Bayesian inference for Short term Traffic Forecasting

A thesis submitted to the University of Dublin, Trinity College
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Department of Statistics, Trinity College Dublin

April 2013

Tiep K. Mai

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

The copyright belongs jointly to the University of Dublin and Tiep K. Mai.

Tiep K. Mai

Dated: August 29, 2013

Abstract

In intelligent transport systems, short term traffic forecasting is one of the most important problems, reflecting the network state in the near future and feeding information to other application modules. Even though there have been quite a lot of works in this area, most of them are univariate models which may not be able to exploit the spatial relationship of traffic variables. So, this thesis explores the domain of two renowned modelling classes, the auto regressive moving average model and the dynamic model, taking into account the spatial dependency.

The sparse-form vector autoregressive moving average model is applied to the short term traffic forecasting problem with different preprocessing methods. Network information is used to constrain the matrix parameters of the model, reducing the number of parameters. For the estimation problem, an improved MCMC method is proposed to tackle the variable correlation problem, using the marginalisation and the correlation direction information. Multi-step-ahead prediction results of different models are compared with two Dublin traffic datasets.

A second model, consisting of four sub-models, targets the multi-step-ahead flow prediction for traffic data with incidents, where the data pattern may shift unexpectedly. The model is designed to satisfy the scalability property so that the inference of each component can be done conditionally independently. Furthermore, each sub-model supports sequential inference, which is essential for real-time applications. The first two sub-models are analysed with a VISSIM dataset and the discussion of the last two is given at the end.

A sequential approximation method is developed for both the state vector and the parameters of the dynamic model that is part of the second model. To avoid the degeneracy problem of the sampling-based particle filter, the method uses a continuous functional approximation which is a modified implementation of the iterated Laplace approximation (Bornkamp, 2011*a*). Both the modified iterated Laplace approximation and the sequential approximation method are illustrated and analysed with several examples.

Acknowledgements

First of all, I would like to express my utmost gratitude to my supervisor Simon Wilson. For all four years, he has been very helpful and provided invaluable guidance to me who knew almost nothing about statistics at that time. Whenever I have a problem or an idea, Simon always has time to listen and give encouraging advice, no matter how ridiculous the subject and how incomprehensible my English explanation. I am also very grateful to many research opportunities to international conferences that he has granted. Thanks to Simon, I am open to world of statistics and have found something that really interests me.

Next, I sincerely thank Bidisha Ghosh in the Department of Civil, Structural and Environmental Engineering for her essential assistance, especially in the field of transportation and for all supplied traffic datasets used in this thesis.

I am very thankful to all wonderful people in our department. In particular, I have learnt many things of various topics from the group meetings organised nicely by Brett Houlding. The discussions I had with Ji Won Yoon and Chaitanya Joshi during the first two years clarified my thought greatly. I also thank Louis Aslett for setting up and maintaining the STATICA cluster, which has saved me a lot of time, and for the very useful programming guideline. The sequential inference part of this thesis is motivated by discussions with Arnab Bhattacharya who excels in this area, among other things. Susanne Schmitz is always my friend whom i have shared many joyful experiences with, especially in the APTS course. Another big thank to Jason Wyse, Thinh Doan, Sen Rordin, Cristina De Persis, Shuaiwei Zhou and everyone else in TCD, also to all my friends in Ireland, Korea and Vietnam.

I am indebted to my parents, Hai Mai and Phuong Pham, for their long-lasting encouragement and support. It is their love that makes me who I am today. Finally, my dearest wife, Thuy Truong, and my little Moon have my huge special thank for being by my side all the time.

This thesis is made possible by the kindly four-year SFI funding of the STATICA project.

Tiep K. Mai

Trinity College Dublin

April 2013

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The transportation infrastructure is becoming more and more complex and the number of vehicles is increasing in cities across the world. To meet this challenge, transportation management systems have been improved significantly over the past few decades, most recently through intelligent transport systems (ITS). Supported by other advanced technologies such as global positioning systems, wireless networks, parallel computational systems, etc., ITS are capable of collecting the data from traffic networks, analysing the network state and giving out management decisions in real time.

One of the most important modules in ITS is the short term forecasting module which continuously updates the multi-step-ahead prediction of the network state. This network state prediction is then fed to other decision modules such as route optimisation or traffic light sequencing. Using the prediction state, instead of the current state, as a decision module input better reflects the network in the near future, resulting in more compatible decision results. In this thesis, our work thesis focuses on the short term forecasting of traffic flow which is one of the most important variables in transportation management.

1.1 Statistical motivation

With the abundant data from traffic networks, we want to construct a spatial temporal model for multi-step-ahead flow prediction, taking into account the spatial relationship between traffic variables. In short term traffic forecasting (STFF), only a few works explore the spatial dependency, including Chandra and Al-Deek (2009); Min and Wynter (2011); Queen and Albers (2009); Anacleto Junior et al. (2013*b*) and among these works, only Min and Wynter (2011) analyses the multi-step-ahead prediction.

Among many STFF models, there are two important model classes, the autoregressive moving average (ARMA) model and the dynamic model (DM), which have been studied extensively in the statistics literature. So, we explore these two model classes and propose suitable models for the multi-step-ahead flow forecasting problem, taking into account the spatial dependency of traffic networks.

During our work, we realised that in order to meet the constraint of real time applications and the expansion of traffic networks, the target model should be scalable, e.g. its inference process can be broken down to multiple independent tasks, and support sequential inference. Both properties may be achieved by careful modelling and we are particular interested in sequential inference of the DM.

It is noted that for the DM, the sampling-based particle filter is the most popular sequential inference method but very prone to the degeneracy issue and usually only supports inference of the state vector. Sequential inference for parameters is notoriously difficult and only addressed by some sampling-based works, which again suffer from the degeneracy. Hence, this reason is our driving force towards a new sequential continuous approximation for both state vector and parameters.

1.2 Outline of the thesis and contributions

This thesis is divided into following chapters:

Chapter 2: Bayesian statistics and time series modelling

This chapter firstly reviews the basis of statistical inference and the Bayesian approach. Secondly, we move on with the underlying theory of Monte Carlo methods and its two most important classes, importance sampling and Markov chain Monte Carlo (MCMC). Next, popular function approximations such as Laplace approximation and variational Bayes are given. Then, the chapter discusses classical time series models and dynamic models, together with related problems such as parameter estimation, filtering, smoothing and prediction. Finally, we conclude the chapter with a discussion about sequential inference of dynamic linear and non-linear models for both state vector and parameters.

Chapter 3: Transportation networks and some related problems

The summary of previous works in the STFF problem is given in this chapter, with various references of different models such as the ARMA model, the DM, neural networks and non-parametric approaches. Then, we illustrate two related problems in transportation, the road traffic control problem and the estimation of number of vehicles within a road link, for the sake of spatial-temporal modelling.

Chapter 4: The vector autoregressive moving average model (VARMA) for the short term traffic flow forecasting

This chapter contains two research contributions of the thesis. Firstly, after presenting the sparse-form VARMA model with a user-defined parameter structure, we propose an improved MCMC scheme targeting the variable correlation problem of the VARMA model. The performance of this algorithm is then analysed with a simulated dataset. Secondly, we apply the VARMA model and MCMC to two Dublin datasets, employing three different preprocessing methods and imposing the network structure into the parameters. Multi-step-ahead prediction results of different models are compared with a concluding discussion on the STFF problem and the effect of traffic incidents.

Chapter 5: Sequential inference with the iterLap approximation

The first contribution of this chapter is applying modifications to the iterated Laplace approximation (Bornkamp, 2011*a*) to improve the method performance. Both the original and modified algorithms are extensively compared with several examples. Then, based on the iterated Laplace approximation, we propose a new continuous sequential approximation for both state vector and parameters and analyse the approximation performance with some simulated examples.

Chapter 6: Spatial-temporal modelling of transportation networks

In this chapter, a spatial-temporal model is proposed for multi-step-ahead forecasting, satisfying the scalability property. The model consists of four sub-models each of which corresponds to a particular section of traffic networks and supports sequential inference. Analysis of the link-outflow and junction sub-models is given and the sequential inference method of the previous chapter is applied to the junction sub-model. Finally, discussion of the root and link-vehicle sub-models is provided.

Chapter 7: Conclusion

The last chapter concludes the thesis with several remarks of future work.

Chapter 2

Bayesian statistics and time series modelling

In this chapter, firstly, a brief overview of statistical inference and the Bayesian approach is given. After that, Monte Carlo methods are introduced as asymptotic approximations for many cases where the inference is intractable. Next, the Laplace approximation and its extended version, the iterated Laplace approximation, are discussed as fast approximation solutions, as well as supportive elements of other methods. Then, as a background for subsequent chapters, this chapter outlines a summary of time series modelling, including variations of the autoregressive moving average model and the dynamic model. Finally, sequential inference is presented as a tool for real time problems where statistical inference is updated iteratively by the arrival of new data, under a time constraint for computation of the inference.

2.1 Statistical inference

Statistics is about discovering and exploiting patterns in data. A statistical solution is found when such a discovered pattern can accommodate user's needs. Imagine that a business manager wants to learn about the pattern of sale items and customer's interests: How much can a customer of a specific class spend on average? Which items are usually bought together? Which items are sold the most at the specific time? By learning these, the manager can develop a new marketing strategy with the most benefit.

The initial step in solving a statistical problem is hence *statistical modelling*, in which an analyst projects his or her own subjective idea of the target relationship into mathematical formulae. One simple, but perhaps useless in terms of beneficial use, example is a relationship between a person's walking stride length and other variables such as leg length, weight, etc. So, one modeller may expect that the stride length is proportional with the leg length and the inverse weight and assume that on average, the stride length is the linear combination of two other variables. This element, the mean, is constructed so that by the modeller's expectation, it is as close to the true observed value as possible within the constraint of data availability.

The observed value, however, will never be exactly the same as the formulated mean and the modeller has to add an uncertainty term, a *random noise*, allowing the observed value to stray from the mean. To answer the question where that uncertainty comes from, one needs to examine the data availability. In the above example, the walking stride could depend on the leg length, person's weight, terrain type, the location and the person's mood which are in turn derived from other unknown variables. Theoretically, there may exist a very sophisticated and perfect formula that links all the variables together but because only a subset of data is collected and that perfect formula is never found, the modeller can guess the true value to a certain degree of error by using a simpler formula. Also, the uncertainty is caused by the measurement error of the obtained dataset.

By now, notice that the obtained data is divided into an *explanatory* part (*covariates, input*) and an *observation* part (*output*). The observation is a function of the explanatory variables with a random error.

When using a random noise, the modeller should choose its shape, the noise *distribution*, according to its properties in the target problem: The noise is skewed to the positive side; the magnitude of the first noise is much larger than the other noise; the two noise values seem correlated, e.g. when one noise is large, so is the other one. When unsure about the noise distribution, the modeller should select a basic and convenient form, simplifying subsequent steps of the statistical inference procedure.

Most of the time, the defined model has unknown parameters, linking the observation and the explanatory variables. For example, the modeller may know that the stride length is proportional to the leg length but is clueless about the slope of such a linear pattern. Estimating such unknown parameters (*parameter estimation*) by using the obtained dataset is carried out after the statistical modelling, realising the full subjective idea of the modeller towards the target problem.

From this step, statistics is divided into two different schools of thought: the *frequentist* (or the *classical*) and the *Bayesian*. In the frequentist approach, it is assumed that there is an unknown fixed value for the parameter and the probability of one outcome of an event is the proportion of times such an outcome re-occurs when repeating the experiment infinitely. Hence, the probability only exists for the repeatable event. Also, the inference result is completely defined by the data, without any regard to subjective belief.

On the other hand, Bayesian statistics regards the probability as the degree of individual belief. Hence, although the event is not repeatable and the frequency interpretation does not exist, one can somehow guess or evaluate how likely an outcome can occur based on the experience and apply a probability distribution for that event. Additionally, a probability of initial subjective belief, the prior distribution, is applied on unknown parameters, allowing the incorporation of an expert's experience into the statistical solution.

A discussion of the frequentist approach is outside the scope of this thesis. The interested readers should refer to the excellent book of Casella and Berger (2002) or the rigorous reference of Cox (2006). For Bayesian inference, there are very comprehensive references: Box and Tiao (1992) and Gelman et al. (2003). Valuable texts for a good introduction of theory of probability, random processes, include: Grimmett and Stirzaker (2001) and Leon-Garcia (2008).

2.1.1 The Bayesian approach

Bayes' theorem is the most fundamental tool in Bayesian statistics, transforming from initial belief to post-belief about the parameter θ after taking the information contained in the data into account.

$$p_{\theta|y}(\theta) = \frac{p_\theta(\theta)p_{y|\theta}(y)}{p_y(y)} \tag{2.1}$$

$$\propto p_\theta(\theta)p_{y|\theta}(y). \tag{2.2}$$

The likelihood

In the frequentist interpretation, the *likelihood* term, $p_{y|\theta}(y)$, measures the proportion of times that an outcome occurs if the random event y is repeated for an infinite number of times under the fixed value of the unknown random parameter θ.

In the Bayesian setting, the likelihood is extended to non-repeatable event. For example, the question "What is the chance for that particular person to pass the final exam?" seems reasonable but the frequency interpretation can never be obtained due to the uniqueness of the event. However, if you know that person and the test well, you may have an idea about the answer. In such a case, the likelihood expresses the subjective preference value for every outcome of the event.

Notice that when conditioning on the observed value y, the likelihood function, $L(\theta|y) = p_{y|\theta}(y)$, a function with respect to θ, is not a probability and its integral over the parameter space is not equal to 1.

The prior

As mentioned previously, in Bayesian statistics, the unknown parameter is not fixed and can take any value in a predefined support. Hence, one can apply a probability on the unknown parameter, the *prior* $p_\theta(\cdot)$, to illustrate one's surmise about the value of that parameter. Injecting expert opinion or background information into a statistical inference may enhance the result, e.g. the result converges faster, if the subjective idea is supported by the data.

Another strategy of prior selection is to make it as vague and non-informative as possible. This can be done by using an improper uniform distribution or a distribution with very large variance (Gelman et al., 2003). In contrast to the above purpose, one usually utilises this strategy to make sure the result is only determined by the observation, reducing the influence of prior subjective belief.

There is also a concern about inference consistency in a situation where statistical procedures are implemented by different reparameterisations. Obviously, two different statistical processes of the same model with different priors lead to different answers and this feature may be unwanted scientifically. To remedy this issue, Jeffreys (1946) proposed the *Jeffreys' prior*:

$$p_\theta(\theta) = |\mathcal{I}(\theta)|^{1/2}, \tag{2.3}$$

with the *Fisher information* $\mathcal{I}(\theta)$:

$$\mathcal{I}(\theta) = -\,\mathrm{E}_{y|\theta}\left[\frac{\partial^2 \log(p_{y|\theta)}(y))}{\partial\theta^2}\right], \tag{2.4}$$

where $-\frac{\partial^2 \log(p_{y|\theta)}(y))}{\partial\theta^2}$ is called the *observed information*. This rule ensures that the priors of different classes of reparameterisations of the parameters are actually equal, hence yielding the same results.

Finally, the prior can be chosen to simplify the implementation of the inference process. This idea, *conjugacy*, is explained a later.

The term $p_y(y)$ is the *normalising constant* with respect to θ and often ignored when calculating the non-normalised left hand side (LHS) of Equation (2.1).

$$p_y(y) = \int p_{\theta,y}(\theta, y)d\theta = \int p_\theta(\theta)p_{y|\theta}(y)d\theta. \tag{2.5}$$

The posterior

The LHS term of Equation (2.1), $p_{\theta|y}(\theta)$, is the *posterior*, indicating how likely the unknown parameter is, with the obtained knowledge from the data.

Most Bayesian inference tasks involve expectations of the posterior. For summary purposes, a statistician may want to know the expectation $\mathrm{E}_{\theta|y}(\theta)$ or the variance $\mathrm{Var}_{\theta|y}(\theta)$:

$$\mathrm{E}_{\theta|y}(\theta) = \mu_{\theta|y} = \int \theta\; p_{\theta|y}(\theta)d\theta, \tag{2.6}$$

$$\begin{aligned}\mathrm{Var}_{\theta|y}(\theta) &= \int (\theta - \mu_{\theta|y})^2\; p_{\theta|y}(\theta)d\theta \\ &= \mathrm{E}_{\theta|y}[(\theta - \mu_{\theta|y})^2],\end{aligned} \tag{2.7}$$

or in the case where only a random part θ_a (*parameter of interest*) of vector $\theta = (\theta_a, \theta_b)$ is needed. One may want to marginalise out the *nuisance parameter* θ_b:

$$\begin{aligned}p_{\theta_a|y}(\theta_a) &= \int p_{\theta_a,\theta_b|y}(\theta_a, \theta_b)d\theta_b \\ &= \int p_{\theta_a|y,\theta_b}(\theta_a)\; p_{\theta_b|y}(\theta_b)d\theta_b \\ &= \mathrm{E}_{\theta_b|y}[p_{\theta_a|y,\theta_b}(\theta_a)].\end{aligned} \tag{2.8}$$

The prediction formula:

$$\begin{aligned}p_{y_2|y}(y_2) &= \int p_{y_2|\theta}(y_2)\; p_{\theta|y}(\theta)d\theta \\ &= \mathrm{E}_{\theta|y}[p_{y_2|\theta}(y_2)]\end{aligned} \tag{2.9}$$

gives the probability distribution of a future observation y_2. Such integrations rarely have closed form solutions and statisticians usually rely on the *sampling methods* of Section 2.2, the *functional approximations* of Section 2.3 or the *numerical quadrature* (Monahan, 2011,chap. 10). The last class suffers significantly from the *curse of dimensionality* and is not discussed in this thesis.

The exponential family and conjugacy

The *exponential family* is a very broad class of distributions, including the *normal*, *multinomial*, *Dirichlet*, *Wishart*, *inverse Wishart* distributions and many others. A *random variable* y belongs to the exponential family with a parameter θ if its density follows the form:

$$p_{y|\theta}(y) = h(y)g(\theta)\exp[\eta(\theta)^T u(y)], \tag{2.10}$$

where the vector $u(y)$ is the *sufficient statistic* and the vector $\eta(\theta)$ is the *natural parameter*. The exponential family has some interesting properties like sufficiency, reparameterisation and factorisation of parameters of interest and nuisance parameters, which are essential in the frequentist approach (Cox, 2006,chap. 2). In Bayesian statistics, the existence of a conjugate prior for the exponential family distribution is one of the most important features (Bishop, 2006,chap. 2).

A prior $p_\theta(\theta)$ is conjugate, with respect to the likelihood $L(\theta|y) = p_{y|\theta}(y)$, when the resulting posterior $p_{\theta|y}(\theta)$ follows the same functional form as the prior. Consider a prior with fixed hyperparameters a, b of the form:

$$p_{\theta|a,b}(\theta) = r(a,b)g(\theta)^b \exp[\eta(\theta)^T a]. \tag{2.11}$$

When combined with the likelihood of Equation (2.10), Equation (2.11) leads to the posterior:

$$\begin{aligned} p_{\theta|y,a,b}(\theta) &\propto r(a,b)g(\theta)^{b+1}\exp[\eta(\theta)^T(a+u(y))] \qquad (2.12)\\ &\propto r_2(a_2,b_2)g(\theta)^{b_2}\exp[\eta(\theta)^T a_2], \end{aligned}$$

which yields the same form of Equation (2.11). Notice that with fixed parameter b, Equation (2.11) also belongs to the exponential family.

Random processes and stationarity

A *random process* X is a collection of random variables $\{x_t : t \in T\}$ with an index set T. This thesis focuses on discrete time series in which the ordered time index set is used. One of the most important concepts of random processes is the *strictly stationary* property.

Definition 1 (Strict stationarity) *A random process X is strictly stationary if and only if the joint distribution of $(x_{t_1}, ..., x_{t_k})$ is independent of the time origin:*

$$p_{x_{t_1},...,x_{t_k}}(x_{t_1}, ..., x_{t_k}) = p_{x_{t_1+h},...,x_{t_k+h}}(x_{t_1}, ..., x_{t_k}), \tag{2.13}$$

for all k, h, $(t_1, .., t_k)$ and $(x_{t_1}, ..., x_{t_k})$.

Some random processes do not satisfy the above requirement, but the weaker version, *stationarity (weakly stationarity or second-order stationarity).*

Definition 2 (Stationarity) *A random process X is stationary if and only if its expectation and covariance matrix are independent of the time origin:*

$$\mathrm{E}_{x_{t_1}}(x_{t_1}) = \mathrm{E}_{x_{t_1+h}}(x_{t_1+h}), \tag{2.14}$$

$$\mathrm{Cov}_{x_{t_1},x_{t_2}}(x_{t_1}, x_{t_2}) = \mathrm{Cov}_{x_{t_1+h},x_{t_2+h}}(x_{t_1+h}, x_{t_2+h}), \tag{2.15}$$

for all k, h and (t_1, t_2).

Model comparison

When there are finite different competing models, the problem of selecting the most suitable arises. One popular method is by checking the *Bayes factor*:

$$K_1 = \frac{p_{y|M=M_2}(y)}{p_{y|M=M_1}(y)} = \frac{\int_{\theta_2} p_{y|\theta_2,M=M_2}(y)\ p_{\theta_2|M=M_2}(\theta_2)d\theta_2}{\int_{\theta_1} p_{y|\theta_1,M=M_1}(y)\ p_{\theta_1|M=M_1}(\theta_1)d\theta_1}, \tag{2.16}$$

where the likelihood has been marginalised with respect to the parameter θ_1 (θ_2) in each model M_1 (M_2). In a full Bayesian setting with a prior on each model $p_M(\cdot)$, the Bayes factor can be replaced by the *posterior odd*:

$$K_2 = \frac{p_{M|y}(M_2)}{p_{M|y}(M_1)} = \frac{p_{y|M=M_2}(y)p_M(M_2)}{p_{y|M=M_1}(y)p_M(M_1)}. \tag{2.17}$$

Ratios K_1 (K_2) are then usually checked against some predefined values to justify the selection of a model with a high value of likelihood or posterior. MacKay (2002,chap.

28) discusses the model evidence in the Bayes factor and how the marginal likelihood $p_{y|M=M_i}(y)$ penalises the model complexity. The problem of calculating the Bayes factor is non-trivial and is referred to Kass and Raftery (1995) and Weinberg (2012). Alternatively, methods such as *Akaike information criterion (AIC)* or *Bayesian information criterion (BIC)*, which are based on penalised likelihood, can be used:

$$AIC(M_i) = 2[-\log(p_{y|\theta=\widehat{\theta},M_i}(y)) + p], \tag{2.18}$$

$$BIC(M_i) = 2[-\log(p_{y|\theta=\widehat{\theta},M_i}(y)) + p\log(n)], \tag{2.19}$$

where $\widehat{\theta}$ is the *maximum likelihood estimation (MLE)*, p is the dimension of vector θ and n is the number of observations in vector y. The model with a low AIC (BIC) value is preferred. Derivations and interpretations of AIC and BIC are referred to Wood (2006,chap. 2) and Davison (2003,chap. 4,11).

Still, all above methods may not be intuitive enough in terms of performance interpretation. Hence, one practical way for model selection is comparing both the prediction capabilities and the running time or inference cost of corresponding models. For example, the model with the best prediction value within the running time limit interval may be chosen.

2.2 Monte Carlo methods

As mentioned, most Bayesian statistical problems are related with the expectations which have no closed-form solution. Hence, this section is devoted to the *Monte Carlo method*, one of the most dominant inference methods in Bayesian statistics. The foundation of the Monte Carlo method is explained first then there comes two techniques: *importance sampling* and *Markov chain Monte Carlo (MCMC)*.

2.2.1 Asymptotic theorems

The justification of Monte Carlo methods is based on two theorems: the *strong law of large numbers (SLLN)* and the *central limit theorem (CLT)* (Grimmett and Stirzaker, 2001,chap. 5, 7).

Theorem 1 (Strong law of large numbers) *Let x_i $(i = 1 : n)$ be independent and identically distributed (iid) random variables generated from the density $p_x(\cdot)$: $x_i \overset{iid}{\sim}$*

$p_x(\cdot)$ with $\mathrm{E}_x(x) = \mu < \infty$ and $\mathrm{E}_x(x^2) < \infty$. Then:

$$\overline{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i \xrightarrow{as} \mu. \tag{2.20}$$

Theorem 2 (Central limit theorem) *Let $x_i \overset{iid}{\sim} p_x(\cdot)$ $(i = 1 : n)$ with $\mathrm{E}_x(x) = \mu < \infty$ and $\mathrm{Var}_x(x) = \sigma^2 < \infty$, then:*

$$\overline{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i \xrightarrow{d} N(\mu, \frac{\sigma^2}{n}), \tag{2.21}$$

where "$\xrightarrow{as}$" denotes *almost sure convergence* and "$\xrightarrow{d}$" denotes *convergence in distribution* (Grimmett and Stirzaker, 2001,chap. 7).

These two theorems complement each other and have different interpretations. The SLLN theorem states that almost every realisation of the sequence $\overline{x}_{1:n} = (\overline{x}_1, ..., \overline{x}_n)$ converges to the same mean μ, justifying the convergence of repeated generation of the whole sequence $\overline{x}_{1:n}$. On the other hand, according to the CLT theorem, the random error at index n, by a random realisation of the sequence, follows an approximate normal distribution. Furthermore, the variance of this random error is shrinking with respect to the number of generated variables n. The Monte Carlo convergence rate is hence $n^{-1/2}$ (The largest order of the noise standard deviation $\sigma n^{-1/2}$, against n).

Sampling methods

By using the SLLN and the CLT, an expectation function can be approximated by the sample mean of the independent sequence:

$$\begin{aligned} \mathrm{E}_x(f(x)) &= \int f(x)\ p_x(x)dx \\ &\approx \frac{1}{n}\sum_{i=1}^{n} f(x_i), \end{aligned} \tag{2.22}$$

with $x_i \overset{iid}{\sim} p_x(\cdot)$. So, the focus turns to the sampling of the density $p_x(\cdot)$ ($p_x(\cdot)$ and $q_x(\cdot)$ denote normalised and non-normalised densities accordingly) . For standard probability densities like uniform, normal, multinomial, Dirichlet, and Wishart, there are efficient classical sampling methods already.

For an arbitrary density $p_x(\cdot)$, *inverse transform sampling* generates a uniform random variable u from $U(0, 1)$ and calculates $x = F_x^{-1}(u)$ where $F_x(\cdot)$ is the *cumulative distribution function* of the density $p_x(\cdot)$.

In the *ratio-of-uniforms* method, a predefined region of random variables $\{(u, v) : D = (u, v) : u^2 \leq q_x(v/u)\}$ is enveloped by a simple region E from which (u, v) is uniformly generated. If (u, v) is in D, the sample $x = v/u$ is accepted; otherwise, the process repeats with a new uniform sample of (u, v).

The *rejection sampling* method is also based on a non-normalised envelope function $q_{r;x}(\cdot) : q_{r;x}(\cdot) \geq q_x(\cdot)$. The variable x is repeatedly sampled from $q_{r;x}(\cdot)$ and accepted with rate $q_x(\cdot)/q_{r;x}(\cdot)$. *Adaptive rejection sampling* (Gilks and Wild, 1992) is a very useful extension for a log concave density $q_x(\cdot)$, enveloped by an exponential piecewise linear function $q_{r;x}(\cdot)$. When a sample is rejected, it is added as a new knot of the piecewise linear function, making the envelope function closer to the target function; hence, the rejection rate decreases significantly. *Adaptive rejection Metropolis sampling* (Gilks et al., 1995) can relax the log-concave condition by correcting the sample with a Metropolis-Hastings step.

All the above methods are mostly used for a univariate density (or a low dimensional density) as it is hard to find an easy-to-sample envelope function in a high dimensional space. More details of these methods can be found in Gilks et al. (1996,chap. 5) and Grimmett and Stirzaker (2001,chap. 4).

2.2.2 Importance sampling

One popular sampling method which can be applicable to multivariate density functions is the *importance sampling (IS)* method (Geweke, 1989). For the expectation of Equation (2.22), rather than sampling directly from density $p_x(\cdot)$, which for some reason is impossible, the IS algorithm samples from a proposal density function $p_{r;x}(\cdot)$ and corrects the samples with a weight function. The procedure is in Algorithm 1.

The IS approximation is actually a consequence of the Monte Carlo approximation,

Algorithm 1 Importance sampling

1. Generate n iid samples x_i from a non-normalised density $q_{r;x}(\cdot)$.

2. Define the weight function $w(x) = q_x(x)/q_{r;x}(x)$ and evaluate: $w_i = w(x_i)$. Normalise the weight by: $w_{n;i} = w_i/(\sum(w_i))$.

3. Approximate the target expectation by:

$$\mathrm{E}_x(f(x)) \approx \sum_{i=1}^{n} w_{n;i} f(x_i). \tag{2.23}$$

---

by the derivation:

$$\begin{aligned}
\mathrm{E}_x(f(x)) &= \frac{\int f(x)\ q_x(x)dx}{\int q_x(x)dx} \\
&= \frac{\int f(x)\ \frac{q_x(x)}{p_{r;x}(x)} p_{r;x}(x)dx}{\int \frac{q_x(x)}{p_{r;x}(x)} p_{r;x}(x)dx} \\
&= \frac{\int f(x)\ \frac{q_x(x)}{q_{r;x}(x)} p_{r;x}(x)dx}{\int \frac{q_x(x)}{q_{r;x}(x)} p_{r;x}(x)dx} \\
&\approx \frac{\sum_{i=1}^{n} w_i f(x_i)}{\sum_{i=1}^{n} w_i} \\
&\approx \sum_{i=1}^{n} w_{n;i} f(x_i).
\end{aligned}$$

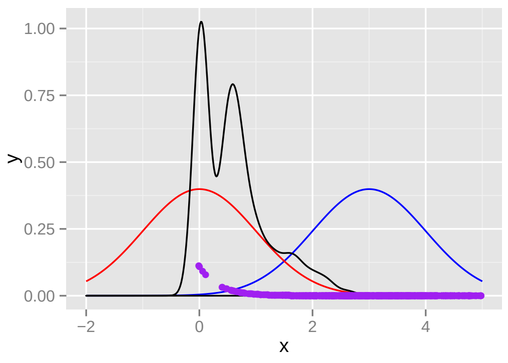
By this algorithm, the normalised density $p_x(\cdot)$ can be approximated by a discrete density:

$$p_x(x) \approx \sum_{i=1}^{n} w_{n;i} \mathbb{1}(x = x_i), \tag{2.24}$$

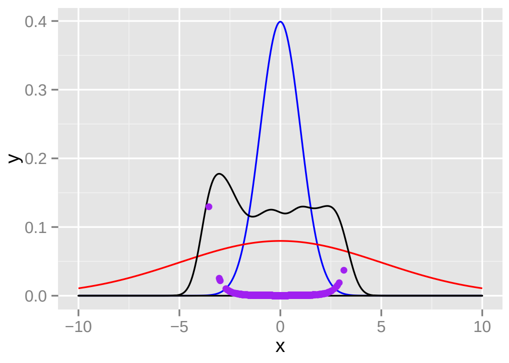
where $\mathbb{1}(\cdot)$ is the *indicator function*.

The efficiency of the IS method

When using the IS method, one may want the proposal density $p_{r;x}(\cdot)$ to be as "close" ("similar") as possible to the target density $p_x(\cdot)$ so that the samples x_i look as if they are generated from the target density. If $p_{r;x}(\cdot) = p_x(\cdot)$ (impossible in practise as it would remove the need for IS), all the weights are equal and the variance of the weights is zero: $\mathrm{Var}_{r;x}(w(x)) = 0$.



(a) $p_x(\cdot) = N(0,1)$; $p_{r;x}(\cdot) = N(3,1)$



(b) $p_x(\cdot) = N(0,5)$; $p_{r;x}(\cdot) = N(0,1)$

Fig. 2.1: Examples of bad proposal functions in the IS method: The red curves are the target densities. The blue curves are the proposal densities. The pink dots are 1000 samples with the corresponding weights w_i. The black curves are the kernel density approximations of the samples x_i.

The IS samples may degenerate into only a few important samples, i.e. few samples have significant weights, resulting in a poor approximation when the target and proposal densities are so different. Examples of bad proposal functions are shown in Figure 2.1. The mean of $p_{r;x}(\cdot)$ in the left figure is misplaced while the variance of $p_{r;x}(\cdot)$ in the right figure is smaller than it should be. Hence, the samples of the chosen proposal should span the 95% probability interval of the target density and using an over-dispersed proposal is safer than using a focused proposal. To improve the IS method, usually a functional approximation of the target density (in Section 2.3) is chosen as the proposal, whenever it is possible. This *degeneracy* becomes worse in high dimensional space and in a sequential setting.

2.2.3 Markov chain Monte Carlo

This section briefly discusses the theory of MCMC and the two most popular methods: *Metropolis-Hastings algorithm* (Metropolis et al., 1953; Hastings, 1970) and *Gibbs sampling* (Geman and Geman, 1984). Further discussion of MCMC can be found in Gilks et al. (1996), Roberts and Rosenthal (2004) and Geyer (2011).

The basis of MCMC method

As mentioned before, this thesis focuses on a random process with an ordered discrete time index $(x_1, ..., x_n, ...)$, and a common *state space* S, i.e. all possible values of any random variable x_t $(1 \leq t \leq n)$ are in the set S. Such a random process can be specified by the probability of the initial random variable x_1, $p_{x_1}(\cdot)$, and the *transition density* (or *transition kernel*), $p_{x_t|x_{1:(t-1)}}(\cdot)$. The definition of *Markov chain* and its important properties are listed as follows:

Definition 3 (Markov chain and Markov property) *A random process (with an ordered time index and a common state space) is called a Markov chain if and only if its transition density satisfies the Markov property, i.e. the transition from x_{t-1} to x_t is independent of $x_{1:(t-2)}$:*

$$p_{x_t|x_{1:(t-1)}}(x_t) = p_{x_t|x_{t-1}}(x_t). \tag{2.25}$$

Definition 4 (Homogeneous Markov chain) *A Markov chain is homogeneous if and only if the transition density is time-independent:*

$$p_{x_t|x_{t-1}}(x_t) = p_{x_{t+k}|x_{t+k-1}}(x_t) \quad \forall k \in N. \tag{2.26}$$

For this section, $p_{y|x}(\cdot)$ is denoted as the transition density of a *homogeneous* Markov chain from the current state $x = a$ to the next state $y = b$: $p_{y|x=a}(b) = p_{x_t|x_{t-1}=a}(b)$ $\forall t$.

Definition 5 (Stationary density (stationary distribution) of a Markov chain) *A homogeneous Markov chain with a transition density $p_{y|x}(\cdot)$ on a common state space S is said to have a stationary density $\pi_x(\cdot)$ if and only if:*

$$\int_S \pi_x(a) p_{y|x=a}(b) dx = \pi_x(b). \tag{2.27}$$

Assume that the density of the initial state x_1 of a Markov chain is its *stationary density*: $p_{x_1}(\cdot) = \pi_x(\cdot)$, then that Markov chain is a strictly stationary random process.

The MCMC method generates samples from a target density $p_x(\cdot)$ by first designing a Markov chain (imposing the condition on the transition density) so that its stationary distribution is the target density: $\pi_x(\cdot) = p_x(\cdot)$. Then, starting at a single value of the random variable x_1, the method iteratively applies the Markov transition to obtain the whole realisation $x_{1:t}$. After many iterations, it is hoped that the marginal density of $p_{x_t|x_1}(\cdot)$ is "similar" to the stationary density $\pi_x(\cdot)$, and the target density $p_x(\cdot)$ (the conditions for this convergence will be mentioned later). Hence, the realised sample x_t of the Markov chain can be considered as a sample of the target density.

So, a rule to design a Markov chain to admit the target density $p(\cdot)$ as its stationary density is needed. However, using the definition of stationary density directly to impose the condition on the transition density is non-trivial; a simpler method is preferred. That is the reason the *detailed balance* comes into play.

Definition 6 (Detailed balance) *A transition density $p_{y|x}(\cdot)$ and a density $p_x(\cdot)$ are said to satisfy the detailed balance if and only if:*

$$p_x(a)p_{y|x=a}(b) = p_x(b)p_{y|x=b}(a) \quad \forall a, b \in S. \tag{2.28}$$

It can be seen that if $p_{y|x}(\cdot)$ and a density $p_x(\cdot)$ satisfy the detailed balance, then the Markov chain with transition density $p_{y|x}(\cdot)$ admits the density $p_x(\cdot)$ as its stationary density.

As the MCMC method starts with a single specified value (rather than a sample from the target density), it is needed that $p_{x_t|x_1}(\cdot)$ "converges" to the target density $p_x(\cdot)$ by some conditions. Furthermore, for computational efficiency, not only the last variable x_n of the Markov chain is picked up as a sample but the whole sequence $x_{r:n}$ are used (after a *burn-in* $x_{1:(r-1)}$). Hence, a version of the SLLN theorem for Markov chain (dependent sample) is needed.

The following definitions are written in an informal way for simplicity. A formal and rigorous description can be found in Roberts and Rosenthal (2004) and Meyn and Tweedie (2009).

- A Markov chain is *ϕ-irreducible* if and only if for all subsets A of the state space S (with positive measure $\phi(A) > 0$) and all $x \in S$, the probability that the Markov chain can reach the subset A, starting at x, is positive.

- A Markov chain is *Harris recurrent* if and only if X is ϕ-irreducible and for all subsets $A \subset S$ with positive *measure* $\phi(A) > 0$ and all $x \in S$, the probability that the Markov chain visits the subset A infinitely, starting at x, is equal to 1.

- A Markov chain is *periodic* with period $d \geq 2$ if and only if the state space S can be divided into d disjoint subspaces $S_{1:d}$ such that for all $a \in S_i$ $(i = 1 :$

$(d-1)$), the probability $\int_{S_{i+1}} p_{y|x=a}(b)\ db = 1$, and for all $a \in S_d$, the probability $\int_{S_1} p_{y|x=a}(b)\ db = 1$.

- A Markov chain is *aperiodic* if and only if it is not periodic with any period.

From these definitions, there are important results:

- If a Markov chain is ϕ-irreducible, aperiodic and has a stationary density then $p_{x_t|x_1}(\cdot)$ "converges" to the target density $p_x(\cdot)\ \forall x_1$(in the sense of *total variation distance*) (Roberts and Rosenthal, 2004)).

- If a Markov chain is Harris recurrent, aperiodic and has a stationary density $\pi(\cdot)$ then

$$\overline{f}_n = \frac{1}{n}\sum_{i=1}^{n} f(x_i) \xrightarrow{as} \mathrm{E}_x(f(x)) \quad \forall f \in L^1(\pi), \tag{2.29}$$

  where the expectation is taken with respect to the stationary density $\pi_x(\cdot)$; x_i are the samples produced by the Markov chain.

The Monte Carlo standard error

By the Markov chain CLT theorem, given in Meyn and Tweedie (2009,chap. 17), we have:

$$\overline{f}_n - \mathrm{E}_x(f(x)) \xrightarrow{d} N(0, \frac{\sigma_{\overline{f}}^2}{n}). \tag{2.30}$$

Notice that with MCMC dependent samples, $\sigma^2_{\overline{f}} \neq Var_x(f(x))$ in general. The evaluation of $\sigma^2_{\overline{f}}$ is however non-trivial and is hence usually approximated by $\widehat{\sigma}^2_{\overline{f};n}$ under certain conditions (Geyer, 2011; Flegal and Jones, 2011; Flegal et al., 2012).

The quantity $\widehat{\sigma}_{\overline{f};n}/\sqrt{n}$ is called the *Monte Carlo standard error (MCSE)* and the $(1-\alpha)100\%$ interval of the estimation $\mathrm{E}_x(f(x))$ is $[\,\overline{f}_n - \varsigma_{\alpha/2}\widehat{\sigma}_{\overline{f};n}/\sqrt{n}, \overline{f}_n + \varsigma_{\alpha/2}\widehat{\sigma}_{\overline{f};n}/\sqrt{n}\,]$ where $\varsigma_{\alpha/2}$ is an appropriate quantile of the *student-t distribution* or the normal distribution, e.g $(1-\alpha/2)100\%$ quantile of the normal distribution. The term $\varsigma_{\alpha/2}\widehat{\sigma}_{\overline{f};n}/\sqrt{n}$ is the half-width of the interval.

The MCSE is obtained by *batch means methods* and can be extended for quantiles and functions of expectations. If the half-width is small, it can be concluded that the MCMC algorithm has run long enough. In this sense, the MCSE can be used as a stopping criteria of MCMC. A large half-width may imply:

- There are not enough MCMC samples to achieve a reasonable precision.
- The batch means are quite different, caused by a mixing issue or a convergence issue.

This procedure is implemented in the *R package mcmcse* (Flegal, 2012).

The Metropolis-Hastings algorithm

Assuming that the target non-normalised density is $q_x(\cdot)$, the Metropolis-Hastings algorithm is in Algorithm 2.

Algorithm 2 Metropolis-Hastings

```
1. Define a non-normalised proposal density $q_{r;y|x}(\cdot)$. At iteration 1, choose a starting point $x_1 = s$.
2. For iterations $t = 2 : n$, assume $a$ is the value of the previous state ($x_{t-1} = a$) draw a sample $b$ from the density $q_{r;y|x=a}(\cdot)$. Evaluate the acceptance rate:
       $\alpha(a,b) = \min\left(\frac{q_x(b)q_{r;y|x=b}(a)}{q_x(a)q_{r;y|x=a}(b)}, 1\right).$   (2.31)
3. Accept the movement with probability $\alpha(a,b)$: $x_t = b$. Otherwise, $x_t = a$. Repeat step 2, until $t \geq n$
```

It can be proved that the Metropolis-Hastings algorithm satisfies the detailed balance. There are some common ways to choose the proposal density $q_{r;y|x}(\cdot)$:

- *Random walk proposal*: $q_{r;y|x=a}(b) = h(b-a)$. For example, $q_{r;y|x=a}(b) = N(b|\mu = a, \sigma^2)$
- *Symmetric proposal*: $q_{r;y|x=a}(b) = q_{r;y|x=b}(a)$
- *Independent proposal*: $q_{r;y|x=a}(b) = h(b)$

Gibbs sampling

Assume that the random vector x_t (at time t) can be divided into multiple blocks $x_t = (x_{1,t}, ..., x_{d,t})$. The Gibbs sampling schemes update each random block $x_{i,\cdot}$ one by one, conditioning on the other blocks $x_{-i,\cdot} = (x_{1,\cdot}, ..., x_{i-1,\cdot}, x_{i+1,\cdot}, ..., x_{d,\cdot})$. There are two popular schemes: the systematic scan and the random scan.

A *systematic-scan* Gibbs sampling in Algorithm 3 updates each block in a predefined sequence.

Algorithm 3 Gibbs sampling: systematic scan

```
1. Choose a starting point $x_t = (x_{1,t}, ..., x_{d,t}) = s$.
2. For iterations $t = 2 : n$, iteratively update the random blocks
    2-1. Sample $x_{1,t}$ from $q_{x_1|x_2=x_{2,t-1},x_3=x_{3,t-1},...,x_d=x_{d,t-1}}(\cdot)$
    2-2. Sample $x_{2,t}$ from $q_{x_2|x_1=x_{2,t},x_3=x_{3,t-1},...,x_d=x_{d,t-1}}(\cdot)$
    2-3. ...
    2-d. Sample $x_{d,t}$ from $q_{x_2|x_1=x_{2,t},x_2=x_{2,t},...,x_{d-1}=x_{d-1,t}}(\cdot)$
3. Repeat step 2, until $t \geq n$
```

A *random-scan* in Algorithm 4 updates a random block in each iteration.

Algorithm 4 Gibbs sampling: random scan

```
1. Choose a starting point $x_t = (x_{1,t}, ..., x_{d,t}) = s$.
2. For iterations $t = 2 : n$, choose a block $i \in (1 : d)$ with probability $w_i$.
   Then, update the corresponding $x_{i,t}$ from $q_{x_i|x_{-i}=x_{-i,t-1}}(\cdot)$. The rest is unchanged:
   $x_{-i,t} = x_{-i,t-1}$
3. Repeat step 2, until $t \geq n$
```

In the above two algorithms, it is assumed that $x_{i,t}$ can be sampled directly by a standard form of the conditional density $q_{x_i|x_{-i}}(\cdot)$. Otherwise, sampling methods in Section 2.2.1, IS or MCMC can be used for a single block update x_i.

Combining MCMC chains

Aside from using the Gibbs sampling on the conditional density, combining different MCMC chains is another strategy. For example, iteratively chaining two MCMC with transition densities $p_{1;y|x}(\cdot)$ and $p_{2;y|x}(\cdot)$ is equivalent to a MCMC with a transition:

$$p_{y|x=a}(c) = \int p_{1;y|x=a}(b) p_{2;y|x=b}(c) \; db. \tag{2.32}$$

And a random MCMC with two sub-MCMC transitions:

$$p_{y|x=a}(b) = w_1 p_{1;y|x=a}(b) + (1 - w_1) p_{2;y|x=a}(b). \tag{2.33}$$

It is clear that if two MCMC with transitions $p_{1;y|x}(\cdot)$ and $p_{2;y|x}(\cdot)$ admit the same stationary density $\pi_x(\cdot)$, then $\pi_x(\cdot)$ is also the stationary density of $p_{y|x}(\cdot)$.

Variable correlation and MCMC

One of the most serious issues in implementing a good MCMC is the correlation between random variables. Let x_1 and x_2 be two random scalars which are highly correlated, i.e. $x = (x_1, x_2)$ follows normal distribution with correlation coefficient of 0.95. Doing Gibbs sampling results in a very small movement in each iteration as the conditional mean of variable x_1 is strongly coupled with the other variable $x_2 = x_{2,t}$ and is very close to its current position $x_{1,t}$.

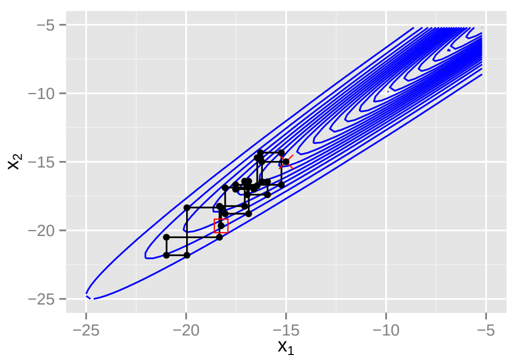
In this situation, updating both variables jointly, i.e. using a MCMC with random walk proposal of variance matrix Σ, is problematic too. A focused proposal (small variance) allows a tiny movement from the current point, causing inefficiency. On the other hand, a flat proposal with large variance risks a low acceptance rate and rejects the movement. For the following target density of random variable $x = (x_1, x_2)$:

$$q_x(x) = N(x|\mu = 0, \Sigma), \tag{2.34}$$

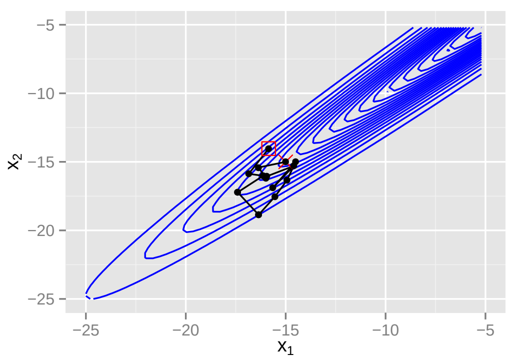
with:

$$\Sigma = \begin{pmatrix} 100 & 99 \\ 99 & 100 \end{pmatrix}. \tag{2.35}$$

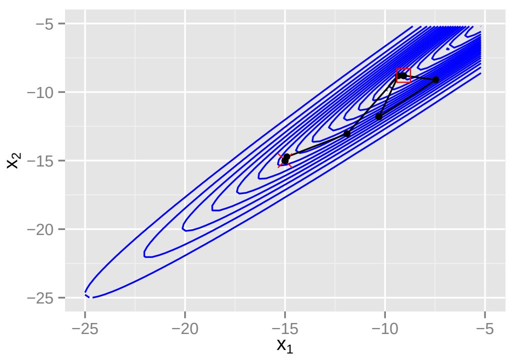
Figure 2.2 shows the MCMC paths of Gibbs sampling and two Metropolis-Hastings for 15 iterations, starting at a same location $s = (-20, -20)$. The normal random walk proposals (from x to y) in Figures 2.2(b) and 2.2(c) have the corresponding covariance



(a) Gibbs sampling



(b) Random walk proposal $N(y|x, \Sigma_1)$



(c) Random walk proposal $N(y|x, \Sigma_2)$

Fig. 2.2: Traced paths of different MCMC algorithms for a highly correlated target density in 15 iterations: The target density is drawn by blue contours and the traced paths are in black. The red crosses and squares denote starting and ending points correspondingly. The MCMC implementation with Σ_1 has made 13 accepted small movements while the MCMC implementation with Σ_2 has 6 accepted big movements.

matrices:

$$\Sigma_1 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \tag{2.36}$$

$$\Sigma_2 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}. \tag{2.37}$$

All three algorithms can only move locally after 15 iterations, illustrating the correlation problem in MCMC, which becomes worse with a high dimensional density.

The neatest way to deal with the correlation issue is to reparameterise the density. By transforming x to $y = f(x)$, the correlation may be completely removed. However, in practise, such a parameterisation may not exist or be unknown (especially in high dimensional space) and hence, general numerical methods are preferred. This includes using the geometrical information in the local derivatives of the target density or complementing the MCMC algorithm with functional approximations, as described in the next section.

2.3 Functional approximation methods

This section summarises functional approximation methods which are usually computationally fast and cheap. Compared to Monte Carlo methods, functional approximation does not have the asymptotic convergence property, e.g. even when a method is applied repeatedly, the approximation may not converge to the target density at all. Still, these methods are extremely useful in a practical application where time is the most invaluable resource. Furthermore, functional approximations can be used to complement Monte Carlo methods (used as IS or MCMC proposal functions). For an IS algorithm, this may produce samples appropriately in the target support and result in evenly weighted samples. A MCMC trajectory may move adaptively to the local correlation or escape a modal trap by using these approximations.

2.3.1 Standard approximations

The Laplace approximation
Among all methods, the *Laplace approximation* (Laplace, 1774) is the simplest but still very useful. In general, this method approximates a target non-normalised density $q_x(\cdot)$ by a normal distribution. The procedure is in Algorithm 5.

The Laplace approximation has all the useful properties of a normal distribution, e.g. the normal form of the conditional and marginal distributions. Also, it is used to evaluate the integral of $q_x(\cdot)$, which is not necessarily a probability density (one of the first use of Laplace approximation).

As distributions tend to converge to normal form asymptotically, the Laplace approximation is very efficient in these cases. However, for non-normal distribution, this approximation suffers from two shortcomings. Firstly, it only works well with a

Algorithm 5 The Laplace approximation

1. Denote $g(\cdot) = -\log(q_x(\cdot))$. Starting at an initial point s, minimise $g(\cdot)$ (usually by numerical optimisation method like gradient-based method, Newton's method):

$$\widehat{x} = \arg\min_x \{g(x)\}. \tag{2.38}$$

2. Evaluate the Hessian matrix of $g(\cdot)$ at the mode $\widehat{x}$ (by analytical formula, finite difference or automatic difference methods):

$$\widehat{Q} = \left.\frac{\partial^2 g(x)}{\partial x^2}\right|_{x=\widehat{x}}. \tag{2.39}$$

3. Use the Taylor expansion on $g(x)$ at $\widehat{x}$:

$$\begin{aligned} g(x) &\approx g(\widehat{x}) + \left.\frac{\partial g(x)}{\partial x}\right|_{x=\widehat{x}} (x-\widehat{x}) + \frac{1}{2}(x-\widehat{x})^T \left.\frac{\partial^2 g(x)}{\partial x}\right|_{x=\widehat{x}} (x-\widehat{x}) \\ &\approx g(\widehat{x}) + \frac{1}{2}(x-\widehat{x})^T \widehat{Q}(x-\widehat{x}), \end{aligned} \tag{2.40}$$

and obtain the approximation:

$$q_x(x) \propto N(x|\mu = \widehat{x}, Q = \widehat{Q}). \tag{2.41}$$

---

uni-modal function and ignores the other modes if the target density is multi-modal. Secondly, the normal distribution implies a linear correlation between random variables and hence cannot accommodate non-linear dependency.

Variational Bayes method

In the *variational Bayes method (VB)* (Jordan et al., 1999), the approximation density $\widetilde{p}_x(\cdot)$ of the target $p_x(\cdot)$ is found by minimising the *Kullback-Leibler divergence (KL)*:

$$\text{KL}(\widetilde{p}|p) = \int \log\left(\frac{\widetilde{p}_x(x)}{p_x(x)}\right) \widetilde{p}_x(x)dx. \tag{2.42}$$

By *Jensen's inequality*, the KL term, $\text{KL}(\widetilde{p}|p)$, is always positive and $\text{KL}(\widetilde{p}|p) = 0$ if $\widetilde{p}_x(\cdot) = p_x(\cdot)$. So, it is a reasonable criteria to measure the "distance" between two densities.

In order to evaluate and minimise the KL in Equation (2.42), the approximation $\widetilde{p}_x(\cdot)$ must be assumed to follow a form. Any parameterised density $\widetilde{p}_{x|\theta}(\cdot)$ can be used;

but the VB method is only computationally efficient if the KL term can be obtained directly and analytically, according to $\widetilde{p}_{x|\theta}(\cdot)$.

One of the most popular forms for the multivariate density $\widetilde{p}_x(\cdot)$ (with $x = x_{1:d}$) is the factorisation:

$$\widetilde{p}_x(x) = \prod_{i=1}^{d} \widetilde{p}_{x_i}(x_i). \tag{2.43}$$

Notice that there is no need to parameterise this approximation form. The VB procedure for this factorisation iteratively minimises $\mathrm{KL}(\widetilde{p}|p)$ with respect to the component $\widetilde{p}_{x_i}(\cdot)$ when fixing the other components $\widetilde{p}_{x_j}(\cdot)$ $(j \neq i)$:

$$\begin{aligned}\mathrm{KL}(\widetilde{p}|p) &= \int \log\left(\frac{\widetilde{p}_{x_i}(x_i)}{p_x(x)}\right) \widetilde{p}_x(x)dx + const \qquad (2.44)\\ &= \int \left[\log(\widetilde{p}_{x_i}(x_i)) - \int \log(p_x(x)) \prod_{j\neq i}^{d} \widetilde{p}_{x_j}(x_j)dx_{j\neq i}\right] \widetilde{p}_{x_i}(x_i)dx_i + const\\ &= \int \left[\log(\widetilde{p}_{x_i}(x_i)) - \mathrm{E}_{x_{j\neq i}} \log(p_x(x))\right] \widetilde{p}_{x_i}(x_i)dx_i + const.\end{aligned}$$

Define a normalised density $p_{a;x_i}(x_i) \propto \exp[\mathrm{E}_{x_{j\neq i}} \log(p_x(x))]$ (a function with respect to x_i only), then:

$$\begin{aligned}\mathrm{KL}(\widetilde{p}|p) &= \int \log\left(\frac{\widetilde{p}_{x_i}(x_i)}{p_{a;x_i}(x_i)}\right) \widetilde{p}_{x_i}(x_i)dx_i + const \qquad (2.45)\\ &= \mathrm{KL}(\widetilde{p}_{x_i}|p_{a;x_i}) + const.\end{aligned}$$

Hence, conditioning on $\widetilde{p}_{x_j}(\cdot)$ with $j \neq i$, the KL term is minimised when $\widetilde{p}_{x_i}(x_i) = p_{a;x_i}(x_i) \propto \exp[\mathrm{E}_{x_{j\neq i}} \log(p_x(x))] \propto \exp[\mathrm{E}_{x_{j\neq i}} \log(q_x(x))]$.

Again, this procedure is efficient if the closed form of $\mathrm{E}_{x_{j\neq i}} \log(p_x(x))$ is available, which is true for some models (Bishop, 2006,chap. 10). However, one may not have such a closed form for a complex density $p_x(x)$ and have to resort to a sampling method to evaluate the expectation, resulting in big computation cost. Another disadvantage of this procedure is the independence assumption of the factorisation, which could be unsatisfying in terms of accuracy.

One method related to VB is the *expectation propagation method (EP)*, which uses $\mathrm{KL}(p|\widetilde{p})$ instead of $\mathrm{KL}(\widetilde{p}|p)$. Interestingly, it is observed that VB approximation tends to be under-dispersed (the variance of the approximation is underestimated) while EP approximation is over-dispersed (Bishop, 2006,chap. 10).

2.3.2 The iterated Laplace approximation

As an extension of the normal form in Laplace approximation, the mixture of normal distributions is a very flexible family of distributions, capable of accommodating multi-modal and non-linear functions:

$$\widetilde{q}_x(x) = \sum_{i=1}^{n} w_i N(x|\mu_i, Q_i). \tag{2.46}$$

Let $g(\cdot) = -\log(q_x(\cdot))$. The procedure of Gelman et al. (2003,chap. 12) for estimating $w_{1:n}$, $\mu_{1:n}$ and $Q_{1:n}$ is described in Algorithm 6.

Algorithm 6 Approximation by a mixture of normal distributions

---

1. Find n local modes by minimising the target non-normalised density $g(x)$, starting at different initial points. Use these local modes as the component means $\mu_{1:n}$.

2. The component precision matrix can be estimated by:

$$Q_i = \left. \frac{\partial^2 g(x)}{\partial x^2} \right|_{x=\mu_i}. \tag{2.47}$$

3. Estimate the weights by solving:

$$q_x(\mu_i) = \widetilde{q}_x(\mu_i), \quad (i = 1 : n). \tag{2.48}$$

There is an issue with this algorithm. For example, with a skewed uni-modal target density, all the modes and precisions, found by this algorithm, will be almost same (the difference is caused by numerical computation) and the resulting approximation is symmetric (by normal distribution), not able to reflect the skewness of the target. Furthermore, it may be wasteful to do the mode optimisation in step 1 by random starting points. Hence, it is better to do the approximation iteratively, correcting the discrepancy by adding a new component at an appropriate location. Using this approach, Bornkamp (2011*a*) proposed the *iterated Laplace approximation (iterLap)* as in Algorithm 7.

Algorithm 7 iterLap

1. Find n_1 local minima μ_i and their corresponding Hessian matrices Q_i of $g(\cdot) = -\log(q_x(\cdot))$ (like the above algorithm). Let n_c be the current number of components (initially, $n_c = n_1$). The current approximation $\widetilde{q}_{n_c;x}(\cdot)$ with unknown non-normalised weights is:

$$\widetilde{q}_{n_c;x}(x) = \sum_{i=1}^{n_c} w_i N(x|\mu_i, Q_i). \tag{2.49}$$

2. Assume that for each component i, there are n_x location vectors $x_{g;i,j}$ $(j = 1 : n_x)$ generated from the distribution $N(\mu_i, Q_i)$; d_x is the length of vector x. Let X be a $m \times d_x$ matrix comprising all the location vectors $x_{g;i,j}$ and the mean μ_i (each row $X_{k,1:d_x}$ $(k = 1 : m)$ is either $x_{g;i,j}$ or μ_i). Let y be a vector of length m comprising the values of the target density, evaluated for all locations in X: $y_k = q_x(X_{k,1:d_x})$. Let Z be a $m \times n_c$ matrix, satisfying: $Z_{k,i} = N(X_{k,1:d_x}|\mu_i, Q_i)$. The weights $w = w_{1:n_c}$ can be obtained by using *quadratic programming*:

$$w = \arg\min_b [(y - Zb)^T (y - Zb)], \quad b \geq 0. \tag{2.50}$$

3. Define a residual function $g_{n_c}(\cdot)$ with:

$$z = q_x(x) - \widetilde{q}_{n_c;x}(x), \tag{2.51}$$

$$g_{n_c}(x) = \begin{cases} -\log(z) & \text{if } z \geq z_l \\ -\log(\exp(z - z_l)z_l) & \text{if } z < z_l \end{cases}. \tag{2.52}$$

A lower bound $z_l = 1e^{-4}$ is used in the *R package iterLap* (Bornkamp, 2011*b*). Find a new component by minimising the residual function:

$$\mu_{n_c+1} = \arg\min_x \{g_{n_c}(x)\}, \tag{2.53}$$

$$Q_{n_c+1} = \left. \frac{\partial^2 g_{n_c}(x)}{\partial x^2} \right|_{x=\mu_{n_c+1}}. \tag{2.54}$$

The starting points of this optimisation step are chosen by checking the ratio $q_x(x)/\widetilde{q}_{n_c;x}(x)$. Points with large ratios are preferred (the locations where $\widetilde{q}_{n_c;x}(x)$ underestimates $q_x(x)$).

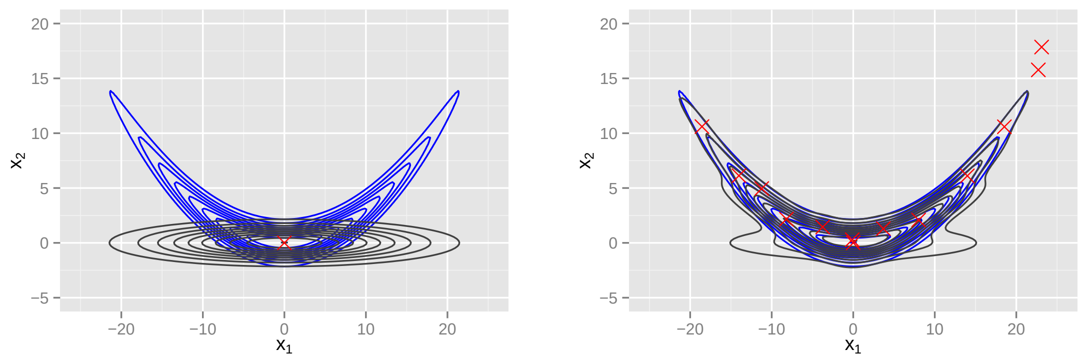
Algorithm 7 iterLap (Cont)

```
4. Update the number of components: $n_c = n_c + 1$. If one of the following criteria
   is satisfied:
     • $n_c \geq n_{c;max}$ where $n_{c;max}$ is a predefined maximum number of components.
     • $|q_x(x) - \widetilde{q}_{n_c;x}(x)| \leq \delta$ with a predefined error bound $\delta$.
     • $\sum_{i=1}^{n_c} w_i$ does not improve (by comparing with previous sums of weights).
     • Cannot find a new component.
   then re-estimate the weights by step 2 and stop the algorithm. Otherwise, repeat
   steps 2 → 4.
```

Figure 2.3 shows both the Laplace and iterLap approximations for the following target density of a random variable $x = (x_1, x_2)$:

$$q_x(x) \propto N(x_1|0, \sigma_1^2 = 10^2)N(x_2|\mu_2 = 0.03x_1^2, \sigma_2^2 = 1^2). \tag{2.55}$$

In Figure 2.3, the iterLap approximation is much better in terms of adapting to the non-linear dependency between two variables. Still, the iterLap method can be further improved by some modifications, which will be discussed later in Section 5.1.



(a) The Laplace approximation (b) The iterLap approximation

Fig. 2.3: The Laplace and iterLap approximations. The target density and approximations are drawn by blue and black contours accordingly. The red crosses are the means of normal components. The iterLap approximation has 15 components.

2.4 Time series modelling

In this section, we briefly discuss *time series modelling*, used in time-dependent data such as stock values, chemical processes or traffic flows, and two general classes: the *classical time series models* and the *dynamic models.* Very detailed references of the former include Brockwell and Davis (2002); Shumway and Stoffer (2006); Box et al. (2008). Brockwell and Davis (2002,chap. 8) and Shumway and Stoffer (2006,chap. 6) are introductory texts for dynamic models. An extensive study of dynamic models can be found in West and Harrison (1997) and Prado and West (2010).

2.4.1 Classical time series models

In classical time series models, the target time-dependent process is usually decomposed into two parts: a mean m_t and an error y_t:

$$z_t = m_t + y_t. \tag{2.56}$$

Eliminating the trend and seasonality

The mean m_t may be a combination of a trend and multiple seasonal effects. So, classical methods usually remove the mean first and then model the remainder as the error. A very common way of extracting the error is using the difference operator $\bigtriangledown$ or the s-lagged difference operator $\bigtriangledown_s$:

$$\bigtriangledown z_t = z_t - \mathrm{B}\, z_t \ = z_t - z_{t-1}, \tag{2.57}$$

$$\bigtriangledown_s z_t = z_t - \mathrm{B}^s\, z_t = z_t - z_{t-s}, \tag{2.58}$$

where B is the backward shift operator: $\mathrm{B}\, z_t = z_{t-1}$.

Such differencing is applied repeatedly to the observation z_t to eliminate the trend and seasonality. If $m_t = \sum_{i=0}^{m}(a_i t^i)$, then $\bigtriangledown^m z_t = m! a_m + \bigtriangledown^m y_t$ (The common mean $m! a_m$ may be removed by a further step or a non-zero constant mean model should be used for the remainder). Differencing is also used for *non-stationary* models such as random walk $z_t = z_{t-1} + y_t$: $\bigtriangledown z_t = y_t$. A limitation of differencing lies in the fact that z_t is only allowed to be non-stationary in a very special way, i.e. $\bigtriangledown^m z_t$ must be stationary. Such a method will not work for the model $z_t = -z_{t-2} + y_t$ (cannot be used to extract the stationary error y_t), for example, see Brockwell and Davis (2002,chap. 6) for more details.

Other methods of cancelling the trend and the seasonality include moving average smoother ($m_t = \sum_{i=-k}^{k} a_{t,i} z_{t+i}$ with $a_{t,i} \geq 0$ and $\sum_{i=-k}^{k} a_{t,i} = 1$) and smoothing splines (modelling m_t as a piecewise polynomial of time, with or without a linear combination of other covariates); see Brockwell and Davis (2002,chap. 1) and Shumway and Stoffer (2006,chap. 2).

The autoregressive (AR) model

In the classical approach, a suitable model for the error y_t is found by inspecting and discovering a pattern of some statistics, derived from the error y_t. Such diagnostics often rely on the *autocorrelation function (ACF)* and *partial autocorrelation function (PACF)* (The derivation and interpretation of PACF is in Brockwell and Davis (2002,chap. 2-3)) and Box et al. (2008,chap. 3).

One of the most accepted models for the time-dependent error y_t is the *autoregressive model, AR(p)*, with a common mean μ:

$$\begin{aligned} & \phi(\mathrm{B})(y_t - \mu) = e_t \qquad (2.59) \\ \Leftrightarrow \quad & (y_t - \mu) - \sum_{i=1}^{p} \phi_i (y_{t-i} - \mu) = e_t, \end{aligned}$$

where $e_t \overset{iid}{\sim} N(0, \sigma^2)$; $\phi(\mathrm{B}) = 1 - \sum_{i=1}^{p} \phi_i \mathrm{B}^i$ is the p-order polynomial of the backward shift operator B.

There are some properties about the roots z of the equality $\phi(z) = 0$:

- If all the roots z lie outside of the unit circle, $|z| > 1$, then there exists a stationary solution for Equation (2.59) and $(y_t - \mu)$ can be written as a linear combination of future-independent e_t: $y_t - \mu = \sum_{i=0}^{\infty} \psi_i e_{t-i}$. In that case, the AR model is *causal.*

- If there is a root z on the unit circle, then there is no stationary solution.

- If all the roots z lie inside of the unit circle, $|z| < 1$, then there exists a stationary solution and $(y_t - \mu)$ can be written as a linear combination of future-dependent e_t: $y_t - \mu = \sum_{i=0}^{-\infty} e_{t-i}$. The realisation of such an AR model usually explodes over time.

Notice that if the distribution of the starting random variable, $p_{y_{0:(-p+1)}}(\cdot)$ is predefined, then the resulting random process is not a stationary solution of an AR model.

The autocorrelation function of the AR model is a mixture of damped exponentials or damped sine waves and the partial autocorrelation is cut off after a certain time point. These two features are used to identify the AR pattern of a particular dataset.

The moving average (MA) model

Another common model is the *moving average model, MA(q)*, with mean μ:

$$\begin{aligned} & y_t - \mu = \theta(\mathrm{B}) e_t \\ \Leftrightarrow \quad & y_t - \mu = e_t + \sum_{j=1}^{q} \theta_i e_{t-i}, \end{aligned} \tag{2.60}$$

where $e_t \overset{iid}{\sim} N(0, \sigma^2)$; $\theta(\mathrm{B}) = 1 + \sum_{i=1}^{q} \theta_i \mathrm{B}^i$ is the q-order polynomial.

A MA model is *invertible* if and only if all the roots z of the equality $\theta(z) = 0$ lie outside of the unit circle, $|z| > 1$. In that case, e_t can be written as a linear combination of future-independent y_t: $e_t = \sum_{i=0}^{\infty} \pi_i (y_{t-i} - \mu)$. This invertibility constraint is imposed to avoid the *identifiability problem* of the MA model, e.g. the model M_a: $y_t = e_t + \theta_{a,1} e_{t-1}$ with noise variance σ_a^2 and the model M_b: $y_t = e_t + \theta_{b,1} e_{t-1}$ with noise variance σ_b^2 will have the same autocovariance function if $\theta_{b,1} = 1/\theta_{a,1}$ and $\sigma_b^2 = \theta_{a,1}^2 \sigma_a^2$ (Shumway and Stoffer, 2006,chap. 3).

In contrast to the AR model, the autocorrelation of the MA model is zero after a certain time point and its partial autocorrelation is a mixture of damped exponentials or damped sine waves.

The autoregressive moving average (ARMA) model

An *ARMA(p,q)* model, with a mean μ is a combination of AR and MA models:

$$\begin{aligned} & \phi(\mathrm{B})(y_t - \mu) = \theta(\mathrm{B}) e_t \\ \Leftrightarrow \quad & (y_t - \mu) - \sum_{i=1}^{p} \phi_i (y_{t-i} - \mu) = e_t + \sum_{j=1}^{q} \theta_j e_{t-j}, \end{aligned} \tag{2.61}$$

There are also properties of causality and invertibility for the ARMA model (with the same condition on the equalities $\phi(z) = 0$ and $\theta(z) = 0$). The autocorrelation function and the partial autocorrelation function of the ARMA model are mixtures of damped exponentials or damped sine waves. Formulae for the ARMA autocorrelation can be found in Box et al. (2008). It is also assumed that the two polynomials $\phi(B)$ and $\theta(B)$ have no common roots for the sake of identifiability.

Solving the equality $\phi(z) = 0$ to impose the causality constraint is tedious and non-trivial. Fortunately, there is a simpler test through the one-to-one transformation of the vector $\phi_{1:p}$ (Monahan, 1984). The test can be applied to both causality and invertibility constraints.

Parameter estimation for the ARMA model

There are quite a few methods to estimate the parameters of the ARMA model (Brockwell and Davis, 2002,chap. 5). The *Yule-Walker method*, its recursive version (the *Durbin-Levinson* algorithm), and *Burg's algorithm* only work with AR models (an ARMA model needs to be converted to an AR model with an infinite number of parameters before applying these methods). The *innovation algorithm* is for the MA model and the *Hannan-Rissanen algorithm* is applicable to the ARMA model.

Various schemes of MLE, based on different transformations, include Ansley (1979), Brockwell and Davis (2002,chap. 5) and Box et al. (2008,chap. 7). All these methods use an unconditional likelihood, i.e. without conditioning on the starting latent variables $y_{0:(-p+1)}$ and $e_{0:(-q+1)}$ and are quite computationally costly.

For the Bayesian approach, Monahan (1983) focuses on the ARMA order selection problem. The parameter space of $\phi_{1:p}$ and $\theta_{1:q}$, constrained by the stationarity and invertibility requirements, is explored by the numerical quadrature method. Chib and Greenberg (1994) adopt a sampling-based approach with MCMC and suggests a transformation to facilitate the conjugacy form of $\phi_{1:p}$. For $\theta_{1:q}$, the Metropolis-Hastings algorithm is applied with a proposal function based on a quadratic approximation of the log density. In the work of Marriott et al. (1996), the parameters $\phi_{1:p}$ and $\theta_{1:q}$ are transformed to an unconstrained version, $r_{1:p}$ and $s_{1:q}$, of the partial autocorrelation coefficients, then jointly updated with a Metropolis-Hastings proposal, centred at the MLE estimates of $(\widehat{r}_{1:p},\widehat{s}_{1:q})$ and its asymptotic covariance matrix. Barnett et al. (1997) instead put a prior directly on the partial autocorrelation coefficients, which are updated one by one in the Gibbs sampling. This work extends the flexibility of the ARMA model by the use of indicator variables on the partial autocorrelation, additive outliers (the observation is the sum of an ARMA variable and an additive outlier) and innovation outliers (the ARMA error e_t follows a mixture of normal distributions). Another way to parameterise the ARMA model is discussed in Prado and West (2010,chap. 2), where the real and complex roots of the polynomial are considered. All the above works, except Monahan (1983), use conditional likelihood (conditioning on the starting latent variables).

Variants of the ARMA model

There are many extensions of the ARMA model. The *periodic autoregressive moving average model (PARMA)* uses multiple AR, MA parameter sets corresponding to each periodic phase:

$$\phi(\mathrm{B}, t)(y_t - \mu) = \theta(\mathrm{B}, t)e_t, \tag{2.62}$$

where $\phi(\mathrm{B}, d + \nu) = \phi(\mathrm{B}, \nu)$ and $\theta(\mathrm{B}, d + \nu) = \theta(\mathrm{B}, \nu)$ for $\nu = 0 : (d - 1)$. The resulting autocovariance is hence periodic (or cyclic time-independent): $\mathrm{Cov}_(\, y_{d+t}, y_{d+s}) = \mathrm{Cov}_(\, y_t, y_s)$.

The *generalised autoregressive conditional heteroskedasticity model, GARCH(p,q)*, permits a time-dependent conditional variance but leaves the unconditional variance unchanged:

$$\begin{aligned} (y_t - \mu) &= \sigma_t e_t, \\ \sigma_t^2 &= \phi_0 + \sum_{i=1}^{p} \phi_i y_{t-i}^2 + \sum_{j=1}^{q} \theta_j \sigma_{t-j}^2. \end{aligned} \tag{2.63}$$

This model is useful when the variation of the observed time series fluctuates accordingly at a specific time.

It has been shown that the autocorrelation $\rho(h)$ of the ARMA model decays quickly due to the damped exponential (or sine wave) form: there exists $r > 1$ so that $r^h \rho(h) \to 0$ when $h \to \infty$. To handle "long memory" time series, the *autoregressive fractionally integrated moving average model, ARFIMA(p,d,q)*, where the autocorrelation has a very long tail, can be used:

$$\phi(\mathrm{B})(y_t - \mu) = \theta(\mathrm{B})(1 - \mathrm{B})^{-d} e_t, \quad -0.5 < d < 0.5, \tag{2.64}$$

with

$$(1-\mathrm{B})^{-d} = \sum_{i=0}^{\infty} \frac{\Gamma(i+d)}{\Gamma(i+1)\Gamma(d)} \mathrm{B}^i, \tag{2.65}$$

where $\Gamma(\cdot)$ is the gamma function.

There are also the *bilinear extension* (allow the interactive effects of y_t and e_t), the *general threshold autoregressive model* (this model has multiple AR, MA parameter sets like the PARMA model but there is also a threshold variable specifying the active parameter set at each time point) and the *smooth transition autoregressive model (STAR)* (a mixture of ARMA models in which the weight series are either deterministic or stochastic). Further discussion about these ARMA variants can be found in Brockwell and Davis (2002,chap. 10), Shumway and Stoffer (2006,chap. 5) and Holan et al. (2010).

The vector autoregressive moving average (VARMA) model

One useful extension of the ARMA model is its multivariate version, the *vector autoregressive moving average model (VARMA).* The d_y-variate VARMA(p,q) model has the following representation:

$$\begin{aligned} \mathbb{\Phi}(\mathrm{B})(y_t-\mu) &= \mathbb{\Theta}(\mathrm{B})e_t \qquad (2.66)\\ \Leftrightarrow \quad (y_t-\mu) - \sum_{i=1}^{p} \mathit{\Phi}_i(y_{t-i}-\mu) &= e_t + \sum_{j=1}^{q} \mathit{\Theta}_j e_{t-i} \end{aligned}$$

where y_t and μ are vectors of length d_y; $e_t \overset{iid}{\sim} N(0, \mathit{\Sigma}_e = Q_e^{-1})$; each AR (or MA) parameter, $\mathit{\Phi}_i$ (or $\mathit{\Theta}_j$) is a $d_y \times d_y$ matrix; $\mathbb{\Phi}(\mathrm{B})$ (or $\mathbb{\Theta}(\mathrm{B})$) is a $d_y \times d_y$ matrix of polynomials of the backward shift operator B: $\mathbb{\Phi}(\mathrm{B}) = I_k - \sum_{i=1}^{p} \mathit{\Phi}_i \mathrm{B}^i$ ($\mathbb{\Theta}(\mathrm{B}) = I_k - \sum_{j=1}^{q} \mathit{\Theta}_j \mathrm{B}^j$).

All properties of the causality, stationarity and invertibility are still valid for the VARMA but the condition is more complicated. A VARMA model is causal (or invertible) if and only if all the roots of the equality $|\mathbb{\Phi}(\mathrm{B})| = 0$ (or $|\mathbb{\Theta}(\mathrm{B})| = 0$) lie outside the unit circle. We do not know any convenient test for this requirement. Furthermore, the causality and invertibility conditions are not enough to ensure the uniqueness of the VARMA model, i.e. two causal and invertible VARMA models may yield the same covariance structure. Holan et al. (2010) discuss this issue and a sufficient condition for the identifiability (echelon form).

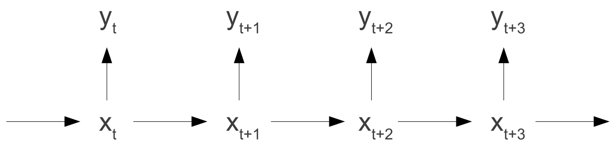


Fig. 2.4: A DAG of a DLM

Ravishanker and Ray (1997) presented a Bayesian approach for parameter estimation of the VARMA model. A Bayesian variable selection is employed with indicator variables for all elements of matrices Φ_i and Θ_j. All parameters, except the covariance matrix Σ_e, are sampled with the Metropolis-Hastings of a normal random walk proposal (the covariance of that proposal is computed by the observed information at the MLE estimator). All matrices $(\Phi, \Theta) = (\Phi_{1:p}, \Theta_{1:q})$ are updated jointly.

2.4.2 The dynamic model

The *dynamic linear model (DLM)* is another popular model for time series. In a DLM, there are a *state equation* describing the evolution of an unobserved *state vector (state variable)* x_t of length d_x and an *observation equation* modelling an observation vector y_t of length d_y, conditioning on the state vector x_t at a specific time:

$$y_t = F_t x_t + v_t, \tag{2.67}$$

$$x_t = G_t x_{t-1} + u_t, \tag{2.68}$$

where $v_t \overset{iid}{\sim} N(0, \Sigma_v)$; $u_t \overset{iid}{\sim} N(0, \Sigma_u)$; F_t are $d_y \times d_x$ matrices; G_t are $d_x \times d_x$ matrices; there may be also an unknown parameter vector φ, e.g. $\varphi = (\Sigma_v, \Sigma_u)$. The separation of the state and observation equations provides a natural way to model dynamics in practise. To include other available covariates H_t with coefficient α, the observation equation can be changed to:

$$y_t = H_t\alpha + F_t x_t + v_t. \tag{2.69}$$

A DLM with fixed parameters in Equation (2.67) can be represented by a *directed acyclic graph (DAG)* in Figure 2.4, showing the sequential generation of the random variables. Let a DAG represent a collection of variables $y_{1:n} = (y_1, ..., y_n)$, then the joint density $p_{y_{1:n}}(y_{1:n})$ is $\prod_i p_{y_i|parents(y_i)}(y_i)$.

Another useful graphical representation of the DLM is the *undirected graph*, showing the conditional independence of random variables. In a undirected graph, if variables

y_i and y_j are separated by a vector variable y_k, i.e. there is no path connecting y_i and y_j without crossing variable y_k, then y_i is conditional independent of y_j, given y_k. The undirected graph of the DLM with fixed parameters is the same as its DAG (replace directed links by undirected ones).

Extensions of the DLM

The *dynamic model (DM)* is one significant DLM extension where either one or both of the state equation and the observation equation are non-linear (both equations may be non-Gaussian too), and are represented by:

$$y_t \sim p_{y_t|x_t,\varphi}(\cdot), \tag{2.70}$$

$$x_t \sim p_{x_t|x_{t-1},\varphi}(\cdot), \tag{2.71}$$

accordingly (φ is an unknown parameter vector).

Another variant is the *conditional dynamic linear model (CDLM)*. In this model, there are multiple DLMs and the active DLM at time t is decided by another auxiliary state variable λ_t.

Filtering, prediction, smoothing with known parameters

There are some common problems in the DLM and the DM. Assuming that all the parameters are known, the first problem is the *filtering* where we are interested in the density of state vector x_t, given all the data until now $y_{1:t}$: $p_{x_t|y_{1:t}}(x_t)$.

The next problem is the h-step *prediction*, formulated by the density: $p_{y_{t+h}|y_{1:t}}(y_{t+h})$ ($h \geq 0$). In the prediction problem, the h-step forecasting function is defined by:

$$\mathrm{E}_{y_{t+h}|y_{1:t}}(y_{t+h}) = \int y_{t+h}\ p_{y_{t+h}|y_{1:t}}(y_{t+h})dy_{t+h}. \tag{2.72}$$

For the DLM with time-invariant matrices $F_t = F$ and $G_t = G$, the h-step forecasting function is given by:

$$\mathrm{E}_{y_{t+h}|y_{1:t}}(y_{t+h}) = FG^h\,\mathrm{E}_{x_t|y_{1:t}}(x_t) \tag{2.73}$$

$$= \sum_i a_{i,h}\lambda_i^h. \tag{2.74}$$

where $\lambda_{1:d_x}$ are the eigenvalues of matrix G. These eigenvalues will decide whether the forecasting function explodes, decays or follows a random walk.

Finally, in the *smoothing* problem, one wants to trace back the past with the all data until now. Hence, the smoothing density $p_{x_{t-h}|y_{1:t}}(x_{t-h})$ $(h \geq 0)$ is needed. This thesis does not discuss the smoothing problem. More details of the problem can be found in West and Harrison (1997,chap. 4) or Prado and West (2010,chap. 4). The *sequential (online)* solutions of these three problems are usually more interesting to practitioners than the *offline* solutions. Sequential inference for the filtering problem is discussed in details in Section 2.5.

Parameter estimation

With the existence of unknown parameters, these parameters must be estimated to complete the model. The following are some offline strategies for parameter estimation of the DM. In some cases, the unconditional likelihood $p_{y_{1:n}|\varphi}(y_{1:n})$ can be obtained directly by marginalising out the state vector $x_{1:n}$:

$$p_{y_{1:n}|\varphi}(y_{1:n}) = \int p_{x_{1:n},y_{1:n}|\varphi}(x_{1:n}, y_{1:n})dx_{1:n} \tag{2.75}$$

This can be done in the DLM thanks to the linear and Gaussian form (Brockwell and Davis, 2002,chap. 8). Then, MCMC or MLE can be applied to estimate φ.

Another solution is using MCMC iteratively on $p_{\varphi|x_{1:n},y_{1:n}}(\cdot)$ and $p_{x_{1:n}|\varphi,y_{1:n}}(\cdot)$ (the MLE can also obtained in this manner by an iterative optimisation). The *forward filtering backward sampling (FFBS)* scheme for $p_{x_{1:n}|\varphi,y_{1:n}}(\cdot)$ can be done exactly in the DLM (Carter and Kohn, 1994), or approximately in the (non-linear) DM (Niemi, 2010). Shephard and Pitt (1997) divide the vector $x_{1:n}$ into small blocks and updates each block one by one by MCMC.

In a different light, Andrieu et al. (2010) employ the particle filter as a natural proposal to complement the MCMC method (*particle Markov chain Monte Carlo - PMCMC*); there are some strategies to deal with the strong correlation of $x_{1:n}$ and φ, e.g. update both variables jointly. The *iterated filtering* method, proposed by Ionides et al. (2011), resorts to the gradient-based optimisation for the MLE where the gradient is approximated by the use of the particle filter in each iteration.

There is also the non-sampling method, *integrated nested Laplace approximations (INLA)* (Rue et al., 2009), in which a grid of the approximate density $\widetilde{p}_{\varphi|y_{1:n}}(\cdot)$ is given by:

$$\widetilde{p}_{\varphi|y_{1:n}}(\varphi) \propto \frac{p_{x_{1:n},y_{1:n},\varphi}(x_{1:n}, y_{1:n}, \varphi)}{\widetilde{p}_{x_{1:n}|y_{1:n},\varphi}(x_{1:n})} \tag{2.76}$$

where $\widetilde{p}_{x_{1:n}|y_{1:n},\varphi}(\cdot)$ is an approximation of $p_{x_{1:n}|y_{1:n},\varphi}(\cdot)$, e.g. by normal approximation. The density of marginal interests $x_{1:n}$, $\widetilde{p}_{x_{1:n}|y_{1:n}}(x_{1:n})$, is the weighted sum with respect to the grid:

$$\widetilde{p}_{x_{1:n}|y_{1:n}}(x_{1:n}) = \sum_i \widetilde{p}_{x_{1:n}|y_{1:n},\varphi_i}(x_{1:n})\widetilde{p}_{\varphi|y_{1:n}}(\varphi_i) \tag{2.77}$$

Sequential inference of the DM is notoriously difficult and is discussed in the next section.

2.5 Sequential inference

This section focuses on sequential inference of the filtering problem (usually, the prediction problem can be solved easily after the filtering is done). In the case of fixed parameters, the classical *Kalman filter* for the DLM and the *particle filter* for the DM are summarised in Section 2.5.1. Several existing methods for the combined sequential inference of the state vector and parameters are discussed in Section 2.5.2.

2.5.1 The Kalman filter and particle filter

The Kalman filter for the DLM

For the DLM defined in Equation (2.67), the Kalman filter is a classic exact method. Assume that at time t, the filtering density $p_{x_t|y_{1:t}}(\cdot)$ is $N(\cdot|\mu = a_t, \Sigma = P_t)$ (for $t = 0$, an initial density $p_{x_0}(\cdot) = N(\cdot|\mu = a_0, \Sigma = P_0)$ can be used). The Kalman filter (Kalman, 1960) updates the filtering density by Algorithm 8.

The formulae of the means and variances, a_t, b_t, c_t, P_t, R_t and S_t, can be derived by using the linearity and normal assumptions (West and Harrison, 1997,chap. 4). The Kalman filter can also be generalised for the DLM with white noises u_t and v_t (uncorrelated random variables with zero-mean and constant variance - which are not necessarily iid or normal). All the formulae of the means and variances remain the same but the densities do not follow the normal distribution any more. The proof

Algorithm 8 The Kalman filter

```
1. Obtain the 1-step state prediction density $p_{x_{t+1}|y_{1:t}}(\cdot)$, which has the form $N(\cdot|\mu = b_t, \Sigma = R_t)$.
2. The 1-step observation prediction density $p_{y_{t+1}|y_{1:t}}(\cdot)$ also follows $N(\cdot|\mu = c_t, \Sigma = S_t)$.
3. Combine the prior $p_{x_{t+1}|y_{1:t}}(\cdot)$ and the likelihood $p_{y_{t+1}|x_{t+1}}(\cdot)$ to update $p_{x_{t+1}|y_{1:(t+1)}}(\cdot) = N(\cdot|\mu = a_{t+1}, \Sigma = P_{t+1})$.
```

for the white noise Kalman filter is based on best linear predictor and is referred to Brockwell and Davis (2002,chap. 8).

The Kalman filter can be used as an approximate solution for the (non-linear) DM too. Assume that the state equation is $x_t = g(x_{t-1}) + u_t$. The non-linear function $g(x)$ can be linearised by a Taylor expansion at s:

$$g(x) \approx g(s) + \left.\frac{\partial g(x)}{\partial x}\right|_{x=s} (x - s), \tag{2.78}$$

and so is the function $f(x)$ of the observation equation: $y_t = f(x_t) + v_t$. Then, the Kalman filter can be applied to the linearised model. This scheme is called the *extended Kalman filter (EKF)* (Arulampalam et al., 2002). Details about EKF and another well-known extension, the *unscented Kalman filter (UKF)*, can be found in Julier and Uhlmann (1997); Wan and Merwe (2000); Haykin (2001).

The particle filter for the DM

A very common method for the general DM, defined in Equation (2.70), is the particle filter method (Doucet, Godsill and Andrieu, 2000; Prado and West, 2010; Doucet and Johansen, 2011). Based on importance sampling, the filtering density of the whole sequence, $p_{x_{1:t}|y_{1:t}}(\cdot)$, can be approximated by n samples $x_{1:t,j}$ with normalised weights $w_{n;t,j}$ from a non-normalised proposal density $q_{r;x_{1:t}|y_{1:t}}(\cdot)$:

$$p_{x_{1:t}|y_{1:t}}(x_{1:t}) \approx \widetilde{p}_{x_{1:t}|y_{1:t}}(x_{1:t}) = \sum_{j=1}^{n} w_{n;t,j} \mathbb{1}(x_{1:t} = x_{1:t,j}), \tag{2.79}$$

where the proposal density has a special form:

$$q_{r;x_{1:t}|y_{1:t}}(x_{1:t}) = q_{r;x_1|y_1}(x_1)\prod_{i=2}^{t} q_{r;x_i|x_{i-1},y_i}(x_i). \tag{2.80}$$

Notice that $p_{x_{1:t}|y_{1:t}}(x_{1:t}) \propto q_{x_{1:t}}(x_{1:t})q_{y_{1:t}|x_{1:t}}(y_{1:t})$. So, the weight function $w_t(x_{1:t})$ can be defined as:

$$\begin{aligned} w_t(x_{1:t}) &= \frac{q_{x_{1:t}}(x_{1:t})q_{y_{1:t}|x_{1:t}}(y_{1:t})}{q_{r;x_{1:t}|y_{1:t}}(x_{1:t})} \qquad (2.81)\\ &= \frac{q_{x_1}(x_1)q_{y_1|x_1}(y_1)}{q_{r;x_1|y_1}(x_1)}\prod_{i=2}^{t}\frac{q_{x_i|x_{i-1}}(x_i)q_{y_i|x_i}(y_i)}{q_{r;x_i|x_{i-1};y_i}(x_1)}\\ &= w_1(x_1)\prod_{i=2}^{t}\alpha_i(x_i)\\ &= w_{t-1}(x_{1:(t-1)})\alpha_t(x_t), \end{aligned}$$

where $\alpha_i(x_i)$ is called the increment weight. The particle weights are then:

$$\begin{aligned} w_{t,j} &= w_t(x_{1:t,j}) \qquad (2.82)\\ &= w_{t-1}(x_{1:(t-1),j})\alpha_t(x_{t,j})\\ &= w_{t-1,j}\alpha_{t,j},\\ w_{n;t,j} &= \frac{w_{t,j}}{\sum_k w_{t,k}}. \qquad (2.83) \end{aligned}$$

Thanks to the recursive form of Equations (2.80) and (2.82), the sequential procedure for the approximation of Equation (2.79) is in Algorithm 9.

Algorithm 9 The particle filter

1. At time $t = 1$, generate n samples $x_{1,j}$ from $q_{r;x_1|y_1}(\cdot)$. Calculate the weights $w_{n;1,j}$ and $w_{1,j}$.

2. At time $t \geq 2$, conditioning on the sample $x_{1:(t-1),j}$, sample $x_{t,j}$ from $q_{r;x_t|x_{t-1}=x_{t-1,j},y_t}(\cdot)$. Calculate the weights $w_{n;t,j}$ and $w_{t,j}$ by the recursive form.

Theoretically, the particle filter is a Monte Carlo method and hence provides a asymptotically converged approximation. Practically, as the space of the sequence $x_{1:t}$ increases with t, it would be impossible to explore such a high dimensional space with a reasonable number of particles. So, the approximation of the whole sequence in Equation (2.79) is never accurate (this point will be revisited in the discussion of degeneracy). Still, the filtering approximation at time t, $\widetilde{p}_{x_t|y_{1:t}}(\cdot)$, may be reasonable.

Degeneracy, optimal importance function and resampling

The degeneracy problem becomes much more severe in the sequential setting and is recognised in two ways. Firstly, like importance sampling, the weights become unbalanced (the variance of $w_t(x_{1:t})$ increases) and eventually only some particles have significant weights. Practically, there is no solution for this issue; still, there is a sub-optimal approach by balancing the increment weight $\alpha_i(\cdot)$. If the proposal function is chosen that $q_{r;x_t|x_{t-1},y_t}(\cdot) = q_{x_t|x_{t-1},y_t}(\cdot)$ then conditioning on x_{t-1}, the variance of the increment weight $\alpha_t(\cdot)$ is zero (again, the variance of $w_t(\cdot)$ is not zero) (Doucet, Godsill and Andrieu, 2000):

$$\mathrm{Var}_{x_t|x_{t-1},y_t}(\alpha_t(x_t)) = 0. \tag{2.84}$$

In that case, the proposal is called the *optimal importance function.*

If direct sampling and evaluation of $p_{x_t|x_{t-1},y_t}(\cdot)$ is available, e.g. in the situation where the observation equation is linear and both two equations follow normal distribution, then the optimal importance sampling is computationally efficient. Otherwise, an extra cost of approximating the optimal importance function, e.g. by the Laplace approximation, must be paid; which is highly priced as it is done conditioning on each previous particle $x_{t-1,j}$.

Secondly, it is observed that for a large enough t, almost all the particles $x_{1:t,j}$ share the same root path $x_{1:k,j}$, i.e. the density $p_{x_{1:k}|y_{1:t}}(\cdot)$ $(k < t)$ is approximated by a few, if not a single, distinct particles $x_{1:k,j}$ (Andrieu et al., 2005). So, the approximate filtering density of $p_{x_k|y_{1:k}}(\cdot)$ almost does not have any effect on sequential inference of $p_{x_t|y_{1:t}}(\cdot)$.

Resampling is another approach to balance the weights. In the particle filter procedure, $x_{1:t,j}$ can be resampled with weights $w_{n;t,j}$ (multinomial sampling) and the weights are reset to $1/n$. This trivially fakes equal weights; but not by any means, does it imply that the particles become "better" (all the particles may admit the same path). Resampling simply removes the particles of small weights and gives preference to the rest. However, it is possible that a particle of small weight at time t can lead to a significant particle at time $t+1$.

The auxiliary particle filter

The key point in the particle filter is to propose some samples x_t "suitable" for the new data y_t at time t. If these samples are far from truth (e.g. they cannot cover the 95% probability interval of the density $p_{x_t|y_{1:t}}(\cdot)$), the particles degenerate.

Intuitively, the optimal importance sampling tries to generate the most suitable samples x_t, conditioning on x_{t-1} and the new observation y_t. On the other hand, the auxiliary particle filter (Pitt and Shephard, 1999) improves the original method in a different direction: selecting the particles $x_{t-1,j}$ most "compatible" to y_t. Assuming the filtering density at time $t-1$ is:

$$p_{x_{t-1}|y_{1:(t-1)}}(x_{t-1}) \approx \widetilde{p}_{x_{t-1}|y_{1:(t-1)}}(x_{t-1}) = \sum_{j=1}^{n} w_{n;t-1,j}\mathbb{1}(x_{t-1} = x_{t-1,j}), \tag{2.85}$$

this method adds a new auxiliary random variable k for selecting a particle $x_{t-1,k}$ among $x_{t-1,j}$ $(j = 1 : n)$ with weights $w_{n;t-1,j}$. The approximate full density for (k, x_t, y_t) is:

$$\widetilde{q}_{k,x_t,y_t|y_{1:(t-1)}}(k, x_t, y_t) = w_{n;t-1,k}q_{x_t|x_{t-1}=x_{t-1,k}}(x_t)q_{y_t|x_t}(y_t), \tag{2.86}$$

and the posterior is $\widetilde{q}_{k,x_t|y_{1:t}}(k, x_t) \propto \widetilde{q}_{k,x_t,y_t|y_{1:(t-1)}}(k, x_t, y_t)$. An auxiliary particle filter uses importance sampling on (k, x_t) with a proposal:

$$\begin{aligned} q_{r;k,x_t|y_{1:t}}(k, x_t) &= q_{r;k|y_{1:t}}(k)q_{r;x_t|x_{t-1}=x_{t-1,k},y_t}(x_t) & (2.87)\\ &= w_{n;t-1,k}f_{r;y_t|x_{t-1}=x_{t-1,k}}(y_t)q_{r;x_t|x_{t-1}=x_{t-1,k},y_t}(x_t), & (2.88)\end{aligned}$$

The weight function is then:

$$w_t(k, x_t) = \frac{q_{x_t|x_{t-1}=x_{t-1,k}}(x_t)q_{y_t|x_t}(y_t)}{f_{r;y_t|x_{t-1}=x_{t-1,k}}(y_t)q_{r;x_t|x_{t-1}=x_{t-1,k};y_t}(x_t)}. \tag{2.89}$$

For this method, the best proposal is $f_{r;y_t|x_{t-1}}(\cdot) = q_{y_t|x_{t-1}}(\cdot)$ and $q_{r;x_t|x_{t-1},y_t}(\cdot) = q_{x_t|x_{t-1},y_t}(\cdot)$ (functional approximation may be used if the closed form formulae are not available), in which the variance of $w_t(k, x_t)$ is zero. Another popular choice is $f_{r;y_t|x_{t-1}}(\cdot) = q_{y_t|x_t=m(x_{t-1})}(\cdot)$ and $q_{r;x_t|x_{t-1},y_t}(\cdot) = q_{x_t|x_{t-1}}(\cdot)$, where $m(x_{t-1})$ is a mean, mode or median of $q_{x_t|x_{t-1}}(\cdot)$ (a function of x_{t-1}).

An improved version of this method, utilising the look-ahead strategy, is discussed in Carpenter et al. (1999) and Doucet and Johansen (2011). In another popular particle filter, the *Rao-Blackwell particle filter* (Doucet, de Freitas, Murphy and Russell, 2000; Murphy and Russell, 2001; Doucet and Johansen, 2011), the state vector x_t is partitioned as (u_t, v_t) where the particle filter is used for $p_{u_{1:t}|y_{1:t}}(\cdot)$ and the Kalman filter is used for the conditional density $p_{v_t|u_{1:t},y_{1:t}}(\cdot)$. Hence, the estimated filtering density is:

$$\widetilde{p}_{u_t,v_t|y_{1:t}}(u_t, v_t) = \sum_{j=1}^{n} w_{n;t,j} \mathbb{1}(u_t = u_{t,j}) p_{v_t|u_t=u_{t,j},y_{1:t}}(v_t). \tag{2.90}$$

This method is feasible under certain conditions, yielding a better estimator (with smaller variance) by the *Rao-Blackwell theorem* (Casella and Berger, 2002,chap. 7).

2.5.2 Sequential parameter estimation

The focus of this section is turned to the DM with unknown parameters, characterised by Equation (2.70). The prior for φ is $p_\varphi(\cdot)$. Ultimately, statisticians are interested in the sequential update of the density $p_{x_t,\varphi|y_{1:t}}(\cdot)$

One might want to apply the particle filter directly to $p_{x_{1:t},\varphi|y_{1:t}}(\cdot)$, i.e. sample $(x_{1,j}, \varphi_j)$ to approximate $p_{x_1,\varphi|y_1}(\cdot)$ and then use importance sampling iteratively for the sequence of densities $p_{x_{1:t}|y_{1:t},\varphi}(\cdot)$. However, by this approach, the samples φ_j are never regenerated at each time point (only $x_{t,j}$ are generated at time t). As the density $p_{\varphi|y_{1:t}}(\cdot)$ is usually much different from the proposal density $p_{\varphi|y_1}(\cdot)$, degeneracy will surely occur to φ, i.e. $p_{\varphi|y_{1:t}}(\cdot)$ may be only approximated by a single particle φ_j.

A simple treatment to this problem is to serialise the parameter vector, i.e. creating an artificial evolution for φ:

$$y_t \sim p_{y_t|x_t,\varphi_t}(\cdot), \tag{2.91}$$

$$x_t \sim p_{x_t|x_{t-1},\varphi_t}(\cdot), \tag{2.92}$$

$$\varphi_t = \varphi_{t-1} + w_t. \tag{2.93}$$

However, there are some issues for this artificial dynamic (Kantas et al., 2009). Firstly, the model is changed, allowing the parameter to adjust at each time. Consequently the density $p_{\varphi_t|y_{1:t}}(\cdot)$ may be completely different from the density $p_{\varphi|y_{1:t}}(\cdot)$ of the original model. Next, even if this modification is accepted, one still have the question of choosing an appropriate random noise w_t for the parameter evolution. Finally, by using a random walk with independent noise w_t, the density $p_{\varphi_t|y_{1:(t-1)}}(\cdot)$ becomes over-dispersed, compared to the density $p_{\varphi_{t-1}|y_{1:(t-1)}}(\cdot)$. Notice that this artificial evolution is equivalent to using a kernel density estimator for $p_{\varphi|y_{1:t}}(\cdot)$ at each time point by some works.

Liu and West's method

Based on the work of West (1993), Liu and West (2001) proposed a filtering method for both the parameters and the state vector, solving the last two issues of the artificial evolution. Let the sample mean and the sample variance-covariance matrix of the density $p_{\varphi_{t-1}|y_{1:(t-1)}}(\cdot)$ be $\overline{\mu}_{\varphi;t-1}$ and $\overline{\Sigma}_{\varphi;t-1}$ (obtained by the particles at time $t-1$). Instead of using the random walk in Equation (2.93), Liu and West shrink the current parameter value towards the mean:

$$\varphi_t = a\varphi_{t-1} + (1-a)\overline{\mu}_{\varphi;t-1} + w_t, \tag{2.94}$$

where a is a predefined value ($0 < a < 1$) and the variance of the noise w_t is $\Sigma_{w;t} = (1-a^2)\overline{\Sigma}_{\varphi;t-1}$. Consequently, it can be proved that the variances of $p_{\varphi_t|y_{1:(t-1)}}(\cdot)$ and $p_{\varphi_{t-1}|y_{1:(t-1)}}(\cdot)$ are equal.

There are some notable points about this method. Shrinking modifies the shape of the filtering density and the particles are still allowed to adjust at each time with a bias towards the sample mean. Critically, when the degeneracy actually occurs (few particles with significant weights at time $t-1$), the sample variance $\overline{\Sigma}_{\varphi;t-1}$ underestimates the true variance and is very small; so is the noise variance of w_t, limiting the original purpose of the artificial evolution: the parameter regeneration. In the worst case where there is only one significant sample $\varphi_{t-1,j}$, the parameter will never ever regenerate.

The particle filter and sufficient statistics

Fearnhead (2002), or similarly Storvik (2002), proposed a sequential inference method for both the state and the parameter vectors, exploiting the sufficiency property. Assume that a sufficient statistic $s_t = s_t(x_{1:t}, y_{1:t})$ is available for the parameter vector φ:

$$p_{\varphi|x_{1:t},y_{1:t}}(\cdot) = p_{\varphi|s_t}(\cdot), \tag{2.95}$$

and for the computational efficiency, there should exist an update function for the sufficient statistic: $s_t = f(x_t, y_t, s_{t-1})$.

At time $t-1$, the density $p_{x_{1:(t-1)},s_{t-1}|y_{1:(t-1)}}(\cdot)$ is approximated by particles $(x_{1:(t-1),j}, s_{t-1,j})$ with weights $w_{n;t-1,j}$:

$$\begin{aligned} p_{x_{1:(t-1)},s_{t-1}|y_{1:(t-1)}}(x_{1:(t-1)}, s_{t-1}) &\approx \widetilde{p}_{x_{1:(t-1)},s_{t-1}|y_{1:(t-1)}}(x_{1:(t-1)}, s_{t-1}) \\ &= \sum_{j=1}^{n} w_{n;t-1,j} \mathbb{1}((x_{1:(t-1)}, s_{t-1}) = (x_{1:(t-1),j}, s_{t-1,j})) \end{aligned} \tag{2.96}$$

Importance sampling is then applied to the density:

$$\widetilde{p}_{x_{1:t},\varphi|y_{1:t}}(x_{1:t}, \varphi) \propto \widetilde{p}_{x_{1:(t-1)}|y_{1:(t-1)}}(x_{1:(t-1)}) p_{\varphi|s_{t-1}}(\varphi) p_{x_t|x_{t-1},\varphi}(x_t) p_{y_t|x_t,\varphi}(y_t) \tag{2.97}$$

to obtain samples $(x_{1:t,j}, \varphi_j)$ with weights $w_{n;t,j}$. Next, the sufficient statistic is updated by $s_{t,j} = f(x_{t,j}, y_t, s_{t-1,j})$ and the new particles are $(x_{1:t,j}, s_{t,j})$. Notice that in this method, φ_j are generated directly and exactly from the density $p_{\varphi|s_{t-1}}(\varphi)$.

Carvalho et al. (2010) combine the sufficient statistic with the auxiliary particle filter . In certain conditions (the observation equation is linear and iid random noises follows the normal distribution), the best proposal for the auxiliary particle filter can be used as mentioned in Section 2.5.1.

However, these sufficient statistics approaches are not robust due to the path degeneracy of the particle filter. Implicitly, the sample sufficient statistics $s_{t,j}$ depend on the whole sample paths $x_{1:t,j}$, many of which share a long but similar sub-path. These paths usually only diverge near the current time point t, contributing to the variation of $s_{t,j}$ (Andrieu et al., 2005; Kantas et al., 2009). This may in turn lead to a poor approximation of $\widetilde{p}_{\varphi|y_{1:t}}(\cdot)$. Another problem is the dimension of the sufficient statistic vector s_t, which may be very large in some cases, e.g. a matrix parameter with a quadratic form. The longer the statistic vector, the more particles are needed, reducing the computational efficiency of the online method.

2.6 Conclusion

In this chapter, Bayesian statistics and some basic concepts have been reviewed. We have gone through two popular classes of statistical methods, Monte Carlo methods and functional approximations, for the evaluation of expectations and integrals, usually representing as mathematical forms of interested problems. For the spatial-temporal modelling in general and the transportation modelling specifically, some well-known time series models have been explored. Also, we have summarised several sequential inference techniques as essential tools of real-time statistical applications. It is noted that the degeneracy is the cause of the instability of the particle filter, especially in the case of unknown parameters. In the next chapter, we will survey the transportation networks and some related issues, including the short term traffic flow forecasting.

Chapter 3

Transportation networks and some related problems

This chapter will introduce *intelligent transportation systems (ITS)* as very powerful tools of transportation management. Then, we will summarise some popular models for the short term traffic flow forecasting problem. Finally, the signal control and the vehicle counting problems are discussed for the sake of subsequent chapters.

3.1 Intelligent transport system

Over the past few decades, the blooming increase of transportation means and the infrastructure development has made the problem of planning and managing the transport networks much more complex. This requires powerful transportation management systems which are capable of accommodating various needs of million users. Fortunately, thanks to technology advances, intelligent transportation systems have emerged as potential tools to this challenging problem. The success of ITS is enabled by the following technologies:

- The most important module in ITS is the data collection which is available from traffic detectors in the transport networks. They can provide varied information such as flow, speed, density and so on to other ITS modules for data analysis. Furthermore, *global positioning systems (GPS)* have become cheaper and more popular, making it feasible to obtain seamless tracking information of vehicles in traffic networks. Also, with the support of computer vision techniques, smart cameras have been used to capture and process complete visual information with better coverage, e.g. counting the number of vehicles in a link and evaluating congestion levels.

- After being recorded, data must be relayed to a system centre for analysis. Wireless networks have improved so that transmitting such data only takes milliseconds (or seconds for visual data), making real-time information available to ITS. With recent advances of wireless sensor networks, it is expected that sensor detectors can be deployed without any prior network set-up; randomly located sensors can automatically form a network backbone to relay data back to a system centre.
- Finally, ITS are supported by computation power and data storage which continue to grow significantly. Still, with such an enormous amount of data, the computation power seems to be lagging behind. Hopefully, with growing research in parallel systems and algorithms, real-time transportation services can be accommodated.

In terms of functionality, example applications of ITS include:

- Search for the best transportation mode, route or parking location at a specific time. These functionalities may be embedded into in-vehicle devices.
- On the system scale, optimise the transport performance for safety, pollution, throughput, cost and congestion by different means such as traffic light signal, message sign, radio service, etc.
- Detect and manage traffic incidents. For example, in the case of traffic accident, ITS have to reroute traffic in the network, matching the traffic supplies and demands.
- Enable the use of unique smart card for different transportation modes.
- For a smart car, provide the capabilities of warning and avoiding collision, monitoring the driver behaviour for drowsiness, etc.

In the initial stage of ITS, some works directly used the current network state to derive outputs (route choice, optimised traffic sequence, etc). However, in early reports, it was realised that the full potential of ITS is only realised if the short term forecasting of the network state is continuously updated, i.e. route optimisation in the next two hours should use the network state information in two hours as inputs, instead of the current state information. By taking into account traffic dynamics from the current moment to the near future, this approach may yield better performance. Hence, aside from end services, ITS also include intermediate modules which provide derived data to other modules as inputs.

The following is a short list of special terms in transportation, which are necessary for later discussion:

- A *(traffic) flow* is defined as the number of vehicles passing a detector in a link during a unit time period. For a link with multiple lanes, flows can be counted for each lane or aggregated for the whole link.
- *Speed (time mean speed)* is defined as the average velocity of vehicles passing a detector during a unit time period.
- *Time occupancy*: When vehicles pass a specific detector in a time period T, the percentage of time that detector is occupied is called time occupancy. If the length and the speed of vehicle i passing a detector in time period T are v_i, l_i and the detector length is l_d, then the time occupancy is $o_t = \sum_i (l_i + l_d)/(v_i T))$.
- *Space occupancy*: In a link with multiple lanes, the percentage of the road length occupied by vehicles with respect to the total length is called space occupancy.
- *Density* is the ratio of the number of vehicles occupying a link to the total length of that link.

The theory of transportation and ITS is described further in Roess et al. (2004) and Slinn et al. (2005). For interested readers, more discussion of ITS can be found in Ghosh and Lee (2000). This thesis only focuses on the short term flow forecasting (STFF) of transportation networks.

3.2 STFF modelling

In this section, we will summarise some of the work on STFF. For an overview of STFF models and other transportation aspects of STFF, such as data resolution, variable types and implementation characteristics, please refer to Vlahogianni et al. (2004) and Chang et al. (2011). The meanings of notations for each following model are independent as they can overlap.

ARMA models

Classical time series modelling is one of the dominant approaches in STFF. Different variants of the ARMA have been proposed, including the simple ARMA model with differenced series (Ahmed and Cook, 1979; Hamed et al., 1995), the seasonal ARMA model (using the multiplicative form and seasonal difference operator) (Williams and Hoel, 2003; Ghosh et al., 2007) and the VAR model (Chandra and Al-Deek, 2009).

Let z_t be a univariate traffic flow at time t. In Ahmed and Cook (1979) and Hamed et al. (1995), the ARMA model is applied to the univariate differenced series $y_t = \bigtriangledown^d z_t$:

$$\begin{aligned} \phi(\mathrm{B}) \bigtriangledown^d z_t &= \theta(\mathrm{B}) e_t \qquad (3.1)\\ \Leftrightarrow \quad y_t - \sum_{i=1}^{p} \phi_i y_{t-i} &= e_t + \sum_{j=1}^{q} \theta_j e_{t-j}. \end{aligned}$$

In that case, it is said that z_t follows the *autoregressive integrated moving average model*, *ARIMA*(p, d, q). Williams and Hoel (2003) and Ghosh et al. (2007) use the *seasonal autoregressive integrated moving average* model, *SARIMA*$(p, d, q)(p_s, d_s, q_s)_s$ by taking into account a seasonal pattern with a period s. The multiplicative SARIMA is as follows:

$$\phi(\mathrm{B})\phi_s(\mathrm{B}^s) \bigtriangledown^d \bigtriangledown_s^{d_s} z_t = \theta(\mathrm{B})\theta_s(\mathrm{B}^s) e_t, \qquad (3.2)$$

with the seasonal polynomials $\phi_s(\mathrm{B}^s)$ and $\theta_s(\mathrm{B}^s)$ of the sth backward operator B^s:

$$\phi_s(\mathrm{B}^s) = 1 - \sum_{i=1}^{p_s} \phi_{s;i}\, \mathrm{B}^{si}, \qquad (3.3)$$

$$\theta_s(\mathrm{B}^s) = 1 - \sum_{i=1}^{q_s} \theta_{s;i}\, \mathrm{B}^{si}\,. \qquad (3.4)$$

The orders d and d_s of the difference operators are usually found by a preliminary analysis of autocorrelation. Basically, the original flow z_t is differenced so that the autocorrelation of the transformed series y_t is similar to that of the ARMA model. Instead of using difference operator, Hu et al. (2011) use the historical daily mean and model residuals with the ARMA. The usage of daily mean is similar to the MP preprocessing method in this thesis and Mai et al. (2012).

Despite much research on univariate ARMA variants for STFF, there are very few works on the multivariate VARMA. According to our knowledge, Chandra and Al-Deek (2009) are the first to apply the vector autoregressive model (VAR), which is a special case of the VARMA model, to STFF, exploiting the spatial correlation of traffic flows in various locations of a transportation network (z_t is then a vector variable of traffic flows). In this work, the trend and seasonality of the flow series are also eliminated by the difference operator. Recently, Min and Wynter (2011) proposed using geographical information of transportation networks in the AR and MA matrices of the VARMA model to reduce the number of unknown parameters. Still, only the result of the VAR model fit was reported in the evaluation part.

Most ARMA models in STFF are inferred by the MLE method. The only work with Bayesian inference that we are aware of is Ghosh et al. (2007) in which the authors implement MCMC for the SARIMA$(1,0,0)(0,1,1)_s$ by using an approximated likelihood. In terms of forecasting error, generally, it has been reported that the VAR model gives better prediction than the univariate ARMA model and exploiting the seasonal pattern by the s-lagged difference operator yields better performance. Detailed discussion of the ARMA model for STFF is deferred until Chapter 4.

DLMs

Ghosh et al. (2009) use a DLM model for a univariate flow z_t as follows:

$$z_t = \mu_t + \sum_{j=1}^{\lfloor s/2 \rfloor} \gamma_{t,j} + \beta^T x_t + v_t, \tag{3.5}$$

$$\mu_t = \mu_{t-1} + w_t, \tag{3.6}$$

$$\begin{pmatrix} \gamma_{t,j} \\ \gamma^\star_{t,j} \end{pmatrix} = \begin{pmatrix} \cos\lambda_j & \sin\lambda_j \\ -\sin\lambda_j & \cos\lambda_j \end{pmatrix} \begin{pmatrix} \gamma_{t-1,j} \\ \gamma^\star_{t-1,j} \end{pmatrix} + e_{t,j}, \tag{3.7}$$

with the seasonal period s; $\lambda_j = 2\pi j/s$ $(i = 1 : \lfloor s/2 \rfloor)$; $v_t \overset{iid}{\sim} N(0, \sigma_v^2)$; $w_t \overset{iid}{\sim} N(0, \sigma_w^2)$; $e_{t,j} \overset{iid}{\sim} N(0, \Sigma_e)$; x_t is a vector of known explanatory variables (previous upstream flows at time $t-k$ used as covariates at time t) with its corresponding coefficient vector β. In this model, μ_t can be interpreted as a dynamic bias corresponding to network state. Without the random noise e_t, the variables $\gamma_{t,j}$ and $\gamma^\star_{t,j}$ are exactly cyclical with respect

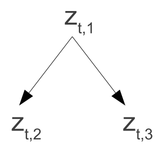


Fig. 3.1: A DAG of traffic flows in transportation networks.

to a time period s/j (the linear transformation is a rotation with angle λ_j). By using $e_{t,j}$, the magnitude of the cyclic function can be adapted to transportation networks.

The parameters σ_v^2, σ_w^2, Σ_e and β are estimated offline by the MLE; then with fixed parameters, the state vector $(\mu_t, \gamma_{t,1:\lfloor s/2 \rfloor}, \gamma^\star_{t,1:\lfloor s/2 \rfloor})$ can be inferred by the Kalman filter.

Different from the above model, Tebaldi et al. (2002) use a cubic spline for the daily pattern and allow the linear coefficients of explanatory variables to be time-dependent. Let d and t be the day and time-within-day indices. Tebaldi et al. (2002) model a traffic flow $z_{d,t}$ as:

$$z_{d,t} = \alpha_d^T y_{d,t} + \beta_{d,t}^T x_{d,t} + v_{d,t}, \tag{3.8}$$

$$\beta_{d,t} = \beta_{d,t-1} + w_{d,t}, \tag{3.9}$$

with $v_{d,t} \overset{iid}{\sim} N(0, \sigma_v^2)$; $w_{d,t} \overset{iid}{\sim} N(0, \Sigma_w)$. The daily pattern is modelled by the cubic spline part $\alpha_d^T y_{d,t}$ in which a bias term is dependent on the day d. For example, if there are two terminal knots at $t_0 = 0$ and $t_3 = 288$ and two inner knots at $t_1 = 60$ and $t_2 = 150$, then the covariate vector is $y_{d,t} = (1, t, t^2, t^3, (t - t_1)_+^3, (t - t_2)_+^3)$ and its corresponding coefficient is $\alpha_d = (\phi_d, \theta_{1:5})$. If information about previous upstream flows $x_{d,t}$ (previous upstream flows at time $t-k$ used as covariates at time t) is available, its contribution is modelled by the dynamic linear combination $\beta_{d,t}^T x_{d,t}$.

Bayesian sequential inference is used for this DLM model. The discount method (West and Harrison, 1997; Prado and West, 2010) is applied to Σ_w and a conjugate prior (inverse gamma distribution) is used on σ_v^2. Notice that for the Kalman filter, the state vector of this model is $(\alpha_d, \beta_{d,t})$ in which the variable α_d does not evolve (fixed).

Another notable work of the DLM in STFF is the *linear multiregression dynamic model (LMDM)* (Queen and Albers, 2009; Anacleto Junior et al., 2013*b*). In this work, a transportation network is represented by a DAG where each node corresponds to an observed traffic flow. For example, Figure 3.1 shows a traffic flow $z_{t,1}$ forking into two branches $z_{t,2}$ and $z_{t,3}$ (For this model, $z_{t,i}$ is a flow at location i and time t) . The nodes are divided into three categories: root nodes, children nodes and derived nodes: a root node is a node without upstream information ($z_{t,1}$ in Figure 3.1) and is modelled completely based on its previous flow values; a child node with upstream information ($z_{t,2}$ in Figure 3.1) is modelled as a dynamic linear combination of instantaneous parent flows and a derived node is calculated directly and deterministically, conditioning on its parent and sibling nodes ($z_{t,3} = z_{t,1} - z_{t,2}$, in Figure 3.1).

Assume that the seasonal period is s. The model of a root node $z_{t,i}$ is:

$$z_{t,i} = \alpha_{t,i,1} + v_{t,i}, \tag{3.10}$$

$$\alpha_{t,i,1:s} = \begin{pmatrix} a & (1-a) & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \alpha_{t-1,i,1:s} + \begin{pmatrix} w_{t,i} \\ 0 \\ \dots \\ 0 \\ 0 \end{pmatrix}, \tag{3.11}$$

where $v_{t,i} \overset{iid}{\sim} N(0, \sigma^2_{v_i})$; $w_{t,i} \overset{iid}{\sim} N(0, \sigma^2_{w_i})$; a is a predefined value; $\alpha_{t,i,1:s} = (\alpha_{t,i,1}, ..., \alpha_{t,i,s})$. Intuitively, the vector $\alpha_{t,i,1:s}$ is a revolving flow vector for a full day. A child node is modelled as:

$$z_{t,i} = \Phi^T_{t,i,1:d_x,1} x_{t,i} + v_{t,i}, \tag{3.12}$$

$$\Phi^T_{t,i,j,1:s} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \Phi^T_{t-1,i,j,1:s} + \begin{pmatrix} w_{t,i,j} \\ 0 \\ \dots \\ 0 \\ 0 \end{pmatrix}, \tag{3.13}$$

where $\Phi_{t,i,1:d_x,k}$, $\Phi_{t,i,j,1:s}$ are a column and a row of a $d_x \times s$ matrix $\Phi_{t,i,1:d_x,1:s}$ ($\dim(x_{t,i}) = d_x$); $v_{t,i} \overset{iid}{\sim} N(0, \sigma^2_{v_i})$; $w_{t,i,j} \overset{iid}{\sim} N(0, \sigma^2_{w_{i,j}})$. The instantaneous parent flow vector $x_{t,i}$ (flows at time t) is linearly combined with the coefficient vector $\Phi_{t,i,1:d_x,1}$. Each element of the coefficient vector is revolving by a daily pattern of Equation (3.13).

Each sub-model is a DLM. Hence, conditioning on all the flows $z_{1:t,i}$ ($1 \leq i \leq n_l$), the filtering of the state, $\alpha_{t,i,1:s}$ and $\Phi_{t,i,1:d_x,1:s}$, can be carried out sequentially. The discount method is used for the evolution variance and a conjugate prior (gamma distribution) can be used for the observation variance. For the forecasting, the prediction densities of root-nodes follow normal distributions but the marginal prediction densities of children nodes are non-trivial. So, for this purpose, the prediction means and variances of children nodes are evaluated recursively from root nodes by using the conditional expectation and variance formulae:

$$\begin{aligned} \mathrm{E}_x(x) &= \mathrm{E}_y(\mathrm{E}_{x|y}(x)), && (3.14) \\ \mathrm{Var}_x(x) &= \mathrm{Var}_y(\mathrm{E}_{x|y}(x)) + \mathrm{E}_y(\mathrm{Var}_{x|y}(x)), && (3.15) \end{aligned}$$

of two random variables x, y.

Queen et al. (2007) discuss in details the modelling of traffic networks using DAG by the same LMDM. However, modelling equations for root nodes and child nodes are different from Equations (3.10) and (3.12).

Even though these works belong to the DLM, the interpretation of each model is quite different. Compared to ARMA models, these models focus on the mean of the traffic flow, taking into account the seasonal traffic pattern. Further discussion of the DLM (and the DM) is in Chapter 6.

Neural networks

The *Neural network* model (*NN*) is another popular modelling class in STFF. The probability representation of a two-layer NN for an output flow z_t with an input vector x_t of length d_x and d_{u_2} hidden units $u_{2,i}$ is:

$$\begin{aligned} z_t &= f(x_t, \beta) + v_t = u_{3,1} + v_t, && (3.16) \\ u_{3,1} &= g_{3,1}(\beta_{3,1,0} + \beta^T_{3,1,1:d_{u_2}} u_{2,1:d_{u_2}}) \\ &= g_{3,1}(\beta^T_{3,1,0:d_{u_2}} u_{2,0:d_{u_2}}), && (3.17) \\ u_{2,i} &= g_{2,i}(\beta_{2,i,0} + \beta^T_{2,i,1:d_x} x_t) \\ &= g_{2,i}(\beta^T_{2,1,0:d_x} u_{1,0:d_x}), && (3.18) \end{aligned}$$

where $\beta = (\beta_{2,1,0:d_x}, ..., \beta_{2,d_{u_2},0:d_x}, \beta_{3,1,0:d_{u_2}})$; $v_t \overset{iid}{\sim} N(0, \sigma^2_v)$. The activation function for the output unit in STFF is usually the identity function: $g_{3,1}(y) = y$; for hidden units, the activation $g_{2,i}$ is either the Gaussian function or the tanh function. Notice that by this definition, the NN is not a time-dependent model (every data point is treated as an iid sample).

Details about the NN model can be found in Bishop (2006,chap. 5). The parameter vector β is estimated by minimising the sum of squared error (equivalent to the MLE method). An advantageous point of the NN model is that derivatives of $f(x_t, \beta)$ with respect to β can be obtained computationally efficiently by the chain rule (and hence the derivatives of the sum of squared error). Then, an optimisation procedure such as gradient or Newton methods can be used for parameter estimation.

In STFF context, the input vector x_t may include previous traffic flows at the same location as the output (Chen and Grant-Muller, 2001), traffic flows at various sites (Vlahogianni et al., 2005) and other traffic variables such as speed and occupancy (Dougherty and Cobbett, 1997; Zhang, 2000). Chen and Grant-Muller (2001) apply a rule to increase the number of hidden units when the error prediction by the current model exceeds a predefined threshold. Chen et al. (2001) use the NN to classify the network state first and then fit either a ARMA or another NN for the traffic flow in that state.

The most important problem of the NN is the ambiguity in model interpretation. Aside from understanding that each hidden unit acts as a basis function in linear regression, it is almost impossible to interpret the meaning of each hidden unit. This issue becomes more notable when multiple NN layers are used. Without understanding the relation between variables, it is very difficult to analyse a model and propose a better one. Also, the sequential variant of the NN estimation is questionable in terms of convergence. Instead of minimising the sum of squared error for all data points, the sequential procedure only minimises the error of the newest data point by using a previous optimisation value as a starting point.

Finite mixture of distributions

In Sun et al. (2006), previous flows z_{t-1} and previous upstream flows $z_{u;t-1}$ are used in one-step prediction of z_t by finite mixture of Gaussian distributions. Let $y_t = z_t$ and $x_t = (z_{t-1}, z_{u;t-1})$. The modelling of Sun et al. (2006) is as follows:

$$p_{x_t,y_t|w,\mu,\Sigma}(x_t, y_t) = \sum_{i=1}^{m} w_i N(x_t, y_t|\mu_i, \Sigma_i), \tag{3.19}$$

with normalized weights w, normal means $\mu_{1:m}$ and variances $\Sigma_{1:m}$, which are estimated by a modified *Expectation-Maximisation (EM)* algorithm. All the observations

(x_t, y_t) are iid samples from the above distribution, which may not portray the temporal dynamics of traffic flows. Also, in this modelling, there is a potential statistical inconsistency as p_{z_t} and $p_{z_{t-1}}$ may have different marginal mixture distributions which should be identical according the iid assumption.

Nonparametric approaches

Aside from the above approaches in STFF, Smith et al. (2002) and Clark (2003) use the *k-nearest neighbour method (kNN)* (Härdle, 2004) for a regression problem of an output flow z_t and a covariate vector x_t. The covariate vector x_t may include previous flows, speed and occupancy. Then, given a new covariate $x_{t'}$, the estimated flow $\widehat{z}_{t'}$ is:

$$\widehat{z}_{t'} = \sum_{t \in \mathscr{K}} z_t \alpha_{t,t'}, \tag{3.20}$$

with $\mathscr{K}$ is an index list of k-nearest neighbours of $x_{t'}$; $\sum_{t \in \mathscr{K}} \alpha_{t,t'} = 1$. Despite being robust, these *nonparametric* approaches may need a lot of data and may be quite slow. It is seen that the kNN does not consider the time dynamic of the traffic networks.

Multi-step-ahead forecasting

Among all the above works, only a few works analyse and report the multi-step-ahead forecasting, e.g. Tebaldi et al. (2002); Ghosh et al. (2009) and Min and Wynter (2011). Personally, we think that multi-step-ahead forecasting module must be implemented to fully support functionalities of ITS (e.g. find a best route in the next two hours). Hence, this issue will be discussed later in Chapters 4 and 6.

3.3 Related transportation problems

This section illustrates two related problems in ITS which may motivate STFF modelling.

Road traffic control

The first problem we look at is the *traffic signal control* optimisation (Papageorgiou et al., 2003). However, we will only focus on the modelling of urban transportation networks, not the optimisation method. In Aboudolas et al. (2009) and Aboudolas

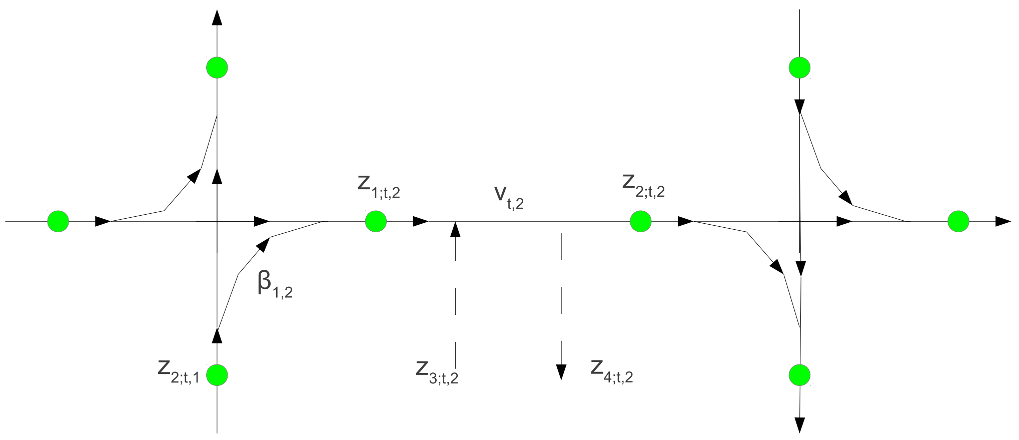


Fig. 3.2: An illustrated traffic network for the model in Aboudolas et al. (2010) with inflow $z_{1;t,i}$, outflow $z_{2;t,i}$, demand-flow $z_{3;t,i}$, exit-flow $z_{4;t,i}$, turning rate $\beta_{i',i}$ and the number of vehicles within a link $\nu_{t,i}$. Green circles denote the detector locations for inflows and outflows.

et al. (2010), a deterministic model is used for a urban transportation network of links i $(1 \leq i \leq n_l)$. A link in this work is a whole road link connecting two junctions, not a segment of a road.

Denote $\mathscr{B}_i$ as the set of immediate upstream links i' of link i (a traffic flow comes from link i to link i' through a connecting junction); $z_{1;t,i}$, $z_{2;t,i}$, $z_{3;t,i}$ and $z_{4;t,i}$ are the inflow, the outflow, the demand-flow and the exit-flow of link i at time t (a demand-flow of a link accounts for the vehicles coming from car parking, houses to that link; an exit-flow is for vehicles stopping at that link); $\nu_{t,i}$ is the number of vehicles within link i at the end of time t.

In a signal cycle of length c, a traffic signal sequence is divided into many stages k of green time and the green time of stage k of junction j is $g_{j,k}$. For a link i and its immediate downstream junction $j_{2;i}$, $\mathscr{D}_i$ is the set of signal stages of junction $j_{2;i}$ that link i has its right of way.

The dynamic of link i is modelled by the following conservation equation:

$$\nu_{t,i} = \nu_{t-1,i} + z_{1;t,i} - z_{2;t,i} + z_{3;t,i} - z_{4;t,i}. \tag{3.21}$$

The demand-flow $z_{3;t,i}$ is assumed to be known and the exit-flow is proportional to the inflow:

$$z_{4;t,i} = \alpha_i z_{1;t,i}, \tag{3.22}$$

with a known exit rate α_i. The inflow of link i is in turn the sum of all flows coming from the upstream links $i' \in \mathscr{B}_i$:

$$z_{1;t,i} = \sum_{i' \in \mathscr{B}_i} \beta_{i',i} z_{2;t,i'}, \tag{3.23}$$

where $\beta_{i',i}$ is the turning rate from link i' to link i. Finally, the outflow $z_{2;t,i}$ is modelled as

$$z_{2;t,i} = \frac{\gamma_i (\sum_{k \in \mathscr{D}_i} g_{j_{2;i},k})}{c}, \tag{3.24}$$

with a known saturation flow γ_i. The multivariate version of Equation (3.21) for all the links is:

$$\nu_{t,1:n_l} = \nu_{t-1,1:n_l} + Ag + z_{3;t,1:n_l}, \tag{3.25}$$

where the matrix A can be obtained from Equations (3.21) to (3.24); g is the vector of all green time $g_{j,k}$. Equation (3.25) is then used to optimise some criteria to obtain the best green time g.

Two notable remarks of this deterministic model are the introduction of the variable $\nu_{t,i}$ (the number of vehicles within a link) and the conservation equation. It turns out later that the current number of vehicles $\nu_{t-1,i}$ is strongly related to the next outflow $z_{2;t,i}$ and can be used as a very good regressor. Furthermore, for multi-step ahead forecasting, we usually predict network state $(t + 1)$ from state t and continue the prediction of time $(t + 2)$, conditionally based on the prediction at time $(t + 1)$. Whenever the prediction step increases, more network information is lost and the uncertainty increases. Even though this fact always occurs with any model, we think that the loss can be reduced by applying logical constraints of traffic theory. In this model, the conservation equation is used and the network state is preserved to an extent; vehicles in a network cannot disappear all of a sudden. A forecasting model will be proposed in Chapter 6 based on these two remarks.

Estimating the number of vehicles within a link

The number of vehicles in a road signal control study may be observed directly by a video detector. Though this approach is feasible thanks to computer vision techniques, the implementation cost may not be affordable for large scale transportation networks at the moment. Hence, there has been significant research on estimating this variable.

Gazis and Liu (2003) use a different version of Equation (3.21), omitting the demand-flow, exit-flow and adding uncertainty to the inflow and outflow observations:

$$\nu_{t,i} = \nu_{t-1,i} + z_{1;t,i} - z_{2;t,i} + w_{1;t,i} - w_{2;t,i}, \tag{3.26}$$

where $w_{j;t,i}$ is a corresponding noise for $z_{j;t,i}$. The density of link i is $k_{t,i} = \nu_{t,i}/l_i$ with the link length l_i; a constant known density of link i in free-flow condition is denoted by $k_{f;i}$. The observed speed $v_{t,i}$ is then linked to the density by a well known relationship in traffic theory:

$$\begin{aligned} v_{t,i} &= v_{f;i} \exp\left[-\frac{1}{2}\left(\frac{k_{t,i}}{k_{f;i}}\right)^2\right] + e_{t,i} \qquad (3.27) \\ &= f(\nu_{t,i}) + e_{t,i}, \end{aligned}$$

where $v_{f;i}$ is a known free-flow speed of link i. Then, the EKF can be used for the sequential inference of the DM defined by Equations (3.26) and (3.27).

Singh and Li (2012) use transformed log-speed for the linear observation equation (instead of the non-linear Equation (3.27)) and investigates the estimation method for the scenario of multilane link (with lane-changing property).

Papageorgiou and Vigos (2008) and Vigos and Papageorgiou (2010) use a different relationship between $\nu_{t,i}$ and the observed time occupancy $o_{1;t,i}$:

$$\begin{aligned} o_{1;t,i} &\approx o_{2;t,i} \qquad (3.28) \\ &= \frac{\nu_{t,i} l_v}{l_i m_i}, \qquad (3.29) \end{aligned}$$

where $o_{2;t,i}$ is the space occupancy of link i; l_v is the average vehicle length; l_i and m_i are the link length and the number of lanes in the link. For interested readers, the logical meaning of the approximation is discussed in Papageorgiou and Vigos (2008) and Vigos and Papageorgiou (2010). It has also been reported that the error becomes smaller if the approximation is used for smaller divided segments of a link and the aggregated estimate is obtained by the sum of all element estimates.

So, with a transform function, the "observed" (imaginary) number of vehicles $\nu_{o;t,i}$ is related with the unknown state $\nu_{t,i}$ (true number of vehicles):

$$\begin{aligned} \frac{o_{1;t,i} l_i m_i}{l_v} &= \nu_{o;t,i} \qquad (3.30) \\ &= \nu_{t,i} + e_{t,i}, \qquad (3.31) \end{aligned}$$

where $e_{t,i}$ is the observation error. The Kalman filter is then can be applied to Equations (3.26) and (3.30).

3.4 Conclusion

ITS has been briefly summarised in this chapter, with the dependency between ITS applications and the short term forecasting models. For the specific STFF problem, most of modelling classes are presented. Further discussion of the ARMA model and the DM will be found in subsequent chapters. We have also mentioned two other transportation problems: the traffic signal control optimisation and the estimation of number of vehicles. These are related to the spatial temporal model in Chapter 6.

Chapter 4

The VARMA model for the short term traffic flow forecasting

In this chapter, we will apply the VARMA model to the STFF problem. Firstly, the VARMA model is presented again with some modifications, regarding a sparse form of the VARMA, a user-defined constraint on the AR and MA matrices and the DLM-VARMA representation. For the statistical inference, the likelihood of the DLM-VARMA representation and some useful transformations are derived; the Bayesian inference is implemented by MCMC. Specifically, a MCMC scheme is proposed to handle the serial correlation issue of the VARMA model. Then, two real datasets of traffic flow $z_{t,i}$ in Dublin City centre will be described with a preliminary analysis of autocorrelation. The residual $y_{t,i}$ is extracted from the flow $z_{t,i}$ by different preprocessing procedures, e.g. using the difference operator or the daily mean, and modelled by the VARMA model. Finally, we discuss and analyse the result of the multi-step-ahead forecasting. The work in this chapter is being published in Mai et al. (2012) and Mai et al. (2013).

4.1 Bayesian inference for the VARMA model

As in Section 2.4.1, the VARMA model of a random vector $y_t = y_{t,1:d_y}$ at time t ($1 \leq t \leq n$) is:

$$\begin{aligned} \mathbb{\Phi}(\mathrm{B})(y_t - \beta) &= \mathbb{\Theta}(\mathrm{B})e_t \qquad (4.1) \\ \Leftrightarrow \quad (y_t - \beta) - \sum_{j=1}^{p} \mathit{\Phi}_j(y_{t-j} - \beta) &= e_t + \sum_{j'=1}^{q} \mathit{\Theta}_{j'} e_{t-j'} \end{aligned}$$

where β is a mean vector of length d_y; $e_t \overset{iid}{\sim} N(0, \mathit{\Sigma}_e = Q_e^{-1})$; each AR (or MA) parameter, $\mathit{\Phi}_j$ (or $\mathit{\Theta}_{j'}$) is a $d_y \times d_y$ matrix; $\mathbb{\Phi}(\mathrm{B})$ (or $\mathbb{\Theta}(\mathrm{B})$) is a $d_y \times d_y$ matrix of polynomials of the backward shift operator B: $\mathbb{\Phi}(\mathrm{B}) = I_{d_y} - \sum_{j=1}^{p} \mathit{\Phi}_j \, \mathrm{B}^j$ ($\mathbb{\Theta}(\mathrm{B}) = I_{d_y} - \sum_{j'=1}^{q} \mathit{\Theta}_{j'} \, \mathrm{B}^{j'}$).

For a possible seasonal effect, e.g. an effect at periodic lag s, instead of using the multiplicative form like Equation (3.2), we directly use a AR matrix $\mathit{\Phi}_s$ (or $\mathit{\Theta}_s$). The reason for this is that the MCMC can be implemented more efficiently, which will be explained later. However, not all the matrices $\mathit{\Phi}_j$ ($j = 1 : s$) are needed as we only want to include AR or MA matrices of lag j at which there is a possible influence of y_{t-j} or e_{t-j} to the current y_t. Hence, for a sparse form of the VARMA model, define sets $\mathscr{P}$ and $\mathscr{Q}$ where:

$$j \in \mathscr{P} \Leftrightarrow \mathit{\Phi}_j \neq 0, \qquad (4.2)$$

$$j \in \mathscr{Q} \Leftrightarrow \mathit{\Theta}_j \neq 0. \qquad (4.3)$$

Note that $p = \max(\mathscr{P})$ and $q = \max(\mathscr{Q})$.

At lag j, the matrix element $\mathit{\Phi}_{j,i,i'}$ (at row i and column i') is the linear coefficient of $y_{t-j,i'}$ to $y_{t,i}$, according to Equation (4.1). On a specific application, we may know or assume whether such a linear effect of $y_{t-j,i'}$ on $y_{t,i}$ exists. So, rather than using a full matrix $\mathit{\Phi}_j$, we can specify the locations of non-zero elements for the AR matrix $\mathit{\Phi}_j$ by using an indicator matrix $M_{\mathit{\Phi};j}$ where:

$$M_{\mathit{\Phi};j,i,i'} = 1 \Leftrightarrow \mathit{\Phi}_{j,i,i'} \neq 0, \qquad (4.4)$$

$$M_{\mathit{\Phi};j,i,i'} = 0 \Leftrightarrow \mathit{\Phi}_{j,i,i'} = 0. \qquad (4.5)$$

Similarly, a matrix $M_{\mathit{\Theta};j}$ is defined for each MA matrix $\mathit{\Theta}_j$. These constraints on the AR and MA matrices reduce the number of unknown parameters based on the information of the target problem. Also, for notation purposes, denote the vector representations of non-zero elements of $\Phi = \Phi_{1:p}$ and $\Theta = \Theta_{1:q}$ by φ ($\dim(\varphi) = d_\varphi$) and ϑ ($\dim(\vartheta) = d_\vartheta$) correspondingly (the vector and matrix representations are used interchangeably in this chapter).

4.1.1 The DLM representation of the VARMA model

The d_y-variate VARMA model of Equation (4.1) can be represented by a DLM:

$$y_t = \beta + \alpha_{t,1}, \tag{4.6}$$

$$\alpha_{t,1:m} = G\alpha_{t-1,1:m} + u_t, \tag{4.7}$$

and:

$$G = \begin{pmatrix} \Phi_1 & I_{d_y} & 0 & \dots & 0 \\ \Phi_2 & 0 & I_{d_y} & \dots & 0 \\ \dots & \dots & \dots & \dots & \\ \Phi_{m-1} & 0 & 0 & \dots & I_{d_y} \\ \Phi_m & 0 & 0 & \dots & 0 \end{pmatrix}, \tag{4.8}$$

$$u_t = He_t, \tag{4.9}$$

$$H = (I_{d_y}, \Theta_1, ..., \Theta_{m-1})^T, \tag{4.10}$$

with $m = \max(p, q+1)$; $\Phi_j = 0$ for $j > p$; $\Theta_j = 0$ for $j > q$; the symbol I_d denotes the $d \times d$ identity matrix; each segment $\alpha_{t,k}$ is a vector of length d_y; $\alpha_{t,1:m} = (\alpha_{t-1,1}, ..., \alpha_{t-1,m})$ is a vector of length $d_\alpha = d_y m$. Notice that there is no observation error in this DLM. This representation reduces the dimension of latent variables from $(p+q)d_y$ ($y_{0:(-p+1)}$ and $e_{0:(-q+1)}$) to $m = \max(p, q+1)d_y$ ($\alpha_0 = \alpha_{0,1:m}$), which can be useful in high dimension problems. We denote this representation by DLM-VARMA.

The unknown parameters and latent variables of the DLM-VARMA include φ (or Φ), ϑ (or Θ), β, Σ_e and α_0 (other elements such as $\mathscr{P}$, $\mathscr{Q}$, $M_{\Phi;j}$ and $M_{\Theta;j}$ must be predefined).

The likelihood of the DLM-VARMA and some transformations

From Equation (4.8), we have:

$$\alpha_{t,k} = \Phi_k \alpha_{t-1,1} + \alpha_{t-1,k+1} + \Theta_{k-1} e_t, \tag{4.11}$$

which is true for all integer $k \geq 1$ (the notation is extended by: $\alpha_{t,k} = 0$ for $k > m$, $\Phi_j = 0$ for $j > p$, $\Theta_j = 0$ for $j > q$ and $\Theta_0 = I$). Then with Equations (4.6) and (4.11):

$$\begin{aligned} y_t &= \beta + \alpha_{t,1} \\ &= \beta + \Phi_1 \alpha_{t-1,1} + \alpha_{t-1,2} + \Theta_0 e_t, \\ &= \beta + \sum_{j=1}^{t} \Phi_j \alpha_{t-j,1} + \alpha_{0,t+1} + \sum_{j_2=1}^{t} \Theta_{j_2-1} e_{t-j_2+1}, \\ &= \beta + \sum_{j=1}^{t-1} \Phi_j (y_{t-j} - \beta) + \Phi_t \alpha_{0,1} + \alpha_{0,t+1} + e_t + \sum_{j'=1}^{t-1} \Theta_{j'} e_{t-j'}, \end{aligned} \tag{4.12}$$

Hence, conditioning on $(\Phi, \Theta, \beta, \Sigma_e, \alpha_0)$, $y_{1:n}$ is an one-to-one linear transform of $e_{1:n}$ with *Jacobian* $|\mathfrak{J}_{e_{1:n}}(y_{1:n})| = 1$ and:

$$p_{y_{1:n}|\Phi,\Theta,\beta,\Sigma_e,\alpha_0}(y_{1:n}) = p_{e_{1:n}|\Phi,\Theta,\beta,\Sigma_e,\alpha_0}(e_{1:n}) \tag{4.13}$$

The subsequent transforms are re-formulated for the multivariate version, following the outline proof of the univariate version in Chib and Greenberg (1994).

Theorem 3 *(Chib and Greenberg, 1994) For* $t = 1 : n$*, define:*

$$a_{1;t} = y_t - \Phi_t \alpha_{0,1} - \alpha_{0,t+1} - \sum_{j=1}^{t-1} \Phi_j y_{t-j} - \sum_{j'=1}^{t-1} \Theta_{j'} a_{1;t-j'}, \tag{4.14}$$

$$B_{1;t} = I_{d_y} - \sum_{j=1}^{t-1} \Phi_j - \sum_{j'=1}^{t-1} \Theta_{j'} B_{1;t-j'}, \tag{4.15}$$

then:

$$a_{1;t} - B_{1;t}\beta = e_t, \tag{4.16}$$

for $t = 1 : n$*.*

Proof Prove Theorem 3 for the special case $t = 1$ by using Equation (4.12). Then, assume that Equation (4.16) is true for $t = 1 : (m-1)$, substitute e_t $(t = 1 : (m-1))$ in Equation (4.12) to obtain the proof for $t = m$. Consequently, by induction, Theorem 3 is true for $t = 1 : n$. ∎

Theorem 4 *(Chib and Greenberg, 1994) Let* $x_t = y_t - \beta$*. Define:*

$$a_{2;t} = x_t - \sum_{j=1}^{t-1} \Phi_j x_{t-j} - \sum_{j'=1}^{t-1} \Theta_{j'} a_{2;t-j'}, \tag{4.17}$$

$$B_{2;t} = C_t - \sum_{j'=1}^{t-1} \Theta_{j'} B_{2;t-j'}, \tag{4.18}$$

where the matrix $C_t = (C_{t,1}, ..., C_{t,m})$ $(\dim(C_t) = (d_y, d_y m)$ *with* $C_{t,k=1} = \Phi_t$, $C_{t,k=t+1} = I_{d_y}$ *and* $C_{t,k} = 0$ *for* $k \neq 1, (t+1)$. *Then,*

$$a_{2;t} - B_{2;t}\alpha_0 = e_t, \tag{4.19}$$

for $t = 1 : n$.

Theorem 5 *(Chib and Greenberg, 1994) Let* $x_t = y_t - \beta$. *Define:*

$$a_{3;t} = x_t - \alpha_{0,t+1} - \sum_{j'=1}^{t-1} \Theta_{j'} a_{3;t-j'}, \tag{4.20}$$

$$B_{3;t}\varphi = \Phi_t \alpha_{0,1} + \sum_{j=1}^{t-1} \Phi_j x_{t-j} - \sum_{j'=1}^{t-1} \Theta_{j'} B_{3;t-j'}\varphi \tag{4.21}$$

then:

$$a_{3;t} - B_{3;t}\varphi = e_t, \tag{4.22}$$

for $t = 1 : n$.

Theorems 4 and 5 are proved similarly by induction. Notice that Equation (4.21) involves the transformation of AR parameters from the matrix format Φ to the vector format φ.

The priors and posteriors

We use the following priors for the DLM-VARMA:

$$\begin{aligned} p_{\varphi,\vartheta,\beta,\Sigma_e,\alpha_0}(\varphi, \vartheta, \beta, \Sigma_e, \alpha_0) &= p_\varphi(\varphi)p_\vartheta(\vartheta)p_\beta(\beta)p_{\Sigma_e}(\Sigma_e)p_{\alpha_0}(\alpha_0) && (4.23)\\ &= N(\varphi|\mu_\varphi, \Sigma_\varphi = Q_\varphi^{-1})N(\vartheta|\mu_\vartheta, \Sigma_\vartheta = Q_\vartheta^{-1}) \\ &\quad N(\beta|\mu_\beta, \Sigma_\beta = Q_\beta^{-1})IW(\Sigma_e|\Psi_{\Sigma_e}, \nu_{\Sigma_e}) \\ &\quad N(\alpha_0|\mu_{\alpha_0}, \Sigma_{\alpha_0} = Q_{\alpha_0}^{-1}) && (4.24) \end{aligned}$$

The full conditional posterior of β is derived from Equations (4.13), (4.16) and (4.24):

$$\begin{aligned} & p_{\beta|\varphi,\vartheta,\Sigma_e,\alpha_0,y_{1:n}}(\beta) \propto p_\beta(\beta)p_{e_{1:n}|\varphi,\vartheta,\beta,\Sigma_e,\alpha_0}(e_{1:n}) && (4.25)\\ & \propto \exp\left(-\frac{1}{2}(\beta - \mu_\beta)^T Q_\beta(\beta - \mu_\beta) - \frac{1}{2}\sum_{t=1}^n (a_{1;t} - B_{1;t}\beta)^T Q_e (a_{1;t} - B_{1;t}\beta)\right) \\ & \propto \exp\left(-\frac{1}{2}\left[\beta^T(Q_\beta + \sum_{t=1}^n B_{1;t}^T Q_e B_{1;t})\beta - 2\beta^T(Q_\beta\mu_\beta + \sum_{t=1}^n B_{1;t}^T Q_e a_{1;t})\right]\right) \\ & \propto N(\beta|\mu'_\beta, \Sigma'_\beta = Q'^{-1}_\beta), && (4.26) \end{aligned}$$

with:

$$Q'_\beta = Q_\beta + \sum_{t=1}^{n} B_{1;t}^T Q_e B_{1;t}, \tag{4.27}$$

$$\mu'_\beta = Q'^{-1}_\beta (Q_\beta \mu_\beta + \sum_{t=1}^{n} B_{1;t}^T Q_e a_{1;t}). \tag{4.28}$$

Similarly, we have the full conditional posteriors of α_0 and φ:

$$p_{\alpha_0|\varphi,\vartheta,\beta,\Sigma_e,y_{1:n}}(\alpha_0) = N(\alpha_0|\mu'_{\alpha_0}, \Sigma'_{\alpha_0} = Q'^{-1}_{\alpha_0}), \tag{4.29}$$

$$p_{\varphi|\vartheta,\beta,\Sigma_e,\alpha_0,y_{1:n}}(\varphi) = N(\varphi|\mu'_\varphi, \Sigma'_\varphi = Q'^{-1}_\varphi), \tag{4.30}$$

where μ'_{α_0}, Q'_{α_0}, μ'_φ and Q'_φ are:

$$Q'_{\alpha_0} = Q_{\alpha_0} + \sum_{t=1}^{n} B_{2;t}^T Q_e B_{2;t}, \tag{4.31}$$

$$\mu'_{\alpha_0} = Q'^{-1}_{\alpha_0} (Q_{\alpha_0} \mu_{\alpha_0} + \sum_{t=1}^{n} B_{2;t}^T Q_e a_{2;t}), \tag{4.32}$$

$$Q'_\varphi = Q_\varphi + \sum_{t=1}^{n} B_{3;t}^T Q_e B_{3;t}, \tag{4.33}$$

$$\mu'_\varphi = Q'^{-1}_\varphi (Q_\varphi \mu_\varphi + \sum_{t=1}^{n} B_{3;t}^T Q_e a_{3;t}). \tag{4.34}$$

By conjugacy, the full conditional posterior of Σ_e follows the inverse Wishart distribution:

$$p_{\Sigma_e|\varphi,\vartheta,\beta,\Sigma_e,\alpha_0,y_{1:n}}(\Sigma_e) \propto p_{\Sigma_e}(\Sigma_e) p_{e_{1:n}|\varphi,\vartheta,\beta,\Sigma_e,\alpha_0}(e_{1:n}) \tag{4.35}$$

$$\propto |\Sigma_e|^{-\left(\frac{\nu_{\Sigma_e}+d_y+1}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{tr}(\Psi_{\Sigma_e}\Sigma_e^{-1})\right) |\Sigma_e|^{-\frac{n}{2}} \exp\left(-\frac{1}{2}\sum_{t=1}^{n} e_t^T \Sigma_e^{-1} e_t\right)$$

$$\propto |\Sigma_e|^{-\left(\frac{n+\nu_{\Sigma_e}+d_y+1}{2}\right)} \exp\left(-\frac{1}{2}\left[\sum_{i,i'} \Sigma_{e;i,i'}^{-1}\left(\Psi_{\Sigma_e;i',i} + \sum_{t=1}^{n} e_{t,i}e_{t,i'}\right)\right]\right)$$

$$\propto IW(\Sigma_e|\Psi'_{\Sigma_e}, \nu'_{\Sigma_e}) \tag{4.36}$$

with:

$$\Psi'_{\Sigma_e} = \Psi_{\Sigma_e} + \sum_{t=1}^{n} e_t e_t^T, \tag{4.37}$$

$$\nu'_{\Sigma_e} = \nu_{\Sigma_e} + n. \tag{4.38}$$

Unfortunately, the full conditional posterior of ϑ does not follow any standard distribution form. Hence, Metropolis-Hastings must be used to sample ϑ in the next section.

4.1.2 The first MCMC algorithm

Bayesian inference on the joint posterior $p_{\varphi,\vartheta,\beta,\Sigma_e,\alpha_0|y_{1:n}}(\cdot)$ can be implemented by a standard MCMC in Algorithm 10, using Gibbs sampling and Metropolis-Hastings.

Algorithm 10 The first MCMC implementation on the DLM-VARMA

```
1. Sample $\beta$ from $p_{\beta|\varphi,\vartheta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot) = N(\cdot|\mu'_\beta, \Sigma'_\beta = Q'^{-1}_\beta)$.
2. Sample $\alpha_0$ from $p_{\alpha_0|\varphi,\vartheta,\beta,\Sigma_e,y_{1:n}}(\cdot) = N(\cdot|\mu'_{\alpha_0}, \Sigma'_{\alpha_0} = Q'^{-1}_{\alpha_0})$.
3. Sample $\Sigma_e$ from $p_{\Sigma_e|\varphi,\vartheta,\beta,\alpha_0,y_{1:n}}(\cdot) = IW(\cdot|\,\Psi'_{\Sigma_e}, \nu'_{\Sigma_e})$.
4. Sample $\varphi$ from $p_{\varphi|\vartheta,\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot) = N(\cdot|\mu'_\varphi, \Sigma'_\varphi = Q'^{-1}_\varphi)$.
5. Use Metropolis-Hastings on $p_{\vartheta|\varphi,\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$
    5-1 Propose a new value $\vartheta_b$ by a random walk centred on the current value $\vartheta_a$:
        $\vartheta_b \sim N(\cdot|\vartheta_a, \sigma_r^2 I_{d_\vartheta})$.
    5-2 Accept $\vartheta_b$ with the Metropolis-Hastings rate $\gamma(\vartheta_a, \vartheta_b)$.
```

We only sample α_0 for a number of iterations m_{α_0} (usually 1000 iterations) and then fix this variable to reduce the running time and the storage requirement as the length of vector α_0 is usually large, requiring a lot of computation. In a long time series, the starting latent variable has little effect on the inference (Box et al., 2008,chap. 7). For this purpose, we try the MLE with different fixed values of α_0 and find that the MLE results are indeed very similar to each other. So, the above step of sampling and fixing α_0 in the MCMC procedure can be regarded as choosing a reasonable starting value for α_0.

The MCMC algorithm will be tested with the following examples.

Example 1 1000 data points y_t are generated from the DLM-VARMA with parameters:

$$\mathscr{P}^\star = (1, 10), \qquad \mathscr{Q}^\star = (1, 10), \tag{4.39}$$

$$\Phi_1^\star = \begin{pmatrix} 0.70 & -0.15 \\ 0.18 & -0.60 \end{pmatrix}, \qquad \Theta_1^\star = \begin{pmatrix} -0.60 & 0 \\ 0 & 0.50 \end{pmatrix},$$

$$\Phi_{10}^\star = \begin{pmatrix} -0.30 & 0 \\ 0 & 0.20 \end{pmatrix}, \qquad \Theta_{10}^\star = \begin{pmatrix} 0.20 & 0 \\ 0 & -0.10 \end{pmatrix},$$

$$\beta^\star = (5, 100), \qquad \Sigma_e^\star = \begin{pmatrix} 67.02 & -4.87 \\ -4.87 & 44.92 \end{pmatrix},$$

and $\alpha_0^\star$ is randomly chosen ($\dim(\alpha_0^\star) = 22$). Notice that the corresponding AR and MA parameters in vector format are $\varphi^\star = (0.70, 0.18, -0.15, -0.60, -0.30, 0.20)$, $\vartheta^\star = (-0.60, 0.50, 0.20, -0.10)$. The indicator matrices for the true model are:

$$M_{\Phi;1}^\star = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad M_{\Theta;1}^\star = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{4.40}$$

$$M_{\Phi;10}^\star = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad M_{\Theta;10}^\star = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Example 2 Another dataset with 1000 data points are generated from a model similar to Example 1 but with different Θ, $\Sigma_e^\star$ and $\alpha_0^\star$ (the last two are randomly chosen):

$$\Theta_1^\star = \begin{pmatrix} -0.60 & 0.12 \\ -0.15 & 0.50 \end{pmatrix}, \Theta_{10}^\star = \begin{pmatrix} 0.20 & 0 \\ 0 & -0.10 \end{pmatrix}, \Sigma_e^\star = \begin{pmatrix} 31.69 & -25.19 \\ -25.19 & 61.02 \end{pmatrix}. \tag{4.41}$$

Compared to Example 1, there are non-zero non-diagonal elements in $\Theta_1^\star$ in this example. Correspondingly, we have $\varphi^\star = (0.70, 0.18, -0.15, -0.60, -0.30, 0.20)$, $\vartheta^\star = (-0.60, -0.150.12, 0.50, 0.20, -0.10)$.

The prior parameters for both examples are chosen similarly (only different in the dimension of ϑ):

$$p_\varphi(\varphi) = N(\varphi|\mu_\varphi = 0, \Sigma_\varphi = 1000\ I_{d_\varphi}), \tag{4.42}$$

$$p_\vartheta(\vartheta) = N(\vartheta|\mu_\vartheta = 0, \Sigma_\vartheta = 1000\ I_{d_\vartheta}), \tag{4.43}$$

$$p_\beta(\beta) = N(\beta|\mu_\beta, \Sigma_\beta = 1000\ I_{d_y}), \tag{4.44}$$

$$p_{\Sigma_e}(\Sigma_e) = IW(\Sigma_e|\,\Psi_{\Sigma_e} = 500\ I_{d_y}, \nu_{\Sigma_e} = 4), \tag{4.45}$$

$$p_{\alpha_0}(\alpha_0) = N(\alpha_0|\mu_{\alpha_0} = 0, \Sigma_{\alpha_0} = 4\ I_{d_\alpha}). \tag{4.46}$$

These selected priors are very flat and hence, the inference result is mainly driven by the likelihood.

A note on multi-modal likelihood

As mentioned in Section 2.4.1, the full VARMA model generally has an identifiability problem, leading to a multi-modal likelihood. Notice that the modified VARMA model applies some constraints to the AR and MA parameters (the sparse form and the indicator matrices) and hence may or may not suffer from the multi-modality issue in the reduced parameter space. So, we will investigate this issue by numerical approach.

For each dataset of Examples 1 and 2, 50 different starting points are randomly selected for φ and ϑ. The remaining parameters are assumed to have true values ($\beta^\star$, $\varSigma_e^\star$ and $\alpha_0^\star$). Then, the MLEs are obtained by maximising the likelihood, conditioning on each starting point.

It turns out that 50 MLEs of Example 1 converge to the same location, showing no sign of multi-modality. Of course, by this test, we cannot disprove the multi-modality likelihood of this dataset but at least the chance of being stuck in local modes (if the likelihood is truly multi-modal) seems to be very small. There should be no issue with using MCMC on this dataset.

However, in the dataset of Example 2, the MLEs converge to different points. Table 4.1 lists some MLEs along with the corresponding log likelihood values l. The Hessian matrices of the log likelihood function, evaluated at these MLEs, are indeed negative-definite matrices, indicating that these MLEs are truly (local) maximum points and not wrong results caused by any numerical issue of the optimisation method.

These MLEs may seem quite different from $(\varphi^\star, \vartheta^\star)$. However, the inverse of the observed information matrix E (the inverse of the Hessian of the negative log likelihood, which is analogous to a covariance matrix) indicates that the true parameter is still within the 95% confidence interval of an MLE. For example, with the MLE $(\varphi_c^\star, \vartheta_c^\star)$, the square root of diagonal entries of E_c (analogous to the standard deviation) is $(\widehat{\sigma}_{\varphi;c}, \widehat{\sigma}_{\vartheta;c})$ = (0.045126, 0.171133, 0.226765, 0.125015, 0.033920, 0.131615, 0.056261, 0.164078, 0.216838, 0.138399, 0.048490, 0.143037). Hence, $(\varphi^\star, \vartheta^\star)$ is still within the interval $[(\varphi_c^\star, \vartheta_c^\star) - 2(\widehat{\sigma}_{\varphi;c}, \widehat{\sigma}_{\vartheta;c}), (\varphi_c^\star, \vartheta_c^\star) + 2(\widehat{\sigma}_{\varphi;c}, \widehat{\sigma}_{\vartheta;c})]$.

Another way to check for multi-modality is plotting the log likelihood along the

Table 4.1: MLEs $(\widehat{\varphi}, \widehat{\vartheta})$ and the log likelihood values of l Example 2. The true value is $\varphi^\star = (0.70, 0.18, -0.15, -0.60, -0.30, 0.20)$ and $\vartheta^\star = (-0.60, -0.15, 0.12, 0.50, 0.20, -0.10)$.

	Running index		
	a	b	c
$\widehat{\varphi}$	0.470630	0.483867	0.703966
	0.101301	0.101038	0.071664
	1.612006	1.528161	−0.323767
	0.006220	−0.004746	−0.547983
	−0.162506	−0.176315	−0.322214
	−0.597383	−0.597475	0.300066
$\widehat{\vartheta}$	−0.394762	−0.407280	−0.607141
	−0.083197	−0.082749	−0.035852
	−1.680363	−1.595364	0.303210
	0.015432	0.027063	0.471322
	0.022600	0.039087	0.219272
	0.600675	0.600419	−0.237067
l	−6461.312	−6461.327	−6435.911

line connecting between two MLEs: $(\widehat{\varphi}_a, \widehat{\vartheta}_a)$ and $(\widehat{\varphi}_c, \widehat{\vartheta}_c)$ as in Figure 4.1. In this case, the likelihood is a bimodal function along the specified line. However, the log difference of the likelihood is quite significant, suggesting that the function peak of smaller likelihood value may be ignorable.

Unfortunately, for a problem with multi-modal likelihood, we are not aware of any efficient MCMC method. The difficult lies in two issues: finding the function modes by exploration and jumping between explored modes, which become more complex in high dimensional space. Hence, in such a case, we use MCMC to explore only the local neighbourhood of the global maximum point (global MLE). The procedure starts by multiple optimisations at different initial points. If all MLEs are similar (like Example 1), MCMC can be used with any starting point. Otherwise, if the MLEs are different (like Example 2), a random point near the global MLE can be chosen as the MCMC starting point.

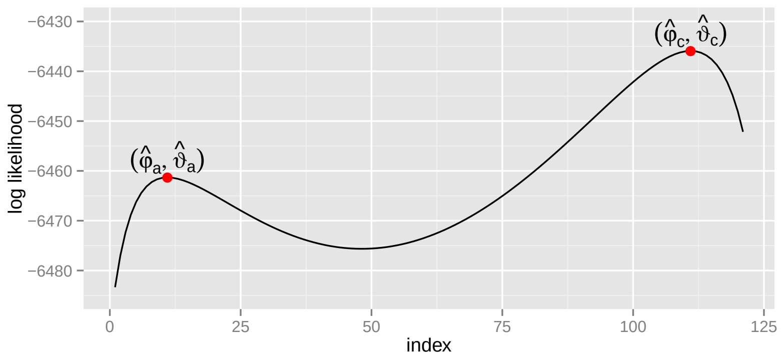
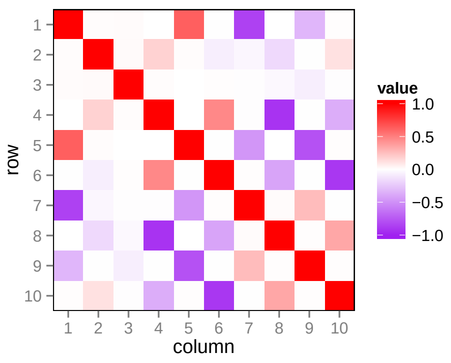


Fig. 4.1: The log likelihood along the line connecting $(\widehat{\varphi}_a, \widehat{\vartheta}_a)$ and $(\widehat{\varphi}_c, \widehat{\vartheta}_c)$ of Example 2.

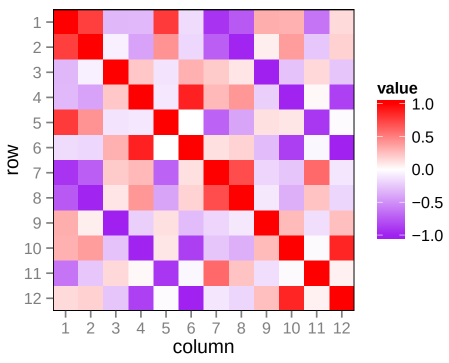
Variable correlation of the DLM-VARMA

As mentioned in Section 2.2.3, one of the most important problems in MCMC is variable correlation. Simple methods such as Gibbs sampling and Metropolis-Hastings with random walk proposals may move in small steps, resulting in a highly dependent MCMC sample series.

In the DLM-VARMA, φ and ϑ are related by Equation (4.12). The "correlation matrix" K of (φ, ϑ) in the likelihood part can be obtained by standardising matrix E above to a correlation matrix. The more the number of non-diagonal entries with absolute values near 1, the stronger the correlation. The correlation matrices K of Examples 1 and 2 are reported in Figure 4.2. As the parameter dimension increases, the correlation seems to be stronger too.



(a) Example 1



(b) Example 2: matrix K is evaluated at $(\widehat{\varphi}_c, \widehat{\vartheta}_c)$

Fig. 4.2: The correlation matrices K of (φ, ϑ).

Applying a linear transform $y = V_x^T x$ to a random vector x with covariance matrix

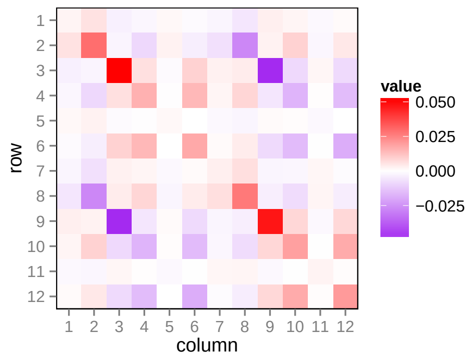
$\Sigma_x = V_x \Lambda_x V_x^T$ (eigen decomposition with orthonormal matrix V_x) gives a variable y with diagonal covariance matrix $\Sigma_y = \Lambda_x$. So, it can be said that the direction of the linear relationship in vector x is encoded in the eigenvectors V_x (or equivalently the covariance matrix Σ_x). Correspondingly, in the DLM-VARMA context, matrix E stores the direction information of the local correlation of (φ, ϑ) at a specific point and a similar linear transformation results in a locally orthogonal random vector.

Interestingly, the evaluation of matrix E at different locations gives different values. In the model of Example 2, matrices E_c, E_d are evaluated at the MLE $(\widehat{\varphi}_c, \widehat{\vartheta}_c)$ and a random location near that MLE $(\varphi_d, \vartheta_d) = (\widehat{\varphi}_c, \widehat{\vartheta}_c) + N(0, \Sigma = 0.02^2 I)$ correspondingly. These matrices and the differenced matrix are shown in Figure 4.3. This difference indicates that the correlation of (φ, ϑ) is not globally similar in the whole parameter space, i.e. a linear transform can only makes the random vector orthogonal locally, but not globally, in the parameter space. This point is important in designing a good MCMC method for the DLM-VARMA.

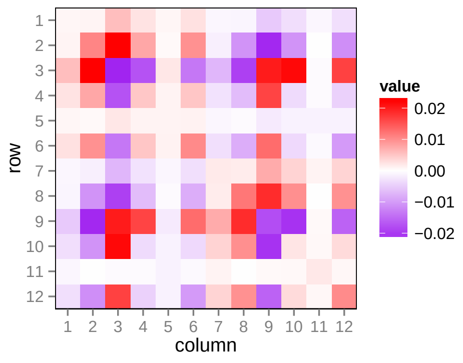
MCMC simulations of the first algorithm

So, we run the MCMC of Algorithm 10 for Examples 1 and 2 with 200000 samples each. As there are many parameters, only the whole trace plots of ϑ_1 are shown in Figure 4.4. As there is no sign of unwanted trends, the MCMC convergence should be safe (trace plots of other parameters have no trend either). To check the MCMC mixing, we enlarge the trace plots of ϑ_1 with 1000 samples in Figure 4.5. Unfortunately, the sample movements are local and small, especially in Figure 4.5(b), indicating the variable correlation problem.

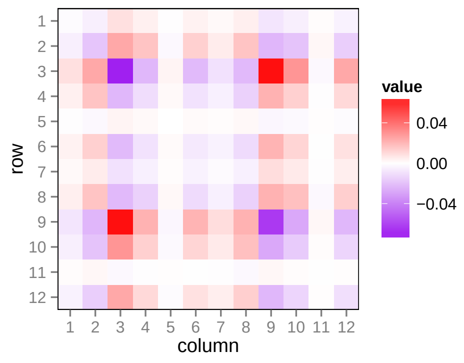
Another way of checking the performance of a MCMC algorithm is based on the autocorrelation of its samples. After thinning with interval 10, the autocorrelation plots of ϑ are shown in Figures 4.6 and 4.7. While the autocorrelation of the first model seems normal, the autocorrelation of Example 2 only decays to zero completely after about a lag of 80. Taking into account the thinning interval, this means that only one independent sample is obtained after 800 iterations, implying that the MCMC of Algorithm 10 suffers a great deal from the variable correlation issue. This problem becomes worse as the number of AR, MA parameters increases. Hence, a better MCMC algorithm is proposed in the next section for the DLM-VARMA.



(a) E_c

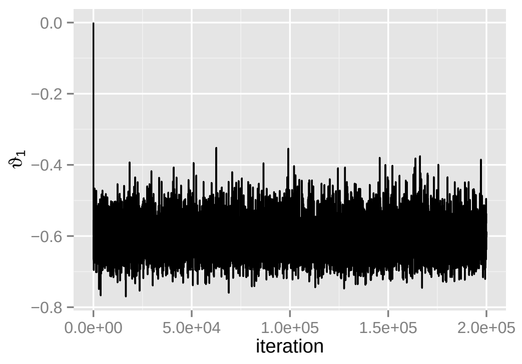


(b) E_d

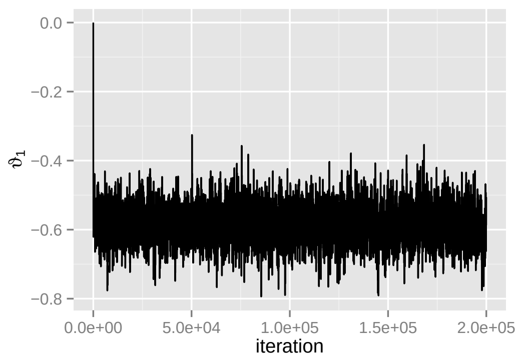


(c) $E_d - E_c$

Fig. 4.3: Matrices E evaluated at $(\widehat{\varphi}_c, \widehat{\vartheta}_c)$ and a random location $(\varphi_d, \vartheta_d) = (\widehat{\varphi}_c, \widehat{\vartheta}_c) + N(0, \varSigma = 0.02^2 I)$ for Example 2.

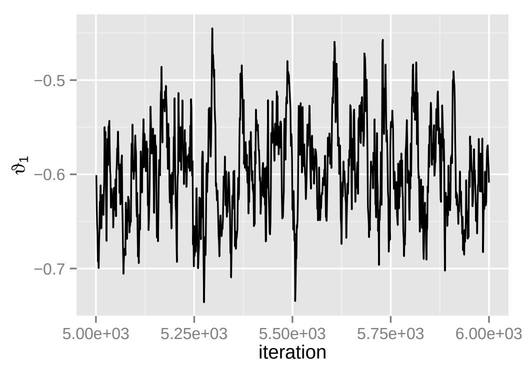


(a) Example 1

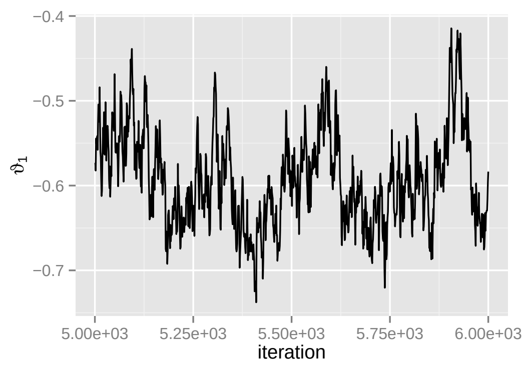


(b) Example 2

Fig. 4.4: Algorithm 10: the trace plots of all samples of ϑ_1.

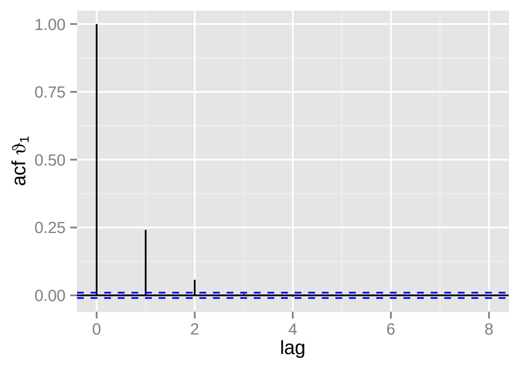


(a) Example 1

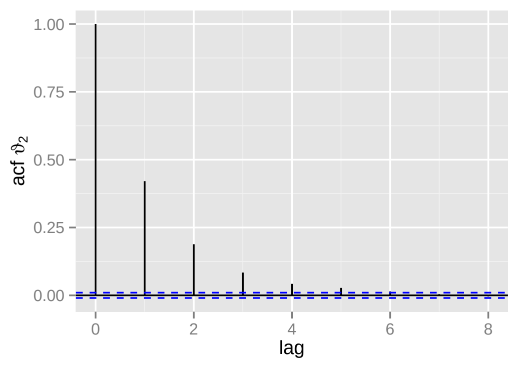


(b) Example 2

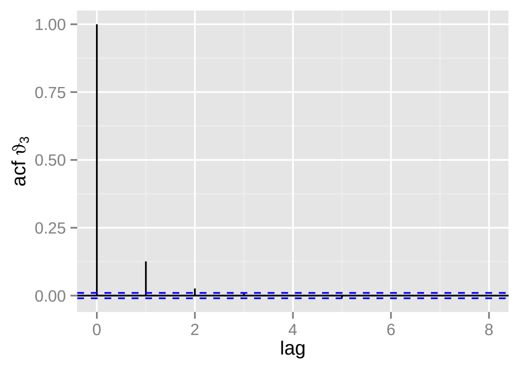
Fig. 4.5: Algorithm 10: the trace plots of 1000 samples of ϑ_1.



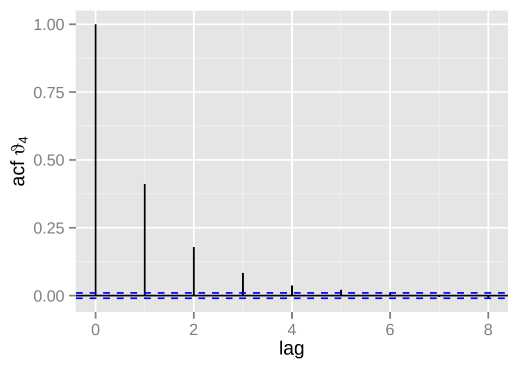
(a) ϑ_1



(b) ϑ_2

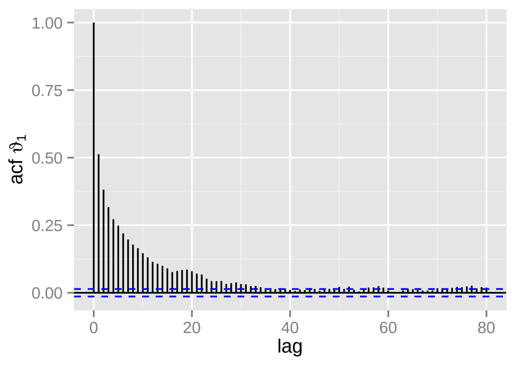


(c) ϑ_3

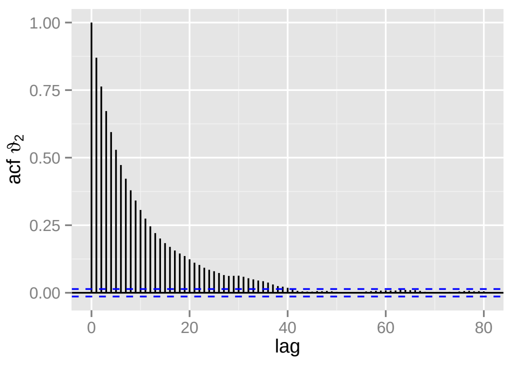


(d) ϑ_4

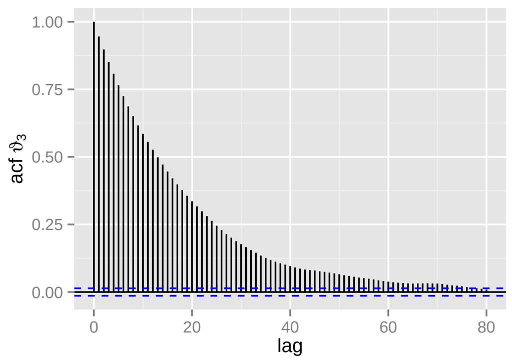
Fig. 4.6: Algorithm 10: the autocorrelation of ϑ for Example 1 after thinning by a factor of 10.



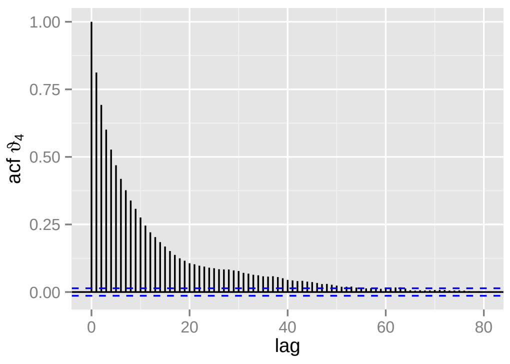
(a) ϑ_1



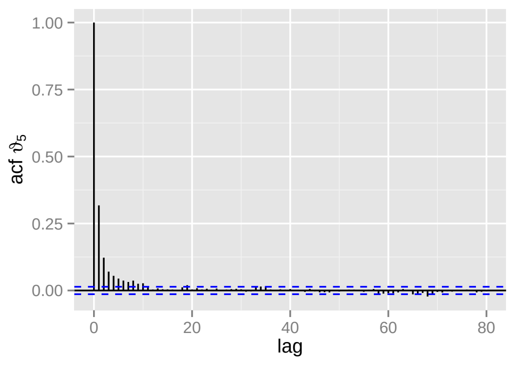
(b) ϑ_2



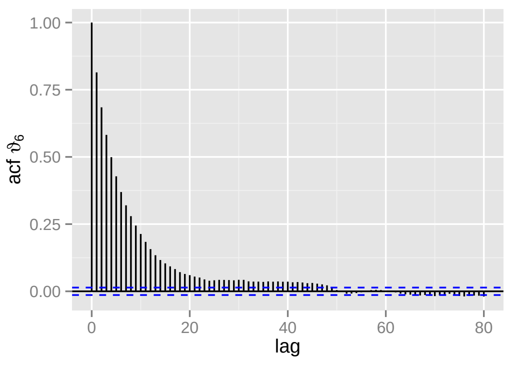
(c) ϑ_3



(d) ϑ_4



(e) ϑ_5



(f) ϑ_6

Fig. 4.7: Algorithm 10: the autocorrelation of ϑ for Example 2 after thinning by a factor of 10.

4.1.3 An improved MCMC algorithm

The marginal posterior of ϑ

First, the correlation between φ and ϑ needs removing in the sampling procedure. Instead of doing Gibbs sampling on (φ, ϑ) with $p_{\varphi|\vartheta,\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$ and $p_{\vartheta|\varphi,\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$, we sample ϑ from the marginal posterior $p_{\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$ and then φ from the usual full conditional posterior $p_{\varphi|\vartheta,\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$. By doing so, the sample of ϑ is not constrained by the current value of φ and the correlation between φ and ϑ is completely removed. Hence, the marginal density $p_{\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$ is derived as follows.

From Equations (4.13), (4.22) and (4.23), the joint posterior density $p_{\varphi,\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$ is:

$$p_{\varphi,\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\varphi,\vartheta) \propto p_\vartheta(\vartheta)p_\varphi(\varphi)p_{e_{1:n}|\varphi,\vartheta,\beta,\Sigma_e,\alpha_0}(e_{1:n}) \tag{4.47}$$

$$\propto \exp\left(-\frac{1}{2}\left[(\vartheta-\mu_\vartheta)^T Q_\vartheta(\vartheta-\mu_\vartheta) + (\varphi-\mu_\varphi)^T Q_\varphi(\varphi-\mu_\varphi) + \sum_{t=1}^{n}(a_{3;t} - B_{3;t}\varphi)^T Q_e (a_{3;t} - B_{3;t}\varphi)\right]\right)$$

$$\propto \exp\left(-\frac{1}{2}\left[(\vartheta-\mu_\vartheta)^T Q_\vartheta(\vartheta-\mu_\vartheta) + \varphi^T Q'_\varphi \varphi - 2\varphi^T Q'_\varphi \mu'_\varphi + \sum_{t=1}^{n} a_{3;t}^T Q_e a_{3;t}\right]\right), \tag{4.48}$$

and $p_{\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$ is obtained by marginalising Equation (4.48) with respect to φ:

$$p_{\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\vartheta) = \int_\varphi p_{\varphi,\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\varphi,\vartheta)d\varphi \tag{4.49}$$

$$\propto |Q'_\varphi|^{-1/2} \exp\left(-\frac{1}{2}\left[(\vartheta-\mu_\vartheta)^T Q_\vartheta(\vartheta-\mu_\vartheta) - {\mu'_\varphi}^T Q'_\varphi \mu'_\varphi + \sum_{t=1}^{n} a_{3;t}^T Q_e a_{3;t}\right]\right). \tag{4.50}$$

Compared to the sparse additive form of VARMA in Equation (4.1) which has only two AR, MA parameter blocks, the multiplicative form, e.g. SARIMA in Equation (3.2), has four parameter blocks: φ, φ_s, ϑ and ϑ_s. Conditioning on the rest, the posterior of $p_{\varphi|\varphi_s,\vartheta,\vartheta_s,y_{1:n},\ldots}$ can be marginalised analytically but the resulting marginal posterior $p_{\varphi_s,\vartheta,\vartheta_s|y_{1:n},\ldots}$ is intractable with three correlated parameter blocks. As we would like to reduce the number of correlated parameters, the sparse additive form of VARMA is preferred.

The second MCMC algorithm

Like the first algorithm, Metropolis-Hastings is used on the intractable marginal posterior $p_{\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$. Such a sampling is free from the constraint of φ but still suffers the variable correlation issue between the entries of the random vector φ. So, based on the analysis of matrix E in the previous section, the correlation direction the marginal posterior of ϑ should be taken into account in choosing the proposed sample of Metropolis-Hastings. Also, the correlation directions at different locations are dissimilar, suggesting the usage of local direction at a point instead of global direction.

For a specific value ϑ_a, denote a matrix function:

$$L(\vartheta_a) = \left.\frac{-\partial^2 \log p_{\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\vartheta)}{\partial\vartheta^2}\right|_{\vartheta=\vartheta_a}. \tag{4.51}$$

A proposal $N(\cdot|\mu = \vartheta_a, \Sigma = \kappa L(\vartheta_a)^{-1})$ with scale parameter κ will generate a sample ϑ_b along the correlation direction of $p_{\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\vartheta_a)$. Both the magnitude and the direction of the MCMC sample is adapted to the curvature of the current location.

In summary, the improved MCMC implementation is shown in Algorithm 11. The sub-step 4-2 is done repeatedly to save the computation cost of matrix $L(\cdot)$. Usually, we choose $\kappa = 0.5$ and $m_\vartheta = 10$.

Algorithm 11 The second MCMC implementation on the DLM-VARMA

```
1. Sample $\beta$ from $p_{\beta|\varphi,\vartheta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot) = N(\cdot|\mu'_\beta, \Sigma'_\beta = Q'^{-1}_\beta)$.

2. Sample $\alpha_0$ from $p_{\alpha_0|\varphi,\vartheta,\beta,\Sigma_e,y_{1:n}}(\cdot) = N(\cdot|\mu'_{\alpha_0}, \Sigma'_{\alpha_0} = Q'^{-1}_{\alpha_0})$.

3. Sample $\Sigma_e$ from $p_{\Sigma_e|\varphi,\vartheta,\beta,\alpha_0,y_{1:n}}(\cdot) = IW(\cdot|\Psi'_{\Sigma_e}, \nu'_{\Sigma_e})$.

4. Sample $\vartheta$ by the marginal posterior $p_{\vartheta|\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot)$

    4-1 Calculate matrix $L(\vartheta_a)$ at the current location $\vartheta_a$.

    4-2 Metropolis Hastings: Propose a new value $\vartheta_c$ by a random walk centred on
        the current value $\vartheta_b$: $\vartheta_c \sim N(\cdot|\vartheta_b, \kappa L(\vartheta_a)^{-1})$. Accept the new value $\vartheta_c$ with
        the Metropolis-Hastings rate. This step is done $m_\vartheta$ times with the same
        matrix $L(\vartheta_a)$ (in the first turn $\vartheta_b = \vartheta_a$).

5. Sample $\varphi$ from $p_{\varphi|\vartheta,\beta,\Sigma_e,\alpha_0,y_{1:n}}(\cdot) = N(\cdot|\mu'_\varphi, \Sigma'_\varphi = Q'^{-1}_\varphi)$.
```

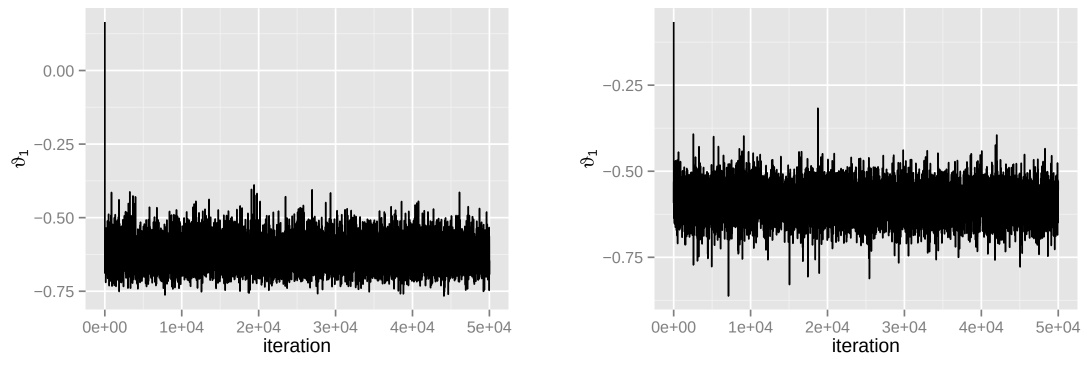
Again, like the first MCMC algorithm, α_0 is sampled for a number of iterations m_{α_0}

(1000 iterations) and then fixed this to reduce the computation cost.

A note on missing data

In this thesis, we only work on a complete data $y_{1:n}$ but MCMC of Bayesian approach suggests a natural solution to such problem. In the case of missing data, divide the whole data $y_{1:n}$ into two parts: the observed part $y_{o;1:n}$ and the missing part $y_{m;1:n}$. Then, Bayesian inference can be applied on the posterior $p_{\varphi,\vartheta,\beta,\Sigma_e,\alpha_0,y_{m;1:n}|y_{o;1:n}}(\cdot)$ with Gibbs sampling on $p_{\varphi,\vartheta,\beta,\Sigma_e,\alpha_0|y_{o;1:n},y_{m;1:n}}(\cdot) = p_{\varphi,\vartheta,\beta,\Sigma_e,\alpha_0|y_{1:n}}(\cdot)$ and $p_{y_{m;1:n}|\varphi,\vartheta,\beta,\Sigma_e,\alpha_0,y_{o;1:n}}(\cdot)$. As the sampling of $p_{\varphi,\vartheta,\beta,\Sigma_e,\alpha_0|y_{o;1:n},y_{m;1:n}}(\cdot)$ has been already implemented by the proposed second MCMC algorithm, we only need to deal with the conditional sampling of the missing part by another Metropolis Hastings algorithm. This interesting extension may be considered in future work.

Discussion on this multiple imputation problem can be found in Gelman et al. (2003,chap. 21) and Gelman and Hill (2007,chap. 25). A detailed study on missing data is referred to Little and Rubin (2002).

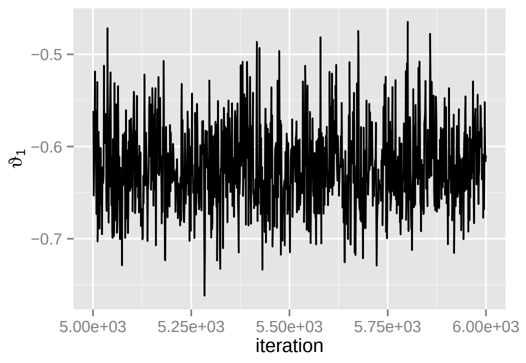


(a) Example 1 (b) Example 2

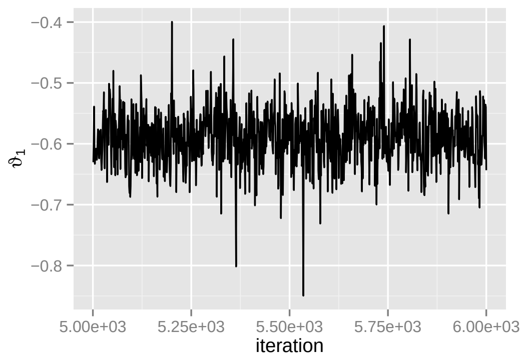
Fig. 4.8: Algorithm 11: the trace plots of all samples of ϑ_1.

MCMC simulations of the second algorithm

For Examples 1 and 2, we run MCMC of Algorithm 11 with 50000 samples for each example. The whole trace plots and the enlarged ones of ϑ_1 are in Figures 4.8 and 4.9. There are no trends in the traces and the mixing seems fine with samples fluctuating strongly.



(a) Example 1

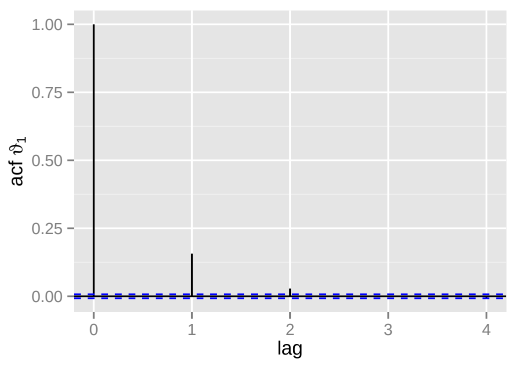


(b) Example 2

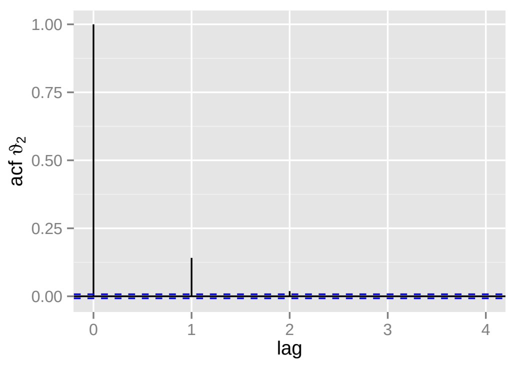
Fig. 4.9: Algorithm 11: the trace plots of 1000 samples of ϑ_1.

The autocorrelation without thinning are plotted in Figures 4.10 and 4.11 which can be compared to Figures 4.6 and 4.7 (the first MCMC algorithm uses thinning with interval 10 while the second algorithm runs the sub-step 4-2 $m_\vartheta = 10$ times for each main MCMC iteration). The autocorrelation of the new algorithm decays more quickly, compared to the first algorithm, especially for Example 2. So, in a DLM-VARMA model with many correlated parameters, the performance of Algorithm 11 is better.

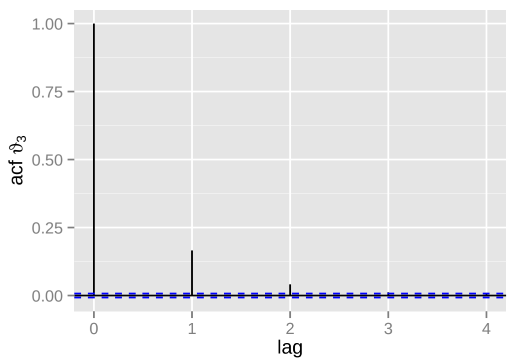
After removing the first 100 samples, the box plots of MCMC for two examples are shown in Figures 4.12 and 4.13. As the parameter priors are non-informative (flat), the inference is mainly affected by the likelihood part of the data. Some true values lie outside of the $25\% - 75\%$ credible intervals but are still in whisker-intervals.



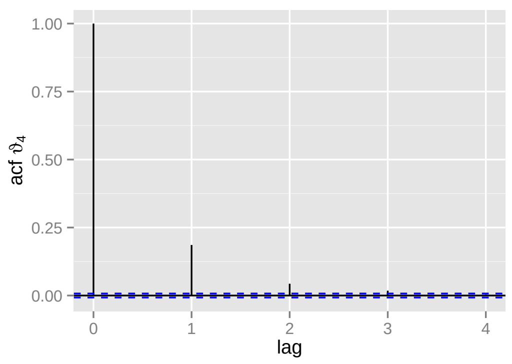
(a) ϑ_1



(b) ϑ_2

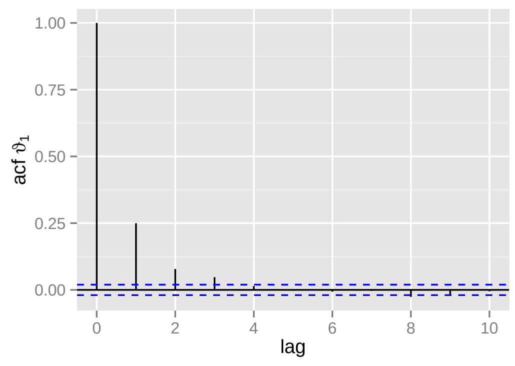


(c) ϑ_3

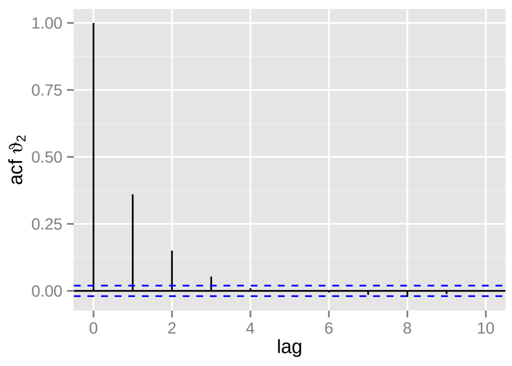


(d) ϑ_4

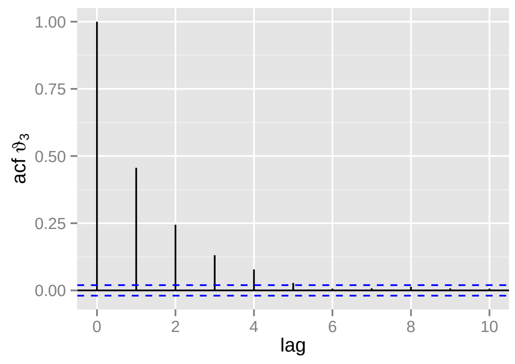
Fig. 4.10: Algorithm 11: the autocorrelation of ϑ for Example 1.



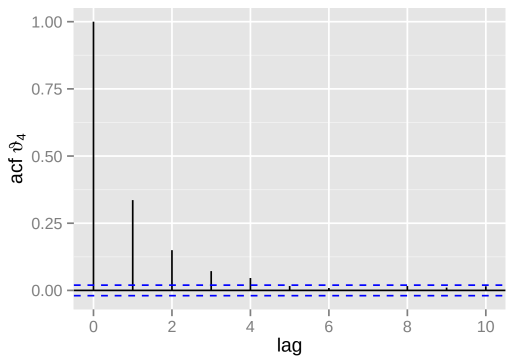
(a) ϑ_1



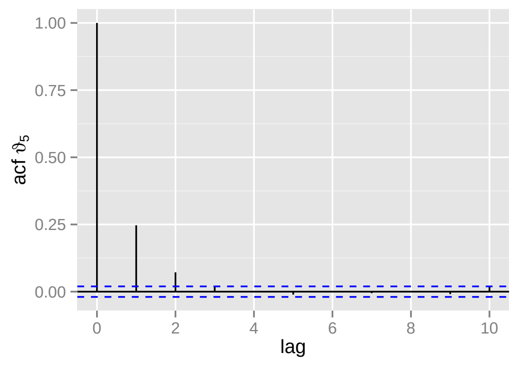
(b) ϑ_2



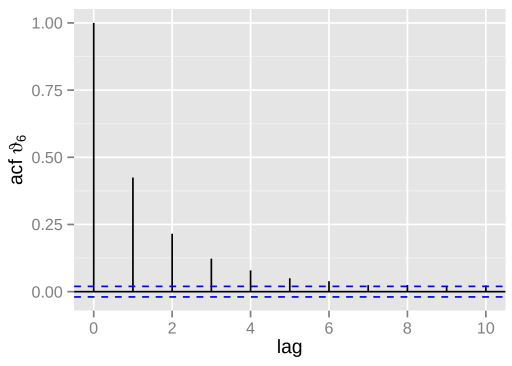
(c) ϑ_3



(d) ϑ_4

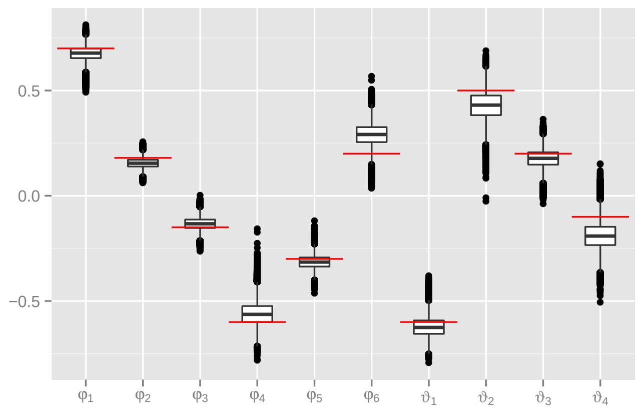


(e) ϑ_5

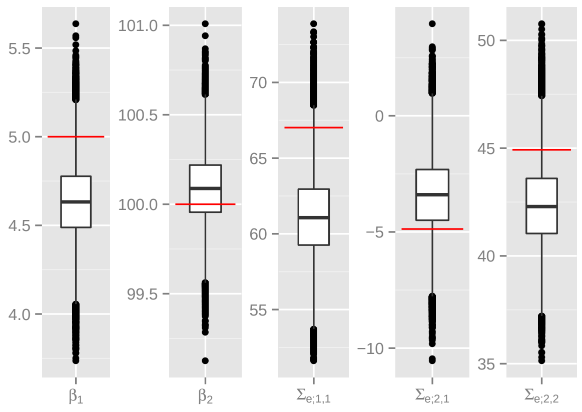


(f) ϑ_6

Fig. 4.11: Algorithm 11: the autocorrelation of ϑ for Example 2.

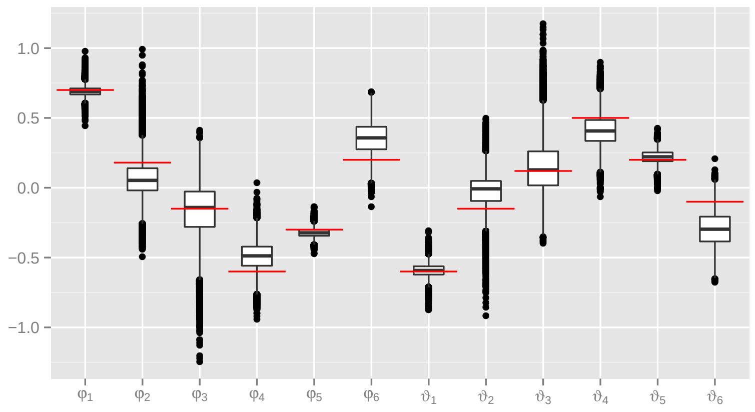


(a) φ and ϑ

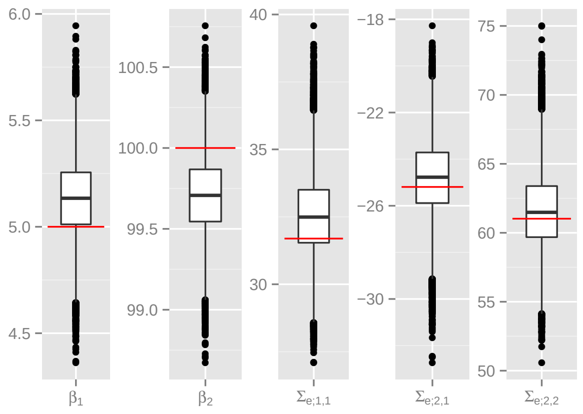


(b) β and Σ_e

Fig. 4.12: Algorithm 11: the MCMC sample box plots for Example 1. The red segments indicate the true parameter values.



(a) φ and ϑ



(b) β and Σ_e

Fig. 4.13: Algorithm 11: the MCMC sample box plots for Example 2. The red segments indicate the true parameter values.

Table 4.2: Algorithm 11: the quantile estimators and their corresponding half-widths of 95% intervals for Example 1.

	Quantiles		
	10%	50%	90%
φ	0.630103 (0.000534)	0.677792 (0.000324)	0.717531 (0.000421)
	0.124915 (0.000282)	0.154725 (0.000213)	0.185344 (0.000321)
	−0.171120 (0.000360)	−0.133211 (0.000272)	−0.094925 (0.000350)
	−0.631257 (0.000672)	−0.563578 (0.000579)	−0.485021 (0.000856)
	−0.354932 (0.000388)	−0.315185 (0.000318)	−0.272173 (0.000455)
	0.220383 (0.000834)	0.291072 (0.000538)	0.357303 (0.000775)
ϑ	−0.681789 (0.000568)	−0.625665 (0.000472)	−0.559207 (0.000738)
	0.337510 (0.001016)	0.430663 (0.000693)	0.514219 (0.000861)
	0.119725 (0.000597)	0.177736 (0.000403)	0.230906 (0.000523)
	−0.272776 (0.000838)	−0.191452 (0.000673)	−0.105400 (0.000988)
β	4.357437 (0.002721)	4.632227 (0.002028)	4.906308 (0.002707)
	99.838955 (0.002345)	100.088218 (0.001812)	100.339123 (0.002501)
Σ_e	57.698347 (0.030977)	61.067509 (0.025839)	64.695709 (0.036757)
	−5.485462 (0.021580)	−3.397460 (0.014548)	−1.346688 (0.020372)
	39.950243 (0.023548)	42.289396 (0.018735)	44.798508 (0.024962)

To justify that the MCMC has run long enough and achieved a reasonable precision, Tables 4.2 and 4.3 show the quantile estimators of all parameters, along with their half-width calculated from MCSE as in Section 2.2.3. Also, from these tables, there seems to be no convergence issue.

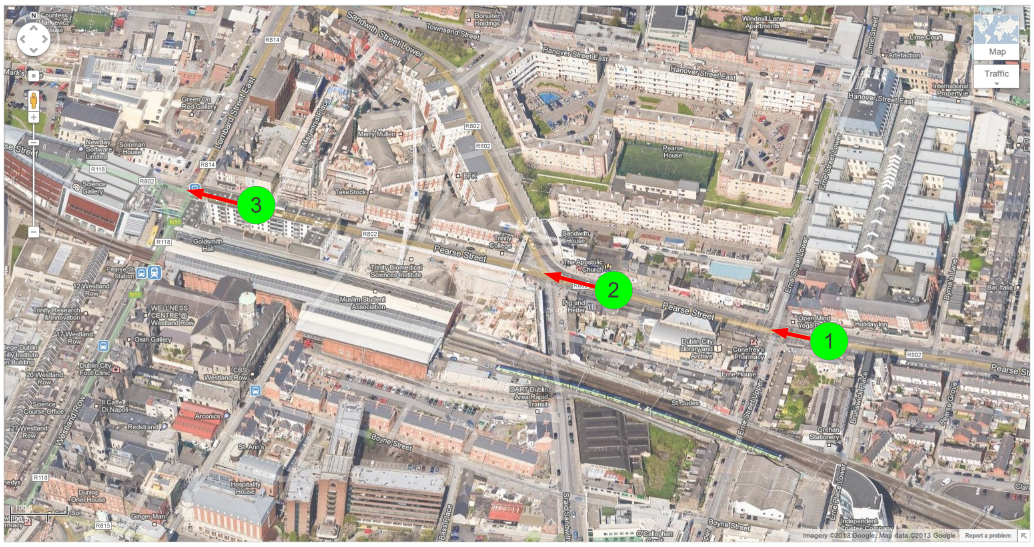
So, we have discussed Bayesian inference on the VARMA model and proposed an improved MCMC algorithm to tackle the variable correlation issue of the VARMA. The application of the VARMA model in the STFF problem is discussed in the next section.

Table 4.3: Algorithm 11: the quantile estimators and their corresponding half-widths of 95% intervals for Example 2.

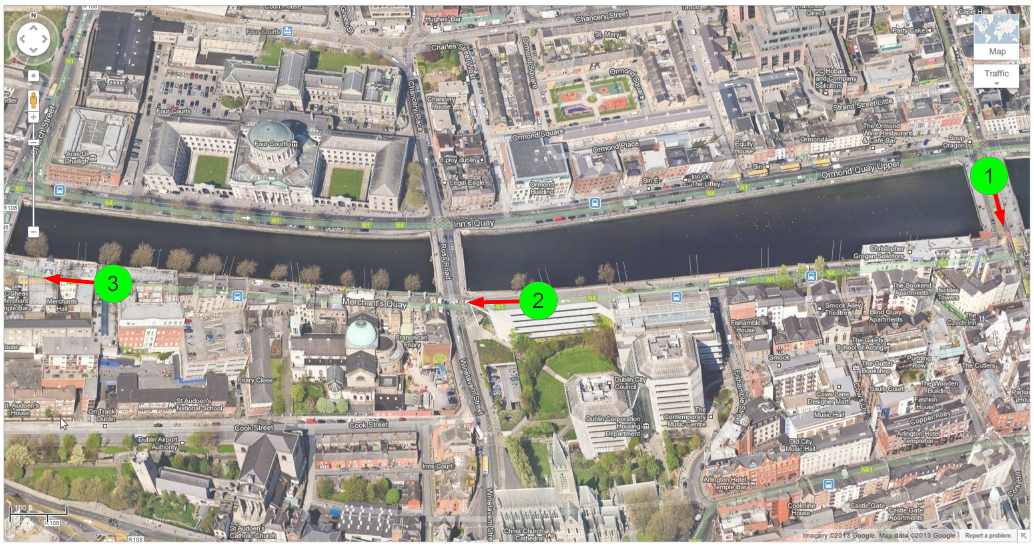
	Quantiles		
	10%	50%	90%
φ	0.648956 (0.000532)	0.689529 (0.000411)	0.733160 (0.000656)
	−0.099740 (0.002737)	0.052663 (0.001590)	0.247851 (0.003657)
	−0.415159 (0.004435)	−0.141608 (0.003696)	0.031363 (0.001773)
	−0.624530 (0.002295)	−0.487918 (0.001507)	−0.367755 (0.001531)
	−0.361362 (0.000485)	−0.323131 (0.000397)	−0.280684 (0.000604)
	0.202173 (0.003151)	0.357235 (0.001804)	0.498704 (0.001808)
ϑ	−0.648997 (0.000811)	−0.592445 (0.000572)	−0.534600 (0.000827)
	−0.200569 (0.003414)	−0.007711 (0.001262)	0.131168 (0.002816)
	−0.033622 (0.001722)	0.127923 (0.003498)	0.390458 (0.004334)
	0.278285 (0.001610)	0.406751 (0.001622)	0.560909 (0.002703)
	0.160441 (0.000833)	0.221540 (0.000575)	0.279688 (0.000796)
	−0.454912 (0.002049)	−0.296946 (0.002047)	−0.128110 (0.003336)
β	4.899950 (0.002991)	5.134024 (0.002046)	5.365594 (0.002757)
	99.397477 (0.003961)	99.707065 (0.002482)	100.012686 (0.003477)
Σ_e	30.698244 (0.020413)	32.490382 (0.016370)	34.434052 (0.025448)
	−26.925412 (0.026959)	−24.775442 (0.017094)	−22.759542 (0.022971)
	58.103896 (0.036598)	61.485124 (0.032431)	65.163310 (0.043969)

4.2 STFF with the VARMA model

This section first presents the two real traffic datasets used in the STFF problem. Then, for each of the datasets, three different preprocessing procedures are used to extract residuals which are then fitted to VARMA models. Next, the multi-step-ahead predictions are compared for all models. Finally, the section is concluded with several remarks about the VARMA model and STFF.



(a) Example 3: Pearse Street



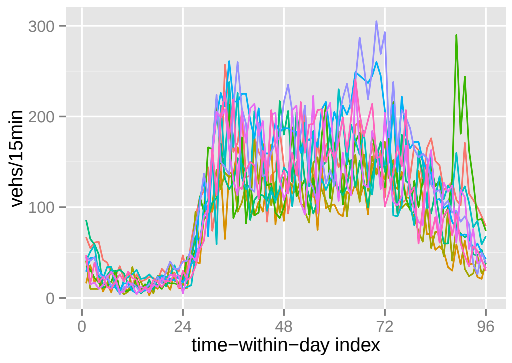
(b) Example 4: South Quays of River Liffey

Fig. 4.14: Network maps for two traffic datasets in Dublin.

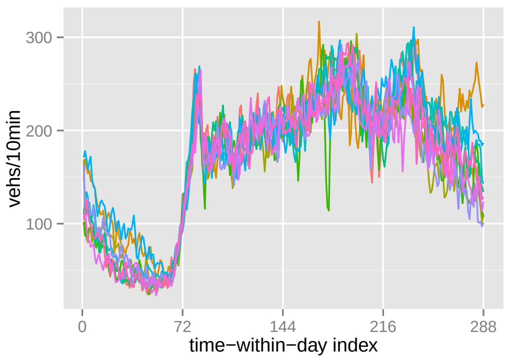
4.2.1 Traffic datasets in Dublin

The inference of this section is evaluated by modelling traffic volume observations from a busy thoroughfare in the city centre of Dublin in Ireland. Two datasets corresponding to two chosen sections of Dublin traffic map were considered as in Figure 4.14. In each dataset, three locations had been chosen for continuous traffic data collection over 24 hours daily. As seen in the figure, the selected sites are situated on/near Pearse Street and the South Quays of the River Liffey, which are some of the busiest roads in Dublin City.

The data, obtained collectively from all the inductive loop detectors on any approach, is used for the modelling. These loop detectors were embedded near the stop line of each lane of the modelled intersections and vehicles were counted by analysing the changes in electro-magnetic profiles of vehicles passing over the loops. Both datasets are extracted by the software SCATS Traffic Reporter, then are filtered and converted to a standard row-column format by a Perl script. As the weekend traffic dynamics are very different from traffic dynamics of the weekdays, the modelling is carried out on the data observed during weekdays only.



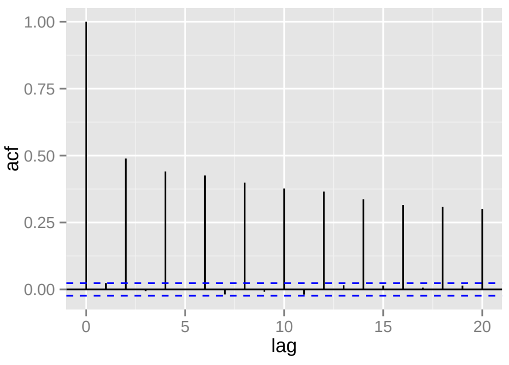
(a) $z_{t,3}$ of Example 3



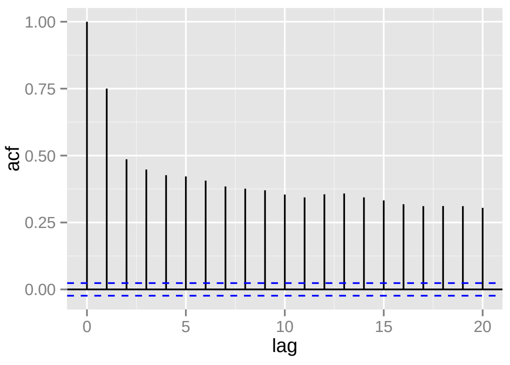
(b) $z_{t,3}$ of Example 4

Fig. 4.15: Day-by-day traffic flow plots of z_t for weekdays.

Example 3 Dataset 3 comprises of traffic volumes collected at three locations along the roadway of Pearse Street: the westbound (station 1) approach of the intersection of Pearse and Lower Erne Streets, the westbound (station 2) approach of the intersection at Pearse and Lower Sandwith Streets and the westbound (station 3) approach of



(a) The differenced 5-min traffic flow



(b) The differenced aggregated 10-min traffic flow

Fig. 4.16: Example 4: the autocorrelation of the differenced 5-min traffic flow and the differenced aggregated 10-min traffic flow.

the intersection at Pearse and Lombard Streets. All intersections are four-way cross-sections, with two-way traffic on all road links. All the junctions have three-phase signals with turning protection on Pearse Street. The number of lanes for the traffic volume recording of three stations 1, 2 and 3 are 1, 3 and 2 respectively.

This dataset was recorded from 22nd July 2010 to 26th August 2010. 15-minute aggregate traffic volume observations were used in modelling purposes (the daily period for this dataset is $s = 96$). A total of 26 days of data from weekdays were used. The traffic observations from first 20 days were used for fitting and the rest of data was used for model evaluation (this original flow data is denoted by $z_t = z_{t,1:3}$ for this dataset). The day-by-day traffic flow of station 3 for this dataset, $z_{t,3}$, is in Figure 4.15(a).

Example 4 Dataset 4 consists of 25 weekdays recorded from 11st June 2008 to 7th July 2008 in the South Quays of River Liffey. Again, three sites are chosen with longer links: the southbound approach (station 1 - 3 lanes) of the Grattan Bridge, the westbound approach (station 2 - 2 lanes) of the intersection of Wood Quay and Wine Tavern Street and the westbound approach (station 3 - 3 lanes) of the intersection of Merchants Quay and Church Street.

Originally, this dataset contains 5-minute aggregate traffic volume observations. However, due to the effect of stop-and-go sequences of the traffic light, this original flow series exhibit an on-off state dependency of the flow observations. For example, this effect is shown in the autocorrelation plot of the differenced 5-min traffic flow of

Figure 4.16(a).

So, to eliminate the traffic light effect, 2 consecutive data points are aggregated. In each day, there are 288 data points (daily period $s = 288$) each of which is a 10-minute aggregate traffic flow (the autocorrelation of the differenced 10-min flow is in Figure 4.16(b)). Denote the aggregate series by $z_t = z_{t,1:3}$ ($z_{t,3}$ is plotted in Figure 4.15(b)). Observations from the first 20 day were used for fitting and the rest of data was used for model evaluation.

4.2.2 Data preprocessing

As the VARMA model is not for data with a time-dependent mean such as series in Figure 4.15, we will present some preprocessing procedures to extract residuals y_t from the flows z_t in this part.

One of the most popular preprocessing method in time series is the seasonal difference operator. In this case, we use the s-lagged difference operator on z_t with s as the daily period and denote this preprocessing by SD:

$$y_{SD;t} = \bigtriangledown_s z_t = z_t - z_{t-s}. \tag{4.52}$$

Also, a further difference operator is tried on the flow (this preprocessing procedure is named DSD):

$$y_{DSD;t} = \bigtriangledown \bigtriangledown_s z_t = (z_t - z_{t-s}) - (z_{t-1} - z_{t-s-1}). \tag{4.53}$$

If $y_{SD;t}$ and $y_{DSD;t}$ follow VARMA models then the original flows z_t belong to the *seasonal vector autoregressive integrated moving average (SVARIMA)* class, which is commonly used for non-stationary series.

For the third preprocessing MP, we first get the time-varying daily mean m_t:

$$m_t = \begin{cases} \frac{1}{n_{t_1}} \sum_{t_1:t_1=t+sk}^{t_1 \le n_z} z_{t_1} & \text{if } t \le s \\ m_{t-s} & \text{if } t > s \end{cases} \tag{4.54}$$

where n_z is the number of data points z_t. Then $y_{MP;t}$ is obtained by subtraction:

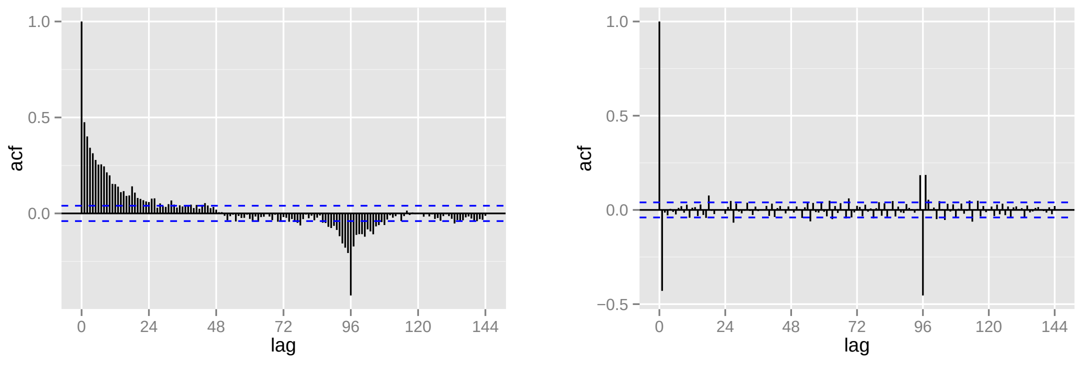
$$y_{MP;t} = z_t - m_t. \tag{4.55}$$

The autocorrelation plot of the residuals $y_{\cdot;t,3}$ (station 3) for both Examples 3 and 4 are plotted in Figures 4.17 and 4.18 (the autocorrelation of other stations are quite similar). There are some remarks about these plots:

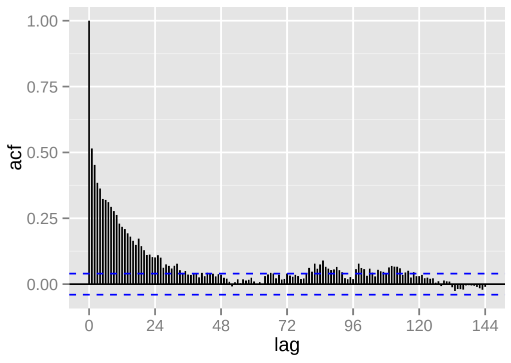
- For $y_{SD;t}$ and $y_{MP;t}$, the autocorrelation seems to decay by the exponential rate, suggesting the use of AR terms Φ_j in the VARMA models.

- For $y_{SD;t}$ and $y_{DSD;t}$, the autocorrelation at lag s (daily period) is quite significant. So, AR, MA terms at lag s should be included.

- The autocorrelation plots of $y_{DSD;t}$ contain several spikes. The two most significant spikes are located at lag 1 and lag s. This shape is usually achieved by adding MA terms at the corresponding lags.



(a) $y_{SD;t,3}$ (b) $y_{DSD;t,3}$

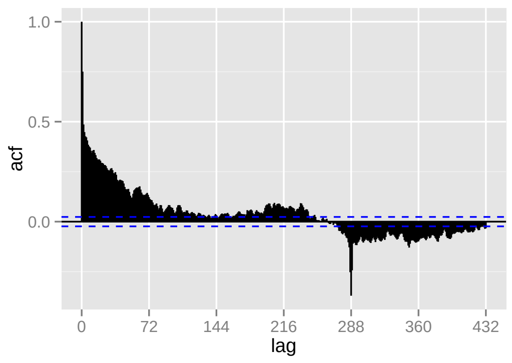


(c) $y_{MP;t,3}$

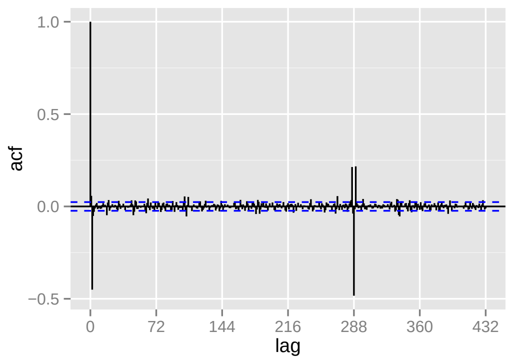
Fig. 4.17: Example 3: the autocorrelation of the residuals $y_{\cdot;t,3}$.

4.2.3 Model specification

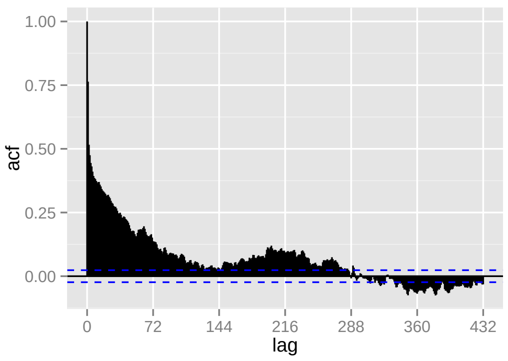
For both datasets, a VARMA model will be applied to each residual series $y_{\cdot;t}$. We will try to use the same specification (as similar as possible) for these VARMA models for



(a) $y_{SD;t,3}$



(b) $y_{DSD;t,3}$



(c) $y_{MP;t,3}$

Fig. 4.18: Example 4: the autocorrelation of the residuals $y_{\cdot;t,3}$.

the comparison purpose. By the analysis of autocorrelation, we decide to use the AR, MA terms at lags 1 and s for all VARMA models:

$$\mathscr{P} = \mathscr{Q} = (1, s). \tag{4.56}$$

Some terms may not be needed but are still included in models to be safe (this is like over-specifying a model for a specific dataset).

The information about the traffic networks in Figure 4.14 is used in specifying the non-zero entries of AR matrices Φ_j. As seen in the figure, the traffic starts from station 1 and reaches station 2 and then 3 respectively. Hence, logically, $y_{\cdot;t,2}$ should be dependent on $y_{\cdot;t-j,1}$ (j-lagged residual at station 1) and $y_{\cdot;t,3}$ is dependent on $y_{\cdot;t-j,1:2}$.

So, the following indicator matrices are used for AR terms:

$$M_{\Phi;1} = M_{\Phi;s} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}. \tag{4.57}$$

For MA terms, simple diagonal indicator matrices are used:

$$M_{\Theta;1} = M_{\Theta;s} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.58}$$

In total, there are 12 scalar parameters for AR terms ($\dim(\varphi) = 12$) and 6 scalar parameters for MA terms ($\dim(\vartheta) = 6$).

We also try univariate ARMA models on the series at station 3 only ($z_{t,3}$). For these univariate models, $\mathscr{P} = \mathscr{Q} = (1, s)$ and $M_{\Phi;1} = M_{\Phi;s} = 1$.

For notation purposes, a VARMA (ARMA) model used on a specific dataset is denoted by the code AAA-B(-CCC) by the following standard:

- AAA indicates the preprocessing procedure with values SD, DSD, MP.
- For the code B, value M is for a multivariate VARMA model used with traffic flows of all stations $z_{t,1:3}$. Value U is used for univariate ARMA on $z_{t,3}$ only.
- The code CCC is optional, indicating the target dataset, e.g. EX03, EX04.

For all models, flat priors are chosen with parameters:

$$p_\varphi(\varphi) = N(\varphi | \mu_\varphi = 0, \Sigma_\varphi = 1000\ I_{d_\varphi}), \tag{4.59}$$

$$p_\vartheta(\vartheta) = N(\vartheta | \mu_\vartheta = 0, \Sigma_\vartheta = 1000\ I_{d_\vartheta}), \tag{4.60}$$

$$p_\beta(\beta) = N(\beta | \mu_\beta, \Sigma_\beta = 1000\ I_{d_y}), \tag{4.61}$$

$$p_{\Sigma_e}(\Sigma_e) = IW(\Sigma_e | \Psi_{\Sigma_e} = 500\ I_{d_y}, \nu_{\Sigma_e}), \tag{4.62}$$

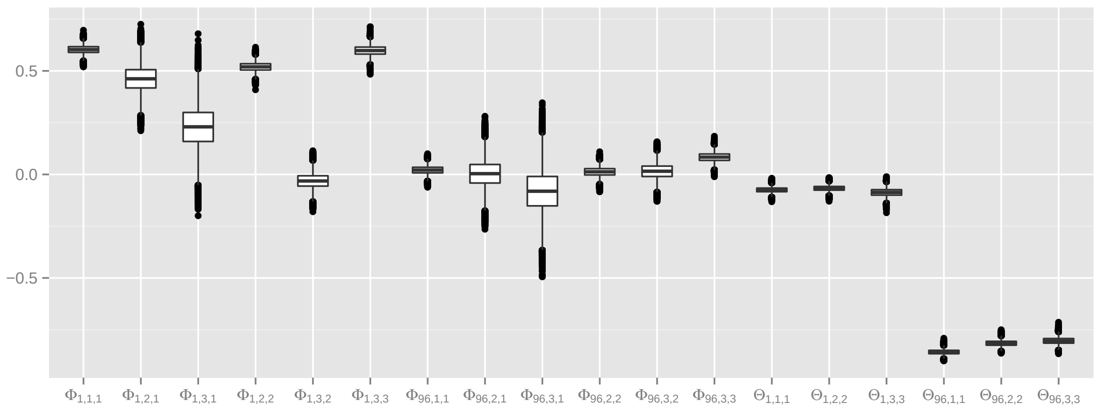
$$p_{\alpha_0}(\alpha_0) = N(\alpha_0 | \mu_{\alpha_0} = 0, \Sigma_{\alpha_0} = 4\ I_{d_\alpha}). \tag{4.63}$$

where $\nu_{\Sigma_e} = 3$ for univariate models and $\nu_{\Sigma_e} = 5$ for multivariate models (this results in: $E_{\Sigma_e}(\Sigma_e) = \Psi_{\Sigma_e}$).

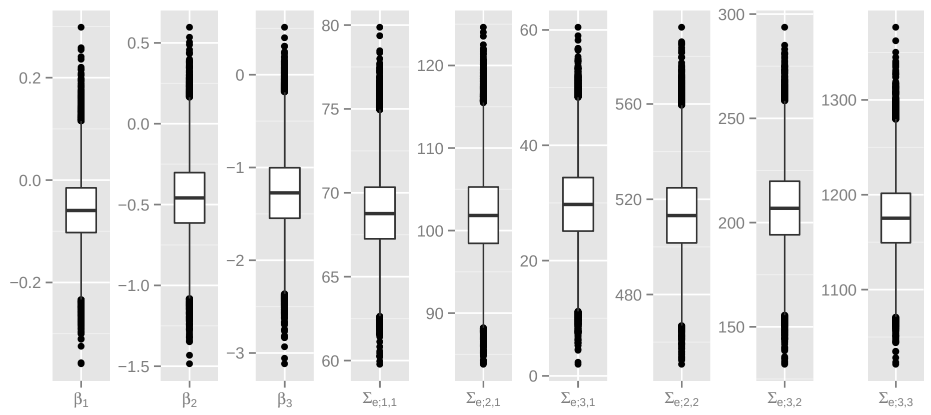
4.2.4 Inference results

For each VARMA (ARMA) model of Examples 3 and 4, we run 10 preliminary optimisations for (φ, ϑ) with different starting points. With diagonal MA indicator matrices defined in Equation (4.58), all optimisations converge to the same result. Hence, multimodality might not be a problem for these models.

MCMCs for Example 3 have 50000 samples each while MCMCs for Example 4 have only 10000 samples. This is due to the costly computational time required for 5-min dataset of Example 4. For a particular interested parameter, the MCSE half-width of 95% interval of its median is denoted by $l_{hw;\cdot}$, e.g. $l_{hw;\Phi}$.



(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.001252$, $\max(l_{hw;\Theta}) = 0.000248$

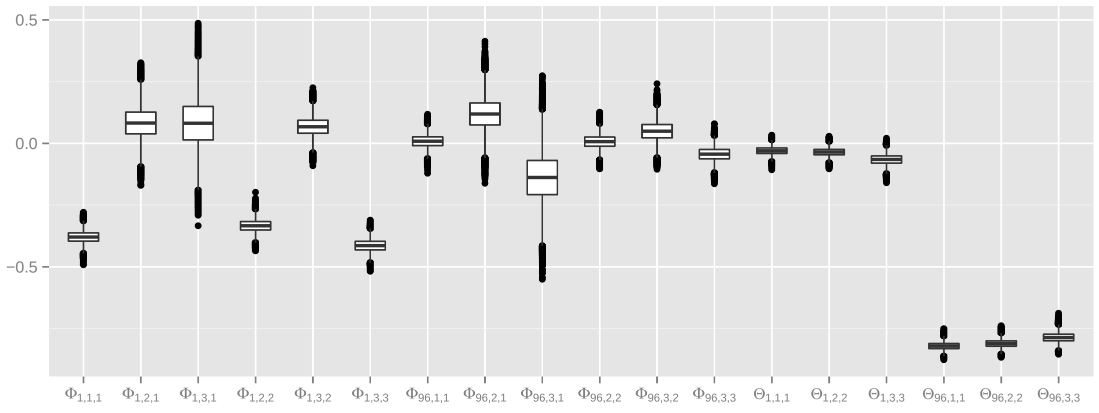


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.004511$, $\max(l_{hw;\Sigma_e}) = 0.421835$

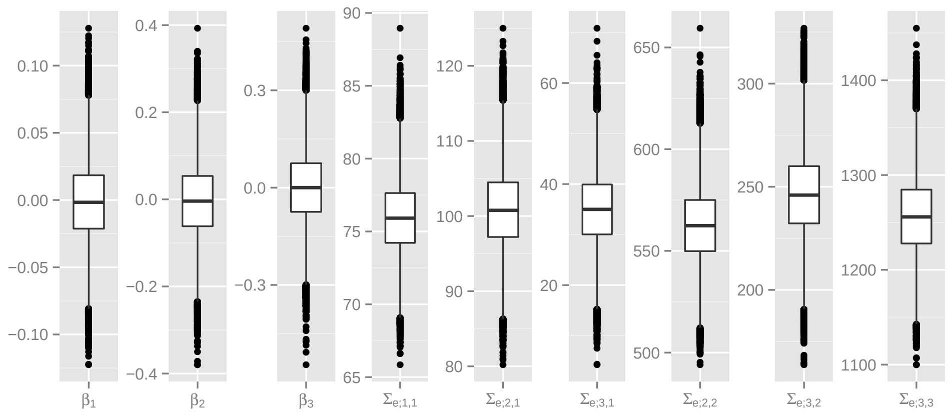
Fig. 4.19: Example 3: the MCMC sample box plots for the model SD-M-EX03.

We use the matrix representation $\varPhi$ ($\varTheta$) instead of the vector representation φ (ϑ) as it is easier to interpret the effect of one station to another, i.e. $\varPhi_{j,i,i'}$ represents the AR effect of lagged residual $y_{\cdot;t-j,i'}$ at station i' to the current residual $y_{\cdot;t,i}$ at station i.

The box plots of MCMC samples for VARMA (ARMA) models of Example 3 are shown in Figures 4.19 to 4.24. The plots for Example 4 are in Figures 4.25 to 4.30.



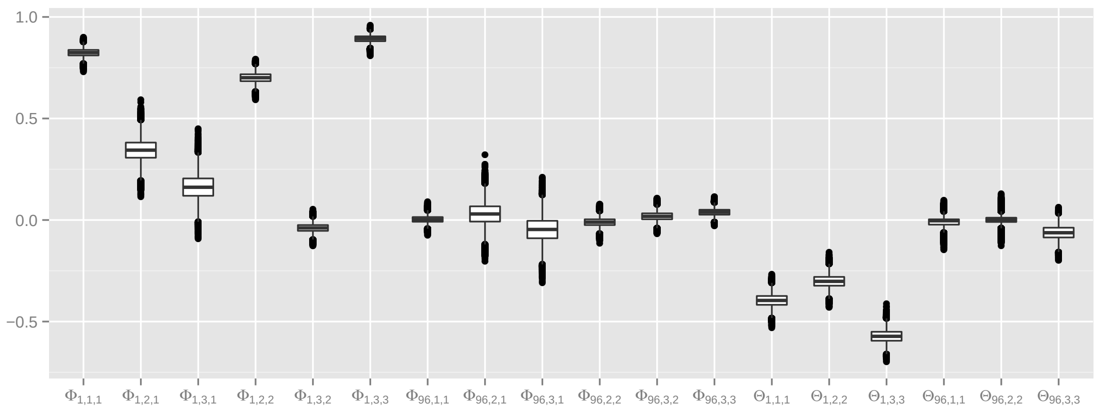
(a) $\varPhi$ (φ) and $\varTheta$ (ϑ): $\max(l_{hw;\varPhi}) = 0.001169$, $\max(l_{hw;\varTheta}) = 0.000271$



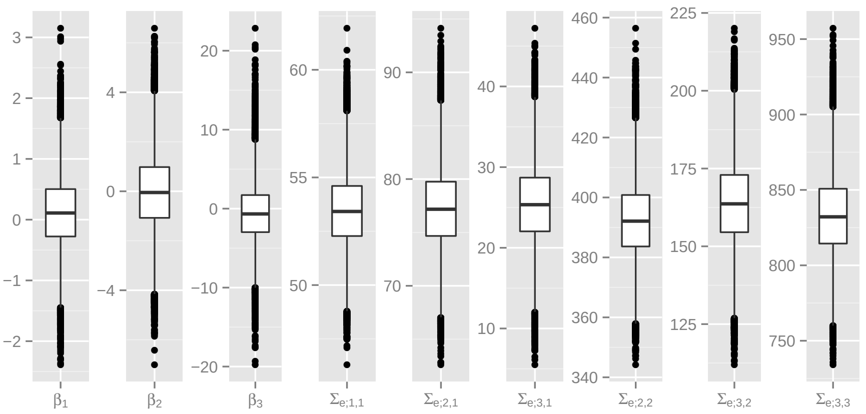
(b) β and $\varSigma_e$: $\max(l_{hw;\beta}) = 0.001219$, $\max(l_{hw;\varSigma_e}) = 0.455669$

Fig. 4.20: Example 3: the MCMC sample box plots for the model DSD-M-EX03.

Consistent with the previous autocorrelation analysis, seasonal MA parameters ($\Theta_{96,\cdot,\cdot}$, $\Theta_{288,\cdot,\cdot}$) of SD and DSD models are evidently negative. On the other hand, seasonal AR, MA parameters of MP models are quite insignificant. Also, even though there are fewer MCMC samples (10000) for models of Example 3, the MCSE half-widths of the corresponding MCMCs are still small enough, mainly because that the posteriors of these models are quite focused already.

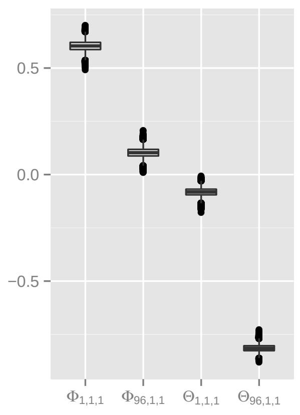


(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.000719$, $\max(l_{hw;\Theta}) = 0.000565$

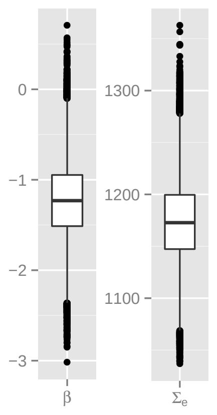


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.037952$, $\max(l_{hw;\Sigma_e}) = 0.312572$

Fig. 4.21: Example 3: the MCMC sample box plots for the model MP-M-EX03.

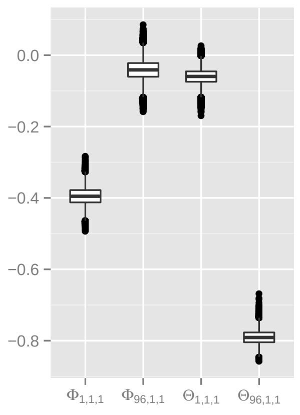


(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.000282$, $\max(l_{hw;\Theta}) = 0.000220$

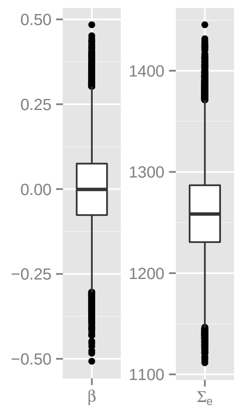


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.004504$, $\max(l_{hw;\Sigma_e}) = 0.433488$

Fig. 4.22: Example 3: the MCMC sample box plots for the model SD-U-EX03.

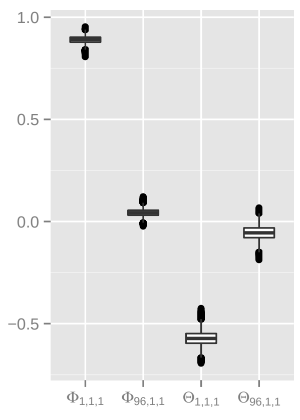


(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.000310$, $\max(l_{hw;\Theta}) = 0.000271$

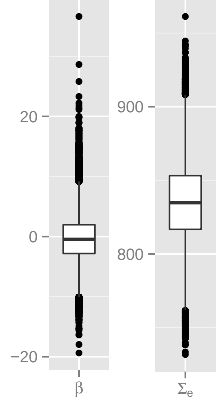


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.001200$, $\max(l_{hw;\Sigma_e}) = 0.465394$

Fig. 4.23: Example 3: the MCMC sample box plots for the model DSD-U-EX03.

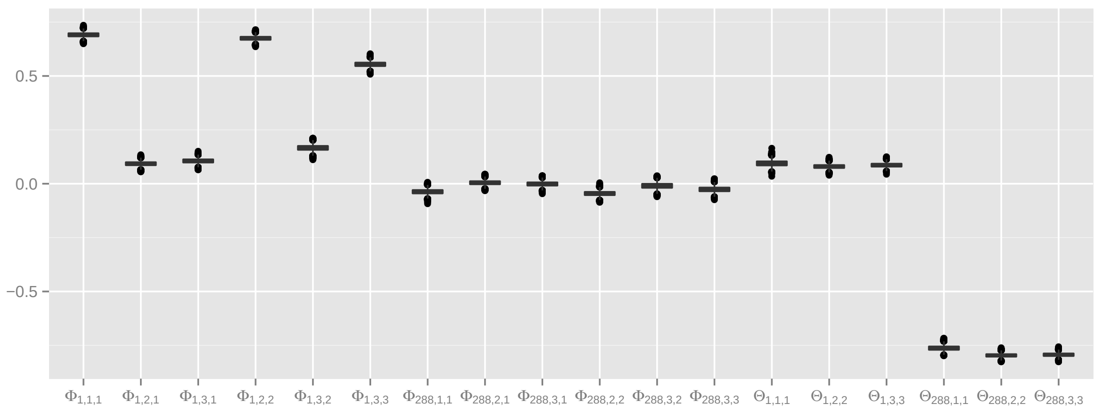


(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.000249$, $\max(l_{hw;\Theta}) = 0.000513$

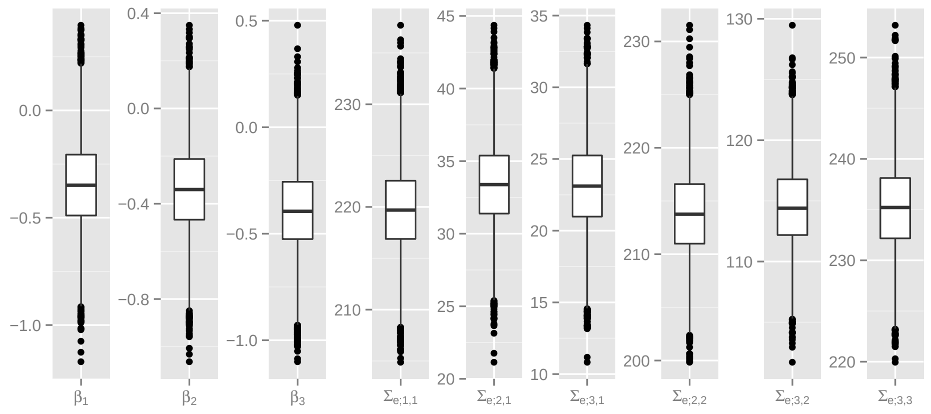


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.039358$, $\max(l_{hw;\Sigma_e}) = 0.283190$

Fig. 4.24: Example 3: the MCMC sample box plots for the model MP-U-EX03.

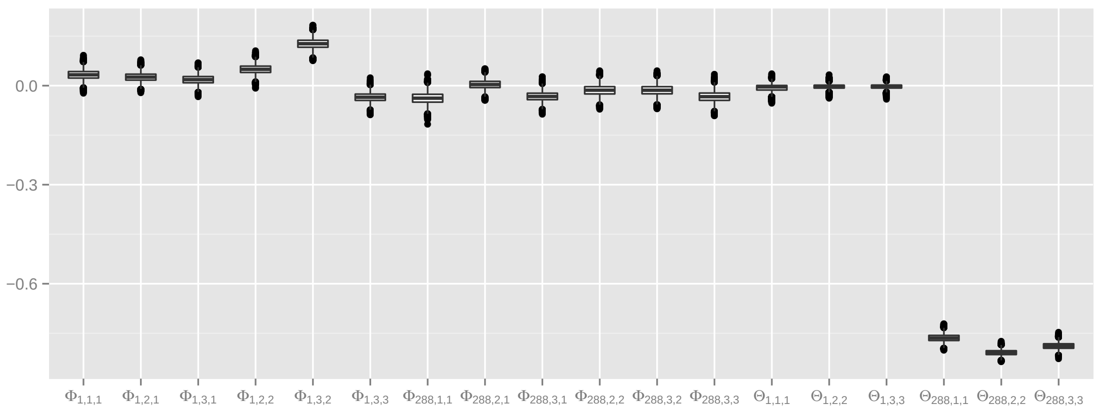


(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.000367$, $\max(l_{hw;\Theta}) = 0.000355$

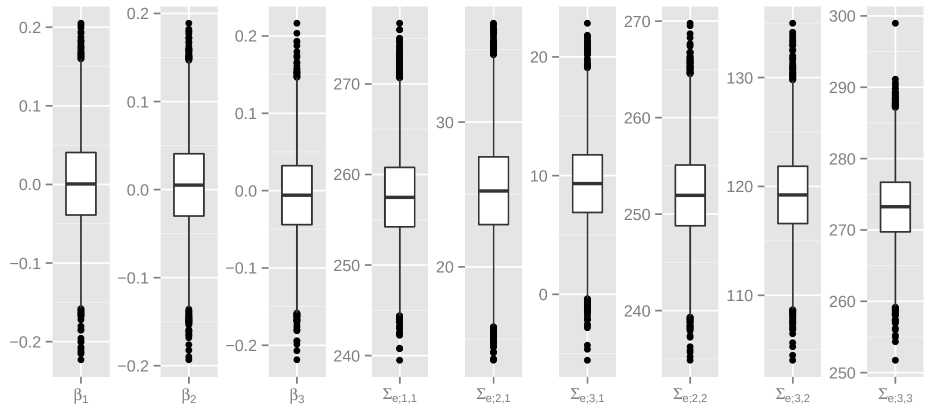


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.005182$, $\max(l_{hw;\Sigma_e}) = 0.111432$

Fig. 4.25: Example 4: the MCMC sample box plots for the model SD-M-EX04.

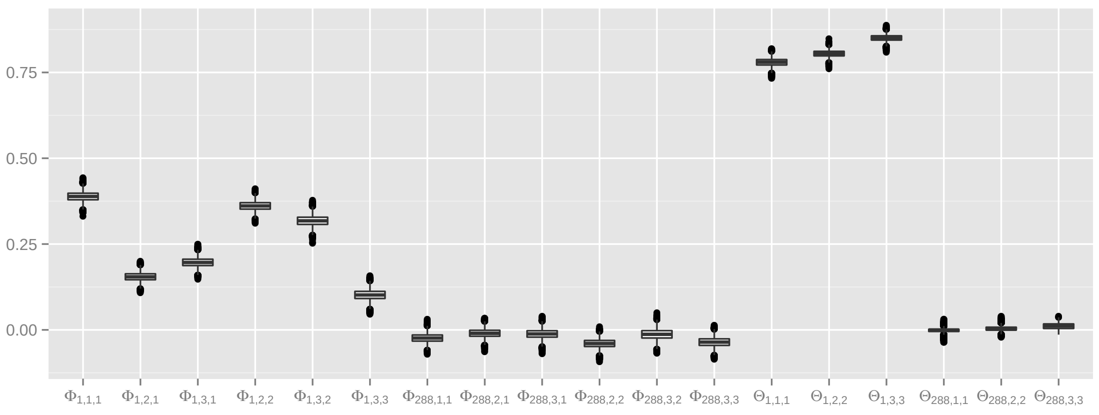


(a) $\varPhi$ (φ) and $\varTheta$ (ϑ): $\max(l_{hw;\varPhi}) = 0.000483$, $\max(l_{hw;\varTheta}) = 0.000523$

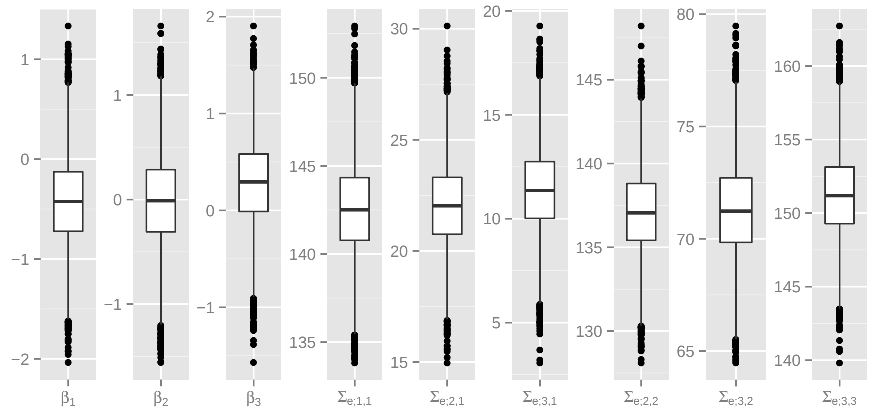


(b) β and $\varSigma_e$: $\max(l_{hw;\beta}) = 0.001561$, $\max(l_{hw;\varSigma_e}) = 0.117709$

Fig. 4.26: Example 4: the MCMC sample box plots for the model DSD-M-EX04.

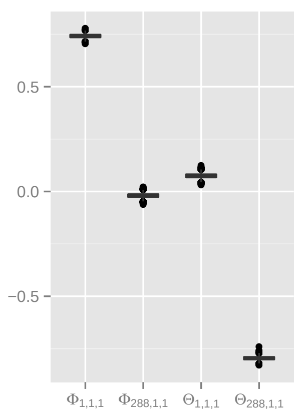


(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.000472$, $\max(l_{hw;\Theta}) = 0.000446$

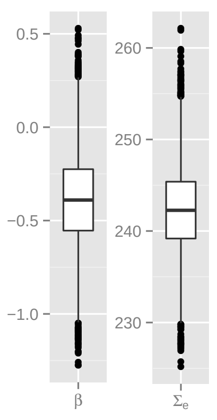


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.001188$, $\max(l_{hw;\Sigma_e}) = 0.062158$

Fig. 4.27: Example 4: the MCMC sample box plots for the model MP-M-EX04.

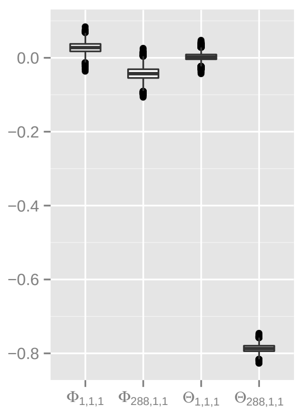


(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.000271$, $\max(l_{hw;\Theta}) = 0.000266$

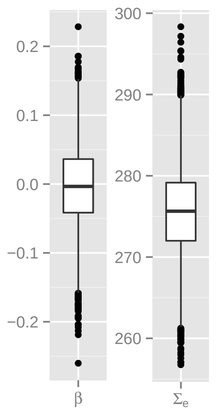


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.006128$, $\max(l_{hw;\Sigma_e}) = 0.115007$

Fig. 4.28: Example 4: the MCMC sample box plots for the model SD-U-EX04.

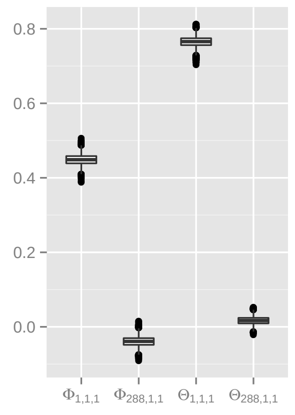


(a) Φ (φ) and Θ (ϑ): $\max(l_{hw;\Phi}) = 0.000412$, $\max(l_{hw;\Theta}) = 0.000278$

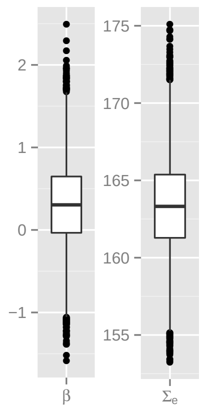


(b) β and Σ_e: $\max(l_{hw;\beta}) = 0.001358$, $\max(l_{hw;\Sigma_e}) = 0.122604$

Fig. 4.29: Example 4: the MCMC sample box plots for the model DSD-U-EX04.



(a) $\varPhi$ (φ) and $\varTheta$ (ϑ): $\max(l_{hw;\varPhi}) = 0.000396$, $\max(l_{hw;\varTheta}) = 0.000354$



(b) β and $\varSigma_e$: $\max(l_{hw;\beta}) = 0.013101$, $\max(l_{hw;\varSigma_e}) = 0.080183$

Fig. 4.30: Example 4: the MCMC sample box plots for the model MP-U-EX04.

Let $\Omega = (\varphi, \vartheta, \beta, \varSigma_e, \alpha_0)$. In the forecasting of VARMA models, the h-step-ahead prediction mean can be approximated by (the model index is dropped in this paragraph):

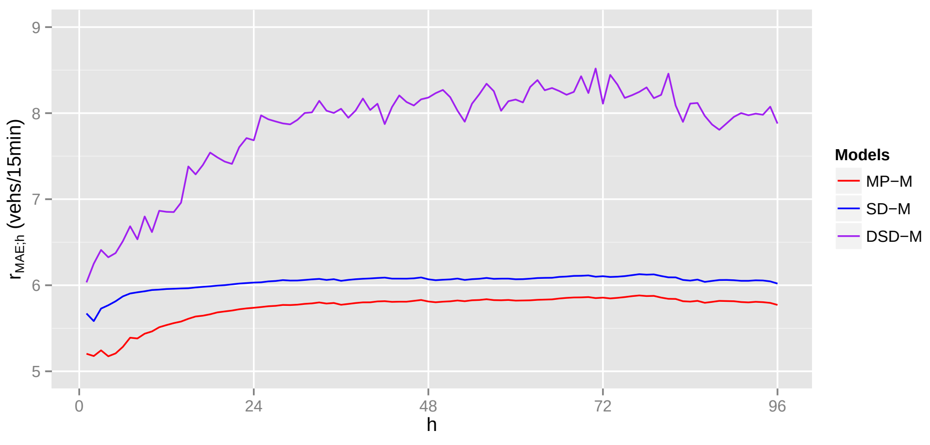
$$\begin{aligned} E_{\Omega|y_{1:n}}(y_{t+h}|y_{1:t}) &= \int y_{t+h}\ p_{y_{(t+1):(t+h)}|\Omega,y_{1:t}}(y_{(t+1):(t+h)})p_{\Omega|y_{1:n}}(\Omega)\ dy_{(t+1):(t+h)}d\Omega \\ &\approx \widehat{y}_{t+h|1:t} = \frac{1}{n_s}\sum_{i=1}^{n_s} y_{t+h,i}, \end{aligned} \tag{4.64}$$

where $y_{1:n}$ is the data part used in fitting; n_s is the number of MCMC samples. $\widehat{y}_{t+h|1:t}$ is the approximated h-step-ahead prediction of y_{t+h}, conditioning on $y_{1:t}$. For the approximation in the second line, conditioning on each MCMC sample $\Omega_i \sim p_{\Omega|y_{1:n}}(\Omega)$, a realisation $y_{(t+1):(t+h),i}$ is generated from $p_{y_{(t+1):(t+h)}|\Omega,y_{1:t}}(y_{(t+1):(t+h)})$ and then all samples $y_{t+h,i}$ are averaged.

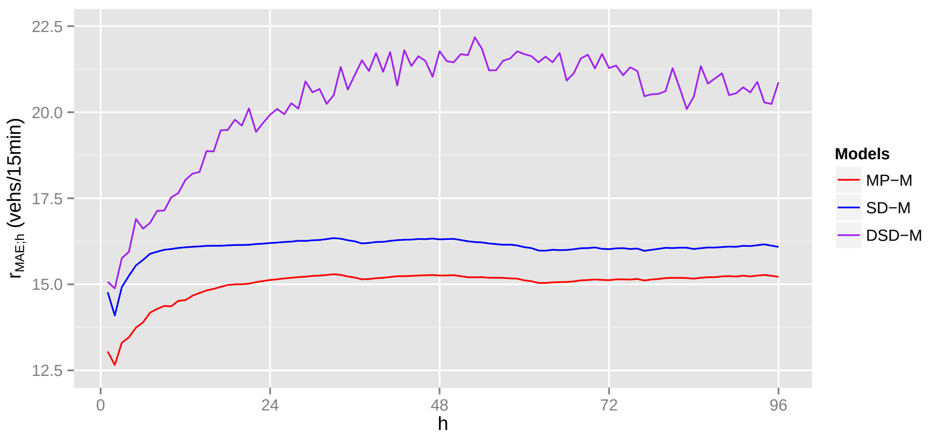
The prediction of the original traffic flow is similar to the above procedure except that each realisation $y_{(t+1):(t+h),i}$ is converted to $z_{(t+1):(t+h),i}$ by Equations (4.52), (4.53) and (4.55). For comparison purposes, the mean absolute error is computed by:

$$r_{MAE;h} = \frac{1}{|\mathscr{T}|}\sum_{t\in\mathscr{T}} |\widehat{z}_{t+h|1:t} - z_{t+h}|. \tag{4.65}$$

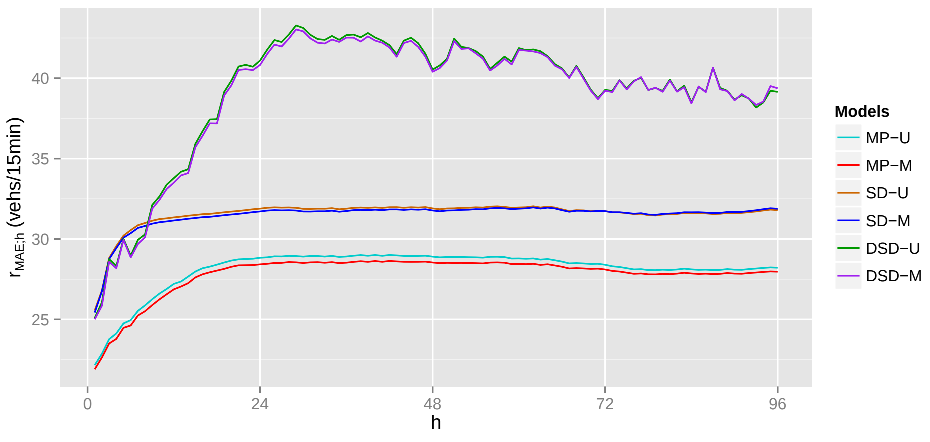
This error is evaluated by using another validation dataset for each example (6-day data for Example 3 and 5-day data for Example 4) and shown in Figures 4.31 and 4.32. Note that univariate models are only used for station 3.



(a) station 1

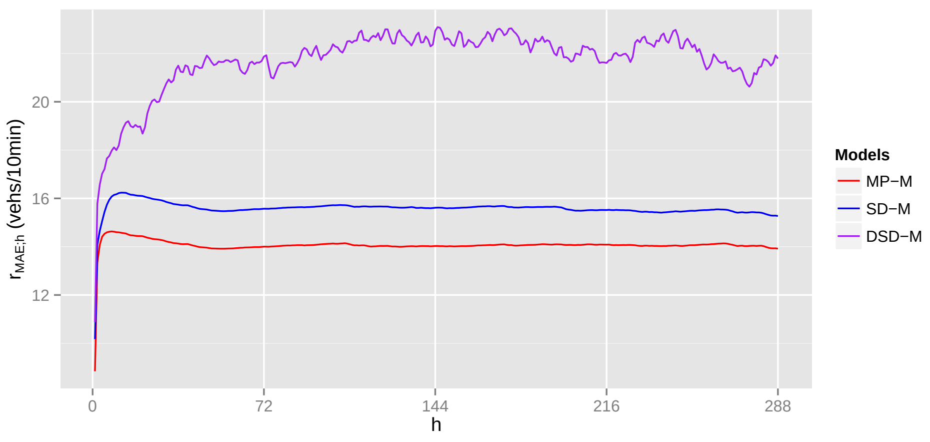


(b) station 2

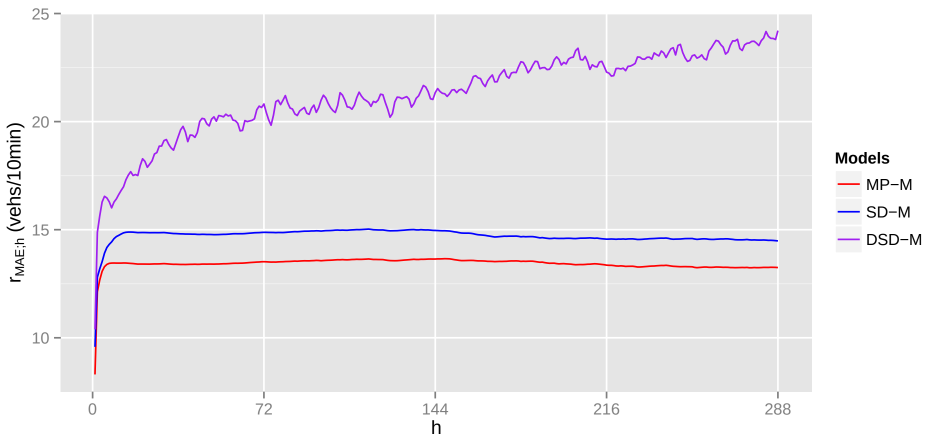


(c) station 3

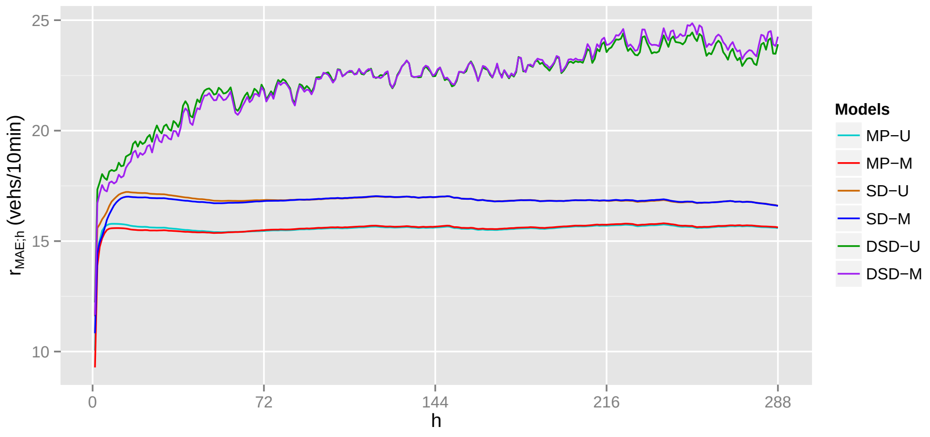
Fig. 4.31: Example 3: the mean absolute errors of h-step-ahead predictions.



(a) station 1



(b) station 2



(c) station 3

Fig. 4.32: Example 4: the mean absolute errors of h-step-ahead predictions.

Logically, the prediction error of step $(h+1)$ should be worse than prediction of step h but it is not always the case in the above plots, especially in the DSD models. This issue may be because that the validation dataset is not large enough, inducing randomness or bias in prediction error, or there is a mismatching between the models and the data.

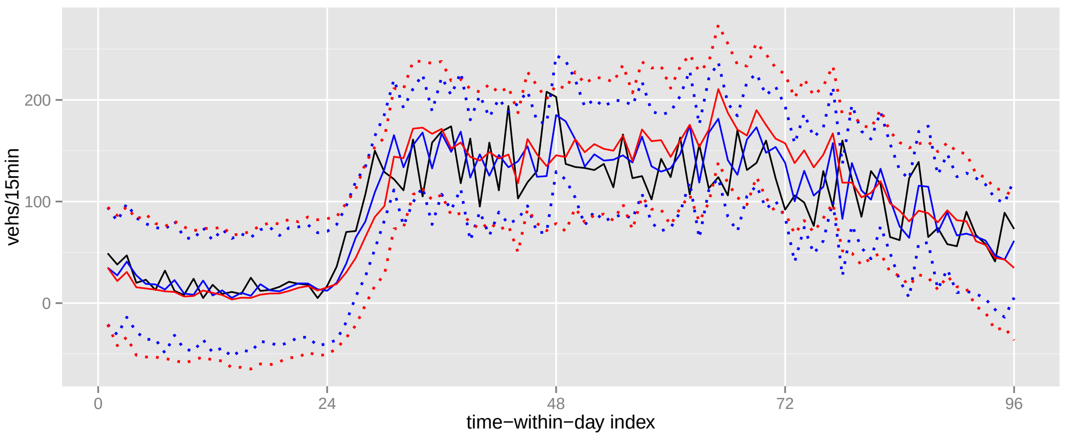
In overall, for these two real datasets, even though the predictions of multivariate models (station 3) are better, the error differences are unfortunately small. So, it may be better to use a univariate model to save the computation cost.

On the other hand, the prediction seems more sensitive to the preprocessing step, or equivalently the modelling of the process mean. MP models are consistently the best while DSD models have worst errors. The reason why the errors of DSD models are so bad can be perceived by examining the conversion from a sample $y_{(t+1):(t+h)}$ to $z_{(t+1):(t+h)}$ when $t+h-s \leq t$ (the sample index is dropped):

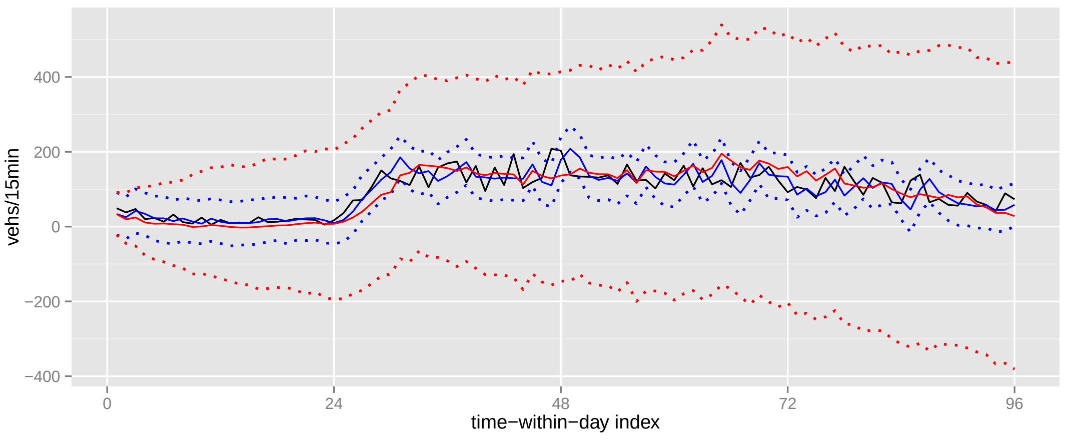
$$\begin{aligned}
&(z_{t+h} - z_{t+h-s}) - (z_{t+h-1} - z_{t+h-s-1}) = y_{t+h} \\
\Rightarrow\ & z_{t+h} = y_{t+h} + z_{t+h-1} + z_{t+h-s} - z_{t+h-s-1} \\
\Rightarrow\ & z_{t+h} = \sum_{i=1}^{h} (y_{t+i}) + z_t + z_{t+h-s} - z_{t-s}. \qquad (4.66)
\end{aligned}$$

As the prediction is conditioning on $z_{1:t}$, the term $(z_t + z_{t+h-s} - z_{t-s})$ is exact. So, the variance of prediction z_{t+h} comes from the sum of residuals y_{t+i} $(1 \leq i \leq h)$, which increases with the prediction step h. This issue of accumulated variance does not happen with SD and MP models.

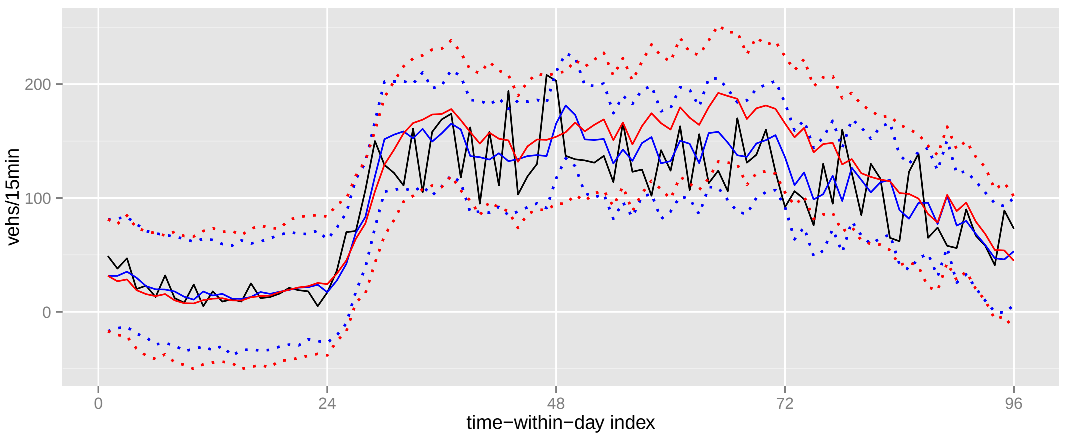
The one-step-ahead predictions and multi-step-ahead predictions (from step 1 to step s) of station 3 on day 21 are plotted in Figures 4.33 and 4.34. The predictions of univariate models are omitted as they are similar to multivariate models' predictions. The large $[5\% - 95\%]$ quantile intervals of DSD models confirm the above analysis of prediction variance. The one-step-predictions are always better and have smaller $[5\% - 95\%]$ quantile intervals.



(a) SD-M

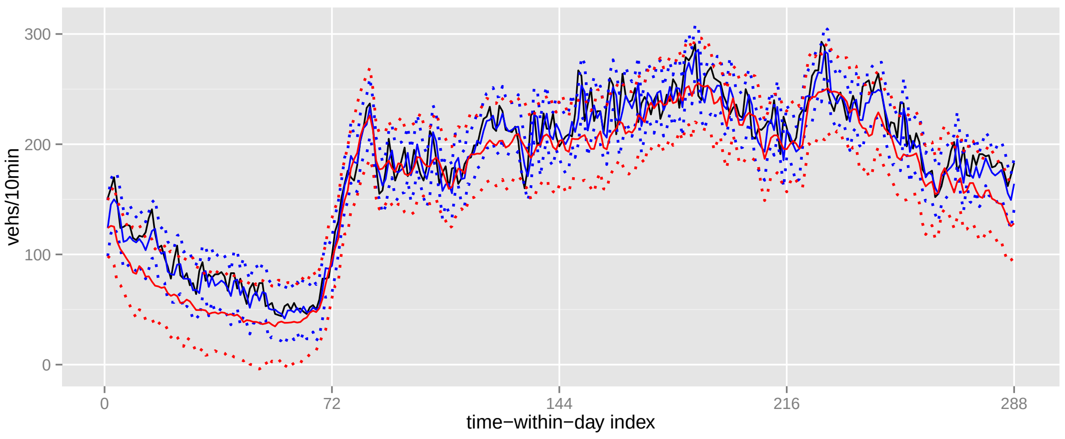


(b) DSD-M

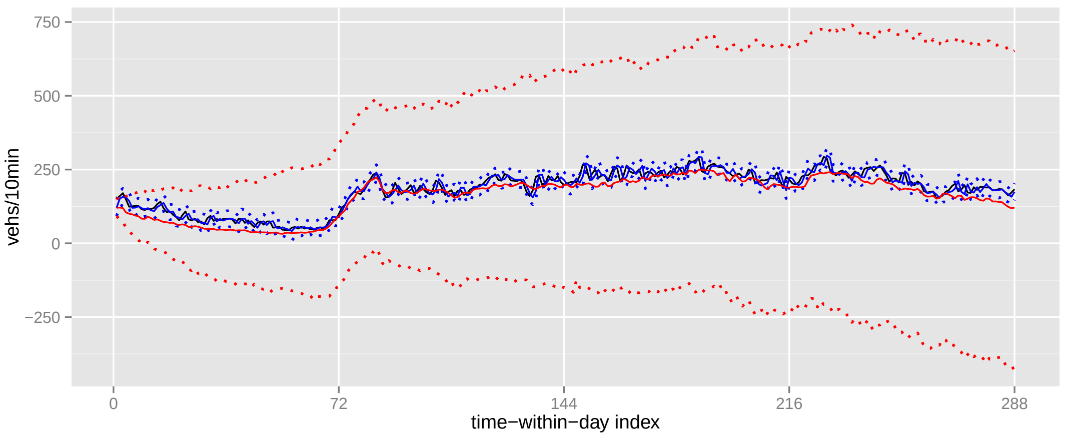


(c) MP-M

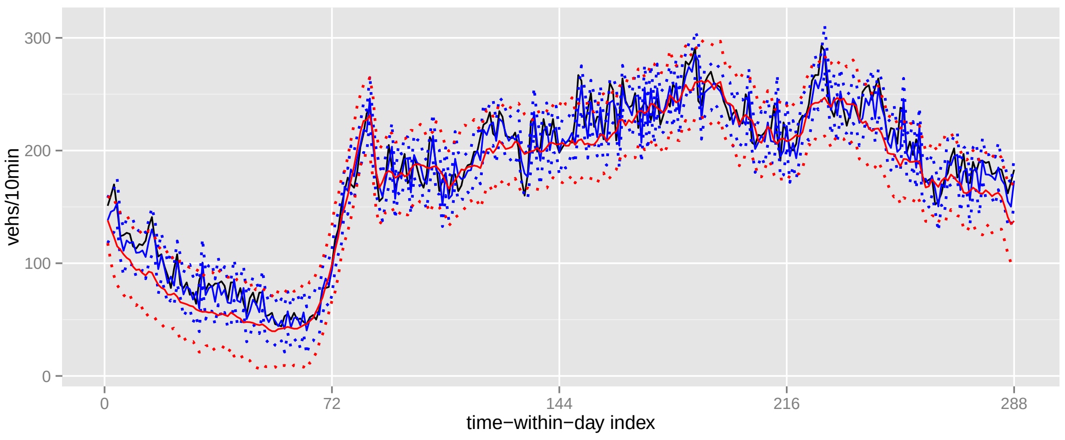
Fig. 4.33: Example 3: the blue and red solid lines are the means of one-step-ahead predictions and multi-step-ahead predictions (from step 1 to step s) of $z_{t,3}$. The coloured dot lines correspond to the 5% and 95% quantiles of the predictions.



(a) SD-M



(b) DSD-M



(c) MP-M

Fig. 4.34: Example 4: the blue and red solid lines are the means of one-step-ahead predictions and multi-step-ahead predictions (from step 1 to step s) of $z_{t,3}$. The coloured dot lines correspond to the 5% and 95% quantiles of the predictions.

4.2.5 Remarks of STFF

From the previous inference and analysis, we have several remarks about the VARMA model and the STFF problem:

- The difference operator of lag 1 should be avoided in the multi-step-ahead prediction as the variance increases with the prediction step.
- The mean and the residual of a time series are strongly related. In previous models, we use simple mean modelling for the traffic flow (implicitly or explicitly by different preprocessing methods) and focus on the VARMA modelling of the residual. However, it has been seen that the prediction result is more sensitive to the preprocessing method, or equivalently the mean. So, maybe it is better to model a mean function as close to the true value as possible first, i.e. it is modelled so that the variance of the resulting residual is small. For the target problem, perhaps using other traffic variables may yield a better mean function.
- When the traffic networks become larger, the number of parameters increases unavoidably. In that case, VARMA models and other spatial-temporal models may have a scalability problem as the computation cost increases sharply and the inference process cannot be parallelised. So, an ideal model should be designed so that its inference process can be divided into as many small tasks or components as possible. This divisibility property can be achieved by either computing parallelism or modelling.
- To meet the time constraint of real-time applications, the modelling should facilitate the application of sequential inference.

For a normal traffic dataset, i.e. there is no incident and the traffic pattern is very similar across all days, a simple model like MP-U works really well for the STFF problem. However, when there is a shift of traffic pattern, the prediction may not be precise any more as the fitted model has been long adapted to the previous pattern. We will illustrate this problem with an unusual traffic dataset.

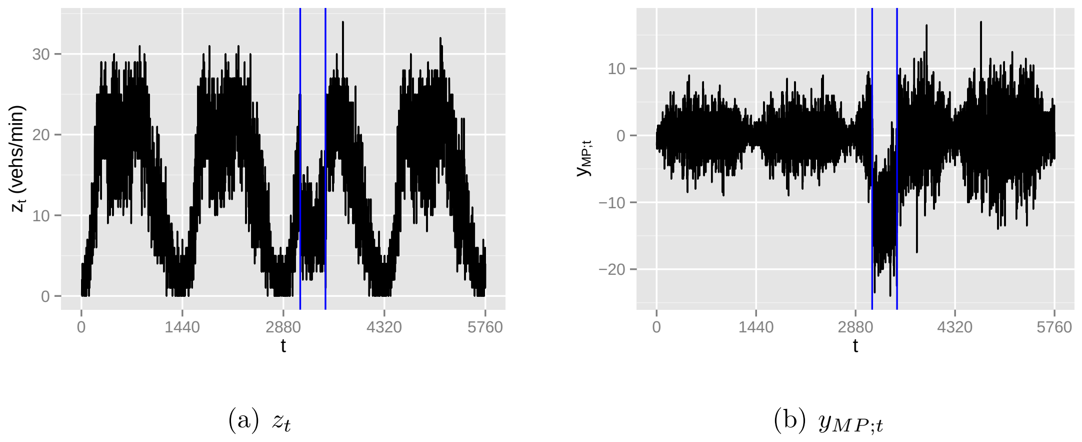
The 4-day traffic flow plotted in Figure 4.35(a) is a part of a 1-minute VISSIM dataset in Section 6.1.2. In the flow series of Figure 4.35(a), a traffic incident occurs on day 3 (at around $t = 3120$) and the traffic pattern changes quite significantly.

The preprocessing method MP is used with a time-varying daily mean m_t which is the daily average of the first 2 day flow z_t. The resulting residual $y_{MP;t}$ is plotted in Figure 4.35(b).

A simple pure zero-mean ARMA(2,2) is fitted to $y_{MP;1:n_f}$ where n_f is the ending point of the fitting set. We would like to see the behaviour of ARMA prediction during the incident interval by adjusting the fitting ending point n_f. For statistical inference, simple MLE and plugged-in prediction are used in this illustration.

The one-step-ahead predictions obtained with $n_f = 3130$, 3145 and 3155 are shown in Figure 4.36(a). It can be seen that even though the incident has occurred at time $t = 3120$, the ARMA(2,2) is only "adapted" to the new pattern and provides reasonable one-step-ahead prediction with $n_f \geq 3155$.

For the purposes of multi-step-ahead prediction, the model needs longer time for the adjustment. The 200-step-ahead predictions with $n_f = 3155$, 3165 and 3175 are plotted in Figure 4.36(b) which clearly shows the exponential decaying effect of ARMA prediction. Only with more traffic incident data points that ARMA can adjust the decaying rate better to the unusual pattern.



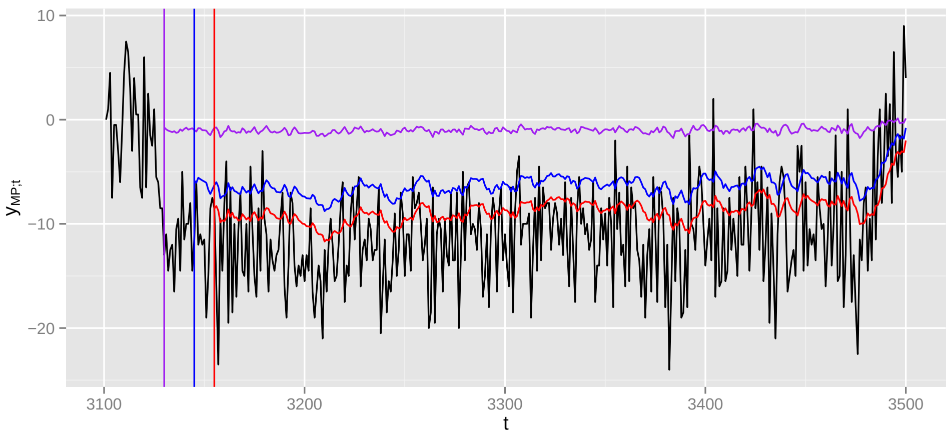
(a) z_t (b) $y_{MP;t}$

Fig. 4.35: The traffic flow z_t and the residual $y_{MP;t}$ of a 1-minute VISSIM dataset. The blue vertical lines mark a duration when a traffic incident occurs.

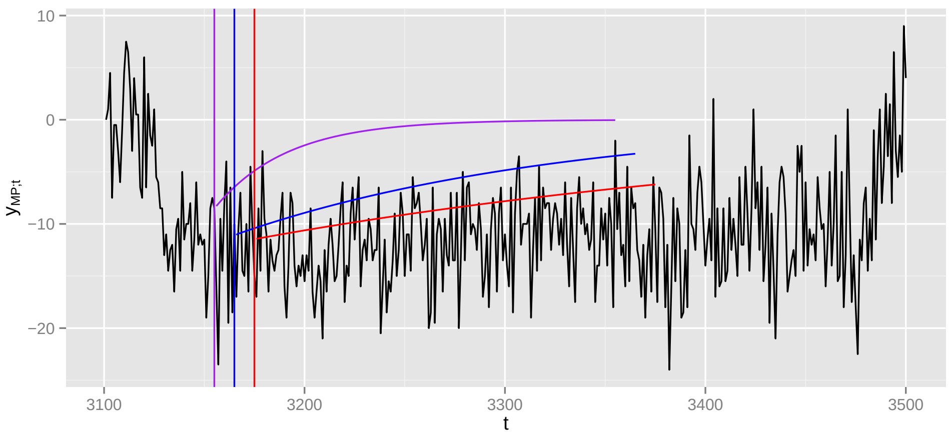
So, in this illustration of unusual traffic pattern, there are two important points:

- A fitted model, e.g. a VARMA model, may be too adapted to the normal traffic pattern. So, when an incident occurs, its performance becomes worse and re-fitting is necessary. This requirement again calls for sequential inference.

- To avoid the case where a model needs to shift back and forth between normal and unusual traffic patterns, a good model should portray a ”stable relationship” between traffic variables. A stable relationship is one that stays almost correct (with small error) in any case. Such a relationship may come from the transportation theory but probably requires more traffic information, e.g. speed, density, etc.



(a) one-step-ahead prediction



(b) 200-step-ahead prediction

Fig. 4.36: One-step-ahead and multi-step-ahead predictions of the residual $y_{MP;t}$ during a traffic incident. The coloured curves are plugged-in prediction means obtained by fitting ARMA(2,2) with different fitting sets. The coloured vertical lines mark the ending points of the corresponding fitting sets.

This concludes the application of the VARMA model in STFF. We will return to the STFF problem with a new spatial-temporal model in Chapter 6.

4.3 Conclusion

In this chapter, we have introduced the VARMA model with a sparse form and user-defined constraint on AR, MA matrices. Such a model may use fewer parameters when designed with extra information from the target problem. For the variable correlation problem of the VARMA model, an improved MCMC algorithm is proposed, making use of the marginalisation and the correlation direction information in the Hessian matrix. Then, we apply the VARMA model to the STFF problem with two traffic datasets in Dublin, using the network information to constrain the structure of AR matrices. VARMA and ARMA models with different preprocessing methods are compared in terms of one-step-ahead and multi-step-ahead predictions. Finally, we conclude the chapter with several remarks on the VARMA model and the STFF problem.

Chapter 5

Sequential inference with the iterLap approximation

This chapter discusses two statistical inference implementation methodologies: functional approximation and sequential inference with unknown parameters. For the functional approximation, we introduce some modifications to the implementation of the iterLap method, previously mentioned in Section 2.3.2, then compare the performance of two versions with some examples. Based on this modified approximation, a sequential inference method for the DM is proposed, characterised by two key points. Firstly, the method relies on a smooth and continuous functional approximation instead of discrete particles, which are very prone to degeneracy. Secondly, it is able to do joint sequential inference on the state vector and the unknown parameter.

The motivation of sequential inference in this thesis originates from a dynamic submodel which is a part of a spatial-temporal model in the next chapter. Also, in a real-time application like STFF, sequential inference is essential in obtaining a feasible solution within a time constraint.

5.1 The iterated Laplace approximation

5.1.1 Modifying the iterLap method

The iterLap method has been summarised in Section 2.3.2 with an example. After checking its R implementation, we try to improve its performance by some modifications.

Stopping rule by the normalising constant

As a reminder, iterLap approximates a non-normalised density $q_x(\cdot)$ at iteration n_c by:

$$q_x(x) \approx \widetilde{q}_{n_c;x}(x) = \sum_{i=1}^{n_c} w_i N(x|\mu_i, Q_i), \tag{5.1}$$

where μ_i, Q_i are found by an optimisation procedure and non-normalised weights w_i are estimated by quadratic programming in Section 2.3.2. By the above approximation, the normalising constant ζ of $q_x(\cdot)$ is estimated by:

$$\zeta = \int q_x(x)dx \approx \widetilde{\zeta}_{n_c} = \int \widetilde{q}_{n_c;x}(x)dx = \sum_{i=1}^{n_c} w_i. \tag{5.2}$$

The constant $\widetilde{\zeta}_{n_c}$ represents the probability mass of the approximated density $\widetilde{q}_{n_c;x}(x)$. Hence, in Bornkamp (2011*a*), the iterative process stops when $\widetilde{\zeta}_{n_c}$ does not improve any more, i.e. satisfying the following equation:

$$|\widetilde{\zeta}_{n_c} - 0.5(\widetilde{\zeta}_{n_c-1} + \widetilde{\zeta}_{n_c-2})|/\widetilde{\zeta}_{n_c} < \delta_\zeta, \tag{5.3}$$

with a predefined threshold δ_ζ.

However, even though a new component $N(x|\mu_i, Q_i)$ does not improve the estimated volume of $\widetilde{q}_{n_c;x}(x)$, it may still correct the density and decrease the approximation error. Furthermore, a new component generates more explored points which may be useful in the optimisation and quadratic programming step. Hence, we remove this stopping criterion from the iterLap source code.

Residual function

At each iteration, iterLap (Bornkamp, 2011*b*) finds a new component by minimising a residual function $g_{a;n_c}(x)$ with:

$$z = q_x(x) - \widetilde{q}_{n_c;x}(x), \tag{5.4}$$

$$g_{a;n_c}(x) = \begin{cases} -\log(z) & \text{if } z \geq z_l > 0 \\ -\log(\exp(z - z_l)z_l) & \text{if } z < z_l \end{cases}, \tag{5.5}$$

where z_l is a predefined positive lower bound. The new component's mean and precision matrix are obtained by:

$$\mu_{n_c+1} = \arg\min_x \{g_{a;n_c}(x)\}, \tag{5.6}$$

$$Q_{n_c+1} = \left.\frac{\partial^2 g_{a;n_c}(x)}{\partial x^2}\right|_{x=\mu_{n_c+1}}. \tag{5.7}$$

We will use the following example to illustrate and discuss the above step.

Example 5 Consider a non-normalised density $q_x(\cdot)$:

$$q_x(x) = \exp\left(-\frac{1}{2\times 5^2}x^2 - \frac{1}{2\times 5^2}(|x| - 0.5)_+^3\right), \tag{5.8}$$

where $a_+ = a$ if $a \geq 0$ and $a_+ = 0$ if $a < 0$.

At iteration 1, $q_x(\cdot)$ is approximated by $\widetilde{q}_{1;x}(\cdot)$ with a single normal component and the residual function is $g_{a;1}(\cdot)$. The functions $q_x(\cdot)$, $\widetilde{q}_{1;x}(\cdot)$ and $\exp\left(-g_{a;1}(\cdot)\right)$ are plotted in Figure 5.1. The maximum points of $\exp\left(-g_{a;1}(\cdot)\right)$ or equivalently, the minimum points of $g_{a;1}(\cdot)$, are also shown, representing the next potential component means.

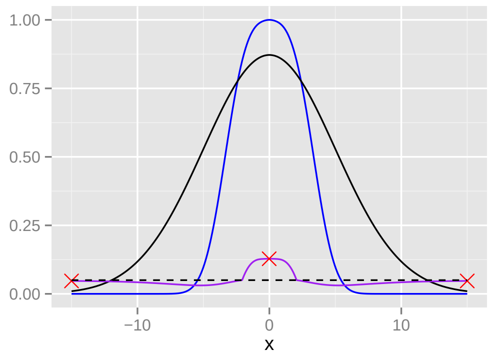


Fig. 5.1: Example 5: the blue and black lines show the target density $q_x(\cdot)$ and the approximated density $\widetilde{q}_{1;x}(\cdot)$; the purple line is $\exp\left(-g_{a;1}(\cdot)\right)$ with red crosses as maximum points (the leftmost and rightmost crosses are actually inf and $-$inf); the lower bound $z_l = 0.05$ is marked by dashed horizontal line.

Figure 5.1 shows that there are two potential maximum points at inf and $-$inf, which are not good locations for new components at all because they do not have any effect on the approximation. So, these locations should be avoided to save computation time.

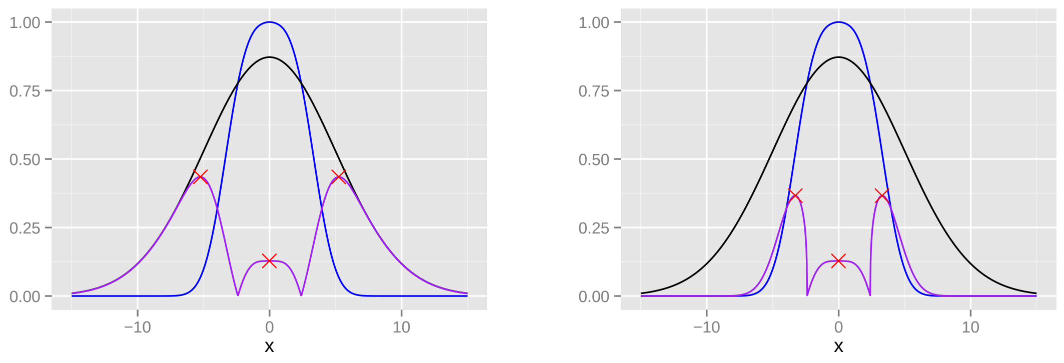
From Equation (5.4) and Figure 5.1, it can be seen that iterLap prefers choosing the locations at which $q_x(x)$ is significantly underestimated by $\widetilde{q}_{n_c;x}(x))$. The locations in the overestimated region ($q_x(x) < \widetilde{q}_{n_c;x}(x))$) are ignored. However, by experimenting with several residual functions and examples, we find that adding new components in the overestimated region does improve the approximation. Partially, this may be because the explored variable space is extended by a more lenient rule, which in turn improves the optimisation and quadratic programming steps.

So, we decide to use a new residual function $g_{n_c;r_b}(x)$ with:

$$z = q_x(x) - \widetilde{q}_{n_c;x}(x), \tag{5.9}$$

$$g_{b;n_c}(x) = \begin{cases} -\log(z+\epsilon_z) & \text{if } z \geq 0 \\ -[\log(-z+\epsilon_z) + \alpha(\log(q_x(x)) - lq_{max;x})]/(1+\alpha) & \text{if } z < 0 \end{cases}, \tag{5.10}$$

where $lq_{max;x}$ is the maximum log value of $\log(q_x(x))$ until the current iteration; a very small constant $\epsilon_z = e^{-10}$ is used only for the positivity condition of the log operator; $\alpha > 0$ is an optional coefficient which pulls the optimisation point of $g_{b;n_c}(x)$ to a location of high log density value, $\log(q_x(x))$. Figure 5.2 illustrates the above residual function $g_{n_c;r_b}(x)$ with $\alpha = 0$ and $\alpha = 3$.



(a) $\alpha = 0$ (b) $\alpha = 3$

Fig. 5.2: Example 5: the blue, black and purple lines show the target density $q_x(\cdot)$, the approximated density $\widetilde{q}_{1;x}(\cdot)$ and the function $\exp(-g_{b;1}(\cdot))$ with red crosses as maximum points of $\exp(-g_{b;1}(\cdot))$.

Select starting points for the optimisation

Using different starting points for the optimisation of $g_{b;n_c}(x)$ results in different component means. Bornkamp (2011*a*) uses the ratio $q_x(x)/\widetilde{q}_{n_c;x}(x)$ as the criterion of choosing the optimisation starting points. However, such a criterion may lead to a point located at the distribution tail. For example, with a t-distribution $q_x(x)$ and a normal distribution $\widetilde{q}_{n_c;x}(x)$, the ratio is extremely large at the tail.

So, we use the absolute difference $|q_x(x) - \widetilde{q}_{n_c;x}(x)|$ as a selection criterion which is closely related to the residual function $g_{b;n_c}(x)$. As a reminder of Section 2.3.2, iterLap keeps a $m \times d_x$ matrix X of all explored locations, a vector y of target density values $q_x(\cdot)$ and a $m \times n_c$ matrix Z of component density values $N(\cdot|\mu_i, Q_i)$ evaluated at the explored locations. The algorithm to select starting points X_s from X is shown in Algorithm 12.

Algorithm 12 Select optimisation starting points

```
1. Let $X_a = X$. Remove from $X_a$ points $x_i$ with small target density values, i.e.
   satisfying the condition: $\log(q_x(x_i)) - lq_{max;x} < \delta_{lq}$
2. Find a point $x_j$ in $X_a$ which maximises $|q_x(x_j) - \widetilde{q}_{n_c;x}(x_j)|$. Add that point $x_j$ to $X_s$.
3. Remove from $X_a$ points $x_k$ close to $x_j$.
4. Stop the algorithm if there are enough starting points in $X_s$ or if $X_a$ is empty.
   Otherwise, repeat steps $2 \to 4$.
```

A note on standardising $q_x(\cdot)$ and $\widetilde{q}_x(\cdot)$

Usually, the comparison between two densities is done based on their normalised densities. However, aside from the difficulty of obtaining the normalising constant, there is another issue. It is almost impossible to have a "good" approximation at the tail by any method. The absolute difference $|q_x(x) - \widetilde{q}_x(\cdot)|$ at a specific location at the tail may be extremely small but in high dimension space, the sum of all the absolute differences in every direction of the tail may become significant. Consequently, even though $\widetilde{q}_x(\cdot)$ is a "good" approximation of $q_x(\cdot)$ locally, the normalised $\widetilde{p}_x(\cdot)$ is however a "poor" approximation of the normalised $p_x(\cdot)$.

Hence, instead of globally normalising $q_x(\cdot)$ and $\widetilde{q}_x(\cdot)$, we only standardise these two densities on a user-defined grid X_g. $q_x(\cdot)$ is evaluated for all grid points x_j and the standardised density $r_x(\cdot)$ on the grid is obtained by:

$$r_x(x_j) = \frac{q_x(x_j)}{\sum_{j'} q_x(x_{j'})}. \tag{5.11}$$

The standardised $\widetilde{r}_x(\cdot)$ is obtained in a similar manner. So, local comparison of two densities can be done by analysing the standardised densities. There is one statistic $s(r_x, \widetilde{r}_x)$ that we find reasonable for the comparison purposes between two standardised densities:

$$s(r_x, \widetilde{r}_x) = \sum_{x_i \in X_g} |r_x(x_i) - \widetilde{r}_x(x_i)|. \tag{5.12}$$

With the same grid X_g, $s(r_x, \widetilde{r}_{a;x})$ can be compared with another $s(r_x, \widetilde{r}_{b;x})$.

Scaling a component's Hessian

In iterLap, the Hessian of $g_{b;n_c}(x)$ at the mode is used as the component's precision matrix. Usually, the sharper the curvature of $g_{b;n_c}(x)$, the larger the eigenvalues of the Hessian matrix at the mode and the more focused the corresponding normal component.

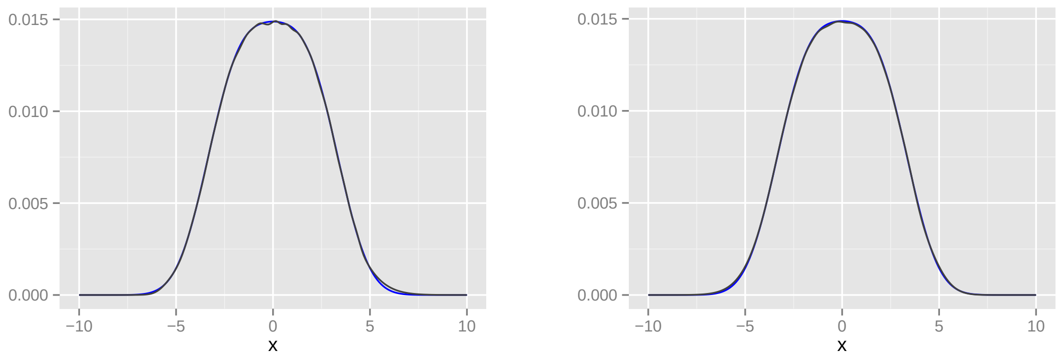
In some cases, the numerically evaluated Hessian matrix at the mode is sharper than the actual curvature and iterLap may fail to improve such a target density. For example, in Example 5, the function $\log(-q_x(x))$ is a quadratic function near the mode $x = 0$ but it becomes a cubic function with much sharper curvature when $|x| > 0.5$. As a result, the approximated $\widetilde{q}_{1;x}(x)$ at the first iteration is much flatter than the target density, which is shown in Figure 5.2. Unfortunately, in this example, even with more extra components, the iterLap approximation does not improve.

There are two ways to get around this issue. Firstly, we can allow a manual scaling of the Hessian matrix. With a user-defined scale factor κ_a, a new component is added with a precision matrix:

$$Q_{n_c+1} = \kappa_a \left. \frac{\partial^2 g_{a;n_c}(x)}{\partial x^2} \right|_{x=\mu_{n_c+1}}. \tag{5.13}$$

This modification is analogous to using kernel density estimation with a small bandwidth. A component with large precision matrix affects the approximation in a small region.

With $\kappa_a = 1.5$, the standardised densities for Example 5 are shown in Figure 5.3(a). Clearly, the approximation becomes much better.



(a) $\kappa_a = 1.5$ with no multiplicative scaling. $\widetilde{r}_{n_c;x}(\cdot)$ has $n_c = 14$ components

(b) Multiplicative scaling with $n_{dup} = 3$ and $\kappa_b = 1.25$. $\kappa_a = 1$. $\widetilde{r}_{n_c;x}(\cdot)$ has $n_c = 12$ components

Fig. 5.3: Example 5: the iterLap approximation with hessian scaling and multiplicative scaling; the blue and black curves are the standardised densities $r_x(\cdot)$ and $\widetilde{r}_{n_c;x}(\cdot)$.

One potential problem with this manual scaling is that all precision matrices become larger and all corresponding components are sharper, affecting only small local regions centred at the means. Similar to the case of using a small bandwidth in kernel method, the approximation becomes wriggly and only gets smoother with a larger number of components.

In a second attempt, we try to only adjust the Hessian matrix when needed. In Figure 5.2, one of the potential means for the next components is $\mu = 0$ (by the optimisation of $g_{b;1}(\cdot)$). As $\mu = 0$ is already a component mean of the approximated $\widetilde{q}_{1;x}(\cdot)$, iterLap will not accept $\mu = 0$ as a mean of a new component and try to find other locations. Also, even if $\mu = 0$ is accepted, a new component with mean $\mu = 0$ and the usual precision matrix will not make any difference. So, we make the following modifications:

- Allow the duplication of component means but there are no more than n_{dup} duplicates at a specific location. (x_{i_2} is a duplicate of x_{i_1} if x_{i_1}, x_{i_2} are close enough)

- Assume that the component $N(\cdot|\mu = x_{i_1}, Q = Q_{i_1})$ is added first at the location

x_{i_1}. Then, more duplicates x_{i_j} of x_{i_1} are found and corresponding components $N(\cdot|\mu = x_{i_j}, Q_{i_j})$ are added to the iterLap approximation. Q_{i_j} is calculated by:

$$Q_{i_j} = \kappa_b^j Q_{i_1}, \tag{5.14}$$

with a user-defined factor κ_b

We call this multiplicative scaling which can be used in conjunction with the usual Hessian scaling. In that case, only the first Hessian Q_{i_1} is scaled with κ_a. However, usually either Hessian scaling or multiplicative scaling is used. The approximation for Example 5 with multiplicative scaling ($n_{dup} = 3$ and $\kappa_b = 1.25$) is shown in Figure 5.3(b).

Other modifications

There are some other modifications to iterLap, including:

- The functionality of manually adding optimisation starting points to X_s and explored points to X. Generally, when a new normal component with mean μ is added to the iterLap approximation, aside from the starting points and explored points proposed by iterLap, a user can manually add points relative to the mean μ. This is useful in the case a user knows something about the target density, e.g. the support or the variable correlation. For example, a user may add more starting points, checking if there is any unexplored mode in the interested support. Another potential usage of this functionality is when x can be decomposed into (x_a, x_b) and the conditional expectation $E_{x_a|x_b}(x_a)$ is known by a function $f(x_b)$ (this frequently occurs in Bayesian inference with conjugate prior). Hence, when a new component with mean $\mu = (\mu_a, \mu_b)$ is added, it may be worth to check the locations $(f(\mu_b + \delta_b), \mu_b + \delta_b)$.

- In the quadratic programming step for estimating w, we use weighted least squares:

$$w = \arg\min_b [(y - Zb)^T (\omega_x I)(y - Zb)], \quad b \geq 0. \tag{5.15}$$

where $\omega_{x;i}$ is a weight for a explored point x_i in X. It is designed that a user can adjust the weight $\omega_{x;i}$ for a location of a component mean $\mu = x_i$.

- Make the code more robust to computer numerical issues, e.g. derive the matrix Z from a normalised log version, make the Hessian matrix of $g_{b;n_c}(x)$ positive definite and scale the parameters in the quadratic programming step.

The iterLap version with all above modifications is named mod-iterLap and will be compared the original iterLap by some examples in the next section. In each example, both versions are run with a predefined maximum number of components $n_{c;max}$ but the resulting approximations may have less components than $n_{c;max}$ due to stopping rules. Furthermore, in mod-iterLap, we use a simplification step to remove insignificant components, e.g. components with normalised weights $w_{n;i} < e^{-5}$. This simplification reduces the computation cost when the approximation is used for other purposes.

5.1.2 Comparison

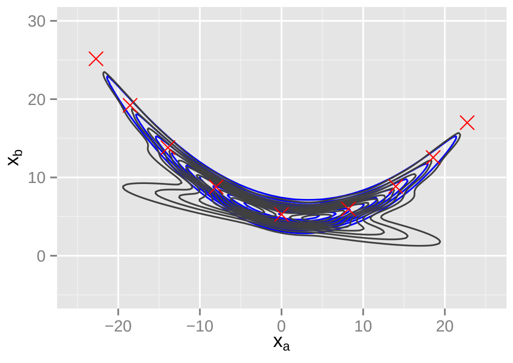
Firstly, we consider a density with a non-linear dependency in two-dimensional space. In this section, notations $\widetilde{q}_{o;x}(\cdot)$, $\widetilde{r}_{o;x}(\cdot)$ are the approximated densities of the original iterLap and $\widetilde{q}_{m;x}(\cdot)$, $\widetilde{r}_{m;x}(\cdot)$ are for the modified iterLap.

Example 6 Define a target density $p_x(\cdot)$ on $x = (x_a, x_b)$:

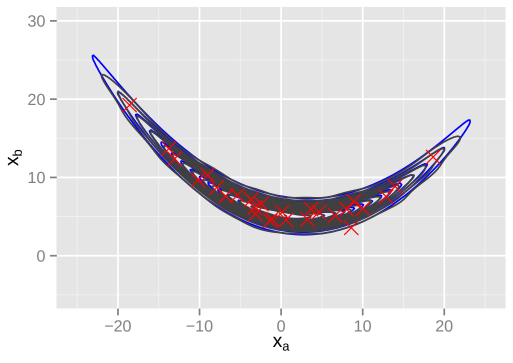
$$p_x(x) = N(x_a|\mu = 0, \sigma^2 = 10^2)N(x_b|\mu = 0.03(x_a - 3)^2 + 5, \sigma^2 = 1^2). \tag{5.16}$$

Both approximations are run with the maximum number of components $n_{c;max} = 50$. $\widetilde{r}_{o;x}(\cdot)$ stops with $n_c = 11$ components while $\widetilde{r}_{m;x}(\cdot)$ has $n_c = 27$ components. The contours of the standardised densities are shown in Figure 5.4. Clearly, the modified version has a better capture of the non-linear dependency.

Figure 5.4 shows the variable correlation but not the approximation error. So, to visually compare two approximations, $r_x(\cdot)$ is evaluated in a grid and then sorted in decreasing order of $r_x(\cdot)$. The approximated $\widetilde{r}_{\cdot;x}(\cdot)$ is sorted by the same order. Both of them are then plotted in Figure 5.5. We also calculate the statistic of Equation (5.12) for two approximations: $s(r_x, \widetilde{r}_{o;x}) = 0.424$ and $s(r_x, \widetilde{r}_{m;x}) = 0.078$. Running times in R language are included in the standardized density plots in this section.

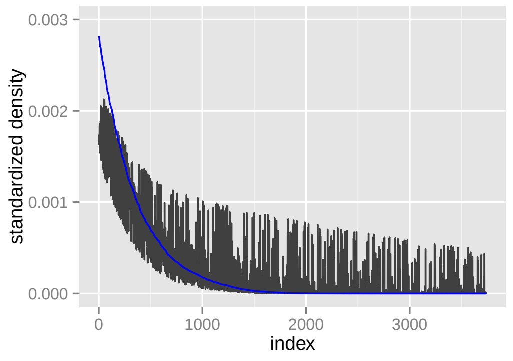


(a) iterLap

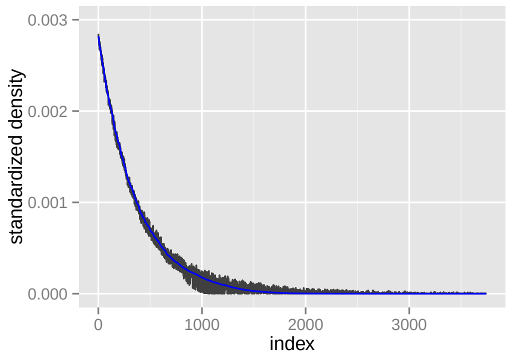


(b) mod-iterLap

Fig. 5.4: Example 6: the blue and black contours are the target standardised density $r_x(\cdot)$ and the approximated standardised density $\widetilde{r}_{\cdot;x}(\cdot)$ respectively. The red crosses are the component means.



(a) iterLap: 0.222 seconds



(b) mod-iterLap: 1.142 seconds

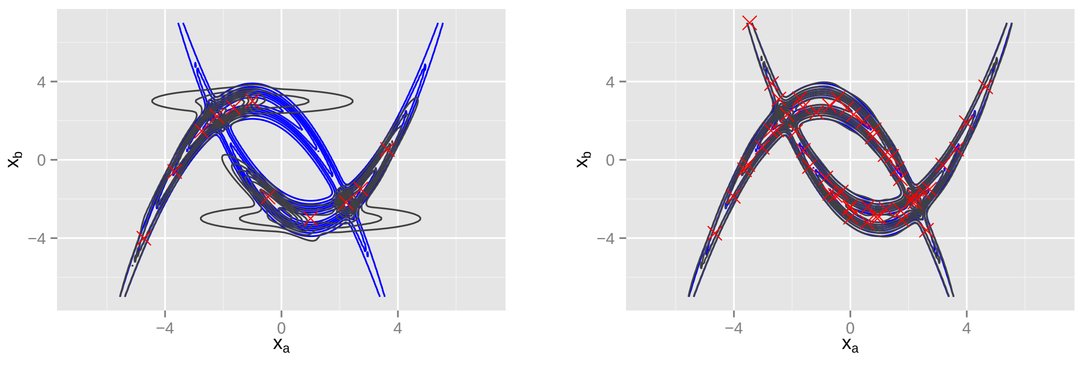
Fig. 5.5: Example 6: the plots of the ordered standardised densities. The blue and black curves are $r_x(\cdot)$ and $\widetilde{r}_{\cdot;x}(\cdot)$ respectively.

The next example is for testing a density with non-linearity and multi-modality.

Example 7 Define a target density $p_x(\cdot)$ on $x = (x_a, x_b)$:

$$p_x(x) = 0.5N(x_a|\mu = -1, \sigma^2 = 6)N(x_b|\mu = -0.5(x_a + 1)^2 + 3, \sigma^2 = 2) \qquad (5.17)$$
$$+ 0.5N(x_a|\mu = 1, \sigma^2 = 6)N(x_b|\mu = 0.5(x_a - 1)^2 - 3, \sigma^2 = 2).$$

With $n_{c;max} = 100$, iterLap stops with $\widetilde{r}_{o;x}(\cdot)$ of $n_c = 12$ components while mod-iterLap has $\widetilde{r}_{m;x}(\cdot)$ with $n_c = 56$ components. The contour plots and ordered standardised density plots are shown in Figure 5.6 and Figure 5.7. $s(r_x, \widetilde{r}_{o;x}) = 0.602$ and $s(r_x, \widetilde{r}_{m;x}) = 0.066$.



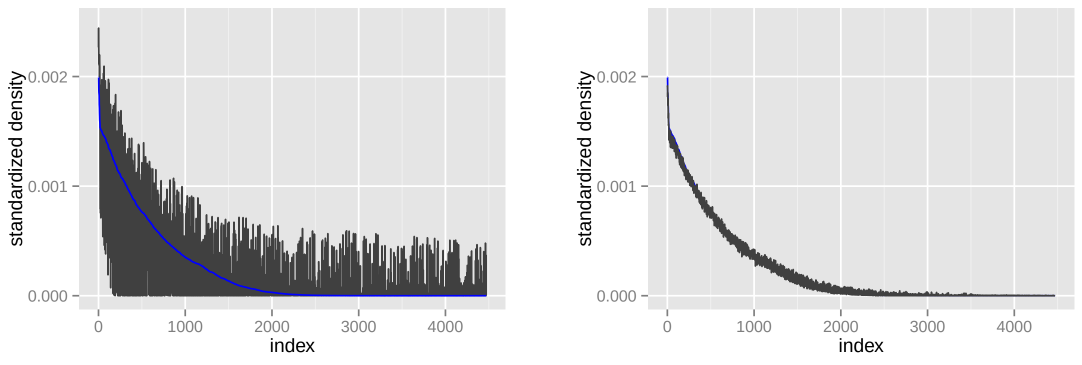
(a) iterLap (b) mod-iterLap

Fig. 5.6: Example 7: the blue and black contours are the target standardised density $r_x(\cdot)$ and the approximated standardised density $\widetilde{r}_{\cdot;x}(\cdot)$ respectively. The red crosses are the component means.

In Example 8, we try increasing the dimension of the parameter space.

Example 8 Define a target density $p_x(\cdot)$ on $x = (x_a, x_b, x_c)$ ($\dim(x_a) = \dim(x_b) = 1$, $\dim(x_c) = 4$):

$$p_x(x) = N(x_a|\mu_a, \Sigma_a = \sigma_a^2.I) \qquad (5.18)$$
$$N(x_b|\mu_b = A(x_a - \mu_a) + b, \Sigma_b = \sigma_b^2.I)$$
$$N(x_c|\mu_c = C(x_1 - \mu_a, x_b - \mu_b)^2, \Sigma_c = \sigma_c^2.I),$$



(a) iterLap: 0.300 seconds (b) mod-iterLap: 5.090 seconds

Fig. 5.7: Example 7: the plots of the ordered standardised densities. The blue and black curves are $r_x(\cdot)$ and $\widetilde{r}_{\cdot;x}(\cdot)$ respectively.

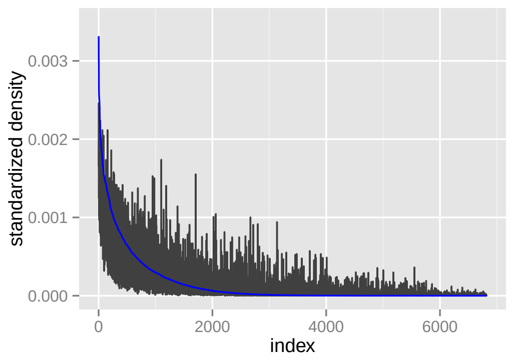
with:

$$\begin{aligned} \mu_a &= -0.5, & \sigma_a^2 &= 6.0, \\ A &= -2.0, & b &= -1.0, \\ \sigma_b^2 &= 0.2, & C &= \begin{pmatrix} 0.9 & 0.3 \\ -0.3 & -1.1 \\ -0.5 & -0.6 \\ 0.3 & 0.2 \end{pmatrix}, \\ \sigma_c^2 &= (0.6, 0.7, 0.8, 0.9)/3. \end{aligned} \tag{5.19}$$

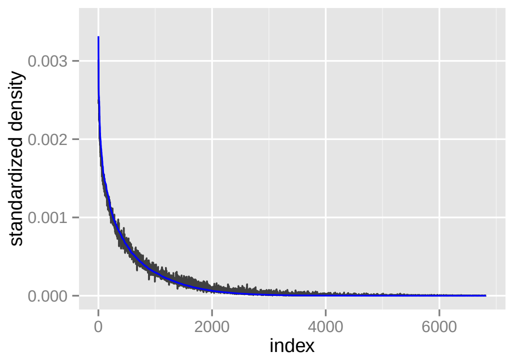
Notice that there is a linear dependency of x_a, x_b and a non-linear dependency of x_a, x_b and x_c. The conditional variances σ_b^2 and σ_c^2 define the dependency strength. As it is impossible to visualise this density, we only show the ordered standardised densities in Figure 5.8. With $n_{c;max} = 200$, $\widetilde{r}_{o;x}(\cdot)$ has $n_c = 17$ components and $\widetilde{r}_{m;x}(\cdot)$ has $n_c = 73$ components. $s(r_x, \widetilde{r}_{o;x}) = 0.733$ and $s(r_x, \widetilde{r}_{m;x}) = 0.115$.

So, the functional approximation method still works with $\dim(x) = 6$. We further increase the dimension to see its performance with Example 9.

Example 9 The same target density form in Equation (5.18) is used with $\dim(x_a) =$



(a) iterLap: 1.480 seconds



(b) mod-iterLap: 122.934 seconds

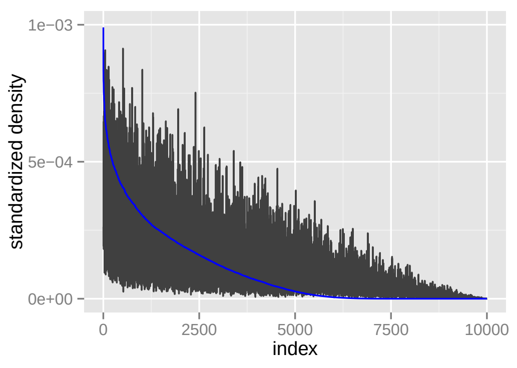
Fig. 5.8: Example 8: the plots of the ordered standardised densities. The blue and black curves are $r_x(\cdot)$ and $\widetilde{r}_{\cdot;x}(\cdot)$ respectively.

$\dim(x_b) = 2$, $\dim(x_c) = 5$ and following parameters:

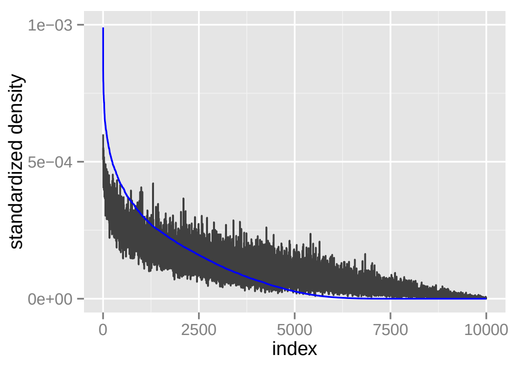
$$\begin{aligned}
\mu_a &= (-0.5, -1.0), & \sigma_a^2 &= (6.0, 7.0), \\
A &= \begin{pmatrix} 0.5 & -1.2 \\ -2.9 & -1.3 \end{pmatrix}, & b &= (-1.0, -1.5), \\
\sigma_b^2 &= (0.2, 0.3), & C &= \begin{pmatrix} 0.9 & -1.3 & -0.3 & 0.8 \\ -0.7 & 0.8 & -0.1 & 0.6 \\ 0.7 & -0.6 & 1.4 & 1.5 \\ 1.2 & -1.2 & 0.3 & 0.0 \\ 1.3 & 1.4 & 1.4 & 0.0 \end{pmatrix}, \\
\sigma_c^2 &= (0.8, 0.9, 1.0, 1.1, 1.2)/4.
\end{aligned} \tag{5.20}$$

The iterLap and mod-iterLap approximations are first run with $n_{c;max} = 200$. $\widetilde{r}_{o;x}(\cdot)$ and $\widetilde{r}_{m,a;x}(\cdot)$ have $n_c = 20$ and $n_c = 50$ components respectively and are plotted in Figures 5.9(a) and 5.9(b). Even though $\widetilde{r}_{m,a;x}(\cdot)$ is better, it may not be very satisfying. Hence, we run mod-iterLap with more components and obtain $\widetilde{r}_{m,b;x}(\cdot)$ with $n_c = 237$ components and $\widetilde{r}_{m,c;x}(\cdot)$ with $n_c = 345$ components in Figures 5.9(c) and 5.9(d). The approximations do get better with $s(r_x, \widetilde{r}_{o;x}) = 0.763$, $s(r_x, \widetilde{r}_{m,a;x}) = 0.522$, $s(r_x, \widetilde{r}_{m,b;x}) = 0.295$ and $s(r_x, \widetilde{r}_{m,c;x}) = 0.244$ but approximation errors are not as good as ones of previous examples.

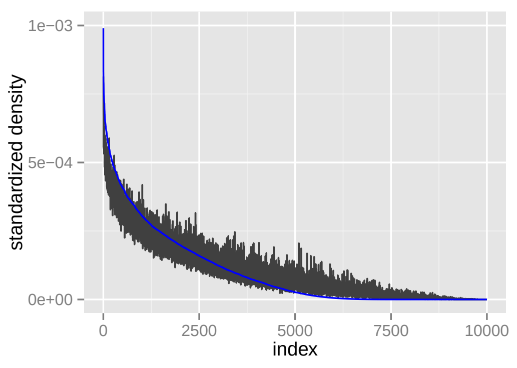
So, like many other methods, iterLap suffers from the curse of dimensionality, especially when there is non-linear dependency between many variables.



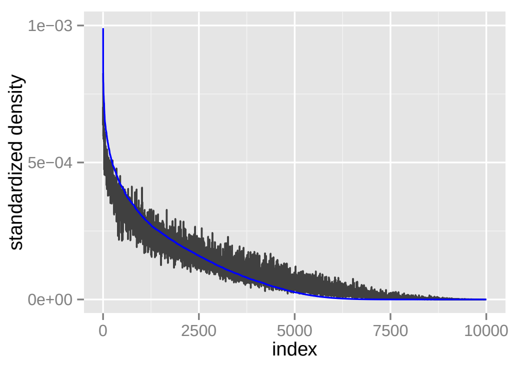
(a) iterLap: $n_c = 20$ (4.441 seconds)



(b) mod-iterLap: $n_c = 50$ (145.620 seconds)



(c) mod-iterLap: $n_c = 237$ (1.5 hours)



(d) mod-iterLap: $n_c = 345$ (7.0 hours)

Fig. 5.9: Example 9: the plots of the ordered standardised densities. The blue and black curves are $r_x(\cdot)$ and $\widetilde{r}_{\cdot;x}(\cdot)$ respectively.

In the last example, we will see how iterLap works with a non-normal noise and a constrained variable space. When used directly on such a constrained space, iterLap may have numerical issues in the optimisation and the Hessian evaluation at locations along the constrained border. Hence, it is better to transform a constrained space to a non-constrained space.

Example 10 Consider a DLM:

$$y_t = x_t + v_t \qquad (t = 1:n), \tag{5.21}$$

$$x_t = x_{t-1} + u_t \qquad (t = 2:n), \tag{5.22}$$

where $u_t \sim N(0, \sigma_u^2 = \lambda_u^{-1})$, $v_t \sim N(0, \sigma_v^2 = \lambda_v^{-1})$. The priors for precision parameters are $\lambda_u \sim Gamma(a = 1, b = 0.5)$, $\lambda_v \sim Gamma(c = 1, d = 0.5)$.

Notice that $x = x_{1:n}$ is an intrinsic Gaussian distribution with a precision matrix:

$$Q_x = \lambda_u \begin{pmatrix} 1 & -1 & 0 & 0 & \dots \\ -1 & 2 & -1 & 0 & \dots \\ 0 & -1 & 2 & -1 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix}. \tag{5.23}$$

One hundred data points $y_{1:n}$ ($n = 100$) are generated from Equation (5.21) with $\lambda_u^\star = 1/2^2$, $\lambda_v^\star = 1/1^2$. The joint posterior of $(\lambda_u, \lambda_v, x)$ is:

$$\begin{aligned} p_{\lambda_u,\lambda_v,x|y}(\lambda_u, \lambda_v, x) \propto\ & Gamma(\lambda_u|a, b)Gamma(\lambda_v|c, d) \\ & N(x|0, Q_x)N(y|x, Q_y = \lambda_v I_n). \end{aligned} \tag{5.24}$$

It can be seen that the full conditional posterior of x is:

$$p_{x|\lambda_u,\lambda_v,y}(x) = N(x|\mu'_x, Q'_x), \tag{5.25}$$

with:

$$Q'_x = Q_x + Q_y, \tag{5.26}$$

$$Q'_x\mu'_x = Q_y y. \tag{5.27}$$

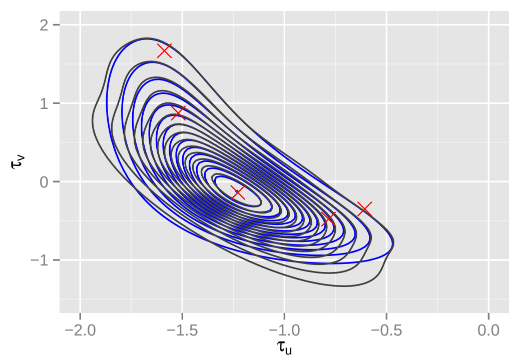
Using Equation (5.25) to marginalise Equation (5.24) with respect to x, we can obtain the marginal density $p_{\lambda_u,\lambda_v|y}(\lambda_u, \lambda_v)$:

$$\begin{aligned} p_{\lambda_u,\lambda_v|y}(\lambda_u, \lambda_v) &\propto q_{\lambda_u,\lambda_v|y}(\lambda_u, \lambda_v) \\ &= Gamma(\lambda_u|a, b)Gamma(\lambda_v|c, d) \\ &\quad (2\pi)^{-(n-1)/2}\lambda_u^{(n-1)/2}|Q'_x|^{-1/2}|Q_y|^{1/2} \\ &\quad \exp\left[\frac{-y^T Q_y y + {\mu'_x}^T Q'_x \mu'_x}{2}\right]. \end{aligned} \tag{5.28}$$

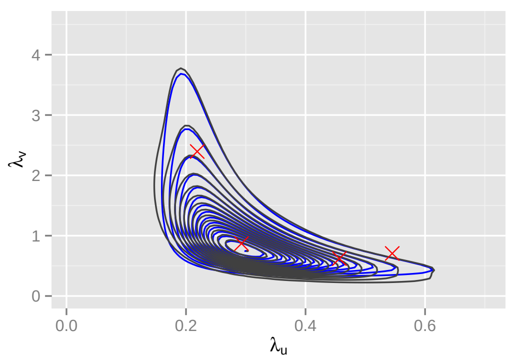
The log transform is used on (λ_u, λ_v) to get $\tau_u = \log(\lambda_u)$ and $\tau_v = \log(\lambda_v)$, which are non-constrained variables. The corresponding non-normalised marginal density $q_{\tau_u,\tau_v|y}(\tau_u, \tau_v)$ is:

$$q_{\tau_u,\tau_v|y}(\tau_u, \tau_v) = q_{\lambda_u,\lambda_v|y}(\lambda_u, \lambda_v)\lambda_u\lambda_v, \tag{5.29}$$

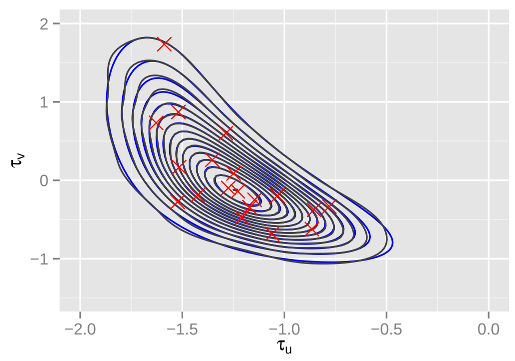
which is then approximated by iterLap. Finally, the approximated density $\widetilde{q}_{\tau_u,\tau_v|y}(\tau_u, \tau_v)$ is converted back to $\widetilde{q}_{\lambda_u,\lambda_v|y}(\lambda_u, \lambda_v)$ by Equation (5.29).



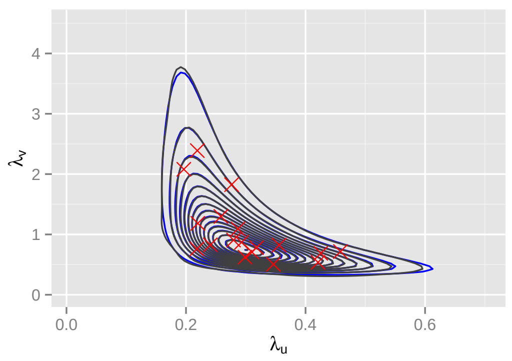
(a) iterLap: (τ_u, τ_v)



(b) iterLap: (λ_u, λ_v)



(c) mod-iterLap: (τ_u, τ_v)



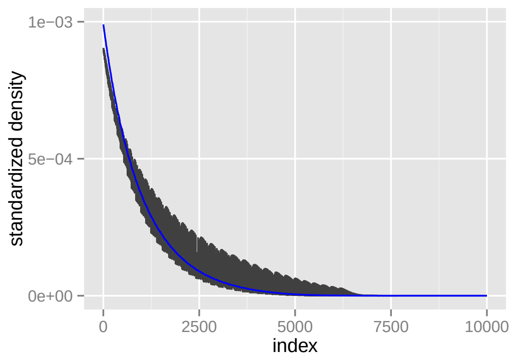
(d) mod-iterLap: (λ_u, λ_v)

Fig. 5.10: Example 10: the blue and black contours are the target standardised density r and the approximated standardised density $\widetilde{r}$ respectively. The red crosses are the iterLap component means (in the parametrisation (τ_u, τ_v) or the corresponding transform (λ_u, λ_v) of component means (τ_u, τ_v). iterLap has $n_c = 6$ components and mod-iterLap has $n_c = 19$.

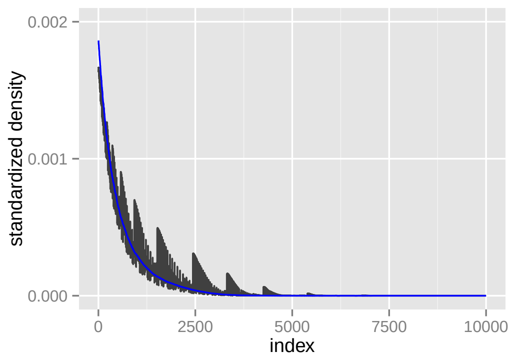
Two versions, iterLap and mod-iterLap, are run with $n_{c;max} = 30$. The contour plots of standardised densities of (τ_u, τ_v) and (λ_u, λ_v) are shown in Figure 5.10

The ordered standardised densities are plotted in Figure 5.11. In the parametrisation (τ_u, τ_v), $s(r, \widetilde{r}_o) = 0.125$, $s(r, \widetilde{r}_m) = 0.03$ while in the parametrisation (λ_u, λ_v), $s(r, \widetilde{r}_o) = 0.135$, $s(r, \widetilde{r}_m) = 0.032$.

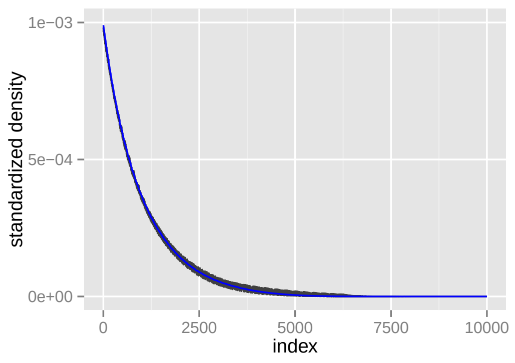
Instead of transforming the densities back and forth between constrained space and non-constrained space, which involves the evaluation of a Jacobian, it may be more practical to work directly on the non-constrained space in some cases, e.g. specify the prior and approximate the posterior on (τ_u, τ_v) in Example 10.



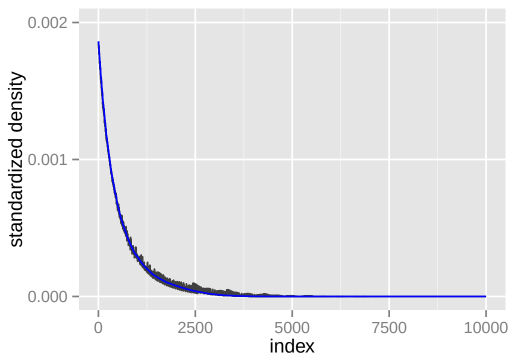
(a) iterLap: (τ_u, τ_v) (0.498 seconds)



(b) iterLap: (λ_u, λ_v) (0.498 seconds)



(c) mod-iterLap: (τ_u, τ_v) (5.024 seconds)



(d) mod-iterLap: (λ_u, λ_v) (5.024 seconds)

Fig. 5.11: Example 10: the plots of the ordered standardised densities. The blue and black curves are $r_x(\cdot)$ and $\widetilde{r}_{\cdot;x}(\cdot)$ respectively.

In all examples, mod-iterLap achieves better performance with longer running time. This computation cost is reasonable as the mod-iterLap add more components to correct the approximation without getting stuck like the original iterLap. The more the number of components, the longer the running time. In practice, the trade-off between correctness and running time can be controlled by the maximum number of components $n_{c;max}$. Another point is that the code of all these examples, iterLap, mod-iterLap is written in R. Hence, the running time should improve significantly if the code is ported to C language.

This concludes the discussion and comparison of functional approximation. In the next section, we will apply iterLap to sequential inference.

5.2 Sequential inference with iterLap

A sequential inference method for both the state vector and unknown parameters is proposed in this section. As we have seen in Chapter 2, the degeneracy problem is caused by a mismatched sampling proposal and can be especially severe in the sequential context. Hence, in this section, the iterLap method is used in each iteration to provide a continuous approximation of the filtering distribution, avoiding the degeneracy of sequential inference.

5.2.1 The sequential inference algorithm

We consider a sub-class of the DM in which the state vector x_t follows a linear transformation:

$$y_t \sim p_{y_t|x_t,\varphi}(\cdot), \tag{5.30}$$

$$x_t = Gx_{t-1} + h + u_t, \tag{5.31}$$

where $u_t \overset{iid}{\sim} N(0, \Sigma_u = Q_u^{-1})$ and φ is a vector including all unknown parameters (G, h and Q_u can all be unknown parameters). The prior of this model is defined by $p_{\varphi,x_1}(\cdot)$ with a starting state x_1.

Assume that up to time t, there is an approximation of the density $p_{\varphi,x_t|y_{1:(t-1)}}(\cdot) \approx \widetilde{p}_{\varphi,x_t|y_{1:(t-1)}}(\cdot)$. At $t = 1$, the above density is the predefined prior.

The first step is to update from $\widetilde{p}_{\varphi,x_t|y_{1:(t-1)}}(\cdot)$ to $\widetilde{p}_{\varphi,x_t|y_{1:t}}(\cdot)$ with a new data point y_t. The mixture approximation $\widetilde{p}_{\varphi,x_t|y_{1:t}}(\cdot)$ is obtained by using iterLap of the previous section:

$$\begin{aligned} \widetilde{p}_{\varphi,x_t|y_{1:t}}(\varphi, x_t) &\propto \widetilde{p}_{\varphi,x_t|y_{1:(t-1)}}(\varphi, x_t) p_{y_t|x_t,\varphi}(y_t) && (5.32) \\ &\approx \sum_i w_{n;i} N_i(\varphi, x_t|\mu_{i;\varphi,x_t}, Q_{i;\varphi,x_t}) && (5.33) \\ &= \sum_i w_{n;i} \widetilde{p}_{i;\varphi,x_t|y_{1:t}}(\varphi, x_t), && (5.34) \end{aligned}$$

where $w_{n;i}$ are normalised weights $\sum_i w_{n;i} = 1$.

In the second step, we want to evolve $\widetilde{p}_{\varphi,x_t|y_{1:t}}(\cdot)$ to $\widetilde{p}_{\varphi,x_{t+1}|y_{1:t}}(\cdot)$ by the state equation Equation (5.31). Decomposing each normal component $\widetilde{p}_{i;\varphi,x_t|y_{1:t}}(\varphi, x_t)$ into $\widetilde{p}_{i;\varphi|y_{1:t}}(\varphi)$

and $\widetilde{p}_{i;x_t|\varphi,y_{1:t}}(x_t)$, we have:

$$\widetilde{p}_{i;\varphi|y_{1:t}}(\varphi) = N_i(\varphi|\mu_{i;\varphi}, Q_{i;\varphi}), \tag{5.35}$$

$$\widetilde{p}_{i;x_t|\varphi,y_{1:t}}(x_t) = N_i(x_t|\mu_{i;x_t|\varphi}, Q_{i;x_t|\varphi}), \tag{5.36}$$

where $\mu_{i;\varphi}$, $Q_{i;\varphi}$, $\mu_{i;x_t|\varphi}$ and $Q_{i;x_t|\varphi}$ can be easily evaluated by properties of the normal distribution. Combined with the linear transformation of x_{t+1}, the joint density $\widetilde{p}_{\varphi,x_t,x_{t+1}|y_{1:t}}(\cdot)$ is:

$$\begin{aligned}&\widetilde{p}_{\varphi,x_t,x_{t+1}|y_{1:t}}(\varphi, x_t, x_{t+1}) \\ &= \sum_i w_{n;i} N_i(\varphi|\mu_{i;\varphi}, Q_{i;\varphi}) N_i(x_t|\mu_{i;x_t|\varphi}, Q_{i;x_t|\varphi}) N(x_{t+1}|Gx_t + h, Q_u).\end{aligned} \tag{5.37}$$

Then, the marginal density $\widetilde{p}_{\varphi,x_{t+1}|y_{1:t}}(\cdot)$ is calculated by:

$$\begin{aligned}\widetilde{p}_{\varphi,x_{t+1}|y_{1:t}}(\varphi, x_{t+1}) &= \int \widetilde{p}_{\varphi,x_t,x_{t+1}|y_{1:t}}(\varphi, x_t, x_{t+1}) dx_t \\ &= \sum_i w_{n;i} N_i(\varphi|\mu_{i;\varphi}, Q_{i;\varphi}) \int N_i(x_t|\mu_{i;x_t|\varphi}, Q_{i;x_t|\varphi}) N(x_{t+1}|Gx_t + h, Q_u) dx_t \\ &= \sum_i w_{n;i} N_i(\varphi|\mu_{i;\varphi}, Q_{i;\varphi}) p_{i;x_{t+1}|\varphi,y_{1:t}}(x_{t+1}).\end{aligned} \tag{5.38}$$

For each component integral, we have:

$$\begin{aligned}p_{i;x_{t+1}|\varphi,y_{1:t}}(x_{t+1}) &= \int N_i(x_t|\mu_{i;x_t|\varphi}, Q_{i;x_t|\varphi}) N(x_{t+1}|Gx_t + h, Q_u) dx_t \\ &= \int (2\pi)^{-d_x} |Q_{i;x_t|\varphi})|^{1/2} |Q_u|^{1/2} \exp\left(-\frac{1}{2}\left[(x_t - \mu_{i;x_t|\varphi})^T Q_{i;x_t|\varphi}(x_t - \mu_{i;x_t|\varphi}) \right.\right. \\ &\qquad \left.\left. + (x_{t+1} - Gx_t - h)^T Q_u (x_{t+1} - Gx_t - h)\right]\right) dx_t \\ &= \int (2\pi)^{-d_x} |Q_{i;x_t|\varphi})|^{1/2} |Q_u|^{1/2} \exp\left(-\frac{1}{2}\left[x_t^T (Q_{i;x_t|\varphi} + G^T Q_u G) x_t \right.\right. \\ &\qquad \left.\left. - 2x_t^T (Q_{i;x_t|\varphi}\mu_{i;x_t|\varphi} + G^T Q_u e) + e^T Q_u e + \mu_{i;x_t|\varphi}^T Q_{i;x_t|\varphi}\mu_{i;x_t|\varphi}\right]\right) dx_t \\ &= \int (2\pi)^{-d_x} |Q_{i;x_t|\varphi})|^{1/2} |Q_u|^{1/2} \exp\left(-\frac{1}{2}\left[(x_t - \mu')^T Q' (x_t - \mu') \right.\right. \\ &\qquad \left.\left. + e^T Q_u e + \mu_{i;x_t|\varphi}^T Q_{i;x_t|\varphi}\mu_{i;x_t|\varphi} - \mu'^T Q' \mu'\right]\right) dx_t \\ &= (2\pi)^{-d_x/2} |Q_{i;x_t|\varphi})|^{1/2} |Q_u|^{1/2} |Q'|^{-1/2} \\ &\quad \exp\left(-\frac{1}{2}\left[e^T Q_u e + \mu_{i;x_t|\varphi}^T Q_{i;x_t|\varphi}\mu_{i;x_t|\varphi} - \mu'^T Q' \mu'\right]\right),\end{aligned} \tag{5.39}$$

with:

$$e = x_{t+1} - h. \tag{5.40}$$

$$Q' = Q_{i;x_t|\varphi} + G^T Q_u G. \tag{5.41}$$

$$Q'\mu' = Q_{i;x_t|\varphi}\mu_{i;x_t|\varphi} + G^T Q_u e. \tag{5.42}$$

Hence, the marginal density $\widetilde{p}_{\varphi,x_{t+1}|y_{1:t}}(\cdot)$ can be evaluated exactly.

These two steps are done repeatedly, leading to the sequential approximation $\widetilde{p}_{\varphi,x_t|y_{1:t}}(\cdot)$ for any t. The above procedure is summarised in Algorithm 13.

Algorithm 13 Sequential inference with iterLap

```
1. With a known density $\widetilde{p}_{\varphi,x_t|y_{1:(t-1)}}(\cdot)$ (that can be numerically evaluated), use iterLap to approximate $\widetilde{p}_{\varphi,x_t|y_{1:t}}(\cdot) \propto \widetilde{p}_{\varphi,x_t|y_{1:(t-1)}}(\cdot)p_{y_t|x_t,\varphi}(\cdot)$ by a mixture of normal distributions.

2. Marginalise the mixture density $\widetilde{p}_{\varphi,x_t,x_{t+1}|y_{1:t}}(\cdot) = \widetilde{p}_{\varphi,x_t|y_{1:t}}(\cdot)N(x_{t+1}|Gx_t + h, Q_u)$ analytically with respect to $x_t$ and get $\widetilde{p}_{\varphi,x_{t+1}|y_{1:t}}(\cdot)$, using Equations (5.38) and (5.39).

3. Repeat steps 1 and 2 with $t = t + 1$.
```

Notice that as the approximated joint density $\widetilde{p}_{\varphi,x_t|y_{1:t}}(\cdot)$ is a mixture of normal distributions, we can easily derive the marginal normal densities $\widetilde{p}_{\varphi_i|y_{1:t}}(\cdot)$ and $\widetilde{p}_{x_{t,i}|y_{1:t}}(\cdot)$ ($\varphi = (\varphi_1, ..., \varphi_{d_\varphi})$; $x_t = (x_{t,1}, ..., x_{t,d_x})$).

In the case that the state equation is non-linear and $x_t \sim p_{x_t|x_{t-1},\varphi}(\cdot)$, the above procedure can be altered to approximate $\widetilde{p}_{\varphi,x_t,x_{t+1}|y_{1:t}}(\cdot)$ by another mixture of normal distributions using iterLap. Then the mixture density can be marginalised by properties of the normal distribution, leading to $\widetilde{p}_{\varphi,x_{t+1}|y_{1:t}}(\cdot)$. Such a modification results in another layer of approximation error. However, in this thesis, we will not consider such a case.

5.2.2 Examples

The proposed sequential inference algorithm is now tested with the following example.

Example 11 Two thousands data points are generated by a model:

$$y_t = bx_t^2 + v_t, \tag{5.43}$$

$$x_t - 10 = a(x_{t-1} - 10) + u_t, \tag{5.44}$$

where $u_t \overset{iid}{\sim} N(0, \sigma_u^2 = \lambda_u^{-1})$ and $v_t \overset{iid}{\sim} N(0, \sigma_v^2 = \lambda_v^{-1})$. Denoting $\tau_u = \log(\lambda_u)$ and $\tau_v = \log(\lambda_v)$, the data is generated with the following parameter values:

$$\begin{aligned} a^\star &= 0.8, & b^\star &= 2.0, \\ \tau_u^\star &= \log(0.4^{-2}), & \tau_v^\star &= \log(0.2^{-2}). \end{aligned} \tag{5.45}$$

In this example, we assume that $b = b^\star$ and $\tau_v = \tau_v^\star$ are known and use sequential inference on (a, τ_u, x_t). The priors for a, τ_u and x_1 are:

$$p_a(\cdot) = N(\cdot|\mu_a = 0, \sigma_a^2 = 0.2^2), \tag{5.46}$$

$$p_{\tau_u}(\cdot) = N(\cdot|\mu_{\tau_u} = \tau_u^\star - 0.5 = 1.333, \sigma_{\tau_u}^2 = 0.5^2), \tag{5.47}$$

$$p_{x_1}(\cdot) = N(\cdot|\mu_{x_1} = 10, \sigma_{x_1}^2 = 0.5^2). \tag{5.48}$$

Algorithm 13 is used on this example and the sequential approximation of density $p_{a,\tau_u,x_t|y_{1:t}}(\cdot)$ is summarised in Figure 5.12. The sequential inference method seems to work well in this example.

We now try sequential inference on a more complex case.

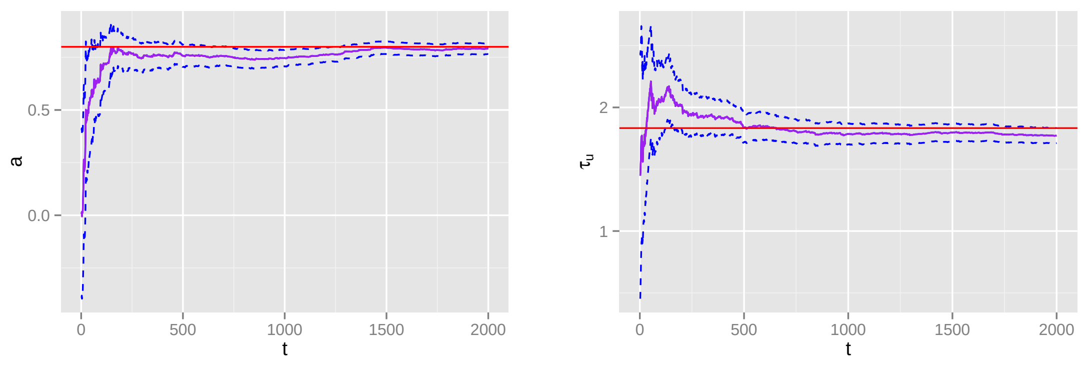
Example 12 This example uses the same model and dataset of Example 11. However, coefficient b is now unknown and sequential inference is applied on (a, b, τ_u, x_t). The priors for these parameters are:

$$p_a(\cdot) = N(\cdot|\mu_a = 0, \sigma_a^2 = 0.2^2), \tag{5.49}$$

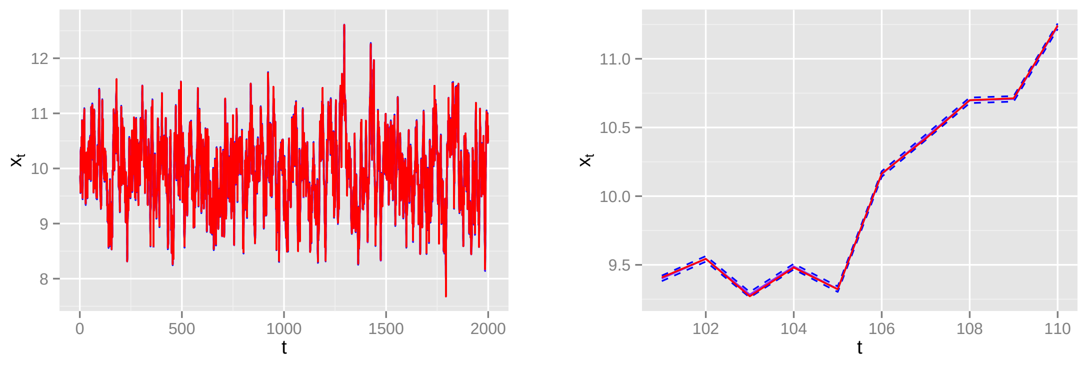
$$p_b(\cdot) = N(\cdot|\mu_b = 1, \sigma_a^2 = 0.2^2), \tag{5.50}$$

$$p_{\tau_u}(\cdot) = N(\cdot|\mu_{\tau_u} = \tau_u^\star - 0.5 = 1.333, \sigma_{\tau_u}^2 = 0.5^2), \tag{5.51}$$

$$p_{x_1}(\cdot) = N(\cdot|\mu_{x_1} = 10, \sigma_{x_1}^2 = 0.5^2). \tag{5.52}$$



(a) a (b) τ_u

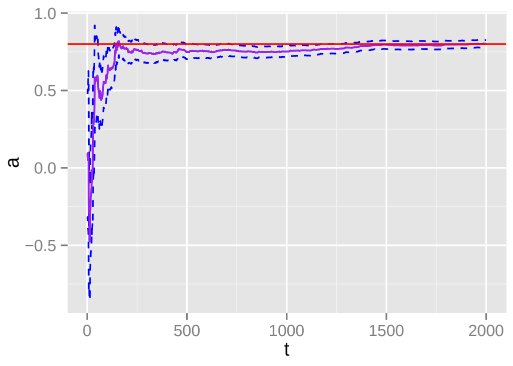


(c) x_t (d) x_t for $t = 101 : 110$

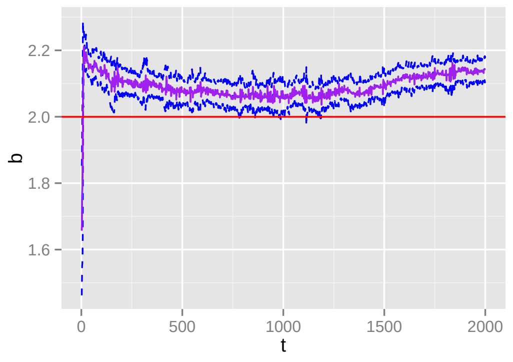
Fig. 5.12: Example 11: summary of filtering densities $p_{a|y_{1:t}}(\cdot)$, $p_{\tau_u|y_{1:t}}(\cdot)$ and $p_{x_t|y_{1:t}}(\cdot)$. The blue and purple solid lines represents the mean $\overline{\mu}$ and the mode $\overline{\xi}$ of a particular density. The blue dashed lines mark the interval $[\overline{\mu}-2\overline{\sigma}, \overline{\mu}+2\overline{\sigma}]$ with standard deviation $\overline{\sigma}$. The red lines represent the true value of the random variables. Figure 5.12(d) is an enlargement of Figure 5.12(c).

Results of $p_{a|y_{1:t}}(\cdot)$, $p_{b|y_{1:t}}(\cdot)$, $p_{\tau_u|y_{1:t}}(\cdot)$ and $p_{x_t|y_{1:t}}(\cdot)$ are shown in Figure 5.13. Inference of a and τ_u seems satisfying as the true values are contained in the intervals $[\overline{\mu} - 2\overline{\sigma}, \overline{\mu} + 2\overline{\sigma}]$. On the other hand, the method overestimates b and underestimates x_t in Figures 5.13(c) and 5.13(d). Still, with a prior of b quite far from the true value $b^\star$, the approximation method has managed to pull the density $p_{b|y_{1:t}}(\cdot)$ close enough towards the true value.

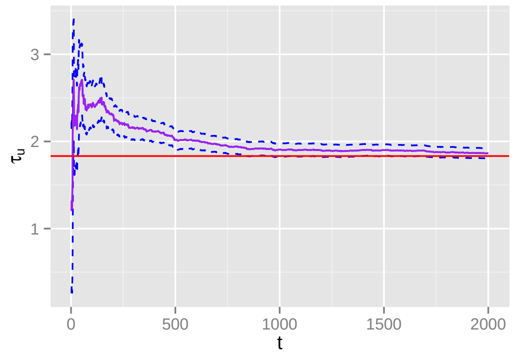
Inference of b and x_t shows us the identifiability issue in the sequential approximation approach of this model. As $y_t = bx_t^2 + v_t$ in Equation (5.43), conditioning on y_t, one can have a large estimated mean $\overline{\mu}_b$ and a small estimated mean $\overline{\mu}_{x_t}$ (as in Figures 5.13(c) and 5.13(d)) or vice versa. This identifiability problem makes it hard to estimate correctly the unknown parameters and becomes more severe with sequential approximation (as there is approximation error in each iteration). Perhaps, an offline method would yield a better estimation (however, in Section 6.3.1, we will see that even an offline estimation of the DM is sensitive to the identifiability problem). Still, by the proposed method, the sequential estimation $\overline{y}_t = \overline{\mu}_b \overline{\mu}_{x_t}^2$ is consistent with y_t and this result may be good enough in practise, depending on the application context.



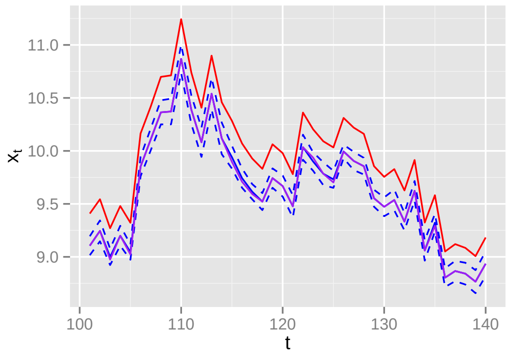
(a) a



(b) b



(c) τ_u



(d) x_t for $t = 101 : 140$

Fig. 5.13: Example 12: summary of filtering densities $p_{a|y_{1:t}}(\cdot)$, $p_{\tau_u|y_{1:t}}(\cdot)$ and $p_{x_t|y_{1:t}}(\cdot)$. The blue and purple solid lines represents the mean $\overline{\mu}$ and the mode $\overline{\xi}$ of a particular density. The blue dashed lines mark the interval $[\overline{\mu}-2\overline{\sigma}, \overline{\mu}+2\overline{\sigma}]$ with standard deviation $\overline{\sigma}$. The red lines represent the true value of the random variables.

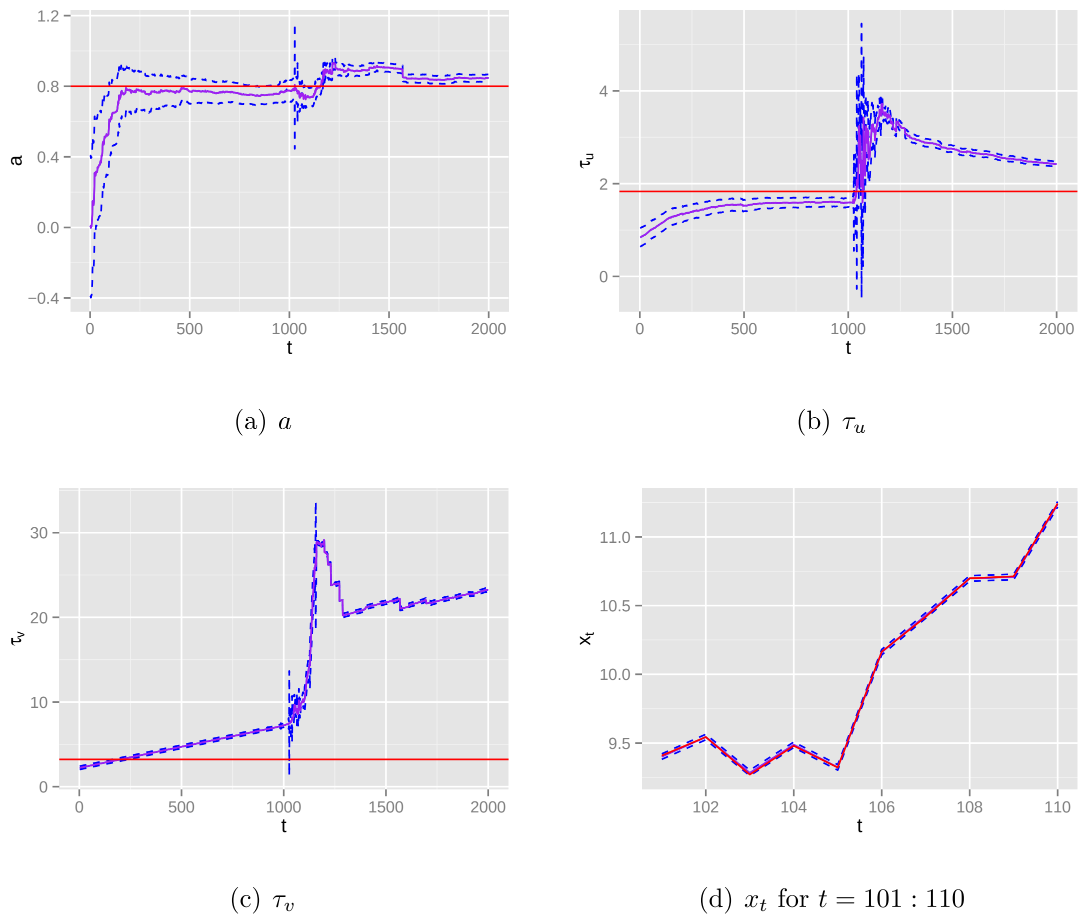
Example 13 In the last example, sequential inference is used on the unknown parameter vector (a, τ_u, τ_v, x_t). The priors for these parameters are:

$$p_a(\cdot) = N(\cdot|\mu_a = 0, \sigma_a^2 = 0.2^2), \tag{5.53}$$

$$p_{\tau_u}(\cdot) = N(\cdot|\mu_{\tau_u} = \tau_u^\star - 1.0 = 0.833, \sigma_{\tau_u}^2 = 0.1^2), \tag{5.54}$$

$$p_{\tau_v}(\cdot) = N(\cdot|\mu_{\tau_v} = \tau_v^\star - 1.0 = 2.219, \sigma_{\tau_v}^2 = 0.1^2), \tag{5.55}$$

$$p_{x_1}(\cdot) = N(\cdot|\mu_{x_1} = 10, \sigma_{x_1}^2 = 0.5^2). \tag{5.56}$$



(a) a (b) τ_u (c) τ_v (d) x_t for $t = 101 : 110$

Fig. 5.14: Example 13: summary of filtering densities $p_{a|y_{1:t}}(\cdot)$, $p_{\tau_u|y_{1:t}}(\cdot)$ and $p_{x_t|y_{1:t}}(\cdot)$. The blue and purple solid lines represents the mean $\overline{\mu}$ and the mode $\overline{\xi}$ of a particular density. The blue dashed lines mark the interval $[\overline{\mu}-2\overline{\sigma}, \overline{\mu}+2\overline{\sigma}]$ with standard deviation $\overline{\sigma}$. The red lines represent the true value of the random variables.

The results of this example are summarised in Figure 5.14. With both unknown τ_u and τ_v, the method has difficulty in parameter estimation, only managing to pull variables a and τ_u closer to the true values. Variable τ_v becomes more and more overestimated over time. When we check the sequential inference code in details, this issue seems to be coming from the optimisation step of iterLap ($p_{y_t|b,\tau_v,x_t}(y_t) = N(y_t|\mu = bx_t^2, \sigma^2 = \exp(-\tau_v))$ is increasing with respect to τ_v).

5.3 Conclusion

In this chapter, we have improved the functional approximation method iterLap and evaluated its performance with several examples. Then, based on iterLap, a new sequential inference method is proposed for both the state vector and parameters in the DM, using a continuous approximation to avoid the degeneracy issue in sampling-based sequential inference. This new method is then analysed with some simulated examples, yielding varying results. In general, the method has good performance but it is sensitive to the identifiability issue and may have trouble with estimating an unknown variance (or precision). Practically, the identifiability issue may be avoided or lessened by careful modelling and the variance estimation problem of the DM may be handled by fixing either the state noise variance or observation noise variance or coupling the two variances, e.g. $\tau_u = \alpha\tau_v$ with a predefined coefficient α. In any case, the estimation of the state vector x_t seems to be fine and compatible with the observation y_t.

The next chapter resumes the discussion of the STFF problem and applies the proposed sequential inference method for a particular sub-model.

Chapter 6

Spatial-temporal modelling of transportation networks

In this chapter, the motivation for a new spatial-temporal model of STFF is first discussed. By the works on estimating the current number of vehicles within a link (Papageorgiou and Vigos, 2008; Vigos and Papageorgiou, 2010), we assume that these variables are known (or estimated) and use them as observations in the new spatial-temporal model. The model is evaluated using data that is generated by the popular transportation micro-simulator VISSIM.

A new model is proposed, consisting of four sub-models. Each sub-model corresponds to a specific part of a traffic network.

- A link-outflow sub-model for the one-step-ahead outflow with inputs as the current inflow and the number of vehicles within a link.
- A junction sub-model that relates inflows and outflows of a particular junction, based on the latent turning rate of vehicles.
- A root sub-model for the inflows of root-links which are located at the border of a traffic network and have no upstream information.
- A link-vehicle sub-model that provides the number of vehicles by the current inflow, outflow and the previous number of vehicles in a specific link.

Specifically, the modelling and inference for the first two sub-models are provided. The sequential method proposed in the previous chapter is used for the DM of the junction sub-model, meeting the time constraint of real-time applications. Then, how inference for the last two sub-models may be done is discussed.

6.1 The spatial-temporal model

6.1.1 Motivation

The motivation of this chapter comes from the discussion of Section 4.2.5. A new model is proposed with some characteristics:

- The model focuses on modelling the mean of the traffic variables and uses simple iid random noise for the residuals.
- To solve the scalability problem, the model can be divided into several components on which the inference can be done conditionally independently.
- For the real-time requirement, sequential inference can be used on each component.
- The model should be robust to the occurrence of a traffic incident and the dependency between variables is as stable as possible.

In our opinion, using only previous flow information may not be enough for an accurate prediction, especially in the case of traffic incident. Denoting the network state information at time t by ψ_t, we want to include relevant traffic information in ψ_t, which is used to obtain the traffic flow prediction $\widehat{z}_{t+1}$.

By a stable and good modelling, accurate state information ψ_t may provide a precise one-step-prediction $\widehat{z}_{t+1}$. However, there is a mutual cause-effect relationship between the flow and the state in STFF. As we want the multi-step-ahead prediction $\widehat{z}_{(t+1):(t+h)}$, the network state $\widehat{\psi}_{(t+1):(t+h-1)}$ forecasting needs to be given too. Furthermore, to maintain the prediction accuracy of $\widehat{z}_{t+h}$, $\widehat{\psi}_{t+h-1}$ should also be as accurate as possible. Usually, the flow information is included in the network state and the flow prediction problem extends to the network state prediction.

So, in this chapter, we decide to use the number of vehicles within a link ν_t and the flow z_t as the network state. The reason is because ν_t is a good explanatory variable for z_{t+1}, which will be seen in the link-outflow sub-model of Section 6.2. Also, there

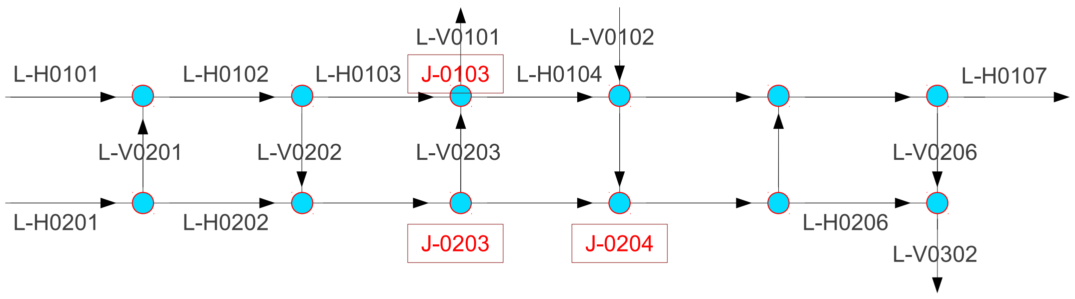


Fig. 6.1: The simulated network map: the black links are the directed road links and the blue circles represent the junctions.

is the conservation equation of vehicles in Section 3.3 to maintain the strong coupling between ν_t and z_t. At the moment, it is assumed that the number of vehicles within a link ν_t can be obtained by either a direct mean such as video detector or an estimation method in Papageorgiou and Vigos (2008) and Vigos and Papageorgiou (2010).

6.1.2 A VISSIM dataset

As the Dublin traffic datasets in Section 4.2.1 do not contain the number of vehicles within a link ν_t, we use the micro-simulator VISSIM to generate a new traffic dataset. In VISSIM, a virtual traffic environment is created where each bus, vehicle or pedestrian is simulated completely as an entity with its own goal and behaviour. A user only needs to specify the high-level information such as the virtual network map, the vehicle input volume, the traffic light sequence, etc. and VISSIM will automatically run the simulation and record the specified traffic variables. To the user (or at least to us), the random process of the VISSIM simulation is unknown.

Example 14 A one-minute dataset of 4 days is simulated by VISSIM with the network map in Figure 6.1. The links and junctions have naming codes as L-ABBCC and J-BBCC with:

- A is either H for a horizontal link or V for a vertical link.
- BB and CC are the row and column indices of a link or a junction.

For each link, the inflow $z_{1;t}$, the outflow $z_{2;t}$ and the number of vehicles ν_t are recorded as in Figure 6.2. As each link has 2 lanes, all the statistics are aggregated for the whole link.

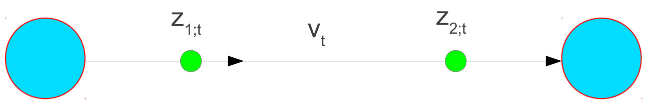


Fig. 6.2: A road link with the inflow $z_{1;t}$, the outflow $z_{2;t}$ and the number of vehicles ν_t: the blue circles are the junctions and the green circles are the detector locations for the inflow and outflow of the link.

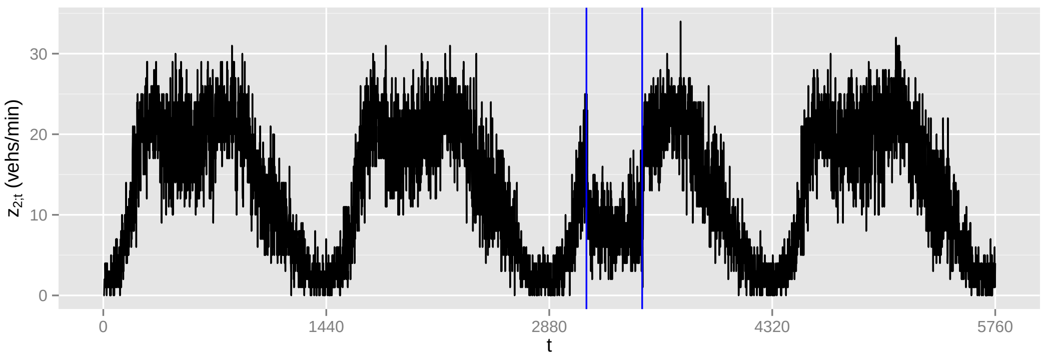


Fig. 6.3: The outflow $z_{2;t}$ of link L-H0203. The blue vertical lines mark a duration when a traffic incident occurs.

We call links L-V0102, L-H0101 and L-H0201 root links as there is no upstream information for these links in this simulated scenario. For the VISSIM simulation, the traffic volumes of these root links must be provided. To simulate a traffic incident in this dataset, the inflow of root link L-H0201 is blocked for 6 hours on the third day and this incident affects the whole traffic network consequently.

There is no restriction for the flow direction in each junction. For example, in junction J-0103, the flow of link L-H0103 can go to either link L-V0101 or L-H0104.

For the number of vehicles within a link, the true quantity $\nu_t^\star$ is obtained from VISSIM. Then, in each link, an iid error is then added to this true quantity for the observed value: $\nu_t = \nu_t^\star + e_{\nu;t}^\star$. The error is chosen so that the *relative mean square error (rmse)* of ν_t and $\nu_t^\star$ is about 15% for 1-minute dataset, according to the result in Vigos and Papageorgiou (2010) (rmse is the ratio between the standard deviation of $e_{\nu;t}^\star$ and the mean of true value). To make the simulation more realistic, the observed value ν_t is used instead of the true value $\nu_t^\star$ for statistical inference in this chapter.

.

The outflow $z_{2;t}$ of link L-H0203 is shown in Figure 6.3 (this plot is a duplicate of Figure 4.35(a)). It can be seen in the plot that the traffic pattern during the incident ($t = 3120 : 3480$) is quite different from the usual pattern.

6.1.3 The general model

Denote the individual inflow, outflow and number of vehicles of link i by $z_{1;t,i}$, $z_{2;t,i}$ and $\nu_{t,i}$. It can be seen from Figure 6.1 that the inflow of one link is the outflow of another junction. For notation purposes, the outflow vector of junction j is denoted by $\zeta_{2;t,j} = (z_{1;t,i_1}, z_{1;t,i_2}, ...)$. Similarly, the inflow vector of junction j is $\zeta_{1;t,j} = (z_{2;t,i'_1}, z_{2;t,i'_2}, ...)$. Depending on a context, the link index and junction index may be dropped for simplicity in this chapter. Also, the link index set of all root links is denoted by $\mathscr{R}$. Notice that the set of junction outflows $\zeta_{2;t,j}$ is equivalent to the set of link inflows $z_{1;t,i}$ with $i \notin \mathscr{R}$.

The general model is constructed by 4 components:

- A link-outflow sub-model for link i is defined by the density $p_{z_{2;t,i}|\Omega_{a;i},z_{1;1:(t-1),i},\nu_{1:(t-1),i}}(\cdot)$ with unknown parameter $\Omega_{a;i}$. In this thesis, only the one-step-behind network information is used for the current outflow. So, the density is simplified to $p_{z_{2;t,i}|\Omega_{a;i},z_{1;t-1,i},\nu_{t-1,i}}(\cdot)$.

- For junction j, the relationship between the junction inflow $\zeta_{1;t,j}$ and the junction outflow $\zeta_{2;t,j}$ is defined by:

$$p_{\zeta_{2;t,j},\chi_{b;t,j}|\Omega_{b;j},D_{b;t-1,j},\zeta_{1;t,j}}(\cdot) = p_{\zeta_{2;t,j}|\Omega_{b;j},\chi_{b;t,j},\zeta_{1;t,j}}(\cdot)p_{\chi_{b;t,j}|\Omega_{b;j},D_{b;t-1,j}}(\cdot), \tag{6.1}$$

  with unknown parameter $\Omega_{b;j}$ and latent variable $\chi_{b;t,j}$. $D_{b;t-1,j}$ is the shorthand for $(\zeta_{1;1:(t-1),j}, \zeta_{2;1:(t-1),j})$. This sub-model is actually a DM and will be clarified in Section 6.3.

- The inflow of a root link $i \in \mathscr{R}$ can be modelled as a DM:

$$p_{z_{1;t,i},\chi_{c;t,i}|\Omega_{c;i},z_{1;1:(t-1),i}}(\cdot) = p_{z_{1;t,i}|\Omega_{c;i},\chi_{c;t,i}}(\cdot)p_{\chi_{c;t,i}|\Omega_{c;i},z_{1;1:(t-1),i}}(\cdot), \tag{6.2}$$

  with unknown parameter $\Omega_{c;i}$ and latent variable $\chi_{c;t,i}$.

- The number of vehicles within link i follows the conservation equation:

$$\nu_{t,i} = \nu_{t-1,i} + z_{1;t,i} - z_{2;t,i} + e_{\nu;t,i}, \tag{6.3}$$

where $e_{\nu;t,i} \overset{iid}{\sim} N(0, \sigma^2_{\nu;i})$. Conditioning on $(\nu_{t-1,i}, z_{1;t,i}, z_{2;t,i})$, this is a simple linear model with unknown parameter $\Omega_{d;i} = \sigma^2_{\nu;i}$.

An independent prior will be used for the above model:

$$p_{\Omega_a,\Omega_b,\Omega_c,\Omega_d}(\cdot) = \prod_i p_{\Omega_{a;i}}(\cdot) \prod_j p_{\Omega_{b;j}}(\cdot) \prod_{i' \in \mathscr{R}} p_{\Omega_{c;i'}}(\cdot) \prod_{i''} p_{\Omega_{d;i''}}(\cdot) \tag{6.4}$$

The only common variables among 4 sub-models are the observations of flows and the number of vehicles. So, with the independent prior, we have:

$$p_{\Omega_{a;i}|z_{1;1:t,i},z_{2;1:t,i},\nu_{1:t,i},\ldots}(\cdot) = p_{\Omega_{a;i}|z_{1;1:t,i},z_{2;1:t,i},\nu_{1;1:t,i}}(\cdot), \tag{6.5}$$

$$p_{\Omega_{b;j},\chi_{b;1:t,j}|\zeta_{1;1:t,i},\zeta_{2;1:t,i},\ldots}(\cdot) = p_{\Omega_{b;j},\chi_{b;1:t,j}|\zeta_{1;1:t,i},\zeta_{2;1:t,i}}(\cdot), \tag{6.6}$$

$$p_{\Omega_{c;i},\chi_{c;1:t,i}|\zeta_{1;1:t,i},\ldots}(\cdot) = p_{\Omega_{c;i},\chi_{c;1:t,i}|\zeta_{1;1:t,i}}(\cdot) \quad i \in \mathscr{R}, \tag{6.7}$$

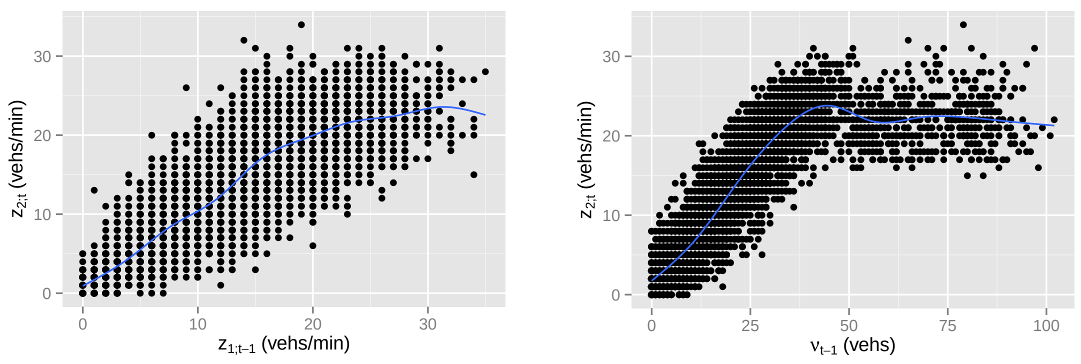
$$p_{\Omega_{d;i}|z_{1;1:t,i},z_{2;1:t,i},\nu_{1:t,i},\ldots}(\cdot) = p_{\Omega_{d;i}|z_{1;1:t,i},z_{2;1:t,i},\nu_{1;1:t,i}}(\cdot). \tag{6.8}$$

This property of conditional independence allows separate inference (estimation and filtering of parameters and latent variables) for each link, junction and sub-model, relieving the scalability problem in an expanded traffic network.

For the forecasting, the link-outflow sub-model allows the one-step-ahead predictions of all link outflows. Combining the link-outflow and the junction sub-models gives us the one-step-ahead inflow predictions of all non-root links $i \notin \mathscr{R}$. The standalone root sub-model is capable of multi-step-ahead inflow predictions of all root links $i \in \mathscr{R}$. The link-vehicle sub-model connects all together, giving the multi-step-ahead predictions for the whole network. The first two sub-models will now be examined in more details.

6.2 The link-outflow sub-model

In this section, we analyse the link outflow $z_{2;t}$ based on the one-step-behind observations $z_{1;t-1}$ and ν_{t-1}. The scatter plots of these quantities for link L-H0203 are shown in Figure 6.4. From the figure, it is seen that the mean of $z_{2;t}$ is reasonably linear against $z_{1;t-1}$ but is better modelled by a piecewise polynomial function of ν_{t-1}. Also, the horizontal variation of $z_{2;t}$ with respect to $z_{1;t-1}$ (analogous to the conditional variance) is larger than the variation of the pair $(\nu_{t-1}, z_{2;t})$. With the smaller variation, using ν_{t-1} should yield better prediction of $z_{2;t}$.

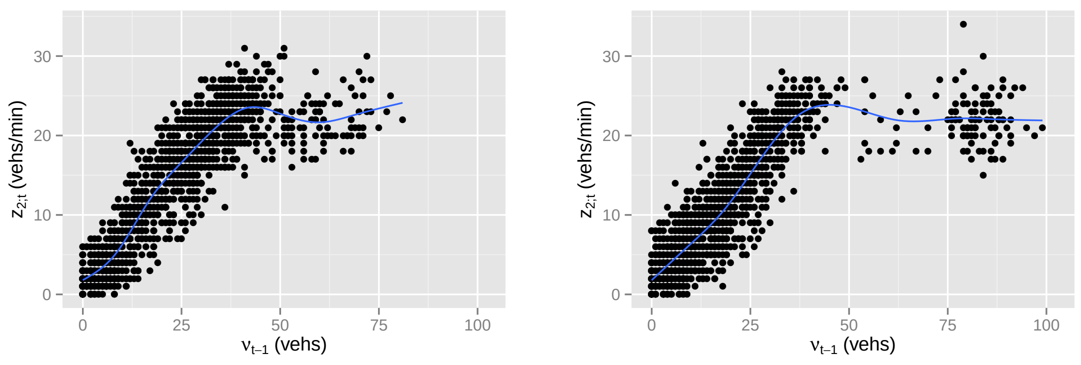


(a) (b)

Fig. 6.4: Link L-H0203: the scatter plots of $(z_{1;t-1}, z_{2;t})$ and $(\nu_{t-1}, z_{2;t})$ for all days. The blue curves are the smoothing means provided by R function geom_smooth.

In Figure 6.4, the dependency between $z_{2;t}$ and ν_{t-1} changes significantly around at $\nu_{t-1} = 45$. This shifting corresponds to the congestion time when the link is full of vehicles (high values of ν_{t-1}).

These dependencies seem to be independent on the day. The scatter plots of $(\nu_{t-1}, z_{2;t})$ on days 2 and 3 are shown in Figure 6.5. As there is a traffic incident on day 3, these plots implies that the dependency of $(\nu_{t-1}, z_{2;t})$ is robust to the incident. The plots of $(z_{1;t-1}, z_{2;t})$ of separate days are not shown, but also exhibit a stable dependency with respect to the incident.



(a) Day 2 (b) Day 3 with a traffic incident

Fig. 6.5: Link L-H0203: the scatter plots of $(\nu_{t-1}, z_{2;t})$ of separate days. The blue curves are the smoothing means provided by R function geom_smooth.

Linear regression with smoothing is now tried with these variables. The following models are fitted with the first 2 days data, using R linear regression function lm:

$$\text{LO-D1-I} : z_{2;t} = \beta_0 + \beta_{z;1} z_{1;t-1} + \sum_i \beta_{z;i+1}(z_{1;t-1} - b_i)_+ + e_{z;t}, \tag{6.9}$$

$$\text{LO-D1-V} : z_{2;t} = \beta_0 + \beta_{\nu;1} \nu_{t-1} + \sum_j \beta_{\nu;j+1}(\nu_{t-1} - c_j)_+ + e_{z;t}, \tag{6.10}$$

$$\begin{aligned}\text{LO-D1-IV} : z_{2;t} = {} & \beta_0 + \beta_{z,1} z_{1;t-1} + \sum_i \beta_{z;i+1}(z_{1;t-1} - b_i)_+ + \\ & \beta_{\nu;1} \nu_{t-1} + \sum_j \beta_{\nu;j+1}(\nu_{t-1} - c_j)_+ + e_{z;t},\end{aligned} \tag{6.11}$$

$$\text{LO-D2-I} : z_{2;t} = \beta_0 + \beta_{z;1} z_{1;t-1} + \beta_{z;2} z_{1;t-1}^2 + \sum_i \beta_{z;i+2}(z_{1;t-1} - b_i)_+^2 + e_{z;t}, \tag{6.12}$$

$$\text{LO-D2-V} : z_{2;t} = \beta_0 + \beta_{\nu;1} \nu_{t-1} + \beta_{\nu;2} \nu_{t-1}^2 + \sum_j \beta_{\nu;i+2}(\nu_{t-1} - c_j)_+^2 + e_{z;t}, \tag{6.13}$$

$$\begin{aligned}\text{LO-D2-IV} : z_{2;t} = {} & \beta_0 + \beta_{z;1} z_{1;t-1} + \beta_{z;2} z_{1;t-1}^2 + \sum_i \beta_{z;i+2}(z_{1;t-1} - b_i)_+^2 + \\ & \beta_{\nu;1} \nu_{t-1} + \beta_{\nu;2} \nu_{t-1}^2 + \sum_j \beta_{\nu;i+2}(\nu_{t-1} - c_j)_+^2 + e_{z;t},\end{aligned} \tag{6.14}$$

where the knots are manually chosen with $b = (5, 15, 25)$ and $c = (35, 45, 55)$; $e_{z;t}$ is iid normal random noise; the codes on the rightmost are used to denote the linear models. After these models are fitted, we obtain the one-step-ahead prediction $\widehat{z}_{2;t|t-1}$ and calculate the mean square error. The mean square error $s_{mse;1}$ is calculated from the last 2 days' data while $s_{mse;2}$ is only for the duration during the traffic incident ($t = 3120 : 3480$). These errors are summarised in Table 6.1. The one-step-ahead prediction of model LO-D1-IV for the last 2-day outflow of link L-H0203 is shown in Figure 6.6(a).

The same procedure is used on the outflow of link L-H0104. The resulting mean square error is in Table 6.2 and the one-step-ahead prediction is plotted in Figure 6.6(b).

So, even though the models are fitted with first 2-day data, its prediction is still useful during the time of the traffic incident. Furthermore, using the number of vehicles ν_{t-1} yields better prediction as we have expected. The difference in using polynomial splines of degree 1 (D1) and degree 2 (D2) is quite small in this scenario.

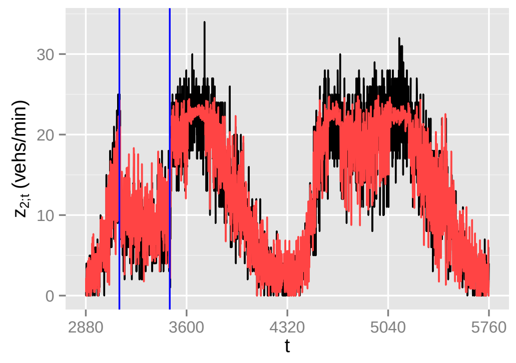
Bayesian sequential inference of a linear model is done quite simply by using the conjugate prior, normal and inverse gamma distributions, on the linear coefficient and the variance parameters (Bishop, 2006,chap. 3).

Table 6.1: The mean square error of one-step-ahead prediction for the link-outflow sub-model (Link L-H0203): $s_{mse;1}$ is for the last 2-day data and $s_{mse;2}$ is for the duration during the traffic incident.

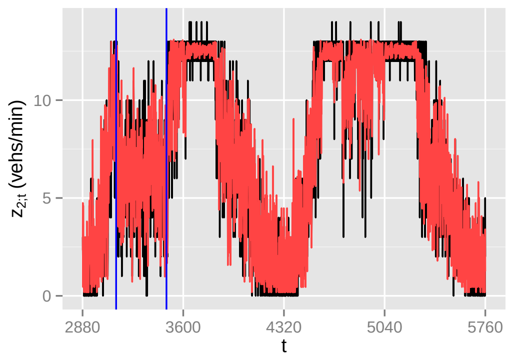
	LO-D1-I	LO-D1-V	LO-D1-IV	LO-D2-I	LO-D2-V	LO-D2-IV
$s_{mse;1}$	13.4687	9.9678	9.1324	13.4687	10.0524	8.9887
$s_{mse;2}$	19.4746	14.4983	13.9870	19.4746	14.7115	14.2640

Table 6.2: The mean square error of one-step-ahead prediction for the link-outflow sub-model (Link L-H0104): $s_{mse;1}$ is for the last 2-day data and $s_{mse;2}$ is for the duration during the traffic incident.

	LO-D1-I	LO-D1-V	LO-D1-IV	LO-D2-I	LO-D2-V	LO-D2-IV
$s_{mse;1}$	6.3415	3.1562	3.0876	6.3415	3.1027	3.0008
$s_{mse;2}$	9.7636	4.5032	4.5551	9.7636	4.5132	4.7681



(a) Link L-H0203



(b) Link L-H0104

Fig. 6.6: The one-step-ahead prediction of link outflow for the last two days: the black and red curves are the true outflow and the prediction respectively. The blue vertical lines mark a duration when a traffic incident occurs.

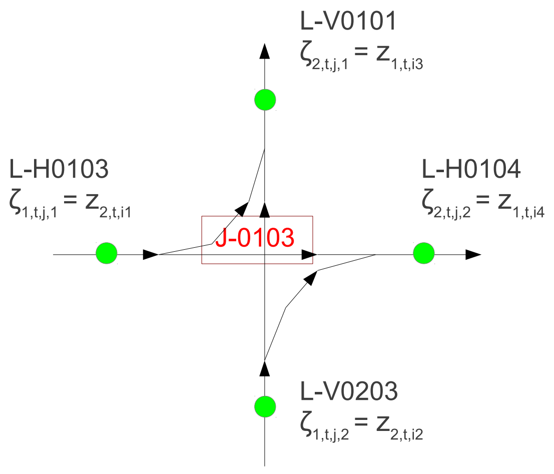


Fig. 6.7: The junction J-0103 with the inflow $\zeta_{1;t}$, the outflow $\zeta_{2;t}$: the green circles are the detector locations for the flows.

6.3 The junction sub-model

We now examine the junction sub-model which relates $\zeta_{2;t}$ ($d_{\zeta;2} = \dim(\zeta_{2;t})$) and $\zeta_{1;t}$ ($d_{\zeta;1} = \dim(\zeta_{1;t})$) (or equivalent $z_{1;t}$ and $z_{2;t}$). An enlarged map for the junction J-0103 is shown in Figure 6.7. We propose using the following DM model for a junction (the junction index is dropped):

$$\zeta_{2;t} = A_t(x_t)\zeta_{1;t} + v_t, \tag{6.15}$$

$$x_t = x_{t-1} + u_t, \tag{6.16}$$

where $v_t \overset{iid}{\sim} N(0, \sigma_v^2 = \lambda_v^{-1})$ (a multivariate normal noise with a diagonal variance matrix $\Sigma_v = \sigma_v^2 I$); $u_t \overset{iid}{\sim} N(0, \Sigma_u = Q_u^{-1})$; x_t is a state vector of length d_x. The $d_{\zeta;2} \times d_{\zeta;1}$ matrix A_t with column-sums 1 represents the turning rate from the inflow $\zeta_{1;t}$ to the outflow $\zeta_{2;t}$. As it is hard to work directly on the constrained space of A_t, a non-linear transform is used on the non-constrained space x_t.

Denote a single entry of A_t at row i and column i' by $A_{t,i,i'}$. Note that $A_{t,i,i'}$ is non-negative and if $A_{t,i,i'} = 0$, then there is no connection between the inflow $\zeta_{1;t,i'}$ and the outflow $\zeta_{2;t,i}$. Let B_t be another $d_{\zeta;2} \times d_{\zeta;1}$ matrix with $B_{t,i,i'} = -\inf$ if there is network connection between $\zeta_{1;t,i'}$ and the outflow $\zeta_{2;t,i}$. Also, for each column i' of B_t, there must be one predefined entry of value 0: $B_{t,i,i'} = 0$. The locations of these zero-entries can be randomly chosen. Then A_t can be constructed from B_t using the logistic transformation:

$$A_{t,i,i'} = \frac{\exp(B_{t,i,i'})}{\sum_k \exp(B_{t,k,i'})}. \tag{6.17}$$

For example, the matrix B_t for the junction J-0103 in Figure 6.7 can be defined as:

$$B_t = \begin{pmatrix} x_{t,1} & x_{t,2} \\ 0 & 0 \end{pmatrix}. \tag{6.18}$$

So, the state $x_t = (x_{t,1}, x_{t,2})$ is a vector of non-zero finite entries $B_{t,i,i'}$.

For the analysis purposes, in addition to the data of junction J-0103, we use another simulated dataset described in Example 15.

Example 15 Consider an imaginary junction with 2 inflows ($d_{\zeta;1} = 2$) and 3 outflows ($d_{\zeta;2} = 3$). There are 4 inflow-outflow connection pairs $(1 \to 1)$, $(1 \to 2)$, $(1 \to 3)$, $(2 \to 1)$ and B_t is chosen as

$$B_t = \begin{pmatrix} x_{t,1} & 0 \\ x_{t,2} & -\inf \\ 0 & -\inf \end{pmatrix}. \tag{6.19}$$

Then, instead of using from a random walk, we generate $n = 2000$ points $x_t^\star$ from a deterministic function:

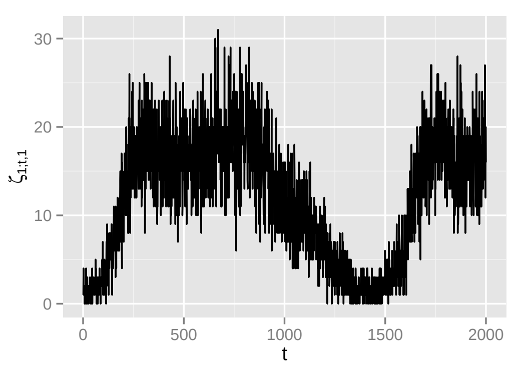
$$x_{t,1}^\star = \sin(\pi t/100), \tag{6.20}$$

$$x_{t,2}^\star = \cos(\pi t/100). \tag{6.21}$$

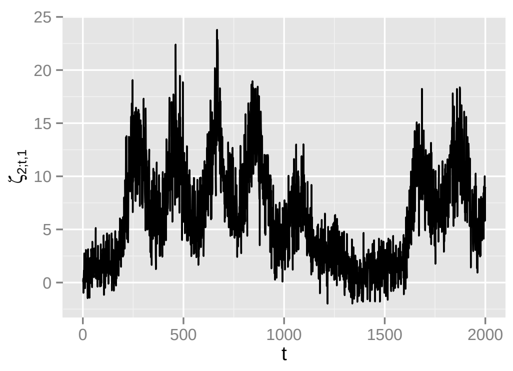
Assuming that the above deterministic trace is from a random walk, the precision matrix $Q_u^\star$ is estimated:

$$Q_u^\star = \begin{pmatrix} 2026.59 & -0.03 \\ -0.03 & 2024.56 \end{pmatrix}. \tag{6.22}$$

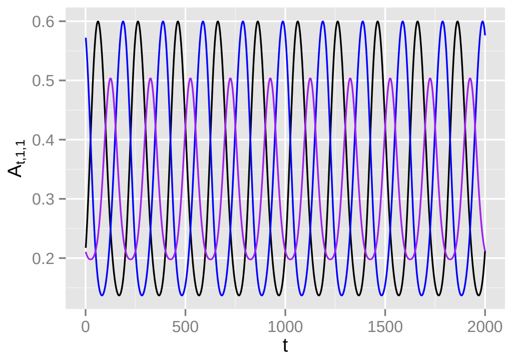
The inflow series $\zeta_{1;t}$ is chosen from the VISSIM flow series. Then, conditioning on $\zeta_{1;t}$ and x_t, $n = 2000$ outflow points $\zeta_{2;t}$ are generated by Equation (6.15) with $\lambda_v^\star = (1, 4/9, 0.25)$. The plots of $\zeta_{1;t,1}$, $\zeta_{2;t,1}$ and $A_{t,\cdot,1}$ are shown in Figure 6.8.



(a) $\zeta_{1;t,1}$



(b) $\zeta_{2;t,1}$



(c) $A_{t,\cdot,1}$

Fig. 6.8: Example 15: the plots of inflow $\zeta_{1;t,1}$, outflow $\zeta_{2;t,1}$ and turning rates $A_{t,\cdot,1}$. In Figure 6.8(c), the black, blue and purple curves are $A_{t,1,1}$, $A_{t,2,1}$ and $A_{t,3,1}$ respectively.

6.3.1 Offline estimation

Before working on sequential inference, it is best to analyse the junction sub-model using an offline method, which is free of approximation error. In practise, such an offline method can be used for a small portion of the dataset and the result is then used as a prior for subsequent sequential inference.

The *maximum a posteriori probability (MAP)* estimation method is used for this model. Denoting the priors of λ_v and Q_u by $p_{\lambda_v}(\cdot)$ and $p_{Q_u}(\cdot)$, MAP maximises the joint posterior of parameters and state vector $p_{\lambda_v,Q_u,x_{1:n}|\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot)$. This optimisation process for the DM is however not trivial due to the strong dependency within $(\lambda_v, Q_u, x_{1:n})$ and the high dimension of the variable space. So, we will use a special optimisation procedure for this issue.

Conjugate priors are used for λ_v and Q_u with:

$$p_{\lambda_v}(\cdot) = \prod_i p_{\lambda_{v;i}}(\cdot) = \prod_i Gamma(\cdot|\alpha_{\lambda;i}, \beta_{\lambda;i}), \tag{6.23}$$

$$p_{Q_u}(\cdot) = W(\cdot|V_u, n_u), \tag{6.24}$$

where $\alpha_{\lambda;i}$ and $\beta_{\lambda;i}$ are the shape and rate parameters of the gamma distribution; V_u and n_u are the scale matrix and the degree of the Wishart distribution. The joint posterior $p_{\lambda_v,Q_u,x_{1:n}|\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot)$ is then:

$$\begin{aligned}
&p_{\lambda_v,Q_u,x_{1:n}|\zeta_{1;1:n},\zeta_{2;1:n}}(\lambda_v, Q_u, x_{1:n}) \qquad (6.25)\\
\propto &\prod_i Gamma(\lambda_{v;i}|\alpha_{\lambda;i}, \beta_{\lambda;i})\ W(Q_u|V_u, n_u)\\
&\prod_{t=2}^{n} N(x_t|x_{t-1}, Q_u)\ \prod_{t=1}^{n} N(\zeta_{2;t}|c_t = A_t(x_t)\zeta_{1;t}, Q = \lambda_v I),\\
\propto &\prod_i \lambda_{v;i}^{\alpha_{\lambda;i}-1} \exp(-\beta_{\lambda;i}\lambda_{v;i})\ |Q_u|^{(n_u-d_x-1)/2} \exp(-\operatorname{tr}(V_u^{-1}Q_u)/2)\\
&|Q_u|^{(n-1)/2} \exp\left[-\frac{\sum_{t=2}^{n}(x_t - x_{t-1})^T Q_u (x_t - x_{t-1})}{2}\right]\\
&\prod_i \lambda_{v;i}^{n/2} \exp\left[-\frac{\sum_{t=1}^{n}(\zeta_{2;t} - c_t)^T \lambda_v I (\zeta_{2;t} - c_t)}{2}\right].
\end{aligned}$$

where $c_t = A_t(x_t)\zeta_{1;t}$ is the conditional mean of $\zeta_{2;t}$.

From Equation (6.25), it can be seen that the full conditional posteriors of λ_v and Q_u are still gamma and Wishart distributions :

$$p_{\lambda_v|Q_u,x_{1:n},\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot) = \prod_i Gamma(\cdot|\alpha'_{\lambda;i}, \beta'_{\lambda;i}), \tag{6.26}$$

$$p_{Q_u|\lambda_v,x_{1:n},\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot) = W(\cdot|V'_u, n'_u), \tag{6.27}$$

with:

$$\alpha'_{\lambda;i} = \alpha_{\lambda;i} + n/2, \tag{6.28}$$

$$\beta'_{\lambda;i} = \beta_{\lambda;i} + \frac{\sum_{t=1}^{n}(\zeta_{2;t,i} - c_{t,i})^2}{2}, \tag{6.29}$$

$$V_u'^{-1} = V_u^{-1} + \sum_{t=2}^{n}(x_t - x_{t-1})(x_t - x_{t-1})^T, \tag{6.30}$$

$$n'_u = n_u + n - 1. \tag{6.31}$$

Unfortunately, the full conditional posterior of the state vector $x_{1:n}$ is intractable. Still, it is noted that the gradient and the hessian of $-\log(p_{x_{1:n}|\lambda_v,Q_u,\zeta_{1;1:n},\zeta_{2;1:n}})$ can be computed analytically by the chain rule.

With the full conditional posteriors of λ_v, Q_u and $x_{1:n}$, the optimisation procedure is summarised in Algorithm 14.

Algorithm 14 Optimisation of $p_{\lambda_v,Q_u,x_{1:n}|\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot)$ for the junction sub-model

```
1. Calculate the mode $\overline{\lambda}_v$ of $p_{\lambda_v|Q_u,x_{1:n},\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot) = \prod_i Gamma(\cdot|\alpha'_{\lambda;i},\beta'_{\lambda;i})$. Set $\lambda_v = \overline{\lambda}_v$.

2. Calculate the mode $\overline{Q}_u$ of $p_{Q_u|\lambda_v,x_{1:n},\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot) = W(\cdot|V'_u,n'_u)$. Set $Q_u = \overline{Q}_u$.

3. Use Newton optimisation on the state vector for $n_{opt;x}$ iterations:

   $$\overline{x}_{1:n} = \arg\min_x(-\log(p_{x_{1:n}|\lambda_v,Q_u,\zeta_{1;1:n},\zeta_{2;1:n}})), \quad (6.32)$$

   and set $x_{1:n} = \overline{x}_{1:n}$.
```

Like Gibbs sampling, Algorithm 14 is applied repeatedly for n_{opt} iterations. Notice that while Gibbs sampling samples the full conditional posterior, the above algorithm directly locates the modes of the full conditional posteriors. For λ_v and Q_u, those modes can be obtained analytically by using the properties of gamma and Wishart distributions. For notation purposes, denote the optimisation result of the above procedure by $(\overline{\lambda}_v, \overline{Q}_u, \overline{x}_{1:n})$.

The optimisation is implemented both in R and C languages. For the dataset of Example 15, the evaluation of $p_{\lambda_v,Q_u,x_{1:n}|\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot)$ for 100 points takes 580.965 seconds in R and 0.122 seconds in C (4762 times faster). Also, to avoid the finite difference of the gradient evaluation used in Newton optimisation and speed up the above process, we directly obtain the gradient as mentioned previously.

Optimisation results

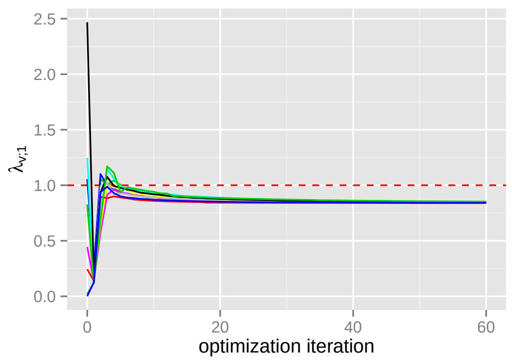
For Example 15, the priors are set with :

$$\alpha_\lambda = (2.000, 2.020, 2.040), \qquad \beta_\lambda = (0.100, 0.101.0.102), \qquad (6.33)$$
$$V_u = \begin{pmatrix} 2.500 & -0.003 \\ -0.003 & 2.692 \end{pmatrix}, \qquad n_u = 4.$$

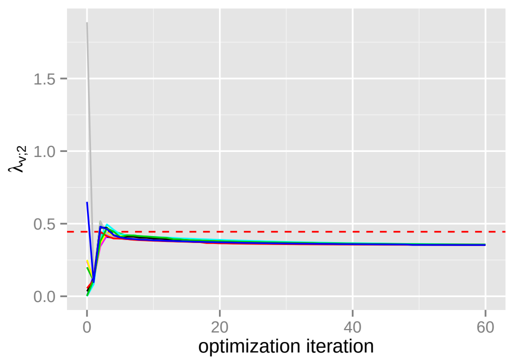
Ten optimisations with random initial values $(\lambda_v, Q_u, x_{1:n})$ are run for Example 15. In each run, there are $n_{opt} = 5000$ iterations for the main loop and $n_{opt;x} = 10$ iterations for the sub-optimisation of $x_{1:n}$. The optimisation traces of $\overline{\lambda}_v$ and $\overline{Q}_u$ in the first 60 iterations are plotted in Figure 6.9. The whole series of the state vector $\overline{x}_{1:n}$ in the last optimisation iteration is shown in Figure 6.10.

In Figure 6.10, during some intervals ($t = 1 : 100$ and $t = 1100 : 1600$) the estimation $\overline{x}_t$ is much smoother than the true series $x^\star_t$. These intervals correspond to the durations where both the inflow $\zeta_{1;t}$ and the outflow $\zeta_{2;t}$ are very small, as plotted in Figure 6.8. Small values of $\zeta_{1;t}$ and $\zeta_{2;t}$ give less information to the turning rate matrix A_t (or equivalently x_t), according to Equation (6.15). Hence, the estimation of x_t during these interval is driven mostly by the random walk.

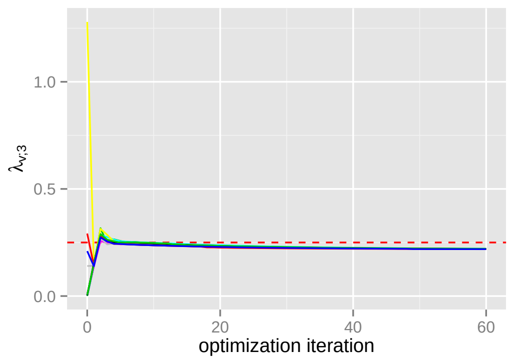
The estimation $\overline{\zeta}_{2;t+k|t} = A_t(\overline{x}_t)\zeta_{1;t+k}$ of the outflow $\overline{\zeta}_{2;t+k|t}$, using the optimisation state $\overline{x}_t$ and the inflow $\zeta_{1;t+k}$, is not a "true" prediction as $\overline{x}_t$ is obtained by conditioning on the whole data. Still, the comparison between $\overline{\zeta}_{2;t|t-k}$ and $\zeta_{2;t}$ could confirm the compatibility between the offline estimation and the data. Hence, the estimations $\overline{\zeta}_{2;t|t}$ and $\overline{\zeta}_{2;t|t-10}$ for the duration $t = 1000 : 1200$ are shown in Figures 6.11 and 6.12. The mean square error $s_{mse;i,k}$ between $\overline{\zeta}_{2;t+k,i|t}$ and $\zeta_{2;t+k,i}$ is also summarised in Table 6.3.



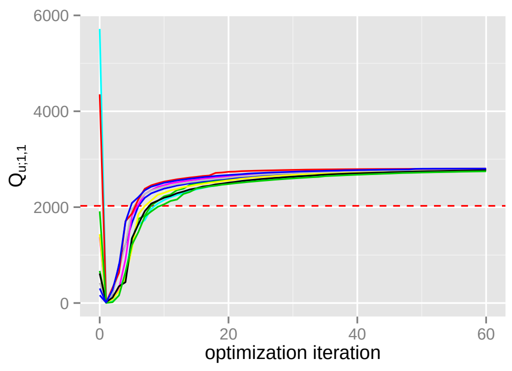
(a) $\lambda_{v;1}$



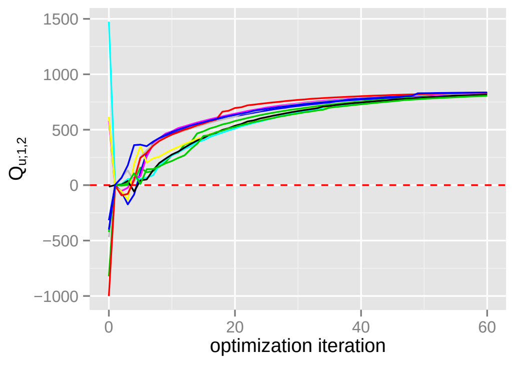
(b) $\lambda_{v;2}$



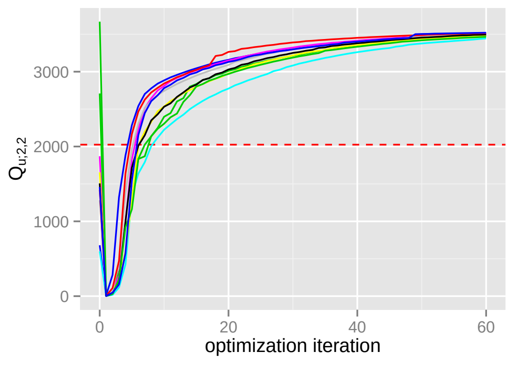
(c) $\lambda_{v;3}$



(d) $Q_{u;1,1}$

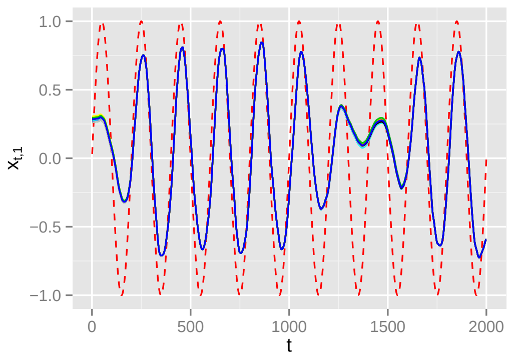


(e) $Q_{u;1,2}$

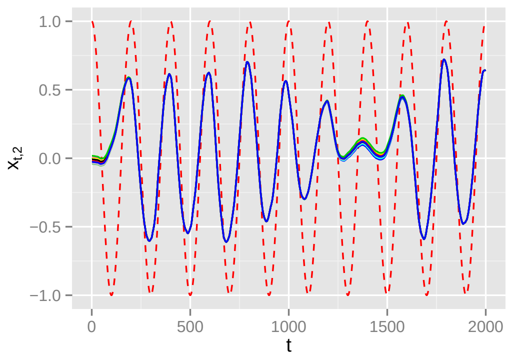


(f) $Q_{u;2,2}$

Fig. 6.9: Example 15: the optimisation traces of $\overline{\lambda}_v$ and $\overline{Q}_u$ in the first 60 iterations. Different coloured curves correspond to the traces of different optimisation runs. The red dashed lines mark the true value $\lambda_v^\star$ and $Q_u^\star$.

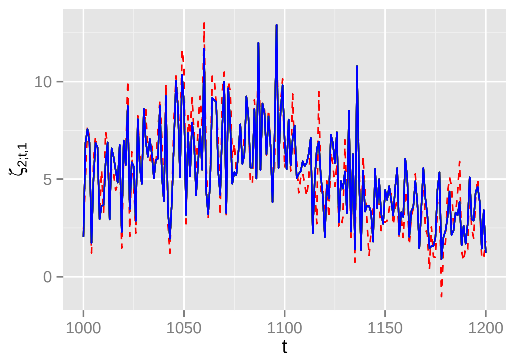


(a) $x_{t,1}$

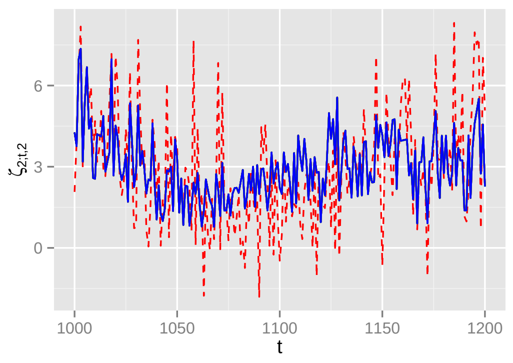


(b) $x_{t,2}$

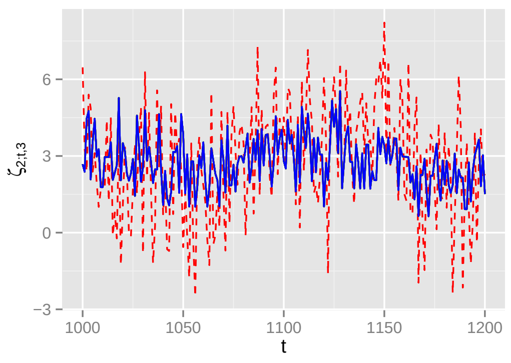
Fig. 6.10: Example 15: the estimation series of x_t. Different coloured curves correspond to the series of different optimisation runs. The red dashed curves mark the true series $x_t^\star$.



(a) $\zeta_{2;t,1}$

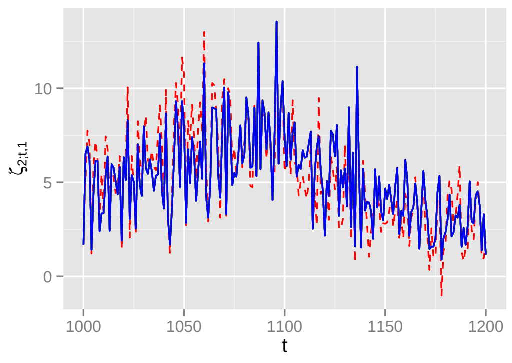


(b) $\zeta_{2;t,2}$

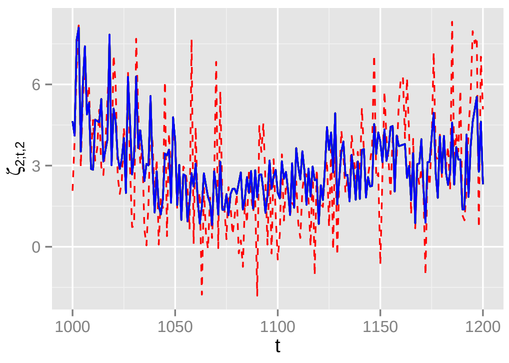


(c) $\zeta_{2;t,3}$

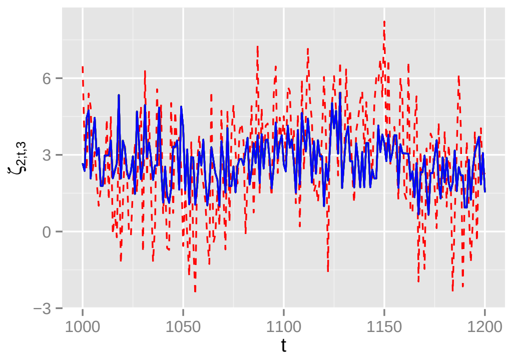
Fig. 6.11: Example 15: the estimation $\overline{\zeta}_{2;t|t}$. Different coloured curves correspond to the series of different optimisation runs. The red dashed curves represent the true outflow $\zeta_{2;t}$.



(a) $\zeta_{2;t,1}$



(b) $\zeta_{2;t,2}$



(c) $\zeta_{2;t,3}$

Fig. 6.12: Example 15: the estimation $\overline{\zeta}_{2;t|t-10}$. Different coloured curves correspond to the series of different optimisation runs. The red dashed curves represent the true outflow $\zeta_{2;t}$.

The last puzzle of these results is that the estimation $\overline{Q}_u$ is much higher than the true precision matrix $Q_u^\star$. This in turn causes the estimation $\overline{x}_t$ to be globally smoother than $x_t^\star$. This issue may be clearer by analysing the components of the joint posterior $p_{\lambda_v,Q_u,x_{1:n}|\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot)$. The log density components $lp_a = \log(p_{\lambda_v})$, $lp_b = \log(p_{Q_u})$, $lp_c = \log(p_{x_{1:n}|Q_u})$ and $lp_d = \log(p_{\zeta_{2;1:n}|\lambda_v,x_{1:n},\zeta_{2;1:n}})$, evaluated at the true value $(\lambda_v^\star, Q_u^\star, x_{1:n}^\star)$ and the estimation $(\overline{\lambda}_v, \overline{Q}_u, \overline{x}_{1:n})$, are shown in Table 6.4.

The component lp_d measures the matching between the state vector x_t (or $c_t = A_t(x_t)\zeta_{1;t}$) and the outflow $\zeta_{2;t}$. The higher the log density lp_d, the closer the distance between c_t and $\zeta_{2;t}$. So, in Table 6.4, lp_d is indeed higher at the true value $(\lambda_v^\star, Q_u^\star, x_{1:n}^\star)$ than at the estimated value $(\overline{\lambda}_v, \overline{Q}_u, \overline{x}_{1:n})$. However, this is countered by the component lp_c. The higher the precision Q_u and the smoother the state vector x_t, the larger the

Table 6.3: Example 15: the mean square error $s_{mse;i,k}$ between $\overline{\zeta}_{2;t+k,i|t}$ and $\zeta_{2;t+k,i}$.

	k			
	0	10	20	30
$s_{mse;1,k}$	1.183	1.832	3.299	5.425
$s_{mse;2,k}$	2.827	2.739	3.380	4.720
$s_{mse;3,k}$	4.525	4.737	5.160	5.737

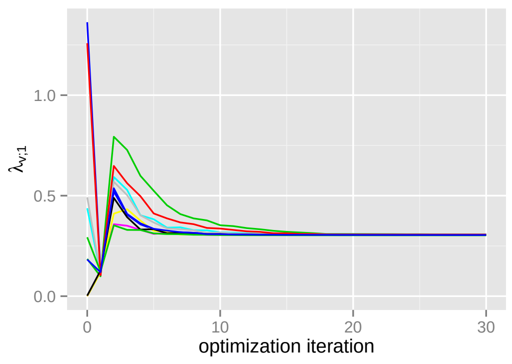
log density $lp_c = \log(p_{x_{1:n}|Q_u})$ of the random walk. In total, the log of joint posterior is truly higher at the estimated value.

This issue is partially solved by using a stronger prior on Q_u to balance the density of the random walk, preventing the precision matrix from going too high. Depending on the context, one can also fix either the observation precision λ_v (how close the conditional mean c_t and the outflow are) or the state precision matrix Q_u (how smooth the state vector is) to its own preference and estimate the remaining variables. Maybe, the best solution is by looking at the marginal density $p_{\lambda_v,Q_u|\zeta_{1;1:n},\zeta_{2;1:n}}(\cdot)$. However, for this non-linear model, the marginalisation cannot be done analytically and exactly. The INLA method (Rue et al., 2009) can approximate the marginal density but one needs to be wary of the approximation of the state vector.

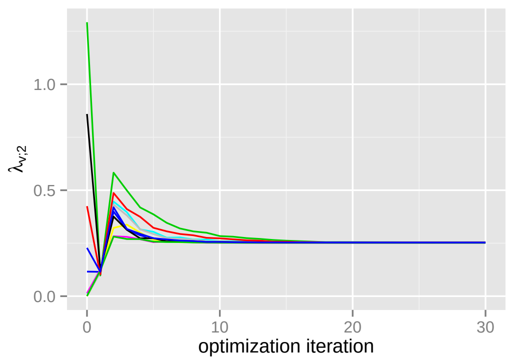
Even though there is a difficulty with parameter estimation of this model, the results of the state vector are still reasonable, leading to the matching between $\overline{\zeta}_{2;t+k|t}$ and $\zeta_{2;t+k}$. For our purposes in the traffic modelling, this seems to be good enough.

Table 6.4: Example 15: the log density components $lp_a = \log(p_{\lambda_v})$, $lp_b = \log(p_{Q_u})$, $lp_c = \log(p_{x_{1:n}|Q_u})$ and $lp_d = \log(p_{\zeta_{2;1:n}|\lambda_v,x_{1:n},\zeta_{2;1:n}})$, evaluated at the true value $(\lambda_v^\star, Q_u^\star, x_{1:n}^\star)$ and the estimation $(\overline{\lambda}_v, \overline{Q}_u, \overline{x}_{1:n})$.

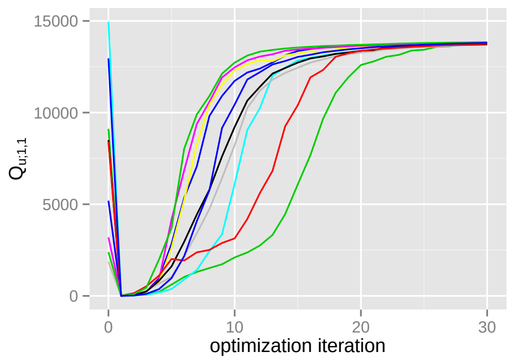
	lp_a	lp_b	lp_c	lp_d	Total
$(\lambda_v^\star, Q_u^\star, x_{1:n}^\star)$	−16.358	−780.709	9547.690	−10832.427	−2081.805
$(\overline{\lambda}_v, \overline{Q}_u, \overline{x}_{1:n})$	−16.880	−1219.176	11576.200	−11253.044	−912.899



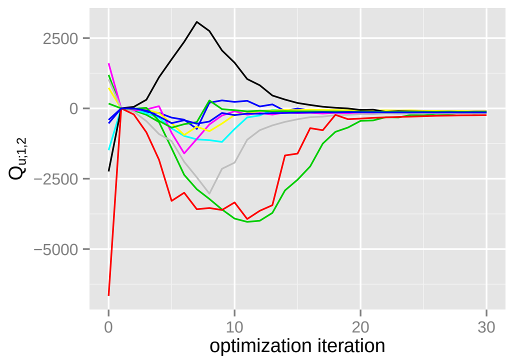
(a) $\lambda_{v;1}$



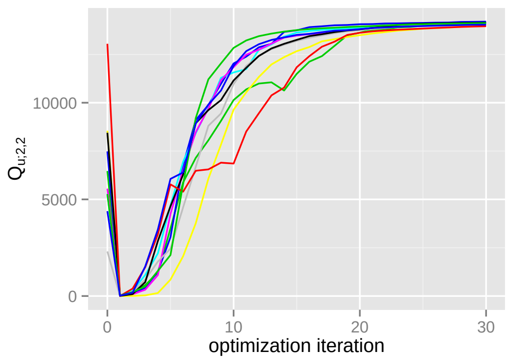
(b) $\lambda_{v;2}$



(c) $Q_{u;1,1}$

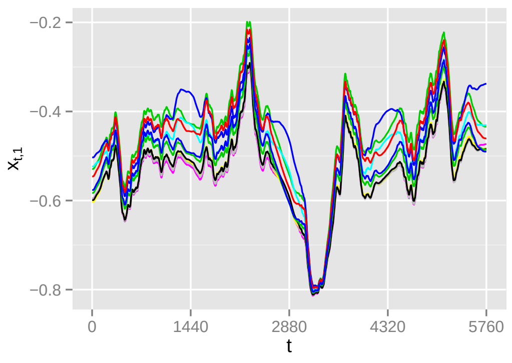


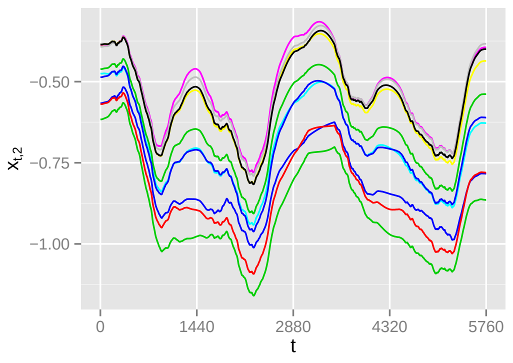
(d) $Q_{u;1,2}$



(e) $Q_{u;2,2}$

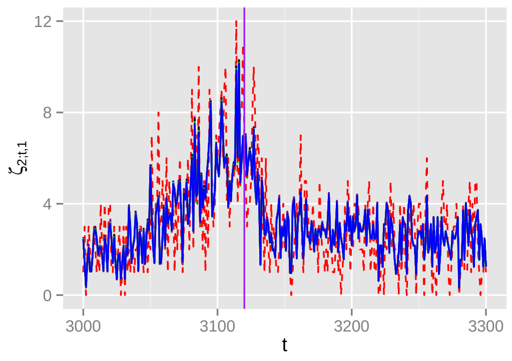
Fig. 6.13: Junction J-0103 of Example 14: the optimisation traces of $\overline{\lambda}_v$ and $\overline{Q}_u$ in the first 30 iterations. Different coloured curves correspond to the traces of different optimisation runs.

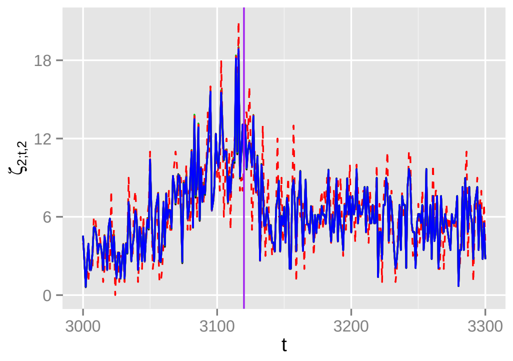




(a) $x_{t,1}$ (b) $x_{t,2}$

Fig. 6.14: Junction J-0103 of Example 14: the estimation series of x_t. Different coloured curves correspond to the series of different optimisation runs.





(a) $\zeta_{2;t,1}$ (b) $\zeta_{2;t,2}$

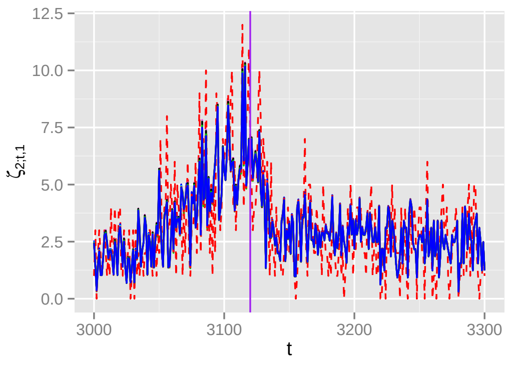
Fig. 6.15: Junction J-0103 of Example 14: the estimation $\overline{\zeta}_{2;t|t}$. Different coloured curves correspond to the series of different optimisation runs. The red dashed curves represent the true outflow $\zeta_{2;t}$. The purple vertical lines mark the starting point of the traffic incident.

A similar procedure is applied on the VISSIM dataset of junction J-0103 in Figure 6.7. The priors for this dataset are set with:

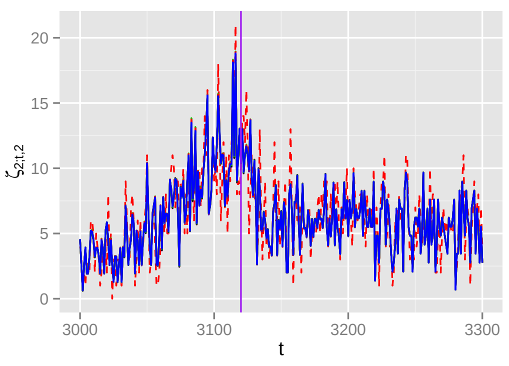
$$\alpha_\lambda = (2, 2), \qquad \beta_\lambda = (0.1, 0.1), \tag{6.34}$$

$$V_u = \begin{pmatrix} 2.5 & 0.0 \\ 0.0 & 2.5 \end{pmatrix}, \qquad n_u = 4.$$

The estimations of λ_v, Q_u, x_t, $\overline{\zeta}_{2;t|t}$ and $\overline{\zeta}_{2;t|t-10}$ are shown in Figures 6.13 to 6.16.



(a) $\zeta_{2;t,1}$



(b) $\zeta_{2;t,2}$

Fig. 6.16: Junction J-0103 of Example 14: the estimation $\overline{\zeta}_{2;t|t-10}$. Different coloured curves correspond to the series of different optimisation runs. The red dashed curves represent the true outflow $\zeta_{2;t}$. The purple vertical lines mark the starting point of the traffic incident.

Table 6.5: Junction J-0103 of Example 14: the mean square error $s_{mse;i,k}$ between $\overline{\zeta}_{2;t+k,i|t}$ and $\zeta_{2;t+k,i}$.

	k			
	0	10	20	30
$s_{mse;1,k}$	3.104	3.134	3.163	3.173
$s_{mse;2,k}$	3.753	3.802	3.842	3.855

There is an interesting point in these results. Even though (λ_v, Q_u) converges very quickly after 30 iterations, the series x_t of different optimisation runs are quite different (after 5000 iterations), being about parallel with each other. This is explained by examining the map of junction J-0103 in Figure 6.7 and the corresponding matrix B_t in Equation (6.18). In the modelling of B_t of junction J-0103, $x_{t,1}$ and $x_{t,2}$ affects the turning rate from the inflows $\zeta_{1;t,1}$ and $\zeta_{1;t,2}$ to the outflow $\zeta_{2;t,1}$ respectively. Hence, conditioning on $\zeta_{2;t,1}$, one estimation may have a high value of $x_{t,1}$ and a low value of $x_{t,2}$ while another estimation may be completely opposite (notice the relative order of series $x_{t,1}$ and $x_{t,2}$ in Figures 6.14(a) and 6.14(b)). This is an identifiability issue for the state vector of this model.

The estimations $\overline{\zeta}_{2;t|t}$ and $\overline{\zeta}_{2;t|t-10}$ seem fine even in the occurrence of the traffic incident. The mean square error $s_{mse;i,k}$ is summarised in Table 6.5.

This concludes the offline estimation for the junction model. We move on with sequential inference for this dynamic non-linear model.

6.3.2 Sequential inference

Algorithm 13 in Section 5.2 is applied to the sequential inference of the junction sub-model of Equation (6.15). As iterLap has better performance in a non-constrained space, we use the log transform on $\tau_v = \log(\lambda_v)$ and the *Cholesky decomposition* on the symmetric positive definite precision matrix $Q_u = R_u^T R_u$ with an upper triangular matrix R_u. For notation purposes, denote r_u as a vector of upper triangular entries of R_u. Then, the junction sub-model can be parametrised by (τ_v, r_u). Also, sequential inference of this sub-model requires a prior for the initial state vector x_1.

Notice that the Cholesky decomposition is unique only under a certain constraint (diagonal entries of R_u are strictly positive). Hence, the mapping from $r_u \in \mathbb{R}^{d_{r_u}}$ to Q_u is many-to-one, which may lead to the identifiability problem. In this case, using Bayesian inference with a prior centred on a particular region will favour values of r_u near that region, lessening the identifiability issue.

The priors for Example 15 are $N(\tau_v|\mu_{\tau_v}, Q_{\tau_v})$, $N(r_u|\mu_{r_u}, Q_{r_u})$ and $N(x_1|\mu_{x_1}, Q_{x_1})$ with parameters:

$$\begin{aligned} \mu_{\tau_v} &= (-1,-1,-1), & \mu_{r_u} &= (3,0,3), & \mu_{x_1} &= (0,0), \\ Q_{\tau_v} &= 25\ I_3, & Q_{r_u} &= 1\ I_3, & Q_{x_1} &= 0.04\ I_2, \end{aligned} \tag{6.35}$$

where the arrangement of r_u in R_u is:

$$R_u = \begin{pmatrix} r_{u;1} & r_{u;2} \\ 0 & r_{u;3} \end{pmatrix}. \tag{6.36}$$

The true values of this parametrisation for Example 15 are $\tau_v^\star = (0.000, -0.811, -1.386)$ and $r_u = (45.018, -0.001, 44.995)$.

To speed up the inference, we implement the sequential procedure for this junction sub-model both in R and C languages. The speed comparison between these 2 implementations is shown in Table 6.6, indicating that the C implementation is significantly faster.

The filtering distributions of (τ_v, r_u, x_t) are summarised in Figures 6.17 and 6.18. The results of τ_v and x_t seem to be consistent with the true values. On the other hand,

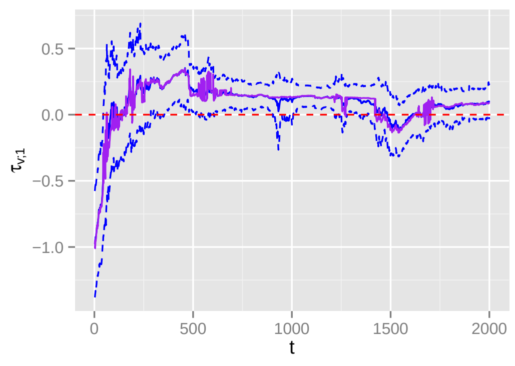
Table 6.6: Speed comparison for Example 15: the running time (seconds) for 1000 evaluations of $p_{\tau_v,r_u,x_t|\zeta_{1;1:(t-1)},\zeta_{2;1:(t-1)}}(\cdot)$ and $p_{\tau_v,r_u,x_t|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$.

	$p_{\tau_v,r_u,x_t\mid\zeta_{1;1:(t-1)},\zeta_{2;1:(t-1)}}(\cdot)$	$p_{\tau_v,r_u,x_t\mid\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$
R	9.145	9.814
C	0.075	0.087

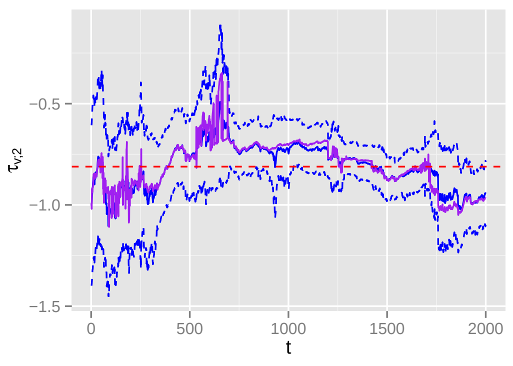
inference of r_u fails to contain the true value within its interval, again indicating the difficulty in the joint sequential inference of both state and observation noise precisions.

In Figure 6.18, during time intervals $t = 1 : 100$ and $t = 1100 : 1600$, the intervals of $p_{x_t|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$ are wider than usual. This is due to the same reason of low values of $\zeta_{1;t}$ and $\zeta_{2;t}$, mentioned in the offline estimation section (during the initial interval $t = 1 : 100$, the uncertainty of x_t is also caused by lack of data in sequential inference).

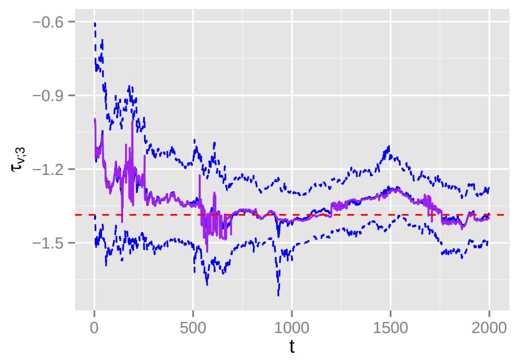
The estimation $\overline{\xi}_{\zeta_{2;t}} = A_t(\overline{\xi}_{x_t})\zeta_{1;t}$ ($\overline{\xi}_{x_t}$ is the mode of $p_{x_t|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$) is computed and plotted together with $\zeta_{2;t}$ in Figure 6.19, showing the compatibility between this inference and the dataset.



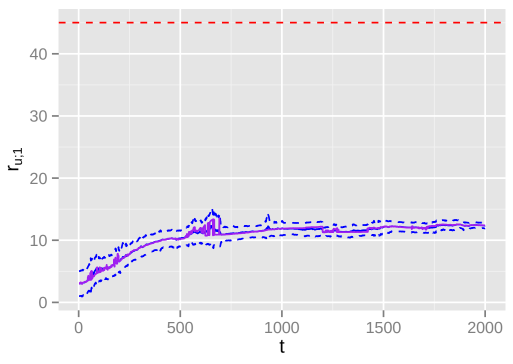
(a) $\tau_{v;1}$



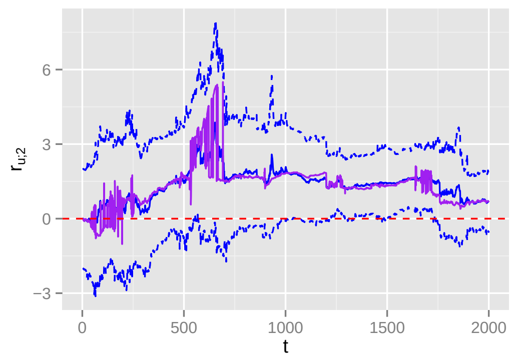
(b) $\tau_{v;2}$



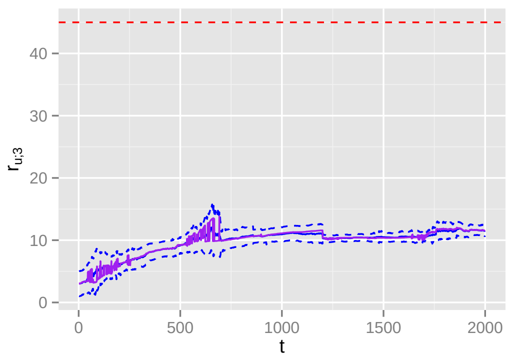
(c) $\tau_{v;3}$



(d) $r_{u;1}$

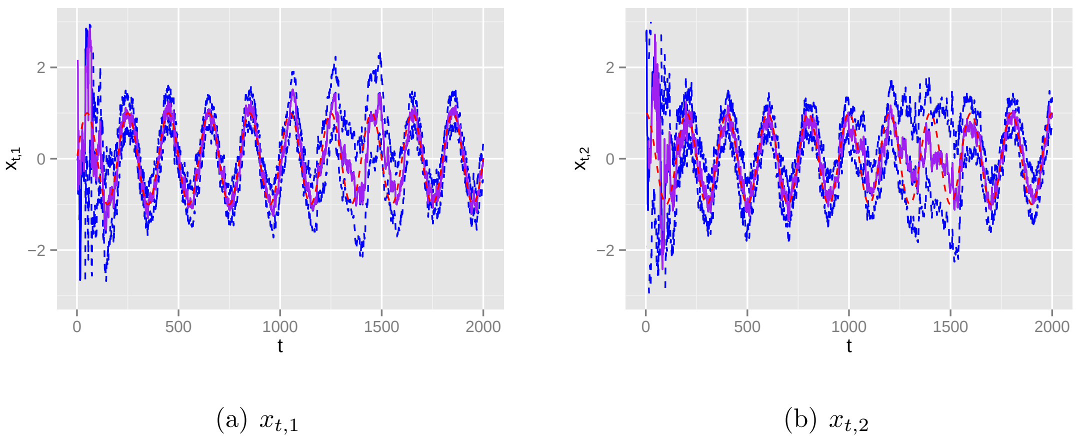


(e) $r_{u;2}$



(f) $r_{u;3}$

Fig. 6.17: Example 15: summary of filtering densities $p_{\tau_v|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$ and $p_{r_u|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$. The blue and purple solid lines represents the mean $\overline{\mu}$ and the mode $\overline{\xi}$ of a particular density. The blue dashed lines mark the interval $[\overline{\mu} - 2\overline{\sigma}, \overline{\mu} + 2\overline{\sigma}]$ with standard deviation $\overline{\sigma}$. The red dashed lines represent the true value of the random variables.



(a) $x_{t,1}$ (b) $x_{t,2}$

Fig. 6.18: Example 15: summary of filtering density $p_{x_t|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$. The blue and purple solid lines represents the mean $\overline{\mu}$ and the mode $\overline{\xi}$ of a particular density. The blue dashed lines mark the interval $[\overline{\mu} - 2\overline{\sigma}, \overline{\mu} + 2\overline{\sigma}]$ with standard deviation $\overline{\sigma}$. The red dashed lines represent the true value of the random variables.

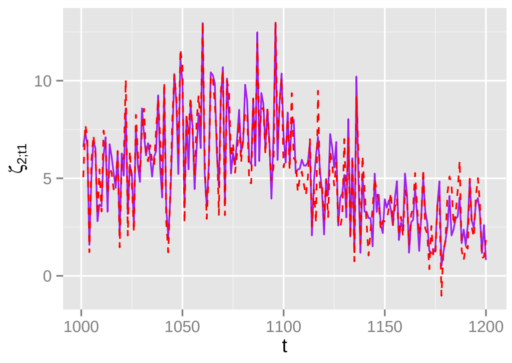
Sequential inference is now used for the dataset of junction J-0103 in Example 14 with prior parameters:

$$\begin{aligned} \mu_{\tau_v} &= (-1, -1), & \mu_{r_u} &= (15, 0, 15), & \mu_{x_1} &= (0, 0), \\ Q_{\tau_v} &= 4\ I_3, & Q_{r_u} &= 1\ I_3, & Q_{x_1} &= 0.04\ I_2, \end{aligned} \tag{6.37}$$

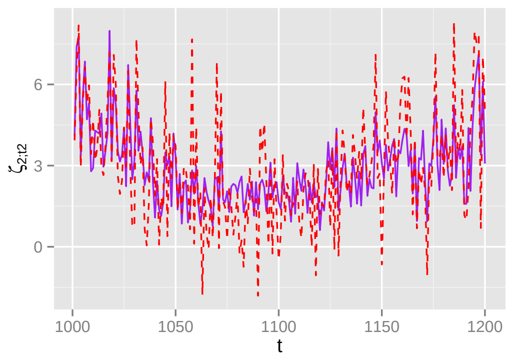
From the optimisation results of (λ_v, Q_u), the transformed optimisation values of τ_v and r_u are $\overline{\tau}_v = \log(\overline{\lambda}_v) = (-1.168, -1.338)$ and $\overline{r}_u = (119.323, -0.059, 119.806)$. Note that these values are from the optimisation conditioning on all the data.

The sequential inference results are summarised in Figures 6.20 and 6.21. In this dataset, estimations of τ_v and r_u are very different from the optimisation results. This difference may be caused by many factors: the parametrisation, the prior, the accumulated error in the sequential method or the mismatching between this particular dataset and the model. Still, the method provides reasonable results of x_t where the optimised series $\overline{x}_t$ is quite close to the mean $\overline{\mu}_{x_t}$ and the mode $\overline{\xi}_{x_t}$ of the filtering distribution $p_{x_t|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$. Also, the estimation $\overline{\xi}_{\zeta_{2;t}}$ plotted in Figure 6.22 is near to the observation $\zeta_{2;t}$, confirming this model's usefulness.

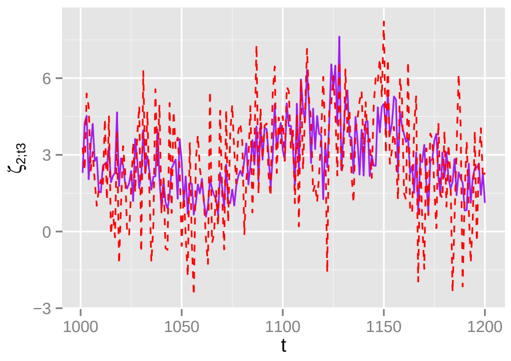
In Figure 6.20, the filtering densities $p_{\tau_v|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$ and $p_{r_u|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$ seem to be shifting according to time index t. This fact suggests that the state and observation noise precisions may be time-dependent, which is quite usual in real datasets. Still, in



(a) $\zeta_{2;t,1}$



(b) $\zeta_{2;t,2}$



(c) $\zeta_{2;t,3}$

Fig. 6.19: Example 15: plots of $\bar{\xi}_{\zeta_{2;t}} = A_t(\bar{\xi}_{x_t})\zeta_{1;t}$ where $\bar{\xi}_{x_t}$ is the mode of $p_{x_t|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$ for $t = 1000 : 1200$. The purple solid lines and red dashed lines are $\bar{\xi}_{\zeta_{2;t}}$ and $\zeta_{2;t}$ respectively.

most cases, statisticians insist on a simple model with a constant precision. So, we will follow this practise and may consider using a time-dependent precision in the future.

In summary, there are issues in both offline and online parameter estimation methods for the proposed junction sub-model. For the simulated dataset, two methods have difficulties in estimating the state noise precision Q_u (or r_u). For the VISSIM dataset, these two methods hand out two different parameter estimation results. As there is no comparison with the true values, it is hard to say either one of them is close to the truth or is better than the other one. Still, with no approximation error, we expect that the offline estimation results are more "stable".

From another point of view, both methods manage to provide "reasonable" estimated parameter values so that inference of state vector is agreeable, giving a good estimation of the junction outflows. As this is our most important goal, we judge that this model and both inference methods are satisfying for our purposes.

6.4 Discussion

It is assumed that there is no upstream information for a root-link and we are back to the univariate STFF problem. Also, notice that even if there are other traffic variables such as speed, volume, density, etc. for the root-link, these variables cannot be used as regressors for the target sub-model. The reason is because we are doing the multi-step ahead prediction of time $t+h$ given only the data up to time t. So, at time t, there is no information of other variables at time $t+h-1$ to be used for the flow prediction of time $t+h$.

One possible solution is to use a univariate AR model with a MP preprocessing method as in Chapter 4. In this case, if a traffic incident occurs at the root-link, the AR model may not have a good prediction of what will happen but it is still able to adapt to the incident (by using online estimation for the AR model). Once an incident takes place and the root sub-model adapts, the other sub-models can propagate the incident effect to the rest of the network.

To make the root sub-model more sensitive and adaptive to an incident occurrence, perhaps a smoothing spline model may work. We plan to use a spline for the root-link inflow:

$$\zeta 1; t = \sum_{i=0}^{p} \alpha_i t^i + \sum_{j=1}^{j_t} \beta_j (t - jk)_+^p + e_t, \tag{6.38}$$

where p is the spline degree; k is a time interval between two consecutive knots; j_t is a natural number satisfying $j_t < \lfloor t/k \rfloor \leq j_t + 1$; e_t may be a normal noise $N(0, \sigma_e^2)$. The parameter vector for this model is $(\alpha, \beta, \sigma_e^2)$. With Equation (6.38), we expect that the smoothing spline can adapt to the data by changing every k time steps. Tebaldi et al. (2002) and Anacleto Junior et al. (2013*a*) also use smoothing spline but the models are different from the above.

The link-vehicle sub-model is simple, following the linear conservation Equation (6.3). The only parameter is the noise precision which can be sequentially estimated by using a conjugate gamma distribution.

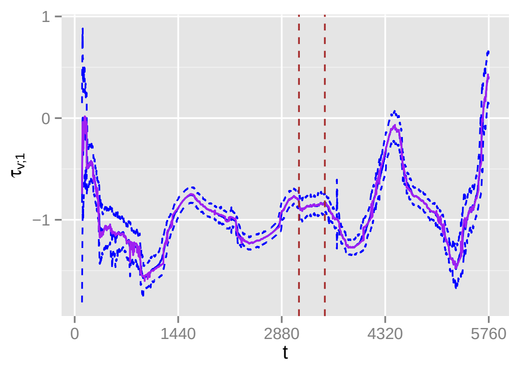
This sub-model however is very crucial in the general modelling as it conserves the number of vehicles traffic networks. Usually multi-step ahead prediction decays quite quickly and the performance becomes worse over time. These shortcomings may be overcome by preserving the network state as much as possible.

Bayesian network and traffic cycles
Modelling a traffic cycle may be intriguing due to the causality issue among flows and some models such as Queen and Albers (2009) and Anacleto Junior et al. (2013*b*) cannot handle such a traffic cycle. However, this problem is solved completely in our model by breaking a traffic cycle to temporal dependencies of flows of different time points. Figure 6.23 shows a network map with a traffic cycle. Bayesian network for this network map is shown in Figure 6.24 at time t and $t+1$. Latent variables such as ones in the junction sub-model are omitted for the sake of simplicity. Notice that a junction-inflow is a vector of link-outflows of some links, e.g. $\zeta_{J1,1;t} = (z_{L1,2;t}, z_{L3,2;t})$, and a junction-outflow comprises link-inflows, e.g. $\zeta_{J1,2;t} = z_{L2,1;t}$. Clearly, it is seen that there is no cycle in such a DAG.

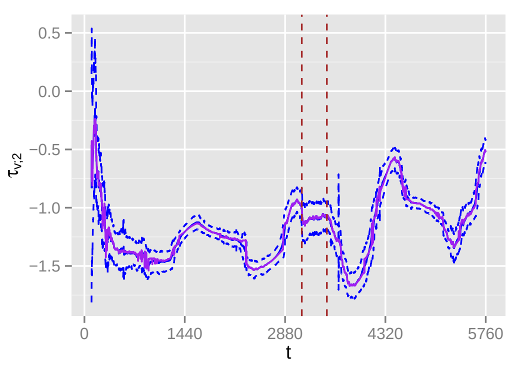
6.5 Conclusion

We have proposed the spatial-temporal model for the STFF problem, primarily focusing on the multi-step-ahead prediction in the occurrence of a incident. This general model comprises four sub-models, each of which is responsible for a particular section of traffic networks.

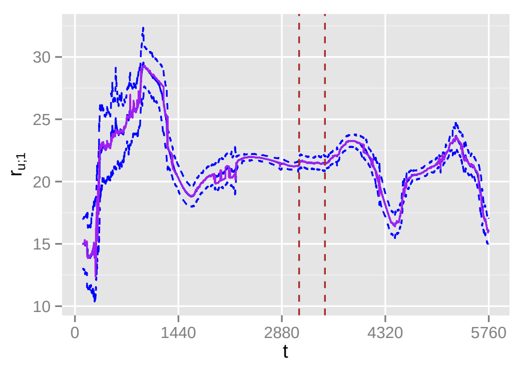
To handle the scalability problem of extending traffic networks, statistical inference of each sub-model can be done conditionally independent with each other. Furthermore, each sub-model is designed so that sequential inference is applicable, meeting the constraints of real time applications. The link-outflow and the junction sub-models are explicitly analysed with proposed inference methods. We have also discussed about possible modelling of the root and link-vehicle sub-models and expect to implement them soon.



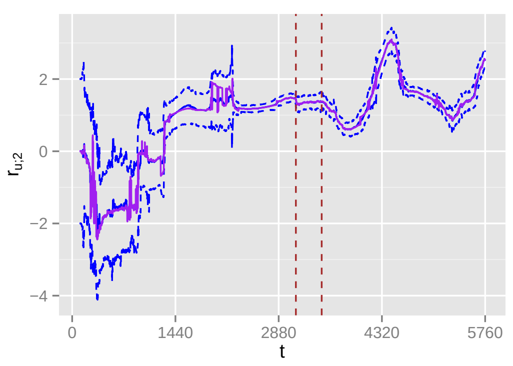
(a) $\tau_{v;1}$



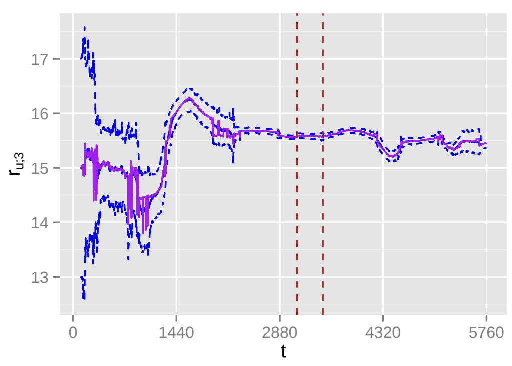
(b) $\tau_{v;2}$



(c) $r_{u;1}$

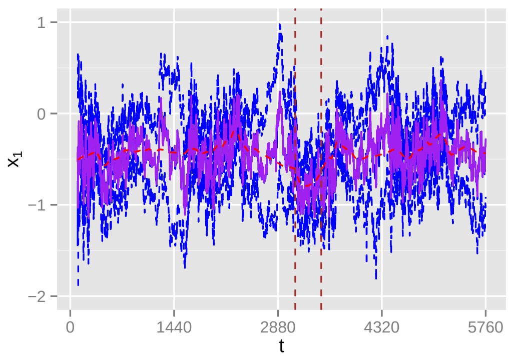


(d) $r_{u;2}$

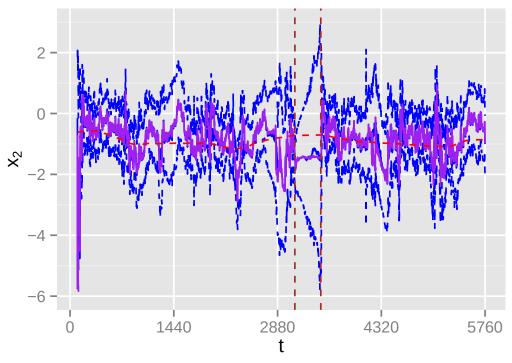


(e) $r_{u;3}$

Fig. 6.20: Junction J-0103 of Example 14: summary of filtering densities $p_{\tau_v|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$ and $p_{r_u|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$. The blue and purple solid lines represents the mean $\overline{\mu}$ and the mode $\overline{\xi}$ of a particular density. The blue dashed lines mark the interval $[\overline{\mu}-2\overline{\sigma}, \overline{\mu}+2\overline{\sigma}]$ with standard deviation $\overline{\sigma}$. The brown dashed vertical lines mark a duration when a traffic incident occurs.

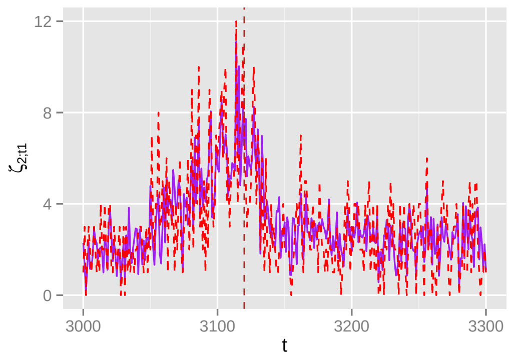


(a) $x_{t,1}$

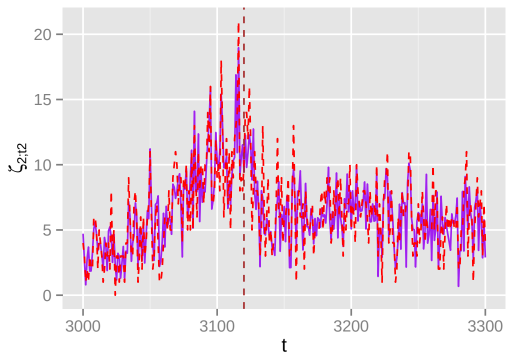


(b) $x_{t,2}$

Fig. 6.21: Junction J-0103 of Example 14: summary of filtering density $p_{x_t|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$. The blue and purple solid lines represents the mean $\overline{\mu}$ and the mode $\overline{\xi}$ of a particular density. The blue dashed lines mark the interval $[\overline{\mu}-2\overline{\sigma}, \overline{\mu}+2\overline{\sigma}]$ with standard deviation $\overline{\sigma}$. The red dashed lines represent the optimised series $\overline{x}_t$, using all data. The brown dashed vertical lines mark a duration when a traffic incident occurs.



(a) $\zeta_{2;t,1}$



(b) $\zeta_{2;t,2}$

Fig. 6.22: Junction J-0103 of Example 14: plots of $\overline{\xi}_{\zeta_{2;t}} = A_t(\overline{\xi}_{x_t})\zeta_{1;t}$ where $\overline{\xi}_{x_t}$ is the mode of $p_{x_t|\zeta_{1;1:t},\zeta_{2;1:t}}(\cdot)$ for $t = 3000 : 3300$. The purple solid lines and red dashed lines are $\overline{\xi}_{\zeta_{2;t}}$ and $\zeta_{2;t}$ respectively. The brown dashed vertical lines mark the starting point of the traffic incident.

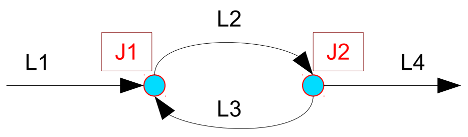


Fig. 6.23: A network map with a traffic cycle.

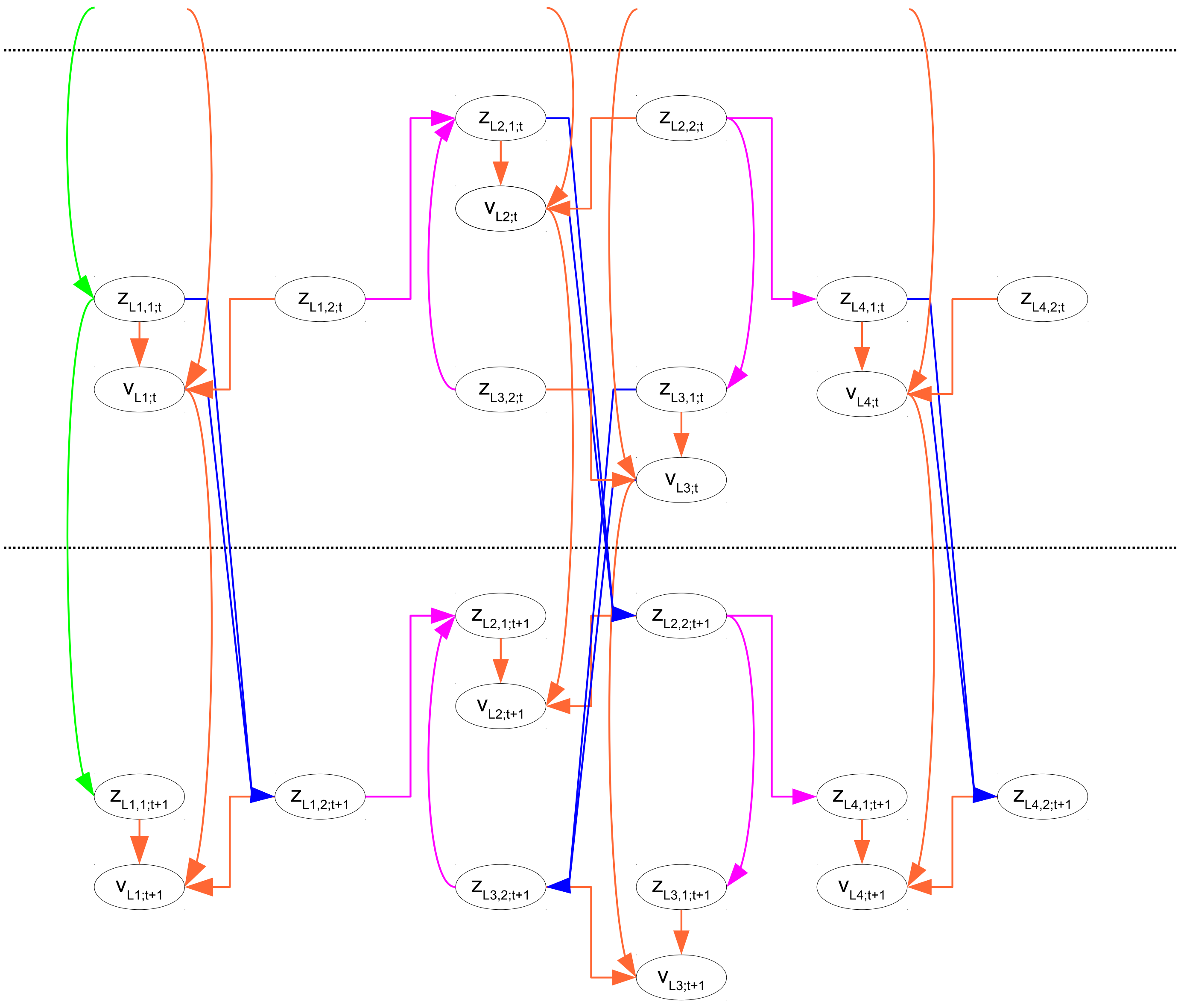


Fig. 6.24: A Bayesian network for the network map in Figure 6.23. Blue, pink, green and red connectors are for the link-inflow, junction, root-inflow and number-of-vehicle-within-a-link sub-models respectively. Connectors with no source nodes originate from variables at time $t-1$ which are omitted together with latent variables for the sake of simplicity.

Chapter 7

Conclusion

In the first part of this thesis, we have applied the VARMA model to the STFF problem, together with an improved MCMC algorithm designed to deal with the variable correlation issue of the VARMA model. For a dataset with a normal traffic pattern, using a daily mean with a VARMA noise would be satisfying enough. However, when an incident occurs and the traffic pattern shift significantly, such a model may suffer from the exponential decaying and the slow adaptation of the VARMA model.

So, the second proposed model is designed to be as robust to the incident occurrence as possible, with a stable relationship between traffic variables. The model comprises of four sub-models, each of which is conditionally independent with each other and can be analysed separately, satisfying the scalability property. Also, to meet the real-time requirements, each sub-model supports sequential inference. The link-outflow and junction sub-models have been implemented and tested with VISSIM dataset, giving reasonable results.

For sequential inference of the DM, we propose a new continuous sequential approximation for both state vector and parameters. This new algorithm has been examined with several examples, giving varying results. In general, the algorithm works fine but has difficulties with identifiability and variance estimation. The core of this sequential approximation is iterLap (Bornkamp, 2011*a*), in which we have made several modifications to improve the performance.

As this research progresses, several challenges arises, giving way to possible future research:

- Implement the root and link-vehicle sub-models then combine together and analyse multi-step-ahead prediction with the VISSIM data.

- For the junction sub-model, the random walk of the state vector can be replaced by a smoothing spline like Equation (6.38). In that case, the random smoothing spline can be thought as a time-variant mean of the usual state series, which might be more interesting to multi-step-ahead prediction. Also, such a model moves all the uncertainty to the observation equation, which is a good thing to our sequential approximation algorithm.

- For sequential inference of the DM, we would like to compare the proposed algorithm with the particle filter, especially for datasets with outliers where the degeneracy is the most severe. More examples of sequential parameter estimation to illustrate the method performance is also preferred. Furthermore, a new sequential algorithm is needed for the following model:

$$y_t \sim p_{y_t|x_t,\varphi}(\cdot), \tag{7.1}$$

$$x_t = \sum_{i=0}^{p} \alpha_i t^i + \sum_{j=1}^{j_t} \beta_j (t - jk)_+^p, \tag{7.2}$$

  which is the generalized model for the proposed root sub-model. Comparison between such a smoothing model and a dynamic model would be interesting in both theory and practice.

Bibliography

Aboudolas, K., Papageorgiou, M. and Kosmatopoulos, E. (2009), 'Store-and-forward based methods for the signal control problem in large-scale congested urban road networks', *Transportation Research Part C: Emerging Technologies* **17**(2), 163 – 174.

Aboudolas, K., Papageorgiou, M., Kouvelas, A. and Kosmatopoulos, E. (2010), 'A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks', *Transportation Research Part C: Emerging Technologies* **18**(5), 680 – 694.

Ahmed, M. S. and Cook, A. R. (1979), 'Analysis of freeway traffic time series data by using Box-Jenkins techniques', *Transportation Research Record No. 722, Urban Systems Operations* pp. 1–9.

Anacleto Junior, O., Queen, C. and Albers, C. J. (2013*a*), 'Forecasting multivariate road traffic flows using bayesian dynamic graphical models, splines and other traffic variables', *Australian & New Zealand Journal of Statistics* . to appear.

Anacleto Junior, O., Queen, C. and Albers, C. J. (2013*b*), 'Multivariate forecasting of road traffic flows in the presence of heteroscedasticity and measurement errors', *Journal of the Royal Statistical Society: Series C (Applied Statistics)* .

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. (2005), Online parameter estimation in general state space models, *in* 'Proceedings of the 44th Conference on Decision and Control', pp. 332–337.

Ansley, C. (1979), 'An algorithm for the exact likelihood of a mixed autoregressive moving average process', *Biometrika* **66**(1), 59–65.

Arulampalam, M. S., Maskell, S. and Gordon, N. (2002), 'A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking', *IEEE Transactions on Signal Processing* **50**, 174–188.

Barnett, G., Kohn, R. and Sheather, S. (1997), 'Robust Bayesian estimation of autoregressive moving average models', *Journal of Time Series Analysis* **18**(1), 11–28.

Bishop, C. M. (2006), *Pattern recognition and machine learning*, 1st edn, Springer.

Bornkamp, B. (2011*a*), 'Approximating probability densities by iterated Laplace approximations', *Journal of Computational and Graphical Statistics* **20**, 656–669.

Bornkamp, B. (2011*b*), *iterLap: iterated Laplace approximations.*
URL: *http://CRAN.R-project.org/package=iterLap*

Box, G., Jenkins, G. and Reinsel, G. (2008), *Time series analysis: forecasting and control*, Wiley Series in Probability and Statistics, Wiley.

Box, G. and Tiao, G. (1992), *Bayesian inference in statistical analysis*, Wiley classics library, Wiley.

Brockwell, P. and Davis, R. (2002), *Introduction to time series and forecasting*, Springer Texts in Statistics, Springer.

Carpenter, J., Clifford, P. and Fearnhead, P. (1999), 'An improved particle filter for non-linear problems', *IEE Proceedings – Radar, Sonar and Navigation,* **46**(1), 2–7.

Carter, C. K. and Kohn, R. (1994), 'On Gibbs sampling for state space models', *Biometrika* **81**(3), 541–553.

Carvalho, C. M., Johannes, M. S., Lopes, H. F. and Polson, N. G. (2010), 'Particle learning and smoothing', *Statistical Science* **25**, 88–106.

Casella, G. and Berger, R. (2002), *Statistical inference*, Duxbury advanced series in statistics and decision sciences, Thomson Learning.

Chandra, S. R. and Al-Deek, H. (2009), 'Predictions of freeway traffic speeds and volumes using vector autoregressive models', *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations* **13**(2), 53–72.

Chang, G., Zhang, Y., Yao, D. and Yue, Y. (2011), A summary of short-term traffic flow forecasting methods, *in* 'Proceedings of the 11th International Conference of Chinese Transportation Professionals', pp. 1696–1707.

Chen, H. and Grant-Muller, S. (2001), 'Use of sequential learning for short term traffic flow forecasting', *Transportation Research Part C: Emerging Technologies* **9**(5), 319–336.

Chen, H., Grant-Muller, S., Mussone, L. and Montgomery, F. (2001), 'A study of hybrid neural network approaches and the effects of missing data on traffic forecasting', *Neural Computing and Applications* **10**(3), 277–286.

Chib, S. and Greenberg, E. (1994), 'Bayes inference in regression models with ARMA(p, q) errors', *Journal of Econometrics* **64**(1-2), 183–206.

Clark, S. (2003), 'Traffic prediction using multivariate nonparametric regression', *Journal of Transportation Engineering* **129**(2), 161–168.

Cox, D. (2006), *Principles of statistical inference*, Cambridge University Press.

Davison, A. (2003), *Statistical models*, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press.

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 Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)', Morgan Kaufmann, San Francisco, CA, pp. 176–183.

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. (2011), A tutorial on particle filtering and smoothing: fifteen years later, *in* 'Oxford handbook of nonlinear filtering', pp. 656–704.

Dougherty, M. S. and Cobbett, M. R. (1997), 'Short term inter-urban traffic forecasts using neural networks', *International Journal of Forecasting* **13**(1), 21–31.

Fearnhead, P. (2002), 'Markov chain Monte Carlo, sufficient statistics, and particle filters', *Journal of Computational and Graphical Statistics* **11**(4), 848–862.

Flegal, J. and Jones, G. (2011), Implementing Markov chain Monte Carlo: estimating with confidence, *in* S. Brooks, A. Gelman, G. L. Jones and X.-L. Meng, eds, 'Handbook of Markov chain Monte Carlo', Boca Raton, FL: CRC Press, pp. 175–197.

Flegal, J. M. (2012), *mcmcse: Monte Carlo standard errors for MCMC.*
URL: *http://CRAN.R-project.org/package=mcmcse*

Flegal, J. M., Jones, G. L. and Neath, R. C. (2012), Markov chain Monte Carlo estimation of quantiles. arXiv:1207.6432.

Gazis, D. and Liu, C. (2003), 'Kalman filtering estimation of traffic counts for two network links in tandem', *Transportation Research Part B: Methodological* **37**(8), 737–745.

Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. (2003), *Bayesian data analysis*, 2nd edn, Chapman and Hall/CRC.

Gelman, A. and Hill, J. (2007), *Data analysis using regression and multilevel/hierarchical models*, Analytical Methods for Social Research, Cambridge University Press.

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* **57**(6), 1317–39.

Geyer, C. J. (2011), Introduction to Markov chain Monte Carlo, *in* S. Brooks, A. Gelman, G. L. Jones and X.-L. Meng, eds, 'Handbook of Markov chain Monte Carlo', Boca Raton, FL: CRC Press, pp. 3–48.

Ghosh, B., Basu, B. and O'Mahony, M. (2007), 'Bayesian time-series model for short term traffic flow forecasting', *Journal of Transportation Engineering* **133**(3), 180–189.

Ghosh, B., Basu, B. and O'Mahony, M. (2009), 'Multivariate short term traffic flow forecasting using time-series analysis', *IEEE Transactions on Intelligent Transportation Systems* **10**, 246–254.

Ghosh, S. and Lee, T. (2000), *Intelligent transportation systems: new principles and architectures*, Mechanical Engineering Handbook Series, Taylor & Francis Group.

Gilks, W. R., Best, N. G. and Tan, K. K. C. (1995), 'Adaptive rejection Metropolis sampling within Gibbs sampling', *Applied Statistics* **44**(4), 455–472.

Gilks, W. R. and Wild, P. (1992), 'Adaptive rejection sampling for Gibbs sampling', *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **41**(2), 337–348.

Gilks, W., Richardson, S. and Spiegelhalter, D. (1996), *Markov chain Monte Carlo in practice: interdisciplinary statistics*, Interdisciplinary Statistics Series, Chapman & Hall.

Grimmett, G. R. and Stirzaker, D. R. (2001), *Probability and random processes*, 3rd edn, Oxford University Press, USA.

Hamed, M., Al-Masaeid, H. and Said, Z. (1995), 'Short term prediction of traffic volume in urban arterials', *Journal of Transportation Engineering* **121**(3), 249254.

Härdle, W. (2004), *Nonparametric and semiparametric models*, Springer Series in Statistics Series, Springer-Verlag.

Hastings, W. K. (1970), 'Monte Carlo sampling methods using Markov chains and their applications', *Biometrika* **57**(1), 97–109.

Haykin, S. (2001), *Kalman filtering and neural networks*, Wiley-Interscience.

Holan, S. H., Lund, R. and Davis, G. (2010), 'The ARMA alphabet soup: a tour of ARMA model variants', *Statistical Surveys* **4**, 232–274.

Hu, P., Tian, Z., Yang, F. and Johnson, L. (2011), Short-term traffic flow prediction based on time series analysis, *in* 'Proceedings of the 11th International Conference of Chinese Transportation Professionals', pp. 3987–3996.

Ionides, E. L., Bhadra, A., Atchadé, Y. and King, A. (2011), 'Iterated filtering.', *Annals of Statistics* **39**(3), 1776–1802.

Jeffreys, H. (1946), 'An invariant form for the prior probability in estimation problems', *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* **186**(1007), 453–461.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S. and Saul, L. K. (1999), 'An introduction to variational methods for graphical models', *Mach. Learn.* **37**(2), 183–233.

Julier, S. J. and Uhlmann, J. K. (1997), A new extension of the Kalman filter to nonlinear systems, pp. 182–193.

Kalman, R. E. (1960), 'A new approach to linear filtering and prediction problems', *Transactions of the ASME - Journal of Basic Engineering (Series D)* **82**, 35–45.

Kantas, N., Doucet, A., Singh, S. and Maciejowski, J. (2009), An overview of sequential Monte Carlo methods for parameter estimation in general state-space models, *in* 'Proceedings of the IFAC Symposium on System Identification (SYSID)'.

Kass, R. E. and Raftery, A. E. (1995), 'Bayes factors', *Journal of the American Statistical Association* **90**(430), 773–795.

Laplace, P. S. (1774), 'Memoire sur la probabilité des causes par les évènements', Memoires de mathématique et de physique presentés a l'académie royale des sciences, par divers savants, et lûs dans ses assemblées. Reprinted in Laplace's Oeuvres Complètes 8 27-65. English translation by S Stigler (1986) *Statistical Science* **1** 359-378.

Leon-Garcia, A. (2008), *Probability, statistics, and random processes for electrical engineering*, 3rd edn, Pearson/Prentice Hall/Pearson Education international.

Little, R. and Rubin, D. (2002), *Statistical analysis with missing data*, Wiley series in probability and mathematical statistics. Probability and mathematical statistics, Wiley.

Liu, J. and West, M. (2001), Combined parameter and state estimation in simulation-based filtering, *in* 'Sequential Monte Carlo Methods in Practice', New York: Springer-Verlag, pp. 197–217.

MacKay, D. J. C. (2002), *Information theory, inference & learning algorithms*, Cambridge University Press, New York, NY, USA.

Mai, T., Ghosh, B. and Wilson, S. (2012), Multivariate short term traffic flow forecasting using Bayesian vector autoregressive moving average model, *in* 'Transportation Research Board 91st Annual Meeting Compendium of Papers DVD', Washington DC, pp. 1–16.

Mai, T., Ghosh, B. and Wilson, S. (2013), 'Short term traffic flow forecasting with A-SVARMA', *Proceedings of The Institution of Civil Engineers - Transport* . Accepted.

Marriott, J., Ravishanker, N., Gelfand, A. and Pai, J. (1996), Bayesian analysis of ARMA processes: complete sampling-based inference under exact likelihoods, *in* D. Berry, K. Chaloner and J. Geweke, eds, 'Bayesian analysis in statistics and econometrics: essays in honor of Arnold Zellner', John Wiley & Sons, pp. 243–256.

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.

Meyn, S. and Tweedie, R. L. (2009), *Markov chains and stochastic stability*, 2nd edn, Cambridge University Press, New York, NY, USA.

Min, W. and Wynter, L. (2011), 'Real-time road traffic prediction with spatio-temporal correlations', *Transportation Research Part C: Emerging Technologies* **19**(4), 606 – 616.

Monahan, J. (1983), 'Fully Bayesian analysis of ARMA time series models', *Journal of Econometrics* **21**(3), 307–331.

Monahan, J. (1984), 'A note on enforcing stationarity in autoregressive-moving average models', *Biometrika* **71**(2), 403–404.

Monahan, J. (2011), *Numerical methods of statistics*, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press.

Murphy, K. and Russell, S. (2001), Rao-Blackwellised particle filtering for dynamic Bayesian networks, *in* N. d. F. A. Doucet and N. Gordon, eds, 'Sequential Monte Carlo Methods in Practice', Springer-Verlag, pp. 499–515.

Niemi, J. (2010), 'Adaptive mixture modeling Metropolis methods for Bayesian analysis of nonlinear state-space models', *Journal of Computational and Graphical Statistics* **19**(2), 260–280.

Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A. and Wang, Y. (2003), 'Review of road traffic control strategies', *Proceedings of the IEEE* **91**(12), 2043–2067.

Papageorgiou, M. and Vigos, G. (2008), 'Relating time-occupancy measurements to space-occupancy and link vehicle-count', *Transportation Research Part C: Emerging Technologies* **16**(1), 1–17.

Pitt, M. K. and Shephard, N. (1999), 'Filtering via simulation: auxiliary particle filters', *Journal of the American Statistical Association* **94**(446), 590–599.

Prado, R. and West, M. (2010), *Time series: modeling, computation, and inference*, Chapman & Hall.

Queen, C. and Albers, C. (2009), 'Intervention and causality: forecasting traffic flows using a dynamic Bayesian network', *Journal of the American Statistical Association* **104**(486), 669–681.

Queen, C. M., Wright, B. J. and Albers, C. J. (2007), 'Eliciting a directed acyclic graph for a multivariate time series of vehicle counts in a traffic network', *Australian & New Zealand Journal of Statistics* **49**(3), 221–239.

Ravishanker, N. and Ray, B. K. (1997), 'Bayesian analysis of vector ARMA models using Gibbs sampling', *Journal of Forecasting* **16**(3), 177–194.

Roberts, G. O. and Rosenthal, J. S. (2004), 'General state space Markov chains and MCMC algorithms', *Probability Surveys* **1**, 20–71.

Roess, R., Prassas, E. and McShane, W. (2004), *Traffic engineering*, Pearson/Prentice Hall.

Rue, H., Martino, S. and Chopin, N. (2009), 'Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **71**(2), 319–392.

Shephard, N. and Pitt, M. K. (1997), 'Likelihood analysis of non-Gaussian measurement time series', *Biometrika* **84**, 653–667.

Shumway, R. H. and Stoffer, D. S. (2006), *Time series analysis and its applications: with R examples (Springer Texts in Statistics)*, 2nd edn, Springer.

Singh, K. and Li, B. (2012), 'Estimation of traffic densities for multilane roadways using a Markov model approach', *IEEE Transactions on Industrial Electronics* **59**(11), 4369–4376.

Slinn, M., Matthews, P. and Guest, P. (2005), *Traffic engineering design: principles and practice*, Arnold.

Smith, B. L., Williams, B. M. and Oswald, R. K. (2002), 'Comparison of parametric and nonparametric models for traffic flow forecasting', *Transportation Research Part C: Emerging Technologies* **10**(4), 303–321.

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.

Sun, S., Zhang, C. and Yu, G. (2006), 'A bayesian network approach to traffic flow forecasting', *IEEE Transactions on Intelligent Transportation Systems* **7**(1), 124–132.

Tebaldi, C., West, M. and Karr, A. F. (2002), 'Statistical analyses of freeway traffic flows', *Journal of Forecasting* **21**(1), 39–68.

Vigos, G. and Papageorgiou, M. (2010), 'A simplified estimation scheme for the number of vehicles in signalized links', *IEEE Transactions on Intelligent Transportation Systems* **11**(2), 312–321.

Vlahogianni, E. I., Golias, J. C. and Karlaftis, M. G. (2004), 'Short term traffic forecasting: overview of objectives and methods', *Transport Reviews: A Transnational Transdisciplinary Journal* **24**(5), 533–557.

Vlahogianni, E. I., Karlaftis, M. G. and Golias, J. C. (2005), 'Optimized and meta-optimized neural networks for short term traffic flow prediction: a genetic approach', *Transportation Research Part C: Emerging Technologies* **13**(3), 211 – 234.

Wan, E. A. and Merwe, R. V. D. (2000), The unscented Kalman filter for nonlinear estimation, pp. 153–158.

Weinberg, M. D. (2012), 'Computing the Bayes factor from a Markov chain Monte Carlo simulation of the posterior distribution', *Bayesian Analysis* **7**(3), 737–770.

West, M. (1993), 'Approximating posterior distributions by mixtures', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **54**, 553–568.

West, M. and Harrison, J. (1997), *Bayesian forecasting and dynamic models*, Springer Series in Statistics, Springer.

Williams, B. M. and Hoel, L. A. (2003), 'Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: theoretical basis and empirical results', *Journal of Transportation Engineering* **129**(6), 664–672.

Wood, S. (2006), *Generalized additive models: an introduction with R*, Chapman and Hall/CRC Texts in Statistical Science Series, Chapman and Hall/CRC Press.

Zhang, H. (2000), 'Recursive prediction of traffic conditions with neural network models', *Journal of Transportation Engineering* **126**(6), 472–481.