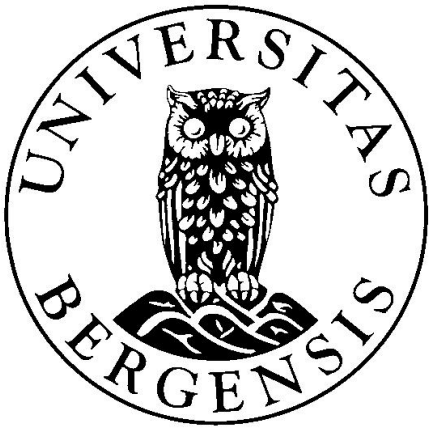
Parameter Estimation of Multivariate Factor Stochastic Volatility Models



Jens Christian Wahl

Department of Mathematics

University of Bergen

A thesis submitted in partial fulfillment of the requirements for the degree

Master of Science

Financial Theory and Insurance Mathematics

June 2018

Acknowledgements

I would first like to thank my advisor, Professor Hans J. Skaug for introducing me to the topic and for all the help writing this thesis. He has helped me a great deal to understanding this topic and has patiently answered my many questions.

I would also like to thank Berent Lunde for all the discussions regarding the technical details of this thesis, especially with C++. Thanks to Sean Murray and Tommy Odland for reading drafts of the thesis and giving me valuable feedback. Finally, I would like to thank family and friends for their moral support and words of encouragement throughout the study period.

Abstract

Volatility is a crucial aspect of risk management and important to accurately quantify. A broad range of models and methods tackle this problem, but there is no consensus to exactly which method or model that solves this problem best. We use maximum likelihood and Hamiltonian Monte Carlo to estimate parameters in multivariate factor stochastic volatility models and compare the two alternative methods with the new interweaving strategy proposed in Kastner et al. (2017). Through simulation studies, we show that convergence of the likelihood is unstable and very data dependent. We investigate possible restrictions on our parameters by calculating the characteristic function of our model. We find that restricting the loading matrix (in two dimensions) makes convergence more stable. Furthermore, we introduce the "Nested Laplace Approximation" (where we integrate over the latent variables in a sequential way) and compare it to the classical Laplace approximation on two state space models. We also compare the methods on exchange data from 2005-2015. All methods give similar results, but Hamiltonian Monte Carlo is sensitive to the choice of priors.

Table of contents

Chapter 1

Introduction

Volatility, a measure of uncertainty in financial returns, is an important factor when quantifying risk. Empirical studies show that volatility varies with time, is autocorrelated and therefore has a tendency to appear in clusters. It is therefore important to develop statistical models that captures this behaviour.

There are two popular classes of models typically used to tackle the problem of modelling volatility. The first is ARCH/GARCH models, introduced by Engle (1982) and Bollerslev (1986), where the volatility is captured by letting the conditional variance be a function of the squares of previous observations and past variance. The second is Stochastic Volatility models (SV) (Taylor (1982)), where one models the volatility as an unobserved process. We will focus on the latter, and in particular on so called Multivariate Factor Stochastic Volatility Models (MFSV), which combine classical factor analysis with several independent univariate SV models, allowing the latent factors to have time-varying variance.

Parameter estimation of stochastic volatility models is hard due to the fact the likelihood function is expressed as a high dimensional integral over the latent variables that cannot be evaluated analytically. If $\boldsymbol{y}$ denotes our observations, $\boldsymbol{u}$ our latent variables and $\boldsymbol{\theta}$ the parameters of interest, the likelihood of $\boldsymbol{\theta}$ is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \int f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u}) f_{\boldsymbol{u}}(\boldsymbol{u})\, d\boldsymbol{u}. \tag{1.1}$$

A variety of methods have been proposed for stochastic volatility models, such as simulated likelihood (Liesenfeld (2006)), quasi-maximum likelihood (Harvey et al. (1994)) and Bayesian Markov Chain Monte Carlo (MCMC) methods (Pitt and Shephard (1999), Kastner et al. (2017)).

In a recent paper, Kastner et al. (2017) proposes a Gibbs sampler that utilizes an ancillarity-sufficiency interweaving strategy (ASIS) introduced by Yu and Meng (2011) for sampling MFSV models, where different data augmentation schemes are combined to obtain efficient sampling. In this thesis we take a maximum likelihood (ML) approach, where the Laplace approximation is used to evaluate the integral in Equation (1.1). This has earlier been done for the univariate and multivariate basic SV model (Skaug and Yu (2014)), but not for MFSV models. A ML approach can be motivated by both statistical efficiency and by the fact that ML often requires less computational power than MCMC.

Hamiltonian Monte Carlo (HMC) (Neal (2010), Neal (1996)), where Hamiltonian dynamics is used to propose new states, is also considered as an alternative MCMC algorithm. This has not been investigated earlier to our knowledge. HMC is efficient since it is possible to avoid random walk proposals and thus reduce the correlation between states in the Markov Chain.

Identification issues is a known problem for factor models (Harvey (1991), Frühwirth-Schnatter and Lopes (2010)). It's therefore expected that this can cause problems for likelihood estimation. To better understand the structure of the model, we investigate the characteristic function and the cumulative generating function to find possible restrictions on our parameters. We also introduce the "Nested Laplace approximation", where we integrate over the latent variables in a sequential way, as an alternative to the standard Laplace approximation.

This thesis is structured as follows: Chapter 2 introduces the classical factor model and introduces both the ARCH/GARCH model and the basic SV model. The general theory of ML, MCMC and HMC is discussed in chapter 3, 4 and 5. Chapter 6 give an introduction to Automatic Differentiation (AD), a technique for evaluating derivatives

of functions, the R package Template Model Builder (**TMB**), used for ML and the probabilistic programming language **Stan**, used for HMC. Chapter 7 introduce the MFSV model and discusses some of its properties. An overview of the sampling algorithm proposed in Kastner et al. (2017) (deep interweaving MCMC (DIMCMC)) is also given. A simulation study comparing ML and DIMCMC is presented in chapter 8 and the characteristic function and cumulative generating function of our model is investigated for possible restrictions on our parameters. Chapter 9 introduces the "Nested Laplace approximation", an alternative to the classical Laplace approximation. In chapter 10 we apply the different estimation methods to exchange rates. Chapter 11 briefly discusses other methods investigated in this thesis, and finally, chapter 12 concludes.

Chapter 2

Factor Analysis and Stochastic Volatility

2.1 Factor Analysis

We will follow Jolliffe (1986), Lawley and Maxwell (1971) and Tsay (2010) in this section.

Factor analysis is a statistical technique used to capture the correlation structure of multivariate data. It was initially developed by psychologists and focused on capturing the underlying general mental ability that entered in to a variety of mental tests. Charles Spearman (Spearman (1904)) was trying to discover the hidden structure of human intelligence. He observed that schoolchildren's grades in different subjects were all correlated. He also observed a particular pattern of correlations which he though could explain why grades in different subjects where correlated. It was because they where all correlated with something else, a general or common factor, which he called "general intelligence", denoted G. Spearman introduced what is known as a 1-factor model. We will in this section introduce the classical q-factor model that will be the building block of the dynamical factor model discussed in chapter 8.

A common problem in multivariate analysis is the so called "curse of dimensionality". As the dimensionality of the data increases, the number of parameters often increases

polynomially or exponentially. Take the covariance matrix as an example. Increasing the dimensionality of the data from p to $p+1$, increases the number of parameters in the covariance matrix by $p+1$. Factor analysis tries to solve this problem by imposing a lower dimensional latent structure on the covariance matrix.

Given a multivariate random variable $\boldsymbol{y} = (y_1, \ldots, y_p)$, the idea of factor analysis is that these variables can be expressed as linear functions of $q < p$ latent random variables (also called common factors) and an error term $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_p)$. If $f_1, \ldots, f_q$ denotes factors, then the factor model is defined as

$$\begin{aligned} y_1 &= \beta_{11} f_1 + \beta_{12} f_2 + \cdots + \beta_{1q} f_q + \epsilon_1 \\ y_2 &= \beta_{21} f_1 + \beta_{22} f_2 + \cdots + \beta_{21} f_q + \epsilon_2 \\ &\vdots \\ y_p &= \beta_{p1} f_1 + \beta_{p2} f_2 + \cdots + \beta_{pq} f_q + \epsilon_p. \end{aligned} \tag{2.1}$$

For $j = 1, \ldots, p$ and $k = 1, \ldots, q$, β_{jk} are constants known as factor loadings and ϵ_j are error terms, called specific factors or idiosyncratic noise, because ϵ_j is specific to the variable y_j. We can rewrite Equation (2.1) more compactly in matrix notation as

$$\boldsymbol{y} = \boldsymbol{\beta f} + \boldsymbol{\epsilon}. \tag{2.2}$$

A number of assumptions need to be made:

1. $\mathbb{E}(\boldsymbol{\epsilon}) = \mathbf{0}$,
2. $\mathbb{E}(\boldsymbol{f}) = \mathbf{0}$,
3. $\mathbb{E}(\boldsymbol{y}) = \mathbf{0}$,
4. $\mathbb{E}(\boldsymbol{\epsilon\epsilon'}) = \boldsymbol{\Psi}$ (Diagonal matrix),
5. $\mathbb{E}(\boldsymbol{f\epsilon'}) = \mathbf{0}$ (Zero matrix),
6. $\mathbb{E}(\boldsymbol{ff'}) = \boldsymbol{I}$ (Identity matrix),

Assumptions (1) and (4) states that the idiosyncratic error has expectation zero and are uncorrelated. Assumptions (2) and (3) is just for convenience, since we can always center our variables. If $\boldsymbol{y}$ has mean $\boldsymbol{\mu}$, we can simple replace $\boldsymbol{y}$ with $\boldsymbol{y}-\boldsymbol{\mu}$. Assumptions (5) and (6) states that the factors and idiosyncratic errors are uncorrelated and that the factors have unit variance. Since $\boldsymbol{\Psi}$ is diagonal, *all* the systematic patterns in $\boldsymbol{y}$ should be captured in $\boldsymbol{\beta f}$, and $\boldsymbol{\epsilon}$. It should also be noted that assumption (6) can be relaxed, so that the common factors may be correlated rather than uncorrelated. We will not go into the theory of correlated factors, since we assume that they are uncorrelated in all cases.

In practice $\boldsymbol{\beta}$ and $\boldsymbol{\Psi}$ are unknown and need to be estimated. This can be done by maximum likelihood estimation and is implemented as a part of Base R.

Given the assumptions, we can find the covariance matrix of $\boldsymbol{y}$:

$$\text{Cov}(\boldsymbol{y}) = \boldsymbol{\Sigma} = \text{Cov}(\boldsymbol{\beta f} + \boldsymbol{\epsilon}) = \boldsymbol{\beta I \beta}' + \boldsymbol{\Psi} = \boldsymbol{\beta\beta}' + \boldsymbol{\Psi} \tag{2.3}$$

It we inspect Equation (2.3), we observe that there are an infinite number of solutions. Let $\boldsymbol{Q}$ be any $q \times q$ orthogonal matrix, such that $\boldsymbol{QQ}' = \boldsymbol{I}$, where $\boldsymbol{Q}'$ is the transpose of $\boldsymbol{Q}$. If we replace $\boldsymbol{\beta}$ by $\widetilde{\boldsymbol{\beta}} = \boldsymbol{\beta Q}$, then

$$\widetilde{\boldsymbol{\beta}}\widetilde{\boldsymbol{\beta}}' = (\boldsymbol{\beta Q})(\boldsymbol{\beta Q})' = \boldsymbol{\beta}(\boldsymbol{QQ}')\boldsymbol{\beta}' = \boldsymbol{\beta\beta}'.$$

In the context of factor analysis this transformation corresponds to a rotation of the factors, and the "best" one is chosen according to some criteria.

2.2 Volatility Models

Time varying volatility is one of the characteristics of financial returns. It is also known that the volatility is autocorrelated, leading to so called *volatility clusters* (Pitt and Shephard (1999)). This means if we observe small changes in the price today, it is

often followed by a small change tomorrow, while big changes more often is followed by a big change.

The issue of modelling returns accounting for time-varying volatility has been widely analyzed in the financial econometrics literature. The problem is usually analyzed with two classes of models, namely the ARCH/GARCH models and the Stochastic Volatility (SV) models. We will first discuss ARCH/GARCH models. The main references for this section is Tsay (2010) and Shephard (2005).

2.2.1 ARCH/GARCH Models

Autoregressive Conditional Heteroskedasticity models (ARCH) is a class of stochastic processes which are widely used to estimate heteroskedasticity and volatility clustering in the financial industry. The ARCH(p) model was introduced by Engle (1982) and is given by

$$z_t = \sqrt{h_t}\epsilon_t, \tag{2.4}$$

$$h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i z_{t-i}^2 \tag{2.5}$$

where ϵ_t is a sequence of independent and identically distributed (i.i.d.) random variables with zero mean and unit variance, $\alpha_0 > 0$ and $\alpha_i \geq 0$ for $i > 0$.

The conditions put on the coefficients are to secure stationarity of z_t. The error term ϵ_t is usually assumed to follow a normal distribution, a t-distribution or a Generalized Error Distribution (GED). We can observe that the conditional variance of z_t given the information at time $t-1$, denoted by $\mathscr{F}_{t-1}$, is given by

$$\mathrm{Var}(z_t|\mathscr{F}_{t-1}) = \mathbb{E}(h_t\epsilon_t^2) = h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i z_{t-i}^2.$$

From the expression above, we can observe that if the returns in the past are big, the conditional variance h_t for today's observation will be big. The same holds true

for small returns. Thus, the conditional variance captures clusters of high and low volatility.

Tsay (2010) also point out some of the weaknesses of the ARCH model:

- Positive and negative returns have the same effect on the volatility because it depends on the square of the returns. In practice, the price of financial assets responds differently to positive and negative returns.
- ARCH model does not provide us with any new insight for the sources of the volatility, but only a deterministic method to describe the conditional variance.
- It tends to overpredict the volatility because it responds slowly to large isolated returns.

A few years later Bollerslev (1986) introduced an extension of the ARCH(p) model, known as the GARCH(p, q). The GARCH(p, q) model is defined as

$$z_t = \sqrt{h_t}\epsilon_t, \tag{2.6}$$

$$h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i z_{t-i}^2 + \sum_{j=1}^{q} \beta_j h_{t-j}, \tag{2.7}$$

where ϵ_t is a sequence of i.i.d. random variables with zero mean and unit variance, $\alpha_0 > 0$, $\alpha_i \geq 0$, $\beta_j \geq 0$, and $\sum_{i=1}^{\max(p,q)}(\alpha_i + \beta_i) < 1$. Again, the restrictions on the parameters is to ensure stationarity. The ϵ_t usually follows a normal distribution, t-distribution or a GED. The conditional variance of z_t is given by

$$\text{Var}(z_t|\mathscr{F}_{t-1}) = \mathbb{E}(h_t\epsilon_t^2|\mathscr{F}_{t-1}) = h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i z_{t-i}^2 + \sum_{j=1}^{q} \beta_j h_{t-j}. \tag{2.8}$$

We can observe that in comparison to the ARCH model, the conditional variance of z_t depends not only on the square of earlier returns, but also on the variance of earlier returns. If $\beta_j = 0$, the model is reduced to a ARCH(p). Just as the ARCH model, the GARCH process suffers from responding the same way to both positive and negative earlier returns, and empirical studies indicate that the tail behavior of GARCH models

remains too short even when ϵ_t follows a standardized t-distribution (Tsay (2010)). In the next subsection, we will see that in comparison to the ARCH/GARCH models, SV models treat the volatility as a latent stochastic process.

2.2.2 The Basic Stochastic Volatility Model

In contrast to ARCH/GARCH models, stochastic volatility (SV) models model the volatility as a latent process. The basic SV model is defined by

$$\begin{aligned} y_t &= \sigma_y e^{h_t/2}\epsilon_t, \quad t = 1, \dots, T, \\ h_{t+1} &= \phi h_t + \sigma\eta_t, \quad t = 1, \dots, T-1, \end{aligned} \tag{2.9}$$

where y_t is the observed log returns, h_t is the logarithm of the variance on day t and $\epsilon_t, \eta_t \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$. To ensure stationarity for h_t, we assume $|\phi| < 1$. It can be shown that the unconditional distribution of h_t is $\mathcal{N}(0, \sigma^2/(1-\phi^2))$, and we assume $h_1 \sim \mathcal{N}(0, \sigma^2/(1-\phi^2))$. A interpretation of the latent process $\{h_t\}$ is that is represents the random and uneven flow of new information into the marked (Pitt and Shephard (1999)). For different time points, the variance will be dependent of this unobserved "flow" of information, i.e. conditioning on h_t, $\text{Var}(y_t|h_t) = \sigma_x^2 e^{h_t}$.

A characteristic of volatility clustering is that the squared returns are autocorrelated. We therefore derive some of the properties of y_t^2. We first need the following proposition:

Proposition 2.1. *Let $\boldsymbol{Z} = (Z_1, \dots, Z_k)$ be a multivariate normal variable with expectation $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma} = \{\sigma_{ij}\}$. Let $\boldsymbol{U} = \exp(\boldsymbol{Z}) = (\exp(Z_1), \dots \exp(Z_k))$. Then $\mathbb{E}(U_i) = \exp(\mu_i + \frac{1}{2}\sigma_{ii})$ and $\text{Cov}(U_i, U_j) = \mathbb{E}(U_i)\mathbb{E}(U_j)(\exp(\sigma_{ij}) - 1)$ for $i, j = 1, \dots, k$.*

Proof. Let $\boldsymbol{a} \in \mathbb{R}^k$. Then

$$\begin{aligned}\mathbb{E}\left(\prod_{j=1}^{k} U_j^{a_j}\right) &= \mathbb{E}\left(\prod_{j=1}^{k} \exp\left(a_j Z_j\right)\right) \\ &= \mathbb{E}\left(\exp\left(\sum_{j=1}^{k} a_j Z_j\right)\right) = \mathbb{E}\left(\exp\left(\boldsymbol{a}'\boldsymbol{Z}\right)\right).\end{aligned} \tag{2.10}$$

Using the fact that the moment generating function for a multivariate normal variable $\boldsymbol{X}$ is given by

$$M_{\boldsymbol{X}}(\boldsymbol{t}) = \mathbb{E}\left(\exp\left(\boldsymbol{t}'\boldsymbol{X}\right)\right) = \exp\left(\boldsymbol{t}'\boldsymbol{\mu} + \frac{1}{2}\boldsymbol{t}'\boldsymbol{\Sigma}\boldsymbol{t}\right),$$

we get

$$\mathbb{E}\left(\exp\left(\boldsymbol{a}'\boldsymbol{Z}\right)\right) = \exp\left(\boldsymbol{a}'\boldsymbol{\mu} + \frac{1}{2}\boldsymbol{a}'\boldsymbol{\Sigma}\boldsymbol{a}\right). \tag{2.11}$$

For $i = 1, \ldots, k$, we choose $\boldsymbol{a}$ such that the ith element is equal to one and zero otherwise and obtain $\mathbb{E}(U_i) = \exp\left(\mu_i + \frac{1}{2}\sigma_{ii}\right)$. Now, choose $\boldsymbol{a} \in \mathbb{R}^k$ such that for $i, j = 1, \ldots, k$ the ith and jth element is equal to one and zero otherwise. It then follows from Equation (2.11) that

$$\mathbb{E}(U_i U_j) = \exp\left(\mu_1 + \mu_2 + \frac{1}{2}\sigma_{ii} + \sigma_{ij} + \frac{1}{2}\sigma_{jj}\right) = \mathbb{E}(U_i)\mathbb{E}(U_j)\exp\left(\sigma_{ij}\right),$$

and thus $\mathrm{Cov}(U_i, U_j) = \mathbb{E}(U_i)\mathbb{E}(U_j)(\exp\left(\sigma_{ij}\right) - 1)$ □

Using proposition 2.1 and assuming finite fourth moment, we can find the the first to even moments of y_t (Hautsch and Ou (2008)):

$$\mathbb{E}(y_t^2) = \sigma_y^2 \mathbb{E}(e^{h_t}) = \sigma_y^2 \exp\left(\frac{\sigma^2}{2(1-\phi^2)}\right), \tag{2.12a}$$

$$\mathbb{E}(y_t^4) = \sigma_y^4 \mathbb{E}(\epsilon^4 e^{2h_t}) = 3\sigma_y^4 \mathbb{E}(e^{2h_t}) = 3\sigma_y^4 \exp\left(\frac{2\sigma^2}{(1-\phi^2)}\right). \tag{2.12b}$$

Combining these we get

$$\mathrm{Var}(y_t^2) = 3\sigma_y^4 \exp\left(\frac{\sigma^2}{1-\phi^2}\right)\left(\exp\left(\frac{\sigma^2}{1-\phi^2}\right) - 1\right). \tag{2.13}$$

Thus, y_t is a non-Gaussian weakly stationary [1] time series. Using Equation (2.12) the kurtosis of y_t is

$$K(y_t) \coloneqq \frac{\mathbb{E}(y_t^4)}{\mathbb{E}(y_t^2)^2} = \frac{3\sigma_y^4 \exp\left(\frac{2\sigma^2}{(1-\phi^2)}\right)}{\sigma_y^4 \exp\left(\frac{\sigma^2}{2(1-\phi^2)}\right)} = 3\exp\left(\frac{\sigma^2}{1-\phi^2}\right). \tag{2.14}$$

This implies that $K(y_t) > 3$ as long as $\sigma^2 > 0$ and is increasing as $|\phi|$ and σ^2 increases (assuming $|\phi| < 1$), making the distribution more "heavy-tailed" compared to the normal distribution.

To find the autocorrelation function (ACF[2]) of y_t^2 we can apply proposition 2.1:

$$\begin{aligned}
\mathrm{Cov}(y_{t+\tau}^2, y_t^2) &= \mathbb{E}(y_{t+\tau}^2 y_t^2) - \mathbb{E}(y_{t+\tau}^2)\mathbb{E}(y_t^2) \\
&= \sigma_y^4\Big(\mathbb{E}(\exp(h_{t+\tau})\exp(h_t)) - \mathbb{E}(\exp(h_{t+\tau}))\mathbb{E}(\exp(h_t)\Big) \quad \text{(Independence of } \epsilon_t\text{)} \\
&= \sigma_y^4 \mathrm{Cov}(\exp(h_{t+\tau}), \exp(h_t)) \\
&= \sigma_y^4 \mathbb{E}\Big(\exp(h_{t+\tau})\Big)\mathbb{E}\Big(\exp(h_t)\Big)\Big(\exp\Big(\mathrm{Cov}(h_{t+\tau}, h_t\Big) - 1\Big) \quad \text{By prop. (2.1)} \\
&= \sigma_y^4 \exp\left(\frac{\sigma^2}{1-\phi^2}\right)\left(\exp\left(\phi^\tau \frac{\sigma^2}{1-\phi^2}\right) - 1\right)
\end{aligned} \tag{2.15}$$

Dividing Equation (2.15) by Equation (2.13) we obtain the ACF:

$$\rho(\tau) = \frac{\exp\left(\phi^\tau \frac{\sigma^2}{1-\phi^2}\right) - 1}{3\exp\left(\frac{\sigma^2}{1-\phi^2}\right) - 1}, \quad \tau = 0, 1, \ldots. \tag{2.16}$$

Thus, for $\phi \in (0, 1)$ the basic SV model predicts a positive autocorrelation function that is exponentially decaying in τ for squared returns.

[1] A stochastic process $\{X_t\}$ is said to be weakly stationary if $\mathbb{E}(X_t)$ is independent of t and $\mathrm{Cov}(X_{t+h}, X_t)$ is independent of t for each h.

[2] The ACF of a stationary time series $\{X_t\}$ is defined as $\rho(h) = \frac{\mathrm{Cov}(X_{t+h}, X_t)}{\mathrm{Var}(X_t)}$

As we will see in chapter 7, the Multivariate Factor Stochastic Volatility Model is a result of combining the factor model in section 2.1 with several independent SV models from this section.

Chapter 3

Likelihood and Marginal Likelihood

Parameter estimation of stochastic volatility models can be a challenging task and has been subject of much research. A variety of methods have been proposed, such as simulated likelihood (Liesenfeld (2006)), quasi-maximum likelihood (Harvey et al. (1994)) and Bayesian MCMC methods (Pitt and Shephard (1999),Kastner et al. (2017)). In this thesis we focus on the maximum likelihood framework, and this chapter will serve as an introduction to the topic. We will mostly follow Casella and Berger (2001), Rice (1988) and Pawitan (2001), which give good introductions. For a more rigorous and theoretical treatment, the reader is referred to Schervish (1996).

Let $f_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{\theta})$ denote the joint pdf or pmf of the sample $\boldsymbol{X} = (X_1, \ldots, X_n)$. Then, given that $\boldsymbol{X} = \boldsymbol{x}$, the function $\mathcal{L}(\boldsymbol{\theta}) = f_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{\theta})$ is called the *likelihood function*.

We can interpret the likelihood function as how well different values of the parameter explain the observed data. This gives rise to the concept of the *maximum likelihood estimate* (MLE), which is the value of $\boldsymbol{\theta}$ that maximizes the likelihood function. We define the MLE as

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{3.1}$$

If we assume that the likelihood function is differentiable w.r.t $\boldsymbol{\theta}$, possible candidates for the MLE are all points $\boldsymbol{\theta_i}, i = 1, \ldots, k$ such that

$$\nabla \mathcal{L}(\boldsymbol{\theta_i}) = \mathbf{0}. \tag{3.2}$$

In practice, we are often working with the logarithm of the likelihood, $l = \log \mathcal{L}$, due to the fact that it is often easier to maximize the logarithm of the likelihood than the likelihood itself. Since the logarithm is a strictly increasing function, the maximum of l will also be the maximum of $\mathcal{L}$.

3.1 Properties of MLE

We will now give a sketch of the proofs of two nice properties of MLE, namely consistency and asymptotic normality. For simplicity, we assume that our observations are i.i.d. (Pawitan (2001)).

3.1.1 Consistency

One of the most appealing properties of the maximum likelihood estimator is consistency, meaning that as the sample size increases the sequence of estimators $\{\hat{\theta}_n\}_{n=1}^{\infty}$ converges to the true parameter value θ_0.

To prove consistency, we need the following theorem from Pawitan (2001).

Theorem 3.1 (Information inequality). *If f and g are two densities, then*

$$\mathbb{E}_g\left(\log \frac{g(X)}{f(X)}\right) \geq 0$$

with strict inequality unless $f = g$. $\mathbb{E}_g$ means that the expectation is taken assuming X has density g.

Theorem 3.2. *Under appropriate smoothness conditions on f, the MLE from an i.i.d. sample is consistent, i.e. $\hat{\theta} \xrightarrow{P} \theta_0$.*

Proof. For an i.i.d. sample the likelihood function is given by $\mathcal{L}(\theta) = \prod_{i=1}^{n} f(X_i|\theta)$ and maximizing the likelihood is equivalent to maximizing $l(\theta) = \log \mathcal{L}(\theta)$. This is the same as maximizing

$$\frac{1}{n}l(\theta) = \frac{1}{n}\sum_{i=1}^{n}\log f(X_i|\theta).$$

By the law of large numbers, we have

$$\frac{1}{n}l(\theta) \xrightarrow{P} \mathbb{E}(\log f(x|\theta)) = \int \log[f(x|\theta)]f(x|\theta_0)\,dx,$$

where θ_0 denotes the true value of θ. Assuming that we can interchange integration and differentiation:

$$\frac{\partial}{\partial\theta}\int \log[f(x|\theta)]f(x|\theta_0)\,dx = \int \frac{\partial}{\partial\theta}\Big(\log[f(x|\theta)]f(x|\theta_0)\Big)\,dx = \int \frac{\frac{\partial}{\partial\theta}f(x|\theta)}{f(x|\theta)}f(x|\theta_0)\,dx.$$

If now $\theta = \theta_0$, f cancels and

$$\int \frac{\partial}{\partial\theta}f(x|\theta_0)dx = \frac{\partial}{\partial\theta}\int f(x|\theta_0)dx = 0,$$

so θ_0 is at least a stationary point. We need to show that this is a maximum, i.e. that for any fixed $\epsilon > 0$,

$$\mathcal{L}(\theta_0) > \mathcal{L}(\theta_0 - \epsilon)$$
$$\mathcal{L}(\theta_0) > \mathcal{L}(\theta_0 + \epsilon),$$

with probability tending to one as $n \to \infty$. By the law of large number and the Information inequality

$$\frac{1}{n}\log\frac{\mathcal{L}(\theta_0)}{\mathcal{L}(\theta_0-\epsilon)} = \frac{1}{n}\log\frac{\prod_i f(x_i|\theta_0)}{\prod_i f(x_i|\theta_0-\epsilon)} = \frac{1}{n}\sum_i \log\frac{f(x_i|\theta_0)}{f(x_i|\theta_0-\epsilon)}$$
$$\xrightarrow{P} \mathbb{E}\left(\log\frac{f(X|\theta_0)}{f(X|\theta_0-\epsilon)}\right) > 0$$

Multiplying both sides by n and taking the exponential proves the first inequality. The second is proved the same way. □

3.1.2 Asymptotic normality

When doing likelihood estimation in practice, closed form expressions for the standard error is often not available and we need to approximate it. This can be done by the method of bootstrapping (but is often time consuming) or by using the inverse of the Fisher Information matrix, defined below. We will now sketch the proof of why we are justified using this approximation.
We need the following lemma (Rice (1988)):

Lemma 3.3. *Define the Fisher Information, $I(\theta)$, by*

$$I(\theta) = \mathbb{E}\left[\frac{\partial}{\partial\theta}\log f(X|\theta)\right]^2.$$

Under appropriate smoothness conditions on f, $I(\theta)$ may also be expressed as

$$I(\theta) = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta^2}\log f(X|\theta)\right].$$

Theorem 3.4. *Let $X_1, \ldots, X_n, \ldots$ be a sequence of i.i.d. observations from $f(x|\theta)$, let $\hat{\theta}_n$ be the MLE of the univariate θ based on $X_1, \ldots, X_n$, and let θ_0 be the true value of θ. Under appropriate smoothness conditions on f, the probability distribution of $\sqrt{n}(\hat{\theta}_n - \theta_0)$ tends to $\mathcal{N}(0, I^{-1}(\theta_0))$.*

Proof. By a first order Taylor expansion where the remainder term is set to zero,

$$\begin{aligned}
0 = l'(\hat{\theta}_n) &= l'(\theta_0) + (\hat{\theta}_n - \theta_0)l''(\theta_0) \\
&\Rightarrow (\hat{\theta}_n - \theta_0) = -\frac{l'(\theta_0)}{l''(\theta_0)} \\
&\Leftrightarrow n^{1/2}(\hat{\theta}_n - \theta_0) = -\frac{n^{-1/2}l'(\theta_0)}{n^{-1}l''(\theta_0)}
\end{aligned}$$

We first study the expectation and variance of the numerator:

$$\begin{aligned}\mathbb{E}(l'(\theta_0)) &= \mathbb{E}\left[\sum_i \frac{\partial}{\partial\theta_0}\log f(X_i|\theta_0)\right] \\ &= \sum_i \mathbb{E}\left[\frac{\partial}{\partial\theta_0}\log f(X_i|\theta_0)\right] \\ &= \sum_i \int \frac{\partial}{\partial\theta_0}\log[f(x_i|\theta_0)]f(x_i|\theta_0)\,dx \\ &= \sum_i \int \frac{\partial}{\partial\theta_0} f(x_i|\theta_0)dx = \sum_i \frac{\partial}{\partial\theta_0}\int f(x_i|\theta_0)dx = 0,\end{aligned}$$

since the density integrates to one. By lemma 3.3 and independence, the variance is given by

$$\begin{aligned}\mathrm{Var}(n^{-1/2}l'(\theta_0)) &= \mathrm{Var}\left[n^{-1/2}\sum_i \frac{\partial}{\partial\theta_0}\log f(X_i|\theta_0)\right] \\ &= n^{-1}\sum_i \mathbb{E}\left[\frac{\partial}{\partial\theta_0}\log f(X_i|\theta_0)\right]^2 = I(\theta_0).\end{aligned}$$

For the denominator we have by the law of large numbers and lemma 3.3

$$-\frac{1}{n}l''(\theta_0) = -\frac{1}{n}\sum_i \frac{\partial^2}{\partial\theta_0^2}\log f(x_i|\theta_0) \xrightarrow{P} -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_0^2}\log f(X_i|\theta_0)\right] = I(\theta_0).$$

Thus,

$$-\frac{n^{-1/2}l'(\theta_0)}{n^{-1}l''(\theta_0)} \approx \frac{n^{-1/2}l'(\theta_0)}{I(\theta_0)},$$

with expectation zero and variance given by

$$\mathrm{Var}(n^{-1/2}(\hat{\theta}_n - \theta_0)) = \frac{I(\theta_0)}{I^2(\theta_0)} = \frac{1}{I(\theta_0)}.$$

Since $l'(\theta_0) = \sum_i \frac{\partial}{\partial\theta_0}\log f(X_i|\theta_0)$ is a sum of i.i.d. random variables, by the central limit theorem

$$n^{-1/2}(\hat{\theta}_n - \theta_0) \xrightarrow{d} \mathcal{N}(0, I^{-1}(\theta_0)).$$

□

We can observe that small Fisher information results in large asymptotic variance, and large Fisher information results in a smaller asymptotic variance of the MLE. The geometric interpretation is that if the curvature is big, $I(\theta)$ will be large, and the variance small. Thus, the MLE is much more likely to be true than other possible values nearby. Small Fisher information means that the curvature is small and could mean that there are other values that are almost equally likely.

Theorem 3.4 generalizes to $\boldsymbol{\theta}$ being multidimensional. The Fisher Information $I(\theta)$ is then a matrix and under certain regularity conditions the entries are given by

$$[\mathcal{I}(\boldsymbol{\theta})]_{i,j} = \mathbb{E}\left[\frac{\partial}{\partial\theta_i}\log f(X|\boldsymbol{\theta})\frac{\partial}{\partial\theta_j}\log f(X|\boldsymbol{\theta})\right] = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log f(X|\boldsymbol{\theta})\right].$$

The covariance matrix is then given by the inverse Fisher Information. In practice, the *observed* Fisher Information, the negative Hessian of the log-likelihood, is used.

3.2 Likelihood estimation of models with latent variables

All models considered in this thesis contain latent variables. As a consequence, the likelihood will be an integral over the latent variables. This integral is often high dimensional and can't be evaluated analytically. One solution is to apply quadrature rules for numerical integration, but this does not scale well to high dimensions (Fournier et al. (2011)). An efficient alternative to numerical integration is the Laplace approximation, which we derive below. Since many optimization algorithms minimizes the function we will working with the negative log-likelihood.

Let $\boldsymbol{y}$ be a vector of observations, $\boldsymbol{\theta}$ our parameters of interest and $\boldsymbol{u}$ be a random vector of latent variables. The conditional density of our observations given $\boldsymbol{u}$ is denoted by $f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u})$, and $f_{\boldsymbol{u}}(\boldsymbol{u})$ denotes the marginal density of $\boldsymbol{u}$. Let $g(\boldsymbol{u},\boldsymbol{\theta})$ denote the negative joint log-likelihood. The likelihood of $\boldsymbol{\theta}$ is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \int f(\boldsymbol{y}, \boldsymbol{u})\, d\boldsymbol{u} = \int f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u}) f_{\boldsymbol{u}}(\boldsymbol{u})\, d\boldsymbol{u} = \int \exp\{-g(\boldsymbol{u}, \boldsymbol{\theta})\}\, d\boldsymbol{u}. \tag{3.3}$$

We assume that g has a global minimum at $\hat{\boldsymbol{u}}$ for a given $\boldsymbol{\theta}$, i.e. $\hat{\boldsymbol{u}} = \arg\min_{\boldsymbol{u}} g(\boldsymbol{u}, \boldsymbol{\theta})$, and that g is twice differentiable. The solution $\hat{\boldsymbol{u}}$ is known as the *Empirical Bayes* (EB) estimate. A second order Taylor expansion around $\hat{\boldsymbol{u}}$ yields

$$g(\boldsymbol{u}, \boldsymbol{\theta}) \approx g(\hat{\boldsymbol{u}}, \boldsymbol{\theta}) + \nabla g(\hat{\boldsymbol{u}}, \boldsymbol{\theta})(\boldsymbol{u} - \hat{\boldsymbol{u}}) + \frac{1}{2}(\boldsymbol{u} - \hat{\boldsymbol{u}})^T \mathbb{H} (\boldsymbol{u} - \hat{\boldsymbol{u}}) \tag{3.4}$$

Since $\hat{\boldsymbol{u}}$ is a minimum, $\nabla g(\hat{\boldsymbol{u}}, \boldsymbol{\theta}) = 0$. Therefore

$$\mathcal{L}(\boldsymbol{\theta}) \approx \exp\{-g(\hat{\boldsymbol{u}}, \boldsymbol{\theta})\} \int \exp\left\{ -\frac{1}{2}(\boldsymbol{u} - \hat{\boldsymbol{u}})^T \mathbb{H} (\boldsymbol{u} - \hat{\boldsymbol{u}}) \right\} d\boldsymbol{u} \tag{3.5}$$

We can observe that the integrand is the kernel of a multivariate normal density with covariance matrix $\mathbb{H}^{-1}$. The approximation is therefore given by

$$\mathcal{L}(\boldsymbol{\theta}) \approx \exp\{-g(\hat{\boldsymbol{u}}, \boldsymbol{\theta})\} (2\pi)^{\dim(\boldsymbol{u})/2} \det(\mathbb{H})^{-1/2}, \tag{3.6}$$

where we have used the fact that $\det(\mathbb{H}^{-1}) = \det(\mathbb{H})^{-1}$. The corresponding negative log-likelihood is

$$-l(\boldsymbol{\theta}) = -\frac{\dim(\boldsymbol{u})}{2} \log(2\pi) + \frac{1}{2} \log\det(\mathbb{H}) + g(\hat{\boldsymbol{u}}, \boldsymbol{\theta}) \tag{3.7}$$

The Laplace approximation is exact when the joint density is Gaussian (Fournier et al. (2011)). It also works well when $f_{\boldsymbol{u}}(\boldsymbol{u}|\boldsymbol{\theta})$ is Gaussian and $f_{\boldsymbol{u}}(\boldsymbol{u}|\boldsymbol{\theta})$ is more informative than $f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u})$ with respect to the observed Information matrix. As an example, the Laplace approximation of a t-distribution with five degrees of freedom can be seen in figure 3.1.

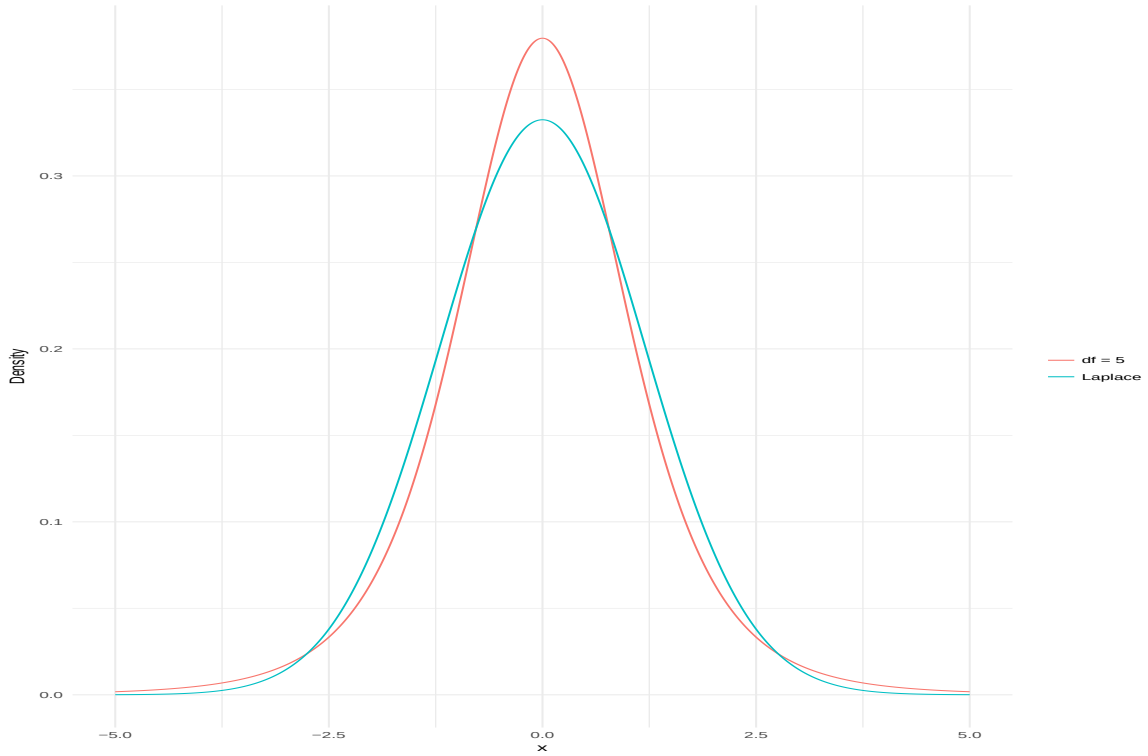


Fig. 3.1 The Laplace approximation of a t-distribution with five degrees of freedom.

Chapter 4

A short introduction to MCMC

4.1 Why MCMC?

When modelling in the Bayesian framework, the parameters $\boldsymbol{\theta}$ are not considered as being fixed to a certain value, but are treated as stochastic variables, with corresponding densities. To be able to do inference about $\boldsymbol{\theta}$ we need to define a model that provides a joint distribution of the parameters $\boldsymbol{\theta}$ and the data $\boldsymbol{y}$, denoted by $f(\boldsymbol{\theta}, \boldsymbol{y})$. The joint distribution can be written as the product $f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$, where $f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta})$ is the likelihood function defined in chapter 3, and $\pi(\boldsymbol{\theta})$ is the *prior distribution* of our parameters, i.e. our belief about $\boldsymbol{\theta}$ before seeing any data. We are interested in doing inference based on our data, and this gives rise to the *posterior distribution* i.e. the distribution of $\boldsymbol{\theta}$ given the data. The posterior distribution represents how we adjust our prior beliefs in response to observed data. By Bayes theorem we have

$$f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{f_{\boldsymbol{y}}(\boldsymbol{y})} = \frac{f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}, \tag{4.1}$$

where the denominator is a normalization constant. This constant is often unknown and hard to obtain in practice, but as we will see, not needed when doing MCMC.

Discarding the normalization constant yields the *unnormalized posterior density*:

$$f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y}) \propto f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}). \tag{4.2}$$

In Bayesian statistics, latent variables are treated as parameters. If $\boldsymbol{u} \sim f_{\boldsymbol{u}}(\boldsymbol{u}|\boldsymbol{\theta})$ denotes our latent variables, and our observations are denoted as $\boldsymbol{y} \sim f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u}, \boldsymbol{\theta})$, the joint posterior of $(\boldsymbol{\theta}, \boldsymbol{u})$ is given by

$$f_{\boldsymbol{\theta},\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{u}|\boldsymbol{y}) = \frac{f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{u}) f_{\boldsymbol{u}}(\boldsymbol{u}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int\int f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{u}) f_{\boldsymbol{u}}(\boldsymbol{u}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) d\boldsymbol{\theta} d\boldsymbol{u}} \propto f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{u}) f_{\boldsymbol{u}}(\boldsymbol{u}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}), \tag{4.3}$$

where, again, the denominator is a normalization constant. For the remaining part of the thesis, latent variables $\boldsymbol{u}$ will be not be stated explicitly, but will be included in the parameter vector $\boldsymbol{\theta}$.

In many complex models the posterior is not available in a closed form and we can't sample directly from $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y})$. One way to approximate $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y})$, is to use *Markov chain Monte Carlo* algorithms (MCMC).

The idea behind MCMC is to construct a Markov chain on the state space $\mathcal{X}$ whose stationary distribution is the target density of interest. A Markov chain is a sequence of random variables $\{\theta_i\}_{i=1}^{\infty}$, for which the conditional distribution of θ_i given $(\theta_1, \ldots, \theta_{i-1})$, for any i, only depends on the most recent value θ_{i-1}, i.e. $\mathbb{P}(\theta_t|\theta_{t-1}, \ldots, \theta_1) = \mathbb{P}(\theta_t|\theta_{t-1})$. For the remainder of this thesis, the stationary distribution is assumed to be the posterior, $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y})$. Given that the Markov chain has reached the stationary distribution, the sample $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ generated by the Markov chain can be used to estimate expectations (Betancourt (2017)):

$$\hat{\boldsymbol{\theta}}_n = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\theta}_i. \tag{4.4}$$

Since, by the law of large numbers,

$$\lim_{n\to\infty} \hat{\boldsymbol{\theta}}_n = \mathbb{E}(\boldsymbol{\theta}).$$

There also exist a Central Limit Theorem for MCMC, which under certain regularity conditions states:

$$\hat{\boldsymbol{\theta}}_n^{\text{MCMC}} \xrightarrow{d} \mathcal{N}(\mathbb{E}(\boldsymbol{\theta}), \text{MCMC-SE}),$$

where *MCMC-SE* denotes the standard error and is given by

$$\text{MCMC-SE} = \sqrt{\frac{\text{Var}(\boldsymbol{\theta})}{\text{ESS}}}.$$

ESS is the *effective sample size*:

$$\text{ESS} = \frac{n}{1 + 2\sum_{i=1}^{\infty} \rho_i}$$

where ρ_i is the ith lag autocorrelation function in the Markov chain. We can observe that the standard error grows as the autocorrelation grows. Thus, if our Markov chain is strongly correlated, the standard error of our estimates will be larger than in the case when the sample is independent. We therefore need a bigger sample than in the i.i.d. case.

In the next sections we introduce two basic MCMC algorithms used in this thesis, namely the *Metropolis-Hastings algorithm* and the *Gibbs sampler*. Our main sources will be Gelman et al. (2015) and Murphy (2012).

4.2 The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm (MH) is a family of Markov chain simulation methods. The purpose of MH is to draw samples from a target probability distribution f by generating a Markov Chain whose stationary distribution is f. We will assume that f is the posterior distribution $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y})$. At each time point t, given the state the state of the chain $\boldsymbol{\theta}_{t-1}$, we propose a new state $\boldsymbol{\theta}^{\star}$ from a proposal distribution $J_t(\boldsymbol{\theta}^{\star}|\boldsymbol{\theta}_{t-1})$. Typical examples of proposal distribution are the normal and uniform distribution. We must now decide if we want to move from $\boldsymbol{\theta}_{t-1}$ to $\boldsymbol{\theta}^{\star}$. This is done by

calculating the acceptance probability

$$\alpha = \min\left(1, \frac{f_{\boldsymbol{\theta}}(\boldsymbol{\theta}^\star|\boldsymbol{y})/J_t(\boldsymbol{\theta}^\star|\boldsymbol{\theta}_{t-1})}{f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{t-1}|\boldsymbol{y})/J_t(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}^\star)}\right), \tag{4.5}$$

and set

$$\boldsymbol{\theta}_t = \begin{cases} \boldsymbol{\theta}^\star & \text{with probability } \alpha \\ \boldsymbol{\theta}_{t-1} & \text{otherwise.} \end{cases}$$

Note that if J_t is symmetric, meaning $J_t(\boldsymbol{\theta}^\star|\boldsymbol{\theta}_{t-1}) = J_t(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}^\star)$, the acceptance probability reduces to the ratio of the target density in the proposed state and the current state. This implies moving to $\boldsymbol{\theta}^\star$ if $\boldsymbol{\theta}^\star$ is more probable than $\boldsymbol{\theta}_{t-1}$, which makes sense, we want to sample from areas of high probability mass. We can still move to $\boldsymbol{\theta}^\star$, even if $\boldsymbol{\theta}^\star$ is less probable that $\boldsymbol{\theta}_{t-1}$, which makes exploration of the entire space of $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y})$ possible.

We can also observe that we don't need the normalized posterior density for the MH algorithm. Let $\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y})$ be the normalized posterior, i.e. $\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y}) = f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y})/Z$, where Z is the normalization constant. Then

$$\frac{\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{\theta}^\star|\boldsymbol{y})}{\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{t-1}|\boldsymbol{y})} = \frac{f_{\boldsymbol{\theta}}(\boldsymbol{\theta}^\star|\boldsymbol{y})/Z}{f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{t-1}|\boldsymbol{y})/Z} = \frac{f_{\boldsymbol{\theta}}(\boldsymbol{\theta}^\star|\boldsymbol{y})}{f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{t-1}|\boldsymbol{y})}$$

We are therefore justified using the unnormalized posterior. The MH algorithm is summarized in Algorithm 1.

We will revisit MH in chapter 5 where Hamiltonian dynamics is used to propose new states in a Markov chain.

4.3 The Gibbs sampler

As we saw in the previous section, the MH algorithm updates all parameters simultaneously. An alternative to this is to sample subvectors of $\boldsymbol{\theta}$ conditioning on all other parameters. This is the strategy used in the Gibbs sampler.

Algorithm 1: Metropolis-Hastings algorithm

Input : $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{y}), J_t, T$
Initialize: $\boldsymbol{\theta}_0$
for $t = 1, 2, \ldots, T$ **do**
  Sample $\boldsymbol{\theta}^\star \sim J_t(\boldsymbol{\theta}^\star|\boldsymbol{\theta}_{t-1})$.
  Compute acceptance probability
  $$\alpha = \min\left(1, \frac{f_{\boldsymbol{\theta}}(\boldsymbol{\theta}^\star|\boldsymbol{y})/J_t(\boldsymbol{\theta}^\star|\boldsymbol{\theta}_{t-1})}{f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{t-1}|\boldsymbol{y})/J_t(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}^\star)}\right)$$
  Sample $u \sim U(0,1)$
  Set new sample to
  $$\boldsymbol{\theta}_t = \begin{cases} \boldsymbol{\theta}^\star & \text{if } u < \alpha \\ \boldsymbol{\theta}_{t-1} & \text{if } u \geq \alpha. \end{cases}$$
end
return $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T)$

Suppose $\boldsymbol{\theta}$ can be divided into d subvectors, $\boldsymbol{\theta} = (\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^d)$, and let $f_{\boldsymbol{\theta}^j}(\boldsymbol{\theta}^j|\boldsymbol{\theta}^{-j}, \boldsymbol{y})$ denote the conditional posterior distribution of $\boldsymbol{\theta}^j$ given all the other components of $\boldsymbol{\theta}$ and the data $\boldsymbol{y}$. At each iteration t in the Markov chain, for $j = 1, \ldots, d$, we sample $\boldsymbol{\theta}_t^j$ from the conditional distribution $f_{\boldsymbol{\theta}^j}(\boldsymbol{\theta}_t^j|\boldsymbol{\theta}_{t-1}^{-j}, \boldsymbol{y})$, where

$$\boldsymbol{\theta}_{t-1}^{-j} = (\boldsymbol{\theta}_t^1, \ldots, \boldsymbol{\theta}_t^{j-1}, \boldsymbol{\theta}_{t-1}^{j+1}, \ldots, \boldsymbol{\theta}_{t-1}^d)$$

represents the current values of all other subvectors. Note that up to component j, we condition on the value from the current iteration, since these parameters have already been sampled, while for components starting from $j+1$ we condition on the values from the last iteration. The Gibbs sampler is summarized in following algorithm:

Algorithm 2: Gibbs sampler

Input : $f_{\boldsymbol{\theta}^1}(\boldsymbol{\theta}^1|\boldsymbol{\theta}^{-1}, \boldsymbol{y}), \ldots, f_{\boldsymbol{\theta}^d}(\boldsymbol{\theta}^d|\boldsymbol{\theta}^{-d}, \boldsymbol{y}), T$
Initialize: $\boldsymbol{\theta}_0^1, \ldots, \boldsymbol{\theta}_0^d$
for $t = 1, 2, \ldots, T$ **do**
  for $j = 1, 2, \ldots, d$ **do**
    Sample $\boldsymbol{\theta}_t^j \sim f_{\boldsymbol{\theta}^j}(\boldsymbol{\theta}_t^j|\boldsymbol{\theta}_{t-1}^{-j}, \boldsymbol{y})$
  end
end
return $(\boldsymbol{\theta}_1^1, \ldots, \boldsymbol{\theta}_T^1), \ldots, (\boldsymbol{\theta}_1^d, \ldots, \boldsymbol{\theta}_T^d)$

The Gibbs sampler is extensively used in the sampling methodology proposed by Kastner et al. (2017) discussed in chapter 7.

In the next chapter we will discuss how Hamiltonian dynamics can be used together with the MH algorithm to generate samples from the posterior distribution.

Chapter 5

Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) was proposed in the field of statistical physics by Duane et al. (1987), and was originally developed to tackle the calculations in Lattice Quantum Chromodynamics. It combined the Metropolis-Hastings algorithm with techniques from molecular dynamics. HMC can be used to produce efficient proposal distributions for a Metropolis-Hastings sampler that allows large moves in the parameter space while keeping a high acceptance rate. It was introduced to the statistical literature by Radford Neal (Neal (1996)), where he applied HMC in his work to Bayesian neural networks. In this chapter we introduce some of the theory of HMC and how it can be applied in parameter estimation. Our main sources for this section are Neal (2010) and Betancourt (2017).

5.1 Hamilton's equations

We will first introduce the equations that govern how the Hamiltonian system evolves over time and some of its properties. We will then discuss how this can be used to sample from the posterior distribution of our model.

Hamiltonian dynamics is dependent on two elements, a position vector $\boldsymbol{\theta} \in \mathbb{R}^d$, and a momentum vector $\boldsymbol{p} \in \mathbb{R}^d$, making the full system $D = 2d$ dimensional. The system is described by a function of $\boldsymbol{\theta}$ and $\boldsymbol{p}$, known as the *Hamiltonian*, $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p})$. How $\boldsymbol{\theta}$ and

$\boldsymbol{p}$ changes over time is governed by *Hamilton's equations*:

$$\frac{d\theta_i}{dt} = \frac{\partial \mathcal{H}}{\partial p_i}, \quad i = 1, \ldots, d, \tag{5.1}$$

$$\frac{dp_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \theta_i}, \quad i = 1, \ldots, d, \tag{5.2}$$

The system can also be written more compactly in vector notation as

$$\frac{d}{dt}\begin{pmatrix}\boldsymbol{\theta} \\ \boldsymbol{p}\end{pmatrix} = J\nabla\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p})' \tag{5.3}$$

where

$$\nabla\mathcal{H} = \left(\frac{\partial \mathcal{H}}{\partial \theta_1}, \ldots, \frac{\partial \mathcal{H}}{\partial \theta_d}, \frac{\partial \mathcal{H}}{\partial p_1}, \ldots, \frac{\partial \mathcal{H}}{\partial p_d}\right),$$

and

$$J = \begin{pmatrix} 0_{d\times d} & I_{d\times d} \\ -I_{d\times d} & 0_{d\times d} \end{pmatrix}$$

is a $2d \times 2d$ matrix.

For HMC, we use Hamiltonian functions that can be written on the form

$$\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p}) = \mathcal{U}(\boldsymbol{\theta}) + \mathcal{K}(\boldsymbol{p}), \tag{5.4}$$

where $\mathcal{U}(\boldsymbol{\theta})$ is called the *potential energy* and $\mathcal{K}(\boldsymbol{p})$ is the *Kinetic energy*. As we will see later $\mathcal{U}(\boldsymbol{\theta})$ will be defined as the negative log posterior of $\boldsymbol{\theta}$. We define $\mathcal{K}(\boldsymbol{p})$ as

$$\mathcal{K}(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}'\boldsymbol{M}^{-1}\boldsymbol{p}. \tag{5.5}$$

The matrix $\boldsymbol{M}$, called the "mass-matrix", is symmetric, positive-definite and is typically diagonal, often a scalar multiple of the identity matrix. We can identify $\mathcal{K}$ as the negative kernel of a multivariate, zero-mean Gaussian distribution with covariance matrix $\boldsymbol{M}$.

Assuming the form given by Equation 5.4, the derivative of $\mathcal{H}$ w.r.t $\boldsymbol{p}$ simplifies to

$\boldsymbol{M}^{-1}\boldsymbol{p}$, and as a consequence, the Hamiltonian equations can be written as

$$\frac{d\theta_i}{dt} = [\boldsymbol{M}^{-1}\boldsymbol{p}]_i \tag{5.6}$$

$$\frac{dp_i}{dt} = -\frac{\partial \mathcal{U}}{\partial \theta_i} \tag{5.7}$$

5.2 Properties of Hamiltonian dynamics

Hamiltonian dynamics have many desirable properties. In this section we cover three of them, namely reversibility, conservation of energy and volume-preservation, and explain why this this is important when constructing Metropolis-Hasting updates.

5.2.1 Reversibility

To talk about reversibility we first need to define what the *flow* of the Hamiltonian means (Bou-Rabee and María Sanz-Serna (2017)).

Definition 5.1. *For fixed t, the t-flow $\varphi_t : \mathbb{R}^D \to \mathbb{R}^D$, is the map that associates with each $\alpha \in \mathbb{R}^D$ the value at time t of the solution of Equation (5.1 and 5.2) that at the initial time 0 takes the initial value α.*

The Hamiltonian is reversible, meaning that the flow $\varphi_{t+s}(\boldsymbol{\theta}, \boldsymbol{p})$, from state $(\boldsymbol{\theta}(t), \boldsymbol{p}(t))$, to state $(\boldsymbol{\theta}(t+s), \boldsymbol{p}(t+s))$ is one-to-one (injective). This implies that there exist an inverse $\varphi^{-1}_{(t+s)}$, so that $\varphi^{-1}_{(t+s)}(\varphi_{t+s}(\boldsymbol{\theta}, \boldsymbol{p}))$ results in the identity mapping, taking $(\boldsymbol{\theta}(t), \boldsymbol{p}(t))$ to $(\boldsymbol{\theta}(t), \boldsymbol{p}(t))$. When the Hamiltonian is additive, as in Equation (5.4), and $\mathcal{K}$ is even, meaning $\mathcal{K}(\boldsymbol{p}) = \mathcal{K}(-\boldsymbol{p})$, the inverse can be obtained by negating $\boldsymbol{p}$, apply $\varphi_{t+s}(\boldsymbol{\theta}, -\boldsymbol{p})$, and then negating $\boldsymbol{p}$ again.

Reversibility of the Hamiltonian is important since it is used to show that the target density is left invariant after MCMC updates. This can be proved by showing the reversibility of the Markov chain transitions, which requires reversibility of the dynamics used to propose new states.

5.2.2 Conservation of the Hamiltonian

That the Hamiltonian is conserved means that the dynamical system is constant along each trajectory of the system. One way to prove this is to show that the derivative is equal to zero. This is straight forward for the Hamiltonian:

$$\begin{aligned}\frac{d\mathcal{H}}{dt} &= \sum_{i=1}^{d}\left[\frac{\partial\mathcal{H}}{\partial p_i}\frac{dp_i}{dt} + \frac{\partial\mathcal{H}}{\partial\theta_i}\frac{d\theta_i}{dt}\right] && \text{(by the chain rule)}\\ &= \sum_{i=1}^{d}\left[-\frac{\partial\mathcal{H}}{\partial p_i}\frac{\partial\mathcal{H}}{\partial\theta_i} + \frac{\partial\mathcal{H}}{\partial\theta_i}\frac{\partial\mathcal{H}}{\partial p_i}\right] && \text{(by Equation (5.1 and 5.2))}\\ &= 0\end{aligned}$$

The theoretical consequence of this, when used to produce new proposals in the Metropolis algorithm, is that the acceptance probability will be one, since $\mathcal{H}$ is conserved, and we are moving along a path of constant probability mass. Unfortunately, we are only able to make $\mathcal{H}$ approximately conserved in most applications, and we are unable to always accept the new proposed state.

5.2.3 Volume preservation

The third property of Hamiltonian dynamics is that it preserves volume in the $(\boldsymbol{\theta}, \boldsymbol{p})$ space. What this means is that if we apply the mapping $\varphi_t(\boldsymbol{\theta}, \boldsymbol{p})$ to the points in some region R in phase space, with volume V, the image, i.e. $\varphi_t(\boldsymbol{\theta}, \boldsymbol{p})\ \forall\ (\boldsymbol{\theta}, \boldsymbol{p}) \in R$, will also have volume V.

It can be shown that a vector field with zero divergence preserve volume (Neal (2010)). Taking the divergence of Equation (5.3):

$$\begin{aligned}\nabla\cdot(J\nabla\mathcal{H}) &= \sum_{i=1}^{d}\left[\frac{\partial}{\partial\theta_i}\frac{d\theta_i}{dt} + \frac{\partial}{\partial p_i}\frac{dp_i}{dt}\right]\\ &= \sum_{i=1}^{d}\left[\frac{\partial}{\partial\theta_i}\frac{\partial\mathcal{H}}{\partial p_i} - \frac{\partial}{\partial p_i}\frac{\partial\mathcal{H}}{\partial\theta_i}\right] = \sum_{i=1}^{d}\left[\frac{\partial^2\mathcal{H}}{\partial\theta_i\partial p_i} - \frac{\partial^2\mathcal{H}}{\partial p_i\partial\theta_i}\right] = 0\end{aligned} \tag{5.8}$$

As Neal notes, the implication of this for MCMC is that we don't need to account for any change in volume in the acceptance probability for Metropolis updates. If this was not the case we would need to calculate the determinant of the Jacobian matrix, something which might be infeasible.

An analytical solution of Hamiltonian dynamics is in general not available, but even when we approximate the dynamics, reversibility and preservation of volume can be maintained.

5.3 The leapfrog method

Due to the fact that in most applications, no analytical solution of Hamilton's equations are available, we need to approximate the solution by discretizing time, using some small step size ϵ, and iteratively approximate the system at time $\epsilon, 2\epsilon, \ldots, L\epsilon$, where L is the number of steps taken. There exists a broad range of methods for approximating systems of differential equations, but we will focus on the so *leapfrog method*. Assuming the Hamiltonian has the form given by Equation (5.4), it works as follows:

$$p_i(t+\epsilon/2) = p_i(t) + \frac{\epsilon}{2}\frac{dp_i}{dt} = p_i(t) - \frac{\epsilon}{2}\frac{\partial \mathcal{U}}{\partial \theta_i}(\theta_i(t)) \tag{5.9}$$

$$\theta_i(t+\epsilon) = \theta_i(t) + \epsilon\frac{d\theta_i}{dt} = \theta_i(t) + \epsilon\frac{\partial \mathcal{K}}{\partial p_i}(p_i(t+\epsilon/2)) \tag{5.10}$$

$$p_i(t+\epsilon) = p_i(t+\epsilon/2) + \frac{\epsilon}{2}\frac{dp_i}{dt} = p_i(t+\epsilon/2) - \frac{\epsilon}{2}\frac{\partial \mathcal{U}}{\partial \theta_i}(\theta_i(t+\epsilon)) \tag{5.11}$$

We start with half a step for the momentum variables, then do a full step in the position variables using the new momentum variables. Since we now know the position of the momentum at $q_i(t+\epsilon)$, we can use this when taking another half step for the momentum variable.

The leapfrog method is a member of a family of numerical methods known as *symplectic integrators* (Betancourt (2017)). These have the property that they generate numerical trajectories that preserve the volume in the phase space, just like the Hamiltonian trajectories they are approximating. Therefore, the numerical trajectories

cannot drift away from the level sets, it instead oscillate near it (even for long integration time).

For the leapfrog method, there are two hyperparameters that need to be selected: the leapfrog step size ϵ, and the number of leapfrog steps L, which together determine the length of the trajectories. Hoffman and Gelman (2011) introduced the No-U-Turn sampler to get around the problem of tuning the hyperparameters manually.

5.4 The No-U-Turn sampler

HMC's performance is highly sensitive to the choice of the step size ϵ and the number of steps L. When L is too small HMC exhibits random walk behavior and will lead to slow mixing. On the other hand, if L is too big, we are using more computational power than we need and may cause *double-back* behavior, where the integrator returns to its starting values (Paquet and Fraccaro (2016)). If the step size ϵ is too small, it will be difficult to explore the whole space, unless a large L is used. A too big ϵ can lead to high rejection rates and an unstable algorithm (Hoffman and Gelman (2011)). The No-U-Turn (NUTS) sampler was introduced to tackle the problem of tuning the step size ϵ and number of steps L. We will not go into the details of the algorithm here, but give a brief description (found in Monnahan et al. (2016)): "A single NUTS trajectory is built by iteratively accumulating steps. In the first iteration, a single leapfrog step is taken from the current state so the trajectory has a total of two steps.Then, two more steps are added(total of four), then four more(total of eight), and so forth, with each iteration doubling the length of the trajectory. This doubling procedure repeats until the trajectory turns back on itself and a 'U-turn' occurs, or the trajectory diverges (i.e. $\mathcal{H}$ goes to infinity). The number of doublings is known as the tree depth.The key aspect of this tree building algorithm is that it automatically creates trajectories that are neither too short nor too long. In practice, this means trajectory lengths vary among transitions: it may take eight steps or 128, depending on the position and momentum vectors."

5.5 Connecting Hamiltonian dynamics to MCMC

To sample $\boldsymbol{\theta}$ from the posterior, we define the joint density of $(\boldsymbol{\theta}, \boldsymbol{p})$ as follows:

$$f(\boldsymbol{\theta}, \boldsymbol{p}) = f_{\boldsymbol{\theta}}(\boldsymbol{\theta}) f_{\boldsymbol{p}}(\boldsymbol{p}) \propto e^{-\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p})} = e^{-(\mathcal{U}(\boldsymbol{\theta}) + \mathcal{K}(\boldsymbol{p}))}, \tag{5.12}$$

where $\mathcal{K}(\boldsymbol{p})$ is defined as in Equation (5.5). We choose our potential energy to be the negative logarithm of the posterior distribution of our parameters of interest:

$$\mathcal{U}(\boldsymbol{\theta}) = -\log\Big[\mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\Big], \tag{5.13}$$

where $\mathcal{L}$ is the likelihood function given the data and π is the prior density.
The HMC algorithm proceeds in two steps. Assume that the current state is $(\boldsymbol{\theta}, \boldsymbol{p})$. The first step only includes the momentum $\boldsymbol{p}$. We sample a new $\boldsymbol{p}$ from the zero-mean multivariate normal distribution with covariance matrix $\boldsymbol{M}$, which can be interpreted as a Gibbs sampling update (Hoffman and Gelman (2011)). Next, using Equation (5.1 and 5.2), we take L steps of length ϵ using the leapfrog method, resulting in a new state $(\boldsymbol{\theta}^\star, \boldsymbol{p}^\star)$.

In the second step a Metropolis update is performed, where we either accept or reject the new state $(\boldsymbol{\theta}^\star, \boldsymbol{p}^\star)$. We accept our new state $(\boldsymbol{\theta}^\star, \boldsymbol{p}^\star)$ of the Markov chain with probability

$$\alpha = \min\left(1, \frac{\exp\Big\{-\mathcal{H}(\boldsymbol{\theta}^\star, \boldsymbol{p}^\star)\Big\}}{\exp\Big\{-\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p})\Big\}}\right) = \min\Big(1, \exp\Big\{\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p}) - \mathcal{H}(\boldsymbol{\theta}^\star, \boldsymbol{p}^\star)\Big\}\Big). \tag{5.14}$$

If we fail to accept the new proposed state, the next state will be the same as the current. Independent of acceptance or rejection, $p^\star$ is discarded after α is calculated. The standard HMC algorithm is summarized in algorithm 3 (inspired by Hoffman and Gelman (2011)).

Algorithm 3: Hamiltonian Monte Carlo

Input : $\boldsymbol{\theta}_0, \epsilon, L, \mathcal{H}, N$

for $n = 1$ *to* N **do**
  Sample $\boldsymbol{p}_0 \sim \mathcal{N}(0, \boldsymbol{M})$.
  Set $\boldsymbol{\theta}_n \leftarrow \boldsymbol{\theta}_{n-1}, \boldsymbol{\theta}^\star \leftarrow \boldsymbol{\theta}_{n-1}, \boldsymbol{p}^\star \leftarrow \boldsymbol{p}_0$.
  for $i = 1$ *to* L **do**
    Set $\boldsymbol{\theta}^\star, \boldsymbol{p}^\star \leftarrow \text{Leapfrog}(\boldsymbol{\theta}^\star, \boldsymbol{p}^\star, \epsilon)$.
  end
  With probability $\alpha = \min\left(1, \exp\left\{\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p}) - \mathcal{H}(\boldsymbol{\theta}^\star, \boldsymbol{p}^\star)\right\}\right)$, set $\boldsymbol{\theta}_n \leftarrow \boldsymbol{\theta}^\star$.
end
return $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N)$

5.6 A one-dimensional example

Consider the case when $\theta, p \in \mathbb{R}$, where the potential and kinetic energy is defined as

$$\mathcal{U}(\theta) = \frac{\theta^2}{2}, \quad \mathcal{K}(p) = \frac{p^2}{2}, \tag{5.15}$$

meaning that both the posterior and the kinetic energy are standard normal variables. The Hamiltonian is defined by

$$\mathcal{H}(\theta, p) = \mathcal{U}(\theta) + \mathcal{K}(p) = \frac{\theta^2}{2} + \frac{p^2}{2}, \tag{5.16}$$

and the dynamics resulting from this is

$$\frac{d\theta}{dt} = p, \quad \frac{dp}{dt} = -\theta. \tag{5.17}$$

This two-dimensional system of differential equations has an analytical solution. For some constants r and a, the solution is

$$\theta(t) = r\cos(a + t), \quad p(t) = -r\sin(a + t). \tag{5.18}$$

Given initial conditions $\theta(0) = \theta_0$ and $p(0) = p_0$, we can solve Equation (5.18) for r and a, and see that

$$a = \tan^{-1}\left(-\frac{p_0}{q_0}\right), \quad r = \sqrt{\theta_0^2 + p_0^2},$$

resulting in

$$\theta(t) = \sqrt{\theta_0^2 + p_0^2}\cos\left(\tan^{-1}\left(-\frac{p_0}{\theta_0}\right) + t\right), \tag{5.19}$$

$$p(t) = -\sqrt{\theta_0^2 + p_0^2}\sin\left(\tan^{-1}\left(-\frac{p_0}{\theta_0}\right) + t\right). \tag{5.20}$$

In this simple example, the flow $\varphi_t(\theta_0, p_0)$ in the phase space is simply a clockwise rotation by t radians (see plot (5.1)). This system is clearly reversible, since φ_{-t} is just a counter-clockwise rotation by t radians, undoing the rotation φ_t. Substituting this into Equation (5.16), and after some calculations, we get

$$\mathcal{H}(q, p) = \theta_0^2 + p_0^2, \tag{5.21}$$

which is constant and represents the squared distance from the origin of the initial conditions. This can be seen in figure 5.1, where the initial state was $\theta_0 = 0, p_0 = 1$, implying $\mathcal{H}(\theta, p) = 1$.

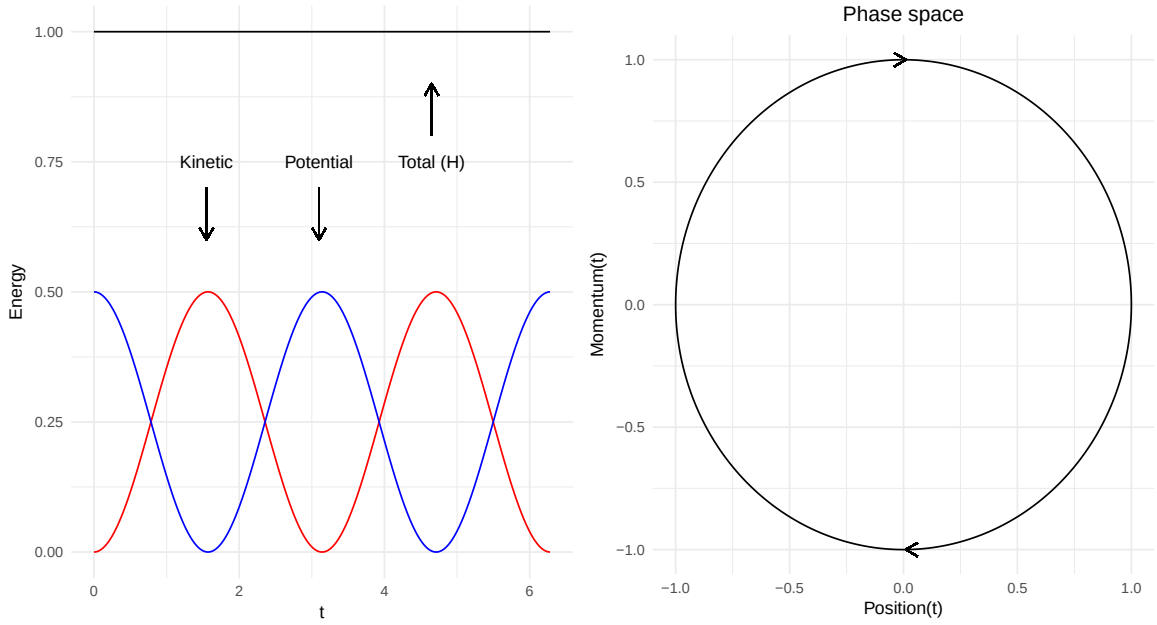


Fig. 5.1 Left: Plot of the potential energy $\mathcal{U}$, kinetic energy $\mathcal{K}$ and the Hamiltonian $\mathcal{H}$ in the 1D example, when the Hamiltonian is given by $\mathcal{H}(q,p) = q^2/2 + p^2/2$. Right: The phase space of the Hamiltonian. The initial state was $q = 0, p = 1$.

Chapter 6

Automatic Differentiation, TMB and Stan

In this chapter we will give a short introduction to automatic differentiation (AD), also known as algorithmic differentiation. We will then discuss the R package **TMB** and the probabilistic programming language **Stan**, where AD is used to calculate derivatives of the likelihood function.

6.1 Automatic Differentiation

In many problems in mathematics and statistics we are not only interested in the function value, but also the gradient ∇f and the hessian $\mathbb{H}$, for example when doing optimization. In fact, when doing maximum likelihood estimation, we are not really interested in the value of the function, as long as we know it is the global maximum.

The derivative of a function can be obtained in several ways, and two classical approaches are numerical and symbolic differentiation. An alternative to these methods is Automatic Differentiation. AD exploits the fact that every function given by a computer algorithm executes a sequence of elementary arithmetic operations (addition, subtraction, multiplication, division, etc.) and elementary functions (exp, sin, cos, log, etc.). Each of these elementary functions can easily be differentiated. The derivative of

Variable	Value	Derivative
v_1	x	$\frac{\partial v_1}{\partial x} = 1$
v_2	v_1^2	$\frac{\partial v_2}{\partial v_1} = 2v_1$
v_3	v_2/ν	$\frac{\partial v_3}{\partial v_2} = \frac{1}{\nu}$
v_4	$1 + v_3$	$\frac{\partial v_4}{\partial v_3} = 1$
v_5	$\log v_4$	$\frac{\partial v_5}{\partial v_4} = \frac{1}{v_4}$
v_6	$-\frac{\nu+1}{2} v_5$	$\frac{\partial v_6}{\partial v_5} = -\frac{\nu+1}{2}$
v_7	$v_6 + \log\left(\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\right)$	$\frac{\partial v_7}{\partial v_6} = 1$

Table 6.1 Example of how each term in Equation (6.1) can be broken down into elementary operations, and gradient calculations for the log density of a t-distribution.

a function can therefore be calculated by evaluating the derivative of these elementary functions and combining them by the chain rule. We will illustrate this with an example:

Example 6.1. We want to approximate the integral of a t-distribution with the Laplace approximation. Define $g(x) = \log f_X(x)$. For this, we need to find $\arg\max_x g(x)$, as this is where we evaluate the function and the second derivative in the approximation. We know that the solution is $x = 0$, but for illustrative purposes we show how AD can be used to evaluate the derivative. The logarithm of the t-distribution is given by

$$g(x) = \log f_X(x) = \log\left(\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\right) - \frac{\nu+1}{2}\log\left(1 + \frac{x^2}{\nu}\right). \tag{6.1}$$

The function can be decomposed into elementary operations as shown in table 6.1. The partial derivatives in the table can then be combined using the chain rule:

$$\begin{aligned} g'(x) &= \frac{\partial v_1}{\partial x}\frac{\partial v_2}{\partial v_1}\frac{\partial v_3}{\partial v_2}\frac{\partial v_4}{\partial v_3}\frac{\partial v_5}{\partial v_4}\frac{\partial v_6}{\partial v_5}\frac{\partial v_7}{\partial v_6}\frac{\partial g}{\partial v_7} \\ &= 1 \times 2v_1 \times \frac{1}{\nu} \times 1 \times \frac{1}{v_4} \times -\frac{\nu+1}{2} \times 1 \times 1 \times 1 \\ &= -\frac{\nu+1}{\nu}\left(\frac{x}{1 + x^2/\nu}\right). \end{aligned}$$

This process can represented as a computational graph, as shown in figure 6.1.

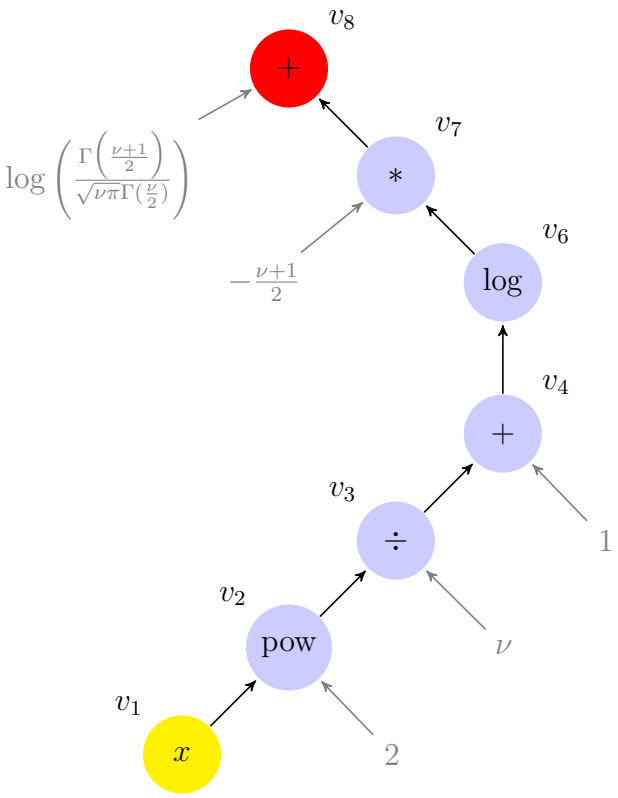


Fig. 6.1 Computational graph for the logarithm of the t-distribution (Equation (6.1)). Each node corresponds to an AD variable, with the variable name outside the node. The independent variable is highlighted in yellow, while the dependent variable is highlighted in red. The function producing each node is displayed inside each node. Constants are shown in gray with gray arrows to indicate that derivatives don't need to propagated to constant operands.

6.1.1 Dual numbers, Reverse/Forward AD mode

The implementation of AD requires *dual numbers*, which are ordered pairs of real numbers on the form $\vec{u} = (u, u')$. The purpose of dual numbers is that we now have pairs of numbers where the first position can hold the value of $f(x_0)$ and the second can hold the value of the derivative $f'(x_0)$. We will assume that $f : \mathbb{R} \to \mathbb{R}$. To be able to do computations with dual numbers, we introduce the following arithmetic rules (Warwick (2010)):

- $\vec{u} + \vec{v} = (u + v, u' + v')$
- $\vec{u} - \vec{v} = (u - v, u' - v')$
- $\vec{u} \times \vec{v} = (uv, u'v + uv')$
- $\frac{\vec{u}}{\vec{v}} = (\frac{u}{v}, \frac{u'v - uv'}{v^2})$, for $v \neq 0$

Note that the arithmetic rules introduced here are just applications of the derivative rules from calculus.

Finally, we must decide how to treat constants and the independent variable x. Again, using basic rules of differentiation, we define

- $\vec{x} = (x, 1)$
- $\vec{c} = (c, 0)$

The next step is to extend the concept from real numbers to real functions. This can be done using the chain rule in the following way. Let $\vec{f}(\vec{u}) = \vec{f}(u, u') = (f(u), u'f'(u))$. We can now add all common functions we know, for example, $\sin(\vec{u}) = \sin(u, u') = (\sin(u), u' \cos(u))$ and $\log(\vec{u}) = \log(u, u') = (\log(u), u'/u)$.

The method described above and in example 6.1 is known as *forward-mode automatic differentiation*, where we start with the independent variables and calculate values in the direction of the arrows.

In general, forward mode calculates directional derivatives

$$\nabla f \cdot \boldsymbol{a} = \sum_{i=1}^{n} a_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}),$$

for some $\boldsymbol{a} \in \mathbb{R}^n$.

In the computational graph, each node k holds the value v_k and a tangent t_k. The tangent represents the directional derivative of v_k with respect to the input variables. Tangents for the independent variables are initialized with values $\boldsymbol{a}$. All tangents can recursively be calculated with the rule

$$t_i = \sum_{j \in \text{children}[i]} \frac{\partial v_i}{\partial v_j} t_j,$$

where the tangents values for the independent variables are initialized with values $\boldsymbol{a}$ (Carpenter et al. (2015)), because that represents the directional derivative of each input variable.

The disadvantage of forward mode is that when calculating the derivative with respect to several independent variables, the cost scales linearly as $O(n)$, where n is the number of independent variables.

There exists another AD method, known as *reverse-mode automatic differentiation*. In reverse mode we start with the dependent variables and propagate through the computational graph in reversed order. Each node k in the graph (see figure 6.1) contains the value v_k, and an adjoint a_k, representing the derivative of an output variable with respect to v_k. The output node's adjoint is initialized to 1, since the derivative of the output variable with respect to itself is 1. In our example the we would set $a_8 = 1$. From the initial values, all adjoint values can be calculated by the formula

$$a_j = \sum_{i \in \text{parent}[j]} \frac{\partial v_i}{\partial v_j} a_i$$

The advantage of reverse mode is that the derivative of a singe output can be calculated with respect to multiple independent variables by doing one pass over the computational graph. This makes reverse mode highly attractive when calculating the Jacobian for functions of many input variables and few output variables, for example the likelihood function (or the posterior in a Bayesian setting). For a detailed example of reverse mode AD, see Carpenter et al. (2015).

In the next sections, we will give an overview of the tools used to implement our models, namely Template Model Builder (**TMB**) for maximum likelihood and **Stan** for Hamiltonian Monte Carlo.

6.2 TMB

We use the R-package **TMB** to implement our models for maximum likelihood estimation, since **TMB** lets us estimate parameters in models with a high number of latent variables. Recall that the joint likelihood function, $f(\boldsymbol{u}, \boldsymbol{\theta})$, is a function of our latent variables $\boldsymbol{u} \in \mathbb{R}^n$ and parameters $\boldsymbol{\theta} \in \mathbb{R}^m$. The user defines the joint likelihood for the data and the latent variables as a C++ template. All other operations are done in R. The Laplace approximation is done by the use of **CHOLMOD**, available through the **Matrix** library in R. To evaluate derivatives **TMB** uses the automatic differentiation library **CppAD** (Bell (2005)). **TMB** calculates up to third order derivatives by the use of AD.

When the program is executed, three computational graphs [1] (see figure 6.1) are created (Kristensen et al. (2016)):

- **T1**: Graph of $f(\boldsymbol{u}, \boldsymbol{\theta})$ (see figure (6.1)).
- **T2**: Graph of $\nabla_{\boldsymbol{u}} f$ generated from T1 by reverse mode AD.
- **T3**: Graph of $\mathbb{H}$ generated from T2 by reverse mode AD.

The computational graphs are only computed once and are then held in memory, ready to be evaluated. Tape T1-T3 are then used to calculate the gradient of the Laplace approximation, see Kristensen et al. (2016) for more details.

Finding the optimal value of $\boldsymbol{\theta}$ can be viewed as a nested optimization problem. To find $\hat{\boldsymbol{u}}(\boldsymbol{\theta})$ and $\mathbb{H}(\boldsymbol{\theta})$ we fix $\boldsymbol{\theta}$ and optimize using a quasi-Newton algorithm or a limited memory Newton method. The Laplace approximation is then optimized w.r.t. $\boldsymbol{\theta}$ using the quasi-Newton algorithm. The process of nested optimization is continued until the convergence criteria described in Fournier et al. (2011) is meet.

We will next give an example to illustrate the process of making a User Template in C++ and optimizing it in R.

[1] In Kristensen et al. (2016) this is also referred to as "tapes".

Example 6.2. Stochastic volatility in **TMB**

Consider the basic SV model introduced in section 2.2.2, where the log-returns of an asset is modeled as

$$\begin{aligned} y_t &= \sigma_y e^{h_t/2}\epsilon_t, \quad t = 1, \dots, T, \\ h_{t+1} &= \phi h_t + \sigma\eta_t, \quad t = 1, \dots, T-1, \end{aligned} \tag{6.2}$$

where $\epsilon_t, \eta_t \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and $h_1 \sim \mathcal{N}(0, \sigma^2/(1-\phi^2))$. Our parameters of interest are $\boldsymbol{\theta} = (\sigma_y, \phi, \sigma)$. The joint density for our observations $\boldsymbol{y}$ and latent variables $\boldsymbol{h}$ is

$$f(\boldsymbol{y}, \boldsymbol{h}) = f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{h}) f_{\boldsymbol{h}}(\boldsymbol{h}) \tag{6.3}$$

$$= \prod_{i=1}^{T} f_{y_i}(y_i|h_i) \times f_{h_1}(h_1) \prod_{i=2}^{T} f_{h_i}(h_i|h_{i-1}), \tag{6.4}$$

The negative joint log-likelihood is given by

$$-l(\boldsymbol{\theta}, \boldsymbol{h}) = -\sum_{i=1}^{T} \log f_{y_i}(y_i|h_i) - \sum_{i=2}^{T} \log f_{h_i}(h_i|h_{i-1}) - \log f_{h_1}(h_1) \tag{6.5}$$

This is the function we want to implement in C++. We will now describe how this can be done. The first lines in a **TMB** is almost always given by

```
// Basic SV-model
#include <TMB.hpp>
template <class Type>
Type objective_function<Type>::operator()(){
```

The **TMB** library is loaded and the objective function is created. The objective function is a templated class where `<Type>` is the data type of both the input values and the return value of the objective function. [2]

Next we import our data and define our parameters:

```
  DATA_VECTOR(y); //Observations
  DATA_INTEGER(n); //number of obs

  //Parameters
```

[2] For more details see https://kaskr.github.io/adcomp/Introduction.html

```
  PARAMETER(log_sigma_y);
  PARAMETER(log_sigma);
  PARAMETER(phi);
  PARAMETER_VECTOR(h); //Latent variables
```

The line `DATA_VECTOR(y)` declares the vector `y` to be same as `dat$y` in R (see below). The lines `PARAMETER(log_sigma_y), PARAMETER(log_sigma), PARAMETER(phi)`, `PARAMETER_VECTOR(h)`, declares the parameters, but note that `h` is the latent variables and will be integrated out. Due to the invariance property of MLEs, we can estimate the logarithm of the standard deviation and then take the exponential transformation, ensuring that the estimate is greater than zero. This is easily done in **TMB**:

```
  // Transform and report parameters
  Type sigma_y = exp(log_sigma_y);
  Type sigma = exp(log_sigma);
  ADREPORT(sigma_y);
  ADREPORT(sigma);
  Type nll = 0 // Negative log likelihood
```

The `ADREPORT`-command will report the estimate and the standard error of the transformed parameters back to R. The last line initializes the negative log likelihood function.

Next, we loop over all our latent variables and observations to get the contributions to the likelihood (see Equation (6.5)), and return the value of the `nll`:

```
  //unobserved process
  nll -= dnorm(h(0),Type(0),sigma/sqrt(1-phi*phi),true);
  for(int i=1;i<n;i++){
    nll -= dnorm(h(i),phi*h(i-1),sigma,true);
  }

  //Observations
  for(int i=0;i<n;i++){
    nll-= dnorm(y(i),Type(0),exp(h(i)/2)*sigma_y,true);
  }
  return nll;
}
```

In R, we first import our data. The data consists of 945 observations of daily returns of pound/dollar exchange rate from 01/10/1981 to 28/06/1985 (as found in (Skaug and Yu (2014))).

```
y <- c(scan(file="sv_basic.dat"))
```

Next, we compile our C++ file and load our model object into R:

```
compile("basic_sv_t.cpp")
dyn.load(dynlib("basic_sv_t"))
```

Defining our data and parameters (with starting values):

```
n <- length(y)
dat <- list(n=n,y=y)
par <- list(log_sigma_y = log(0.2),
        log_sigma = log(0.4),
        phi = 0.9, h =rep(0,n))
```

We now make a object `obj`, containing the data, parameters and the methods that access the objective function and its derivatives (Kristensen et al. (2016)):

```
obj <- MakeADFun(data = dat,
             parameters = par,
             random = "h",
             DLL = "basic_sv_t")
```

Note the `random`-argument. This specifies the parameter(s) we are integrating out. The following lines minimize the objective function (negative log likelihood), calculates the standard error and prints the parameter estimates with corresponding standard error:

```
opt <- nlminb(obj$par,
          obj$fn,
          obj$gr,
          control = list(trace = TRUE))
rep <- sdreport(obj)
srep <- summary(srep)
srep[rownames(srep) != "h",]

              Estimate Std. Error
log_sigma_y -0.4591556 0.10875121
log_sigma   -1.7735580 0.21368841
```

```
    phi          0.9743236 0.01224302
    sigma_y      0.6318169 0.06871085
    sigma        0.1697280 0.03626891
```

The standard errors is calculated using the diagonal of the inverse Hessian w.r.t. the parameters. For transformed parameters, standard errors is obtained by the delta-method (Kristensen et al. (2016)).

6.3 Stan - A probabilistic programming language

In this thesis, Bayesian inference is done by the use of Hamiltonian Monte Carlo. Our models are implemented in the probabilistic programming language **Stan** [3] (Carpenter et al. (2017)). **Stan** is a programming language for specifying a broad range of statistical models. The user defines the posterior distribution of the model, and **Stan** uses Hamiltonian Monte Carlo together with the NUTS algorithm to draw samples from the posterior. The reader is referred to chapter 5 for more details regarding HMC and NUTS.

In contrast to many MCMC methods, HMC needs to calculate the gradient of the posterior density to enable the Hamiltonian system to evolve. As with **TMB**, **Stan** evaluates the gradient by automatic differentiation (Carpenter et al. (2015)). To do this efficiently, **Stan** makes use of the **Stan Math Library**, a C++ reverse-mode automatic differentiation library. We illustrate the programming language through an example.

Example 6.3. Stochastic volatility in **Stan**

We consider the same model as in the **TMB** example. The first step in a **Stan** program is to define our data and parameters. As with **TMB**, these correspond to variables with the same name in R:

```
data {
  int<lower=0> n;   // number of observations
```

[3] Named after the mathematician Stanislaw Ulam, known for his contribution to the development of Monte Carlo methods.

```
  vector[n] y;       // log-returns
}
parameters {
  real<lower=0> sigma_y_std;
  real<lower=-1,upper=1> phi;
  real<lower=0> sigma_std;
  vector[n] h_std;
}
```

We explicitly define the valid parameter space for our parameters, for example, $|\phi| < 1$. **Stan** automatically transforms all constrained parameters to the real line, so that they can be sampled unrestricted, but this is hidden from the user.

The next coding block, `transformed parameters`, allows us to define new parameters as functions of original parameters:

```
transformed parameters {
  vector[n] h;
  real<lower=0> sigma_y;
  real<lower=0> sigma;
  sigma_y = sqrt(sigma_y_std);
  sigma = sqrt(sigma_std);

  h = h_std * sigma;
  h[1] = h[1] / sqrt(1-phi*phi);
  for (t in 2:n){
    h[t] = h[t] + phi*h[t-1];
  }
}
```

The transformed parameters are not sampled. **Stan** samples the parameters declared in `parameters` block, then applies the formulas in the `transformed parameters` block post sampling.

For the stochastic volatility model, mixing is improved if sampling is done in terms of a standardized volatility, then rescaled. We therefore declare the standardized `h_std` in the `parameters` block, and the original value of `h` is then defined in the `transformed parameters` block.

In the `model` block we define the posterior distribution:

```
model {
```

```
  // priors
  sigma_std ~ gamma(0.5,0.5);
  sigma_y_std ~ gamma(0.5,0.5);
  phi ~ beta(20,1.5);

  // likelihood
  h_std ~ normal(0,1);
  y ~ normal(0,exp(h/2)*sigma_y);
}
```

As can be seen from the code, the following priors are used: σ_y^2 and σ is given a gamma prior $\mathcal{G}(0.5, 0.5)$, and $(\phi + 1)/2$ is given a beta prior $\mathcal{B}(20, 1.5)$, which are the same as in Kastner et al. (2017).

A nice feature in **Stan** is the possibility to simulate from the posterior predictive distribution (ppd), the distribution of the outcome variable implied by the model after using the observed data to update the distribution of our parameters. This is done in the `generated quantities` block:

```
    generated quantities {
      vector[n] y_new;
      for(i in 1:n){
      y_new[i] = normal_rng(0,exp(0.5*h[i])*sigma_y);
      }
    }
```

For each draw $s = 1, \ldots, S$ from the posterior distribution, we draw n outcomes $\tilde{y}^s$ from the ppd by simulating from the data model conditioning on the parameters from sample s. The result will be a $S \times N$ matrix of draws from the ppd (Team (2017)). See figure 6.5.

We can now fit our model in R, through the package **rstan** by calling on our **Stan** model (called `SV.stan` in this example):

```
fit <- stan(file = 'SV.stan',
    data = dat,
    chains = 1,
    seed = 123122,
    control = list(adapt_delta = 0.9))
```

`adapt_delta` is the target average proposal acceptance probability during **Stan**'s adaptation period (warm up/burn-in), and increasing it will force **Stan** to take smaller steps (the default is 0.8), making divergence of the Hamiltonian less likely. From experience, this has been necessary when working with stochastic volatility.

By extracting a summary of the **Stan** object, we can print the parameter estimates:

```
names <- c("sigma_y","phi","sigma")
estimates <- summary(fit)$summary
estimates[which(rownames(estimates) %in% names),1:3]

              mean      se_mean          sd
phi      0.9720216 0.0005257844 0.01302553
sigma_y  0.6562835 0.0046112077 0.09994974
sigma    0.1846702 0.0018556930 0.03803346
```

The package **bayesplot** can be used to visually investigate our Markov Chain through traceplots (figure 6.2), plots of the marginal posterior densities (figure 6.3), autocorrelation of the sample (figure 6.4) and if our model fits the data through plotting the posterior predictive density (figure 6.5).

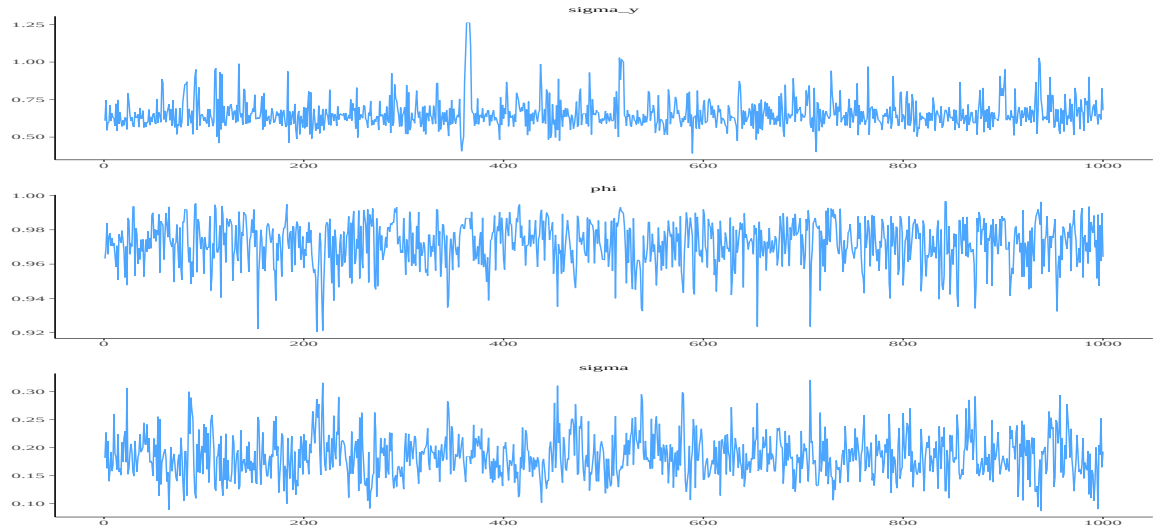


Fig. 6.2 Traceplots of 1000 draws from $p(\sigma_y, \phi, \sigma|\boldsymbol{y})$.

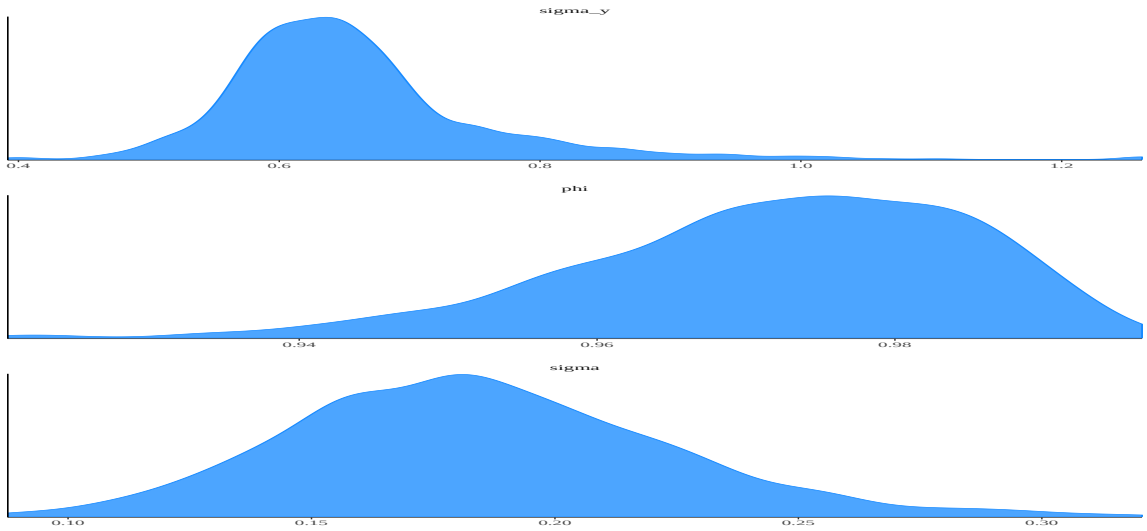


Fig. 6.3 Kernel plots of the marginal posterior densities.

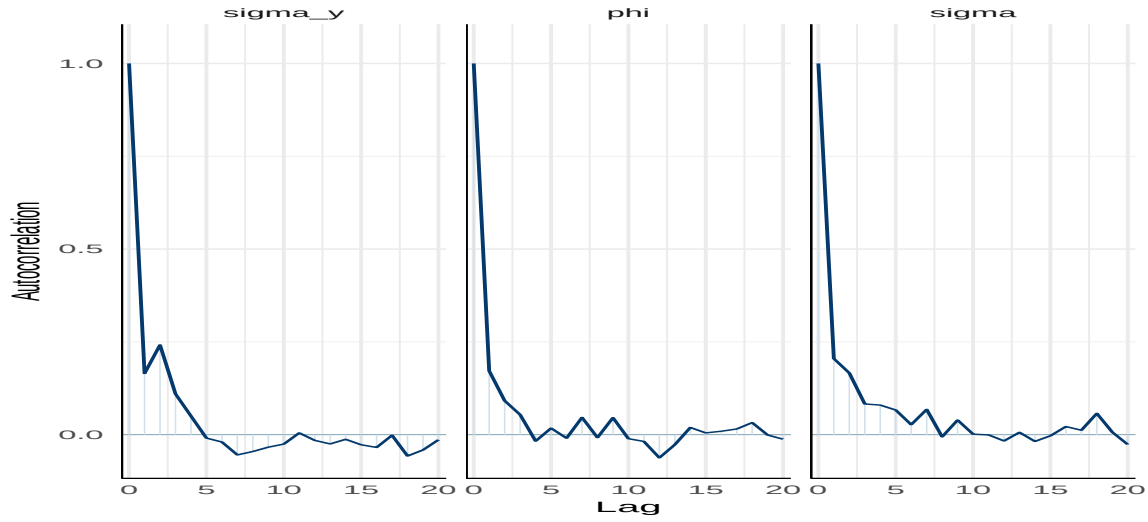


Fig. 6.4 Autocorrelation plot of the draws.

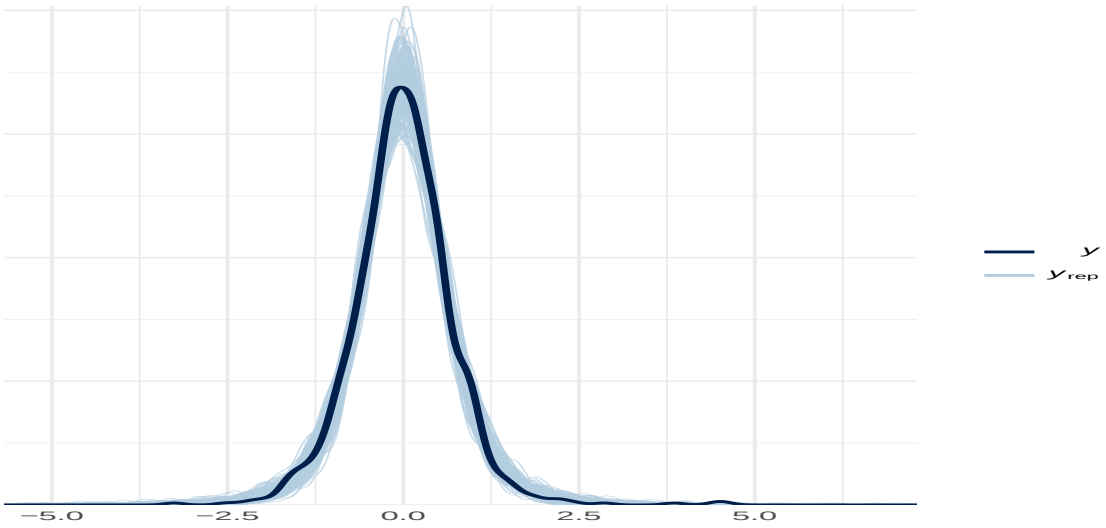


Fig. 6.5 Posterior predictive density. The dark blue line is the distribution of the data y. Each of the 200 lighter lines is the kernel density estimate of one of the replications of $\tilde{y}^s$ from the posterior predictive distribution.

Chapter 7

The Multivariate Factor Stochastic Volatility Model and DIMCMC

7.1 The Multivariate Stochastic Volatility Model

We will consider the multivariate factor stochastic volatility (MFSV) model introduced by Pitt and Shephard (1999). We are going to follow the parameterization used in Kastner et al. (2017). For each point in time $t = 1, \ldots, T$, let $\boldsymbol{y}_t = (y_{1t}, \ldots y_{pt})'$ be a vector of p observed returns, and let $\boldsymbol{f}_t = (f_{1t}, \ldots, f_{qt})'$ be a vector of q unobserved latent factors. The observations are assumed to be driven by the latent factors and the idiosyncratic innovations. In our model, both the idiosyncratic innovations and the latent factors are allowed to have time-varying variances, depending on $p + q$ latent volatilities $\boldsymbol{x}_t = (x_{1t}, \ldots, x_{pt})'$ and $\boldsymbol{h}_t = (h_{1t}, \ldots, h_{qt})'$. Putting this together, our model has the form:

$$\boldsymbol{y}_t = \boldsymbol{\beta}\boldsymbol{f}_t + \boldsymbol{U}_t(\boldsymbol{x}_t)^{1/2}\boldsymbol{\epsilon}_t, \quad \boldsymbol{f}_t = \boldsymbol{V}_t(\boldsymbol{h}_t)^{1/2}\boldsymbol{\zeta}_t, \tag{7.1}$$

where $\boldsymbol{\beta}$ is an unknown $p \times q$ factor loading matrix, $\boldsymbol{U}_t(\boldsymbol{x}_t)$ is a diagonal $p \times p$ matrix containing the idiosyncratic variances, and $\boldsymbol{V}_t(\boldsymbol{h}_t)$ is a diagonal $q \times q$ matrix

containing the factor variances. We also assume that $\boldsymbol{\zeta}_t \sim \mathcal{N}_q(\mathbf{0}, \boldsymbol{I})$, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}_p(\mathbf{0}, \boldsymbol{I})$ are independent.

Both the idiosyncratic and the factor variance are themselves modeled as latent variables whose logarithms follow independent autoregressive processes of order one. $\boldsymbol{h}_t$ is assumed to to have expectation zero due to identification issues discussed below, while $\boldsymbol{x}_t$ is assumed to have expectation $\boldsymbol{\mu_x}$. Equation (7.1) can thus be restated as

$$\begin{pmatrix} y_{1t} \\ y_{2t} \\ \vdots \\ y_{pt} \end{pmatrix} = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,q} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,q} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{p,1} & \beta_{p,2} & \cdots & \beta_{p,q} \end{pmatrix} \begin{pmatrix} \exp\left(\frac{1}{2}h_{1,t}\right)\zeta_{1t} & 0 & \cdots & 0 \\ 0 & \exp\left(\frac{1}{2}h_{2,t}\right)\zeta_{2t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \exp\left(\frac{1}{2}h_{q,t}\right)\zeta_{qt} \end{pmatrix}$$

$$+ \begin{pmatrix} \exp\left(\frac{1}{2}x_{1,t}\right)\epsilon_{1t} & 0 & \cdots & 0 \\ 0 & \exp\left(\frac{1}{2}x_{2,t}\right)\epsilon_{2t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \exp\left(\frac{1}{2}x_{p,t}\right)\epsilon_{pt} \end{pmatrix}.$$

We observe that $\mathbb{E}(\boldsymbol{y}_t) = \mathbf{0}$ and that the conditional covariance matrix is given by

$$\mathrm{Cov}(\boldsymbol{y}_t|\boldsymbol{h}_t, \boldsymbol{x}_t) = \boldsymbol{\beta}\boldsymbol{V}_t(\boldsymbol{h}_t)\boldsymbol{\beta}' + \boldsymbol{U}_t(\boldsymbol{x}_t). \tag{7.2}$$

The unconditional covariance matrix of $\boldsymbol{y}_t$ can be found by applying the law of total covariance:

$$\begin{aligned} \mathrm{Cov}(\boldsymbol{y}_t) &= \mathbb{E}(\mathrm{Cov}(\boldsymbol{y}_t|\boldsymbol{h}_t, \boldsymbol{x}_t)) + \mathrm{Cov}(\underbrace{\mathbb{E}(\boldsymbol{y}_t|\boldsymbol{h}_t, \boldsymbol{x}_t)}_{=\mathbf{0}}) \\ &= \mathbb{E}(\boldsymbol{\beta}\boldsymbol{V}_t(\boldsymbol{h}_t)\boldsymbol{\beta}' + \boldsymbol{U}_t(\boldsymbol{x}_t)) \\ &= \boldsymbol{\beta}\mathbb{E}(\boldsymbol{V}_t(\boldsymbol{h}_t))\boldsymbol{\beta}' + \mathbb{E}(\boldsymbol{U}_t(\boldsymbol{x}_t)) \\ &= \boldsymbol{\beta}\boldsymbol{V}_t\left(\frac{1}{2}\frac{\boldsymbol{\sigma_h^2}}{(1-\boldsymbol{\phi_h^2})}\right)\boldsymbol{\beta}' + \boldsymbol{U}_t\left(\boldsymbol{\mu_x} + \frac{1}{2}\frac{\boldsymbol{\sigma_x^2}}{(1-\boldsymbol{\phi_x^2})}\right), \end{aligned} \tag{7.3}$$

where

$$\boldsymbol{V}_t\left(\frac{1}{2}\frac{\boldsymbol{\sigma}_{\boldsymbol{h}}^2}{(1-\boldsymbol{\phi}_{\boldsymbol{h}}^2)}\right) = \text{diag}\left[\exp\left(\frac{1}{2}\frac{\sigma_{h_1}^2}{(1-\phi_{h_1}^2)}\right),\ldots,\exp\left(\frac{1}{2}\frac{\sigma_{h_q}^2}{(1-\phi_{h_q}^2)}\right)\right]$$

and

$$\boldsymbol{U}_t\left(\boldsymbol{\mu}_{\boldsymbol{x}}+\frac{1}{2}\frac{\boldsymbol{\sigma}_{\boldsymbol{x}}^2}{(1-\boldsymbol{\phi}_{\boldsymbol{x}}^2)}\right) = \text{diag}\left[\exp\left(\mu_{x_1}+\frac{1}{2}\frac{\sigma_{x_1}^2}{(1-\phi_{x_1}^2)}\right),\ldots,\exp\left(\mu_{x_p}+\frac{1}{2}\frac{\sigma_{x_p}^2}{(1-\phi_{x_p}^2)}\right)\right],$$

due to the stationarity of $\boldsymbol{h}_t$ and $\boldsymbol{x}_t$.

7.2 Identification issues

To prevent factor rotation we set the upper triangular part of $\boldsymbol{\beta}$ equal to zero, i.e. $\beta_{ij} = 0$ for $j > i$. If we don't identify the scaling of the jth column of $\boldsymbol{\beta}$ or the variance of f_{jt}, the model is not identified. There are different approaches to this problem in the literature. In Pitt and Shephard (1999) they set the diagonal of $\boldsymbol{\beta}$ equal to 1, i.e. $\beta_{ii} = 1$, and the mean of h_j equal to zero, i.e. $\mu_{h_j} = 0$ for $j = 1,\ldots q$, but they let the covariance of $\boldsymbol{\zeta}_t$ be a diagonal matrix with entries $\zeta_{ii} = \sigma_i$. Zhou et al. (2014) set the diagonal of the loading matrix equal to 1, but let the mean of h_j be unrestricted. By setting the diagonal equal to 1, the first q series are implied to be leading factors. Thus more care must be put into the variable ordering. To put less weight on this decision, we let the diagonal of $\boldsymbol{\beta}$ be unrestricted, but we fix the mean of the latent factor process to zero, i.e $\mu_{h_j} = 0$ (Kastner et al. (2017)):

$$h_{it} = \phi_{h_i} h_{i,t-1} + \sigma_{h_i}\eta_t, \quad i = 1,\ldots,q, \tag{7.4}$$

$$x_{jt} = \mu_{x_j} + \phi_{x_j}(x_{j,t-1} - \mu_{x_j}) + \sigma_{x_j}\epsilon_t, \quad j = 1,\ldots,p. \tag{7.5}$$

7.3 Bayesian Inference by Deep Interweaving MCMC

We will here give an overview of the algorithm proposed in Kastner et al. (2017) for sampling from the posterior of the MFSV model. First, we define priors for the parameters. In section 7.3.2 we discuss the sampling method proposed in Kastner and Frühwirth-Schnatter (2014) for the univariate SV model, as this will be an important building block for sampling the MFSV model. Section 7.3.3 presents the algorithm for sampling from the joint posterior in the MFSV model.

7.3.1 Prior distributions

For $i = 1, \ldots, p+q$ we assume independence of all parameters in the latent processes, i.e. $f(\mu_i, \phi_i, \sigma_i) = f(\mu_i)f(\phi_i)f(\sigma_i)$. The mean of the latent idiosyncratic processes, μ_i, is assumed to have a normal prior $\mathcal{N}(b_\mu, B_\mu)$. To guarantee that $-1 < \phi_i < 1$, we let $(\phi_i + 1)/2 \sim \mathcal{B}(a_0, b_0)$, implying

$$f(\phi_i) = \frac{1}{2B(a_0, b_0)} \left(\frac{\phi_i + 1}{2}\right)^{a_0 - 1} \left(\frac{1 - \phi_i}{2}\right)^{b_0 - 1},$$

where $B(a_0, b_0)$ is the beta function. The volatility of volatility, σ_i^2, is given a gamma prior, $\sigma_i^2 \sim \mathcal{G}\left(\frac{1}{2}, \frac{1}{2B_\sigma}\right)$, implying $\sigma_i > 0$. Lastly, for each element of the factor loading matrix, we choose a zero-mean Gaussian distribution, i.e. $\beta_{ij} \sim \mathcal{N}(0, B_\beta)$.

7.3.2 Sampling the univariate SV model

We will briefly discuss the method proposed in Kastner and Frühwirth-Schnatter (2014) for sampling the univariate SV-model, due to the fact that this is extensively used in the sampling of the MFSV-model.

Consider the univariate SV-model:

$$y_t = e^{h_t/2}\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, 1), \tag{7.6}$$

$$h_t = \mu + \phi(h_{t-1} - \mu) + \sigma\eta_t, \quad \eta_t \sim \mathcal{N}(0, 1). \tag{7.7}$$

This is known as the *centered parameterization* (C). The SV-model can also be parameterized in what is known as the *non-centered parameterization* (NC) by moving the parameter μ from the state equation into the observation equation:

$$y_t = \sqrt{\omega} e^{\sigma \tilde{h}_t/2} \epsilon_t, \tag{7.8}$$

where $\omega = e^{\mu}$ and $\tilde{h}_t = \frac{h_t - \mu}{\sigma}$.
Equation (7.6) can be rewritten as

$$\tilde{y}_t = \log y_t^2 = \log(e^{h_t} \epsilon_t^2) = h_t + \log \epsilon_t^2, \tag{7.9}$$

This is a linear, but non-Gaussian state space model. Omori et al. (2004) show that the distribution of $\log \epsilon_t^2$ can be approximated by a mixture of 10 normal distributions, i.e. $f(\log \epsilon_t^2) \approx \sum_{r_t=1}^{10} p_{r_t} \mathcal{N}(\log \epsilon_t^2 | m_{r_t}, s_{r_t}^2)$, where p_{r_t}, m_{r_t} and $s_{r_t}^2$ are the weight, mean and variance of the r_tth mixture variable, respectively. See Omori et al. (2004) for the values of p_{r_t}, m_{r_t} and $s_{r_t}^2$. We can therefore approximate Equation (7.9) as a linear and conditional Gaussian state space model:

$$\tilde{y}_t | r_t = m_{r_t} + h_t + s_{r_t} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, 1) \tag{7.10}$$

This makes it possible to do efficient MCMC sampling (Kastner and Frühwirth-Schnatter (2014)). The interweaving strategy proposed in Kastner and Frühwirth-Schnatter (2014) makes use of both the centered and the non-centered parameterization of the SV-model when sampling, and is based on the theory developed in Yu and Meng (2011), discussed below. Instead of sampling (μ, ϕ, σ) once per iteration on the Markov chain, we sample them twice, once in C and then again in NC. The sampling algorithm is presented in Algorithm 4.

For details regarding the conditional posterior density of $\boldsymbol{h}$, the parameters (μ, ϕ, σ) and the indicators r_t, the reader is referred to Kastner and Frühwirth-Schnatter (2014).

Algorithm 4: Sampling the univariate SV model

Initialize : Choose appropriate starting values for μ, ϕ, and σ as well as $\boldsymbol{h}$ and repeat the following steps

1 Sample $\boldsymbol{h} = (h_1, \ldots, h_T)$ (C).
2 Sample (μ, ϕ, σ) (C).
3 Move from C to NC by the transformation $\tilde{h}_t = \frac{h_t - \mu}{\sigma}$ for all t.
4 Resample (μ, ϕ, σ) (NC).
5 Move back to C by the transformation $h_t = \mu + \sigma \tilde{h}_t$ for all t.
6 Draw the indicators r_t for all t

7.3.3 Sampling the MFSV model

We start by introducing some notation. To denote specific rows and columns we use the "dot" notation, i.e. $\boldsymbol{\beta}_{\bullet,i}$ refers to the ith column, while $\boldsymbol{\beta}_{j,\bullet}$ refers to the jth row of $\boldsymbol{\beta}$. Let $\boldsymbol{h} = (\boldsymbol{h}_1 \ldots \boldsymbol{h}_T)$ denote the $q \times T$ matrix of the latent volatilities of the factors, $\boldsymbol{x} = (\boldsymbol{x}_1 \ldots \boldsymbol{x}_T)$ the $p \times T$ matrix of the latent volatilites of the idiosyncratic variance, and $\boldsymbol{f} = (\boldsymbol{f}_1 \ldots \boldsymbol{f}_T)$ the $q \times T$ matrix of the latent factors.

The sampling procedure is summarized in Algorithm 5, and we will here comment on each step.

Algorithm 5: Sampling the MFSV model

Initialize : Choose appropriate starting values for $\mu_i,\ i = 1, \ldots, p$, ϕ_j and $\sigma_j,\ j = 1, \ldots, p+q$, as well as $\boldsymbol{\beta}, \boldsymbol{h}, \boldsymbol{x}$ and $\boldsymbol{f}$ and repeat the following steps:

1 Perform p independent univariate SV updates for $\boldsymbol{x}_{\boldsymbol{i},\bullet}$ and the parameters associated with $\boldsymbol{x}_{\boldsymbol{i},\bullet}$ $\{\mu_i, \phi_i, \sigma_i\}$ conditioning on $\boldsymbol{f}$ and $\boldsymbol{\beta}$.
Perform q independent univariate SV updates for $\boldsymbol{h}_{i,\bullet}$ and the parameters associated with $\boldsymbol{h}_{i,\bullet}$ $\{\phi_i, \sigma_i\}$ conditioning on $\boldsymbol{f}$ and $\boldsymbol{\beta}$.
2 For $i = 1, \ldots, q$, sample each row $\beta_{i,\bullet}$ conditioning on $\boldsymbol{f}, \boldsymbol{y}_{i,\bullet}, \boldsymbol{h}_{i,\bullet}, \boldsymbol{x}_{i,\bullet}$.
if *deep interweaving* **then**
Redraw the diagonal elements of $\boldsymbol{\beta}$ through interweaving into the state equation for the latent volatilities.
3 For $t = 1, \ldots, T$, sample $\boldsymbol{f}_t$ conditioning on $\boldsymbol{\beta}, \boldsymbol{y}_{i,\bullet}, \boldsymbol{h}_{i,\bullet}, \boldsymbol{x}_{i,\bullet}$.

1. Sampling $\boldsymbol{h}$ and $\boldsymbol{x}$. Conditioning on knowing the latent factors $\boldsymbol{f}$ and the loading matrix $\boldsymbol{\beta}$ we rewrite the observational Equation (7.1):

$$\log(\boldsymbol{y}_t - \boldsymbol{\beta}\boldsymbol{f}_t)^2 = \log \boldsymbol{U}_t(\boldsymbol{x}_t) + \log \boldsymbol{\epsilon}_t^2 \tag{7.11}$$

$$\log \boldsymbol{f}_t^2 = \log \boldsymbol{V}_t(\boldsymbol{h}_t) + \log \boldsymbol{\zeta}_t \tag{7.12}$$

This implies that for each time series and factor we have

$$\log(y_{it} - \boldsymbol{\beta}_{i,\bullet}\boldsymbol{f}_t)^2 = x_{it} + \log \epsilon_{it}^2 \quad i = 1, \ldots, p, \tag{7.13}$$

$$\log f_{jt}^2 = h_{jt} + \log \zeta_{jt}^2 \quad j = 1, \ldots, q. \tag{7.14}$$

Hence, we have $p+q$ independent univariate SV-models. The latent volatilities and the parameters (μ, ϕ, σ) can therefore be sampled by the method sketched in section 7.3.2.

2. Sampling the loadings. Conditioning on $\boldsymbol{f}$ and the latent volatilites $\boldsymbol{h}$ and $\boldsymbol{x}$,

$$y_{it} = \boldsymbol{\beta}_{i,\bullet}\boldsymbol{f}_t + e^{x_{it}/2}\epsilon_{it} \quad \Rightarrow \quad y_{it} \sim \mathcal{N}(\boldsymbol{\beta}_{i,\bullet}\boldsymbol{f}_t, e^{x_{it}/2}).$$

Multiplying both sides by $e^{-x_{it}/2}$ and defining $\tilde{y}_{it} = y_{it}e^{-x_{it}/2}$ to be our scaled observations implies

$$\tilde{y}_{it} = \boldsymbol{\beta}_{i,\bullet}\boldsymbol{f}_t e^{-x_{it}/2} + \epsilon_{it} \quad \Rightarrow \quad \tilde{y}_{it} \sim \mathcal{N}(\boldsymbol{\beta}_{i,\bullet}\boldsymbol{f}_t e^{-x_{it}/2}, 1).$$

Letting $\tilde{q} = \min(i, q)$, i.e. the number of unrestricted elements in row i of $\boldsymbol{\beta}$, we have

$$\tilde{\boldsymbol{y}}_i \sim \mathcal{N}_T(\boldsymbol{X}_i\boldsymbol{\beta}'_{\boldsymbol{i},\bullet}, \boldsymbol{I}), \tag{7.15}$$

where $\tilde{\boldsymbol{y}}_i = (\tilde{y}_{i1}, \ldots, \tilde{y}_{iT})'$ is the ith observation vector and $\boldsymbol{X}_i$ is defined as

$$\boldsymbol{X}_i = \begin{pmatrix} f_{11}e^{-x_{i1}/2} & \cdots & f_{\tilde{q}1}e^{-x_{i1}/2} \\ \vdots & & \vdots \\ f_{1T}e^{-x_{iT}/2} & \cdots & f_{\tilde{q}T}e^{-x_{iT}/2} \end{pmatrix}.$$

This implies that sampling the ith row of $\boldsymbol{\beta}$ can be viewed as a Bayesian regression problem with design matrix $\boldsymbol{X}_i$, parameter vector $\boldsymbol{\beta_{i,\bullet}}$ and unit variance. For each $i = 1, \ldots, p$, given that all elements of $\boldsymbol{\beta}$ have a $\mathcal{N}(0, B_\beta)$ prior, sampling from $\boldsymbol{\beta_{i,\bullet}}|\boldsymbol{f}, \boldsymbol{y}_{i,\bullet}, \boldsymbol{h}_{i,\bullet}, \boldsymbol{x_{i,\bullet}}$ is achieved by performing a Gibbs-update from

$$\boldsymbol{\beta}'_{i,\bullet}|\boldsymbol{f}, \boldsymbol{y}_{i,\bullet}, \boldsymbol{h}_{i,\bullet}, \boldsymbol{x}_{i,\bullet} \sim \mathcal{N}_{\tilde{q}}(\boldsymbol{b}_{iT}, \boldsymbol{B}_{iT}), \tag{7.16}$$

where $\boldsymbol{B}_{iT} = (\boldsymbol{X}'_i\boldsymbol{X}_i + B_\beta^{-1}\boldsymbol{I})^{-1}$ and $\boldsymbol{b}_{iT} = \boldsymbol{B}_{iT}\boldsymbol{X}'_i\tilde{\boldsymbol{y}}_i$. It is worth mentioning that in the frequentist setting, $\boldsymbol{b}_{iT}$ is the ridge estimator with regularization parameter $\lambda = 1/B_\beta$.

3. Sampling the factors. On the other hand, conditioning on knowing the latent volatilities and the loadings $\boldsymbol{\beta}$, is also a Bayesian regression problem. We have

$$\tilde{\boldsymbol{y}}_t \sim \mathcal{N}_p(\boldsymbol{X}_t\boldsymbol{f}_t, \boldsymbol{I}),$$

where $\tilde{\boldsymbol{y}}_t = (\tilde{y}_{1t}, \ldots, \tilde{y}_{qt})'$ again is our scaled observations and

$$\boldsymbol{X}_t = \begin{pmatrix} \beta_{11}e^{-x_{it}/2} & \cdots & \beta_{1q}e^{-x_{it}/2} \\ \vdots & & \vdots \\ \beta_{p1}e^{-x_{it}/2} & \cdots & \beta_{qp}e^{-x_{it}/2} \end{pmatrix},$$

is our design matrix. For each $t = 1, \ldots, T$, sampling from the posterior of $\boldsymbol{f}_t$ is obtained by a Gibbs-update from

$$\boldsymbol{f}_t|\boldsymbol{\beta}, \boldsymbol{y}_t, \boldsymbol{h}_t, \boldsymbol{x}_t \sim \mathcal{N}_q(\boldsymbol{b}_{pt}, \boldsymbol{B}_{pt}), \tag{7.17}$$

where $\boldsymbol{B}_{pt} = (\boldsymbol{X}'_t\boldsymbol{X}_t + \boldsymbol{V}_t(\boldsymbol{h}_t)^{-1})^{-1}$, since $\boldsymbol{f}_t|\boldsymbol{h_t} \sim \mathcal{N}_q(0, \boldsymbol{V}_t(\boldsymbol{h}_t))$, and $\boldsymbol{b}_{pt} = \boldsymbol{B}_{pt}\boldsymbol{X}'_t\tilde{\boldsymbol{y}}_t$.

7.3.4 Performing Deep Interweaving

Deep interweaving is based on reparametrizing the baseline model (Equation 7.1) and then re-sampling a subset of the parameters of interest. It is an application of the

method proposed in Yu and Meng (2011). Instead of using a single data augmentation scheme, the authors combine two different schemes by "going back and fourth" between them when doing MCMC sampling. This has been shown, both theoretically and empirically, to lead to faster convergence and better mixing of the Markov chain.

In the context of stochastic volatility, this is obtained by moving between the non-centered and centered parameterization of the latent processes. This leads to moving some of the model parameters back and forth between the observational equation and state equation, as seen in section 7.3.2 for the univariate SV model. We will show how this can be used to redraw the diagonal of $\boldsymbol{\beta}$ in Step 2 of Algorithm 5.

The parameterization behind deep interweaving is based on the following factor model:

$$\boldsymbol{y}_t = \boldsymbol{\beta}^\star \boldsymbol{f}_t^\star + \boldsymbol{U}_t(\boldsymbol{x}_t)\boldsymbol{\epsilon_t}, \quad \boldsymbol{f}_t^\star = \boldsymbol{V}_t(\boldsymbol{h}_t^\star)\boldsymbol{\zeta}_t, \tag{7.18}$$

where $\boldsymbol{\beta}^\star$ is a lower triangular matrix with diagonal elements equal to one, but $h_{it}^\star$ now follows a centered parameterization, i.e.

$$h_{it}^\star = \mu_{h_i} + \phi_{h_i}(h_{i,t-1}^\star - \mu_{h_j}) + \sigma_{h_j}\eta_{it}, \tag{7.19}$$

where $\mu_{h_j} = \log \beta_{jj}^2$. This is motivated by the fact that

$$f_{jt}|\beta_{jj}, h_{jt} \sim \mathcal{N}(0, \beta_{jj}^2 e^{h_{jt}}) = \mathcal{N}\left(0, e^{\log \beta_{jj}^2 + h_{jt}}\right) = \mathcal{N}(0, e^{h_{jt}^\star}).$$

By this reparameterization, the diagonal of $\boldsymbol{\beta}$ is moved from the observational equation into the state equation.This parameterization is obtained by applying the linear transformation

$$\boldsymbol{f}_t^\star = \boldsymbol{D}\boldsymbol{f}_t, \quad t = 1, \ldots, T, \quad \boldsymbol{\beta}^\star = \boldsymbol{\beta}\boldsymbol{D}^{-1}, \tag{7.20}$$

to the factors and the loading matrix, where $\boldsymbol{D} = \text{diag}(\beta_{11}, \ldots, \beta_{qq})$, and the following transformation in the volatility of the factor:

$$h_{jt}^\star = h_{jt} + \log \beta_{jj}^2, \quad t = 1, \ldots, T, \ j = 1, \ldots, q. \tag{7.21}$$

Next, denote the original posterior draw before performing deep interweaving by $\boldsymbol{\beta}_{\bullet,j}^{\text{old}}, \boldsymbol{f}_{j,\bullet}^{\text{old}}$ and $\boldsymbol{h}_{j,\bullet}^{\text{old}}$. We then resample $(\beta_{11}^{\text{new}}, \ldots, \beta_{qq}^{\text{new}})$ in the centered parameterization Equation (7.21), conditioning on $\boldsymbol{h}^{\star}$ and the new factor loading matrix $\boldsymbol{\beta}^{\star}$. This is done indirectly by sampling $\mu_{h_j} = \log \beta_{jj}^2$. Kastner et al. (2017) show that μ_{h_j} has a non-standard kernel and sample μ_{h_j} by applying a Metropolis-Hastings update. We then use the new sample of $(\beta_{11}^{\text{new}}, \ldots, \beta_{qq}^{\text{new}})$ to transform back to new draws in the baseline parameterization, which are updated by the following formula:

$$\boldsymbol{\beta}_{\bullet,j} = \frac{\beta_{jj}^{\text{new}}}{\beta_{jj}^{\text{old}}} \boldsymbol{\beta}_{\bullet,j}^{\text{old}}, \quad \boldsymbol{f}_{j,\bullet} = \frac{\beta_{jj}^{\text{new}}}{\beta_{jj}^{\text{old}}} \boldsymbol{f}_{j,\bullet}^{\text{old}}, \quad \boldsymbol{h}_{j,\bullet} = \boldsymbol{h}_{j,\bullet}^{\text{old}} + 2 \log \left| \frac{\beta_{jj}^{\text{new}}}{\beta_{jj}^{\text{old}}} \right|. \tag{7.22}$$

This is then repeated for $j = 1, \ldots, q$.

Note that we have not covered the details on how to derive the conditional posterior of the transformed factors $\mu_j = \log \beta_{jj}^2$. See Kastner et al. (2017) for more details.

Chapter 8

Simulation Study

In this chapter we investigate the parameter behaviour of the two dimensional factor model for two methods:

- **Maximum Likelihood**, where the latent variables are integrated out by the Laplace approximation described in chapter 3. The model is implemented using the R package **TMB** described in section 6.2.
- **Deep Interweaving MCMC**, introduced in chapter 7. The sampling was done by the use of the R package **factorstochvol**.

As we will see, convergence of the likelihood is not a given. We therefore study the characteristic function and the cumulative generating function of our model to investigate possible restrictions that can make convergence easier. Hamiltonian Monte Carlo is very time consuming and is therefore not included in the simulation study.

8.1 Analysis of two dimensions with one factor

We analyze the simplest MFSV model, with two time series and one factor. Even in the simplest case, parameter estimation is non-trivial. We have the following structure:

$$\begin{aligned} y_{1t} &= \beta_1 e^{\frac{1}{2}h_t}\zeta_t + e^{\frac{1}{2}x_{1t}}\epsilon_{1t} \\ y_{2t} &= \beta_2 e^{\frac{1}{2}h_t}\zeta_t + e^{\frac{1}{2}x_{2t}}\epsilon_{2t}, \end{aligned} \tag{8.1}$$

where, again, h_t, x_{1t} and x_{2t} are independent AR(1) processes, with mean zero, μ_1 and μ_2 respectively, and ζ_t, ϵ_{1t}, and ϵ_{2t} are all independent standard normal random variables. As we saw in section 7.1, $\boldsymbol{y_t}$ has expectation zero and the conditional covariance matrix is given by

$$\begin{aligned} \mathrm{Cov}(\boldsymbol{y_t}|h_t, \boldsymbol{x_t}) &= \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} e^{h_t} \begin{pmatrix} \beta_1 & \beta_2 \end{pmatrix} + \begin{pmatrix} e^{x_{1t}} & 0 \\ 0 & e^{x_{2t}} \end{pmatrix} \\ &= \begin{pmatrix} \beta_1^2 e^{h_t} + e^{x_{1t}} & \beta_1\beta_2 e^{h_t} \\ \beta_1\beta_2 e^{h_t} & \beta_2^2 e^{h_t} + e^{x_{2t}} \end{pmatrix} \end{aligned} \tag{8.2}$$

8.1.1 An example using MLE

Consider the model described above. We simulate states and measurements with $T = 3000$ observations, $\boldsymbol{\beta} = (0.7, 1, 3)$, $\phi_h = 0.95$, $\log \sigma_h = -1.4$, $\boldsymbol{\mu_x} = (-1.3, -0.8)$, $\boldsymbol{\phi_x} = (0.95, 0.94)$ and $\log \boldsymbol{\sigma_x} = (-1, -0.8)$. To be able to do unrestricted optimization we do some simple transformations. To ensure a positive standard deviation, we estimate the logarithm, i.e. $\tau = \log \sigma$, such that $\sigma = \exp(\tau)$. By the invariance property of the MLE, $\hat{\sigma} = \exp(\hat{\tau})$. To ensure $\phi \in (-1, 1)$, we estimate

$$\tilde{\phi} = \log\left(\frac{\phi + 1}{1 - \phi}\right),$$

so that

$$\phi = \frac{\exp(\tilde{\phi}) - 1}{1 + \exp(\tilde{\phi})} \in (-1, 1).$$

The observations from the simulation can be seen in figure 8.1. Figure 8.2 shows the true and the estimated latent log-variance $\pm$ two times the standard error. Table 8.1 shows model 8.1 fitted to the simulated values. We observe that the estimated values

are well within two times the standard errors. This indicate that our method manages to capture the latent dynamics in this example.

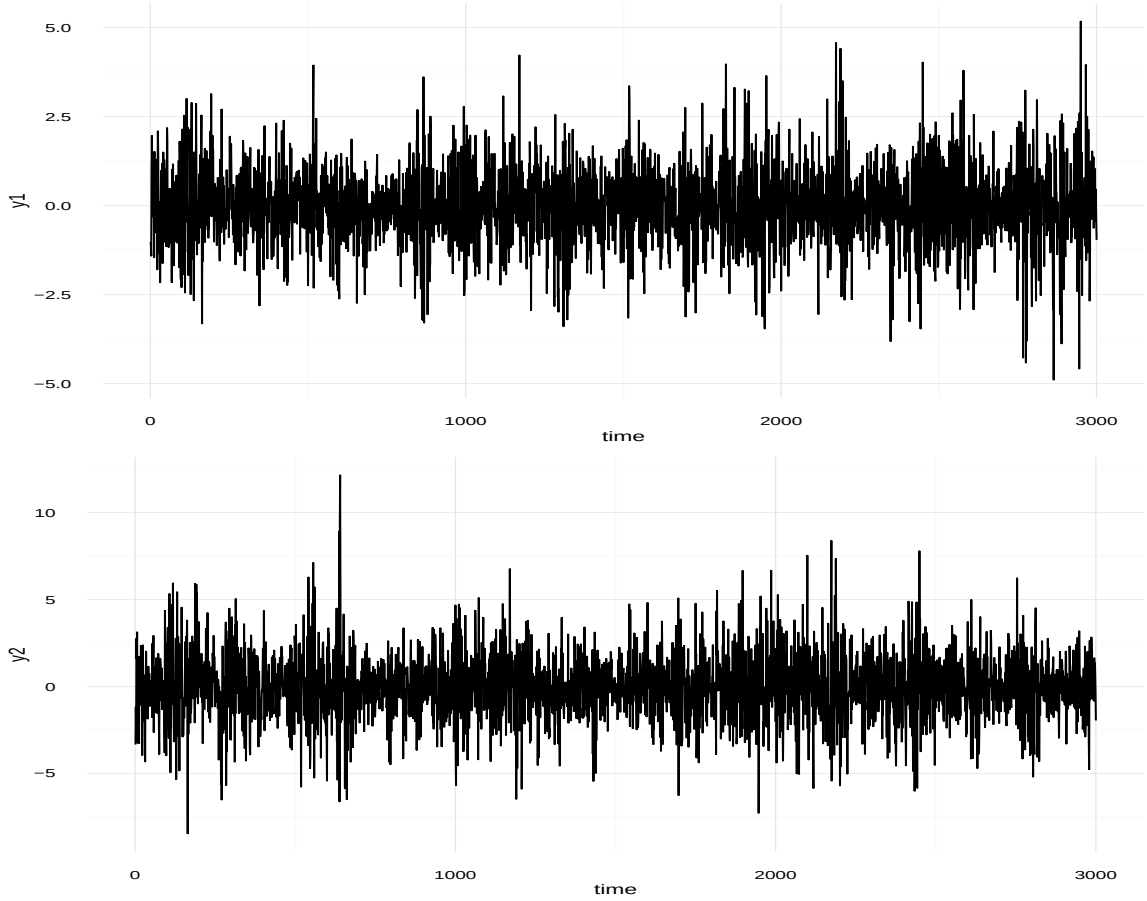


Fig. 8.1 Observations from simulated states.

8.1.2 Simulation study

To investigate the statistical efficiency of our method, we generate 1000 datasets, each with $T = 5000$ observations from model 8.1. We then estimate our model to each simulated dataset. We set $\boldsymbol{\beta} = (0.7, 1, 3), \phi_h = 0.95, \log \sigma_h = -1.4$, $\boldsymbol{\mu_x} = (-1.3, -0.8), \boldsymbol{\phi_x} = (0.95, 0.94)$ and $\log \boldsymbol{\sigma_x} = (-1.1, -1)$. This is also done for the DIMCMC method proposed in Kastner et al. (2017). Since this is in the Bayesian setting, prior distributions need to be specified for all parameters. We apply the same priors as in Kastner et al. (2017), and set $\beta_i \sim \mathcal{N}(0, 1)$, $\mu_i \sim \mathcal{N}(0, 10)$, for $i = 1, 2$,

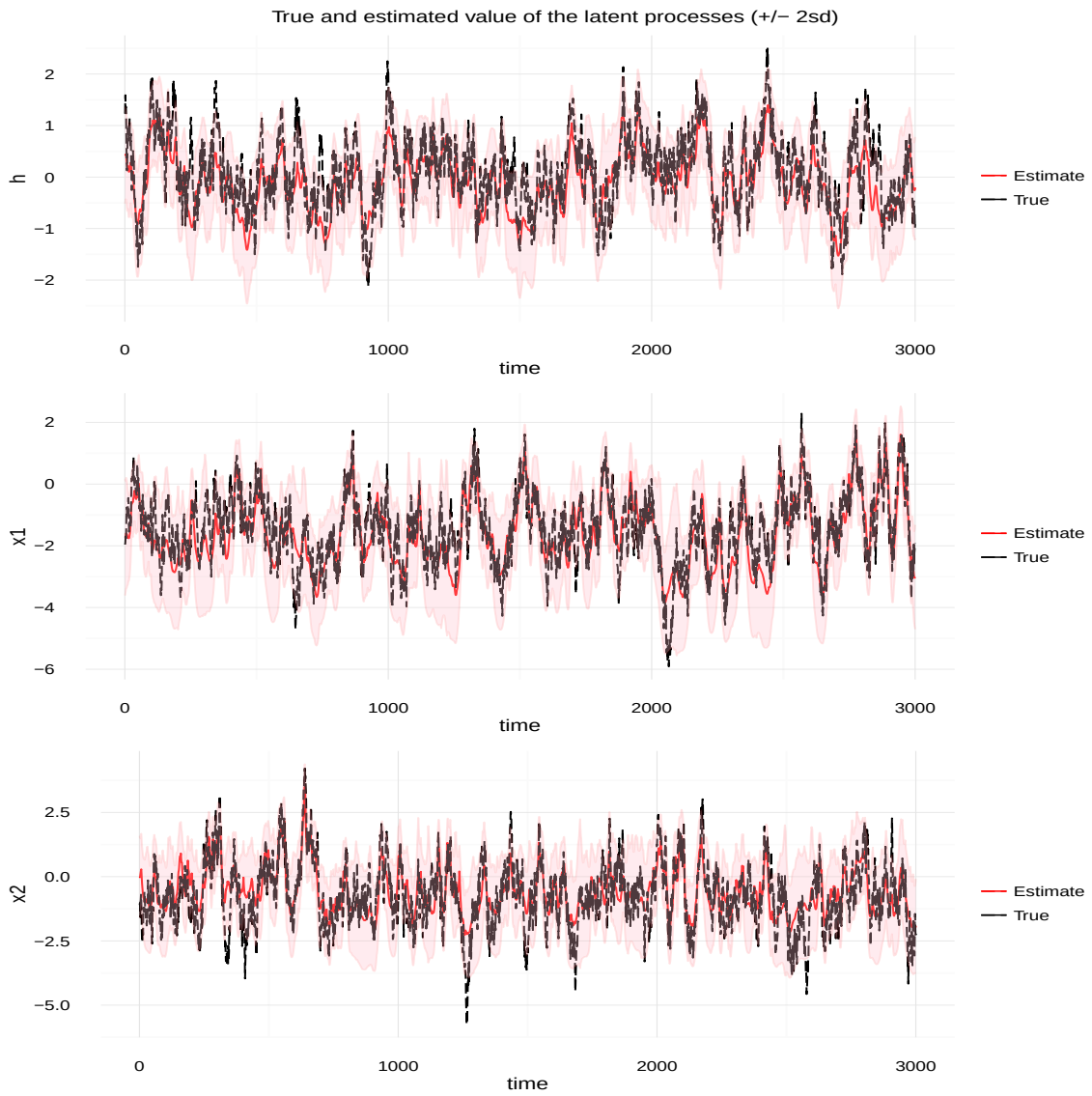


Fig. 8.2 True and estimated value of the latent processes h_t, x_{1t} and x_{2t} ($\pm 2\times$standard error).

and $(\phi_j + 1)/2 \sim \mathcal{B}(20, 1.5)$ and $\sigma_j^2 \sim \mathcal{G}(\frac{1}{2}, \frac{1}{2})$, for $j = 1, 2, 3$. We then draw 110 000 samples, where the first 10 000 are discarded as burn-in. Starting values for the parameters were $\boldsymbol{\beta} = (1, 1)$, $\phi_h = 0.9$, $\log \sigma_h = -1$, $\boldsymbol{\mu_x} = (-1, -1)$, $\boldsymbol{\phi_x} = (0.9, 0.9)$ and $\log \boldsymbol{\sigma_x} = (-1, -1)$. All latent variables were initialized in zero.

In 611 of the 1000 simulations we obtained convergence with ML, and in 389 cases we did not. When the ML converged there was little difference between the methods, and both methods had low bias, as seen in table 8.2.

	Estimate	SE	True Value
β_1	0.764	0.033	0.7
β_2	1.416	0.062	1.3
ϕ_h	0.947	0.014	0.95
$\log \sigma_h$	-1.499	0.144	-1.4
ϕ_{x_1}	0.963	0.008	0.95
ϕ_{x_2}	0.934	0.013	0.94
$\log \sigma_{x_1}$	-1.027	0.082	-1
$\log \sigma_{x_2}$	-0.865	0.094	-0.8
μ_1	-1.649	0.212	-1.3
μ_2	-0.662	0.164	-0.8

Table 8.1 Estimated values, standard errors and true values for the example in section 8.1.1.

As seen in table 8.3, non-convergence does affect some of the parameters, especially μ_1 and μ_2. The bias is substantially bigger compared to when the likelihood converged (see table 8.2). On average, μ_1 and μ_2 misses with 0.21 and -0.17, respectively. This indicates that even tho the algorithm does not converge, the estimates are close to their real value. Standard error is not reported, due to the fact that Hessian was not positive definite. DIMCMC was not affected by the fact that the likelihood did not converge, as can be seen on the right hand side of table 8.3. Figure 8.4 shows histograms of the parameter estimates for MLE, while figure 8.5 shows it for the DIMCMC.

Table 8.4 report the mean, max and minimum efficient sample size over the simulated datasets, together with the mean, max, and minimum efficient sample size per unit time. The R package **coda** was used to estimate the efficient sample size.

	MLE			DIMCMC			
	Estimate	SE	Bias	Estimate	SE	Bias	True value
β_1	0.693	0.027	-0.007	0.690	0.028	-0.010	0.7
β_2	1.291	0.051	-0.009	1.281	0.052	-0.019	1.3
ϕ_h	0.95	0.009	0	0.949	0.009	-0.001	0.95
$\log \sigma_h$	-1.431	0.091	-0.039	-1.392	0.094	0.008	-1.4
ϕ_{x_1}	0.953	0.007	0.003	0.948	0.009	-0.002	0.95
ϕ_{x_2}	0.952	0.008	0.013	0.940	0.013	0.0002	0.94
$\log \sigma_{x_1}$	-1.068	0.070	0.032	-1.083	0.088	0.017	-1.1
$\log \sigma_{x_2}$	-0.987	0.079	0.015	-1.005	0.122	-0.005	-1
μ_{x_1}	-1.286	0.126	0.014	-1.313	0.121	-0.013	-1.3
μ_{x_2}	-0.776	0.164	0.024	-0.810	0.162	-0.010	-0.8

Table 8.2 Mean estimated, mean standard errors and bias for the simulation study across the $n = 611$ simulated datasets for which an MLE was obtained.

	MLE			DIMCMC			
	Estimate	SE	Bias	Estimate	SE	Bias	True value
β_1	0.717	-	0.017	0.708	0.028	-0.010	0.7
β_2	1.267	-	-0.033	1.316	0.052	-0.019	1.3
ϕ_h	0.946	-	-0.004	0.947	0.009	-0.001	0.95
$\log \sigma_h$	-1.410	-	-0.005	-1.392	0.095	0.008	-1.4
ϕ_{x_1}	0.957	-	0.007	0.947	0.009	-0.002	0.95
ϕ_{x_2}	0.959	-	0.019	0.936	0.014	0.0002	0.94
$\log \sigma_{x_1}$	-1.031	-	0.069	-1.050	0.088	0.017	-1.1
$\log \sigma_{x_2}$	-0.973	-	0.027	-0.954	0.122	-0.005	-1
μ_{x_1}	-1.094	-	0.21	-1.339	0.123	-0.013	-1.3
μ_{x_2}	-0.97	-	-0.17	-0.832	0.166	-0.010	-0.8

Table 8.3 Mean estimates, mean standard errors and bias for the simulation study across the $n = 389$ simulated datasets for which an MLE was not obtained.

	Mean ESS	Max ESS	Min ESS	Mean ESS/T	Max ESS/T	Min ESS/T
β_1	11314.704	30282.385	3850.535	13.754	37.443	4.452
β_2	5849.313	16068.385	1606.036	7.110	19.779	1.857
ϕ_h	1818.890	2533.804	1115.217	2.211	3.117	1.323
$\log \sigma_h$	1310.464	1713.710	840.212	1.593	2.111	1.000
ϕ_{x_1}	1650.482	2524.264	1077.297	2.006	3.118	1.277
ϕ_{x_2}	657.824	1027.491	273.080	0.800	1.263	0.324
$\log \sigma_{x_1}$	946.625	1351.090	675.438	1.151	1.669	0.778
$\log \sigma_{x_2}$	448.518	726.852	225.436	0.545	0.895	0.267
μ_{x_1}	1080.586	2027.386	559.354	1.314	2.496	0.631
μ_{x_2}	310.884	667.728	118.607	0.378	0.822	0.140

Table 8.4 Sampling efficiency for DIMCMC based on parameter estimation of 1000 datasets. The mean runtime was 822 seconds (including 10 000 warm-up).

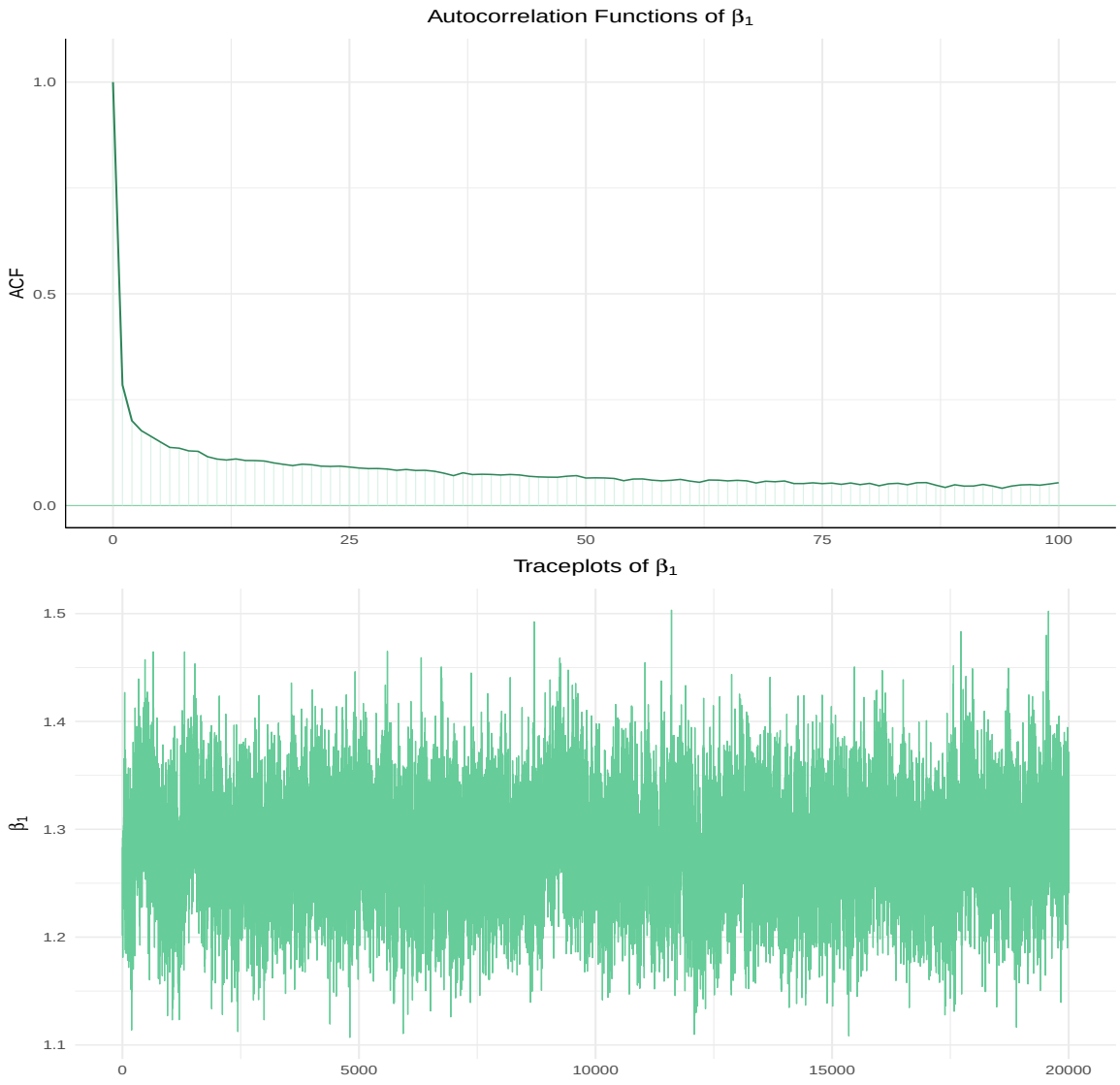


Fig. 8.3 Trace plot of first 20 000 draws from $p(\beta_1|\boldsymbol{y})$ after burn-in (Top) and empirical autocorrelation function of all 100 000 draws.

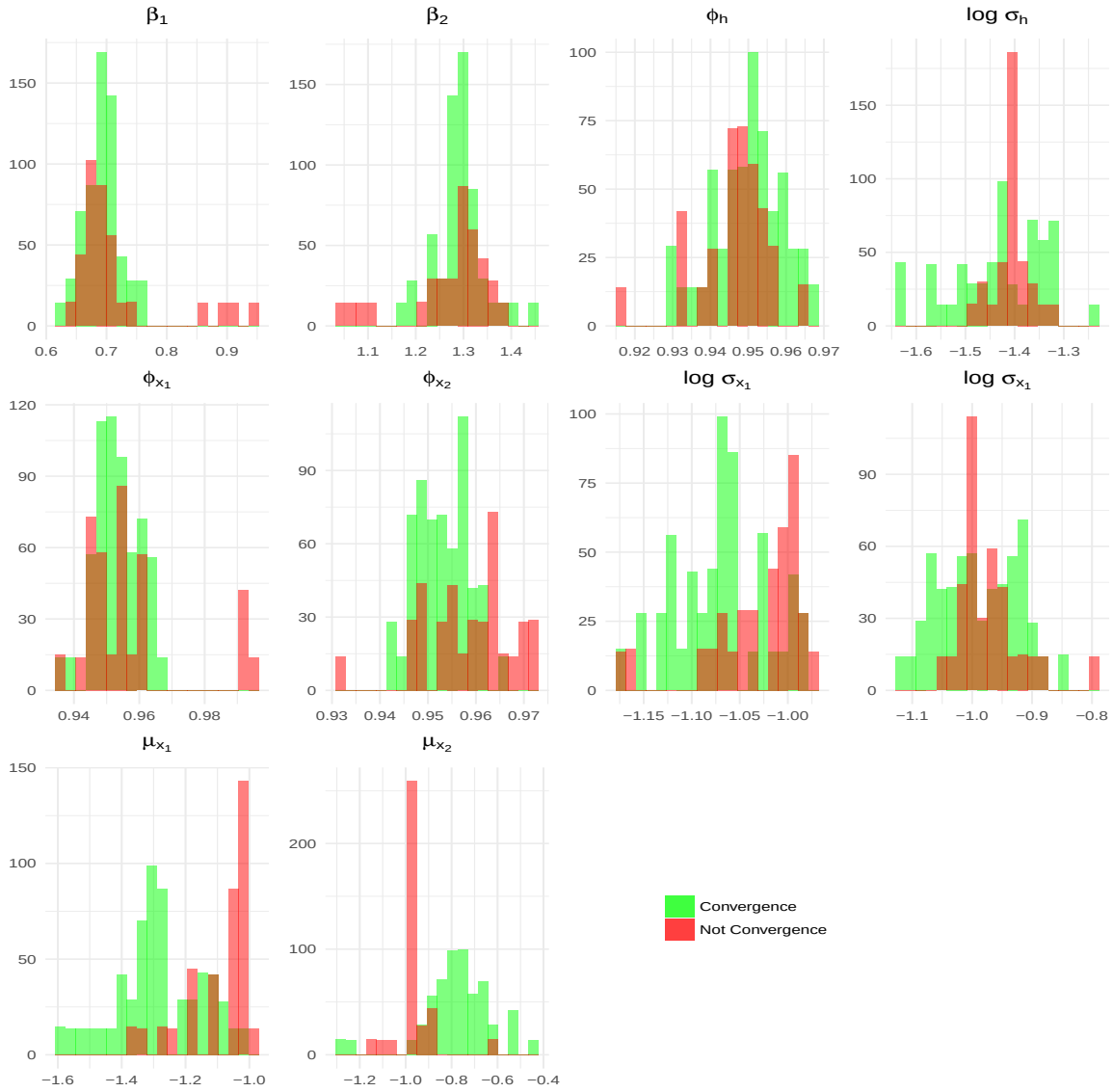


Fig. 8.4 Histogram of maximum likelihood estimates. Green color corresponds to when convergence was obtained, and red to when it was not obtained.

Fig. 8.5 Histogram of the posterior mean divided into two groups, when ML converged (green) and when it did not (red).

To get a better understanding of our model, we investigate the characteristic function and the cumulative generating function (CGF).

8.2 Restrictions found through the CGF

In this section we will study the characteristic function and the cumulative generating function of our model. The characteristic function $\varphi_{\boldsymbol{X}} : \mathbb{R}^n \to \mathbb{C}$ of a random vector

$\boldsymbol{X}$ is defined as $\varphi_{\boldsymbol{X}}(\boldsymbol{t}) = \mathbb{E}\Big(\exp(i\boldsymbol{t}'\boldsymbol{X})\Big)$, where $i = \sqrt{-1}$ is the imaginary unit. The cumulative generating function $H_{\boldsymbol{X}}$ is defined as the logarithm of the characteristic function, i.e. $H_{\boldsymbol{X}}(\boldsymbol{t}) = \log \varphi_{\boldsymbol{X}}(\boldsymbol{t})$.

For easier calculations, we can reparametrize our model in the following way:

$$\begin{aligned} y_{1t} &= \beta_1 v_h + v_1 \\ y_{2t} &= \beta_2 v_h + v_2 \end{aligned} \tag{8.3}$$

where

$$v_h = \exp\left(\frac{1}{2}h_t\right)\zeta_t, \quad v_1 = \exp\left(\frac{1}{2}x_{1t}\right)\epsilon_{1t}, \quad v_2 = \exp\left(\frac{1}{2}x_{2t}\right)\epsilon_{2t}.$$

Based on this reparameterization, we can calculate the characteristic function:

$$\begin{aligned} \varphi_{Y_1,Y_2}(s_1, s_2) &= \mathbb{E}\bigg(\exp\Big(is_1y_1 + is_2y_2 \Big)\bigg) \\ &= \mathbb{E}\bigg(\exp\Big(is_1(\beta_1 v_h + v_1) + is_2(\beta_2 v_h + v_2) \Big)\bigg) \\ &= \mathbb{E}\bigg(\exp\Big(iv_h(s_1\beta_1 + s_2\beta_2) + is_1v_1 + is_2v_2 \Big)\bigg) \\ &= \varphi_{V_h}(s_1\beta_1 + s_2\beta_2)\varphi_{V_1}(s_1)\varphi_{V_2}(s_2), \end{aligned} \tag{8.4}$$

due to the independence of v_h, v_1 and v_2.

The cumulative generating function is just the logarithm of the characteristic function. Therefore,

$$H_{Y_1,Y_2}(s_1, s_2) = \log \varphi_{Y_1,Y_2}(s_1, s_2) = H_{V_h}(s_1\beta_1 + s_2\beta_2) + H_{V_1}(s_1\beta_3) + H_{V_2}(s_2\beta_4). \tag{8.5}$$

The cumulants κ_{rs} of a bivariate CGF is defined as

$$\kappa_{mr} = \frac{\partial^{m+r}}{\partial s_1^m \partial s_2^r} H_{Y_1,Y_2}(s_1, s_2)\Big|_{(s_1,s_2)=(0,0)}. \tag{8.6}$$

Therefore,

$$\begin{aligned}\kappa_{mr} &= \frac{\partial^{m+r}}{\partial s_1^m \partial s_2^r} H_{Y_1,Y_2}(s_1, s_2) \\ &= \frac{\partial^{m+r}}{\partial s_1^m \partial s_2^r} H_{V_h}(s_1\beta_1 + s_2\beta_2) + \frac{\partial^{m+r}}{\partial s_1^m \partial s_2^r} H_{V_1}(s_1\beta_3) + \frac{\partial^{m+r}}{\partial s_1^m \partial s_2^r} H_{V_2}(s_2\beta_4),\end{aligned}$$

evaluated at $(s_1, s_2) = (0, 0)$. We can observe that

$$\begin{aligned}\frac{\partial^{m+r}}{\partial s_1^m \partial s_2^r} H_{V_1}(s_1\beta_3) = 0, \quad r \geq 1 \\ \frac{\partial^{m+r}}{\partial s_1^m \partial s_2^r} H_{V_2}(s_2\beta_4) = 0, \quad m \geq 1,\end{aligned}$$

since they are functions of only s_1 and s_2, respectively.

Furthermore, by the chain rule

$$\kappa_{mr} = \frac{\partial^{m+r}}{\partial s_1^m \partial s_2^r} H_{V_h}(s_1\beta_1 + s_2\beta_2) = \beta_1^m \beta_2^r H_{V_h}^{m+r}(0) \quad \text{for} \quad m, r \geq 1. \tag{8.7}$$

Therefore, κ_{mr} is a function of β_1, β_2 and $H_{V_h}^{m+r}(0)$, which again is a function of ϕ_h and σ_h, so all higher moments decide the product $\beta_1\beta_2$, ϕ_h and σ_h. From the fact that all higher moments include the product of $\beta_1\beta_2$, we investigate what the properties of the model is if $\beta_1 = \beta_2$.

8.3 Simulation with restrictions

Based on the expression derived above, we investigate convergence properties of our model when $\beta_1 = \beta_2$. As in section 8.1.2, we simulate $n = 1000$ datasets, each with $T = 5000$ observations. The parameters are set to the same, except β, which is set to 0.7.

In table 8.5 the results are reported. Convergence was obtained in 996 out of 1000 datasets. In contrast to the simulation in section 8.1.2, from our experience, non-convergence in the restricted model was a result of poor starting values, and was

	Estimate	SE	Bias	True Value
β	0.709	0.027	0.009	0.7
ϕ_h	0.950	0.008	0	0.95
$\log \sigma_h$	-1.420	0.10	-0.02	-1.4
ϕ_{x_1}	0.957	0.008	0.007	0.95
ϕ_{x_2}	0.945	0.008	0.005	0.94
$\log \sigma_{x_1}$	-1.236	0.090	-0.14	-1.1
$\log \sigma_{x_2}$	-1.090	0.072	-0.09	-1
μ_{x_1}	-1.290	0.113	0.01	-1.3
μ_{x_2}	-0.781	0.097	0.019	-0.8

Table 8.5 Estimated values, standard errors, bias and true values for the $n = 996$ datasets where convergence was obtained.

fixed when new starting values were used, while in the unrestricted model this was not the reason for non-convergence. All parameters have low bias, with exception of $\log \sigma_{x_1}$ and $\log \sigma_{x_2}$, which are underestimated. A histogram of the parameter estimates is seen in figure 8.6.

The increase in convergence, from 611 to 996 out of 1000 datasets indicate that for maximum likelihood, there is an identification problem when β_1 and β_2 are unrestricted during estimation. A big drawback of this restriction is that the model can only estimate positive correlations.

This simulation experiment assumes that in the real data generating process, $\beta_1 = \beta_2$. When this is not the case, the likelihood does not converge. Thus, when applied on real data, where the model is just an approximation, the probability of convergence will be lower than in this experiment. This is seen in chapter 10. Another natural question is then if the Laplace approximation is the reason for why the likelihood does not converge. We discuss this in the next chapter.

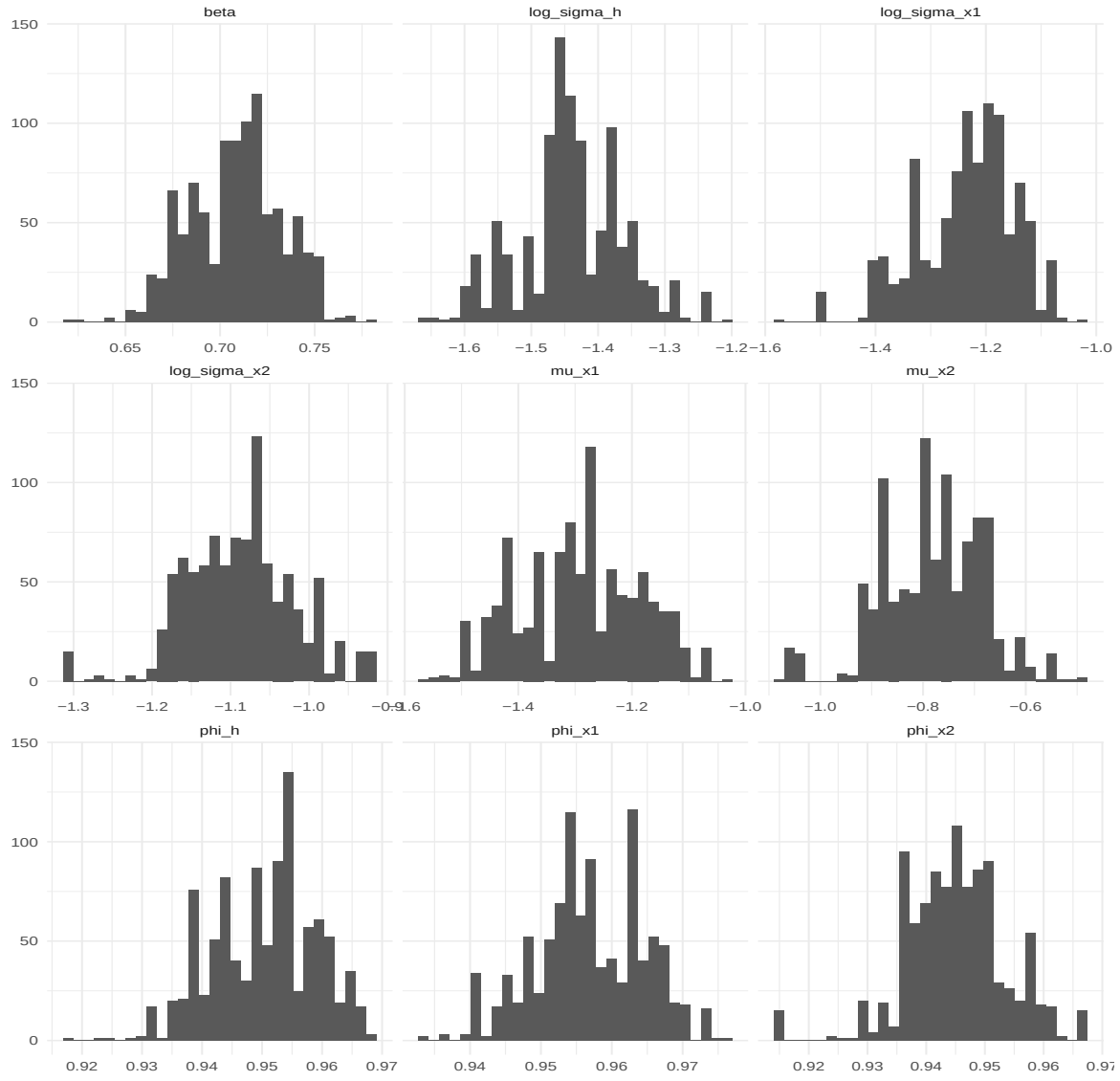


Fig. 8.6 Histogram of maximum likelihood estimates when a MLE was obtained ($n = 996$).

Chapter 9

The Nested Laplace Approximation

9.1 Motivation

Recall from chapter 3, when we apply the Laplace approximation to the integral over the latent variables $\boldsymbol{u}$, we integrate over all variables in one go. In theory there is nothing stopping us from applying the Laplace approximation in a sequential sense on subsets of $\boldsymbol{u}$. For both the univariate and multivariate basic SV models, the Laplace approximation has been shown to be accurate, see Skaug and Yu (2014) for details. Due to the convergence issues with the factor model, we introduce the nested Laplace approximation, where we integrate over subsets of the latent variables in a sequential way. We then apply the method on two models, a linear state space model and the two dimensional factor model introduced in chapter 8.

9.2 Nested Laplace approximation

Let $\boldsymbol{y}$ be a vector of observations, and let $\boldsymbol{u}$ be a random vector of latent variables. The conditional density of our observations given $\boldsymbol{u}$ is denoted by $f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u})$, and $f_{\boldsymbol{u}}(\boldsymbol{u})$ denotes the marginal density of $\boldsymbol{u}$. To make notation easier we will suppress that $\boldsymbol{u}$ and $\mathbb{H}$ is functions of $\boldsymbol{\theta}$, and write $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{\theta}), \mathbb{H} = \mathbb{H}(\boldsymbol{\theta})$. The Laplace approximation

of $\boldsymbol{u}$ is then given by

$$\mathcal{L}(\boldsymbol{\theta}) = \int \exp\{-g(\boldsymbol{u}, \boldsymbol{\theta})\}\, d\boldsymbol{u} = (2\pi)^{\dim(\boldsymbol{u})/2} \det(\mathbb{H})^{-1/2} \exp\{-g(\hat{\boldsymbol{u}}, \boldsymbol{\theta})\}, \tag{9.1}$$

where $\mathcal{Q}$ is the sample space of $\boldsymbol{u}$, $g(\boldsymbol{u}, \boldsymbol{\theta}) = -\log f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u}) f_{\boldsymbol{u}}(\boldsymbol{u})$ and

$$\begin{aligned} \hat{\boldsymbol{u}} &= \arg\min_{\boldsymbol{u}} \; g(\boldsymbol{u}, \boldsymbol{\theta}), \\ \mathbb{H} &= \frac{\partial^2}{\partial^2 \boldsymbol{u}} g(\boldsymbol{u}, \boldsymbol{\theta}) \Big|_{\boldsymbol{u}=\hat{\boldsymbol{u}}} \end{aligned} \tag{9.2}$$

We will now show how this can be applied in a sequential way. Let $(\boldsymbol{u_1}, \boldsymbol{u_2})$ be a partitioning of $\boldsymbol{u}$ and assume that $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ are independent so that $f_{\boldsymbol{u}}(\boldsymbol{u}) = f_{\boldsymbol{u_1}}(\boldsymbol{u_1}) f_{\boldsymbol{u_2}}(\boldsymbol{u_2})$. The likelihood of $\boldsymbol{\theta}$ is

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \int\int f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u_1}, \boldsymbol{u_2}) f_{\boldsymbol{u_1}}(\boldsymbol{u_1}) f_{\boldsymbol{u_2}}(\boldsymbol{u_2})\, d\boldsymbol{u_1} d\boldsymbol{u_2} \\ &= \int \exp\{(\log f_{\boldsymbol{u_2}}(\boldsymbol{u_2})\} \left(\int \exp\{\log f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u_1}, \boldsymbol{u_2}) + \log f_{\boldsymbol{u_1}}(\boldsymbol{u_1})\}\, d\boldsymbol{u_1} \right) d\boldsymbol{u_2} \\ &= \int \exp\{\log f_{\boldsymbol{u_2}}(\boldsymbol{u_2})\} \left(\int \exp\{-g(\boldsymbol{u_1}, \boldsymbol{u_2}, \boldsymbol{\theta})\}\, d\boldsymbol{u_1} \right) d\boldsymbol{u_2}, \end{aligned} \tag{9.3}$$

where $g(\cdot) = -\log f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{u_1}, \boldsymbol{u_2}) f_{\boldsymbol{u_1}}(\boldsymbol{u_1})$.
Applying Equation (9.1) to the inner integral we obtain

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \int \exp\{\log f_{\boldsymbol{u_2}}(\boldsymbol{u_2})\} (2\pi)^{\dim(\boldsymbol{u_1})/2} \det(\mathbb{H}_{\boldsymbol{u_1}})^{-1/2} \exp\{-g(\hat{\boldsymbol{u}}_{\boldsymbol{1}}(\boldsymbol{u_2}), \boldsymbol{\theta})\}\, d\boldsymbol{u_2} \\ &= \int \exp\{-h(\boldsymbol{u_2}, \hat{\boldsymbol{u}}_{\boldsymbol{1}}(\boldsymbol{u_2}), \boldsymbol{\theta})\}\, d\boldsymbol{u_2} \end{aligned} \tag{9.4}$$

where

$$h(\boldsymbol{u_2}, \hat{\boldsymbol{u}}_{\boldsymbol{1}}(\boldsymbol{u_2}), \boldsymbol{\theta}) = -\left(\log f_{\boldsymbol{u_2}}(\boldsymbol{u_2}) + \frac{\dim(\boldsymbol{u_1})}{2} \log 2\pi + \log \det(\mathbb{H}_{\boldsymbol{u_1}})^{-1/2} - g(\boldsymbol{u_2}, \hat{\boldsymbol{u}}_{\boldsymbol{1}}(\boldsymbol{u_2}), \boldsymbol{\theta}) \right).$$

and $\mathbb{H}_{\boldsymbol{u_1}}$ is the matrix of second order derivatives with respect to $\boldsymbol{u_1}$. Applying Equation (9.1) again, our likelihood is given by

$$\mathcal{L}(\boldsymbol{\theta}) = (2\pi)^{\dim(\boldsymbol{u_2})/2}\det(\mathbb{H}_{\boldsymbol{u_2}})^{-1/2}\exp\{-h(\hat{\boldsymbol{u}}_{\boldsymbol{2}}, \boldsymbol{\theta})\}, \tag{9.5}$$

and our corresponding negative log-likelihood (discarding the constant term)

$$-\,l(\boldsymbol{\theta}) = \frac{1}{2}\log\det(\mathbb{H}_{\boldsymbol{u_2}}) + h(\hat{\boldsymbol{u}}_{\boldsymbol{2}}, \boldsymbol{\theta}). \tag{9.6}$$

$-l$ is minimized w.r.t θ in the same way as in chapter 3.

9.3 When $\hat{\mathbf{u}}$ doesn't maximize g.

When $\hat{\boldsymbol{u}}$ does not maximize g in Equation (9.1), in which we write $\boldsymbol{u_0}$, we need to include a correction term in the Laplace approximation. We will here prove it.

Proposition 9.1. *When $\hat{\boldsymbol{u}}$ does not maximize g, denoted by $\boldsymbol{u_0}$, the Laplace approximation is given by*

$$\int \exp\{-g(\boldsymbol{u})\}\,d\boldsymbol{u} = \exp\left\{-g(\boldsymbol{u_0}) + \frac{1}{2}\nabla g(\boldsymbol{u_0})\mathbb{H}^{-1}\nabla' g(\boldsymbol{u_0}))\right\}(2\pi)^{dim(\boldsymbol{u})/2}det(\mathbb{H})^{-1/2} \tag{9.7}$$

Proof. By a second order Taylor expansion at $\boldsymbol{u_0}$,

$$g(\boldsymbol{u}) = g(\boldsymbol{u_0}) + \nabla g(\boldsymbol{u_0})(\boldsymbol{u}-\boldsymbol{u_0}) + \frac{1}{2}(\boldsymbol{u}-\boldsymbol{u_0})'\mathbb{H}(\boldsymbol{u}-\boldsymbol{u_0}).$$

Completing the square we get

$$\begin{aligned} g(\boldsymbol{u}) &= g(\boldsymbol{u_0}) + \nabla g(\boldsymbol{u_0})(\boldsymbol{u}-\boldsymbol{u_0}) + \frac{1}{2}(\boldsymbol{u}-\boldsymbol{u_0})'\mathbb{H}(\boldsymbol{u}-\boldsymbol{u_0}) \\ &= \frac{1}{2}(\boldsymbol{u}-\boldsymbol{u_0}-(-\mathbb{H}_{u_0}^{-1}\nabla' g(\boldsymbol{u_0})))'\mathbb{H}(\boldsymbol{u}-\boldsymbol{u_0}-(-\mathbb{H}_{u_0}^{-1}\nabla' g(\boldsymbol{u_0}))) \\ &+ g(\boldsymbol{u_0}) - \frac{1}{2}\nabla g(\boldsymbol{u_0})\mathbb{H}^{-1}\nabla' g(\boldsymbol{u_0}). \end{aligned}$$

Thus

$$\begin{aligned}\int \exp\{-g(\boldsymbol{u})\}\, d\boldsymbol{u} &= \exp\left\{ -g(\boldsymbol{u_0}) + \frac{1}{2}\nabla g(\boldsymbol{u_0})\mathbb{H}^{-1}\nabla' g(\boldsymbol{u_0})\right\} \\ &\times \int \exp\left\{ -\frac{1}{2}(\boldsymbol{u} - (\boldsymbol{u_0} - \mathbb{H}_{u_0}^{-1}\nabla' g(\boldsymbol{u_0})))'\mathbb{H}(\boldsymbol{u} - (\boldsymbol{u_0} - \mathbb{H}_{u_0}^{-1}\nabla' g(\boldsymbol{u_0})))\right\} d\boldsymbol{u} \\ &= \exp\left\{ -g(\boldsymbol{u_0}) + \frac{1}{2}\nabla g(\boldsymbol{u_0})\mathbb{H}^{-1}\nabla' g(\boldsymbol{u_0})\right\}(2\pi)^{\dim(\boldsymbol{u})/2}\det(\mathbb{H})^{-1/2},\end{aligned}$$

where we in the last step used the fact that the integrand is the kernel of a multivariate normal density with expectation $\boldsymbol{\mu} = \boldsymbol{u_0} - \mathbb{H}^{-1}\nabla' g(\boldsymbol{u_0})$ and covariance matrix $\boldsymbol{\Sigma} = \mathbb{H}^{-1}$. □

9.4 Implementation

The Nested Laplace approximation is implemented as a **TMB** program in C++. The minimum $\hat{\boldsymbol{u}}_{\boldsymbol{1}}$ is obtained by Newtons method. We tested with five and three Newton steps, both giving the same result. Since it is implemented in **TMB**, we make use of the AD library **CppAD** to evaluate both gradient and Hessian of the inner Laplace Approximation (see chapter 6 for more details). This is then used as input to the outer Laplace approximation which is automatically done by **TMB** (see appendix B for an example of the implementation of the linear state space model discussed below). We will now apply the methodology on two state space models.

Example 9.2. Consider the linear state space model

$$\begin{aligned} y_{1t} &= h_t + x_{1t} + \epsilon_{1t} \\ y_{2t} &= h_t + x_{2t} + \epsilon_{2t} \end{aligned} \tag{9.8}$$

where h_t and x_{it} are independent, centered AR(1) processes and $\epsilon_{it} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_i^2)$ for $i = 1, 2$. We assume that x_{11}, x_{21} and h_1 are stationary distributed. The parameters

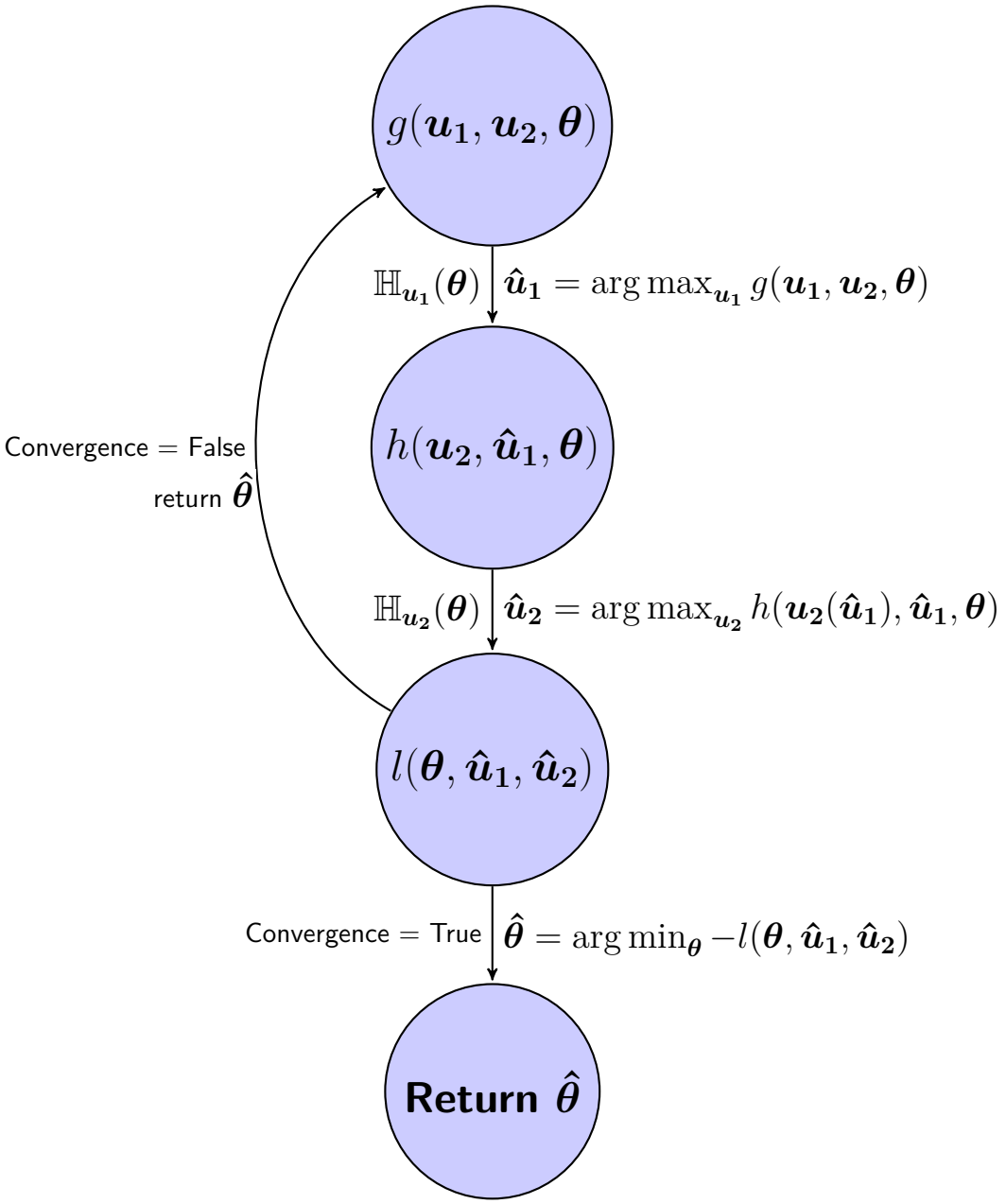


Fig. 9.1 Optimization routine for Nested Laplace. The arguments next to the arrows are the input to the next node. For example, the input to the function h in the second node is the Hessian of g at $\hat{\boldsymbol{u}}_1$ and the solution $\hat{\boldsymbol{u}}_1$. The estimate $\hat{\boldsymbol{\theta}}$ is returned when the convergence criteria described in Fournier et al. (2011) is meet.

of interest is denoted by $\boldsymbol{\theta} = (\phi_h, \phi_{x_1}, \phi_{x_2}, \sigma_h, \sigma_{x_1}, \sigma_{x_2}, \sigma_1, \sigma_2)$. The joint log-likelihood

function is given by:

$$
\begin{aligned}
l(\boldsymbol{y}, \boldsymbol{h}, \boldsymbol{x}|\boldsymbol{\theta}) &= \sum_{j=2}^{n} \left(\sum_{i=1}^{2} \left[\log f_{y_{ij}}(y_{ij}|h_j, x_{ij}) + \log f_{x_{ij}}(x_{ij}|x_{ij-1}) \right] + \log f_{h_j}(h_j|h_{j-1}) \right) \\
&+ \log f_{y_{11}}(y_{11}|h_1, x_{11}, x_{21}) + \log f_{y_{21}}(y_{21}|h_1, x_{11}, x_{21}) \\
&+ \log f_{x_{11}}(x_{11}) + \log f_{x_{21}}(x_{21}) + \log f_{h_1} f(h_1) \\
&\propto \sum_{j=2}^{n} \left(\sum_{i=1}^{2} \left[\frac{1}{2\sigma_i^2}(y_{ij} - (h_i + x_{ij}))^2 - \log \sigma_i - \frac{1}{2\sigma_{x_i}^2}(x_{ij} - \phi_{x_i} x_{ij-1})^2 - \log \sigma_{x_i} \right] \right. \\
&\left. - \log \sigma_h - \frac{1}{2\sigma_h^2}(h_j - \phi_h h_{j-1})^2 \right) - \frac{1}{2\sigma_1^2}(y_{11} - (h_1 + x_{11}))^2 - \frac{1}{2\sigma_2^2}(y_{21} - (h_1 + x_{21}))^2 \\
&- \frac{1 - \phi_{x_1}^2}{2\sigma_{x_1}^2} x_{11}^2 - \frac{1 - \phi_{x_2}^2}{2\sigma_{x_2}^2} x_{21}^2 - \frac{1 - \phi_h^2}{2\sigma_h^2} h_1^2.
\end{aligned}
$$

For computational and visual reasons we only simulate 100 observations and investigate the results and the Hessian matrix returned from **TMB**. The parameters are set to $\boldsymbol{\theta} = (0.9, 0.9, 0.9, 0.37, 0.37, 0.37, 0.37, 0.37)$. We evaluate four methods:

1. Standard Laplace approximation implemented in **TMB**.

2. Nested Laplace approximation where the inner Laplace is done around the optimum $\hat{\boldsymbol{u}}_{\boldsymbol{1}} = \arg\max_{\boldsymbol{u_1}} g$ (see Equation (9.4)).

3. Nested Laplace approximation where the inner Laplace is done around $\mathbf{0}$ (see Equation (9.7)).

4. Laplace approximation around $\mathbf{0}$ for all latent variables (see equation 9.7).

In this example we chose to integrate w.r.t to $\boldsymbol{x} = (x_{11}, \ldots, x_{1n}, x_{21}, \ldots x_{2n})$ first, and then we apply the Laplace approximation provided by **TMB** to integrate out $\boldsymbol{h} = (h_1, \ldots, h_n)$. The Hessian reported to R is therefore only a function of $\boldsymbol{h}$ when we integrate $\boldsymbol{x}$ inside the C++ template. Integrating w.r.t $\boldsymbol{h}$ first gave the same result and is therefore omitted. All four methods gave exactly the same result, as expected when the model is jointly Gaussian and the Laplace approximation is exact. Parameter estimates are reported in table 9.1.

	Estimate	SE
ϕ_h	0.938	0.038
σ_h	0.361	0.058
ϕ_{x_1}	0.742	0.125
ϕ_{x_2}	0.979	0.049
σ_{x_1}	0.426	0.095
σ_{x_2}	0.091	0.127
σ_1	0.282	0.100
σ_2	0.514	0.054

Table 9.1 Parameter estimates for the four methods described above, on $n = 100$ observations.

More interesting is the structure of the Hessian of the Laplace approximation in the nested and the exact case, as seen in figure 9.2. In the case of the nested Laplace, the solution of

$$\hat{\boldsymbol{u}}_{\boldsymbol{1}} = \arg\max_{\boldsymbol{u_1}} g(\boldsymbol{y}, \boldsymbol{u_1}, \boldsymbol{u_2})$$

in the inner Laplace approximation, will be a function of $\boldsymbol{u_2}$. This will be input to the outer Laplace approximation (Equation (9.4)). The implication of this is that we are no longer guaranteed the Markovian structure of the likelihood, which again implies that we are not guaranteed a sparse structure of $\mathbb{H}$. In figure 9.3, a close up of the first twenty rows and columns are shown of the Hessian of the outer Laplace approximation where the tridiagonal is set to zero. This is done to get a greater picture of what is happening to the values as we move away from the diagonal. We can observe that the inner Laplace approximation has had a smoothing effect of the Hessian of the outer approximation. As we go away from the diagonal the value goes toward zero, but never actually zero. The matrix is therefore completely dense and computations are therefore slow.

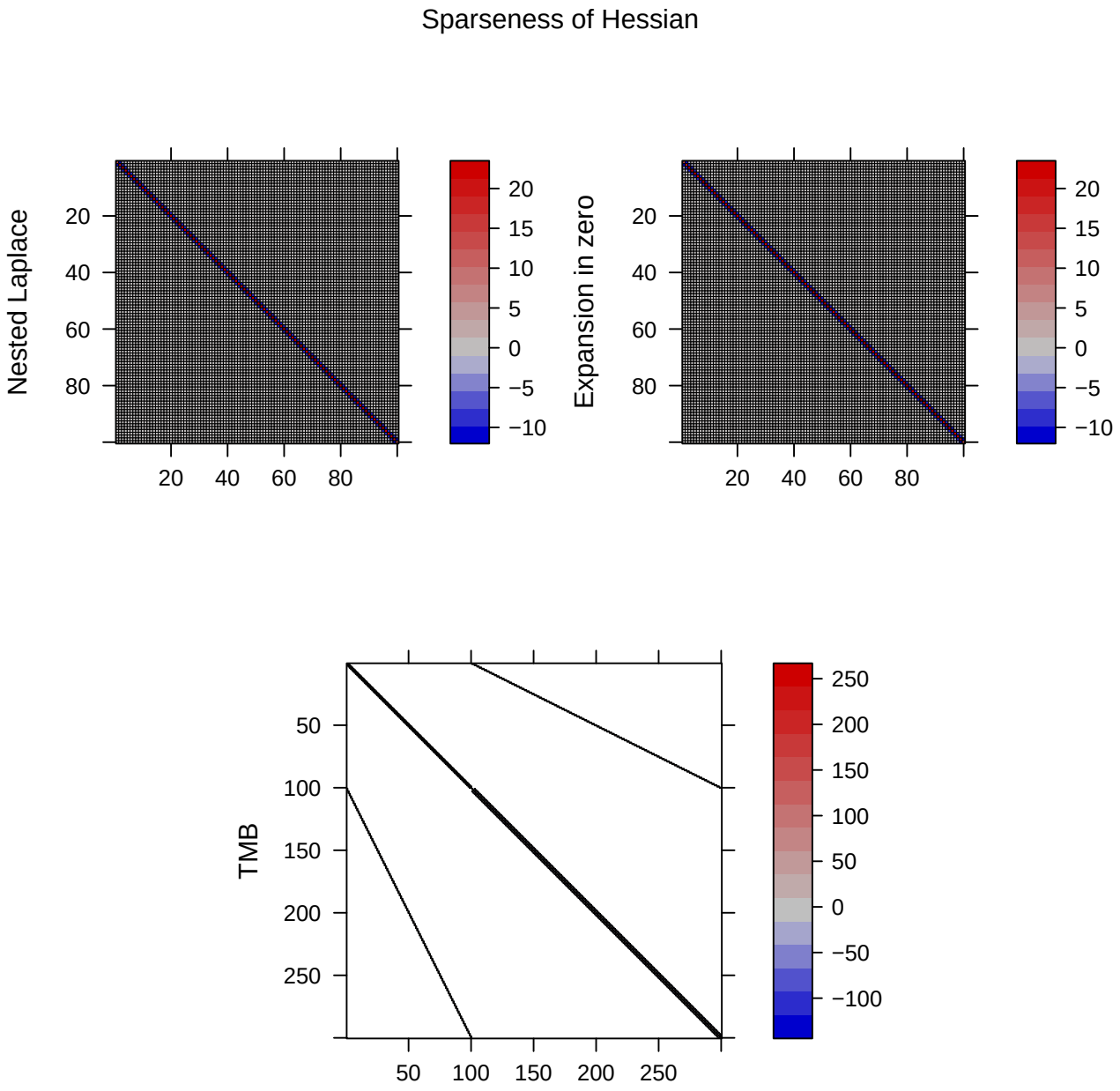


Fig. 9.2 Top left: Hessian matrix in Equation (9.6) when we optimize inner Laplace by Newton. Top right: Hessian matrix in Equation (9.6) when the inner Laplace is done by Taylor expansion in zero. Bottom: Standard **TMB** with full full Laplace.

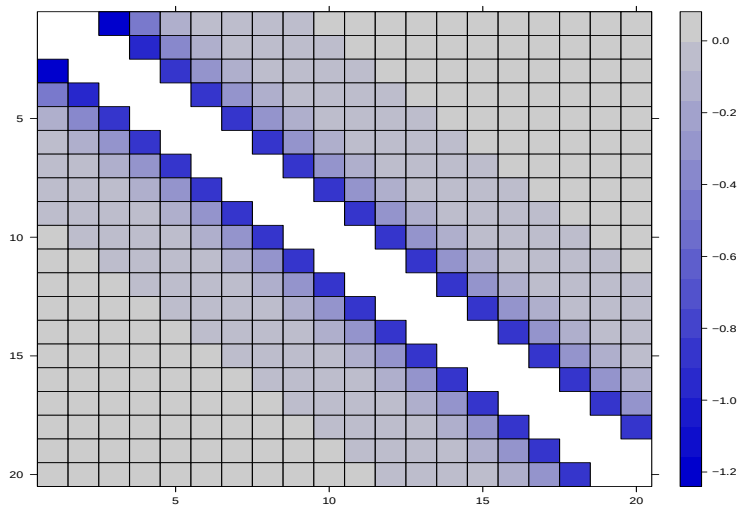


Fig. 9.3 Close up of the first twenty rows and columns of the Hessian of the outer Laplace approximation. The tridiagonal of the matrix is set to zero to get a better image of the values away from the diagonal

Example 9.3. Recall the two dimensional factor model introduced in chapter 8(see section 8.1 for details):

$$y_{1t} = \beta_1 e^{\frac{1}{2}h_t}\zeta_t + e^{\frac{1}{2}x_{1t}}\epsilon_{1t} \tag{9.9}$$

$$y_{2t} = \beta_2 e^{\frac{1}{2}h_t}\zeta_t + e^{\frac{1}{2}x_{2t}}\epsilon_{2t}, \tag{9.10}$$

Let Σ_i denote the conditional covariance matrix of $\boldsymbol{y_i}|h_t, \boldsymbol{x_t}$, where $\boldsymbol{y_t} = (y_{1t}, y_{2t})'$ and $\boldsymbol{x_t} = (x_{1t}, x_{2t})'$. The joint log-likelihood is given by

$$
\begin{aligned}
l(\boldsymbol{y}, \boldsymbol{h}, \boldsymbol{x}|\boldsymbol{\theta}) &= \sum_{i=1}^{n} \log f_{\boldsymbol{y_i}}(\boldsymbol{y_i}|h_i, \boldsymbol{x_i}) \\
&+ \sum_{j=2}^{n} \Big[\log f_{h_j}(h_j|h_{j-1}) + \log f_{x_{1j}}(x_{1j}|x_{1j-1}) + \log f_{x_{2j}}(x_{2j}|x_{2j-1}) \Big] \\
&+ \log f_{h_1}(h_1) + \log f_{x_{11}}(x_{11}) + \log f_{x_{21}}(x_{21}) \\
&\propto \sum_{i=1}^{n} -\frac{1}{2} \log \det(\Sigma_i) - \frac{1}{2} \boldsymbol{y_i}' \Sigma_i^{-1} \boldsymbol{y_i} \\
&- \sum_{j=2}^{n} \Big[\log \sigma_h + \frac{1}{2\sigma_h^2} (h_j - \phi_h h_{j-1})^2 + \log \sigma_{x_1} + \frac{1}{2\sigma_{x_1}^2} (x_{1j} - (\mu_{x_1} + \phi_{x_1}(x_{1j-1} - \mu_{x_1}))^2 \\
&+ \log \sigma_{x_2} + \frac{1}{2\sigma_{x_2}^2} (x_{2j} - (\mu_{x_2} + \phi_{x_2}(x_{2j-1} - \mu_{x_2})) \Big] \\
&- \frac{1}{2} \log \frac{\sigma_h^2}{1 - \phi_h^2} - \frac{1}{2\sigma_h^2} h_1^2 - \frac{1}{2} \log \frac{\sigma_{x_1}^2}{1 - \phi_{x_1}^2} - \frac{1}{2\sigma_{x_1}^2} (x_{11} - \mu_{x_1})^2 \\
&- \frac{1}{2} \log \frac{\sigma_{x_2}^2}{1 - \phi_{x_2}^2} - \frac{1}{2\sigma_{x_2}^2} (x_{21} - \mu_{x_2})^2.
\end{aligned}
$$

Due to memory limitations, we were not able to simulate more that 50 observations. The parameter values in the simulation was $\boldsymbol{\theta} = (\beta_1, \beta_2, \phi_h, \sigma_h, \mu_{x_1}, \phi_{x_1}, \sigma_{x_1}, \mu_{x_2}, \phi_{x_2}, \sigma_{x_2}) = (0.7, 1.3, 0.9, 0.25, -1.3, 0.95, 0.37, -0.8, 0.94, 0.45)$. The standard Laplace approximation did not converge and is therefore omitted. The Nested Laplace approximation converged, but was very slow, taking 5465 seconds to converge. As with the linear state space model, the Hessian of the Nested Laplace was completely dense, as seen in figure 9.4. Parameter estimates and standard errors are reported in table 9.2. As expected, the estimates are not very precise and the standard errors are big. But it is worth noticing that the true values are inside two times the standard deviation. It would be interesting to see how the method would perform on a bigger dataset.

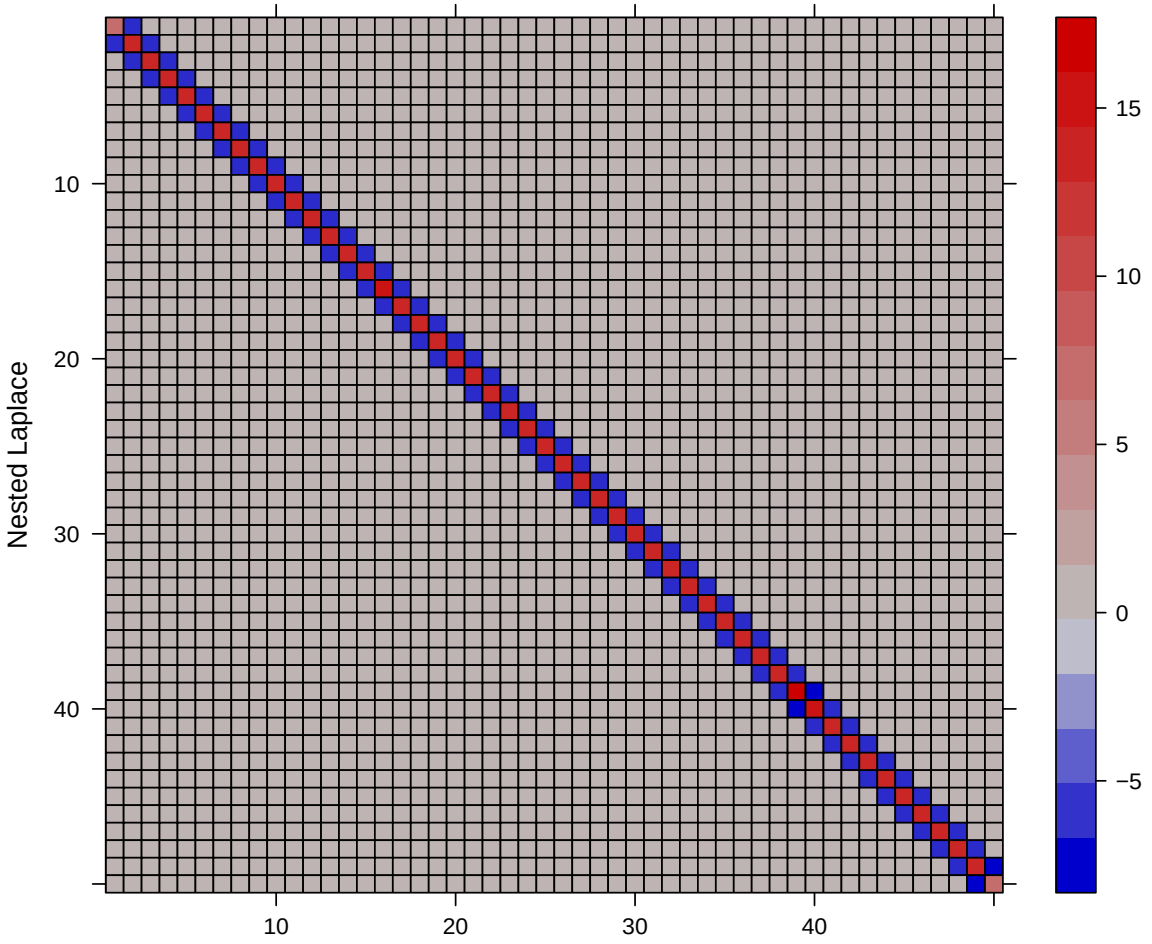


Fig. 9.4 Hessian matrix in factor model when we optimize inner Laplace by Newtons method with respect to x. Only 50 observations. Time to optimize: 5481.549 seconds. The method did not converge in the regular Laplace and therefore no plot is available

	Estimate	SE
β_1	0.667	0.180
β_2	0.956	0.215
μ_{x_1}	−1.850	1.042
μ_{x_2}	−0.116	0.566
σ_{x_1}	0.190	0.603
σ_{x_2}	0.289	0.322
σ_h	0.656	0.337
ϕ_h	0.535	0.321
ϕ_{x_1}	0.850	0.356
ϕ_{x_2}	0.874	0.221

Table 9.2 Parameter estimates for the factor model using the Nested Laplace approximation, on $n = 50$ observations.

Chapter 10

Empirical Analysis

In this chapter, we analyze exchange rates with respect to EUR. Data was obtained from the authors of Kastner et al. (2017) and ranges from April 1, 2005 to August 6, 2015. It contains 26 daily exchange rates on 2650 days. For simplicity we will focus on the two dimensional factor model and compare three different estimation methods: maximum likelihood (MLE) by the use of the Laplace approximation, Hamiltonian Monte Carlo (HMC) and the Deep Interweaving MCMC (DIMCMC) strategy proposed in Kastner et al. (2017). For maximum likelihood estimation we use the **TMB** package Kristensen et al. (2016). **RStan** (Stan Development Team (2016)) is used for HMC sampling and **factorstochvol** for DIMCMC sampling. We investigate two different scenarios:

1. When the likelihood converge.

2. When the likelihood does not converge.

We also investigate the effect of changing the parameters in the $\mathcal{B}(a_0, b_0)$ prior of ϕ due to the fact that the standard $\mathcal{B}(20, 1.5)$ is quite restrictive, and we want to inspect the consequences of using a prior with bigger variance. As we will see, this does effect the estimates of the latent processes.

We run the DIMCMC sampler for 110 000 iterations, discarding the first 10 000 as burn-in. For HMC we run the sampler for 3000 iterations and discard the first 1000 as burn-in (called warm up in **Stan**).

There are 325 different pair combinations in the datasets. The likelihood converged for 105 datasets. There was no observable pattern to when we did not obtain convergence, as it happened with data with high correlation as well as with data with low correlation. When applying the restriction $\beta_1 = \beta_2$, convergence was obtained for 204 datasets. The following examples are meant to show how the different estimation methods performed when the likelihood converged and when it did not. The economic interpretation is not important here.

10.1 Likelihood converge

We here present results from estimating the two dimensional factor stochastic volatility model on exchange rates from Australian dollar (AUD) and Canadian dollar (CAD) with respect to EUR. The data is displayed in Figure 10.1, and the empirical ACF for both the log returns and squared log returns is displayed in figure 10.2 an 10.3. The log returns are close to uncorrelated, while the square log returns are clearly correlated, a typical sign of volatility clustering. The correlation between AUD and CAD is 0.54. In all tables x_1 will refer to AUD and x_2 to CAD.

We can observe from Table 10.1 that all three methods give similar results, with exception of μ_{x_1}, where both HMC and DIMCMC estimates it more negative that MLE. This can also be observed in Figure 10.4, where the EB estimate of the idiosyncratic log-variance of AUD is bigger than for HMC and DIMCMC. It is not quite clear why HMC starts out way below the other methods in figure 10.4.

For ML, we tried different starting values to ensure that our estimates was a global minimum of the negative log likelihood. When starting μ_{x_1} in -5 the likelihood converged to another solution, where σ_{x_1} was estimated to 0.002, leading to x_{1t} being

estimated to a straight line equal to the mean. The negative log likelihood was bigger in this optimum and results is therefore omitted.

Investigating autocorrelations of the draws via the empirical autocorrelation function shows that some of the parameters have a extremely slow decaying ACF. In Figure 10.6 we plot the ACF of the variable with slowest decaying autocorrelation (μ_{x_1} for both DIMCMC and HMC). Changing the priors for ϕ did not change the results substantially and is therefore omitted. Efficient sample size and efficient sample size per unit time is reported in table 10.2. ESS varies a great deal across the different parameters, where for DIMCMC the ratio of the biggest and the smallest ESS is 614 and 3 for HMC. Also notice that MLE is more than ten times faster than DIMCMC in CPU time, which again is more than ten times faster than HMC.

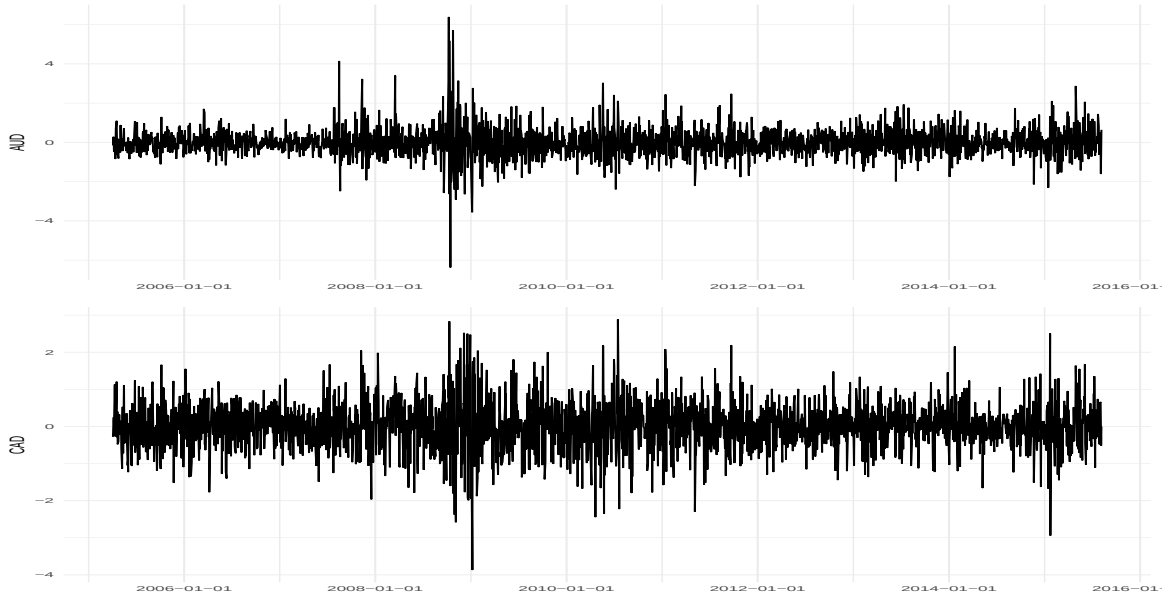


Fig. 10.1 Demeaned log returns for AUD and CAD with respect to EUR.

	MLE		DIMCMC		HMC	
	Estimate	SE	Estimate	SE	Estimate	SE
β_1	0.48	0.06	0.55	0.08	0.55	0.06
β_2	0.36	0.04	0.34	0.05	0.34	0.04
ϕ_h	0.99	0.004	0.99	0.004	0.99	0.004
σ_h	0.10	0.02	0.09	0.02	0.09	0.016
ϕ_{x_1}	0.98	0.008	0.95	0.03	0.96	0.028
ϕ_{x_2}	0.99	0.005	0.99	0.005	0.95	0.013
μ_{x_1}	-2.73	0.41	-5.14	1.09	-5.60	1.99
μ_{x_2}	-1.68	0.16	-1.57	0.18	-1.52	0.09
σ_{x_1}	0.23	0.04	0.75	0.21	0.76	0.27
σ_{x_2}	0.08	0.02	0.08	0.02	0.19	0.03
CPU(s)	38		535		7334	

Table 10.1 The two dimensional factor model fitted to demeaned daily returns for AUD (Australia dollar) and CAD (Canada dollar) with respect to EUR. The data ranges from 1.April 2005 to 6.August 2015.

	DIMCMC		HMC	
	ESS	ESS/T	ESS	ESS/T
β_1	15 352	28.7	965	0.13
β_2	28 586	53.43	954	0.13
ϕ_h	1 976	3.69	670	0.09
σ_h	920	1.72	568	0.08
ϕ_{x_1}	123	0.23	268	0.04
ϕ_{x_2}	1 310	2.45	828	0.11
μ_{x_1}	25	0.05	321	0.04
μ_{x_2}	18 199	34.02	931	0.13
σ_{x_1}	81	0.15	469	0.06
σ_{x_2}	941	1.76	733	0.10

Table 10.2 ESS and ESS per unit time for DIMCMC and HMC.

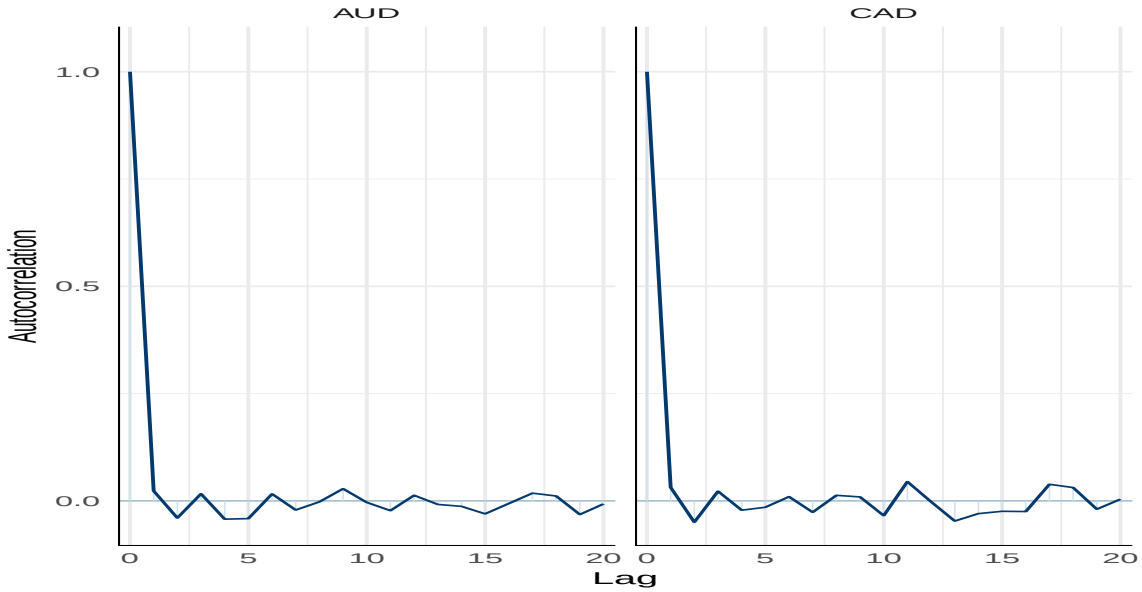


Fig. 10.2 Empirical autocorrelation function for log returns of AUD and CAD.

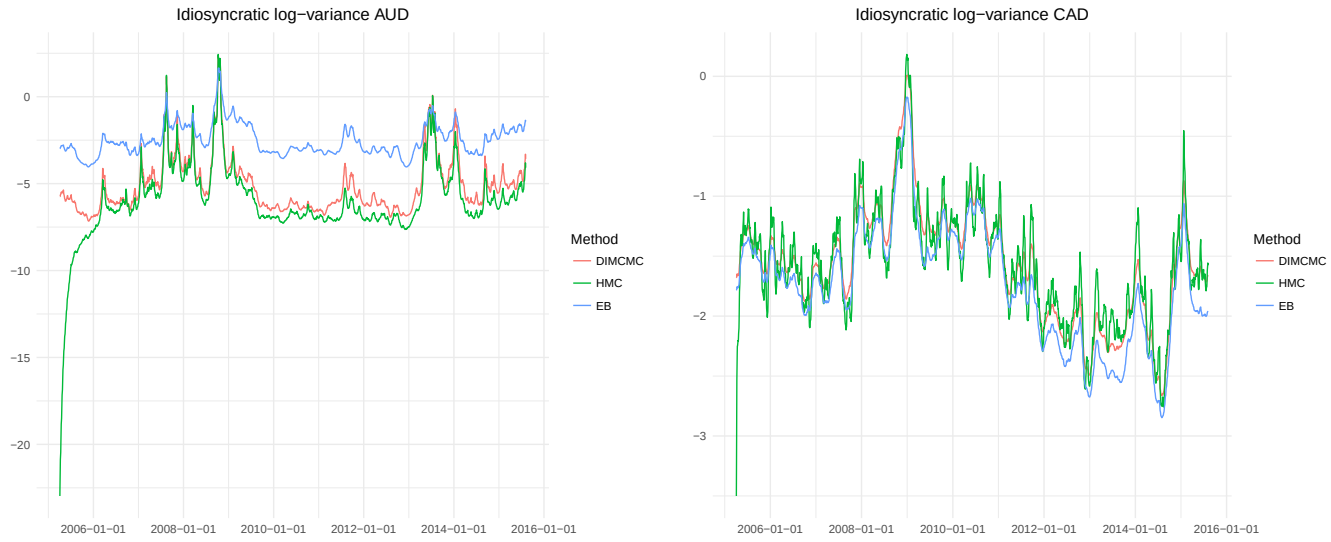


Fig. 10.4 Estimated idiosyncratic log-variance for AUD and CAD.

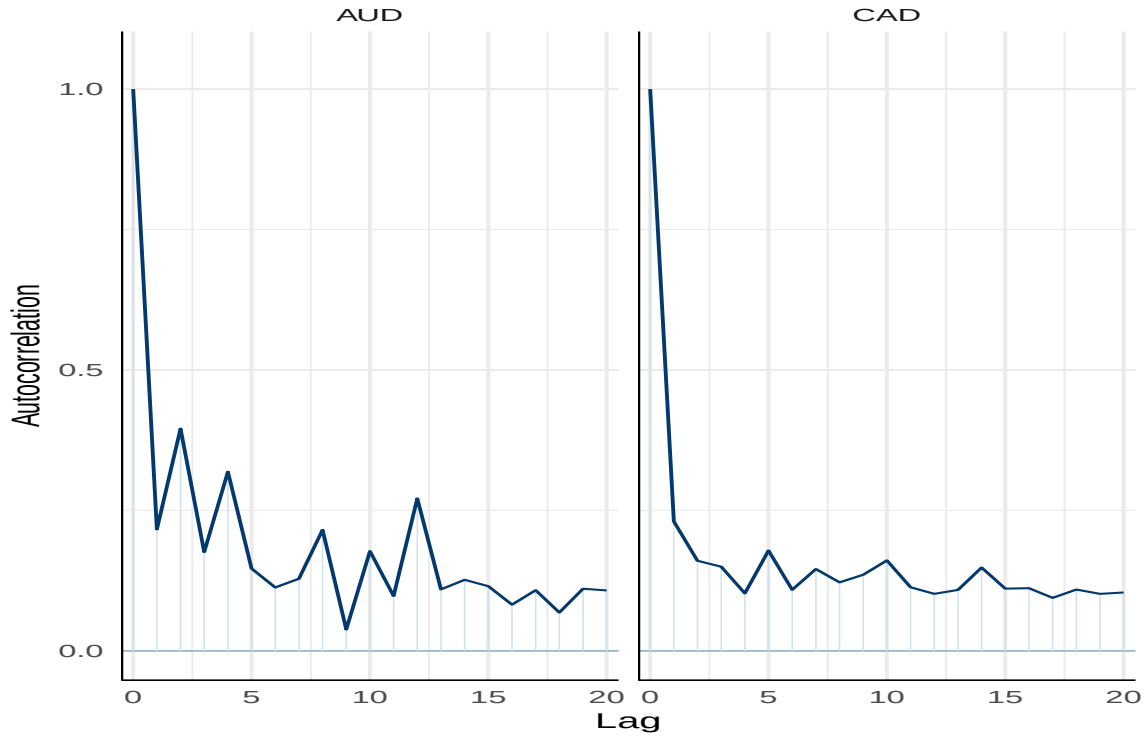


Fig. 10.3 Empirical autocorrelation function for the squared log returns of AUD and CAD.

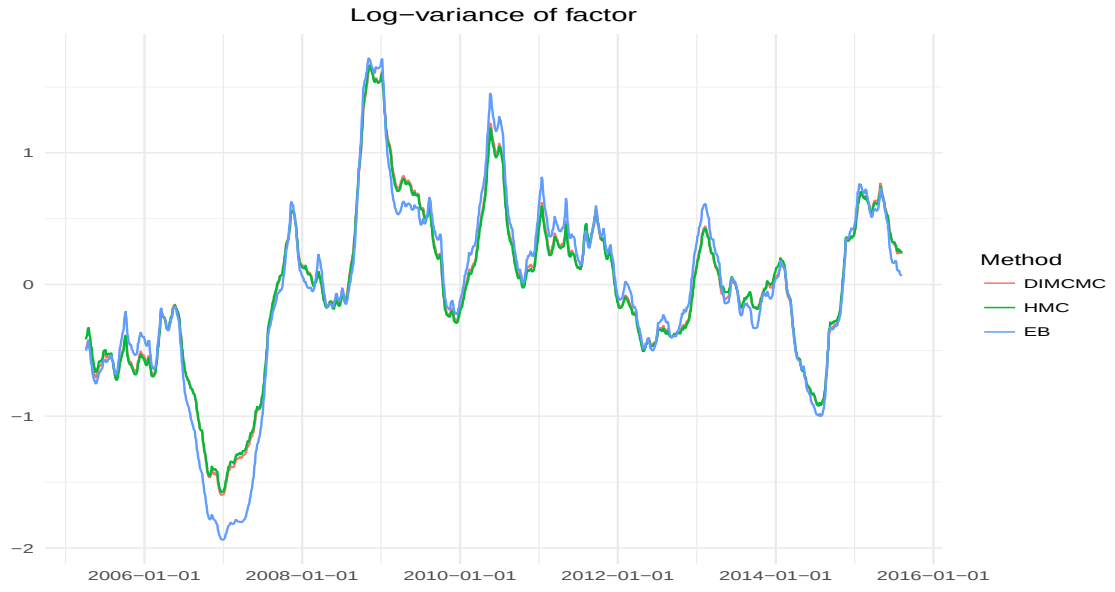


Fig. 10.5 Estimated factor log-variance for AUD and CAD. All methods give similar results.

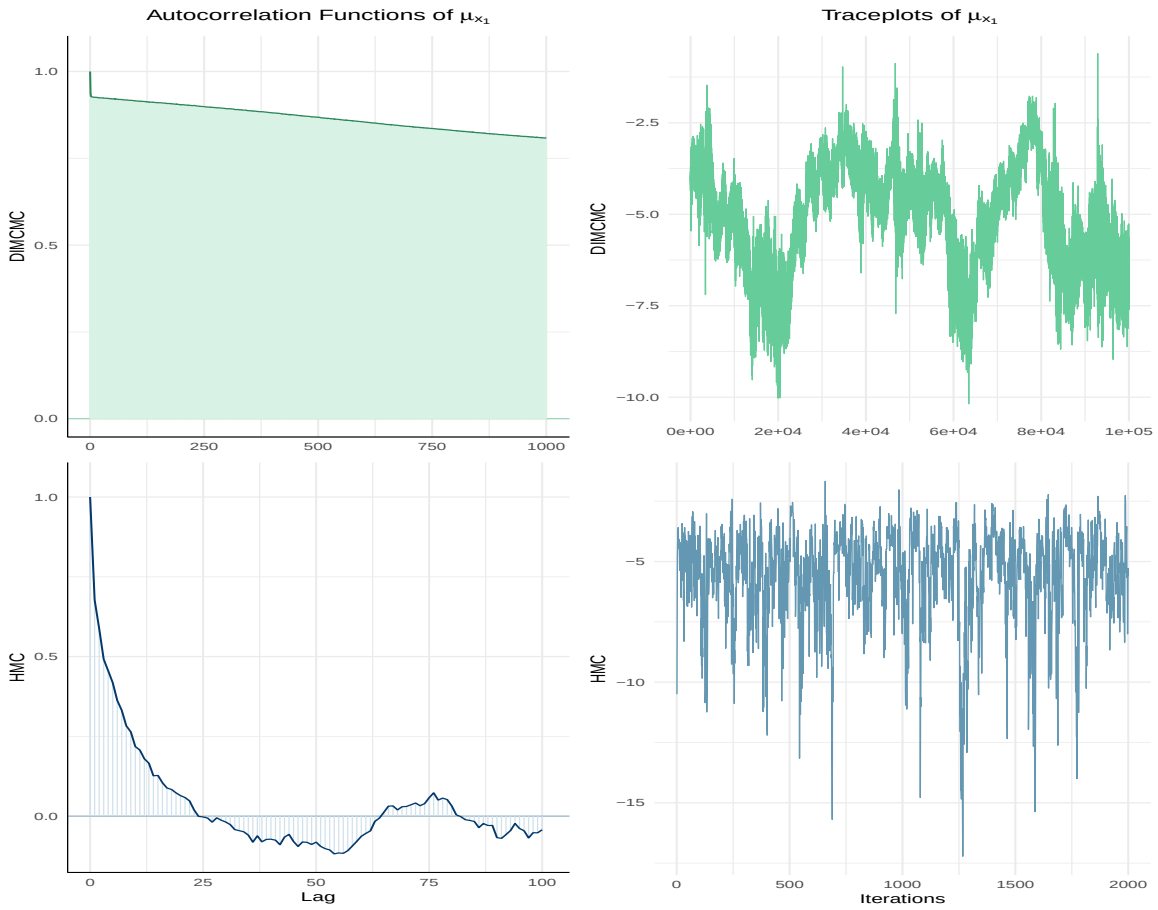


Fig. 10.6 Right hand side: Trace plots of all 100 000 draws using DIMCMC (top) and all 2000 draws using HMC (bottom) for μ_{x_1}. Left hand side: Empirical autocorrelation function functions of all draws using DIMCMC (top) and HMC (bottom).

10.2 Likelihood does not converge

We here estimate our model to log returns of Croatian kuna (HRK) and Philippines peso (PHP). The data is displayed in figure 10.7, and the empirical ACF of log returns and squared log returns is displayed in figure 10.8 an 10.9. As with AUD and CAD we see clear signs of volatility clustering. The correlation between HRK and PHP is 0.02, meaning that the data are almost uncorrelated. In all tables x_1 will refer to HRK and x_2 to PHP.

We fit the model to the data with two different priors on ϕ_i, $i = 1, 2, 3$: The standard $\mathcal{B}(20, 1.5)$ distribution, used frequently in the literature and a $\mathcal{B}(10, 3)$ distribution which is less restrictive. The result using a $\mathcal{B}(20, 1.5)$ prior is reported in table 10.3.

The estimated log variance is shown in the left hand side of figure 10.10, 10.11 and 10.12. We can see that EB and DIMCMC gives similar results, both idiosyncratic log-variances are estimated very similarly, while the log-variance of the factor has very little structure and looks very noisy, which reflect the fact that ϕ_h is estimated to 0.1 and 0.27. HMC on the other hand, estimate the log-variance of the factor with a lot of structure, reflecting that ϕ_h is estimated to 0.98. HMC estimates the log-variance for PHP similar to EB and DIMCMC, while the log-variance of HRK has little structure, since ϕ_{x_1} is estimated to 0.517. Efficient sample size is reported is table 10.5.

Table 10.4 reports the parameter estimates using a $\mathcal{B}(10, 3)$ prior on ϕ. As for DIMCMC this prior did not affect the results in a big manner, with exception to ϕ_h, which is estimated to 0.273. This may reflect the fact that the correlation is very low between HRK and PHP and giving a prior with smaller expectation and bigger variance, we are able to estimate the low correlation. For HMC we can observe two significant differences: (1) ϕ_h is estimated to 0.556, while ϕ_{x_1} is estimated to 0.974 and, (2) σ_h is estimated to 1.193, while σ_{x_1} is estimated to 0.391. So, in comparison to the $\mathcal{B}(20, 1.5)$ prior there has been a flip between the parameter estimates in the latent AR(1) process of the factor and the latent AR(1) process of the idiosyncratic variance HRK. This can also be observed in the right hand side of figure 10.11 and 10.12. Efficient sample size and efficient sample size per unit time is reported is table 10.6.

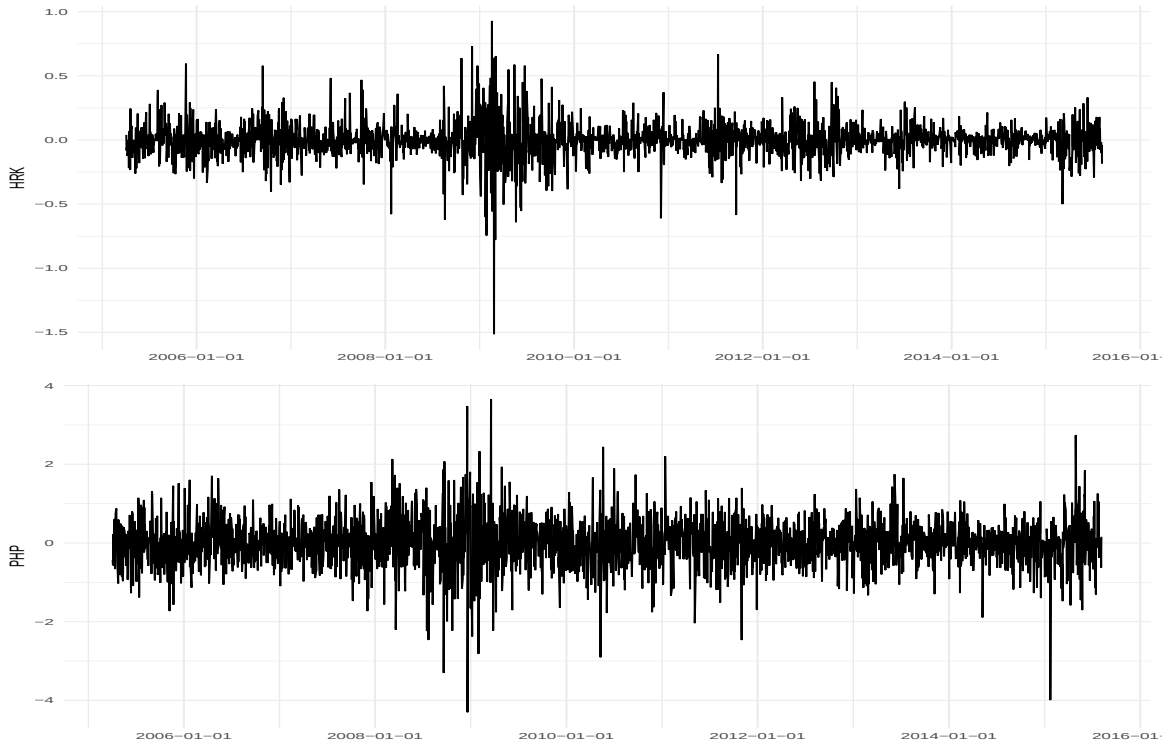


Fig. 10.7 Demeaned log returns for HRK and PHP with respect to EUR.

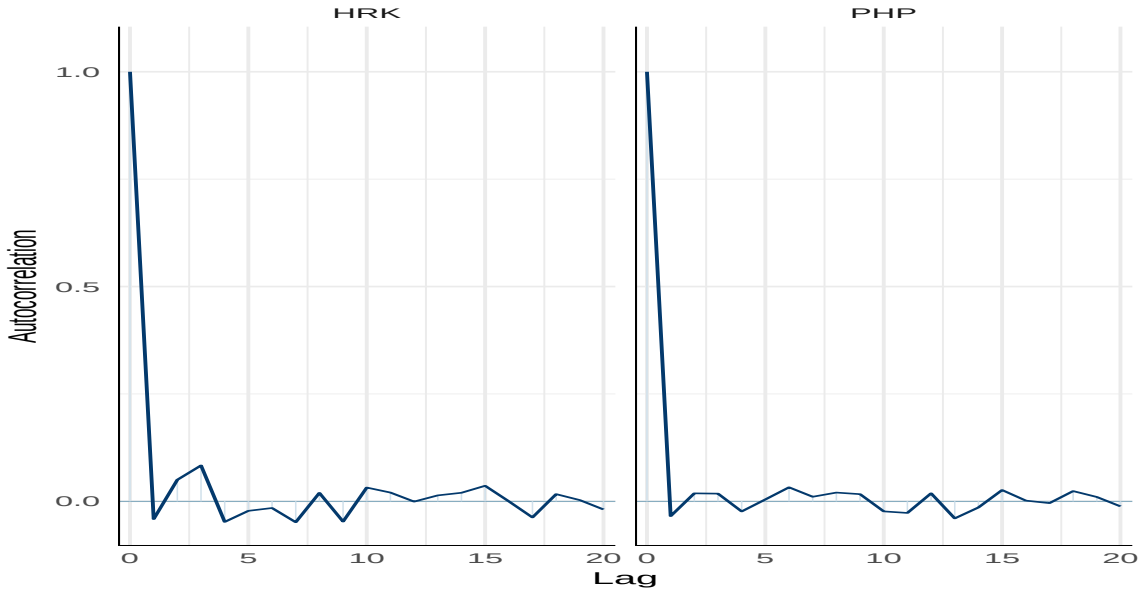


Fig. 10.8 Empirical autocorrelation function for log returns of HRK and PHP.

	MLE		DIMCMC		HMC	
	Estimate	SE	Estimate	SE	Estimate	SE
β_1	-0.003	-	-0.0007	0.0015	0.063	0.015
β_2	0.135	-	0.255	0.094	0.005	0.008
ϕ_h	0.100	-	0.544	0.145	0.982	0.006
σ_h	1.751	-	0.973	0.303	0.305	0.045
ϕ_{x_1}	0.929	-	0.913	0.991	0.517	0.088
ϕ_{x_2}	0.971	-	0.991	0.0005	0.937	0.016
μ_{x_1}	-4.78	-	-4.75	0.116	-6.31	0.287
μ_{x_2}	-1.287	-	-1.83	0.518	-1.171	0.083
σ_{x_1}	0.431	-	0.483	0.052	1.374	0.166
σ_{x_2}	0.167	-	0.109	0.039	0.220	0.031
CPU(s)	125		557		5743	

Table 10.3 The two dimensional factor model fitted to demeaned daily returns for HRK (Croatia kuna) and PHP (Philippines peso) with respect to EUR. The data ranges from 1.April 2005 to 6.August 2015. For MCMC and HMC $(\phi + 1)/2$ had a $\mathcal{B}(20, 1.5)$ distributed prior. Note that standard errors are not reported for MLE, since it did not converge.

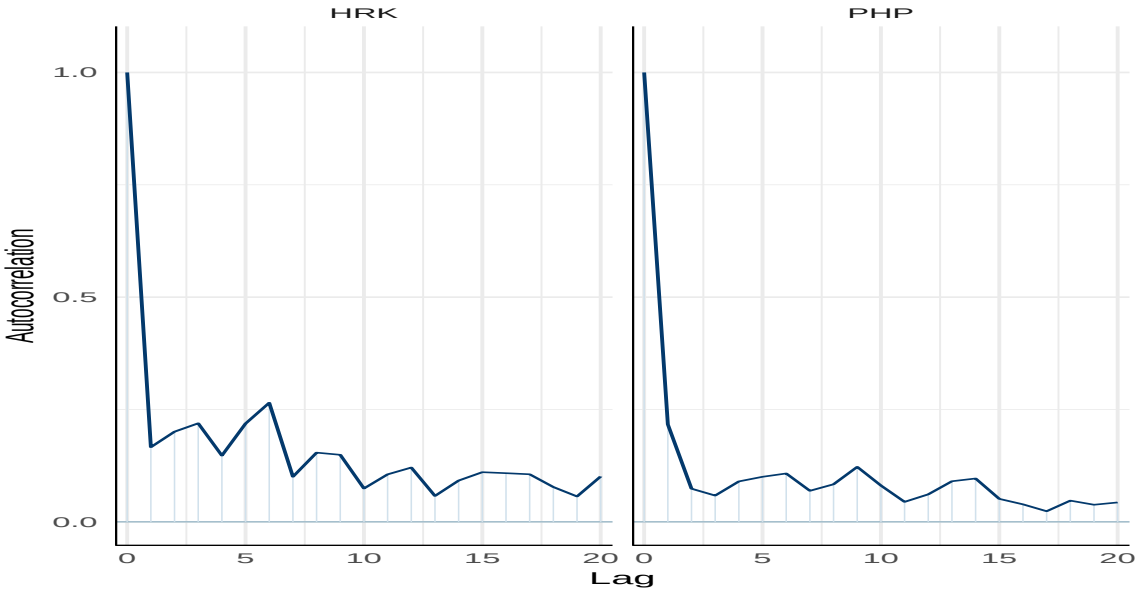


Fig. 10.9 Empirical autocorrelation function for the squared log returns of HRK and PHP.

	MLE		DIMCMC		HMC	
	Estimate	SE	Estimate	SE	Estimate	SE
β_1	-0.003	-	-0.0008	0.0016	0.052	0.006
β_2	0.135	-	0.326	0.070	0.001	0.009
ϕ_h	0.100	-	0.273	0.152	0.556	0.004
σ_h	1.751	-	0.867	0.228	1.193	0.151
ϕ_{x_1}	0.929	-	0.905	0.019	0.974	0.009
ϕ_{x_2}	0.971	-	0.987	0.006	0.929	0.018
μ_{x_1}	-4.78	-	-4.87	0.1122	-5.99	0.430
μ_{x_2}	-1.287	-	-2.238	0.6065	-1.187	0.078
σ_{x_1}	0.431	-	0.501	0548	0.391	0.058
σ_{x_2}	0.167	-	0.171	0.062	0.235	0.035
CPU(s)	125		557		4845	

Table 10.4 The two dimensional factor model fitted to demeaned daily returns for HRK (Croatia kuna) and PHP (Philippines peso) with respect to EUR. The data ranges from 1.April 2005 to 6.August 2015. For MCMC and HMC $(\phi + 1)/2$ had a $\mathcal{B}(10, 3)$ distributed prior. Note that standard errors are not reported for MLE, since it did not converge.

	DIMCMC		HMC	
	ESS	ESS/T	ESS	ESS/T
β_1	7 054	12.67	990	0.17
β_2	19	0.03	2 000	0.35
ϕ_h	153	0.27	390	0.07
σ_h	65	0.12	410	0.07
ϕ_{x_1}	1 518	2.73	258	0.04
ϕ_{x_2}	621	1.12	518	0.09
μ_{x_1}	23 689	42.53	282	0.05
μ_{x_2}	46	0.18	2000	0.35
σ_{x_1}	1 161	2.08	386	0.07
σ_{x_2}	58	0.11	460	0.08

Table 10.5 ESS and ESS per unit time for DIMCMC and HMC when $(\phi + 1)/2$ had a $\mathcal{B}(20, 1.5)$ distributed prior.

	DIMCMC		HMC	
	ESS	ESS/T	ESS	ESS/T
β_1	12 610	22.64	59	0.01
β_2	41	0.07	2000	0.41
ϕ_h	383	0.69	67	0.01
σ_h	123	0.22	68	0.01
ϕ_{x_1}	1 390	2.50	59	0.01
ϕ_{x_2}	459	0.82	499	0.10
μ_{x_1}	272	0.49	317	0.07
μ_{x_2}	91	0.16	1278	0.26
σ_{x_1}	1 082	1.94	117	0.02
σ_{x_2}	120	0.22	509	0.11

Table 10.6 ESS and ESS per unit time for DIMCMC and HMC when $(\phi + 1)/2$ had a $\mathcal{B}(10, 3)$ distributed prior.

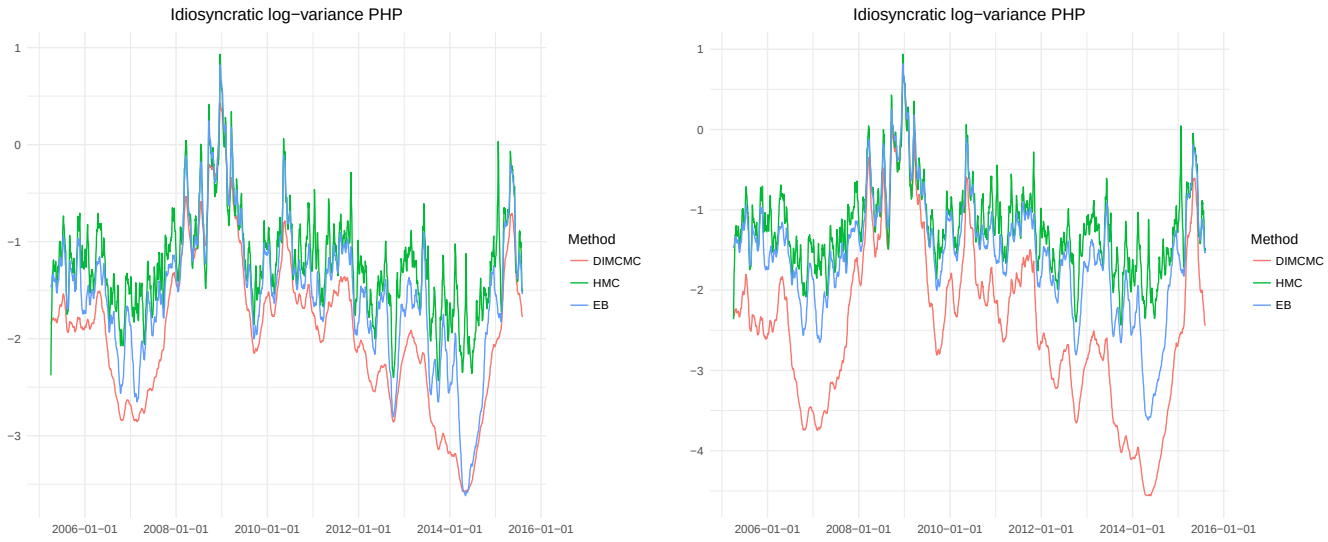


Fig. 10.10 Estimated idiosyncratic log-variance for PHP with a $\mathcal{B}(20, 1.5)$ (left) and $\mathcal{B}(10, 3)$ (right) prior on $\phi_i, i = 1, 2, 3$.

10.3 Summary

The results from this chapter is in agreement with the simulation study in chapter 8. When the likelihood converged, all methods gave similar results, with exception of one parameter, namely μ_{x_1}. When the likelihood did not converge, MLE and DIMCMC still gave similar results, just as in the simulation study. In the example in section 10.1,

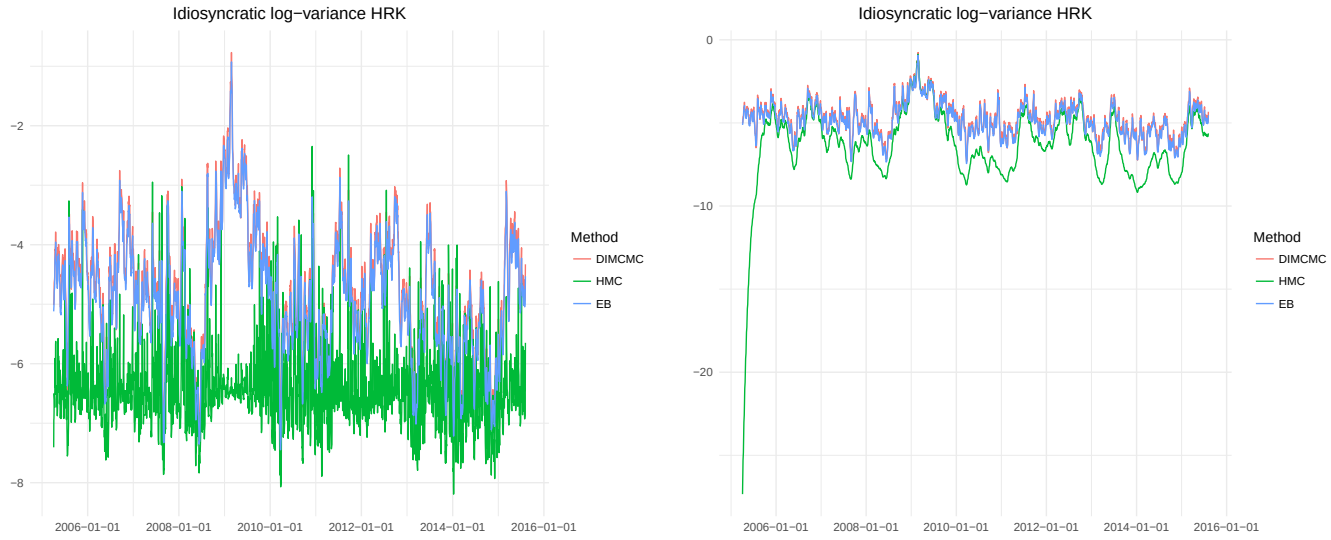


Fig. 10.11 Estimated idiosyncratic log-variance for HRK with a $\mathcal{B}(20, 1.5)$ (left) and $\mathcal{B}(10, 3)$ (right) prior on $\phi_i, i = 1, 2, 3$.

Fig. 10.12 Estimated factor log-variance for HRK and PHP with a $\mathcal{B}(20, 1.5)$ (left) and $\mathcal{B}(10, 3)$ (right) prior on $\phi, i = 1, 2, 3$.

DIMCMC and HMC was not sensitive to the choice of prior for ϕ, but in section 10.2, HMC clearly did not manage to identify the source of the variance. A take away may be that even though a $\mathcal{B}(20, 1.5)$ prior is justified for the idiosyncratic variance, it may be too restrictive for the factor process.

Chapter 11

Other Methods Investigated

11.1 Penalized Maximum Likelihood

Instead of minimizing the negative log-likelihood, we can add a penalizing term $\Omega(\boldsymbol{\theta})$ to the log-likelihood, such that we minimize

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} -l(\boldsymbol{\theta}) - \Omega(\boldsymbol{\theta}). \tag{11.1}$$

In this thesis, $\Omega(\boldsymbol{\theta})$ is set to the negative of the log prior described in chapter 7. For many of our parameters, we do not estimate them directly, but rather transformations of the parameter (for example $\log \sigma$ instead of σ). We therefore need to find the density of the transformed priors. This is done in Appendix A.

This was investigated to see if this could make convergence more robust. This was not the case. Penalizing the likelihood did not help convergence on datasets where the original likelihood did not converge.

11.2 Hybrid of MLE and Moment estimators

As seen in chapter 8, convergence was substantially better when we restricted our model to $\beta_1 = \beta_2$. To make the likelihood estimation easier we tried to estimate β_1

and β_2 inside the likelihood function by matching the empirical and the theoretical variance of y_{1t} and y_{2t}, and then use these estimates as input in the likelihood.

Let $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$ denote the empirical variance of y_{1t} and y_{2t}. Then

$$\text{Var}(y_{1t}) = \hat{\sigma}_1^2 = \beta_1^2 \exp\left(\frac{1}{2}\frac{\sigma_h^2}{1-\phi_h^2}\right) + \exp\left(\mu_{x_1} + \frac{1}{2}\frac{\sigma_{x_1}^2}{1-\phi_{x_1}^2}\right) \tag{11.2}$$

$$\text{Var}(y_{2t}) = \hat{\sigma}_2^2 = \beta_2^2 \exp\left(\frac{1}{2}\frac{\sigma_h^2}{1-\phi_h^2}\right) + \exp\left(\mu_{x_2} + \frac{1}{2}\frac{\sigma_{x_2}^2}{1-\phi_{x_2}^2}\right) \tag{11.3}$$

Solving for β_1 and β_2 we get

$$\hat{\beta}_1 = \frac{\sqrt{\hat{\sigma}_1^2 - \exp\left(\mu_{x_1} + \frac{1}{2}\frac{\sigma_{x_1}^2}{1-\phi_{x_1}^2}\right)}}{\exp\left(\frac{1}{2}\frac{\sigma_h^2}{1-\phi_h^2}\right)} \tag{11.4}$$

$$\hat{\beta}_2 = \frac{\sqrt{\hat{\sigma}_2^2 - \exp\left(\mu_{x_2} + \frac{1}{2}\frac{\sigma_{x_2}^2}{1-\phi_{x_2}^2}\right)}}{\exp\left(\frac{1}{2}\frac{\sigma_h^2}{1-\phi_h^2}\right)} \tag{11.5}$$

This hybrid of methods of moments with ML works well as long as the expression under the square root is positive, but often as the optimization routine is searching for a optimum, this becomes negative and the optimization stops when taking the square root of a negative number.

11.3 Combining the Laplace Approximation with HMC

When using HMC we sample the latent variables together with the parameters. Instead of sampling all latent variables we investigated how integrating out a subset affected the sampling. This was done with the **tmbstan** package in R. The user defines

the posterior density in a C++ template and defines what variables that should be integrated out by the Laplace approximation. We investigated integrating the factor process $\boldsymbol{h} = (h_1, \ldots, h_T)$. The traceplot of a simulation experiment with 500 observations from the two dimensional MFSV model is shown in figure 11.1. As we can see, the mixing for some of the variables are extremely slow making them almost useless. Another problem is the computational speed. The runtime for this example was 23 hours. This is due to the fact that between each leapfrog step used to evolve the Hamiltonian, we need to integrate out $\boldsymbol{h}$. For the MFSV model, we usually need 128 or 256 leapfrog steps between each Metropolis update, and thus need to apply the Laplace approximation 128 or 256 times for every update, making it computationally costly.

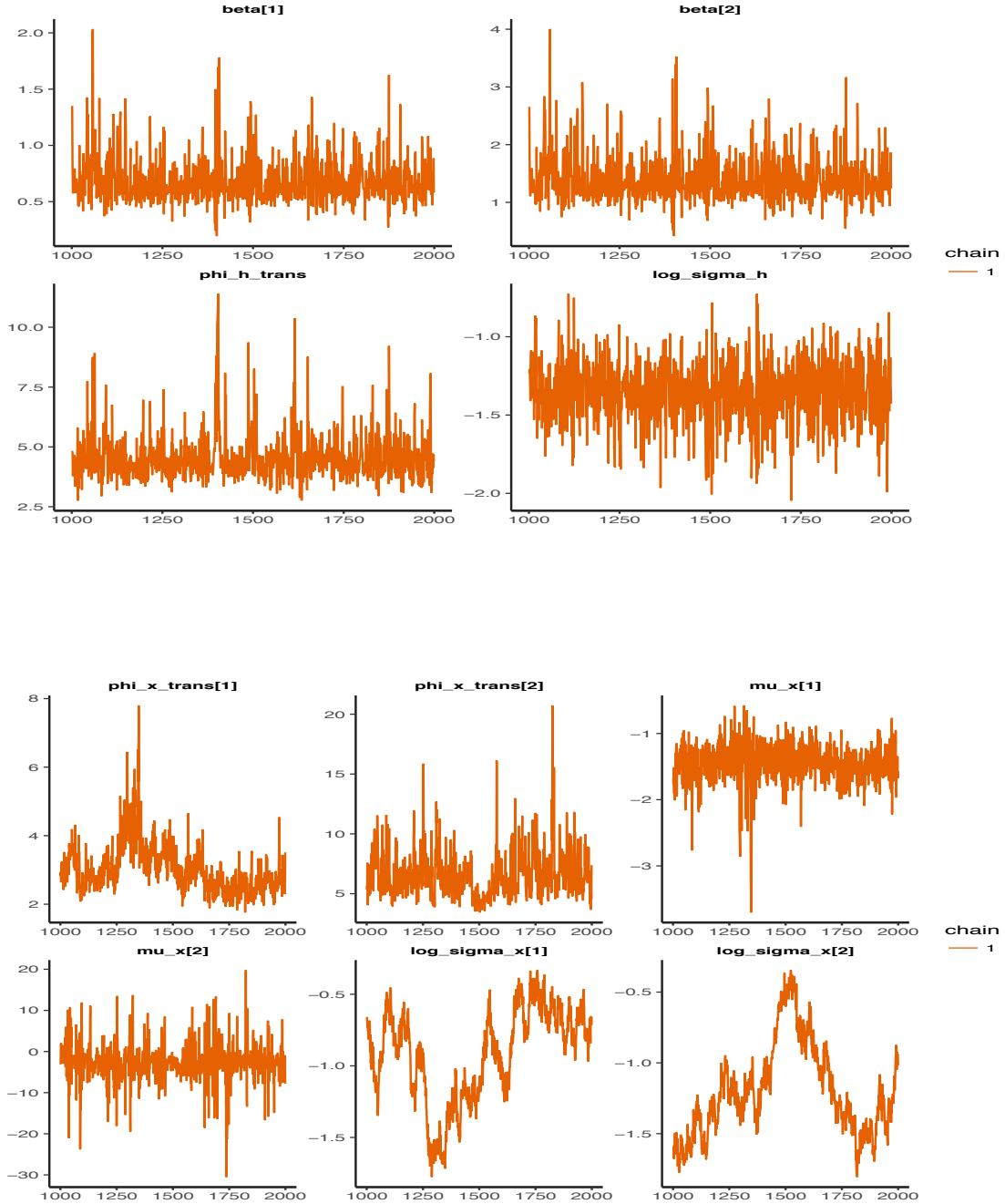


Fig. 11.1 Traceplots of simulation experiment combining HMC and the Laplace approximation. Note that `phi_x_trans` and `phi_h_trans` are the transformation of ϕ mentioned in chapter 8.

Chapter 12

Conclusion and future work

In this thesis we have investigated the problem of parameter estimation of dynamic covariance matrices through a factor stochastic volatility model using maximum likelihood and Hamiltonian Monte Carlo. In chapter 2 we introduced the building blocks of the MFSV model, namely the classical factor model and the univariate SV model. Chapter 3 to 5 consisted of theory needed for likelihood estimation and HMC. Chapter 5 gave an introduction to Automatic Differentiation and the different tools used for implementing the models, namely the R package **TMB** and the probabilistic programming language **Stan**.

In chapter 8 we did a simulation study comparing ML and the deep interweaving strategy (DIMCMC) proposed in Kastner et al. (2017) for the two dimensional factor stochastic volatility model. The results show that convergence of the likelihood is unstable and highly data dependent. Out of 1000 datasets, convergence was obtained in 611 cases, but most parameter estimates were accurate, independent of convergence. To investigate reasons to why the likelihood did not converge we calculated the characteristic function and the cumulative generating function of our model and found an expression for the higher order cumulants, that gave a suggestion of a possible restriction, namely putting the parameters in the loading vector equal. In a new simulation study with this restriction, the likelihood converged 996 out of 1000 datasets, but this was only the case when $\beta_1 = \beta_2$ in the data generating process.

The "Nested Laplace Approximation" was introduced in chapter 9, where instead of integrating over all latent variables, we do it over subsets in a sequential way. We applied the method to a linear state space model and the two dimensional MFSV model. For the linear state space model, all methods gave the same result. For the MFSV model the nested Laplace approximation converged, while the standard Laplace approximation did not. A big drawback was that the Hessian of the outer Laplace approximation was completely dense, making computations very slow.

We applied the different methods to real-world data in chapter 10, where we analyzed exchange rates with respect to EUR. Two scenarios were investigated, when the likelihood converged and when it did not. When the likelihood converged, all methods gave similar results. When it did not, ML and DIMCMC performed similarly, while HMC was sensitive to the choice of prior for the persistent parameters in the latent autoregressive processes.

For future studies, it would be very interesting to investigate simulating from the posterior using Riemann manifold Hamiltonian Monte Carlo (RMHMC) (Girolami et al. (2011)). Recall from chapter 5 that the covariance structure of the Gaussian momentum was a constant matrix $\boldsymbol{M}$. RMHMC generalizes HMC by using a position dependent Fisher information matrix $\boldsymbol{G}(\boldsymbol{\theta})$, that changes as we move through the parameter space. The kinetic energy will then have the form

$$\mathcal{K}(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}'\boldsymbol{G}(\boldsymbol{\theta})^{-1}\boldsymbol{p},$$

with corresponding Hamiltonian

$$\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p}) = -\log[\mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta}] + \frac{1}{2}\log\det(\boldsymbol{G}(\boldsymbol{\theta})) + \frac{1}{2}\boldsymbol{p}'\boldsymbol{G}(\boldsymbol{\theta})^{-1}\boldsymbol{p}$$

When the posterior is not Gaussian, level sets can have big local curvature, making exploration of the posterior hard (Betancourt (2017)). By choosing the the covariance matrix of the momentum to be the inverse Fisher information, we can adept to areas with big curvature.

On some datasets we experienced that the HMC sampler would get stuck in certain areas of the parameter space, for example when ϕ was very close to 1. One could hypothesize that this is caused by big local curvature, and it would therefore be interesting to see how RMHMC performed.

References

Bell, B. M. (2005). Cppad: A package for c++ algorithmic differentiation.

Betancourt, M. (2017). A Conceptual Introduction to Hamiltonian Monte Carlo. *ArXiv e-prints*.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327.

Bou-Rabee, N. and María Sanz-Serna, J. (2017). Geometric integrators and the Hamiltonian Monte Carlo method. *ArXiv e-prints*.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1):1–32.

Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D., Li, P., and Betancourt, M. (2015). The stan math library: Reverse-mode automatic differentiation in C++. *CoRR*, abs/1509.07164.

Casella, G. and Berger, R. (2001). *Statistical Inference*. Duxbury Resource Center.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics Letters B*, 195(2):216 – 222.

Engle, R. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007.

Fournier, D., Skaug, H., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M., Nielsen, A., and Sibert, J. (2011). Ad model builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. *Optimization Methods and Software*, 27(2):233–249.

Frühwirth-Schnatter, S. and Lopes, H. F. (2010). Parsimonious bayesian factor analysis when the number of factors is unknown.

Gelman, A., Robert, C., Chopin, N., and Rousseau, J. (2015). Bayesian data analysis.

Girolami, M., Calderhead, B., and Chin, S. A. (2011). Riemann manifold langevin and hamiltonian monte carlo methods. *J. of the Royal Statistical Society, Series B (Methodological*.

Harvey, A. (1991). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

Harvey, A., Ruiz, E., and Shephard, N. (1994). Multivariate stochastic variance models. *Review of Economic Studies*, 61(2):247–264.

Hautsch, N. and Ou, Y. (2008). Discrete-time stochastic volatility models and mcmc-based statistical inference. SFB 649 Discussion Papers SFB649DP2008-063, Humboldt University, Collaborative Research Center 649.

Hoffman, M. D. and Gelman, A. (2011). The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *ArXiv e-prints*.

Jolliffe, I. (1986). *Principal component analysis*. Springer series in statistics. Springer-Verlang.

Kastner, G. and Frühwirth-Schnatter, S. (2014). Ancillarity-sufficiency interweaving strategy (asis) for boosting mcmc estimation of stochastic volatility models. *Computational Statistics & Data Analysis*, 76:408 – 423. CFEnetwork: The Annals of Computational and Financial Econometrics.

Kastner, G., Frühwirth-Schnatter, S., and Lopes, H. F. (2017). Efficient bayesian inference for multivariate factor stochastic volatility models. *Journal of Computational and Graphical Statistics*, 26(4):905–917.

Kristensen, K., Nielsen, A., Berg, C., Skaug, H., and Bell, B. (2016). Tmb: Automatic differentiation and laplace approximation. *Journal of Statistical Software, Articles*, 70(5):1–21.

Lawley, D. and Maxwell, A. (1971). *Factor Analysis as a Statistical Method*. Butterworths mathematical texts. Butterworths.

Liesenfeld, R. (2006). Classical and bayesian analysis of univariate and multivariate stochastic volatility models.

Monnahan, C. C., Thorson, J. T., and Branch, T. A. (2016). Faster estimation of bayesian models in ecology using hamiltonian monte carlo. *Methods in Ecology and Evolution*, pages n/a–n/a.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Neal, R. M. (2010). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162.

Omori, Y., Chib, S., Shephard, N., and Nakajima, J. (2004). Stochastic volatility with leverage: fast likelihood inference. Economics Papers 2004-W19, Economics Group, Nuffield College, University of Oxford.

Paquet, U. and Fraccaro, M. (2016). An Efficient Implementation of Riemannian Manifold Hamiltonian Monte Carlo for Gaussian Process Models. *ArXiv e-prints*.

Pawitan, Y. (2001). *In All Likelihood*. First edition.

Pitt, M. K. and Shephard, N. (1999). Time-varying covariances: a factor stochastic volatility approach. In *Bayesian statistics, 6 (Alcoceber, 1998)*, pages 547–570. Oxford Univ. Press, New York.

Rice, J. (1988). *Mathematical statistics and data analysis*. Wadsworth & Brooks/Cole statistics/probability series. Brooks/Cole Pub. Co.

Schervish, M. (1996). *Theory of Statistics*. Springer Series in Statistics. Springer New York.

Shephard, N., editor (2005). *Stochastic Volatility: Selected Readings*. Oxford University Press.

Skaug, H. J. and Yu, J. (2014). "a flexible and automated likelihood based framework for inference in stochastic volatility models". *"Computational Statistics & Data Analysis"*, "76":642 – 654. CFEnetwork: The Annals of Computational and Financial Econometrics.

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15:88–103.

Stan Development Team (2016). RStan: the R interface to Stan. R package version 2.14.1.

Taylor, S. J. (1982). Financial returns modelled by the product of two stochastic processes-a study of the daily sugar prices 1961-75. *Time Series Analysis : Theory and Practice*, 1:203–226.

Team, S. D. (2017). *Stan Modeling Language Users Guide and Reference Manual, Version 2.17.0.*

Tsay, R. (2010). *Analysis of Financial Time Series*. CourseSmart. Wiley.

Warwick, T. (2010). Automatic differentiation - lecture no 1.

Yu, Y. and Meng, X.-L. (2011). To center or not to center: That is not the question—an ancillarity–sufficiency interweaving strategy (asis) for boosting mcmc efficiency. *Journal of Computational and Graphical Statistics*, 20(3):531–570.

Zhou, X., Nakajima, J., and West, M. (2014). Bayesian forecasting and portfolio decisions using dynamic dependent sparse factor models. *International Journal of Forecasting*, 30(4):963 – 980.

Appendix A

Transformation of priors

A.1 Transformation of ϕ

As seen in chapter 7, to ensure $\phi \in (-1, 1)$, we choose $\frac{\phi+1}{2} \sim \mathcal{B}(a_0, b_0)$, implying

$$f_\phi(\phi) = \frac{1}{2B(a_0, b_0)} \left(\frac{\phi+1}{2}\right)^{a_0-1} \left(\frac{1-\phi}{2}\right)^{b_0-1}$$

To be able to do unrestricted likelihood estimation we estimate $\tilde{\phi} \in \mathbb{R}$, and then transform back to $(-1, 1)$ by the function

$$\phi = \frac{\exp(\tilde{\phi}) - 1}{1 + \exp(\tilde{\phi})}.$$

We need to find the distribution of

$$\tilde{\phi} = g(\phi) = \log\left(\frac{\phi+1}{1-\phi}\right)$$

The inverse of g is the original transformation

$$h(\tilde{\phi}) = g^{-1}(\phi) = \frac{\exp(\tilde{\phi}) - 1}{1 + \exp(\tilde{\phi})},$$

and the derivative of h is

$$h'(\tilde{\phi}) = \frac{2\exp(\tilde{\phi})}{(1+\exp(\tilde{\phi}))^2}$$

Thus, by the transformation formula the density of $\tilde{\phi}$ is given by

$$\begin{aligned} f_{\tilde{\phi}}(\tilde{\phi}) &= f_{\phi}(h(\tilde{\phi}))\left|h'(\tilde{\phi})\right| \\ &= \frac{1}{2B(a_0,b_0)}\left(\frac{h(\tilde{\phi})+1}{2}\right)^{a_0-1}\left(\frac{1-h(\tilde{\phi})}{2}\right)^{b_0-1}\frac{2\exp(\tilde{\phi})}{(1+\exp(\tilde{\phi}))^2} \\ &= \frac{1}{B(a_0,b_0)}\left(\frac{h(\tilde{\phi})+1}{2}\right)^{a_0-1}\left(\frac{1-h(\tilde{\phi})}{2}\right)^{b_0-1}\frac{\exp(\tilde{\phi})}{(1+\exp(\tilde{\phi}))^2}. \end{aligned}$$

A.2 Transformation of σ

To ensure $\sigma > 0$, we estimate $\log\sigma$ instead of σ. The prior of σ^2 is $\mathcal{G}\left(\frac{1}{2}, \frac{1}{2B_\sigma}\right)$, with density

$$f_{\sigma^2}(\sigma^2) = \frac{1}{\Gamma\left(\frac{1}{2}\right)}\frac{1}{\sqrt{2B_\sigma}}(\sigma^2)^{-1/2}\exp\left(-\sigma^2/(2B_\sigma)\right),$$

We need to find the distribution of $\tilde{\sigma} = g(\sigma^2) = \log\sqrt{\sigma^2}$.

The inverse of g is given by $h(\tilde{\sigma}) = g^{-1}(\sigma^2) = \exp(2\tilde{\sigma})$ with derivative $h'(\tilde{\sigma}) = 2\exp(2\tilde{\sigma})$. The density of $\tilde{\sigma}$ is then given by

$$\begin{aligned} f_{\tilde{\sigma}}(\tilde{\sigma}) &= f_{\sigma^2}(h(\tilde{\sigma}))\left|h'(\tilde{\sigma})\right| \\ &= \frac{1}{\Gamma\left(\frac{1}{2}\right)}\frac{1}{\sqrt{2B_\sigma}}\exp(\tilde{\sigma})^{-1/2}\exp\left(\frac{\exp(2\tilde{\sigma})}{2B_\sigma}\right)2\exp(\tilde{\sigma}) \\ &= \frac{1}{\Gamma\left(\frac{1}{2}\right)}\sqrt{\frac{2}{B_\sigma}}\exp\left(\tilde{\sigma} - \frac{\exp(2\tilde{\sigma})}{2B_\sigma}\right). \end{aligned}$$

Appendix B

Code Snippets

B.1 TMB code for the Multivariate Factor Stochastic Volatility Model

```
//Joint probability distribution of p-dimensional MFSV model with q factors. Factors
    and indiosyncratic variance are driven by latent AR(1) processes.
// Apply same priors as in Kastner (2016) for HMC and penalized likelihood

#include <TMB.hpp>
//Helper function to transform phi
template<class Type>
vector<Type> f(vector<Type> x){
  vector<Type> y =  (exp(x)-Type(1))/(Type(1) + exp(x));
  return y;
}

template <class Type>
Type objective_function<Type>::operator()()
{
  DATA_ARRAY(y);
  DATA_INTEGER(n); // number of obs
  DATA_INTEGER(p); // dim of y
  DATA_INTEGER(q); // dim of factor
  DATA_INTEGER(prior);


  PARAMETER_MATRIX(beta); // loading matrix
```

```
PARAMETER_MATRIX(h); // latent processes for factors
PARAMETER_MATRIX(x); // Idiosyncratic processes
PARAMETER_VECTOR(phi_h_trans);
PARAMETER_VECTOR(log_sigma_h);
PARAMETER_VECTOR(phi_x_trans);
PARAMETER_VECTOR(mu_x);
PARAMETER_VECTOR(log_sigma_x);

vector<Type> phi_h = f(phi_h_trans);
vector<Type> phi_x = f(phi_x_trans);
ADREPORT(phi_h);
ADREPORT(phi_x);

vector<Type> sigma_h = exp(log_sigma_h);
ADREPORT(sigma_h);
vector<Type> sigma_x = exp(log_sigma_x);
ADREPORT(sigma_x);

using namespace density;
using namespace Eigen;

Type nll = 0;

// Contribution from latent variables

// Log-variance of factor
// Stationarity assumption
for(int i = 0; i < q; i++){
  nll -= dnorm(h(i,0), Type(0.0), sigma_h(i)/sqrt(Type(1.0) - phi_h(i)*phi_h(i)),
      true);
}


for(int i = 1; i < n; i++){
  for(int j = 0; j < q; j++){
    nll -= dnorm(h(j,i),phi_h(j)*h(j,i-1),sigma_h(j),true);
  }
}

// Log-variance idiosyncratic
// Stationarity assumption
for(int i = 0; i < p; i++){
  nll -= dnorm(x(i,0), mu_x(i), sigma_x(i)/sqrt(1 - phi_x(i)*phi_x(i)),true);
}
```

```
for(int i = 1; i < n; i++){
  for(int j = 0; j < p; j++){
    nll -= dnorm(x(j,i), mu_x(j) + phi_x(j)*(x(j,i-1) - mu_x(j)), sigma_x(j), true)
        ;
  }
}
// Random effects done

//Contribtution from observations
matrix<Type> factor(q,q); //Variance from factors
matrix<Type> idiosyncratic(p,p); //Idiosyncratic variance
factor.setZero();
idiosyncratic.setZero();

// Covariance matrix
matrix<Type> Sigma(p,p);
matrix<Type> bt = beta.transpose();


for(int i = 0; i < n; i++){
 // Fill diagonal factor varians
  for(int j = 0; j < q; j++) factor(j,j) = exp(h(j,i));
  // Fill diagonal idiosyncratic varians
  for(int j = 0; j < p; j++) idiosyncratic(j,j) = exp(x(j,i));

  matrix<Type> tmp = beta*factor; // Need to multiply one by one not to get error
  matrix<Type> tmp2 = tmp*bt;
  Sigma = tmp2 + idiosyncratic;
  // N_0_Sigma is multivariate normal with covariance matrix Sigma
  MVNORM_t<Type> N_0_Sigma(Sigma);
  nll += N_0_Sigma(vector<Type>(y.col(i)));
}

REPORT(x);
REPORT(h);

// Penalized likelihood?
if(prior == 1){
  //Constants
  Type B_beta = 1.0;
  Type b_mu = 0.0;
  Type B_mu = 100;
  Type a_0 = 20;
```

```
    Type b_0 = 1.5;
    Type B_sigma = 1;

  // Log-priors for all variables of length p.
  // TMB uses shape/scale in the gamma distribution,
  // therefore the scale = 1/rate, where rate = 1/2_B_sigma
    for(int i = 0; i < p; i++){
      // Contribution from mu
      nll -= dnorm(mu_x(i),b_mu,sqrt(B_mu),true);
      // Contribution form phi_x
      nll -= dbeta((phi_x(i)+Type(1))/Type(2),a_0,b_0,true) + phi_x_trans(i) - log((
          Type(1) + exp(phi_x_trans(i)))*(Type(1) + exp(phi_x_trans(i))));
      // Contribution from log_sigma_x
      nll -= -lgamma(Type(0.5)) + log(sqrt(Type(2)/B_sigma)) +  log_sigma_x(i) - exp(
          Type(2)*log_sigma_x(i)/(Type(2)*B_sigma));
    }
    // Priors for all variables of length q
    for(int i = 0; i < q; i++){
      // Contribution from phi_h
      nll -= dbeta((phi_h(i)+Type(1))/Type(2),a_0,b_0,true) + phi_h_trans(i) - log((
          Type(1) + exp(phi_h_trans(i)))*(Type(1) + exp(phi_h_trans(i))));
      // Contribution from log_sigma_h
      nll -= -lgamma(Type(0.5)) + log(sqrt(Type(2)/B_sigma)) +  log_sigma_h(i) - exp(
          Type(2)*log_sigma_h(i)/(Type(2)*B_sigma));
    }

    // Priors from beta need to be split in two: First loop over the lower triangular
        of the first rows and columns, then from q+1 to p where we take contribution
        from entire matrix.

    for(int i = 0; i < q; i++){
      for(int j = 0; j <= i; j++){
        nll -= dnorm(beta(i,j),Type(0),sqrt(B_beta),true);
      }
    }
    for(int i = q; i < p; i++){
      for(int j = 0; j < q; j++){
        nll -= dnorm(beta(i,j),Type(0),sqrt(B_beta),true);
      }
    }
  }

    return nll;
}
```

B.2 TMB code for Nested Laplace of linear state space model

```
#include<TMB.hpp>


// Linear state space model on the form y_it = h_t + x_it + e_it, where h and x are
    uncentered AR(1) processes.


//Inner Laplace Approximation
template<class Type, class Functor>
struct laplace_t {
  Functor f; //Negative-log likelihood
  vector<Type>& u; // Latent variables
  int niter;       // Number of Newton iterations
  laplace_t(Functor f_, vector<Type> &u_, int niter_) :
    f(f_), u(u_), niter(niter_) {}
  Type operator()(){
    // Find optimum of Inner problem
    for (int i=0; i<niter; i++){
      vector<Type> g = autodiff::gradient(f, u);
      matrix<Type> H = autodiff::hessian(f, u);
      u -= atomic::matinv(H) * g;
    }
    // Laplace approximation
    matrix<Type> H = autodiff::hessian(f, u);
    Type ans = .5 * atomic::logdet(H) + f(u);
    ans -= .5 * Type(u.size()) * log(2.0 * M_PI);
    return ans;
  }
};
template<class Type, class Functor>
Type laplace(Functor f, vector<Type> &u, int niter){
  laplace_t<Type, Functor> L(f, u, niter);
  return L();
}


template<class Type>
struct jnll_t{
  matrix<Type> y;
  vector<Type> h, sigma_x,sigma, phi_x;
```

```
Type phi_h, sigma_h;
jnll_t(matrix<Type> y_,
       vector<Type> h_,vector<Type> sigma_x_, vector<Type> sigma_, vector<Type>
           phi_x_,
       Type phi_h_,Type sigma_h_) :
  y(y_), h(h_), sigma_x(sigma_x_), sigma(sigma_), phi_x(phi_x_), phi_h(phi_h_),
      sigma_h(sigma_h_) {}
template<typename T>
T operator()(vector<T> x_vec){
  matrix<T> y_ = y.template cast<T>();
  vector<T> h_ = h.template cast<T>();
  vector<T> sigma_x_ = sigma_x.template cast<T>();
  vector<T> sigma_ = sigma.template cast<T>();
  vector<T> phi_x_ = phi_x.template cast<T>();
  T phi_h_ = T(phi_h);
  T sigma_h_ = T(sigma_h);

  int h = x_vec.size();
  int n = h_.size();
  //Transform to matrix
  matrix<T> x(2,n);
  for(int i = 0; i < h; i++){
    if(i < n){
      x(0,i) = x_vec(i);
    }
    else{
      x(1,i - n) = x_vec(i);
    }
  }

  T nll = 0;

  // Stationary assumption
  nll -= dnorm(x(0,0),T(0),sigma_x_(0)/sqrt(T(1) - phi_x_(0)*phi_x_(0)),true);
  nll -= dnorm(x(1,0),T(0),sigma_x_(1)/sqrt(T(1.0) - phi_x_(1)*phi_x_(1)),true);
  nll -= dnorm(h_(0),T(0), sigma_h_/sqrt(T(1.0) - phi_h_*phi_h_),true);

  // Contribution from h
  for(int i = 1; i < n; i++){
    nll -= dnorm(h_(i),h_(i-1)*phi_h_,sigma_h_,true);
  }

  // Contribution from x
  for(int i = 1; i < n; i++){
```

```
      for(int j = 0; j < 2; j++){
        nll -= dnorm(x(j,i),phi_x_(j)*x(j,i-1),sigma_x_(j),true);
      }
    }

    // Observations
    for(int i = 0; i < n; i++){
      nll -= dnorm(y_(0,i),h_(i) + x(0,i),sigma_(0),true);
      nll -= dnorm(y_(1,i),h_(i) + x(1,i),sigma_(1),true);
    }
    return nll;
  }
};

template<class Type>
Type objective_function<Type>::operator()(){
  DATA_MATRIX(y);
  DATA_INTEGER(n);
  DATA_INTEGER(niter);
  PARAMETER(phi_h);
  PARAMETER(log_sigma_h);
  PARAMETER_VECTOR(phi_x);
  PARAMETER_VECTOR(log_sigma_x)
  PARAMETER_VECTOR(log_sigma);
  PARAMETER_VECTOR(h);


  vector<Type> sigma = exp(log_sigma);
  vector<Type> sigma_x = exp(log_sigma_x);
  Type sigma_h = exp(log_sigma_h);
  ADREPORT(sigma);
  ADREPORT(sigma_x);
  ADREPORT(sigma_h);

  Type nll = 0;

  vector<Type> x_vec(2*n);
  /*
   * We need to vectorize x such that we can get the gradient and Hessian
   */
  x_vec.setZero();
  jnll_t<Type> neg_log(y,h,sigma_x,sigma,phi_x,phi_h,sigma_h);

  nll = laplace(neg_log,x_vec,niter);
```

```
  ADREPORT(x_vec);
  return nll;
}
```

The code for the factor model is similar and there omitted.

B.3 TMB code for Laplace approximation in non-optimum

```
template<class Type, class Functor>
Type laplace(Functor f, vector<Type> x){
    matrix<Type> H = autodiff::hessian(f,x);
    vector<Type> g = autodiff::gradient(f,x);
    matrix<Type> g_mat = g.matrix();
    matrix<Type> tmp = g_mat.transpose()*atomic::matinv(H);
    matrix<Type> tmp2 = tmp*g;
    Type res = Type(0.5)*atomic::logdet(H) + f(x) - Type(0.5)*(tmp2)(0);
    res -= .5 * Type(x.size()) * log(2.0 * M_PI);
    return res;
}
```

B.4 Stan code for MVFS model

```
//2 dim factor model

data {
  int<lower=0> n; // Sample size
  matrix[2,n] y;  // data
  vector<lower = 0>[6] prior_param;  // Prior parameters
}

transformed data {
  real B_beta = prior_param[1];
  real b_mu = prior_param[2];
  real B_mu = prior_param[3];
  real a0 = prior_param[4];
  real b0 = prior_param[5];
  real B_sigma = prior_param[6];
```

```
}
parameters {
  vector[n] h_std; //standardized AR(1) for factor
  matrix[2,n] x_std; //standardized AR(1) idiosyncratic process
  vector[2] beta; //loading matrix (vector in this case)
  real<lower=0,upper=1> phi_h_std;
  real<lower=0> sigma_h_std;
  vector<lower=0,upper=1>[2] phi_x_std;
  vector[2] mu_x;
  vector<lower = 0>[2] sigma_x_std;
}

transformed parameters {
  matrix[2,n] x; //indiosyncratic processes
  vector[n] h;   // factor volatility
  real<lower=-1,upper=1> phi_h;
  real<lower=0> sigma_h;
  vector<lower=-1,upper=1>[2] phi_x;
  vector<lower=0>[2] sigma_x;

  phi_h = 2*phi_h_std - 1;
  sigma_h = sqrt(sigma_h_std);
  phi_x = 2*phi_x_std - 1;
  sigma_x = sqrt(sigma_x_std);

  //Scale with standard deviation
  h = h_std * sigma_h;
  h[1] = h[1]/sqrt(1 - square(phi_h));

  //Scale with standard deviation
  //Stationary assumption on x
  for(t in 1:2){
    x[t,] = x_std[t,]*sigma_x[t];
    x[t,] = x[t,] + mu_x[t];
    x[t,1] = x[t,1]/sqrt(1 - square(phi_x[t]));
  }
  for(t in 2:n){
    h[t] = h[t] + phi_h*h[t-1];
    for(i in 1:2){
      x[i,t] = x[i,t] + phi_x[i]*(x[i,t-1] - mu_x[i]);
    }
  }
}
```

```
model {
  vector[2] null = rep_vector(0,2);
  matrix[2,2] Sigma; //Covariance matrix for observations as function of parameters
  matrix[2,2] Sigma_chol;

  //Priors
  beta ~ normal(null,B_beta);
  mu_x ~ normal(b_mu,B_mu);
  sigma_h_std ~ gamma(0.5,1/(2*B_sigma));
  sigma_x_std ~ gamma(0.5,1/(2*B_sigma));
  phi_h_std ~ beta(a0,b0);
  phi_x_std ~ beta(a0,b0);

  h_std ~ normal(0,1);
  x_std[1,] ~ normal(0,1);
  x_std[2,] ~ normal(0,1);

  //Constribution from observations
  for(i in 1:n){
    Sigma[1,1] = square(beta[1])*exp(h[i]) + exp(x[1,i]);
    Sigma[1,2] = beta[1]*beta[2]*exp(h[i]);
    Sigma[2,2] = square(beta[2])*exp(h[i]) + exp(x[2,i]);
    Sigma[2,1] = Sigma[1,2];
    Sigma_chol = cholesky_decompose(Sigma);
    y[,i] ~ multi_normal_cholesky(null,Sigma_chol);
  }
}

generated quantities{
    matrix[2,n] y_new;
    matrix[2,2] Sigma; //Covariance matrix for observations as function of parameters
    vector[2] null;
    null[1] = 0;
    null[2] = 0;
  for(i in 1:n){
    Sigma[1,1] = square(beta[1])*exp(h[i]) + exp(x[1,i]);
    Sigma[1,2] = beta[1]*beta[2]*exp(h[i]);
    Sigma[2,2] = square(beta[2])*exp(h[i]) + exp(x[2,i]);
    Sigma[2,1] = Sigma[1,2];

  y_new[,i] = multi_normal_rng(null,Sigma);
  }
}
```