Assuming ϵ 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 ϵ is extremely small, the algorithm will keep accepting the proposed x' since α is always approximately be 1 because of 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, the balance between these two extreme situations must exist. This appropriate step size $\hat{\epsilon}$ is optimal, sometimes is suboptimal, the solution to gain a Markov chain. Figure 5.2 illustrates the performances of RWM with different step size ϵ. From these plots we may see that either too large or too small ϵ causes high correlation chains, indicating bad samples in sampling algorithm. An appropriate ϵ 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 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 a proper step size

Figure 5.2: Metropolis algorithm sampling for a single parameter with: 5.2a a large step size, 5.2b a small step size, 5.2c 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 ϵ, 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 ensures 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 certain that one could use the actual arctangent relationship to try to choose a good ϵ: in the univariate example, if α is the desired acceptance rate, then $\epsilon = 2\sigma/\tan(\pi/2\alpha)$, where σ is the posterior standard deviation, will be obtained. In fact, some explorations infer a linear relationship between acceptance rate and step size, which is $\texttt{logit}(\alpha) \approx 0.76 - 1.12\ln\epsilon/\sigma$, 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 σ 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 Σ is known or could be estimated, then X' can be proposed from $q \sim N(X, \epsilon^2\Sigma)$. Thus,the optimal step size ϵ 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 multivariate normal distribution is governed by its covariance matrix Σ, 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

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

where X_0 is assumed to be distributed according to π. Because central limit theorem, the variance of the estimator $\bar{g} = \sum_{i=1}^{n} g(X_i)/n$ for estimating $\operatorname{E}(g(X))$ is approximately $\operatorname{Var}_\pi(g(X)) \times \tau_g/n$. 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

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

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

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

where n is the number of samples, $k_{\texttt{cut}}$ is lag of the first $\rho_k < 0.01$ or 0.05 , and τ 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 research, we now define the Efficiency in Unit Time (EffUT) and ESS in Unit Time (ESSUT) as follows:

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

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

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 ϵ 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 EffUT and ESSUT do. The best-tuned step size gains 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

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

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

5.4.1 Simulation on Regularly Sampled 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 σ and τ are i.i.d.errors occurring in processes and ϕ 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 $y_{1:t}$

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

where $\theta = \{\phi, \sigma, \tau\}$, we should start from the procedure matrix Σ^{-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 10, 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 θ is

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

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

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

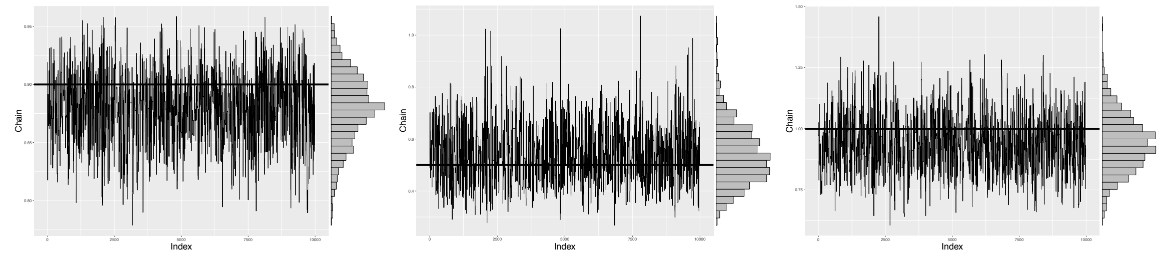
In a simple linear case, we are choosing the parameter $\theta = \{\phi = 0.9, \tau^2 = 0.5, \sigma^2 = 1\}$ as the author did in Lopes and Tsay (2011) and using $n = 500$ dataset, setting initial $L = 0$. Instead of inferring τ and σ, we are estimating $\nu_1 = \ln \tau^2$ and $\nu_2 = \ln \sigma^2$ 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 (10) 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 θ.

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 ϕ. (b) Trace plot of τ^2. (c) Trace plot of σ^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 Σ_{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

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

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).

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 θ 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

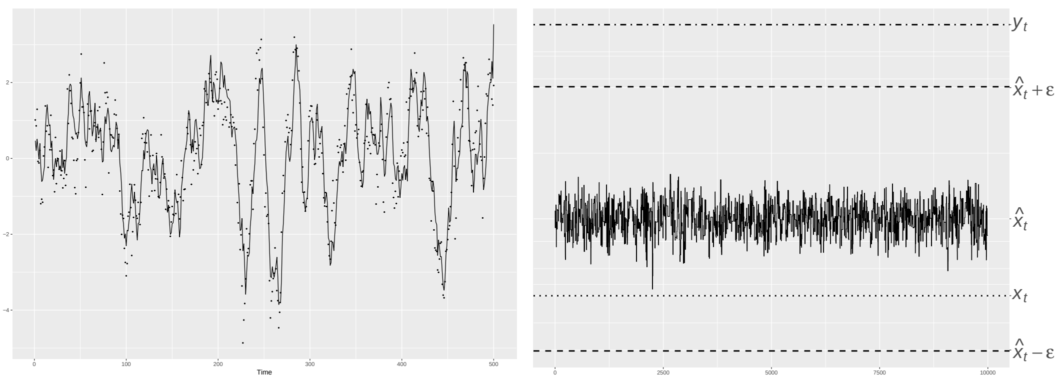
$$C_t^\top A_t^{-1} = \left[-b_t^\top \quad \sigma^2 - K_t \right],$$

thus, for any given θ, 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).



(a) Estimation the whole $x_{1:t}$

(b) Estimation of a single x_t

Figure 5.4: Linear simulation of $x_{1:t}$ and single x_t.In figure 5.4a, the dots is the true $x_{1:t}$ and the solid line is the estimation $\hat{x}_{1:t}$. In figure 5.4b, the estimation $\hat{x}_t$ is very close to the true x. In fact, the true x falls in the interval $[\hat{x} - \varepsilon, \hat{x} + \varepsilon]$.

5.4.2 Simulation on Irregularly Sampled Time Series Data

Irregularly 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 adopts 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 λ 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, τ 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

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

where $\varepsilon \sim N(0, \sigma)$ 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: $\gamma = 0.5$, $\lambda^2 = 0.1$, $\sigma^2 = 1$.

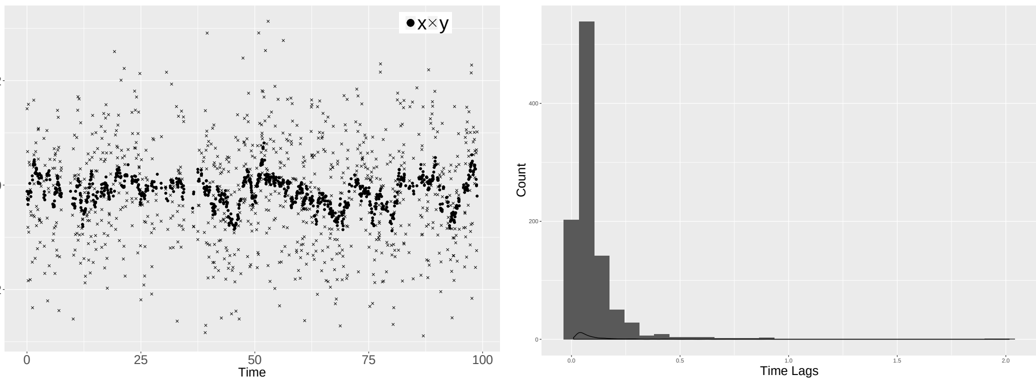


Figure 5.5: Simulated data. The solid dots indicate the true state x and cross dots indicate observation y. Irregular time lag Δ_t are generated from *Inverse Gamma*(2,0.1) distribution.

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

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

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 $\phi_t = e^{-\gamma\Delta_t}, \tau_t^2 = \frac{\lambda^2}{2\gamma}\left(1 - e^{-2\gamma\Delta_t}\right)$, θ 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 10, similarly with section 5.2.1, we firstly need to find the posterior distribution of θ with observations $y_{1:t}$, which in fact is

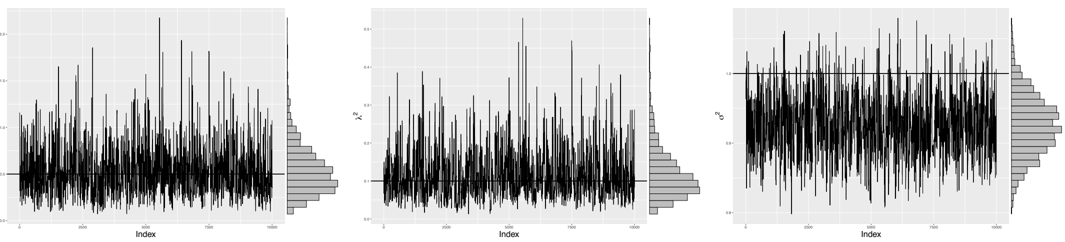
$$p(\theta \mid Y) \propto p(Y \mid \theta)p(\theta) \propto e^{-\frac{1}{2}Y\Sigma_{YY}^{-1}Y}\sqrt{\det \Sigma_{YY}^{-1}p(\theta)}.$$

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

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

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 θ chains are skew to the true value with tails.



(a) Trace plot of γ. (b) Trace plot of λ^2. (c) Trace plot of σ^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 θ.

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

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

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

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 θ 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

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

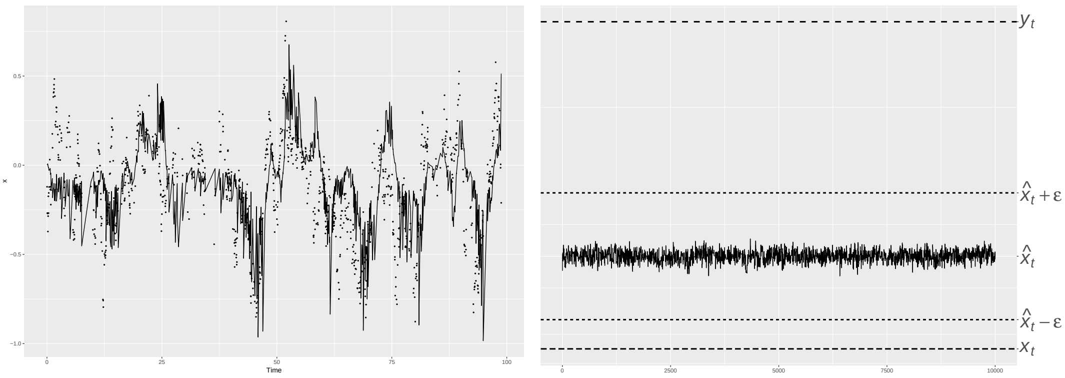
$$\mathrm{Var}(x_t) = \frac{1}{N}\sum_i \left(\mu_{ti}^{(x)}\mu_{ti}^{(x)\top} + \mathrm{Var}(x_t)_i\right) - \frac{1}{N^2}\left(\sum_i \mu_{ti}^{(x)}\right)\left(\sum_i \mu_{ti}^{(x)}\right)^\top, \tag{5.51}$$

The same as we did in section 5.4.1, for any given θ, 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 Δ_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 figure 5.7a, the dots is the true $x_{1:t}$ and the solid line is the estimation $\hat{x}_{1:t}$. In figure 5.7b, 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 Ornstein-Uhlenbeck Process Application

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

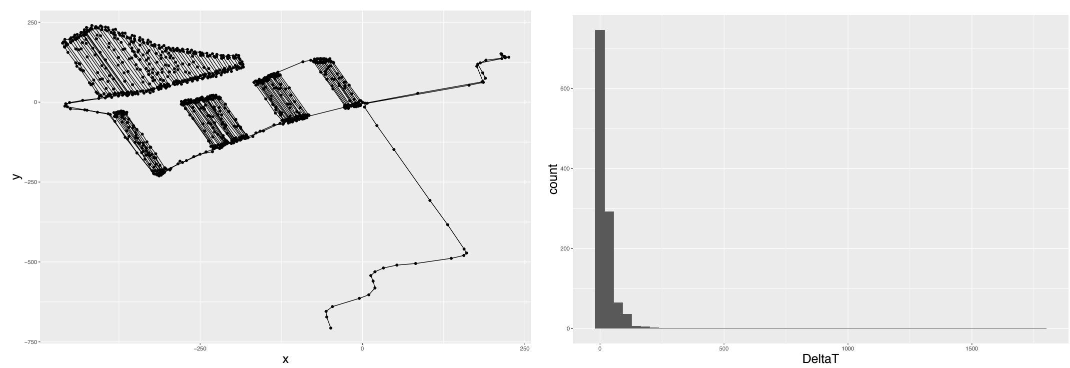


Figure 5.8: The trajectory of a moving tractor. The time lags (right side figure) obtained from GPS units are irregular.

Therefore, in this section, we are introducing an Ornstein-Uhlenbeck process (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 θ 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 θ, it can be found that

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

where $\tilde{X}$ represents for the hidden statues $\{x, u\}$, $\tilde{Y}$ represents for observed $\{y, v\}$, θ 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 Σ^{-1} looks like

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

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. For sake of simplicity, 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

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

A detailed structure of the covariance matrix Σ_{XX} is presented in section B.3.

5.5.1 Approximations of The Parameters Posterior

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

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

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 θ 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 2×2 matrix.

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

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

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 2×2 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.2.

5.5.3 The Estimation Distribution

Because of the joint distribution (5.58), one can find the best estimation with a given θ 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

$$\mu_t^{(X)} = C_t^\top A^{-1} B Y = C_t^\top L^{-\top} W,$$
$$\Sigma_t^{(X)} = C_t^\top A^{-1} C_t = U_t^\top U_t,$$

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 work 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 ω_t and ϵ_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 α 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

$$\begin{aligned} p(y_t \mid x_t, \alpha) &= N(F(x_t), \alpha R), \\ p(x_t \mid x_{t-1}, \alpha) &= N(G(x_{t-1}), \alpha Q). \end{aligned}$$

The parameter α 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 γ is typical velocity falling in the reasonable range of 0.1 to 100 m/s. ξ is the error occurs in transition process, σ and τ 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 do not have any assumptions about the true value of λ, which means it could be anywhere. According to this assumption, the prior distributions are

$$\begin{aligned} \gamma &\sim IG(10, 0.5), \\ \xi^2 &\sim IG(5, 2.5), \\ \sigma^2 &\sim IG(5, 2.5), \end{aligned}$$

where $IG(\alpha, \beta)$ represents the *Inverse Gamma* distribution with two parameters α and β.

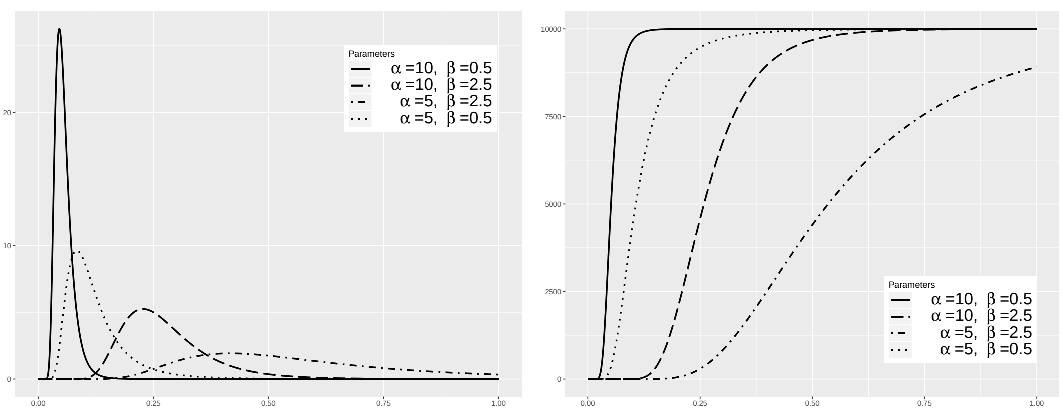


Figure 5.9: Probability density function and cumulative distribution function of *Inverse Gamma* with two parameters α and β.

5.5.5 Efficiency of DA-MH

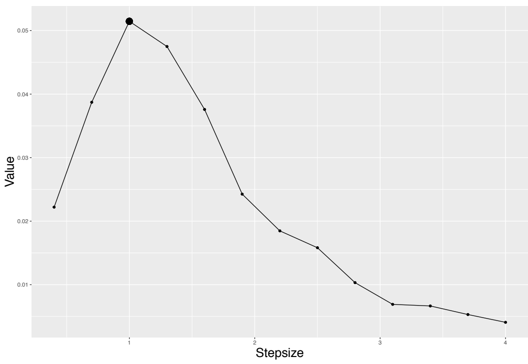
We have discussed the efficiency of Delayed-Acceptance Metropolis-Hastings algorithm and how it is affected by the step size. To explain explicitly, here we give an example comparing Eff, EffUT, ESS and ESSUT, which are calculated by using the same dataset and running 10 000 iterations of DA-MH. We are 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 θ and the acceptance rates α_1 for approximate $\hat{\pi}(\cdot)$ and α_2 for posterior distribution $\pi(\cdot)$ are 0.3097 and 0.8324 respectively. By using EffUT and ESSUT, the best step size is 2.5, which is bigger. The advantages of using this kind of step size are 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 α_1 in table 5.1.

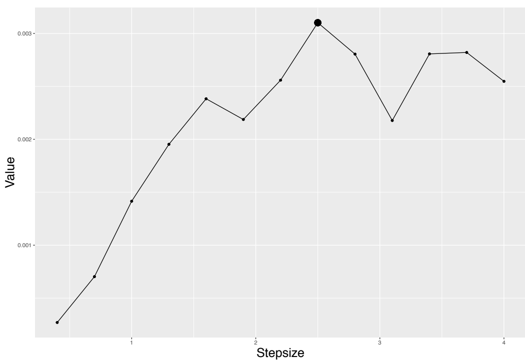
	Values	Time	Step Size	α_1	α_2
Eff	0.0515	36.35	1.0	0.3097	0.8324
EffUT	0.0031	5.10	2.5	0.0360	0.7861
ESS	501.4248	36.35	1.0	0.3097	0.8324
ESSUT	29.8912	5.10	2.5	0.0360	0.7861

Table 5.1: An example of Eff, EffUT, ESS and ESSUT found by running 10 000 iterations with same data. The computation time is measured in seconds s.

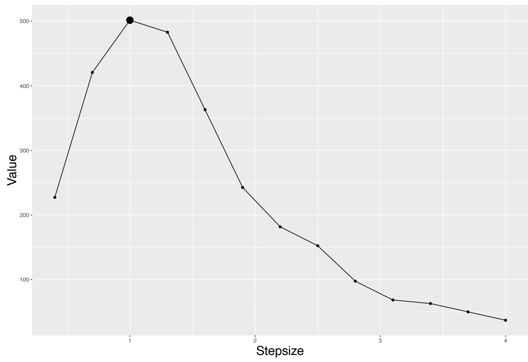
On the surface, a bigger step size causes lower acceptance rates α_1 and it might not be a smart choice. However, on the other hand, one should notice the less time cost. To make it sensible, we are running the Delayed-Acceptance MH with different step sizes, as presented in table 5.1, for the same (or 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, EffUT, ESS and ESSUT separately using the embedded function IAT, Christen *et al.* (2010), and ESS of the package LaplacesDemon in *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



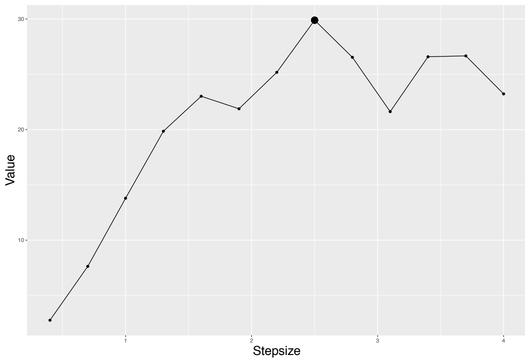
(a) Efficiency against different step sizes



(b) EffUT against different step sizes



(c) ESS against different step sizes



(d) ESSUT against different step sizes

Figure 5.10: Influences of different step sizes on sampling efficiency (Eff), efficiency in unit time (EffUT), effective sample size (ESS) and effective sample size in unit time (ESSUT) found by using the same data

generates more representative samples per second. Figure B.1 is comparing different θ chains found by using different step sizes but running the same amount of time. As we can see that θ with the optimal step size has a lower correlated relationship.

5.5.6 Sliding Window State and Parameters 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.*

Step Size	Length	Time	Eff	EffUT	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	75.3706	22.1678
1.3	1 000*	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*	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*	3.62	**0.1095**	**0.0302**	**113.4063**	**31.3277**

Table 5.2: Comparing Eff, EffUT, 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. The computation time is measured in seconds s.

(2008) and named as *Practical Filtering.* The authors rely on the approximation of

$$p(x_{0:n-L}, \theta \mid y_{0:n-1}) \approx p(x_{0:n-L}, \theta \mid y_{0:n})$$

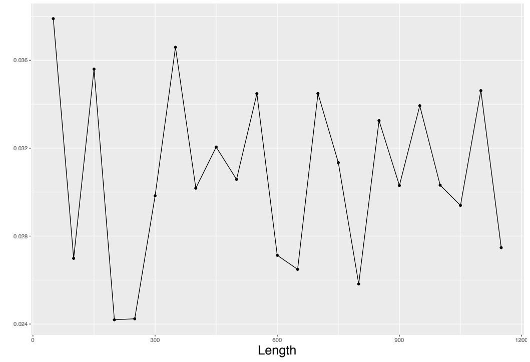
for large L. The new observations coming after the nth data has little influence on $x_{0:n-L}$.

Being inspired, we are not using the first 0 to $n-1$ date and ignoring the latest nth, 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

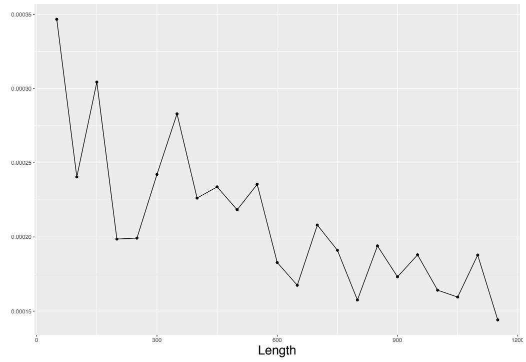
$$p(X_t \mid Y_{t-L+1:t}, \theta), \tag{5.65}$$

where $t > L$. We name this method *Sliding Window Sequential Parameter Learning Filter.*

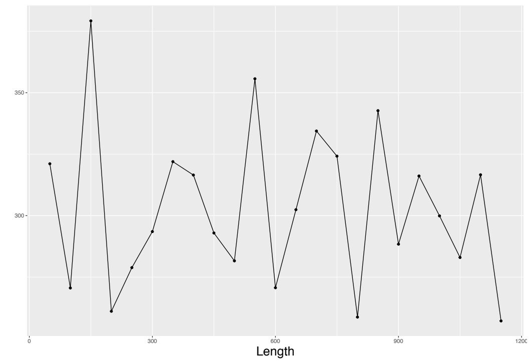
The next question is how to choose an appropriate L. The length of data used in MH and DA-MH algorithms has an influence on the efficiency and accuracy of parameter learning and state estimation. Being tested on real data set, there is no doubt that the more data be in use, the more accurate the estimation is, and lower efficient is in computation. In table B.4, one can see the pattern of parameters γ, ξ, τ follow the same trend with the choice of L and σ 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 is not varying along the sample size used in sampling algorithm, but in unit time, they are decreasing rapidly as data size increases. In addition, from a practical point of view, the observation error σ



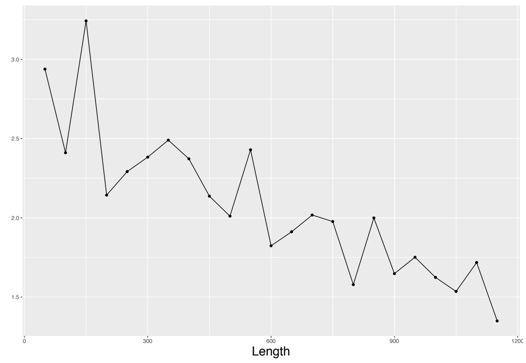
(a) Efficiency against data length



(b) EffUT against data length



(c) ESS against data length



(d) ESSUT against data length

Figure 5.11: Comparison efficiency (Eff), efficiency in unit time (EffUT), effective sample size (ESS) and effective sample size in unit time (ESSUT) against the different length of data. Increasing data length does not significantly improve the efficiency and ESSUT.

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 a one-dimensional linear model, L can be chosen larger and that does not change too much. If the data up to time t is less than or equal to the chosen L, the whole data set is used in learning θ 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 the DA-MH process, the cheap $\hat{\pi}$ does not 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 θ and find out the cheap approximation $\hat{\pi}$. Then, use DA-MH to estimate the true posterior π for θ 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 θ 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.

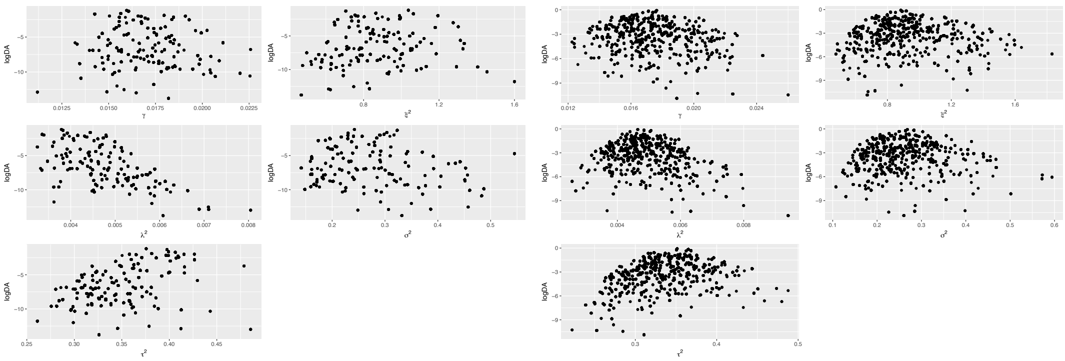
To avoid estimation bias in the algorithm, we are introducing *threshold* and *cut off* processes. *threshold* means when a bias occurs in the algorithm, the cheap $\hat{\pi}$ may not be appropriate and a new one is needed. Thus, we have to update $\hat{\pi}$ with a latest data we have. A *cut off* process stops the algorithm when a large Δ_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 in the following algorithm 5.2:

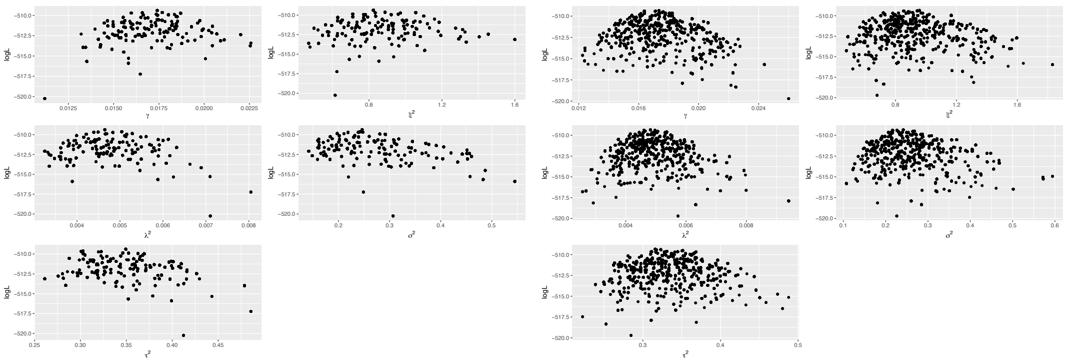
5.5.7 Implementation

To implement the algorithm 5.2, 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 θ samples in 59 seconds. For each parameter of θ, we take 1 000 sub-samples out of 5 000 as new sequences. The new θ^* is representative for the hyper parameter space. Then,the traces and correlation is derived from θ^*. 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 5.2 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 $t =$ 648th 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



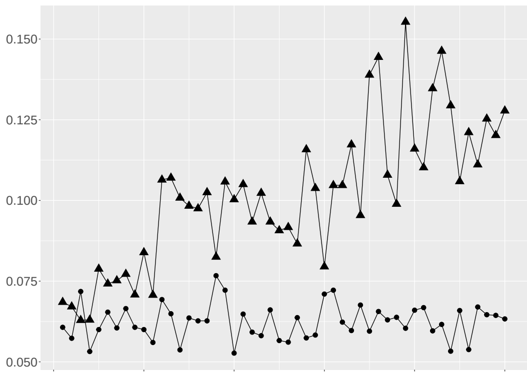
(a) $\ln DA$ surfaces of not-updating-mean (b) $\ln DA$ surfaces of updating-mean



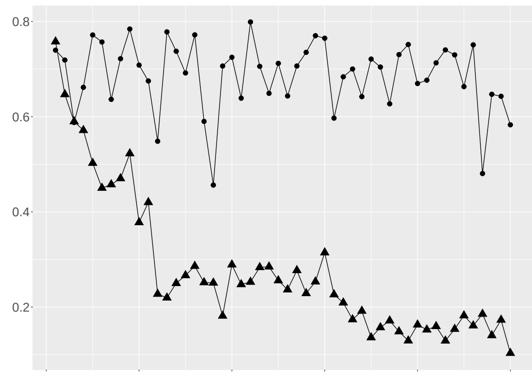
(c) $\ln L$ surfaces of not-updating-mean (d) $\ln L$ surfaces of updating-mean

Figure 5.12: Comparison $\ln DA$ and $\ln L$ surfaces between not-updating-mean and updating-mean methods. It is obviously that the updating-mean method has dense log-surfaces indicating more effective samples.

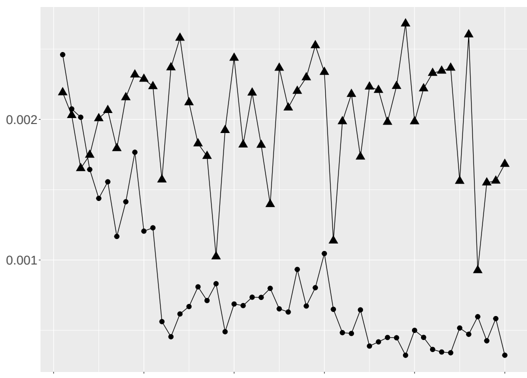
iterations each time.



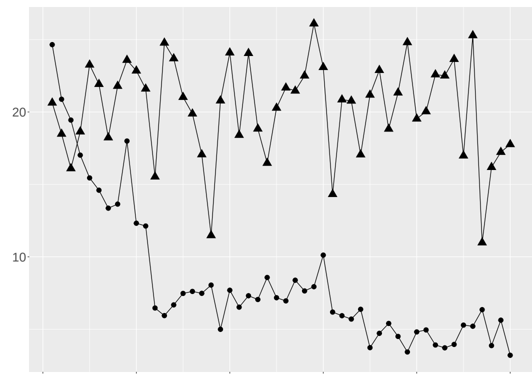
(a) Comparing α_1



(b) Comparing α_2



(c) Comparing EffUT



(d) Comparing ESSUT

Figure 5.13: Comparison of acceptance rates α_1, α_2, EffUT and ESSUT between not-updating-mean and updating-mean methods. Black solid dots $\bullet$ indicate values obtained from not-updating-mean method and black solid triangular $\blacktriangle$ indicate values obtained from updating-mean method. The acceptance rates of the updating-mean method are more stable and effective samples are larger in unit computation time.

Algorithm 5.2: Sliding Window MCMC.

1 Initialization: Set up L, *threshold* and *cut off* criteria.

2 Learning process: Estimate θ 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 θ_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 θ_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 $\mu_X^{(t)} = \frac{1}{N}\sum_i \mu_i^{(t)}$,
$\mathrm{Var}(X^{(t)}) = \frac{1}{N}\sum_i(\mu_i^{(t)}\mu_i^{(t)\top} + \Sigma_i) - \frac{1}{N^2}(\sum_i \mu_i^{(t)})(\sum_i \mu_i^{(t)})^\top$ and $\mu_X^{(t+s)}$, $\mathrm{Var}(X^{(t+s)})$ 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.

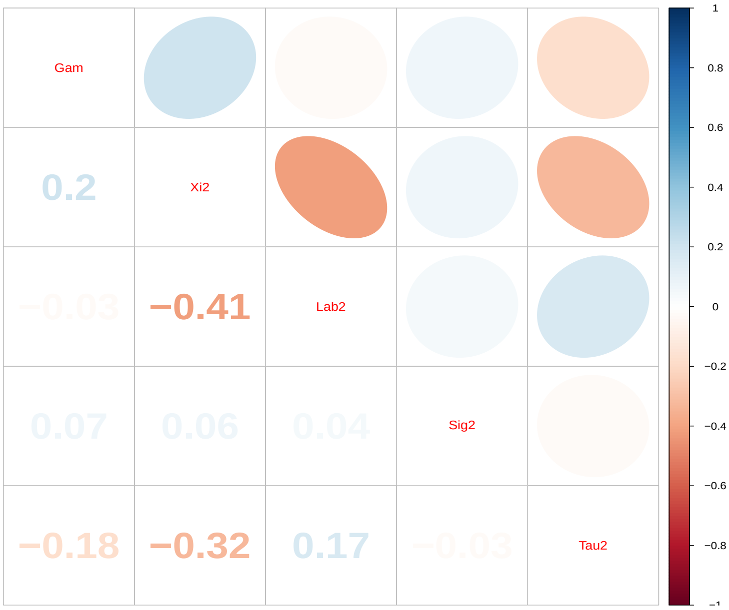
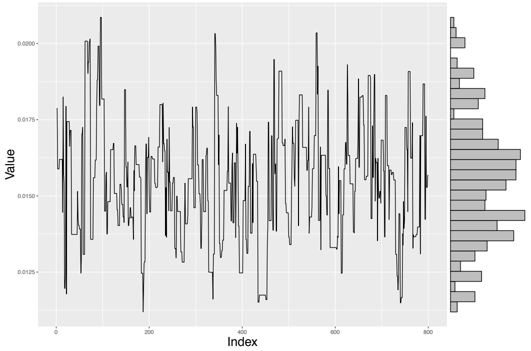
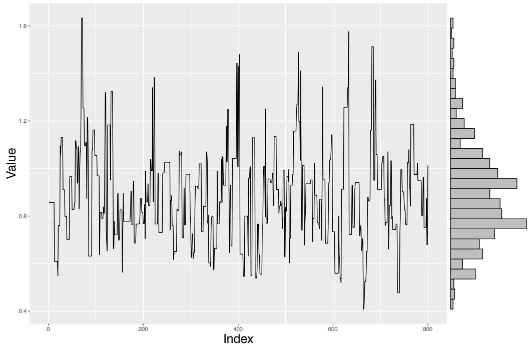


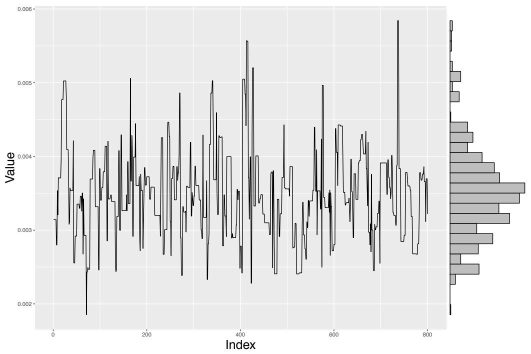
Figure 5.14: Visualization of the parameters correlation matrix, which is found in learning phase.



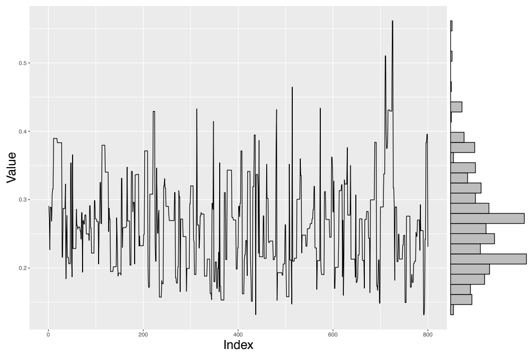
(a) Trace plot of γ



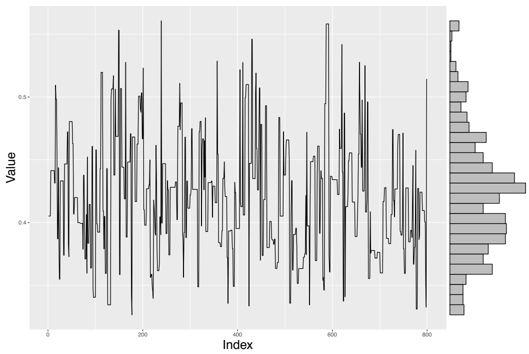
(b) Trace plot of ξ^2



(c) Trace plot of λ^2



(d) Trace plot of σ^2



(e) Trace plot of τ^2

Figure 5.15: Trace plots of θ from learning phase after taking 1 000 burn-in samples out from 5 000.

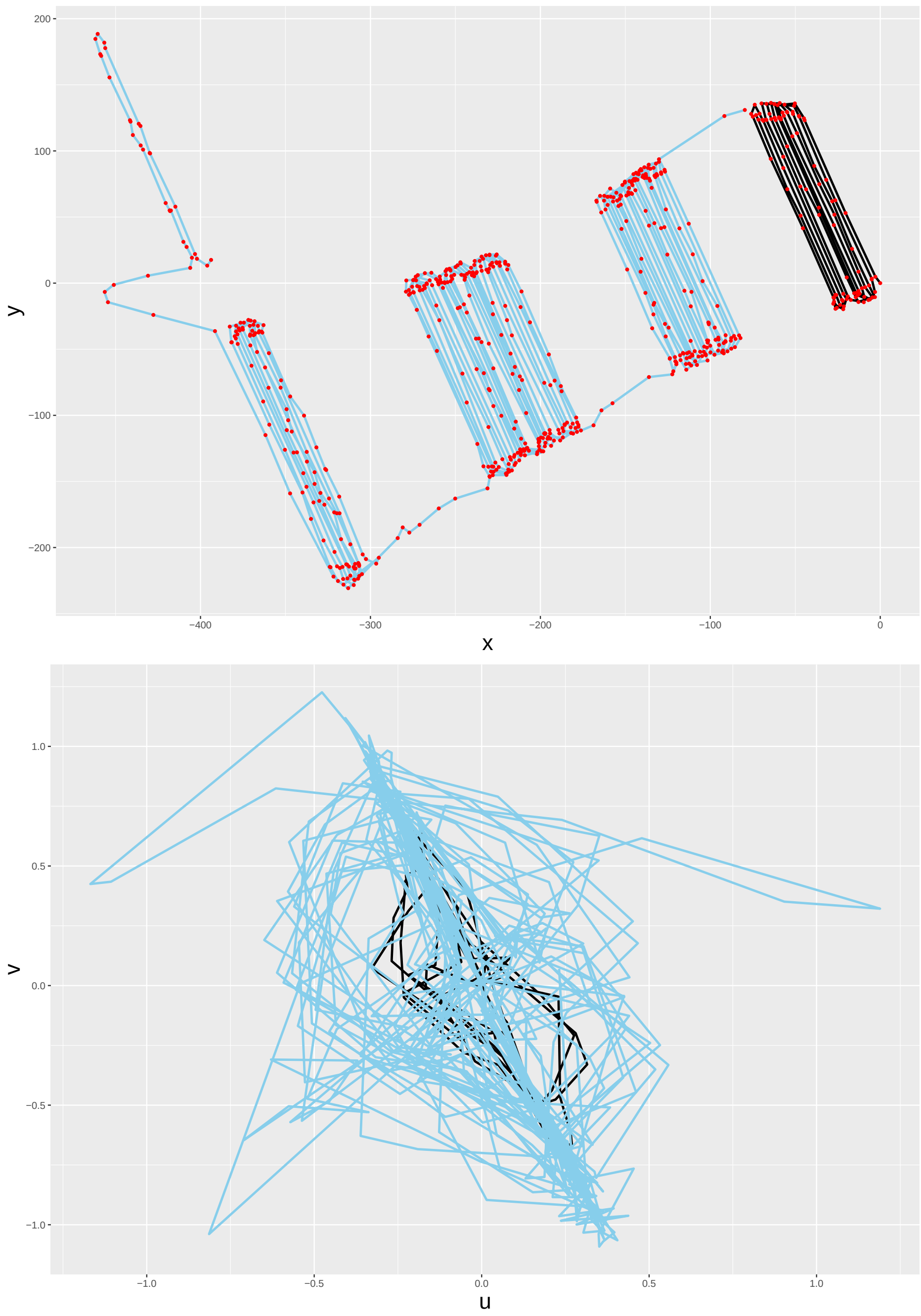


Figure 5.16: Estimations of X and Y found by combined batch and sequential methods. The black line is the estimation by batch method and the blue line is the sequential MCMC filtering estimation. Red dots are the measurements.

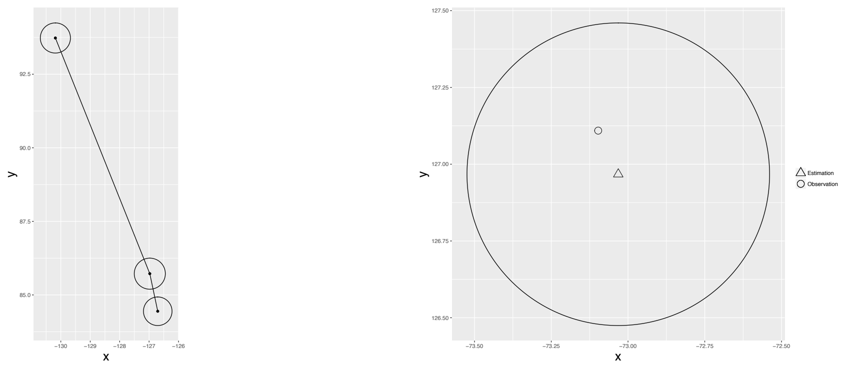


Figure 5.17: Zoom in on estimations. For each estimation $\hat{X}_i (i = 1, \ldots, t)$, there is a error circle around it.

5.6 Discussion and Future Work

In this chapter, 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

Future Work

In this thesis, two main practical algorithms are proposed: the adaptive Tractor spline and the adaptive sequential MCMC algorithm. Being honest, there are a few flaws in these two algorithms and future work remains to be done to improve their performances.

Gradient Boosting Tractor Spline

Tractor spline is an advanced smoothing spline algorithm returning least true mean squared errors. However, to implement this algorithm on-line needs feasible solutions. One of them probably is combining spline method and gradient boosting algorithm.

In machine learning application, it is best to build a non-parametric regression or classification model from the data itself. A connection between the statistical framework and machine learning is the gradient-descent based formulation of boosting methods, which was derived by Freund and Schapire (1995), Friedman (2001). The gradient boosting algorithm is a powerful machine-learning technique that has shown considerable success in a wide range of practical applications, particularly in machine learning competitions on *Kaggle*.

The motivation of gradient boosting algorithms is combining weak learners together as a strong leaner, which keeps minimizing the target loss function. It has highly customizable application to meet particular needs, like being learned with respect to different loss functions. For example, for a continuous response $y \in R$, the loss function can be chosen as a Gaussian L_2 loss function. Hence, the squared error L_2 loss function is

$$L_2(y, f(t)) = \frac{1}{2}(y - f(t))^2,$$

and the best trained f^* is

$$f^* = \arg_f \min E_{t,y} L_2(y, f(t))).$$

To find f^*, it is reducing the loss $\tilde{y}_i = y_i - F_{m-1}(t_i)$ recursively. Consequently, the

$$f_m(t) = f_{m-1}(t) + \rho_m h(t, \alpha_m),$$

is the sum of some basic learners $h_m(t, \alpha)$. $m = 1, \ldots, M$ determines the complexity of the solution.

On-line boosting algorithms are given by Babenko *et al.* (2009) and Beygelzimer *et al.* (2015). It is assumed that the loss over the entire training data can be expressed as a sum of the loss for each point t_i, that is $L(f(t, y)) = \sum_i L(f(t_i, y_i))$. Instead of computing the gradient of the entire loss, the gradient is computed with respect to just one data point. Furthermore, by adding additional regularization term will help to smooth the final learned weights to avoid over-fitting in a penalized regression problem Chen and Guestrin (2016).

Accordingly, the Tractor spline $f^*(t)$ is a sum of several weak learners $f_m(t)$, each of which has $2N$ parameters $\theta_m = \{\theta_m^{(1)}, \ldots, \theta_m^{(2N)}\}$. The optimal θ^* is found by

$$\theta^* = \sum_{m=0}^{M} \theta_m,$$

where θ_m is computed via $\theta_m = -\rho_m \frac{\partial L(\theta)}{\partial \theta}$.

As a result, after M iterations, θ^* is convergence and $f^*(t, y, \theta^*)$ is obtained.

Non-trivial Tractor Spline with Correlated Errors

In Chapter 3, the Bayes estimate of Tractor spline with correlated errors is given. The extended GCV serves to find the optimal parameters for the estimate. Whereas, the extended GCV is only applicable for trivial Tractor spline, where $\lambda(t)$ is a constant.

In Opsomer *et al.* (2001) and Wang (1998), the authors present a few extensions of the generalized maximum likelihood (GML), generalized cross-validation (GCV) and unbiased risk (UBR) methods to find optimal parameters for smoothing spline ANOVA models when observations are correlated. These algorithms in applied on conventional polynomial smoothing spine, but not on Tractor spline. Suppose the errors of the observations y and v are $\varepsilon_y \sim N(0, \sigma^2 W^{-1})$ and $\varepsilon_v \sim N(0, \tau^2 U^{-1})$ with unknown parameters σ^2 and τ^2, I hope to find general solutions, similarly to GML, GCV and UBR, to tune the parameters for Tractor spline returning least true mean squared errors.

Informative Proposals

In Chapter 5, the proposed adaptive MCMC algorithm draws samples of $\theta = \{\gamma, \xi^2, \lambda^2, \sigma^2, \tau^2\}$ from $N(m, C)$, where the information of m and C are learned from a self-tuning learning phase. For each $\theta^{(i)}$, it generates a paired mean and variance $\{\mu_t^{(i)}, \Sigma_t^{(i)}\}$ for mixture Gaussian x_t.

In the sampling step of real data application, the mechanic boom status is not incorporated, which may provide useful information. Like the penalty parameter λ being classified by λ_d and λ_u according to boom status, the MCMC sampler may use this information to propose θ with different strategies, such as different step sizes, different ms and Cs. In this way, the parameter θ is classified by θ_u and θ_d.

(not finished yet)

Grid-based MCMC

In several references, Grid-based methods have been proved that it provides an optimal recursion of the filtered density $p(x_t \mid y_{1:t})$ if the state space is discrete and consists of a finite number of states Ristic *et al.* (2004), Stroud *et al.* (2016), Arulampalam *et al.* (2002) and Hartmann *et al.* (2016).

In the application of real time series dataset, the model is supposed as an OU-process containing five unknown parameters. With the idea of grid-based algorithm, the 5-dimension parameter space $\mathbb{R}^5$ can be initialized by spanning $\theta_0^{(i)}$ with equal weights $w_0 = \{\frac{1}{N}, \frac{1}{N}, \frac{1}{N}, \frac{1}{N}, \frac{1}{N}\}$ at time $t = 0$, where $i = 1, \ldots, N$. When a new observation $Y_t = \{y_t, v_t\}$ comes into the system, the weights for each parameter in each subspace are updated via $w_t \propto p(Y_t \mid \theta, Y_{1:t-1}) w_{t-1}$.

However, the grid-based MCMC may not be practical for a higher n-dimensional space, which requires $\mathrm{O}(N^n)$ computation cost per MCMC step.

Parallel MCMC

Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously on multi-core processors Asanovic *et al.* (2006). A master process controls the strategy on how to split large, expensive computation into smaller slave processes and solved concurrently on each separate processor Almasi and Gottlieb (1994). After computing, slave processes pass results back to the master process, in which the final result is generated.

Parallel MCMC is using this technology to deploy the computation and run samplers on multi-core CPU. Approaches for parallel MCMC are either by implementing parallelization within a single chain or by running multiple chains Wu *et al.* (2012). It is useful for computing complex Bayesian models, which do not only lead to a dramatic speedup in computing but can also be used to optimize model parameters in complex Bayesian models.

A simple parallel Monte Carlo estimation of $\mathrm{E}(p(\theta))$ proceeds in the following way Kontoghiorghes (2005). Suppose there amount to k CPU cores to generate N samples. Thus, on each CPU there are $m = N/k$ samples on average. A master process passes m to each slave process i, $i = 1, \ldots, k$. At each slave process, it generates m samples for $\theta_i^{(m)}$ and passes middle-result S_i back to the master process. At last, the master process computes the final result by

$$\mathrm{E}(p(\theta)) = \frac{\sum_i S_i}{N}.$$

Therefore, parallel MCMC is using MCMC sampler scheme to draw samples on multi-cores simultaneously. A naive yet natural approach to parallel MCMC is simply to generate several independent Markov chains on different processors and then combine the results appropriately Bradford and Thomas (1996) Gelman and Rubin (1992). Or, alternatively, develop parallelism within a single chain. Suppose there are k CPU cores. Give initial values $\theta_i^{(0)}$ for each core. Concurrently update θ chains by a predetermined MCMC sampler with $p(\theta \mid y_{1:t})$ on each slave process. Computes summary statistics for the updated $\theta_i^{(0:N)}$ and passes back to the master process. Finally, achieve a sequence of θ of length kN Wu *et al.* (2012).

A further weighted parallel MCMC and parallelization approach to the Gibbs sampler is proposed by VanDerwerken and Schmidler (2013).

Consequently, the parallel MCMC is a potential alternative approach of sliding window MCMC in Chapter 5 to improve the computation speed in high dimensional space.

Chapter 7

Summary

Inference and characterization of planar trajectories have been confusing researchers for decades. Precise and efficient algorithms are highly demanded in all kinds of applications. In this thesis, an off-line method Tractor spline is proposed to reconstruct the whole trajectory and an on-line adaptive MCMC algorithm is used to update and track unknown state and parameters instantly.

In Chapter 2, the proposed Tractor spline is built up by new basis functions consisting of Hermite spline. For n paired time series data $\{t_i, y_i, v_i\}_{i=1}^n$, the amount of basis functions is $2n$. In the new objective function (2.5), Tractor spline incorporates both location and velocity information but penalizes excessive accelerations. It is not only minimizing the squared residuals of $|y_i - f(t_i)|^2$ but also reducing the squared residuals of $|v_i - f'(t_i)|^2$, for $i = 1, \ldots, n$, with a new parameter γ.

In the objective function of a conventional smoothing spline, the penalty parameter λ is one single constant that controls the trade-off between interpolations ($\lambda \to 0$) and a straight line ($\lambda \to \infty$). Instead, the penalty parameter $\lambda(t)$ of a Tractor spline is a function, which is varying at different intervals. In fact, one can consider it a piecewise constant $\lambda(t_i) = \lambda_i$ in intervals $[t_i, t_{i+1}]$ for $i = 1, \ldots, n-1$. Hence, in Tractor spline objective function, there are overall n parameters, including $n-1$ λs and γ, to be estimated. Additionally, to handle unexpected curvatures in the reconstruction, an adjusted penalty term $\frac{(\Delta t_i)^3}{(\Delta d_i)^2}$ is used to adapt to more complicated curvature status. The idea behind this term is either velocity and acceleration goes to zero, the penalty λ should be large enough to enforce a straight line.

It is proved that with improper priors, smoothing splines are corresponding to Bayes estimates, particularly, can be interpreted by Gaussian process regression in a reproducing kernel Hilbert space. This interesting property can be used in more flexible and general applications. Similarly, if the Tractor spline is equipped with an appropriate inner product (3.13) and keep λ constant, it is corresponding to the posterior mean of the Bayes estimates in a particular reproducing kernel Hilbert space $\mathcal{C}^2_{p.w.}[0, 1]$, in which the second derivatives are piecewise continuous. This result is discussed in Chapter 3. Recall the property of Tractor splines that only if $\lambda(t)$ is constant and $\gamma = 0$ would the second derivatives be continuous in the entire interval. Otherwise, the second derivatives may not be continuous at joint knots but are linear in each interval. That is piecewise continuous. Therefore, this kind of Tractor splines is named trivial Tractor splines. In contrast, a non-trivial Tractor spline has a penalty function $\lambda(t)$.

Furthermore, generic smoothing splines are always involving independent errors. Extended research is considering using correlated errors or correlated observations, that are more common in real life applications. In Chapter 3, it is proved by numerical simulation that the Tractor spline and its Bayes estimate return the same results even if the errors in observed y and v are correlated.

To find the best parameters, an extended leave-one-out cross-validation technique is proposed to find all the smoothing parameters of interest in Chapter 2. This method is using observed data itself to tune the parameters at the optimal level. Accordingly, Tractor spline is a data-driven nonparametric regression solution to handle paired time series data consisting position and velocity information. However, for data with correlated errors, the generalized cross-validation algorithm is more effective. Being modified a generic GCV, in Chapter 3, an extended GCV is used for finding the optimal parameters for Tractor spline and its Bayes estimate containing correlated errors. Suppose the solution of a Tractor spline and its first derivative are in the form of $f = S(\lambda, \gamma)y + \gamma T(\lambda, \gamma)v$ and $f' = U(\lambda, \gamma)y + \gamma V(\lambda, \gamma)v$, the GCV is calculating the trace of matrices S, T, U and V. It is much faster than calculating the sum of the single element in each matrix in leave-one-out CV.

Simulation studies are given to compare the performances of Tractor spline and other methods, such as Wavelet algorithms and penalized B-spline, in Chapter 2. It is obvious that all algorithms are very competitive on reconstructing trajectories with respect to mean squared errors, but only the proposed Tractor spline returns the least true mean squared errors. That means the Tractor spline performs better and closer to the true singles. At the end of that Chapter, a real data example are presented to demonstrate the effectiveness of Tractor spline. Being applied to a real GPS dataset, the parameter λ is classified by λ_u and λ_d representing for two operating status of a mechanic boom. λ_u is a set of $\{\lambda_i\}$ in the intervals where the boom is not-operating and, in contrast, λ_d is a set of $\{\lambda_i\}$ in the intervals where the boom is operating. The reconstruction from Tractor spline can be treated as the real trajectory of a moving vehicle with confidence.

Without loss of generality, $\lambda(t)$ can be classified into more groups to adapt to complex maneuver system and Tractor spline is flexible to be applied on higher dimensional cases.

However, subject to the property that smoothing splines require the solution of a global problem that involves the entire set of points to be interpolated, it might not be suitable for on-line estimation. Then Chapter 4 is the content of a brief overview of existing filtering and estimation algorithms. Some popular algorithms, such as Particle filter, are concentrating on inferring the unknown state but assuming the parameters are known. Moreover, the sample impoverishment has never been solved properly. Liu and West's filter is trying to kill particle degeneracy by incorporating with a shrinkage kernel. Meanwhile, it estimates the unknown parameters simultaneously. Storvik filter and Particle learning algorithms are marginalizing out the parameters through sufficient statistics and achieving a better estimation than Liu and West's filter. In some way, they are advanced algorithms but not practicable any time. In most situations, sufficient statistics are not available or hard to find. A more flexible and easy-implement method is in demand.

As a result, an adaptive sequential MCMC algorithm is proposed in Chapter 5. Similarly, with Tractor spline, the adaptive sequential MCMC is dealing with paired time series dataset including both position and velocity information.

In the case of a linear state-space model and starting with a joint distribution over state x, observation y and parameter θ, an MCMC sampler is implemented with two phases. In the learning phase, a self-tuning sampler is utilizing one-variable-at-a-time random walk Metropolis-Hastings (MH) algorithm to learn the parameter mean and covariance structure by aiming at a target acceptance rate. After exploring the parameter space, the information is used in the subsequent phase — the estimation phase — to inform the proposed mechanism and is also used in a delayed-acceptance algorithm.

Suppose the mean and covariance matrix of θ are m and $C = L^\top L$ respectively, where L is its Cholesky decomposition. Then the proposal $\theta^* = \theta + \epsilon LZ$ and $Z \sim N(0, I)$, ϵ is the step size. The effect of L is to reduce the correlation of proposals and move to the next step on purpose. The step size ϵ makes the proposal process more efficient. Rather than focusing on the criteria of efficiency (Eff) and effective sample size (ESS), the Eff in unit time (EffUT) and ESS in unit time (ESSUT) should be the new criteria to determine the optimal step size. By running the same amount of time, the optimal step size found by EffUT and ESSUT generates more effective and representative samples.

Further, the delayed-acceptance algorithm is using a cheap surrogate $p(\theta \mid m, C)$ to the true posterior $p(\theta \mid y)$ in the first line of defense to keep not-good samples outside. Only good proposals would pass the first line and move forward to the additional expensive calculation. This strategy greatly improves sampling efficiency.

Information on the resulting state of the system is indicated by a Gaussian mixture. For each sample of $\theta^{(i)}$, it matches an $x \sim N(\mu^{(i)}, \sigma^{(i)})$. Consequently, the final estimation of x is given by a set of the Gaussian mixture.

In the on-line mode, the algorithm is adaptive and uses a sliding window approach by cutting off historical data to accelerate sampling speed and to maintain appropriate acceptance. In a simple one parameter simulation in Chapter 4, the proposed adaptive MCMC shows a stable feature comparing with other filtering methods. And in Chapter 5, this algorithm shows an advantage in estimating irregularly sampled time series data.

At the end of Chapter 5, the proposed algorithm is applied to joint state and parameters estimation in the case of irregularly sampled GPS time series data. Suppose it is a four-dimensional linear Ornstein-Uhlenbeck (OU) process containing $z = \{x, u, y, v\}$ with respects to $\{x, u\}$ on X-axis and $\{y, v\}$ on Y-axis, the parameter space is in 5 dimensions, that is $\theta = \{\gamma, \xi^2, \lambda^2, \sigma^2, \tau^2\}$. The proposed algorithm efficiently infers the state z_t through time t within acceptable errors. Not limited, it also predicts the upcoming states simultaneously.

As a conclusion, the proposed algorithms in this thesis are contributing to related areas, nevertheless, are not perfect. The Tractor spline may not be appropriate for on-line estimation and the sliding window adaptive MCMC algorithm is not using the entire data set that might lose some information. Future work and deeper research are required to improve their performances.

Appendices

Appendix A

Proofs And Figures of Tractor Spline Theorems

A.1 Penalty Matrix in (2.12)

The i-th $\Omega^{(i)}$ is a $2n \times 2n$ bandwidth four symmetric matrix and its non-zero elements on the upper triangular are

$$\Omega^{(i)}_{2i-1,2i-1} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{00}(t)}{dt^2} \frac{d^2 h^{(i)}_{00}(t)}{dt^2} dt = \frac{12}{\Delta_i^3} \tag{A.1.1}$$

$$\Omega^{(i)}_{2i-1,2i} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{00}(t)}{dt^2} \frac{d^2 h^{(i)}_{10}(t)}{dt^2} dt = \frac{6}{\Delta_i^2} \tag{A.1.2}$$

$$\Omega^{(i)}_{2i-1,2i+1} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{00}(t)}{dt^2} \frac{d^2 h^{(i)}_{01}(t)}{dt^2} dt = \frac{-12}{\Delta_i^3} \tag{A.1.3}$$

$$\Omega^{(i)}_{2i-1,2i+2} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{00}(t)}{dt^2} \frac{d^2 h^{(i)}_{11}(t)}{dt^2} dt = \frac{6}{\Delta_i^2} \tag{A.1.4}$$

$$\Omega^{(i)}_{2i,2i} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{10}(t)}{dt^2} \frac{d^2 h^{(i)}_{10}(t)}{dt^2} dt = \frac{4}{\Delta_i} \tag{A.1.5}$$

$$\Omega^{(i)}_{2i,2i+1} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{10}(t)}{dt^2} \frac{d^2 h^{(i)}_{01}(t)}{dt^2} dt = \frac{-6}{\Delta_i^2} \tag{A.1.6}$$

$$\Omega^{(i)}_{2i,2i+2} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{10}(t)}{dt^2} \frac{d^2 h^{(i)}_{11}(t)}{dt^2} dt = \frac{2}{\Delta_i} \tag{A.1.7}$$

$$\Omega^{(i)}_{2i+1,2i+1} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{01}(t)}{dt^2} \frac{d^2 h^{(i)}_{01}(t)}{dt^2} dt = \frac{12}{\Delta_i^3} \tag{A.1.8}$$

$$\Omega^{(i)}_{2i+1,2i+2} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{01}(t)}{dt^2} \frac{d^2 h^{(i)}_{11}(t)}{dt^2} dt = \frac{-6}{\Delta_i^2} \tag{A.1.9}$$

$$\Omega^{(i)}_{2i+2,2i+2} = \int_{t_i}^{t_{i+1}} \frac{d^2 h^{(i)}_{11}(t)}{dt^2} \frac{d^2 h^{(i)}_{11}(t)}{dt^2} dt = \frac{4}{\Delta_i} \tag{A.1.10}$$

where $\Delta_i = t_{i+1} - t_i$ and $i = 1, 2, \ldots, n-1$. Then

$$\mathbf{\Omega} = \sum_{i=1}^{n-1} \lambda_i \Omega^{(i)}$$

A.2 Proof of Theorem 1

Proof. If $g : [a, b] \to \mathbb{R}$ is a proposed minimizer, construct a cubic spline $f(t)$ that agrees with $g(t)$ and its first derivatives at $t_1, \ldots, t_n$, and is component-wise linear on $[a, t_1]$ and $[t_n, b]$. Let $h(t) = g(t) - f(t)$. Then, for $i = 1, \ldots, n-1$,

$$\begin{aligned}
\int_{t_i}^{t_{i+1}} f''(t)h''(t)dt &= f''(t)h'(t)\Big|_{t_i}^{t_{i+1}} - \int_{t_i}^{t_{i+1}} f'''(t)h'(t)dt \\
&= 0 - f'''(t_i^+) \int_{t_i}^{t_{i+1}} h'(t)dt \\
&= -f'''(t_i^+) \left(h(t_{i+1}) - h(t_i)\right) \\
&= 0.
\end{aligned}$$

Additionally, $\int_a^{t_1} f''(t)h''(t)dt = \int_{t_n}^{b} f''(t)h''(t)dt = 0$, since $f(t)$ is assumed linear outside the knots. Thus, for $i = 0, \ldots, n$,

$$\begin{aligned}
\int_{t_i}^{t_{i+1}} |g''(t)|^2 dt &= \int_{t_i}^{t_{i+1}} |f''(t) + h''(t)|^2 dt \\
&= \int_{t_i}^{t_{i+1}} |f''(t)|^2 dt + 2\int_{t_i}^{t_{i+1}} f''(t)h''(t)dt + \int_{t_i}^{t_{i+1}} |h''(t)|^2 dt \\
&= \int_{t_i}^{t_{i+1}} |f''(t)|^2 dt + \int_{t_i}^{t_{i+1}} |h''(t)|^2 dt \\
&\geq \int_{t_i}^{t_{i+1}} |f''(t)|^2 dt.
\end{aligned}$$

The result $J[f] \leq J[g]$ follows since $\lambda_i > 0$.

Furthermore, equality of the curvature penalty term only holds if $g(t) = f(t)$. On $[t_1, t_n]$, we require $h''(t) = 0$ but since $h(ti) = h'(ti) = 0$ for $i = 1, \ldots, n$, this means $h(t) = 0$. Meanwhile on $[a, t_1]$ and $[t_n, b]$, $f''(t) = 0$ so that equality requires $g''(t) = 0$. Since $f(t)$ agrees with $g(t)$ and its first derivatives at t_1 and t_n, equality is forced on both intervals. □

A.3 Proof of Theorem 2

To prove Theorem 2, we are introducing a Lemma Peng (1983) in the first place.

Lemma A.3.1. *Functions $N_1(t), N_2(t), \ldots, N_n(t)$ on interval $[a,b]$, if they are linear dependent, the necessary and sufficient condition is for any $c_1, c_2, \ldots, c_n \in [a,b]$, the determinant $D(c_1, c_2, \ldots, c_n) = 0$; if they are linear independent, the necessary and sufficient condition is that there exist $c_1, c_2, \ldots, c_n \in [a,b]$, so that the determinant $D(c_1, c_2, \cdots, c_n) \neq 0$, where*

$$D(c_1, c_2, \ldots, c_n) = \begin{vmatrix} N_1(c_1) & N_1(c_2) & \cdots & N_1(c_n) \\ N_2(c_1) & N_2(c_2) & \cdots & N_2(c_n) \\ \vdots & \vdots & \ddots & \vdots \\ N_n(c_1) & N_n(c_2) & \cdots & N_n(c_n) \end{vmatrix}$$

In the following, we are proving Theorem 2 with the help of Lemma A.3.1.

Proof. It is obvious that these basis functions are continuous on subinterval $[t_i, t_{i+1}]$, where $i = 1, \ldots, n-1$. We will prove that these basis functions are independent at first.

Given $2n$ basis functions and n knots, we 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

$$D(c_1, c_2, \cdots, c_{2n}) = \begin{bmatrix} N_1(c_1) & N_1(c_2) & \cdots & N_1(c_{2n}) \\ N_2(c_1) & N_2(c_2) & \cdots & N_2(c_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ N_{2n}(c_1) & N_{2n}(c_2) & \cdots & N_{2n}(c_{2n}) \end{bmatrix} \tag{A.3.1}$$

where the non-zero elements of matrix D are

$$\begin{array}{ccc} N_1(t_1) = 1 & N_1\left(\frac{t_1+t_2}{2}\right) = a_{12} & N_2(t_1) = 0 \\ N_2\left(\frac{t_1+t_2}{2}\right) = a_{22} & N_{2i+1}\left(\frac{t_i+t_{i+1}}{2}\right) = a_{2i+1,2i} & N_{2i+1}(t_{i+1}) = 1 \\ N_{2i+1}\left(\frac{t_{i+1}+t_{i+2}}{2}\right) = a_{2i+1,2i+2} & N_{2i+2}\left(\frac{t_i+t_{i+1}}{2}\right) = a_{2i+2,2i} & N_{2i+2}\left(\frac{t_{i+1}+t_{i+2}}{2}\right) = a_{2i+2,2i+2} \\ N_{2n-1}(t_{2n-1}) = 0 & N_{2n-1}\left(\frac{t_{2n-1}+t_{2n}}{3}\right) = a_{2n-1,2n-2} & N_{2n-1}\left(\frac{2(t_{2n-1}+t_{2n})}{3}\right) = a_{2n-1,2n-1} \\ N_{2n-1}(t_{2n}) = 1 & N_{2n}(\frac{t_{2n-1}+t_{2n}}{3}) = a_{2n,2n-2} & N_{2n}\left(\frac{2(t_{2n-1}+t_{2n})}{3}\right) = a_{2n,2n-1} \\ N_{2n}(t_{2n}) = 0 & & \end{array}$$

where $i = 1, \ldots, n-1$. By decomposing determinant D in the following way, it will give us

$$\det D = \begin{vmatrix} 1 & a_{12} & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_{22} & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_{32} & 1 & a_{34} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{2n-1,2n-1} & 1 \\ 0 & 0 & 0 & 0 & \cdots & a_{2n,2n-1} & 0 \end{vmatrix} = \begin{vmatrix} a_{22} & 0 & 0 & \cdots & 0 & 0 \\ a_{32} & 1 & a_{34} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{2n-1,2n-1} & 1 \\ 0 & 0 & 0 & \cdots & a_{2n,2n-1} & 0 \end{vmatrix}$$

$$= a_{22} \begin{vmatrix} 1 & a_{34} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{2n-1,2n-1} & 1 \\ 0 & 0 & \cdots & a_{2n,2n-1} & 0 \end{vmatrix} \cdots = a_{22}a_{44}\cdots a_{2n-4,2n-4} \begin{vmatrix} a_{2n-2,2n-2} & a_{2n-2,2n-1} & 0 \\ a_{2n-1,2n-2} & a_{2n-1,2n-1} & 1 \\ a_{2n,2n-2} & a_{2n,2n-1} & 0 \end{vmatrix}$$

$$= a_{22}a_{44}\cdots a_{2n-4,2n-4}(a_{2n-2,2n-1}a_{2n,2n-2} - a_{2n,2n-1}a_{2n-2,2n-2}) \neq 0.$$

With the conclusion of Lemma A.3.1, $N_1(t), \ldots, N_{2n}(t)$ are linearly independent on the entire interval $[t_1, t_n]$.

Secondly, we will prove that basis functions represent any cubic function on each interval $[t_i, t_{i+1}]$. Due to the definition of cubic spline, on interval $[t_i, t_{i+1}]$, a cubic spline $g(t)$ can be written in the form of

$$g(t) = d_i(t-t_i)^3 + c_i(t-t_i)^2 + b_i(t-t_i) + a_i, \text{for } t_i \le t \le t_{i+1} \tag{A.3.2}$$

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 $\forall t \in [t_i, t_{i+1}]$, $i = 1, \ldots, n-1$, we have

$$f(t) = \begin{cases} \theta_{2i-1}N_{2i-1}(t) + \theta_{2i}N_{2i}(t) + \theta_{2i+1}N_{2i+1}(t) + \theta_{2i+2}N_{2i+3}(t), & t_i \le t \le t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

Hence,

$$\begin{aligned} f(t) =& \theta_{2i-1}\left(2\left(\frac{t-t_i}{t_{i+1}-t_i}\right)^3 - 3\left(\frac{t-t_i}{t_{i+1}-t_i}\right)^2 + 1\right) + \theta_{2i}\left(\frac{(t-t_i)^3}{(t_{i+1}-t_i)^2} - 2\frac{(t-t_i)^2}{t_{i+1}-t_i} + (t-t_i)\right) \\ &+ \theta_{2i+1}\left(-2\left(\frac{t-t_i}{t_{i+1}-t_i}\right)^3 + 3\left(\frac{t-t_i}{t_{i+1}-t_i}\right)^2\right) + \theta_{2i+2}\left(\frac{(t-t_i)^3}{(t_{i+1}-t_i)^2} - \frac{(t-t_i)^2}{t_{i+1}-t_i}\right). \end{aligned}$$

After rearranging, we have

$$\begin{aligned} f(t) =& \left(\frac{2\theta_{2i-1}}{(t_{i+1}-t_i)^3} + \frac{\theta_{2i}}{(t_{i+1}-t_i)^2} - \frac{2\theta_{2i+1}}{(t_{i+1}-t_i)^3} + \frac{\theta_{2i+2}}{(t_{i+1}-t_i)^2}\right)(t-t_i)^3 \\ &+ \left(-\frac{3\theta_{2i-1}}{(t_{i+1}-t_i)^2} - \frac{2\theta_{2i}}{(t_{i+1}-t_i)} + \frac{3\theta_{2i+1}}{(t_{i+1}-t_i)^2} - \frac{\theta_{2i+2}}{(t_{i+1}-t_i)}\right)(t-t_i)^2 \\ &+ \theta_{2i}(t-t_i) + \theta_{2i-1}, \end{aligned}$$

where coefficients are

$$\begin{cases} \theta_{2i-1} = a_i \\ \theta_{2i} = b_i \\ -\frac{3\theta_{2i-1}}{(t_{i+1}-t_i)^2} - \frac{2\theta_{2i}}{(t_{i+1}-t_i)} + \frac{3\theta_{2i+1}}{(t_{i+1}-t_i)^2} - \frac{\theta_{2i+2}}{(t_{i+1}-t_i)} = c_i \\ \frac{2\theta_{2i-1}}{(t_{i+1}-t_i)^3} + \frac{\theta_{2i}}{(t_{i+1}-t_i)^2} - \frac{2\theta_{2i+1}}{(t_{i+1}-t_i)^3} + \frac{\theta_{2i+2}}{(t_{i+1}-t_i)^2} = d_i \end{cases}$$

The results can always be solved for $\theta_{2i-1}, \theta_{2i}, \theta_{2i+1}, \theta_{2i+2}$ in terms of a_i, b_i, c_i, d_i on interval $[t_i, t_{i+1}]$. 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_i, where $t_1 < t_i < t_n$, it is known that $f(t_i) = \theta_{2i-1}$. Moreover,

$$\lim_{t \to t_i+} f(t) = \lim_{t \to t_i+} (\theta_{2i-1}N_{2i-1}(t) + \theta_{2i}N_{2i}(t) + \theta_{2i+1}N_{2i+1}(t) + \theta_{2i+2}N_{2i+3}(t)) = \theta_{2i-1},$$
$$\lim_{t \to t_i-} f(t) = \lim_{t \to t_i-} (\theta_{2i-1}N_{2i-1}(t) + \theta_{2i}N_{2i}(t) + \theta_{2i+1}N_{2i+1}(t) + \theta_{2i+2}N_{2i+3}(t)) = \theta_{2i-1}.$$

As a result,

$$\lim_{t \to t_i^+} f(t) = \lim_{t \to t_i^-} f(t) = f(t),$$

$f(t)$ is continuous at joint knots, and then continuous on whole interval $[t_1, t_n]$.

Consequently, $f(t)$ is a continuous cubic spline represented by basis functions $N_1(t), \ldots, N_{2n}(t)$ on the entire interval $[t_1, t_n]$. □

A.4 Proof of Corollary 1

Proof. By setting $\gamma \to 0$, the velocity information v is taken away. The degrees of freedom of parameters decreases from $2n$ to n. Hence, there exists an $n \times 2n$ matrix Q_λ restricting n degrees of freedom of $\hat{\theta}$ and satisfying $Q_\lambda \hat{\theta} = 0$.

It is easily seen that the matrices B and C have the following property:

$$BB^\top = CC^\top = I_n,$$
$$C^\top CB^\top = B^\top BC^\top = 0.$$

Denoting $G = B^\top B + \gamma C^\top C + n\Omega_\lambda$ and giving $\hat{\theta} = (B^\top B + \gamma C^\top C + n\Omega_\lambda)^{-1}(B^\top y + \gamma C^\top v)$, we will have $G\hat{\theta} = B^\top y + \gamma C^\top v$ and

$$BG\hat{\theta} = y + \gamma BC^\top v$$
$$CG\hat{\theta} = CB^\top y + \gamma v.$$

Therefore, $C^\top CG\hat{\theta} = C^\top(CB^\top y + \gamma v) = \gamma C^\top v$. If by setting $\gamma = 0$, then we will have $Q_\lambda = C^\top CG$, which consists of the even rows of Ω_λ.

By integrating by parts and using properties of the basis functions at the knots, one can get the even rows of $-\Omega^{(i)}$, which are

$$\begin{aligned}
-\Omega^{(i)}_{2i,2i-1} &= N''_{2i-1}(t_i^+) - N''_{2i-1}(t_i^-) \\
-\Omega^{(i)}_{2i,2i} &= N''_{2i}(t_i^+) - N''_{2i}(t_i^-) \\
-\Omega^{(i)}_{2i,2i+1} &= N''_{2i+1}(t_i^+) - N''_{2i+1}(t_i^-) \\
-\Omega^{(i)}_{2i,2i+2} &= N''_{2i+2}(t_i^+) - N''_{2i+2}(t_i^-) \\
-\Omega^{(i)}_{2i+2,2i-1} &= N''_{2i-1}(t_{i+1}^+) - N''_{2i-1}(t_{i+1}^-) \\
-\Omega^{(i)}_{2i+2,2i} &= N''_{2i}(t_{i+1}^+) - N''_{2i}(t_{i+1}^-) \\
-\Omega^{(i)}_{2i+2,2i+1} &= N''_{2i+1}(t_{i+1}^+) - N''_{2i+1}(t_{i+1}^-) \\
-\Omega^{(i)}_{2i+2,2i+2} &= N''_{2i+2}(t_{i+1}^+) - N''_{2i+2}(t_i^-)
\end{aligned}$$

Thus

$$Q_\lambda = nC^\top C\Omega_\lambda = nC^\top C\sum_i \lambda_i\Omega^{(i)}.$$

Consequently, if and only if λ is constant, $Q_\lambda\hat{\theta} = -\lambda\left(f''(t_i^+) - f''(t_i^-)\right) = 0$, for $i = 1, \ldots, n$, otherwise $Q_\lambda\theta = 0$ is true but does not represent $f''(t_i^+) - f''(t_i^-)$.

As a result, $f''(t)$ is continuous at knots t_i if $\lambda(t)$ is constant and $\gamma = 0$.

□

A.5 Proof of Lemma 2

Proof. For any smooth curve f with $\mathbf{y}^*$, we have

$$\begin{aligned}
&\frac{1}{n}\sum_{j=1}^{n}\left(y_j^* - f(t_j)\right)^2 + \frac{\gamma}{n}\sum_{j=1}^{n}\left(v_j^* - f'(t_j)\right)^2 + \sum_{j=1}^{n}\lambda_j\int_{t_j}^{t_{j+1}} f''^2 dt \\
\geq&\frac{1}{n}\sum_{j\neq i}\left(y_j^* - f(t_j)\right)^2 + \frac{\gamma}{n}\sum_{j\neq i}\left(v_j^* - f'(t_j)\right)^2 + \sum_{j=1}^{n}\lambda_j\int_{t_j}^{t_{j+1}} f''^2 dt \\
\geq&\frac{1}{n}\sum_{j\neq i}\left(y_j^* - \hat{f}^{(-i)}(t_j)\right)^2 + \frac{\gamma}{n}\sum_{j\neq i}\left(v_j^* - \hat{f}'^{(-i)}(t_j)\right)^2 + \sum_{j=1}^{n}\lambda_j\int_{t_j}^{t_{j+1}} \left(\hat{f}''^{(-i)}\right)^2 dt \\
=&\frac{1}{n}\sum_{j=1}^{n}\left(y_j^* - \hat{f}^{(-i)}(t_j)\right)^2 + \frac{\gamma}{n}\sum_{j=1}^{n}\left(v_j^* - \hat{f}'^{(-i)}(t_j)\right)^2 + \sum_{j=1}^{n}\lambda_j\int_{t_j}^{t_{j+1}} \left(\hat{f}''^{(-i)}\right)^2 dt
\end{aligned}$$

by the definition of $\hat{\mathbf{f}}^{(-i)}$, $\hat{\mathbf{f}}'^{(-i)}$ and the fact that $y_i^* = \hat{f}^{(-i)}(t_i)$, $v_i^* = \hat{f}'^{(-i)}(t_i)$. It follows that $\hat{f}^{(-i)}$ is the minimizer of the objective function (2.5), so that

$$\begin{aligned}\hat{\mathbf{f}}^{(-i)} &= S\mathbf{y}^* + \gamma T\mathbf{v}^* \\ \hat{\mathbf{f}}'^{(-i)} &= U\mathbf{y}^* + \gamma V\mathbf{v}^*\end{aligned}$$

as required. □

A.6 Proof of Theorem 3

Proof.

$$\begin{aligned}\hat{f}^{(-i)}(t_i) - y_i =& \sum_{j=1}^{n} S_{ij}y_j^* + \gamma\sum_{j=1}^{n} T_{ij}v_j^* - y_i^* \\ =& \sum_{j\neq i}^{n} S_{ij}y_j + \gamma\sum_{j\neq i}^{n} T_{ij}v_j + S_{ii}\hat{f}^{(-i)}(t_i) + \gamma T_{ii}\hat{f}'^{(-i)}(t_i) - y_i \\ =& \sum_{j=1}^{n} S_{ij}y_j + \gamma\sum_{j=1}^{n} T_{ij}v_j + S_{ii}(\hat{f}^{(-i)}(t_i) - y_i) + \gamma T_{ii}(\hat{f}'^{(-i)}(t_i) - v_i) - y_i \\ =& (\hat{f}(t_i) - y_i) + S_{ii}(\hat{f}^{(-i)}(t_i) - y_i) + \gamma T_{ii}(\hat{f}'^{(-i)}(t_i) - v_i).\end{aligned} \tag{A.6.1}$$

Additionally,

$$\begin{aligned}\hat{f}'^{(-i)}(t_i) - v_i =& \sum_{j=1}^{n} U_{ij}y_j^* + \gamma\sum_{j=1}^{n} V_{ij}v_j^* - v_i^* \\ =& \sum_{j\neq i}^{n} U_{ij}y_j + \gamma\sum_{j\neq i}^{n} V_{ij}v_j + U_{ii}\hat{f}^{(-i)}(t_i) + \gamma V_{ii}\hat{f}'^{(-i)}(t_i) - v_i \\ =& \sum_{j=1}^{n} U_{ij}y_j + \gamma\sum_{j=1}^{n} V_{ij}v_j + U_{ii}(\hat{f}^{(-i)}(t_i) - y_i) + \gamma V_{ii}(\hat{f}'^{(-i)}(t_i) - v_i) - v_i \\ =& (\hat{f}'(t_i) - v_i) + U_{ii}(\hat{f}^{(-i)}(t_i) - y_i) + \gamma V_{ii}(\hat{f}'^{(-i)}(t_i) - v_i).\end{aligned} \tag{A.6.2}$$

Thus

$$\hat{f}'^{(-i)}(t_i) - v_i = \frac{\hat{f}'(t_i) - v_i}{1 - \gamma V_{ii}} + \frac{U_{ii}(\hat{f}^{(-i)}(t_i) - y_i)}{1 - \gamma V_{ii}}. \tag{A.6.3}$$

By substituting equation (A.6.3) to (A.6.1), we get

$$\hat{f}^{(-i)}(t_i) - y_i = \frac{\hat{f}(t_i) - y_i + \gamma\frac{T_{ii}}{1-\gamma V_{ii}}(\hat{f}'(t_i) - v_i)}{1 - S_{ii} - \gamma\frac{T_{ii}}{1-\gamma V_{ii}}U_{ii}}.$$

Consequently,

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

□

A.7 Reconstructions at SNR=3

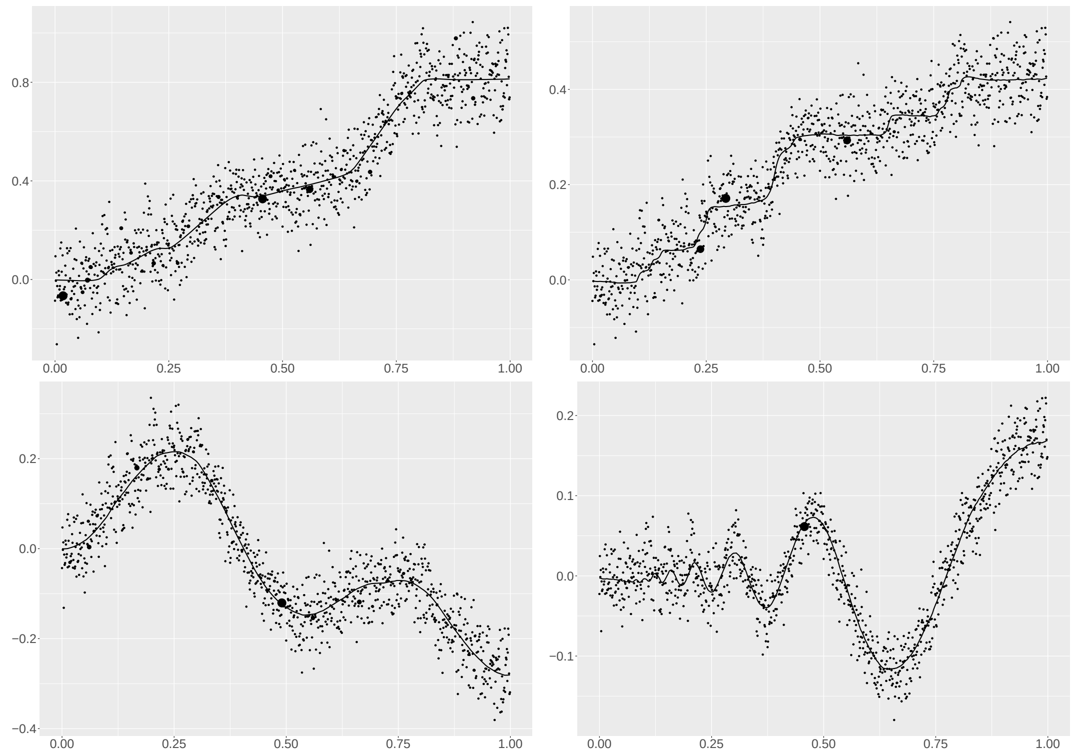
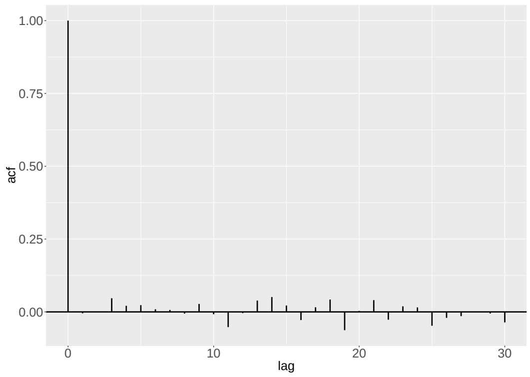
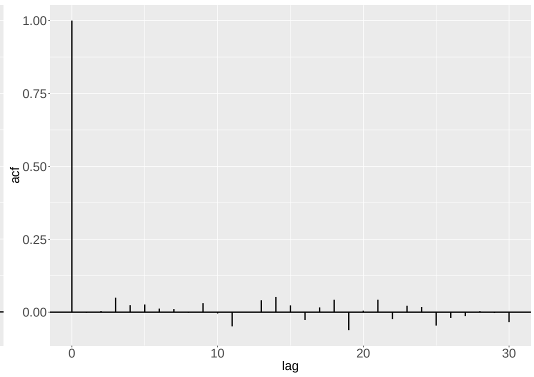


Figure A.1: Reconstructions of generated *Blocks*, *Bumps*, *HeaviSine* and *Doppler* functions by Tractor spline at SNR=3. The penalty values $\lambda(t)$ in Tractor spline are projected into reconstructions. The blacks dots are the measurements. The bigger blacks dots indicate the larger penalty values.

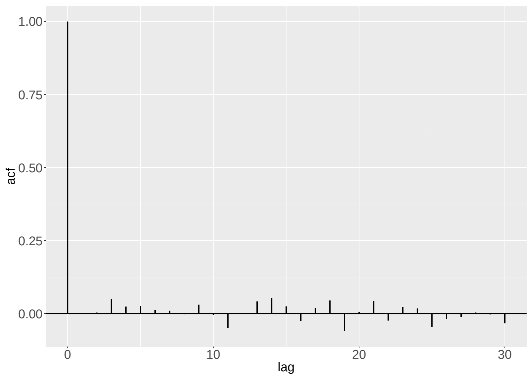
A.8 Residual Analysis of Simulations



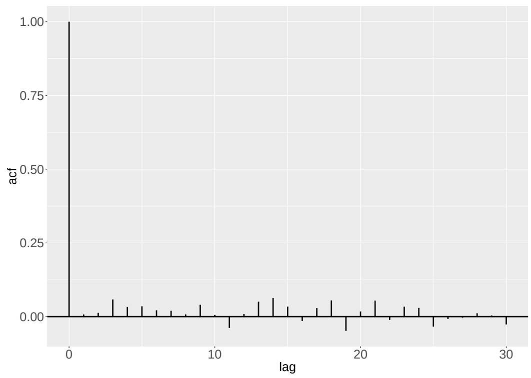
(a) ACF of residuals from *Blocks* with SNR at 7



(b) ACF of residuals from *Bumps* with SNR at 7

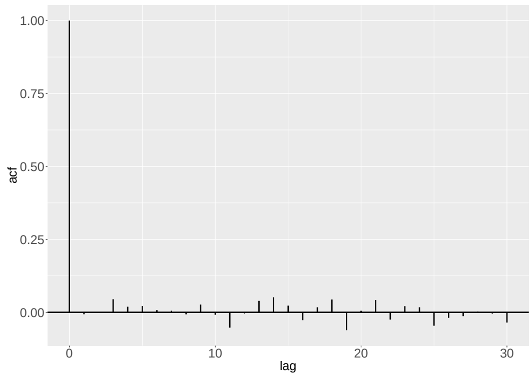


(c) ACF of residuals from *HeaviSine* with SNR at 7

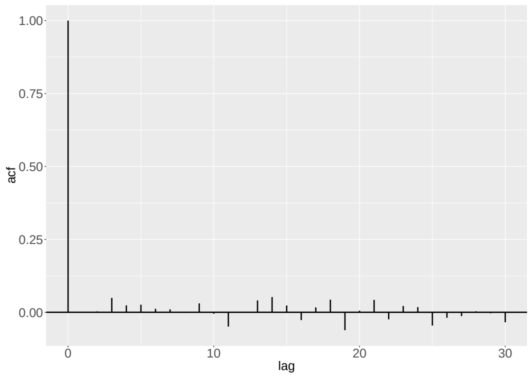


(d) ACF of residuals from *Doppler* with SNR at 7

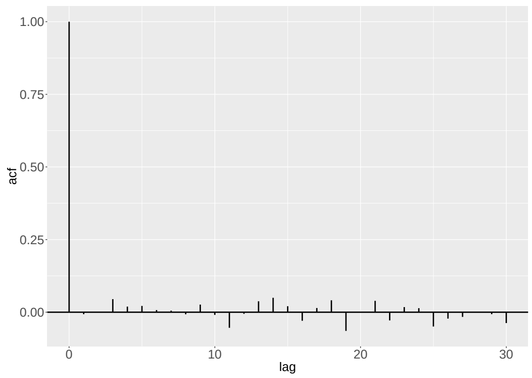
Figure A.2: ACF of residuals at SNR level of 7.



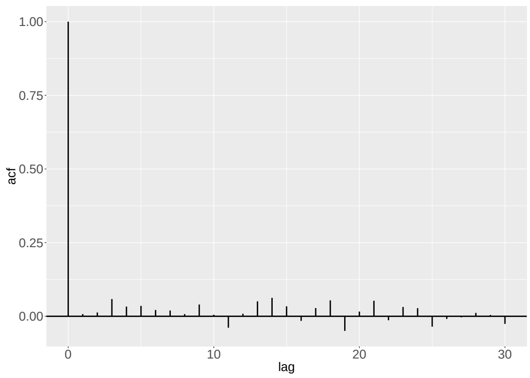
(a) ACF of residuals from *Blocks* with SNR at 3



(b) ACF of residuals from *Bumps* with SNR at 3

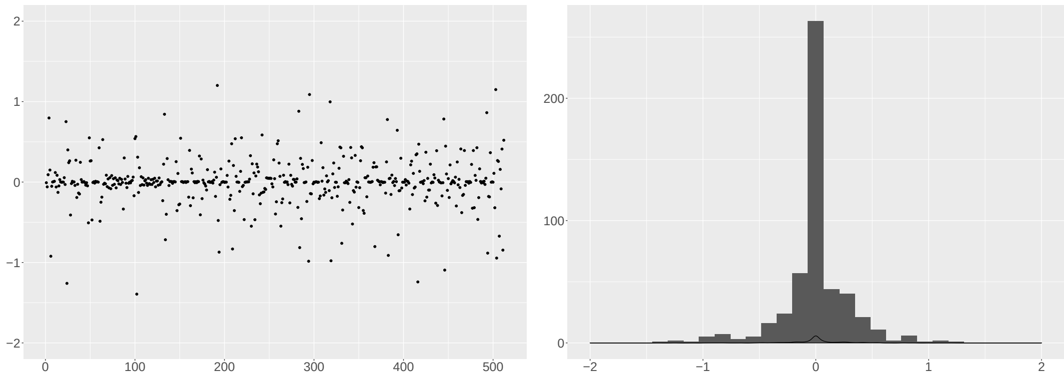


(c) ACF of residuals from *HeaviSine* with SNR at 3

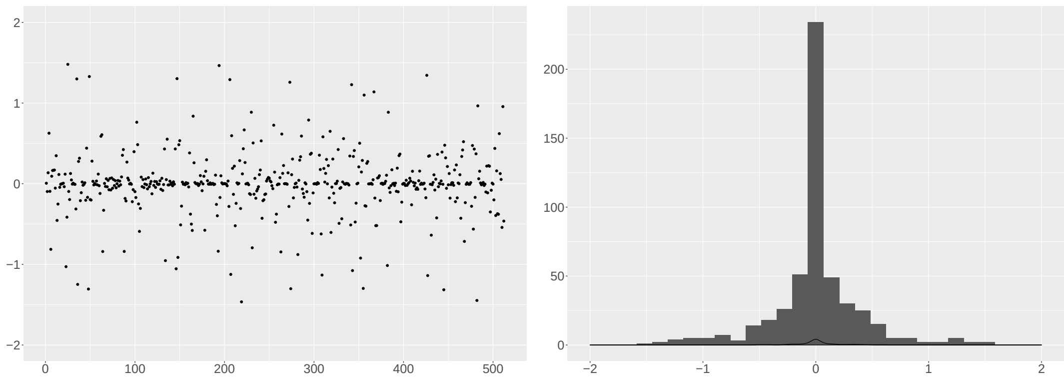


(d) ACF of residuals from *Doppler* with SNR at 3

Figure A.3: ACF of residuals at SNR level of 3.



(a) residuals of x



(b) residuals of y

Figure A.4: Residuals of 2-dimensional real data reconstruction

Appendix B

Calculations And Figures of Adaptive Sequential MCMC

B.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 1×1 constant K_t. By denoting $C_t^\top = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}$ and post-multiplying Σ_{YY}^{-1}, it gives us

$$\Sigma_{YY}^{-1} C_t = \frac{1}{\sigma^4}(\sigma^2 I - A^{-1})C_t = \frac{1}{\sigma^4}\begin{bmatrix} b_t \\ K_t \end{bmatrix}. \tag{B.1.1}$$

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

$$K_t = \frac{\sigma^4}{\tau^2 + \sigma^2 + \phi^2(\sigma^2 - K_{t-1})}, \tag{B.1.2}$$

$$b_t = \begin{bmatrix} \frac{b_{t-1}\phi K_t}{\sigma^2} \\ \frac{K_t(\sigma^2+\tau^2)-\sigma^4}{\phi\sigma^2} \end{bmatrix}. \tag{B.1.3}$$

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

$$\bar{\mu}_t = \frac{\phi}{\sigma^2} K_{t-1}\bar{\mu}_{t-1} + \phi(1 - \frac{K_{t-1}}{\sigma^2})y_{t-1}, \tag{B.1.4}$$

$$\bar{\Sigma}_t = \sigma^4 K_t^{-1}, \tag{B.1.5}$$

where $K_1 = \frac{\sigma^4}{\sigma^2+\tau^2+L^2\phi^2}$. For calculation details, readers can refer to appendices (B.1).

By using the formula, one term of equation (5.30) becomes

$$A_t^{-1}C_t = \left(I - \frac{M_t^{-1}u_t u_t^\top}{1 + u_t^\top M_t^{-1}u_t}\right) M_t^{-1}C_t, \tag{B.1.6}$$

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

$$A_t^{-1}C_t = \sigma^2 C_t - \frac{M_t^{-1}u_t\frac{\sigma^2}{\tau}}{1 + u^\top M_t^{-1}u}. \tag{B.1.7}$$

Moreover,

$$M_t^{-1}u_t = \begin{bmatrix} A_{t-1}^{-1} & 0 \\ 0 & \sigma^2 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -\frac{\phi}{\tau} \\ \frac{1}{\tau} \end{bmatrix} = \begin{bmatrix} A_{t-1}^{-1} & 0 \\ 0 & \sigma^2 \end{bmatrix} \begin{bmatrix} -\frac{\phi}{\tau}C_{t-1} \\ \frac{1}{\tau} \end{bmatrix} = \begin{bmatrix} -\frac{\phi}{\tau}A_{t-1}^{-1}C_{t-1} \\ \frac{\sigma^2}{\tau} \end{bmatrix},$$

$$\begin{aligned} u^\top M_t^{-1}u &= \begin{bmatrix} 0 & \cdots & 0 & -\frac{\phi}{\tau} & \frac{1}{\tau} \end{bmatrix} \begin{bmatrix} -\frac{\phi}{\tau}A_{t-1}^{-1}C_{t-1} \\ \frac{\sigma^2}{\tau} \end{bmatrix} = \begin{bmatrix} -\frac{\phi}{\tau}C_{t-1}^\top & \frac{1}{\tau} \end{bmatrix} \begin{bmatrix} -\frac{\phi}{\tau}A_{t-1}^{-1}C_{t-1} \\ \frac{\sigma^2}{\tau} \end{bmatrix} \\ &= \frac{\phi^2}{\tau^2}C_{t-1}^\top A_{t-1}^{-1}C_{t-1} + \frac{\sigma^2}{\tau^2}. \end{aligned}$$

Thus

$$A_t^{-1}C_t = \begin{bmatrix} -b_t \\ \sigma^2 - K_t \end{bmatrix} = \sigma^2 C_t - \frac{1}{1 + \frac{\phi^2}{\tau^2} C_{t-1}^\top A_{t-1}^{-1} C_{t-1} + \frac{\sigma^2}{\tau^2}} \begin{bmatrix} -\frac{\phi\sigma^2}{\tau^2} A_{t-1}^{-1} C_{t-1} \\ \frac{\sigma^4}{\tau^2} \end{bmatrix}$$
$$= \sigma^2 C_t - \frac{1}{\tau^2 + \phi^2 C_{t-1}^\top A_{t-1}^{-1} C_{t-1} + \sigma^2} \begin{bmatrix} -\phi\sigma^2 A_{t-1}^{-1} C_{t-1} \\ \sigma^4 \end{bmatrix} \tag{B.1.8}$$

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

$$K_t = \frac{\sigma^4}{\tau^2 + \sigma^2 + \phi^2(\sigma^2 - K_{t-1})}, \tag{B.1.9}$$

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 θ and its mean and variance can be found by

$$\mu_x^{(t)} = \frac{1}{N} \sum_i \mu_i \tag{B.1.10}$$

$$\mathrm{Var}(x)^{(t)} = \mathrm{E}(\mathrm{Var}(x \mid y, \theta)) + \mathrm{Var}(\mathrm{E}(x \mid y, \theta)) = \frac{1}{N} \sum_i (\mu_i \mu_i^\top + \Sigma_i) - \frac{1}{N^2} (\sum_i \mu_i)(\sum_i \mu_i)^\top. \tag{B.1.11}$$

To find μ_i and Σ_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 θ_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.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

$$p(Y_{1:t}, \theta) \sim N\left(0, \Sigma_{YY}^{(t)}\right)$$
$$p(Y_{t+1}, Y_{1:t}, \theta) \sim N\left(0, \Sigma_{YY}^{(t+1)}\right)$$
$$p(Y_{t+1} \mid Y_{1:t}, \theta) \sim N\left(\bar{\mu}_{t+1}, \bar{\Sigma}_{t+1}\right)$$

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 2×2 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 2×2 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 $S_{t+1} = \frac{1}{\sqrt{1-\rho_{t+1}^2}} \begin{bmatrix} \frac{1}{\sigma_{t+1}^{(x)}} & 0 \\ \frac{1-e^{-\gamma\Delta_{t+1}}}{\gamma\sigma_{t+1}^{(x)}} - \frac{\rho_{t+1}e^{-\gamma\Delta_{t+1}}}{\sigma_{t+1}^{(u)}} & \frac{\sqrt{1-\rho_{t+1}^2}e^{-\gamma\Delta_{t+1}}}{\sigma_{t+1}^{(u)}} \end{bmatrix}$,

$$
D_{t+1} = \frac{1}{\sqrt{1-\rho_{t+1}^2}} \begin{bmatrix} -\frac{1}{\sigma_{t+1}^{(x)}} & 0 \\ \frac{\rho_{t+1}}{\sigma_{t+1}^{(u)}} & -\frac{\sqrt{1-\rho_{t+1}^2}}{\sigma_{t+1}^{(u)}} \end{bmatrix} \text{ and } C_{t+1} = \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_t \\ I_2 \end{bmatrix}.
$$

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

$$
C_{t+1}^\top A_{t+1}^{-1} C_{t+1} = B_1^{-1} - K_{t+1}, \tag{B.2.1}
$$

$$
K_{t+1} = B_1^{-1} - C_{t+1}^\top A_{t+1}^{-1} C_{t+1}. \tag{B.2.2}
$$

We may use Sherman-Morrison-Woodbury formula to find the inverse of A_{t+1} in a recursive way, which is

$$A_{t+1}^{-1} = (M_{t+1} + U_{t+1}U_{t+1}^\top)^{-1} = M_{t+1}^{-1} - M_{t+1}^{-1}U_{t+1}(I + U_{t+1}^\top M_{t+1}^{-1}U_{t+1})^{-1}U_{t+1}^\top M_{t+1}^{-1}. \tag{B.2.3}$$

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.2.2), we will get

$$K_{t+1} = B_1^{-1}D_{t+1}(I + S_{t+1}^\top (B_1^{-1} - K_t)S_{t+1} + D_{t+1}^\top B_1^{-1}D_{t+1})^{-1} D_{t+1}^\top B_1^{-1}, \tag{B.2.4}$$

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.2.4) 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

$$\begin{cases} \bar{\mu}_{t+1} &= \Phi_{t+1} K_t B_1 \bar{\mu}_t + \Phi_{t+1} (I - K_t B_1) Y_t \\ \bar{\Sigma}_{t+1} &= (B_1 K_{t+1} B_1)^{-1} \end{cases}. \tag{B.2.5}$$

The matrix K_{t+1} is updated via

$$K_{t+1} = B_1^{-1} D_{t+1} (I + S_{t+1}^{\top} (B_1^{-1} - K_t) S_{t+1} + D_{t+1}^{\top} B_1^{-1} D_{t+1})^{-1} D_{t+1}^{\top} B_1^{-1}, \tag{B.2.6}$$

or updating its inverse in the following form makes the computation faster, that is

$$\begin{cases} K_{t+1}^{-1} &= B_1 D_{t+1}^{-\top} D_{t+1}^{-1} B_1 + B_1 \Phi_{t+1} (B_1^{-1} - K_t) \Phi_{t+1}^{\top} B_1 + B_1, \\ \bar{\Sigma}_{t+1} &= D_{t+1}^{-\top} D_{t+1}^{-1} + \Phi_{t+1} (B_1^{-1} - K_t) \Phi_{t+1}^{\top} + B_1^{-1} \end{cases}$$

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 θ by

$$\begin{aligned} X \mid Y, \theta &\sim N\left(A^{-1} B Y, A^{-1}\right) \\ &\sim N(L^{-\top} L^{-1} B Y, L^{-\top} L^{-1}) \\ &\sim N(L^{-\top} W, L^{-\top} L^{-1}),\end{aligned}$$

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 with 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

$$\begin{aligned} \bar{\mu}_{t+1}^{(X)} &= \Phi \hat{x}_t + C_{t+1}^\top (A-B)^{-1} B (I - A^{-1}B) Y_{1:t+1} \\ &= \Phi \hat{x}_t + C_{t+1}^\top A^{-1} B Y_{1:t+1} \\ &= 0 + \begin{bmatrix} -b_{t+1}^\top & B_1^{-1} - K_{t+1} \end{bmatrix} \begin{bmatrix} B_t & 0 \\ 0 & B_1 \end{bmatrix} \begin{bmatrix} Y_{1:t} \\ Y_{y+1} \end{bmatrix} \\ &= -b^\top B_t Y_{1:t} + (I - B_1 K_{t+1}) Y_{t+1} \\ &= K_{t+1} B_1 \bar{\mu}_{t+1} + (I - B_1 K_{t+1}) Y_{t+1} \\ \bar{\sigma}_{t+1}^{(X)2} &= C_{t+1}^\top (A-B)^{-1} C_{t+1} - C_{t+1}^\top (A-B)^{-1} B (I - A^{-1}B)(A-B)^{-1} C_{t+1} \\ &= C_{t+1}^\top (A-B)^{-1} C_{t+1} - C_{t+1}^\top A^{-1} B (A-B)^{-1} C_{t+1} \\ &= C_{t+1}^\top A^{-1} C_{t+1} \\ &= B_1^{-1} - K_{t+1}. \end{aligned}$$

B.3 Covariance Matrix in Details

$$
\begin{aligned}
\Sigma_t &= \begin{bmatrix} \sigma_t^{(x)2} & \rho_t \sigma_t^{(x)} \sigma_t^{(u)} \\ \rho_t \sigma_t^{(x)} \sigma_t^{(u)} & \sigma_t^{(u)2} \end{bmatrix} \\
\Sigma_t^{-1} &= \frac{1}{1-\rho_t^2} \begin{bmatrix} \frac{1}{\sigma_t^{(x)2}} & -\frac{\rho_t}{\sigma_t^{(x)} \sigma_t^{(u)}} \\ -\frac{\rho_t}{\sigma_t^{(x)} \sigma_t^{(u)}} & \frac{1}{\sigma_t^{(u)2}} \end{bmatrix} \\
M_t^\top &= \begin{bmatrix} 1 & 0 \\ \frac{1-e^{-\gamma \Delta_t}}{\gamma} & e^{-\gamma \Delta_t} \end{bmatrix} \\
z_t &= \begin{bmatrix} x_t \\ u_t \end{bmatrix}
\end{aligned}
$$

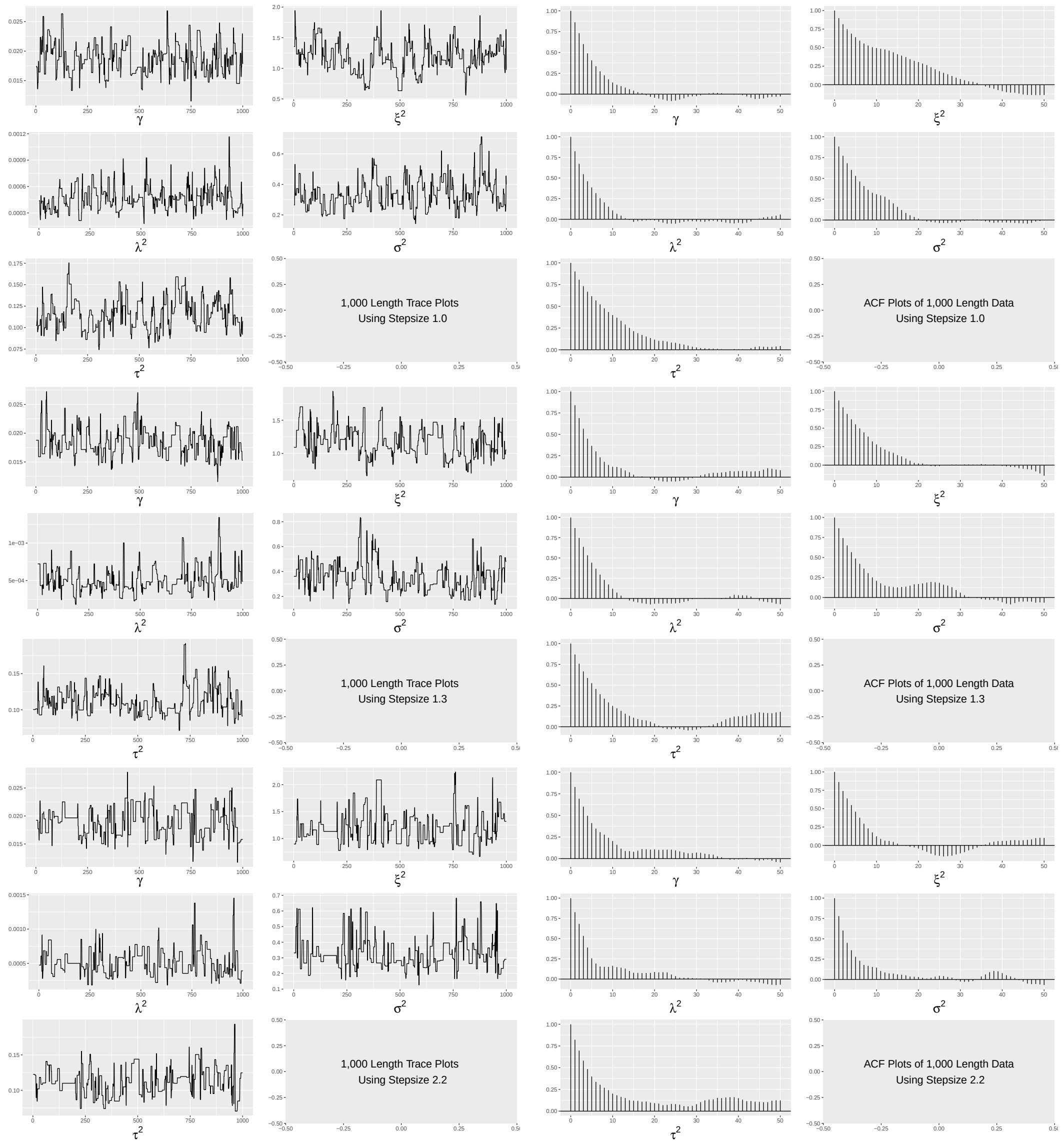
$$
\begin{aligned}
M_t^\top \Sigma_t^{-1} &= \frac{1}{1-\rho_t^2} \begin{bmatrix} \frac{1}{\sigma_t^{(x)2}} & -\frac{\rho_t}{\sigma_t^{(x)} \sigma_t^{(u)}} \\ \frac{1-e^{-\gamma \Delta_t}}{\gamma \sigma_t^{(x)2}} - \frac{\rho_t e^{-\gamma \Delta_t}}{\sigma_t^{(x)} \sigma_t^{(u)}} & -\frac{\rho_t (1-e^{-\gamma \Delta_t})}{\gamma \sigma_t^{(x)} \sigma_t^{(u)}} + \frac{e^{-\gamma \Delta_t}}{\sigma_t^{(u)2}} \end{bmatrix} \\
M_t^\top \Sigma_t^{-1} M_t &= \frac{1}{1-\rho_t^2} \begin{bmatrix} \frac{1}{\sigma_t^{(x)2}} & \frac{1-e^{-\gamma \Delta_t}}{\gamma \sigma_t^{(x)2}} - \frac{\rho_t e^{-\gamma \Delta_t}}{\sigma_t^{(x)} \sigma_t^{(u)}} \\ \frac{1-e^{-\gamma \Delta_t}}{\gamma \sigma_t^{(x)2}} - \frac{\rho_t e^{-\gamma \Delta_t}}{\sigma_t^{(x)} \sigma_t^{(u)}} & \frac{(1-e^{-\gamma \Delta_t})^2}{\gamma^2 \sigma_t^{(x)2}} - \frac{2\rho_t e^{-\gamma \Delta_t} (1-e^{-\gamma \Delta_t})}{\gamma \sigma_t^{(x)} \sigma_t^{(u)}} + \frac{e^{-2\gamma \Delta_t}}{\sigma_t^{(u)2}} \end{bmatrix}
\end{aligned}
$$

$$\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.4 Real Data Implementation

Efficiency Plots



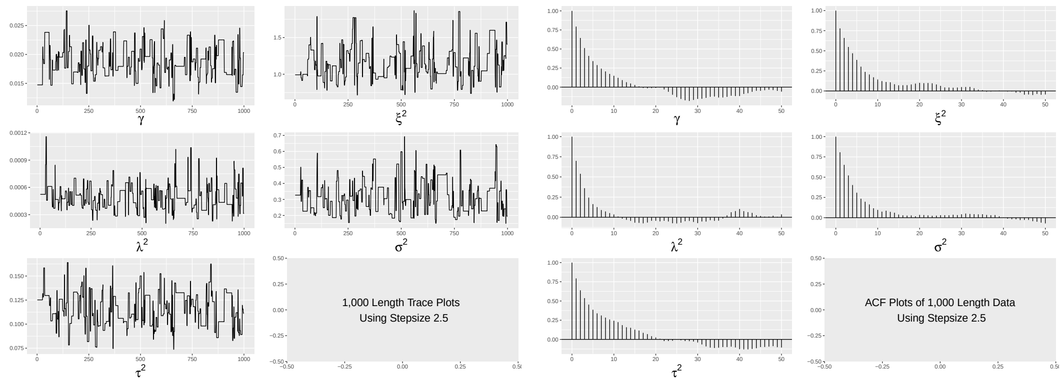


Figure B.1: Running the same amount of time and taking the same length of data, the step size $\epsilon = 2.5$ returns the highest ESSUT value and generates more effective samples with a lower correlation.

Comparing Estimation with Different Length of Data

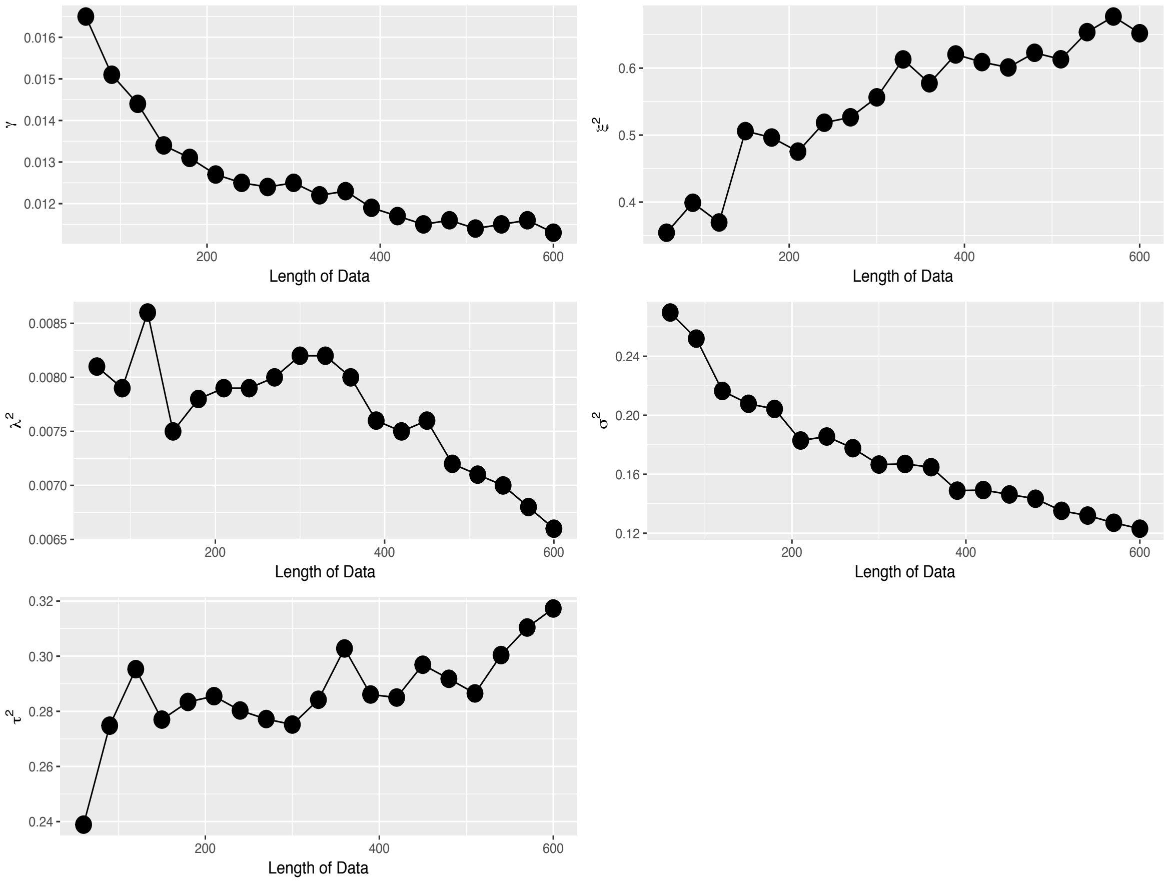
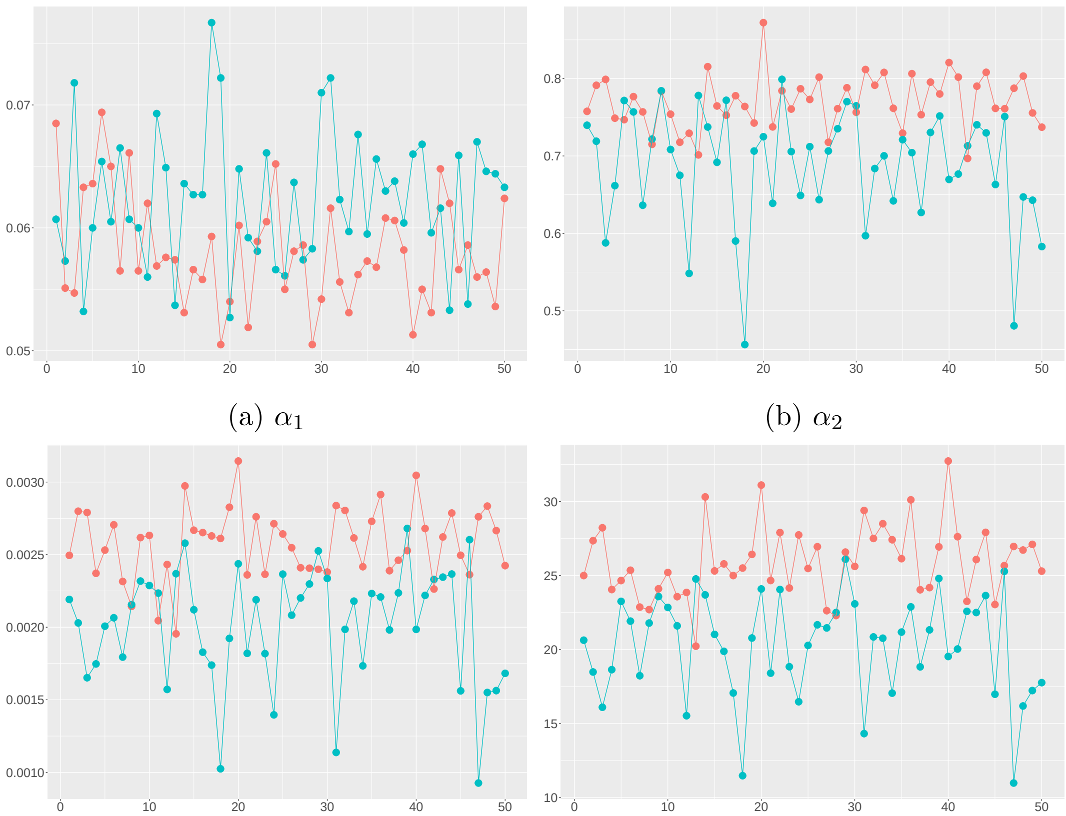


Figure B.2: Impacts of data length on optimal parameter. There is an obvious trend on the estimation against length of data in estimation process.

Table B.1: Parameter estimation by running the whole surface learning and DA-MH processes with different length of data

Length	**Time**	γ	ξ^2	λ^2	σ^2	τ^2	α_1	α_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

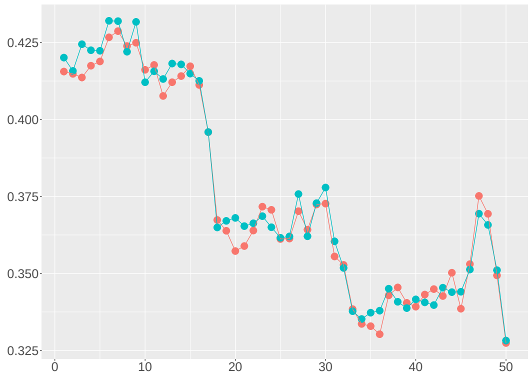
B.5 Comparison Between Batch and Sliding Window Methods



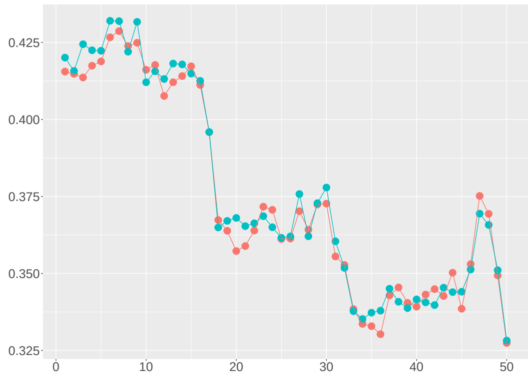
(a) α_1 (b) α_2

(c) EffUT (d) ESSUT

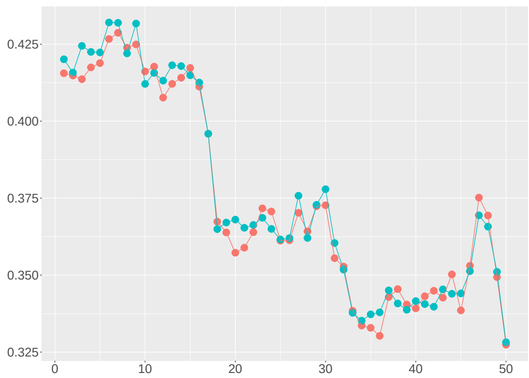
Figure B.3: Comparison of α_1, α_2, EffUT and ESSUT between batch MCMC (orange) and sliding window MCMC (green).



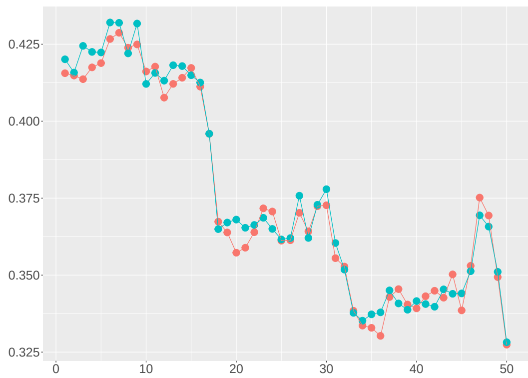
(a) γ



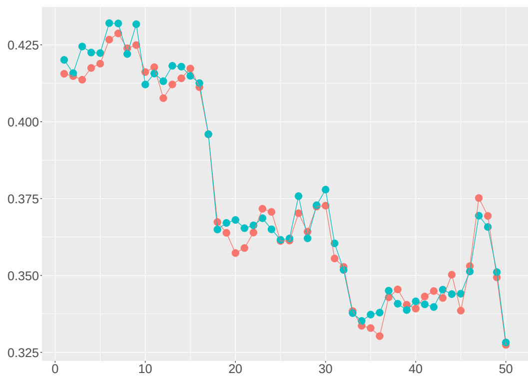
(b) ξ^2



(c) λ^2



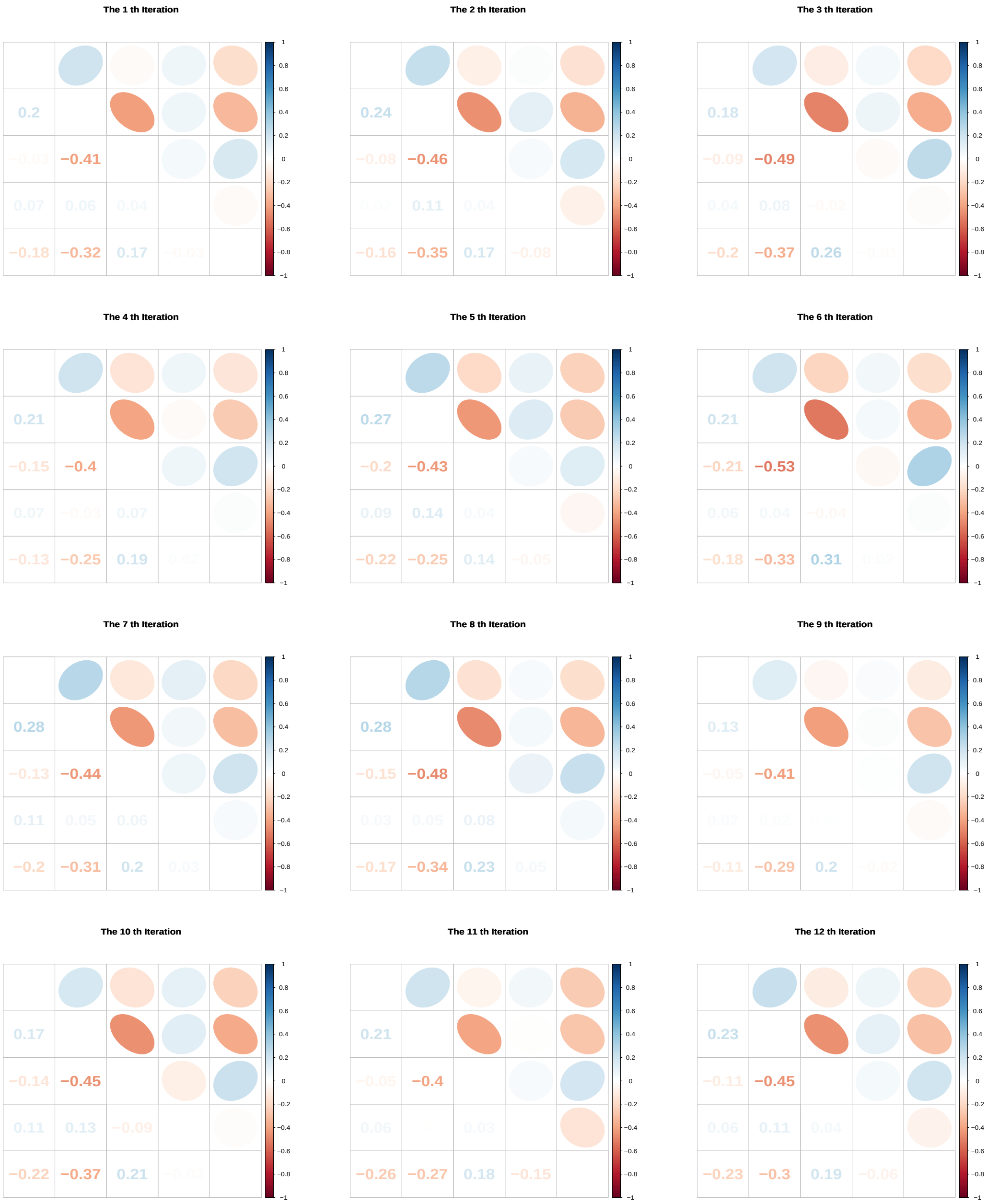
(d) σ^2



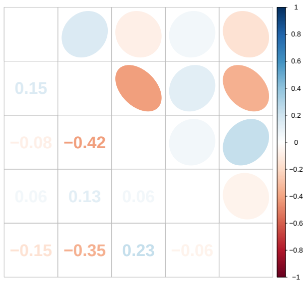
(e) τ^2

Figure B.4: Comparison of parameters estimation between batch MCMC (orange) and sliding window MCMC (green).

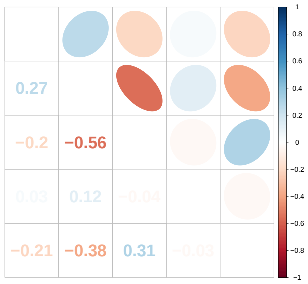
B.6 Parameter Evolution Visualization



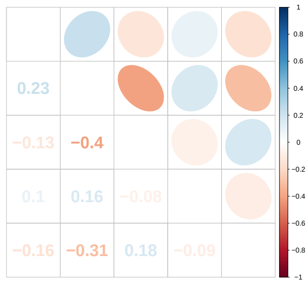
The 13 th Iteration



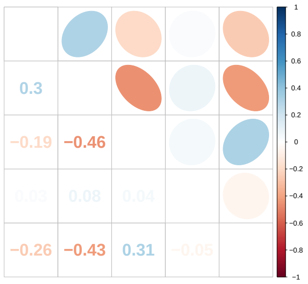
The 14 th Iteration



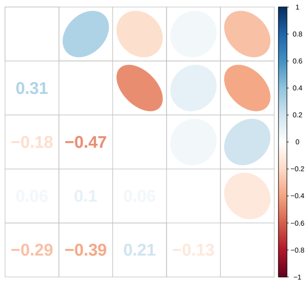
The 15 th Iteration



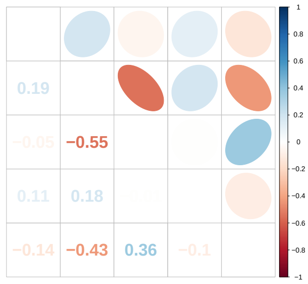
The 16 th Iteration



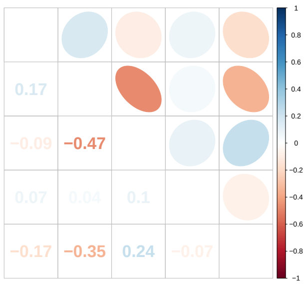
The 17 th Iteration



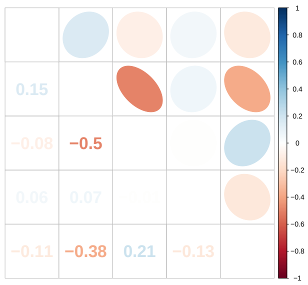
The 18 th Iteration



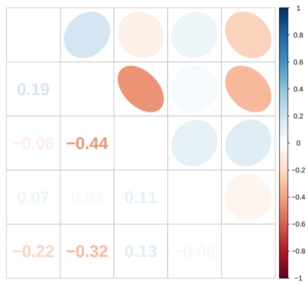
The 19 th Iteration



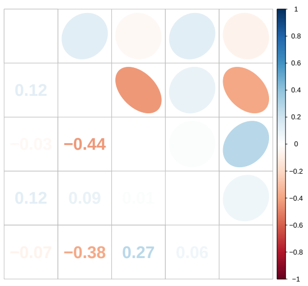
The 20 th Iteration



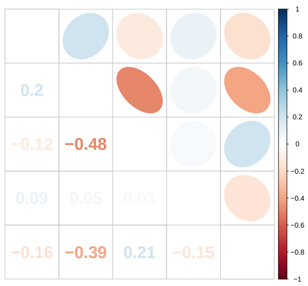
The 21 th Iteration



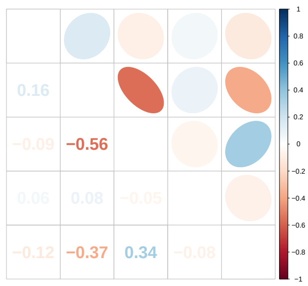
The 22 th Iteration



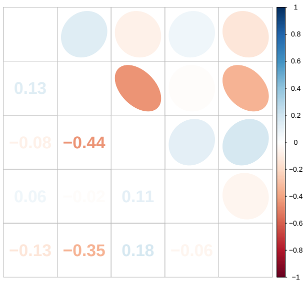
The 23 th Iteration



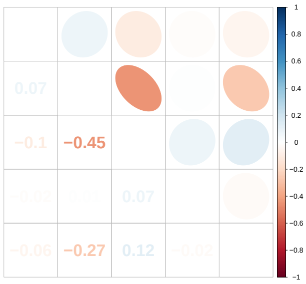
The 24 th Iteration



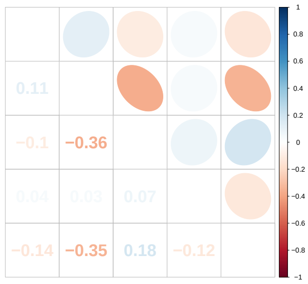
The 25 th Iteration



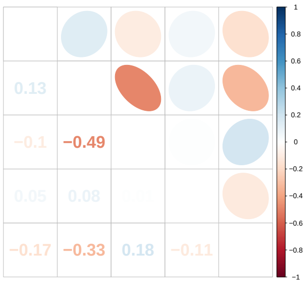
The 26 th Iteration



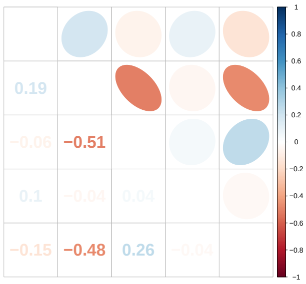
The 27 th Iteration



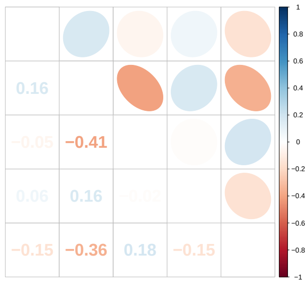
The 28 th Iteration



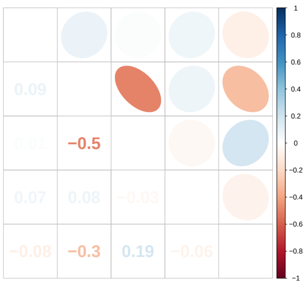
The 29 th Iteration



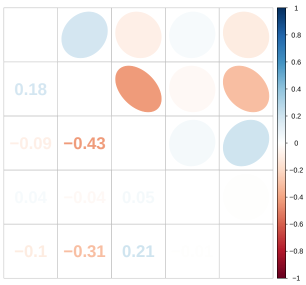
The 30 th Iteration



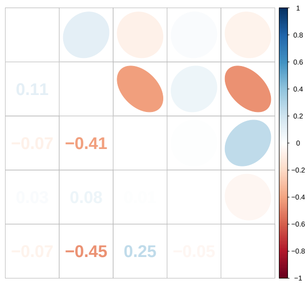
The 31 th Iteration



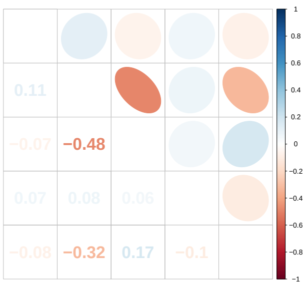
The 32 th Iteration



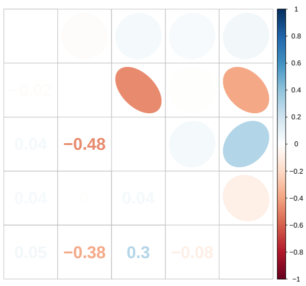
The 33 th Iteration



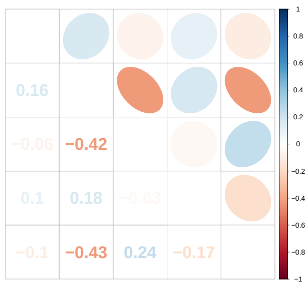
The 34 th Iteration



The 35 th Iteration



The 36 th Iteration



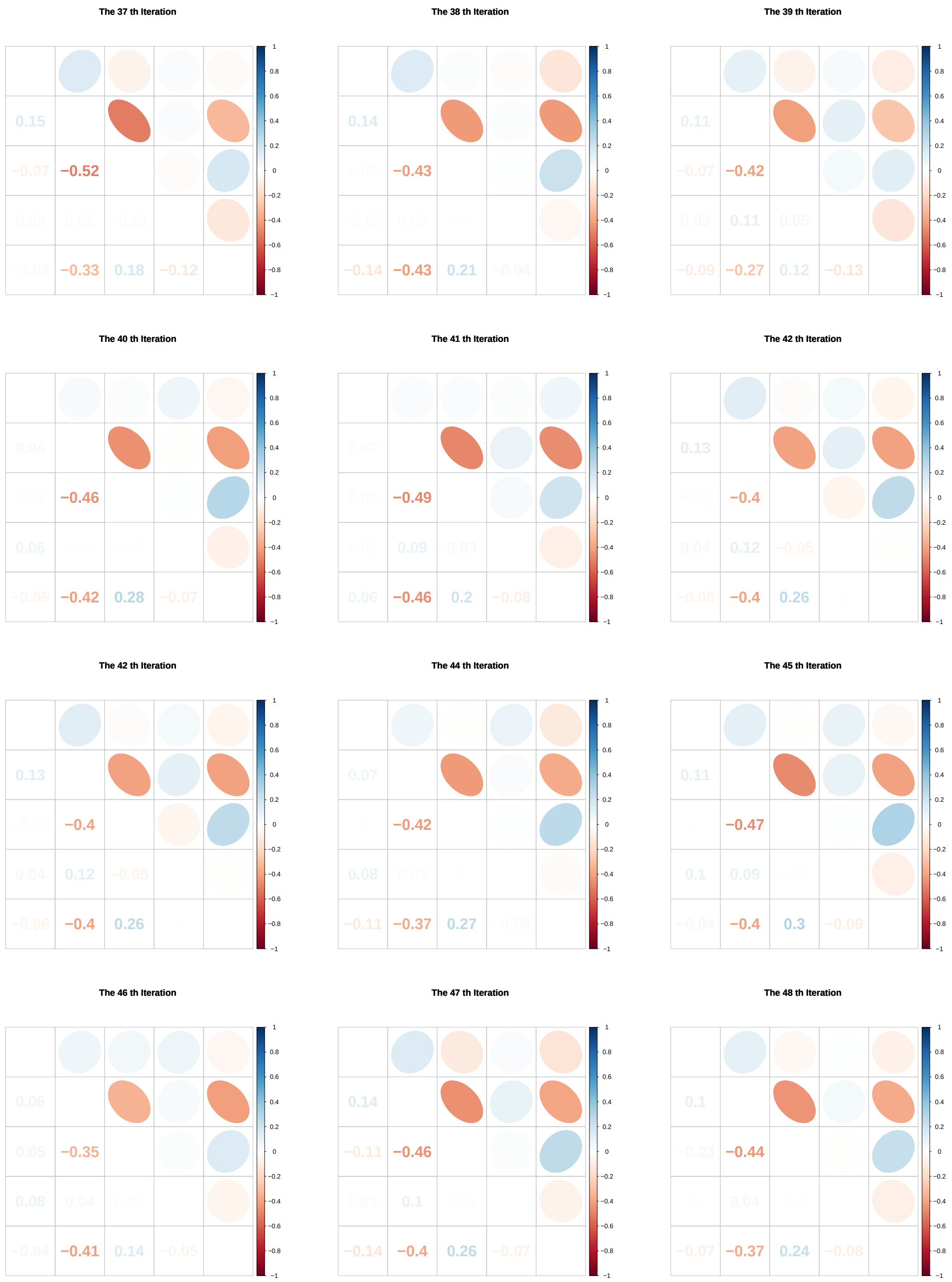


Figure B.4: Parameters Evolution Visualization. The correlation among parameters does not change two much. The parameters are considered static.

Appendix C

A Spin-off Outcome: Data Simplification Method

C.1 Introduction

GPS devices are widely used in orchard planting and maintenance. This location-based system allows orchardist to check trajectory of tractors. The trajectory is a connection by a time series successive positions recorded by GPS devices. A classical GPS device records skeleton information, including time mark, 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).

Normally GPS units record more data than necessary and cause more errors due to weak signals or shelter from branches. To obtain a higher accurate observation dataset and to same local storage space, several data simplification methods were proposed and are focusing on simplifying data set by making either a local or global decision.

A local simplification algorithm focuses on a couple of particular consecutive points. By analyzing the relationship between these points, a decision is made that which point can be deleted or retained. Distance threshold algorithm is one of these algorithms. All points, whose distance to the preceding track point is less than a predetermined threshold, are deleted. Direction changing algorithm is another one. The point is retained if the change in direction is greater than a predetermined threshold Ivanov (2012).

Alternatively, global simplification 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 to retain. The Douglas-Peucker algorithm is the most popular one Douglas and Peucker (1973). A proposed simplification method, represented in Chen *et al.* (2009), consider both the skeleton information and semantic meanings of a trajectory when performing simplification.

Intuitively, the global simplification algorithms can be applied 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.

In our case, a GPS log is a sequence time series points $p_i \in P$, $P = \{p_1, p_2, \ldots, p_n\}$. Each GPS point p_i contains information of time mark, latitude, longitude and semantic information of velocity, heading direction and boom status, which can be written in form of

$$T = \{p_t = [x_t, y_t, v_t, \theta_t, b_t] \mid t \in \mathbb{R}\}. \tag{C.1.1}$$

Sequentially connect these points will give us a trajectory of a moving vehicle. Particularly, a tractor working on an orchard generates two kinds of boom status information: operating and not operating. This information is recorded by GPS units and is indicated by $b = 1$ for operating and $b = 0$ for not-operating.

To move further, here are two concepts that will be useful to understand the simplification scheme.

- **Segment** A segment is a part of the consecutive trajectory. Regarding the status of the boom, the trajectory can be simply divided into two kinds of the segment in our dataset, one is boom-operating, the other is boom-not-operating.

- **Direction**. Direction θ denotes the heading direction of a tractor at a specific point location. This parameter uses north direction as a basis, in which way $0° \leq \theta < 360°$.

C.2 Simplification Algorithm

The first two steps are designed to reduce some errors caused by misoperation and GPS units bugs.

- Merging Phase. If the length of a segment composed of consecutive boom operating or not-operating points is less than a threshold, merge this one into its backward segment.

- Removing Phase. If two or more data points have duplicated time mark, remove the latter ones.

Now only two types of segment points are left in GPS log, boom operating, and not-operating and the length of each segment are greater than the predetermined threshold.

The following algorithm is based on the relationship between a candidate point p_i and its neighboring points p_{i-1} and p_{i+1}, and the importance of the p_i in the segment where it belongs to, $i = 2, \ldots, n-1$.

- 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 $p_{i-1}, p_i, \ldots, p_{i+k}$ 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}, θ 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.

C.3 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 C.1, the green points $P_{t1}, \ldots, P_{t5}$ are original positions. After simplification, the points P_{t2}, P_{t3} and P_{t4} were removed. The black curve is the original trajectory, in contrast, the gray dash-dot line is the simplified trajectory. The blue points P'_{t2},P'_{t3} and P'_{t4} on simplified trajectory have the same time difference as the point P_{t2}, P_{t3} and P_{t4} on original trajectory did. For instance, the time difference between P_{t2} and P_{t3} is the same as that between P'_{t2} and P'_{t3}. Further, the distances between P_{t2} and P'_{t2}, P_{t3} and P'_{t3} and P_{t4}, P'_{t4} can be calculated.

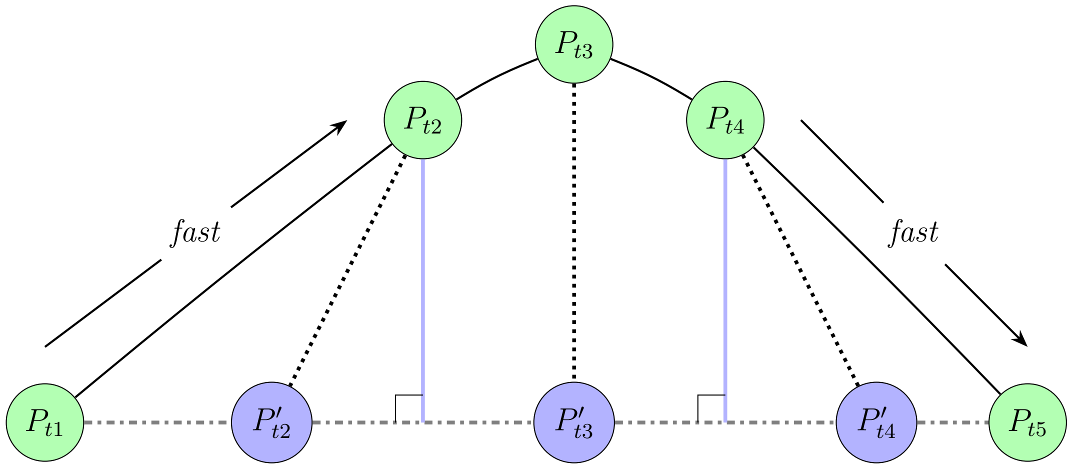
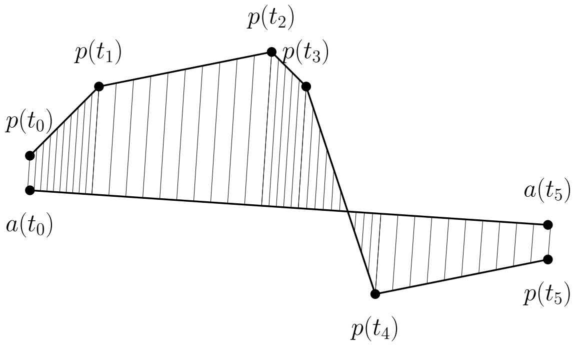
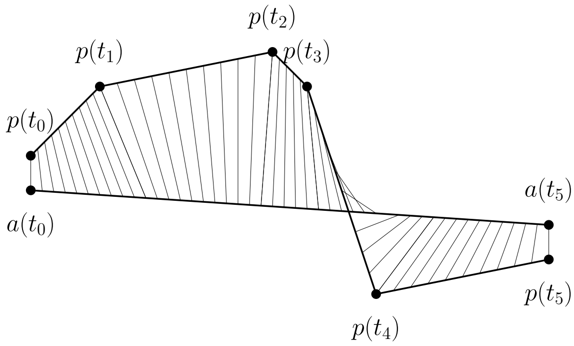


Figure C.1: Synchronized Euclidean Distance



(a) perpendicular distance chords



(b) time-synchronous distance chords

Figure C.2: C.2a indicates that the errors are measured at fixed sampling rate as sum of perpendicular distance chords. C.2b indicates that the errors are measured at fixed sampling rates as sum of time-synchronous distance chords.

Another way to calculate the difference between a GPS trace and its compressed version is to measure the perpendicular distance. This algorithm ignores the temporal component and uses simple perpendicular distance Meratnia and Rolf (2004). The figure C.2 expresses these differences clearly.

C.4 Numerical Study

In the numerical simulation study, we are using *Kalman Filter* (KF) to fit the trajectory after data simplification. The KF equations describe the prediction step in

Table C.1: Comparison between raw data and simplifying algorithms

	Original Data	**DP Algorithm**	**Proposed Algorithm**
Points Left	1021	847	847
Tracked Distances(m)	74041.31	74038.33	74012.56
SED (m)	NA	1316.715	607.9587

the following way:

$$
\begin{aligned}
\hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\
P_k^- &= AP_{k-1}A^\top + Q
\end{aligned}
$$

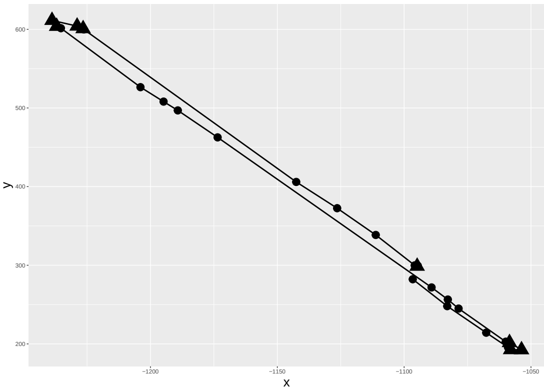
where $\hat{x}_k^-$ is a priori state estimate, $\hat{x}_k$ is a posteriori state estimate, A is status transition matrix, P_k^- is a priori estimate for error covariance, u_k is an input parameter and Q is process noise covariance. When a new observation comes into the data stream, KF update and corrects its estimation by:

$$
\begin{aligned}
K_k &= P_k^- H^\top (HP_k^- H^\top + R)^{-1} \\
\hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\
P_k &= (I - K_k H)P_k^-
\end{aligned}
$$

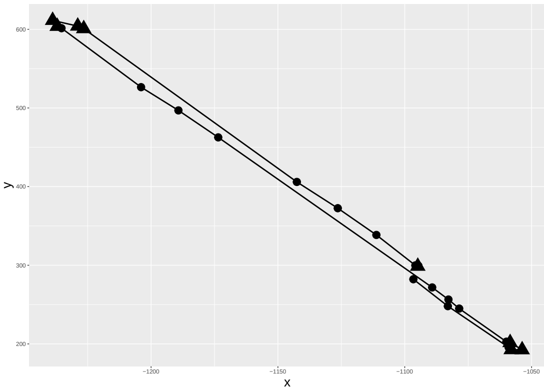
where K_k is the Kalman gain matrix, z_k is the observed data.

The original data set has 1 021 rows, each of them contains latitude, longitude, velocity, bearing (heading direction) and boom status. Douglas-Peucker Algorithm, with distance threshold 0.205m, retained 847 points. The proposed algorithm, given a predictable distance 5m and heading direction changing threshold 30°, returns the same amount of simplified points. Under the same circumstance, we calculated SED and other information.

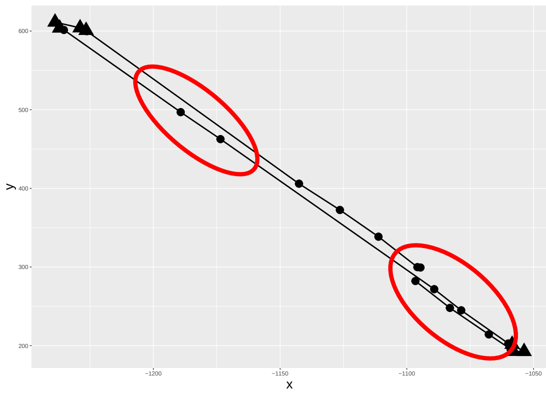
Table C.1 describes the results after being simplified by DP algorithm and the proposed algorithm. Figure C.3 demonstrates the simplified raw data and figure C.4 is the fitted trajectories by KF.



(a) Raw trajectory

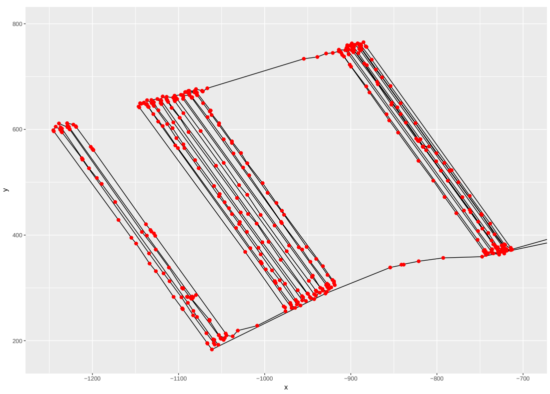


(b) Simplified trajectory by DP

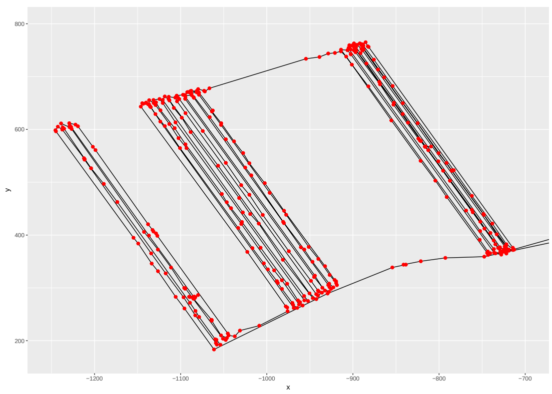


(c) Simplified trajectory by proposed algorithm

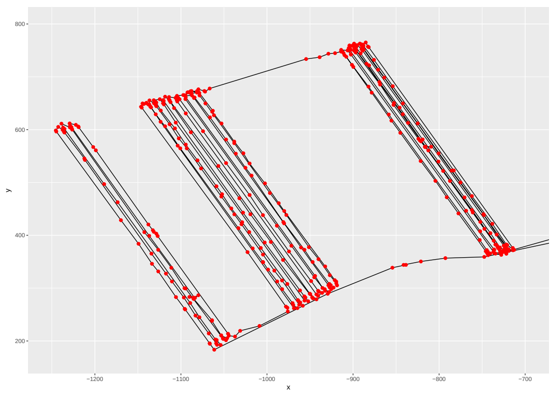
Figure C.3: A segment start from time $t = 2\,000$ to $3\,000$, recorded by GPS units. ▲ indicates that the boom is not-operating. ● indicates that the boom is operating. Figure C.3a, the trajectory connected by raw data with 27 points. Figure C.3b, the trajectory connected by simplified data with Douglas-Peucker algorithm with 24 points. Figure C.3c, the trajectory connected by simplified data with proposed simplification algorithm with 23 points.



(a) Fitted Kalman Filter with raw data



(b) Fitted Kalman Filter with simplified data by DP



(c) Fitted Kalman Filter with simplified data by proposed algorithm

Figure C.4: Trajectory fitted by Kalman Filter. The mean squared errors of raw data, DP and proposed algorithm are 26.8922, 23.9788 and 23.9710 respectively.

C.5 Conclusion

The data simplification algorithm was originally proposed to solve the fitting-drift problem. Duplicated and short-distance points cause reconstruction issues in spline fitting. The advantage of using data simplification algorithm is that less data points will potentially increase computation efficiency and save storage space without of losing information in reconstruction.

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.

Agrawal, D. P. and Zeng, Q.-A. (2015). *Introduction to wireless and mobile systems*. Cengage learning.

Almasi, G. S. and Gottlieb, A. (1994). *Highly Parallel Computing (2Nd Ed.)*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.

Anderson, B. D. and Moore, J. B. (1979). Optimal filtering. *Englewood Cliffs*, *21*, 22–95.

Anderson, J. L. (2001). An ensemble adjustment Kalman filter for data assimilation. *Monthly weather review*, *129*(12), 2884–2903.

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., Djurić, P., and Doucet, A. (2001). Model selection by MCMC computation. *Signal Processing*, *81*(1), 19–37.

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.

Arlot, S., Celisse, A., *et al.* (2010). A survey of cross-validation procedures for model selection. *Statistics surveys*, *4*, 40–79.

Arnaud Doucet, Nando de Freitas, N. G. (2011). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag New York.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, *68*(3), 337–404.

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.

Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., *et al.* (2006). The landscape of parallel computing research: A view from berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley.

Atchade, Y., Fort, G., Moulines, E., and Priouret, P. (2009). Adaptive Markov chain Monte Carlo: theory and methods. *Preprint*.

Atkinson, A.D., S. J. (2004). Propuesta para el levantamiento de vas de comunicacin para navegacin para mediante GPS. *Datum*, *XXI 9*, 29–32.

Aydın, D., Memmedli, M., and Omay, R. E. (2013). Smoothing parameter selection for nonparametric regression using smoothing spline. *European Journal of Pure and applied mathematics*, *6*(2), 222–238.

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.

Babenko, B., Yang, M.-H., and Belongie, S. (2009). A family of online boosting algorithms. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, 1346–1353. IEEE.

Bajaj, R., Ranaweera, S. L., and Agrawal, D. P. (2002). GPS: location-tracking technology. *Computer*, *35*(4), 92–94.

Bar-Shalom, Y. and Li, X.-R. (1993). Estimation and tracking- Principles, techniques, and software. *Norwood, MA: Artech House, Inc, 1993.*.

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.

Betancourt, M. (2017). A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*.

Beygelzimer, A., Hazan, E., Kale, S., and Luo, H. (2015). Online gradient boosting. In *Advances in Neural Information Processing Systems*, 2458–2466.

Biagiotti, L. and Melchiorri, C. (2013). Online trajectory planning and filtering for robotic applications via B-spline smoothing filters. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 5668–5673. IEEE.

Bierkens, J. and Duncan, A. (2017). Limit theorems for the Zig-Zag process. *Advances in Applied Probability*, *49*(3), 791–825.

Bierkens, J., Fearnhead, P., and Roberts, G. (2016a). The Zig-Zag Process and Super-Efficient Sampling for Bayesian Analysis of Big Data. *arXiv preprint arXiv:1607.03188*.

Bierkens, J., Roberts, G., *et al.* (2016b). A piecewise deterministic scaling limit of lifted Metropolis–Hastings in the Curie–Weiss model. *The Annals of Applied Probability*, *27*(2), 846–882.

Bishop, G. and Welch, G. (2001). An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, *8*(27599-23175), 41.

Blaauw, M., Christen, J. A., *et al.* (2011). Flexible paleoclimate age-depth models using an autoregressive gamma process. *Bayesian analysis*, *6*(3), 457–474.

Box, G. E. and Tiao, G. C. (2011). *Bayesian inference in statistical analysis*, Volume 40. John Wiley & Sons.

Bradford, R. and Thomas, A. (1996). Markov chain Monte Carlo methods for family trees using a parallel processor. *Statistics and Computing*, *6*(1), 67–75.

Branson, Z., Rischard, M., Bornn, L., and Miratrix, L. (2017). A Nonparametric Bayesian Methodology for Regression Discontinuity Designs. *arXiv preprint arXiv:1704.04858*.

Cantoni, E. and Ronchetti, E. (2001). Resistant selection of the smoothing parameter for smoothing splines. *Statistics and Computing*, *11*(2), 141–146.

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.

Carlin, B. P., Polson, N. G., and Stoffer, D. S. (1992). A Monte Carlo approach to nonnormal and nonlinear state-space modeling. *Journal of the American Statistical Association*, *87*(418), 493–500.

Carpenter, J., Clifford, P., and Fearnhead, P. (1999). Improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation*, *146*(1), 2–7.

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.

Chadil, N., Russameesawang, A., and Keeratiwintakorn, P. (2008). Real-time tracking management system using GPS, GPRS and Google earth. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-Con 2008. 5th International Conference on*, Volume 1, 393–396. IEEE.

Chandrasekar, J., Ridley, A., and Bernstein, D. (2007). A comparison of the extended and unscented Kalman filters for discrete-time systems with nondifferentiable dynamics. In *American Control Conference, 2007. ACC'07*, 4431–4436. IEEE.

Chen, M.-H., Shao, Q.-M., and Ibrahim, J. G. (2012). *Monte Carlo methods in Bayesian computation.* Springer Science & Business Media.

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, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794. ACM.

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.

Chen, Z. *et al.* (2003). Bayesian filtering: From Kalman filters to particle filters, and beyond. *Statistics*, *182*(1), 1–69.

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.

Christen, J. A., Fox, C., *et al.* (2010). A general purpose sampling algorithm for continuous distributions (the t-walk). *Bayesian Analysis*, *5*(2), 263–281.

Cox, D. D. (1993). An analysis of Bayesian inference for nonparametric regression. *The Annals of Statistics*, 903–923.

Cox, M. G. (1982). Practical spline approximation. In *Topics in Numerical Analysis*, 79–112. Springer.

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.

Dierckx, P. (1995). *Curve and surface fitting with splines.* Oxford University Press.

Dieudonné, J. (2013). *Foundations of modern analysis.* Read Books Ltd.

Diggle, P. J. and Hutchinson, M. F. (1989). On spline smoothing with autocorrelated errors. *Australian & New Zealand Journal of Statistics*, *31*(1), 166–182.

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., Kerkyacharian, G., and Picard, D. (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.

Doucet, A., De Freitas, N., Murphy, K., and Russell, S. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 176–183. Morgan Kaufmann Publishers Inc.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, *10*(3), 197–208.

Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, *12*(656-704), 3.

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.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, *195*(2), 216–222.

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.

Durbin, J. and Koopman, S. J. (2012). *Time series analysis by state space methods*, Volume 38. OUP Oxford.

Eilers, P. H. and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical science*, 89–102.

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.

Eubank, R. (2004). A simple smoothing spline, III. *Computational Statistics*, *19*(2), 227–241.

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.

Freund, Y. and Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, 23–37. Springer.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Gasparetto, A. and Zanotto, V. (2007). A new method for smooth trajectory planning of robot manipulators. *Mechanism and machine theory*, *42*(4), 455–471.

Gelb, A. (1974). *Applied optimal estimation*. MIT press.

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.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, 457–472.

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.

Godsill, S., Doucet, A., and West, M. (2000). Methodology for Monte Carlo smoothing with application to time-varying autoregressions.

Godsill, S., Doucet, A., and West, M. (2001). Maximum a posteriori sequence estimation using Monte Carlo particle filters. *Annals of the Institute of Statistical Mathematics*, *53*(1), 82–96.

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. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, *82*(4), 711–732.

Green, P. J. and Silverman, B. W. (1993). *Nonparametric regression and generalized linear models: a roughness penalty approach*. CRC Press.

Gu, C. (1998). Model indexing and smoothing parameter selection in nonparametric function estimation. *Statistica Sinica*, 607–623.

Gu, C. (2013). *Smoothing spline ANOVA models*, Volume 297. Springer Science & Business Media.

Gu, C. and Wahba, G. (1991). Minimizing GCV/GML scores with multiple smoothing parameters via the Newton method. *SIAM Journal on Scientific and Statistical Computing*, *12*(2), 383–398.

György, K., Kelemen, A., and Dávid, L. (2014). Unscented Kalman filters and Particle Filter methods for nonlinear state estimation. *Procedia Technology*, *12*, 65–74.

Haario, H., Saksman, E., and Tamminen, J. (1999). Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, *14*(3), 375–396.

Haario, H., Saksman, E., Tamminen, J., *et al.* (2001). An adaptive Metropolis algorithm. *Bernoulli*, *7*(2), 223–242.

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.

Härdle, W. (1990). *Applied nonparametric regression.* Number 19. Cambridge university press.

Härdle, W., Hall, P., and Marron, J. S. (1988). How far are automatically chosen regression smoothing parameters from their optimum? *Journal of the American Statistical Association*, *83*(401), 86–95.

Hartmann, M., Nowak, T., Pfandenhauer, O., Thielecke, J., and Heuberger, A. (2016). A grid-based filter for tracking bats applying field strength measurements. In *Wireless On-demand Network Systems and Services (WONS), 2016 12th Annual Conference on*, 1–8. IEEE.

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.

Haykin, S. S. *et al.* (2001). *Kalman filtering and neural networks.* Wiley Online Library.

Heckman, N. E. and Woodroofe, M. (1991). Minimax Bayes estimation in nonparametric regression. *The Annals of Statistics*, 2003–2014.

Hightower, J. and Borriello, G. (2001). Location systems for ubiquitous computing. *Computer*, *34*(8), 57–66.

Higuchi, T. (2001). Self-organizing time series model. In *Sequential Monte Carlo Methods in Practice*, 429–444. Springer.

Hurvich, C. M., Simonoff, J. S., and Tsai, C.-L. (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *60*(2), 271–293.

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.

Jauch, J., Bleimund, F., Rhode, S., and Gauterin, F. (2017). Recursive B-spline approximation using the Kalman filter. *Engineering Science and Technology, an International Journal*, *20*(1), 28–34.

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.

Julier, S. J. and Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems. In *Int. symp. aerospace/defense sensing, simul. and controls*, Volume 3, 182–193. Orlando, FL.

Kailath, T. (1981). Lectures on Wiener and Kalman filtering. In *Lectures on Wiener and Kalman Filtering*, 1–143. Springer.

Kalman, R. E. *et al.* (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, *82*(1), 35–45.

Kalos, M. H. and Whitlock, P. A. (2008). *Monte carlo methods.* John Wiley & Sons.

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.

Kaplan, E. and Hegarty, C. (2005). *Understanding GPS: principles and applications.* Artech house.

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.

Kloek, T. and Van Dijk, H. K. (1978). Bayesian estimates of equation system parameters: an application of integration by Monte Carlo. *Econometrica: Journal of the Econometric Society*, 1–19.

Kohn, R., Ansley, C. F., and Wong, C.-M. (1992). Nonparametric spline regression with autoregressive moving average errors. *Biometrika*, *79*(2), 335–346.

Kokkala, J. *et al.* (2016). Particle and Sigma-Point Methods for State and Parameter Estimation in Nonlinear Dynamic Systems.

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.

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American statistical association*, *89*(425), 278–288.

Kontoghiorghes, E. J. (2005). *Handbook of parallel computing and statistics.* CRC Press.

Krivobokova, T., Crainiceanu, C. M., and Kauermann, G. (2008). Fast adaptive penalized splines. *Journal of Computational and Graphical Statistics*, *17*(1), 1–20.

Larson, S. C. (1931). The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, *22*(1), 45.

LaViola, J. J. (2003). A comparison of unscented and extended Kalman filtering for estimating quaternion motion. In *American Control Conference, 2003. Proceedings of the 2003*, Volume 3, 2435–2440. IEEE.

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, J. S., Liang, F., and Wong, W. H. (2000). The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, *95*(449), 121–134.

Liu, Y., Suo, J., Karimi, H. R., and Liu, X. (2014). A filtering algorithm for maneuvering target tracking based on smoothing spline fitting. In *Abstract and Applied Analysis*, Volume 2014. Hindawi Publishing Corporation.

Liu, Z. and Guo, W. (2010). Data driven adaptive spline smoothing. *Statistica Sinica*, 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.

Mahendran, N., Wang, Z., Hamze, F., and De Freitas, N. (2012). Adaptive MCMC with Bayesian optimization. In *Artificial Intelligence and Statistics*, 751–760.

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.

McDonald, M. (2006). *Intelligent transport systems in Europe: opportunities for future research.* World Scientific.

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.

Neal, R. M. *et al.* (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, *2*(11).

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308–313.

Opsomer, J., Wang, Y., and Yang, Y. (2001). Nonparametric regression with correlated errors. *Statistical Science*, 134–153.

Oussalah, M. and De Schutter, J. (2001). Adaptive Kalman filter for noise identification. In *Proceedings of the International Seminar on Modal Analysis*, Volume 3, 1225–1232. KU Leuven; 1998.

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.

Peng, G. (1983). Necessary and sufficient conditions on Linearly Independent Functions. *Journal of Jishou University(Natural Sciences Edition)*, (0), 005.

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.

Poyiadjis, G., Doucet, A., and Singh, S. S. (2005). Maximum likelihood parameter estimation in general state-space models using particle methods. In *Proc of the American Stat. Assoc.* Citeseer.

Ran, C. and Deng, Z. (2010). Self-tuning weighted measurement fusion Kalman filter and its convergence. *Journal of Control Theory and Applications*, *8*(4), 435–440.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, Volume 1. MIT press Cambridge.

Rhodes, I. (1971). A tutorial introduction to estimation and filtering. *IEEE Transactions on Automatic Control*, *16*(6), 688–706.

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. and Rosenthal, J. S. (2009). Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, *18*(2), 349–367.

Roberts, G. O., Rosenthal, J. S., *et al.* (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical science*, *16*(4), 351–367.

Rubin, D. B. (2004). *Multiple imputation for nonresponse in surveys*, Volume 81. John Wiley & Sons.

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003). *Semiparametric regression.* Number 12. Cambridge university press.

Särkkä, S. (2013). *Bayesian filtering and smoothing*, Volume 3. Cambridge University Press.

Schoenberg, I. J. (1964). Spline functions and the problem of graduation. *Proceedings of the National Academy of Sciences*, *52*(4), 947–950.

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.

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.

Sorenson, H. W. (1985). *Kalman filtering: theory and application.* IEEE.

Speckman, P. L. and Sun, D. (2003). Fully Bayesian spline smoothing and intrinsic autoregressive priors. *Biometrika*, *90*(2), 289–302.

St-Pierre, M. and Gingras, D. (2004). Comparison between the unscented Kalman filter and the extended Kalman filter for the position estimation module of an integrated navigation information system. In *Intelligent Vehicles Symposium, 2004 IEEE*, 831–835. IEEE.

Stavropoulos, P. and Titterington, D. (2001). Improved particle filters and smoothing. In *Sequential Monte Carlo Methods in Practice*, 295–317. 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*.

Syed, A. R. (2011). A review of cross validation and adaptive model selection.

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.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, *82*(398), 528–540.

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.

Tierney, L. and Mira, A. (1999). Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in medicine*, *18*(1718), 2507–2515.

Trevor Hastie, Robert Tibshirani, J. F. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition.* Springer-Verlag.

Turitsyn, K. S., Chertkov, M., and Vucelja, M. (2011). Irreversible Monte Carlo algorithms for efficient sampling. *Physica D: Nonlinear Phenomena*, *240*(4), 410–414.

Tusell, F. *et al.* (2011). Kalman filtering in R. *Journal of Statistical Software*, *39*(2), 1–27.

VanDerwerken, D. N. and Schmidler, S. C. (2013). Parallel markov chain monte carlo. *arXiv preprint arXiv:1312.7479*.

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. (1978). Improper priors, spline smoothing and the problem of guarding against model errors in regression. *Journal of the Royal Statistical Society. Series B (Methodological)*, 364–372.

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 Wang, Y. (1990). When is the optimal regularization parameter insensitive to the choice of the loss function? *Communications in Statistics-Theory and Methods*, *19*(5), 1685–1700.

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.

Wakefield, J. (2013). *Bayesian and frequentist regression methods.* Springer Science & Business Media.

Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, 153–158. Ieee.

Wand, M. and Gutierrez, R. (1997). Exact risk approaches to smoothing parameter selection. *Journal of Nonparametric Statistics*, *8*(4), 337–354.

Wang, Y. (1998). Smoothing spline models with correlated random errors. *Journal of the American Statistical Association*, *93*(441), 341–348.

Wecker, W. E. and Ansley, C. F. (1983). The signal extraction approach to nonlinear regression and spline smoothing. *Journal of the American Statistical Association*, *78*(381), 81–89.

West, M. (1993). Mixture models, Monte Carlo, Bayesian updating, and dynamic models. *Computing Science and Statistics*, 325–325.

Whittaker, E. T. (1922). On a new method of graduation. *Proceedings of the Edinburgh Mathematical Society*, *41*, 63–75.

Wolberg, G. (1988). Cubic spline interpolation: A review.

Woodbury, M. A. (1950). Inverting modified matrices. *Memorandum report*, *42*(106), 336.

Wu, X.-L., Sun, C., Beissinger, T. M., Rosa, G. J., Weigel, K. A., de Leon Gatti, N., and Gianola, D. (2012). Parallel Markov chain Monte Carlo-bridging the gap to high-performance Bayesian computation in animal breeding and genetics. *Genetics Selection Evolution*, *44*(1), 29.

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.

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.

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.