D I P L O M A R B E I T

Bayesian Time Series Analysis

Ausgeführt am Institut für
Statistik und Wahrscheinlichkeitstheorie
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Klaus Felsenstein

durch
Johann Füchsl, BSc
Bennogasse 29
1080 Wien

Wien, Mai 2012

Contents

Chapter 1

Introduction

This chapter serves as an overview of the fundamentals for the subject matter. For more details on Bayesian statistics see [7] and [5] for stochastic processes and time series. An elementary introduction to Bayesian time series analysis is [13] which will be summarized in chapter 2.

Section 1.1 deals with basic concepts of measure- and probability theory. Subsequently we summarize conditional probabilities and conditional expectations with respect to a σ-algebra that represents the state of knowledge.

Sections 1.3 and 1.4 introduce Bayes theorem and families of conjugate prior distributions.

We conclude the chapter in section 1.5 with an introduction to stochastic processes and time series with some examples of time series models.

1.1 Basics

Here we briefly summarize some important definitions and notation that will be used throughout this thesis. For more details see [8], [3] and [2].

Let Ω be a set and $\mathcal{F}$ a σ-algebra on Ω. The tuple $(\Omega, \mathcal{F})$ is a *measurable space* and the elements in $\mathcal{F}$ are the *measurable sets*. A mapping $\mu : \mathcal{F} \to [0, \infty]$ is a *measure*, iff

1. $\mu(\emptyset) = 0$
2. $\mu(\bigcup_i A_i) = \sum_i \mu(A_i)$ for pairwise disjoint $A_i \in \mathcal{F}$, $i \in I \subseteq \mathbb{N}$

is satisfied. Together with a measurable space, a measure μ forms a *measure space*, denoted $(\Omega, \mathcal{F}, \mu)$. A measure P that satisfies $P(\Omega) = 1$ is called a *probability measure* and $(\Omega, \mathcal{F}, P)$ is a *probability space*.

Let $(\Omega_i, \mathcal{F}_i)$, $i = 1, 2$ be measurable spaces and $f : \Omega_1 \to \Omega_2$ a function. f is $(\mathcal{F}_1, \mathcal{F}_2)$-*measurable* if $f^{-1}(A) \in \mathcal{F}_1$ for all $A \in \mathcal{F}_2$. We omit the σ-algebras and simply call f measurable when there is no confusion.

The set $\{\omega \in \Omega : f(\omega) \in A\}$ will sometimes be denoted $[f \in A]$ and similarly $[f = x] := \{\omega \in \Omega : f(\omega) = x\}$.

Let $(\Omega, \mathcal{F}, \mu)$ be a measure space and E some property which makes sense for elements $\omega \in \Omega$. We say the property E holds *almost everywhere with respect to* μ ($\mu - a.e.$), if there exists a set $N \in \mathcal{F} : \mu(N) = 0$, such that E holds for all

elements $\omega \in N^c := \Omega \setminus N$. If μ is a probability measure, we say that E holds *almost surely* with respect to μ ($\mu - a.s.$).

The Borel-σ-algebra of $\mathbb{R}^d$ is denoted $\mathcal{B}(\mathbb{R}^d)$. A measurable function $f : (\Omega, \mathcal{F}) \to (\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ is called *Borel-measurable*.

Let $(\Omega, \mathcal{F}, P)$ be a probability space. A Borel-measurable function $X : (\Omega, \mathcal{F}) \to (\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ is called a *random variable*. The *distribution* of a random variable X is the probability measure defined by $p_X := P \circ X^{-1}$. When X is integrable, the *expectation* of X is defined by $\mathbb{E}(X) := \int_\Omega X dP = \int_{\mathbb{R}^d} p_X(dx)$.

1.2 Conditional probabilities

Definition 1. *Let $(\Omega, \mathcal{F}, P)$ be a probability space and $A, B \in \mathcal{F}$, such that $P(B) > 0$. Then*

$$P(A|B) := \frac{P(A \cap B)}{P(B)} \tag{1.1}$$

is the conditional probability of A given B.

It is easy to verify that $A \mapsto P(A|B)$ is a probability measure on $(B, B \cap \mathcal{F})$. This definition of conditional probability requires the conditioning event B to have non-zero probability and is therefore unsuited for continuous probability distributions, since in this case, events of the form $[X = x]$ have zero probability.

For a more general form of conditional probabilities we first introduce *conditional expectations*.

Definition 2. *Let $(\Omega, \mathcal{F}, P)$ be a probability space, $\mathcal{C} \subseteq \mathcal{F}$ a sub-σ-algebra, X a random variable on $(\Omega, \mathcal{F}, P)$ and $P_0 := P|_\mathcal{C}$ the restriction of P on $\mathcal{C}$. Each $\mathbb{R}$-valued measurable function g_X on $(\Omega, \mathcal{C}, P_0)$, which satisfies*

$$\int_C g_X dP_0 = \int_C X dP \quad \forall C \in \mathcal{C}$$

is called a version of the conditional expectation of X with respect to $\mathcal{C}$ *and is denoted by $\mathbb{E}(X|\mathcal{C}) := \mathbb{E}_P(X|\mathcal{C}) := g_X$.*

For an event $A \in \mathcal{F}$, $P(A|\mathcal{C}) := \mathbb{E}(\chi_A|\mathcal{C})$ is called conditional probability of A with respect to $\mathcal{C}$. *χ_A denotes the indicator function for the set A.*

Theorem 1. *Under the assumptions of definition 2, a version $g_X = \mathbb{E}(X|\mathcal{C})$ of the conditional expectation exists and is $P_0 - a.s.$ unique.*

Proposition 1. *Let $A, A_n \in \mathcal{F}$, $n \in \mathbb{N}$. Then*

1. $P(\Omega|\mathcal{C}) = 1 \quad P_0 - a.s.$
2. $0 \leq P(A|\mathcal{C}) \leq 1 \quad P_0 - a.s.$
3. *For pairwise disjoint $(A_n)_{n\in\mathbb{N}}$ the relation*

$$P(\bigcup_{n\in\mathbb{N}} A_n|\mathcal{C}) = \sum_{n\in\mathbb{N}} P(A_n|\mathcal{C})$$

holds $P_0 - a.s.$

Remark 1. *Since the exceptional sets of $P(A|\mathcal{C})$ depend on the event A there are uncountably infinitely many exceptional sets, who in general do not add up to a set with probability zero. Therefore $P(.|\mathcal{C})$ is not a probability measure in general.*

A measure where this restriction does not apply is called a regular conditional probability measure.

Definition 3. *Let $(\Omega, \mathcal{F}, P)$ be a probability space, $\mathcal{C} \subseteq \mathcal{F}$ a sub-σ-algebra and $A \in \mathcal{F}$ an event. Then $K : \Omega \times \mathcal{C} \to [0, \infty] : (x, A) \mapsto P(A|\mathcal{C})(x)$ with $K(\cdot, A)$ measurable for all $A \in \mathcal{C}$ and $K(\omega, \cdot)$ a probability measure for all $\omega \in \Omega$ is called a version of the* regular conditional probability of A with respect to $\mathcal{C}$ *(under the information $\mathcal{C}$).*

The next theorem states that we can express conditional expectations as ordinary expectations under a regular conditional probability measure.

Theorem 2. *Let $P(.|\mathcal{C})$ be a conditional probability and X a random variable on $(\Omega, \mathcal{F}, P)$. Then*

$$\mathbb{E}(X|\mathcal{C})(x) = \int_\Omega X(\omega) P(d\omega|\mathcal{C})(x) \quad P_0 - a.s.$$

Fortunately, regular conditional probabilities exist under rather general assumptions.

Definition 4. *Let $(\Omega, \mathcal{T})$ be a topological space. If there exists a metric d, that generates $\mathcal{T}$ such that, (Ω, d) is a complete, separable, metric space, we call $(\Omega, \mathcal{T})$ a* polnish space.

Theorem 3. *Let Ω be a polnish spcae, $(\Omega, \mathcal{B}(\Omega), P)$ a probability space and $\mathcal{C} \subseteq \mathcal{B}(\Omega)$ a sub-σ-algebra. There exists a regular version of the conditional probability $P(.|\mathcal{C})$.*

We can describe the conditional expectation (probability) of a random variable X with respect to a different random variable Y through the framework of sub-σ-algebras introduced above. In this case we set the conditioning σ-algebra $\mathcal{C} = \sigma(Y)$ as the σ-algebra generated by Y. This is the smallest σ-algebra under which Y is measurable. When conditioning with respect to events A_i, $i \in I \subseteq \mathbb{N}$ (e.g. $A_i = [Y_i = y_i]$) we take $\mathcal{C} = \sigma((A_i)_{i \in I})$, the smallest σ-algebra that contains all sets A_i, $i \in I$.

1.2.1 Conditional densities

We now take a look at conditional densities, which allow us to make calculations with conditional probabilities. Suppose X and Y are random variables with values in $(\mathbb{R}^k, \mathcal{B}(\mathbb{R}^k))$ and $(\mathbb{R}^m, \mathcal{B}(\mathbb{R}^m))$. Further, let (X, Y) have a density f with respect to $\nu \otimes \mu$, where ν and μ are σ-finite measures on $(\mathbb{R}^k, \mathcal{B}(\mathbb{R}^k))$ and $(\mathbb{R}^m, \mathcal{B}(\mathbb{R}^m))$ respectively. The density of the conditional distribution of $X|Y$ is given by

$$\frac{dP^{X|Y}}{d\nu} = \frac{f(x, y)}{\int f(x, y)\nu(dx)} \quad \mu - a.e.,$$

or if there is no confusion about ν and μ

$$f(x|y) = \frac{f(x,y)}{f_Y(y)},$$

where $f_Y(y) = \int f(x,y)\nu(dx)$ is the marginal density of Y.

1.3 Bayes theorem

In this section we introduce Bayes theorem which allows us to update probabilities of events when presented with new pieces of information (e.g. observations of random variables). We start with a probability distribution that represents our initial state of knowledge, called the *prior* distribution. Updating the prior with the new information through Bayes theorem results in the *posterior* distribution which represents our newly aquired state of knowledge.

The law of total probabilities relates ordinary probabilities to conditional probabilities and forms the basis for Bayes theorem.

Theorem 4. *Let $\mathcal{Z} := \{H_i : i \in I \subseteq \mathbb{N}\}$ be a measurable partition of Ω and $A \in \mathcal{F}$. Then*

$$P(A) = \sum_{i \in I} P(A|H_i)P(H_i).$$

The events H_i are called hypotheses.

In the more general case where hypotheses are realizations of random variables (e.g. $H_i = [X = x_i]$) and the event A is of the form $A = [Y \in A']$ with a random variable Y, the law of total probabiliy is written as the integral

$$P(Y \in A') = \int \int_{A'} f(y|x) f_X(x) dy dx.$$

Here $f(y|x)$ is the conditional density of $Y|X = x$ and $f_X(x)$ is the density of the marginal distribution of X.

Theorem 5 (Bayes formula)**.** *Let $\mathcal{Z}$ be a measurable partition of Ω and $A \in \mathcal{F}$ with $P(A) > 0$. Then*

$$P(H_i|A) = \frac{P(A \cap H_i)}{P(A)} = \frac{P(A|H_i)P(H_i)}{\sum_{i \in I} P(A|H_i)P(H_i)}.$$

Again this can be generalized to events $H_i = [X = x_i]$, $A = [Y \in A']$ and densities $f(y|x)$ and $f_X(x)$, resulting in

$$f(x|y) = \frac{f(y|x)f_X(x)}{\int f(y|x)f_X(x)dx}. \tag{1.2}$$

The denominator is independent of x and serves as normalization constant. Thus (1.2) can be written as $f(x|y) \propto f(y|x)f_X(x)$. The symbol $\propto$ expresses proportionality.

Bayes theorem is the application of the Bayes formula to the situation where we want to fit a statistical model to observed data. The observations $X_1, X_2, \ldots$ are random variables of a model which is denoted $X \sim f(.|\theta)$. This means that

the distribution of X is determined by $f(.|\theta)$ where θ is a parameter. The way we describe the distribution of X through f is not important and can be either the density or the probability distribution function. The parameter θ can be interpreted as the unknown part of the model. In Bayesian statistics it is therefore modelled as a random variable with a prior distribution $\pi(\theta)$. After observing $X = (X_1, X_2, \ldots, X_n)$, the density of X is the likelihood function

$$l(\theta) = l(\theta; x_1, \ldots, x_n) = f(x_1, \ldots, x_n|\theta).$$

For independent and identically distributed (iid) samples X_i, the likelihood is $l(\theta; x_1 \cdots, x_n) = \prod_{i=1}^{n} f(x_i|\theta)$. The Bayes theorem describes the change of knowledge of the parameter θ through the observation X.

Theorem 6 (Bayes theorem)**.** *Let $\pi(\theta)$ be the density of the parameter θ on $\Theta \subseteq \mathbb{R}^r$ and $f(x|\theta)$ the density of the real random variable X given the parameter θ. Then the conditional density of the parameter with respect to X (e.g. $\theta|X$) is*

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{\int_\Theta f(x|\theta)\pi(\theta)d\theta}. \tag{1.3}$$

$\pi(\theta)$ is called the prior density and $\pi(\theta|x)$ is called the posterior density.

Omitting the integral in the denominator, we can write Bayes theorem as

$$\pi(\theta|D) \propto l(\theta; D)\pi(\theta).$$

1.4 Conjugate prior distributions

From Bayes theorem we can see that the structure of the posterior distribution depends soley on the likelihood and the prior. In the fortunate case where the likelihood and the prior are similar from the viewpoint of the parameter, we can expect the posterior to be of the same structure as well. If the prior and posterior belong to the same family we call them *conjugate* to the model.

Definition 5. *A family of distributions $\Pi = \{\pi_\psi : \psi \in \Psi\}$ for the parameter θ with the set of hyperparameters Ψ is* conjugate *to the family $\mathcal{P} = \{P_\theta : \theta \in \Theta\}$, iff there exists a $\psi_x \in \Psi$ such that $\theta|X = x \sim \pi_{\psi_x} \in \Pi$ for each $x \in \Omega$ and $\pi_\psi \in \Pi$.*

With conjugate priors the Bayesian updating procedure is reduced to finding the hyperparameter of the posterior.

Example 1. *The iid data $X_1, \ldots, X_n$ is Bernoulli distributed $X_i \sim A_\theta$ with parameter $\theta \in [0, 1]$. The likelihood is*

$$l(\theta) = \theta^{\sum_i x_i}(1-\theta)^{n-\sum_i x_i}.$$

A conjugate family of distributions is the beta family $\Pi = \{\beta(a, b) : a, b > 0\}$ with densities

$$\pi(\theta) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}.$$

Bayesian updating yields the posterior

$$\begin{aligned}\pi(\theta|x_1,\ldots,x_n) &\propto \theta^{\sum_i x_i}(1-\theta)^{n-\sum_i x_i}\theta^{a-1}(1-\theta)^{b-1}\\ &= \theta^{a+\sum_i x_i-1}(1-\theta)^{b+n-\sum_i x_i-1}\end{aligned}$$

which is a $\beta(a+\sum_i x_i, b+n-\sum_i x_i)$ *distribution. In this example the prior hyperparameter* (a,b) *is updated to the posterior hyperparameter* $(a+\sum_i x_i, b+n-\sum_i x_i)$.

We will often deal with distributions from the exponential family which have densities of the form

$$f_\theta(x) = C(\theta)h(x)\exp\left(\sum_{i=1}^{k} Q_i(\theta)T_i(x)\right). \tag{1.4}$$

Examples of such distributions are the normal, poisson, exponential and binomial distributions. For distributions of the exponential family we can always find a family of conjugate distributions. This follows from the next, more general result.

Theorem 7. *Let* $X_i \sim P_\theta$ *such that a sufficient statistic* T *with fixed dimension exists. Then there exists a conjugate prior family of distributions for* θ.

For the exponential family of distributions the sufficient statistics are the functions T_i, $i=1,\ldots,k$ so the theorem above can be applied.

We end this section with a discussion of the normal family of distributions and their conjugate family.

Let $X_i \sim N(\mu,\sigma_0^2)$ with known variance σ_0^2. The likelihood is

$$\begin{aligned}l(\mu;x_1,\ldots,x_n) &\propto \exp\left(-\frac{1}{2\sigma_0^2}\sum_{i=1}^{n}(x_i-\mu)^2\right)\\ &\propto \exp\left(-\frac{1}{2}\left(\frac{\mu-\overline{x}_n}{\frac{\sigma_0}{\sqrt{n}}}\right)^2\right),\end{aligned}$$

which is a normal distribution $N\left(\overline{x}_n, \frac{\sigma_0^2}{n}\right)$. A conjugate prior family for the parameter μ is the normal family of distributions $\{N(m,d) : m\in\mathbb{R}, d\in\mathbb{R}^+\}$. By using the prior $\mu\sim N(m,d)$ and calculating

$$\begin{aligned}\pi(\mu|x_1,\ldots,x_n) &\propto \exp\left(-\frac{1}{2\sigma_1^2}(\mu-\overline{x}_n)^2\right)\exp\left(-\frac{1}{2d^2}(\mu-m)^2\right)\\ &= \exp\left(-\frac{1}{2}\left(\frac{\mu^2-2\mu\overline{x}_n+\overline{x}_n^2}{\sigma_1^2}+\frac{\mu^2-2\mu m+m^2}{d^2}\right)\right)\\ &\ldots\\ &= \exp\left(-\frac{1}{2}\frac{\left(\mu-\frac{\sigma_1^2 m+d^2\overline{x}_n}{\sigma_1^2+d^2}\right)^2}{\frac{\sigma_1^2 d^2}{\sigma_1^2+d^2}}\right)\end{aligned}$$

it is easy to see that the new hyperparameter for μ is $(m^*, {d^*}^2)$ with

$$m^* = \frac{\sigma_1^2 m + d^2\overline{x}_n}{\sigma_1^2+d^2} \tag{1.5}$$

$${d^*}^2 = \frac{\sigma_1^2 d^2}{\sigma_1^2+d^2} \tag{1.6}$$

where $\sigma_1^2 = \frac{\sigma_0^2}{n}$.

When the mean is known ($X_i \sim N(\mu_0, \sigma^2)$) we substitute the variance σ^2 with the precision $\tau = \frac{1}{\sigma^2}$. A simple calculation yields the gamma distribution $\gamma(a, b)$, $a, b > 0$ with density

$$\pi(\tau) = \frac{b^a}{\Gamma(a)} \tau^{a-1} e^{-b\tau} \tag{1.7}$$

as conjugate distribution with the posterior hyperparameter (a^*, b^*) with

$$\begin{aligned} a^* &= a + \frac{n}{2} \\ b^* &= b + \frac{1}{2} \sum_{i=1}^{n} (x_i - \mu_0)^2. \end{aligned}$$

Finally, when both the mean and the precision are unknown and therefore parameters, the 2-dimensional normal-gamma family $\{NG(m, d^2, a, b) : m \in \mathbb{R},\ d^2, a, b \in \mathbb{R}^+\}$ is a conjugate family of distributions. (X, Y) is distributed after $NG(m, d^2, a, b)$, when

$$\begin{aligned} X|Y = y &\sim N\left(m, \frac{d^2}{y}\right) \quad \text{and} \\ Y &\sim \gamma(a, b). \end{aligned}$$

The prior density for $(\mu, \tau) \sim NG(m, d^2, a, b)$ is

$$\pi(\mu, \tau) = \frac{b^a}{\Gamma(a)} \tau^{a-1} e^{-b\tau} \frac{\sqrt{\tau}}{\sqrt{2\pi d^2}} \exp\left(-\frac{1}{2} \frac{\tau}{d^2} (\mu - m)^2\right) \tag{1.8}$$

and the posterior hyperparameter (m^*, d^{*2}, a^*, b^*) is

$$\begin{aligned} m^* &= \frac{m + d^2 n \overline{x}_n}{d^2 n + 1} \\ d^{*2} &= \frac{d^2}{d^2 n + 1} \\ a^* &= a + \frac{n}{2} \\ b^* &= b + \frac{1}{2} \left(\frac{n(\overline{x}_n - m)^2}{d^2 n + 1} + \sum_i (x_i - \overline{x}_n)^2 \right). \end{aligned}$$

1.5 Stochastic processes and time series

In this section we give an overview of stochastic processes and time series which are concrete realizations of an underlying stochastic process.

The underlying stochastic process of a time series is often unknown and therefore we have to approximize it by fitting an appropriate model to the observed time series data (inverse problem). Some simple time series models are reviewed at the end of this section.

Definition 6. *Let $(\Omega, \mathcal{F}, P)$ be a probability space and T a nonempty set called the* index set*. For all $t \in T$ let*

$$X_t : (\Omega, \mathcal{F}, P) \to (\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n)) \tag{1.9}$$

be a random variable. The family $\mathcal{X} = (X_t)_{t \in T}$ is called a stochastic process *on $(\Omega, \mathcal{F}, P)$.*

Remark 2. *To address realizations of stochastic processes as time series we require the index set T to exhibit features of time (e.g. causation). Typical examples of T are $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $[0, \infty)$.*

Definition 7. *Let $\mathcal{X} = (X_t)_{t \in T}$ be a stochastic process and $\omega \in \Omega$. The mapping*

$$t \mapsto X_t(\omega)$$

is called a trajectory *of the stochastic process.*

Definition 8. *Let $\mathcal{X} = (X_t)_{t \in T}$ be a stochastic process with $\mathbb{E}|X_t| < \infty$ and $\mathbb{E}(X_t^\top X_t) < \infty$ for all $t \in T$. The mapping*

$$t \mapsto \mathbb{E}X_t$$

is called the mean function *and*

$$\gamma : T \times T \to \mathbb{R}^{n \times n} : (s, t) \mapsto \gamma(s, t) = \mathbb{E}(X_s - \mathbb{E}X_s)(X_t - \mathbb{E}X_t)^\top$$

is called the covariance function*.*

Definition 9. *Let $\mathcal{X} = (X_t)_{t \in T}$ be a stochastic process and $t_1, \ldots, t_k \in T$. If the random vectors $(X_{t_1}, \ldots, X_{t_k})$ and $(X_{t_1+u}, \ldots, X_{t_k+u})$ have the same distribution for each u, the stochastic process $\mathcal{X}$ is called* strong stationary*.*

Definition 10. *A stochastic process $\mathcal{X} = (X_t)_{t \in T}$ is called* weak stationary*, iff*

1. $\mathbb{E}(X_t^\top X_t) < \infty \quad \forall t \in T$
2. $\mathbb{E}X_t = \mu \in \mathbb{R}^n \quad \forall t \in T$
3. $\gamma(s, t) = \gamma(s + r, t + r) \quad \forall r, s, t \in T.$

Remark 3. *For a (weak) stationary process the covariance function γ only depends on the time difference $t - s$. In such a case we write $\gamma(s) = \gamma(s, 0)$.*

We now review some important stochastic processes which are often used to model existing time series. Unless stated otherwise, the index set of the processes is $\mathbb{Z}$.

1.5.1 White noise process

The *white noise process* $(\epsilon_t)_{t \in \mathbb{Z}}$ is the simplest case of a stochastic process. It is defined by

1. $\mathbb{E}\epsilon_t = 0 \quad \forall t \in \mathbb{Z}$
2. $\mathbb{E}(\epsilon_s \epsilon_t^\top) = \delta_{st}\Sigma \quad \forall s, t \in \mathbb{Z}$

where $\Sigma \in \mathbb{R}^{n \times n}$ is a non-negative definite matrix and the ϵ_t are $\mathbb{R}^n$-valued random variables. With δ_{st} we denote the Dirac-delta.

The white noise process is stationary, both in the strong and the weak sense. It is often used as the unexplainable residual component in a time series model.

1.5.2 Autoregressive process

An *autoregregressive process* $\mathcal{X} = (X_t)_{t\in\mathbb{Z}}$ of order $p \geq 1$ depends on its own past. At time $t \in \mathbb{Z}$ the process is defined through

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + \epsilon_t \tag{1.10}$$

where $a_i \in \mathbb{R}^{n\times n}$ ($a_p \neq 0$) and $(\epsilon_t)_{t\in\mathbb{Z}}$ is a white noise process. The matrix coefficients a_i determine the dependence of the process on its own past and the ϵ_t's represent the "noise" in each time step.

1.5.3 Moving average process

Let $(\epsilon_t)_{t\in\mathbb{Z}}$ be a white noise process and $\mathcal{X} = (X_t)_{t\in\mathbb{Z}}$ be defined by

$$X_t = \sum_{j=0}^{q} b_j \epsilon_{t-j} \tag{1.11}$$

with $b_j \in \mathbb{R}^{n\times n}$ for all $t \in \mathbb{Z}$. $\mathcal{X}$ is called a *moving average process* of order q ($b_0 \neq 0$ and $b_q \neq 0$). A simple calculation yields

1. $\mathbb{E}(X_t^\top X_t) < \infty \quad \forall t \in \mathbb{Z}$,
2. $\mathbb{E}X_t = 0 \quad \forall t \in \mathbb{Z}$ and
3. $\gamma(s,t) = \begin{cases} \sum_{i=\max\{0,t-s\}}^{\min\{q,q-(t-s)\}} b_i^\top \Sigma b_{t-s+i} & \text{for } |s-t| \leq q \\ 0 & \text{for } |s-t| > q \end{cases}$.

Thus a moving average process is (weak) stationary.

1.5.4 Wiener process

The *Wiener process*, named after Norbert Wiener, is a continuous time ($T = [0,\infty)$) stochastic process. $\mathcal{W} = (W_t)_{t\geq 0}$ is a Wiener process, iff

1. $W_0 = 0 \quad a.s.$
2. $t \mapsto W_t$ is continuous $a.s.$
3. The increments $W_t - W_s$ ($0 \leq s < t$) are independent and $W_t - W_s \sim N(0, t-s)$.

The *Brownian motion* $(B_t)_{t\in T}$ is a prominent example of a Wiener process which describes the random movement of particles in a fluid. A Brownian motion on $T = [0,1]$ with the extra condition $B_1 = 0$ is called a *Brownian bridge*.

Chapter 2

The Dynamic Linear Model

A simple, yet flexible model for time series data is the *dynamic linear model (DLM)*. Pole, West and Harrison describe the DLM in a Bayesian analysis context in [13] and in this chapter we will discuss the DLM as a Bayesian time series model.

The model assumes a normal distribution for all random variables, which means that, because normally distributed random variables are closed under linear transformations, all distributions remain normal. Therefore the model is analytically tractable and we only have to calculate the posteriori parameters.

In Section 2.1 we introduce the structure of the DLM and discuss it in a Bayesian context with a more detailed analysis in sections 2.2, 2.3 and 2.5. The DLM is capable of modelling structural components in time series, like trends, seasonal components or regression components, which we present in section 2.4. In aection 2.7 we apply the DLM to model a real dataset.

2.1 Model structure

The DLM enhances the linear (regression) model through a time dependent parameter with its own dynamic, the *system equation*. Structurally, it can be interpreted as a *hidden Markov model* where the state corresponds to the parameter in the DLM. The observation is modelled through the *observation equation* which depends on the (hidden) parameter of the model. Put together, the DLM is written as

$$Y_t = F_t^\top \theta_t + \nu_t, \quad \nu_t \sim N(0, V_t) \tag{2.1}$$

$$\theta_t = G\theta_{t-1} + \omega_t, \quad \omega_t \sim N(0, W_t) \tag{2.2}$$

where (2.1) is the observation equation and (2.2) is the system equation, Y_t denoting the observation, F_t the (regression) coefficients, G the matrix describing the system dynamics and ν_t and ω_t normally distributed error variables with known variances V_t and W_t at time t respectively. For the DLM we choose the index set $T = \mathbb{N}$. The model parameter at time t is denoted θ_t. We concentrate on the univariate case in with $Y_t \in \mathbb{R}$ and $F_t \in \mathbb{R}^n$. We further assume that the random variables, describing the errors, ν_t and ω_t are mutually and temporally

independent

$$\begin{aligned} cov(\nu_s, \nu_t) &= 0 \quad \forall s,t \in \mathbb{N} : s \neq t \\ cov(\omega_s, \omega_t) &= 0 \quad \forall s,t \in \mathbb{N} : s \neq t \\ cov(\nu_s, \omega_t) &= 0 \quad \forall s,t \in \mathbb{N}. \end{aligned}$$

We also assume the error variable ν_t of the observation equation to be independent of the state variable θ_t (e.g. $cov(\theta_t, \nu_t) = 0 \quad \forall t \in \mathbb{N}$). That G, the coefficient of the system equation, is constant over time, is a restriction that can be relaxed to a time dependent coefficient G_t.

The Bayesian perspective in the DLM concentrates on the parameter θ_t. In each time step t we have to consider realized observations in the past $y_0, \ldots, y_{t-1}$ and the and the prior distribution of the parameter θ_t. We obtain the prior for θ_t through the posterior for θ_{t-1}, moved forward to time t by the system equation.

In the next section we look at this prior to posterior analysis more closely.

2.2 Bayesian Updating in the DLM

Using the normal distribution as prior and the assumption that all the random error variables are also normal makes the Bayesian updating process easy insofar, that we only have to calculate the posterior mean and variance, as the distribution stays normal. Here we will describe the cycle from prior to forecast to posterior to next prior in more detail as in [13]. We start the cycle at time $t+1$ with the prior distribution $\theta_{t+1}|D_t \sim N(a_{t+1}, R_{t+1})$. With D_t we denote our knowledge at time t which in our case consists of the observations $y_1, \ldots, y_t$. From this prior we can generate forecasts for future observations through the observation equation. First we consider the *one-step-ahead* forecast of Y_{t+1}. We have

$$Y_{t+1} = F_{t+1}^\top \theta_{t+1} + \nu_{t+1}$$

which is a linear combination of normally distributed random variables (θ_{t+1} and ν_{t+1}) and is therefore also normal. Thus the one-step-ahead forecast distribution $Y_{t+1}|D_t$ has mean

$$\begin{aligned} f_{t+1} := \mathbb{E}(Y_{t+1}|D_t) &= \mathbb{E}(F_{t+1}^\top \theta_{t+1} + \nu_{t+1}|D_t) \\ &= \mathbb{E}(F_{t+1}^\top \theta_{t+1}|D_t) + \mathbb{E}(\nu_{t+1}|D_t) \\ &= F_{t+1}^\top \mathbb{E}(\theta_{t+1}|D_t) + \mathbb{E}(\nu_{t+1}) \\ &= F_{t+1}^\top a_{t+1} \end{aligned}$$

and variance

$$\begin{aligned} Q_{t+1} := var(Y_{t+1}|D_t) &= var(F_{t+1}^\top \theta_{t+1} + \nu_{t+1}|D_t) \\ &= var(F_{t+1}^\top \theta_{t+1}|D_t) + var(\nu_{t+1}|D_t) \\ &= F_{t+1}^\top var(\theta_{t+1}|D_t) F_{t+1} + var(\nu_{t+1}) \\ &= F_{t+1}^\top R_{t+1} F_{t+1} + V_{t+1}, \end{aligned}$$

so $Y_{t+1} \sim N(f_{t+1}, Q_{t+1})$. In this calculation we assumed that the error variable ν_{t+1} is independent of the parameter θ_{t+1}.

For a *k-step-ahead* forecast of Y_{t+k} we have to project the prior distribution $\theta_{t+1}|D_t$ into the future to obtain $\theta_{t+k}|D_t$. This is accomplished through the repeated application of the system equation. At time $t+2$ we obtain this *implied* prior from

$$\theta_{t+2} = G\theta_{t+1} + \omega_{t+2},$$

which again is a linear combination of normally distributed random variables. The mean and the covariance matrix of $\theta_{t+2}|D_t$ are

$$\begin{aligned}\mathbb{E}(\theta_{t+2}|D_t) &= \mathbb{E}(G\theta_{t+1}|D_t) + \mathbb{E}(\omega_{t+2}|D_t)\\ &= G\mathbb{E}(\theta_{t+1}|D_t) + \mathbb{E}(\omega_{t+2})\\ &= Ga_{t+1}\end{aligned}$$

$$\begin{aligned}var(\theta_{t+2}|D_t) &= var(G\theta_{t+1}|D_t) + var(\omega_{t+2}|D_t)\\ &= Gvar(\theta_{t+1}|D_t)G^\top + var(\omega_{t+2})\\ &= GR_{t+1}G^\top + W_{t+2}.\end{aligned}$$

Repeated application of the system equation yields the *k-step-ahead forecast prior*

$$\theta_{t+k}|D_t \sim N(a_t(k), R_t(k))$$

with

$$a_t(k) = G^{k-1}a_{t+1}$$

and

$$R_t(k) = G^{k-1}R_{t+1}{G^{k-1}}^\top + \sum_{j=2}^{k} G^{k-j}W_{t+j}{G^{k-j}}^\top.$$

After observing the quantity y_t at time t we turn our attention to the Bayesian updating step through Bayes theorem, yielding the posterior distribution $\theta_t|D_{t-1}, y_t$. Besides the prior distribution $\theta_t|D_{t-1}$ we need the likelihood of the observation $Y_t|\theta_t, V_t$ which we obtain through the observation equation

$$Y_t = F_t^\top\theta_t + \nu_t, \quad \nu_t \sim N(0, V_t).$$

Therefore $Y_t|\theta_t, V_t \sim N(F_t^\top\theta_t, V_t)$ and we write the likelihood of the observation y_t as $p(Y_t = y_t|\theta_t, V_t)$. As in [13], p denotes a general probability density function of a random variable. Using Bayes theorem we get the posterior density

$$\begin{aligned}p(\theta_t|D_{t-1}, y_t) &= \frac{p(Y_t = y_t|\theta_t, V_t)p(\theta_t|D_{t-1})}{p(Y_t = y_t)}\\ &\propto p(Y_t = y_t|\theta_t, V_t)p(\theta_t|D_{t-1}).\end{aligned}$$

Both the likelihood and the prior are normally distributed, yielding the posterior density

$$\begin{aligned}p(\theta_t|D_{t-1}, y_t) &\propto \exp\left(-\frac{1}{2}V_t^{-1}(y_t - F_t^\top\theta_t)^2\right)\\ &\quad \exp\left(-\frac{1}{2}(\theta_t - a_t)^\top R_t^{-1}(\theta_t - a_t)\right)\\ &\propto \exp\left(-\frac{1}{2}(\theta_t - m_t)^\top C_t^{-1}(\theta_t - m_t)\right)\end{aligned}$$

which in turn is a normal distribution with

$$\begin{aligned} m_t &= a_t + A_t e_t, \\ C_t &= R_t - A_t A_t^\top Q_t, \\ A_t &= \frac{R_t F_t}{Q_t}, \\ e_t &= y_t - f_t. \end{aligned}$$

Heuristically, we update the prior mean a_t with a multiple of the forecat error e_t to obtain the posterior mean m_t. The posterior variance C_t is in general smaller than the prior variance because of the additional information in the observation y_t.

In the last step of the cycle we use the posterior distribution at time $t-1$ to obtain the prior distribution for time t. This is achieved by applying the posterior $\theta_{t-1}|D_{t-1}$ to the system equation

$$\theta_t = G\theta_{t-1} + \omega_t, \quad \omega_t \sim N(0, W_t).$$

Again, this is a linear combination of normally distributed random variables. Thus, we find the normally distributed prior for time t as

$$\theta_t \sim N(a_t, R_t)$$

with mean

$$a_t = Gm_{t-1}$$

and variance

$$R_t = GC_{t-1}G^\top + W_t.$$

This concludes the cycle of Bayesian updating in the DLM.

2.3 Smoothing

The activity in which we use all the available time series data $y_1, \ldots, y_\tau$ to fit the model retrospectively is called smoothing. In the DLM this corresponds to finding the filtered distributions $\theta_t|D_\tau$, $t = 1, \ldots, \tau$. This is achieved by filtering future information back to time t using Bayes theorem.

In this section we analyze the filtering of information back on time step from time t to $t-1$ as described in [13]. The general result for filtering back over k time steps can be found in [17].

From the Bayesian updating analysis, outlined in the previous section, we have the following prior and posterior distributions

$$\begin{aligned} \theta_{t-1}|D_{t-2} &\sim N(a_{t-1}, R_{t-1}), \\ \theta_{t-1}|D_{t-1} &\sim N(m_{t-1}, C_{t-1}), \\ \theta_t|D_{t-1} &\sim N(a_t, R_t), \\ \theta_t|D_t &\sim N(m_t, C_t) \end{aligned}$$

and we want to calculate the retrospective parameter distribution $\theta_{t-1}|D_t$. We can express the probability density function $p(\theta_{t-1}|D_t)$ through the law of total probability as

$$p(\theta_{t-1}|D_t) = \int p(\theta_{t-1}|\theta_t, D_t) p(\theta_t|D_t) d\theta_t.$$

The second term in the integrand is the posterior density at time t which is known. To obtain the first term in the integrand we split $D_t = \{D_{t-1}, y_t\}$ into the information up to time $t-1$ and the observation y_t, thus $p(\theta_{t-1}|\theta_t, D_t) = p(\theta_{t-1}|\theta_t, D_{t-1}, y_t)$. Applying Bayes theorem we get

$$p(\theta_{t-1}|\theta_t, D_{t-1}, y_t) = \frac{p(y_t|\theta_{t-1}, \theta_t, D_{t-1})p(\theta_{t-1}|\theta_t, D_{t-1})}{p(y_t|\theta_t, D_{t-1})}. \tag{2.3}$$

From the observation equation

$$Y_t = F_t^\top \theta_t + \nu_t$$

it is clear that $Y_t|\theta_t$ is independent of θ_{t-1}, thus

$$p(y_t|\theta_{t-1}, \theta_t, D_{t-1}) = p(y_t|\theta_t, D_{t-1}).$$

From (2.3) we now have

$$p(\theta_{t-1}|\theta_t, D_{t-1}, y_t) = p(\theta_{t-1}|\theta_t, D_{t-1})$$

which means that the observation y_t carries no additional information for θ_{t-1} if we know the parameter θ_t. Finally we apply Bayes theorem one more time to get

$$p(\theta_{t-1}|\theta_t, D_{t-1}) = \frac{p(\theta_t|\theta_{t-1}, D_{t-1})p(\theta_{t-1}|D_{t-1})}{p(\theta_t|D_{t-1})}$$

where all the densities on the right hand side are known:

$$\begin{aligned} \theta_t|\theta_{t-1}, D_{t-1} &\sim N(G\theta_{t-1}, W_t), \\ \theta_{t-1}|D_{t-1} &\sim N(m_{t-1}, C_{t-1}), \\ \theta_t|D_{t-1} &\sim N(a_t, R_t). \end{aligned}$$

From this we obtain that $\theta_{t-1}|\theta_t, D_{t-1}$ is normally distributed

$$\theta_{t-1}|\theta_t, D_{t-1} \sim N(h_t(1), H_t(1))$$

with parameters

$$\begin{aligned} h_t(1) &= m_{t-1} + B_{t-1}(\theta_t - a_t), \\ H_t(1) &= C_{t-1} - B_{t-1}R_tB_{t-1}^\top \end{aligned}$$

where

$$B_t = C_tG^\top R_{t-1}^{-1}.$$

We now know that all the terms of the integrand in (2.3) are normal distribution. With the theory of normally distributed random variables we can now calculate the distribution of $\theta_{t-1}|D_t$ we are seeking:

$$\theta_{t-1}|D_t \sim N(a_t(-1), R_t(-1))$$

with

$$a_t(-1) = m_t + B_{t-1}(m_t - a_t)$$

and

$$R_t(-1) = C_{t-1} - B_{t-1}(R_t - C_t)B_{t-1}^\top.$$

2.4 Modelling structural components

Time series usually comprise some structural components like trends, where the average level of the data changes with time according to some law. Another common component is a seasonal pattern. Here the average level shows a periodic behaviour. A time series can be influenced by a different (exogenous) time series, in which case the original series can be modeled as a regression on the exogenous components with time varying regression coefficients. Finally, a time series can be made up of several different components, for example a time series with seasonal fluctuations and a genereal upward sloping trend.

We will introduce how to model these components in this section.

2.4.1 Trend components

A trend determines the average level of a time series with a dependence on the time. The observation equation in this case simply is

$$Y_t = \mu_t + \nu_t, \quad \nu_t \sim N(0, V_t)$$

where μ_t is the trend level at time t and ν_t the noise around that level.

The simplest case is a constant, noisy trend level in which case the system equation is

$$\mu_t = \mu_{t-1} + \omega_t, \quad \omega_t \sim N(0, W_t).$$

A trend where the level changes with constant increments (linear trend) can be modelled with the system equation

$$\mu_t = \mu_{t-1} + c + \omega_t$$

where c is the deterministic increment in the trend level between two time steps. Should the increment also be time dependent, we have to increase the dimension of the system equation and add the dynamics for the increment c_t to it. A random walk increment, for example, can be written as

$$\begin{aligned} \mu_t &= \mu_{t-1} + c_{t-1} + \omega_{1t} \\ c_t &= c_{t-1} + \omega_{2t} \end{aligned}$$

with $\theta_t = (\mu_t, c_t)^\top$. The observation equation has to be changed to

$$Y_t = F^\top \theta_t + \nu_t$$

with $F = (1, 0)^\top$. This technique allows the construction of more complicated forms of trends.

2.4.2 Seasonal components

Periodic fluctuations of a time series can be modelled through a seasonal component in the DLM.

One way to describe seasonal components is by assigning seasonal effect variables to each season. For a quarterly sampled time series with an annual cycle we need four effect variables

$$\theta_t = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix}$$

which rotate as time moves forward, thus at $t+1$ we have the parameter

$$\theta_{t+1} = \begin{pmatrix} q_4 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}.$$

This can be modelled through the system equation with coefficient matrix

$$G = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

In the observation equation we simply take the first effect variable off the parameter vector, accomplished by setting

$$F = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The problem with this modelling technique is, that the system equation can get high in dimension quickly.

However this problem can be avoided by using periodic functions like sin and cos to describe the periodic fluctuations directly. This way we only need a two-dimensional system equation with coefficient matrix

$$G = \begin{pmatrix} \cos\omega & \sin\omega \\ -\sin\omega & \cos\omega \end{pmatrix}$$

where ω denotes the frequency of the seasonal component, with the period given by $\frac{2\pi}{\omega}$. For the quarterly example above we would set $\omega = \frac{\pi}{2}$. Generally for a cycle of length (period) p we need to set $\omega = \frac{2\pi}{p}$. The observation equation coefficient vector is

$$F = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The parameter $\theta_t = (a_t, b_t)^\top$ can often be difficult to interpret and it would be desireable to transform it to a vector of seasonal effect variables ϕ_t that we introduced above. Fortunately this can be achieved through the linear transformation $\phi_t = L\theta_t$ defined by

$$L = \begin{pmatrix} F^\top \\ F^\top G \\ \vdots \\ F^\top G^{p-1} \end{pmatrix} \in \mathbb{R}^{p \times 2}$$

where p is the period of the seasonal component. Note that $G = G^p$, therefore the seasonal effects also have period p.

2.4.3 Regression components

A regression on an exogenous time series $(x_{1t}, x_{2t}, \ldots, x_{qt})^\top_{t\in\mathbb{N}}$ is easily implemented in the DLM by setting

$$F_t = \begin{pmatrix} x_{1t} \\ x_{2t} \\ \vdots \\ x_{qt} \end{pmatrix}$$

and modelling the regression coefficients (the model parameter)

$$\theta_t = \begin{pmatrix} \beta_{1t} \\ \beta_{2t} \\ \vdots \\ \beta_{qt} \end{pmatrix}$$

with the system equation

$$\theta_t = G\theta_{t-1} + \omega_t$$

where $G = I$ (the identity matrix) and $\omega_t \sim N(0, W_t)$.

2.4.4 Combining structural components

If a time series exhibits several structural components we can superposition them together into one model by assembling the system equation in a block-structure where each block corresponds to an independent structural component of the time series.

We exemplify this approach by modelling a time series with a regression component on $(x_{1t}, x_{2t})^\top_{t\in\mathbb{N}}$ and a seasonal component with period p. The parameter

$$\theta_t = \begin{pmatrix} \beta_{1t} \\ \beta_{2t} \\ a_t \\ b_t \end{pmatrix}$$

consists of the regression coefficients β_{1t} and β_{2t} and the seasonal parameters a_t and b_t. The system coefficient matrix has the form

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\frac{2\pi}{p}) & \sin(\frac{2\pi}{p}) \\ 0 & 0 & -\sin(\frac{2\pi}{p}) & \cos(\frac{2\pi}{p}) \end{pmatrix}$$

and the coefficient vector of the observation equation is

$$F = \begin{pmatrix} x_{1t} \\ x_{2t} \\ 1 \\ 0 \end{pmatrix}.$$

This example shows the ease with which we can model more complicated structural patterns for time series, simply by combining basic components to a block structured model. It would be trivial to add a trend component to the example above.

2.5 Unknown Observation Variance

So far we have assumed that the observation variances V_t are known. This is an unrealistic assumption most of the time. In this section we summarize a simple generalization of the DLM for unknown variance, which is outlined in more detail in [13].

We assume the variance constant over time $V_t \equiv V$ and add the precision $\phi = V^{-1}$ as unknown Bayesian parameter to the DLM which in this case is defined by

$$\begin{aligned} Y_t &= F_t^\top \theta_t + \nu_t, \quad \nu_t \sim N(0, \phi^{-1}), \\ \theta_t &= G\theta_{t-1} + \omega_t, \quad \omega_t \sim N(0, W_t^* \phi^{-1}), \end{aligned}$$

the observation equation and the system equation respectively. The covariance matrix of the system equation is scaled with the observation equation error variance. The reason for this is that the model prior remains conjugate. We can recover the original covariance matrix by setting $W_t = W_t^* \phi^{-1}$.

The gamma distribution is a conjugate prior for the precision of a normally distributed random variable. Therefore the parameter prior for $(\theta_t, \phi)^\top$ at time t consists of

$$\begin{aligned} \theta_t | D_{t-1}, \phi &\sim N\left(a_t, R_t^* \phi^{-1}\right), \\ \phi | D_{t-1} &\sim \gamma\left(\frac{n_{t-1}}{2}, \frac{d_{t-1}}{2}\right), \end{aligned}$$

with the mean of the prior for the precision $\frac{n_{t-1}}{d_{t-1}}$.

The conditional forecast distribution is similar to the one outlined in section 2.2

$$Y_t | D_{t-1}, \phi \sim N(F_t^\top a_t, \phi^{-1} Q_t^*)$$

with

$$Q_t^* = 1 + F_t^\top R_t^* F_t.$$

We get the unconditional forecast distribution by applying the law of total probability

$$p(y_t | D_{t-1}) = \int p(y_t | D_{t-1}, \phi) p(\phi | D_{t-1}) d\phi$$

which results in a t distribution with n_{t-1} degrees of freedom

$$Y_t | D_{t-1} \sim t_{n_{t-1}}(f_t, Q_t).$$

The parameters f_t and Q_t are given by

$$\begin{aligned} f_t &= F_t^\top a_t, \\ Q_t &= S_{t-1} + F_t^\top R_t S_{t-1} F_t \end{aligned}$$

with $S_{t-1} = \frac{d_{t-1}}{n_{t-1}}$.

Finding the posterior of the parameter $(\theta_t, \phi)^\top$ in the DLM with unknown variance is a two-step process. We start by calculating the posterior of the precision $\phi | D_{t-1}, y_t$ by applying Bayes theorem

$$p(\phi | D_{t-1}, y_t) \propto p(y_t | D_{t-1}, \phi) p(\phi | D_{t-1}).$$

The first density on the right hand side is for the conditional forecast distribution which we obtained above. Because the second density belongs to the prior for ϕ at time t we can calculate

$$\begin{aligned} p(\phi|D_{t-1}, y_t) &\propto (\phi^{-1}Q_t^*)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\phi^{-1}Q_t^*)^{-1}(y_t - f_t)^2\right) \\ &\quad \phi^{\frac{n_{t-1}}{2}-1} \exp\left(-\frac{d_{t-1}}{2}\phi\right) \\ &\propto \phi^{\frac{n_{t-1}-1}{2}} \exp\left(-\frac{\phi}{2}\left(\frac{e_t^2}{Q_t^*} + d_{t-1}\right)\right) \end{aligned}$$

which yields a gamma distribution

$$\phi|D_t \sim \gamma\left(\frac{n_t}{2}, \frac{d_t}{2}\right)$$

with the parameters given by

$$\begin{aligned} n_t &= n_{t-1} + 1 \\ d_t &= d_{t-1} + \frac{e_t^2}{Q_t^*}. \end{aligned}$$

To ascertain the posterior for θ_t we consider the joint distribution of $Y_t|D_{t-1}, \phi$ and $\theta_t|D_{t-1}, \phi$. Both distributions are normal so the joint distribution is also normal

$$\begin{pmatrix} Y_t \\ \theta_t \end{pmatrix} \Big| D_{t-1}, \phi \sim N\left(\begin{pmatrix} f_t \\ a_t \end{pmatrix}, \begin{pmatrix} Q_t^*\phi^{-1} & F_t^\top R_t^*\phi^{-1} \\ R_t^{*\top} F_t \phi^{-1} & R_t^*\phi^{-1} \end{pmatrix}\right).$$

From this joint distribution we obtain the posterior by conditioning on the observation, resulting in

$$\theta_t|D_t, \phi = \theta_t|D_{t-1}, y_t, \phi \sim N(m_t, C_t^*\phi^{-1})$$

with the mean

$$m_t = a_t + \frac{R_t^* F_t e_t}{Q_t^*}$$

and variance

$$C_t^*\phi^{-1} = R_t^*\phi^{-1} - \frac{R_t^* F_t F_t^\top R_t^* \phi^{-1}}{Q_t^*}.$$

For the posterior of θ_t, unconditional on the precision, we have to apply the law of total probability which yields

$$p(\theta_t|D_t) = \int p(\theta_t|D_t, \phi) p(\phi|D_t) d\phi.$$

The distribution of $\theta_t|D_t$ is a multivariate t-distribution with n_t degrees of freedom, mean m_t and scale parameter $C_t = C_t^* S_t = C_t^* \frac{d_t}{n_t}$.

2.6 DLM Representation for AR and MA Processes

Autoregressive (AR) (see section 1.5.2) and moving average (MA) (see section 1.5.3) processes are often used to model time series. Therefore, in this short section, we present possible representations for these processes in the DLM. For reasons of simplicity we consider univariate processes.

A possible, and very intuitive representation of an $AR(p)$ process $\mathcal{X} = (X_t)_{t\in\mathbb{Z}}$, defined through equation (1.10), is by choosing

$$F_t = \begin{pmatrix} x_{t-1} \\ x_{t-2} \\ \vdots \\ x_{t-p} \end{pmatrix},$$

$$\theta_t = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix},$$

$G = I_p$ and setting $\nu_t = \epsilon_t$ and $\omega_t \equiv 0$ for all $t \in \mathbb{Z}$. This representation is structurally similar to the representation of regression components in the DLM which we presented in subsection 2.4.3.

A $MA(q)$ process $\mathcal{X} = (X_t)_{t\in\mathbb{Z}}$, defined through equation (1.11) can be represented by the deterministic observation equation

$$X_t = F^\top \theta_t$$

with

$$F = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{q+1\times q+1}$$

and the system equation

$$\theta_{t+1} = G\theta_t + \omega_t$$

where

$$G = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix} \in \mathbb{R}^{q+1\times q+1}$$

and

$$\omega_t = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_q \end{pmatrix} \epsilon_t.$$

It is easy to see that repeatedly applying the system equation and then inserting into the observation equation yields the equation (1.11) for the process.

2.7 CO_2-Emission: An Example

To summarize the DLM an example is in place. In this section we model a famous time series, the atmospheric CO_2 concentration at Mauna Loa [6] which is measured in *parts per million (ppm)* and is included as dataset in `R`. We used `R` to perform the analysis and `python` to generate the graphics. The source code in `R` is listed in appendix A. This time series consits of monthly observations from the years 1959 to 1997. Figure 2.1 shows a plot of this time series.

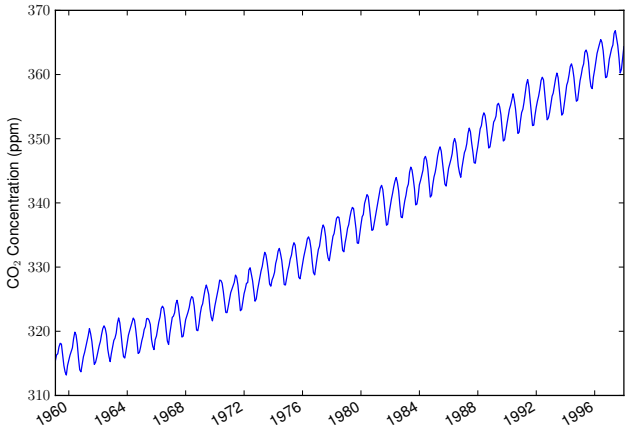


Figure 2.1: Atmospheric CO_2 Concentration at Mauna Loa

The concentration of CO_2 is gradually increasing over time with pronounced seasonal fluctuations. When we zoom in and look at the observations in the years 1990 and 1991 (Figure 2.2), we can see that the seasonal fluctuations seem to be an annual pattern. Indeed in figure 2.3 we can see a spike in the spectral density of the time series at the frequency 0.0833 which corresponds to a period of exactly 12.

Therefore our time series model will consist of a trend component, which we choose to be a polynomial of order 2 and a seasonal component with period 12.

From section 2.4, where we reviewed how to model various components with the DLM, we know that there are two ways to model seasonal components, either by assigning seasonal effects or by using trigonometric functions. In this example we choose the former alternative because it is more robust against imperfect fluctuations.

The basic DLM, introduced in section 2.1, assumes the observation variance and the system covariance matrix to be known, which is not the case in this example. To circumvent this problem we calculate the maximum likelihood estimates for the observation variance V and the diagonal elements of the system covariance matrix W and use them in the model. Alternatively we could add them as parameters to the model. However this would complicate the Bayesian anslysis to a point that is not desired in this example.

Before we can go through the Bayesian analysis of the model, outlined in

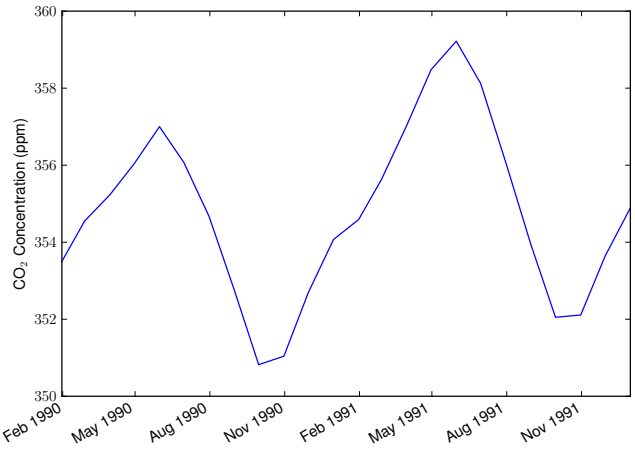


Figure 2.2: CO_2 Concentration in the years 1990 and 1991

section 2.2, we need to set an initial prior for the first time step. For the trend level e choose a normal distribution with mean 315 and variance 5. For the other parameters we assume approximately uninformative priors by using a normal distribution with mean 0 and variance 10^7.

Figures 2.4 and 2.5 show the time series with the one-step-ahead forecasts and the 95% confidence intervals.

The decomposition of the time series in the trend and seasonal component is shown in figure 2.6.

A prediction of the time series multiple steps into the future is shown in figure 2.7. Note the widening confidence intervals as we predict further ahead into the future.

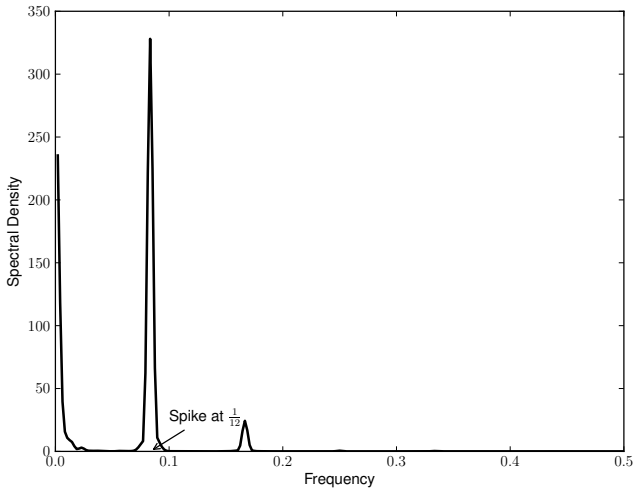


Figure 2.3: Spectral Density of the time series

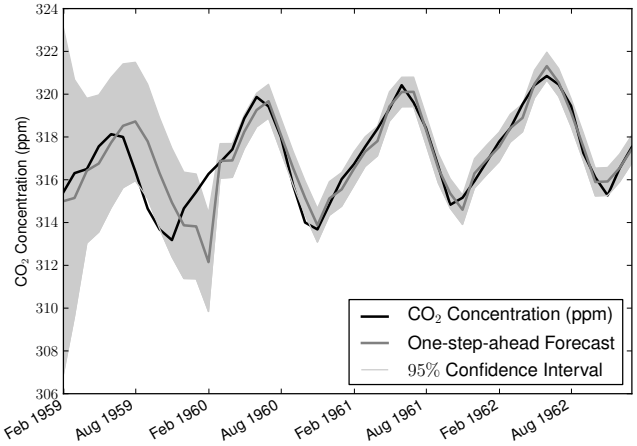


Figure 2.4: One-step-ahead forecast for the first four years

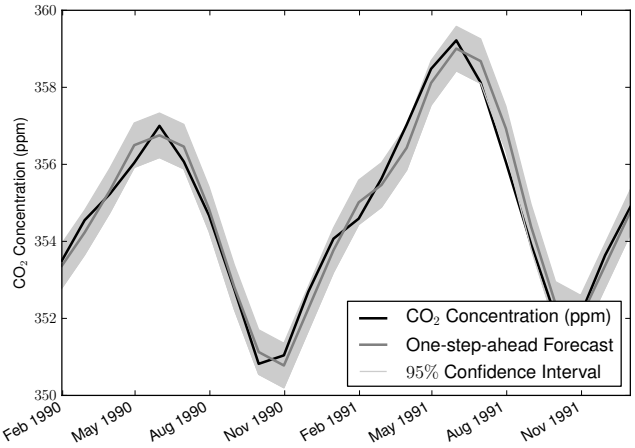


Figure 2.5: One-step-ahead forecast for the years 1990 and 1991

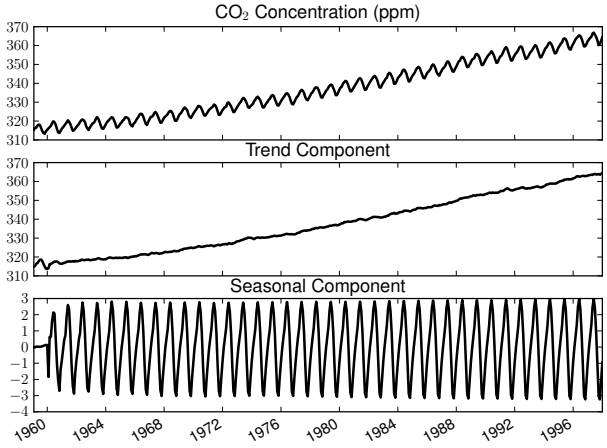


Figure 2.6: Decompositon of the time series in the trend component and seasonal component

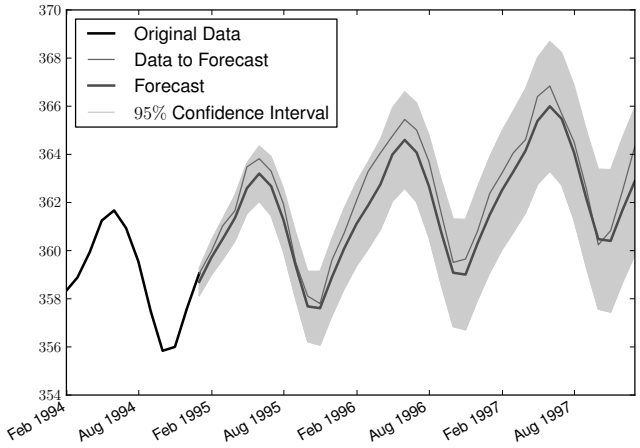


Figure 2.7: Forecast for the years 1995, 1996 and 1997

Chapter 3

Monte Carlo Methods

This Chapter serves as an introduction to Monte Carlo integration methods, Markov chain theory and Markov chain Monte Carlo (MCMC) methods, summarizing the books [14] and [15] by Robert and Casella. The methodological framework of Monte Carlo simulation methods is an important tool to numerically analyze more complicated statistical models that are analytically not tractable.

In the next chapter 4 we will analyze Bayesian time series with non-conjugate priors. Bayes theorem states that calculating the posterior distribution includes evaluating the integral $\int f(x|\theta)\pi(\theta)d\theta$, which usually is difficult and often impossible analytically. However with Monte Carlo integration methods this integral can be approximated numerically.

In section 3.1 we introduce *Monte Carlo integration* methods which can be used to approximately evaluate integrals similar to the one in the example above. The theory of *Markov chains*, fsummarized in section 3.2, is the theoretical foundation for sampling algorithms like the *Metropolis-Hastings algorithm* or the *Gibbs sampler* which we introduce in sections 3.3 and 3.4 respectivelty.

3.1 Monte Carlo Integration

Monte Carlo integration methods attempt to find an approximation for the integral

$$\mathbb{E}_f(h(X)) = \int h(x)f(x)dx. \tag{3.1}$$

The classical approach is to generate a sample $x_1, \ldots, x_n$ from the distribution f and approximate 3.1 by the arithmetic mean

$$\overline{h_n} = \frac{1}{n}\sum_{i=1}^{n} h(x_i). \tag{3.2}$$

By the *Law of Large Numbers*, $\overline{h_n}$ converges to $\mathbb{E}_f(h(X))$ almost surely. If $\mathbb{E}_f(h^2(X))$ is finite we can even estimate the variance of (3.2)

$$var(\overline{h_n}) = \frac{1}{n}\int (h(x) - \mathbb{E}_f(h(X)))^2 f(x)dx$$

through

$$v_n = \frac{1}{n^2}\sum_{i=1}^{n}(h(x_i) - \overline{h_n})^2.$$

Further, the *Central Limit Theorem* states that

$$\frac{\overline{h_n} - \mathbb{E}_f(h(X))}{\sqrt{v_n}} \xrightarrow{d} N(0,1) \quad \text{as} \quad n \to \infty,$$

which allows the construction of confidence intervals for $\overline{h_n}$.

This approach is very powerful since the only requirement is that we are able to generate samples from the distribution f. One disadvantage of this method is that we need to simulate new samples each time we want to evaluate (3.1) for different distributions f which often is the case in Bayesian inference. *Importance Sampling* avoids this limitation by using samples from a different distribution g which usually is easier to simulate. It takes advantage of the fact that (3.1) can be rewritten as

$$\mathbb{E}_f(h(X)) = \int h(x)\frac{f(x)}{g(x)}g(x)dx \tag{3.3}$$

for any distribution g with $\operatorname{supp} g \supseteq \operatorname{supp} f$. Similar to the classical approach (3.3) can be approximated by

$$\frac{1}{n}\sum_{i=1}^{n} h(x_i)\frac{f(x_i)}{g(x_i)} \tag{3.4}$$

where the sample $x_1, \ldots, x_n$ is simulated from the distribution g. Since the two methods are the same if we choose $g = f$, importance sampling is a generalization of the classical Monte Carlo integration method.

We now present a result from [14] which states the optimal choice for the sampling distribution g such that the variance of the estimator (3.4) is minimal. Note that this variance is finite, if and only if

$$\mathbb{E}_g\left(h^2(X)\frac{f^2(X)}{g^2(X)}\right) = \mathbb{E}_f\left(h^2(X)\frac{f(X)}{g(X)}\right) = \int h^2(x)\frac{f^2(x)}{g(x)}dx < \infty.$$

Therefore if the ratio $\frac{f}{g}$ is unbounded the variance will often be infinite. Additionally, the weights $\frac{f(x_i)}{g(x_i)}$ will show a greater variability, thus the result will be more sensitive to the used sample. The optimal choice of g that minimizes the variance of the estimator for a given function h and distribution f is

$$g^*(x) = \frac{|h(x)|f(x)}{\int |h(z)|f(z)dz}. \tag{3.5}$$

See [14], theorem 3.3.4. To prove this result we can write the variance of the estimator (3.4) as

$$var_g\left(\frac{h(X)f(X)}{g(X)}\right) = \mathbb{E}_g\left(\frac{h^2(X)f^2(X)}{g^2(X)}\right) - \left(\mathbb{E}_g\left(\frac{h(X)f(X)}{g(X)}\right)\right)^2.$$

The second term does not depend on g so we can minimize the variance by minimizing the first term. Applying Jensen's inequality

$$\mathbb{E}_g\left(\frac{h^2(X)f^2(X)}{g^2(X)}\right) \geq \left(\mathbb{E}_g\left(\frac{|h(X)|f(X)}{g(X)}\right)\right)^2 = \left(\int |h(x)|f(x)dx\right)^2$$

yields a lower bound which is attained by choosing $g = g^*$.

From this we can see that the choice $g = f$ (the classical Monte Carlo integration method) is always suboptimal except in the trivial case where h is constant.

The dependency of the optimal choice g^* on h and f is not surprising however it reduces its practicality since one of the advantages of importance sampling should be the independence of these functions. Moreover if $h \geq 0$ the choice g^* requires knowledge of the integral $\int h(x)f(x)dx$ which we actually want to approximate. In [14], the authors suggest to use distributions g for which the term

$$\frac{|h(x)|f(x)}{g(x)}$$

is (almost) constant and has finite variance.

3.2 Markov Chains

In this section we present some of the theory for Markov chains, summarizing the more detailed introduction by Robert and Casella in [14], chapter 4.

Later in this chapter when we introduce Markov chain Monte Carlo (MCMC) algorithms we will recourse to results from Markov chain theory presented in this section to establish the validity of these algorithms.

We describe Markov chains through a *transition kernel*, an operator which describes the evolution of the chain given the current state.

Definition 11. *Let $(\Omega, \mathcal{F}, P)$ be a probability space and $K : \Omega \times \mathcal{F} \mapsto [0, 1]$ a function, such that*

- *$K(x, .)$ is a probability measure for all $x \in \Omega$.*
- *$K(., A)$ is measurable for all $A \in \mathcal{F}$.*

Then K is called a transition kernel.

We actually already defined the transition kernel in definition 3 as a version of the regular conditional probability measure with respect to a σ-algebra.

Definition 12. *Let $(\Omega, \mathcal{F}, P)$ be a probability space and K a transition kernel on $(\Omega, \mathcal{F}, P)$. A stochastic process $(X_n)_{n\in\mathbb{N}}$ is a* Markov chain, *if*

$$P(X_{t+1} \in A|x_0, \ldots, x_t) = P(X_{t+1} \in A|x_t) = \int_A K(x_t, dx) \tag{3.6}$$

for all $t \in \mathbb{N}$ and $A \in \mathcal{F}$.

Remark 4. *In this thesis we always assume $\mathbb{N}$ as the index set for Markov chains. Therefore we omit the index set in the notation and denote a Markov chain (X_n).*

Remark 5. *Since Markov chains are defined on the index set $\mathbb{N}$, we need to define the initial distribution of X_0. Given the initial distribution $X_0 \sim \mu_0$ we can construct the whole chain through (3.6). With P_μ we denote the probability measure for the chain given the initial distribution μ. If the initial distribution is a Dirac distribution δ_{x_0} we write $P_{x_0} := P_{\delta_{x_0}}$.*

Remark 6. *If the state space Ω is finite (i.e. $\Omega = \{1, \dots, k\}$), we can represent the transition kernel as a matrix $K(i,j) = (k_{ij})$. The matrix entries then correspond to the transition probabilities between two states. Therefore*

$$P(X_{t+1} = j | X_t = i) = k_{ij} \quad \forall\, i, j \in \Omega.$$

The mapping $x_{n+1} \mapsto K(x_n, x_{n+1})$ is a version of the conditional density of $X_{n+1}|X_n = x_n$ as the following properties of the transition kernel are easy to show:

$$\begin{aligned} P_x(X_1 \in A_1) &= K(x, A_1), \\ P_x((X_1, X_2) \in A_1 \times A_2) &= \int_{A_1} K(y_1, A_2) K(x, dy_1) \\ &\dots \\ P_x((X_1, \dots, X_n) \in A_1 \times \dots \times A_n) &= \\ \int_{A_1} \cdots \int_{A_{n-1}} K(y_{n-1}, A_n) K(y_{n-2}, dy_{n-1}) \cdots K(x, dy_1). & \end{aligned}$$

Therefore the transition kernel K describes the probabilistic law after which the chain evolves between two successive points in time. We can also define a kernel for multiple time steps (e.g. from t to $t+n$) by

$$\begin{aligned} K^1(x, A) &:= K(x, A), \\ K^n(x, A) &:= \int K^{n-1}(y, A) K(x, dy) \quad \text{for } n \geq 1. \end{aligned}$$

The *Chapman-Kolmogorov equation* describes a convolution identity for transition kernels.

Theorem 8 (Chapman-Kolmogorov equation)**.** *Let $(\Omega, \mathcal{F}, P)$ be a probability space, K a transition kernel, $x \in \Omega$, $A \in \mathcal{F}$ and $(n, m) \in \mathbb{N}^2$. Then*

$$K^{n+m}(x, A) = \int K^n(y, A) K^m(x, dy). \tag{3.7}$$

For a Markov chain (X_n) the dependence on its own past is limited per definition. The *Weak-* and *Strong Markow properties* generalize this fact.

Proposition 2 (Weak Markov property)**.** *Let (X_n) be a Markov chain, h a function, μ a probability measure (the initial distribution) and $(x_0, \dots, x_n)$ a sample of (X_n). Then*

$$\mathbb{E}_\mu(h(X_{n+1}, X_{n+2}, \dots) | x_0, \dots, x_n) = \mathbb{E}_{x_n}(h(X_{n+1}, X_{n+2}, \dots)),$$

provided the expectations exist.

Remark 7. *$\mathbb{E}_\mu$ is the expectation under the probability measure P_μ and similarly $\mathbb{E}_{x_n}$ is the expectation under the probability measure P_{x_n}.*

Definition 13. *Let (X_n) be a Markov chain and $A \in \mathcal{F}$. The* stopping time at *A is defined through*

$$\tau_A = \inf\{n \geq 1 : X_n \in A\}$$

and it is the first time in which the chain (X_n) enters the set A.

Proposition 3 (Strong Markov property)**.** *Let (X_n) be a Markov chain, h a function, μ a probability measure and τ a stopping time which is finite almost surely. Then, provided the expectations exist*

$$\mathbb{E}_\mu(h(X_{\tau+1}, X_{\tau+2}, \dots)|x_0, \dots, x_\tau) = \mathbb{E}_{x_\tau}(h(X_{\tau+1}, X_{\tau+2}, \dots)).$$

The *resolvant* associated with a transition kernel is a notion that is used in results concerning the stability and convergence of Markov chains.

Definition 14. *Let K be a transition kernel and $\epsilon \in (0, 1)$. The kernel*

$$K_\epsilon(x, A) = (1 - \epsilon) \sum_{i=1}^{\infty} \epsilon^i K^i(x, A)$$

is called a resolvant *associated with K. A Markov chain with transition kernel K_ϵ is called K_ϵ-chain.*

The goal of MCMC algorithms is to construct a Markov chain that converges to an invariant distribution. Therefore the chain should not be sensitive to the initial distribution or the initial starting point. The concept of *irreducibility* is important to assess this property.

Definition 15. *Let (X_n) be a Markov chain with transition kernel K and ϕ a measure. The chain is* ϕ-irreducible *if for $A \in \mathcal{F}$ with $\phi(A) > 0$ there exists some $n \in \mathbb{N}$ such that*

$$K^n(x, A) > 0 \quad \forall x \in \Omega.$$

The chain is strongly ϕ-irreducible *if $K(x, A) > 0$ for all $x \in \Omega$.*

The next theorem lists some properties that are equivalent to ϕ-irreducibility.

Definition 16. *Let (X_n) be a Markov chain and $A \in \mathcal{F}$. The random variable, defined by*

$$\eta_A = \sum_{n=1}^{\infty} \chi_A(X_n)$$

is the number of passages *of (X_n) in A.*

Theorem 9. *Let (X_n) be a Markov chain, ϕ a measure, $x \in \Omega$ and $A \in \mathcal{F}$: $\phi(A) > 0$. (X_n) is ϕ-irreducible if and only if one of the following properties holds:*

- $\exists n \in \mathbb{N} : K^n(x, A) > 0$
- $\mathbb{E}_x(\eta_A) > 0$
- $K_\epsilon(x, A) > 0 \quad \forall \epsilon \in (0, 1)$.

The next theorem shows that the measure ϕ is not that important in the definition of irreducibility and that it is a property of the Markov chain.

Theorem 10. *Let (X_n) be a ϕ-irreducible Markov chain. There exists a probability measure ψ (called the* maximal irreducibility measure*) such that:*

- *(X_n) is ψ-irreducible*
- *If (X_n) is ξ-irreducible, then $\xi \ll \psi$*
- *If $A \in \mathcal{F} : \psi(A) = 0$, then $\psi(\{y \in \Omega : P_y(\tau_A < \infty) > 0\}) = 0$*
- *The probability measure ψ is equivalent to*

$$\psi_0(A) = \int K_{\frac{1}{2}}(x, A)\phi(dx) \quad \forall A \in \mathcal{F},$$

that is, $\psi \ll \psi_0$ and $\psi_0 \ll \psi$.

Other important concepts for Markov chains are *atoms* and *small sets* which are crucial in determining the stability of a chain.

Definition 17. *Let (X_n) be a Markov chain with transition kernel K. If there exists a set $\alpha \in \mathcal{F}$ and a non-zero measure ν, such that*

$$K(x, A) = \nu(A) \quad \forall x \in \alpha, \forall A \in \mathcal{F},$$

then α is called an atom *of the chain (X_n). If (X_n) is ψ-irreducible the atom is* accessible *if $\psi(A) > 0$.*

In the case of a continuous state space, atoms are often a too strong notion since they imply a constant transition kernel on a set with positive measure. Therefore we introduce the weaker notion of *small sets*.

Definition 18. *Let K be a transition kernel. A set $C \in \mathcal{F}$ is* small *if there exists $m \in \mathbb{N}$ and a non-zero measure ν_m such that*

$$K^m(x, A) \geq \nu_m(A) \quad \forall x \in C, \forall A \in \mathcal{F}.$$

The following theorem states the connection between irreducible chains and small sets.

Theorem 11. *Let (X_n) be a ψ-irreducible Markov chain and $A \in \mathcal{F}$ such that $\psi(A) > 0$. There exists $m \in \mathbb{N}$ and a small set $C \subseteq A$ with the associated non-zero measure ν_m such that $\nu_m(C) > 0$. The state space Ω can be decomposed into a countable infinite partition of small sets.*

Additionally, Meyn and Tweedie [10] present a result which states, that for, in some sense, regular Markov chains every compact set is small.

Next, we present the construction of *split chains* introduced by Athreya and Ney in [1] and summarized in [14] through *renewal times* which serve as a technique of proof.

Definition 19. *A function $\xi(x_1, x_2, \dots)$ is called a* stopping rule *if the set $[\xi = n]$ is measurable for the σ-algebra $\sigma(X_0, X_1, \dots, X_n)$.*

Definition 20. *Let (X_n) be a Markov chain. A stopping rule τ such that $(X_\tau, X_{\tau+1}, \dots)$ and $(X_{\tau-1}, X_{\tau-2}, \dots, X_0)$ are independent is called a* renewal time.

Definition 21. *Let (X_n) be a Markov chain with transition kernel K. If there exists a set $C \in \mathcal{F}$, $\epsilon > 0$ and a probability measure ν such that*

$$K(x, A) \geq \epsilon\nu(A) \quad \forall x \in C, \forall A \in \mathcal{F}$$

the chain (X_n) fulfills the minorizing condition.

If the Markov chain (X_n) with transition kernel K fulfills the minorizing condition for the set C, the constant ϵ and the probability measure ν and additionally $P_x(\tau_C < \infty) = 1$, $\forall x \in \Omega$ we can modify the transition kernel when $X_n \in C$, such that

$$X_{n+1} \sim \begin{cases} \nu & \text{with probability} \quad \epsilon \\ \frac{K(X_n,.)-\epsilon\nu(.)}{1-\epsilon} & \text{with probability} \quad 1-\epsilon. \end{cases}$$

The conditional distribution $X_{n+1}|X_n = x_n$ does not change since

$$\epsilon\nu(A) + (1-\epsilon)\frac{K(x_n, A) - \epsilon\nu(A)}{1-\epsilon} = K(x_n, A).$$

However this modified kernel produces renewal times, whenever $X_n \in C$ and $X_{n+1} \sim \nu$ since ν is independent of the past history. The *split chain* of (X_n) is the chain $(\hat{X}_n) = (X_n, \hat{\omega}_n)$ where $\hat{\omega}_n = 1$ when $X_n \in C$ and $X_{n+1} \sim \nu$. This chain posesses the atom $\hat{\alpha} = C \times \{1\}$. The subchain (X_n) is still a Markov chain.

The notion of *aperiodicity* is another important property for the convergence of Markov chains.

Definition 22. *Let (X_n) be a ψ-irreducible Markov chain. If there exists a small set $C \in \mathcal{F}$ with integer $M \in \mathbb{N}$ and an associated probability measure ν_M then the chain has a* cycle of length *d, where d is the greatest common denominator of the integers*

$$\{m \geq 1 : \exists \delta_m > 0 : C \text{ is small for } \nu_m \geq \delta_m \nu_M\}.$$

The Markov chain (X_n) is aperiodic *if it has a cycle of length $d = 1$.*

Irreducibility of a Markov chain ensures that the chain visits every set with positive measure at least once. However for Markov chain Monte Carlo algorithms we require a stronger property which ensures that each set is visited often enough.

Definition 23. *Let (X_n) be a Markov chain. A set $A \in \mathcal{F}$ is called* recurrent *if $\mathbb{E}_x(\eta_A) = \infty$ for every $x \in A$. The set A is* uniformly transient *if there exists a constant M such that $\mathbb{E}_x(\eta_A) < M$ for all $x \in A$. It is* transient *if there exists a countable collection of uniformly transient sets B_i such that*

$$A = \bigcup_i B_i.$$

The following results establish the connection between irreducibility and recurrence.

Definition 24. *Let (X_n) be a Markov chain. (X_n) is* recurrent*, if*

- *there exists a measure ψ such that the chain is ψ-irreducible.*
- $\mathbb{E}_x(\eta_A) = \infty \quad \forall A \in \mathcal{F} : \psi(A) > 0, \forall x \in A.$

The chain is transient *if it is ψ-irreducible and if Ω is transient.*

Theorem 12. *Let (X_n) be a ψ-irreducible Markov chain with an accessible atom $\alpha \in \mathcal{F}$. Then,*

- *if α is recurrent, every accessible set $A \in \mathcal{F}$ is recurrent.*
- *if α is transient, Ω is transient.*

Theorem 13. *A ψ-irreducible chain is either recurrent or transient.*

The next proposition states a criterion for recurrence that is based on small sets.

Proposition 4. *Let (X_n) be a ψ-irreducible Markov chain. The chain is recurrent if there exists a small set C with $\psi(C) > 0$ such that $P_x(\tau_C < \infty) = 1$ for each $x \in C$.*

A further property to strenghten the stability of Markov chains is by requiring an infinite amount of visits to each small set *for every sample path* and is called *Harris recurrence.*

Definition 25. *Let (X_n) be a Markov chain and $A \in \mathcal{F}$. The set A is* Harris recurrent *if $P_x(\eta_A = \infty) = 1$ for all $x \in A$. The chain (X_n) is* Harris recurrent *if there exists a measure ψ such that (X_n) is ψ-irreducible and if every accessible set A is Harris recurrent.*

The requirement $\mathbb{E}_x(\eta_A) = \infty$ in definition 23 implies $P_x(\eta_A = \infty) > 0$. However Harris recurrence requires $P_x(\eta_A = \infty) = 1$. Therefore it is clear that Harris recurrence is a stronger property than recurrence. The following results compare Harris recurrence to conditions on stopping times.

Proposition 5. *Let (X_n) be a Markov chain. If for every set $A \in \mathcal{F}$*

$$P_x(\tau_A < \infty) = 1 \quad \forall x \in A,$$

then

$$P_x(\eta_A = \infty) = 1 \quad \forall x \in A$$

and the chain (X_n) is Harris recurrent.

Theorem 14. *Let (X_n) be a ψ-irreducible Markov chain with a small set C such that $P_x(\tau_C < \infty) = 1$ for all $x \in \Omega$. The chain (X_n) is Harris recurrent.*

Positive recurrence is an even stronger property, that most Markov chains constructed by Markov chain Monte Carlo algorithms possess. For such chains there exists an *invariant probability measure* which is also called *stationary measure.*

Definition 26. *Let (X_n) be a Markov chain with transition kernel K. A σ-finite measure π is* invariant *for the transition kernel, if*

$$\pi(B) = \int K(x, B)\pi(dx) \quad \forall B \in \mathcal{F}.$$

Definition 27. *A ψ-irreducible Markov chain with an invariant measure is called* positive recurrent*. A recurrent Markov chain without an invariant measure is called* null recurrent*.*

Proposition 6. *Let (X_n) be a Markov chain. If (X_n) is positive recurrent it is also recurrent.*

The following theorem provides an equivalent classification of positivity for chains with an atom.

Theorem 15. *Let (X_n) be a ψ-irreducible Markov chain with an atom α. The chain is positive recurrent if and only if $\mathbb{E}_\alpha(\tau_\alpha) < \infty$. In this case the invariant distribution π for the chain (X_n) satisfies*

$$\pi(\alpha) = (\mathbb{E}_\alpha(\tau_\alpha))^{-1}.$$

Remark 8. *$\mathbb{E}_\alpha$ is the expectation under a probability measure P_x with $x \in \alpha$. The probability measure P_x is the same for all $x \in \alpha$ by definition 17. Therefore the expectation $\mathbb{E}_\alpha$ is well defined.*

The next theorem assures the existence of invariant measures for recurrent Markov chains.

Theorem 16. *Let (X_n) be a recurrent Markov chain. There exists an invariant σ-finite measure which is unique up to a multiplicative factor.*

The next step in the study of Markov chains is to evaluate the limiting behaviour of the elements X_i. Since for recurrent Markov chains there exists an invariant measure π (theorem 16), this measure is a candidate for the limiting distribution of the series of X_n when $n \to \infty$. We denote the distribution of X_n by P^n. One important condition for the convergence of P^n to π is *ergodicity.*

Definition 28. *Let (X_n) be a Harris positive Markov chain with invariant distribution π. An atom α of the chain for which*

$$\lim_{n\to\infty} |K^n(\alpha, \alpha) - \pi(\alpha)| = 0$$

is called ergodic*.*

Remark 9. *Note that the expression $K(\alpha, \alpha)$ is well defined since $K(x, \alpha) = \nu(\alpha)$ for all $x \in \alpha$ (see definition 17).*

Definition 29. *Let $(\Omega, \mathcal{F})$ be a measurable space with measures μ and ν. The distance of between the two measures based on the* total variation norm $\|.\|_{TV}$ *is defined as*

$$\|\mu - \nu\|_{TV} = \sup_{A\in\mathcal{F}} |\mu(A) - \nu(A)|.$$

Proposition 7. *Let (X_n) be a Harris positive Markov chain with transition kernel K on $(\Omega, \mathcal{F}, P)$ where Ω is countable. If there exists an ergodic atom α and an invariant distribution π then*

$$\lim_{n\to\infty} \|K^n(x,.) - \pi\|_{TV} = 0 \quad \forall x \in \Omega.$$

Theorem 17. *Let (X_n) be a positive recurrent, aperiodic Markov chain with transition kernel K on $(\Omega, \mathcal{F}, P)$ where Ω is countable and an invariant distribution π. Then*

$$\lim_{n\to\infty} \|K^n(x,.) - \pi\|_{TV} = 0 \quad \forall x \in \Omega.$$

Theorem 18. *Let (X_n) be a Harris positive and aperiodic Markov chain with transition kernel K. Then, for every initial distribution μ*

$$\lim_{n\to\infty} \left\| \int K^n(x,.)\mu(dx) - \pi \right\|_{TV} = 0.$$

We end this section with two important convergence theorems for Markov chains. The *Ergodic Theorem* (see [14], theorem 4.7.4) which can be interpreted as a *Law of Large Numbers* for Markov chains ensures the convergence of the partial sums

$$S_n(h) = \frac{1}{n}\sum_{i=1}^{n} h(X_i)$$

where h is integrable and (X_n) is a Markov chain.

Theorem 19 (Ergodic Theorem)**.** *Let (X_n) be a Markov chain with a σ-finite invariant measure π. The following statements are equivalent:*

- *If $f, g \in L_1(\Omega, \mathcal{F}, \pi)$ with $\int g d\pi \neq 0$, then*

$$\lim_{n\to\infty} \frac{S_n(f)}{S_n(g)} = \frac{\int f d\pi}{\int g d\pi}.$$

- *The Markov chain (X_n) is Harris recurrent.*

The last theorem in this section is a *Central Limit Theorem* for Markov chains (see [14], theorem 4.7.5). However, other than in the iid case, this theorem requires more restrictive conditions.

Theorem 20 (Central Limit Theorem for Markov chains)**.** *Let (X_n) be a Harris positive Markov chain with an atom α, invariant distribution π and let $h \in L_1(\Omega, \mathcal{F}, \pi)$. If*

$$\mathbb{E}_\alpha(\tau_\alpha^2) < \infty, \quad \mathbb{E}_\alpha\left(\left(\sum_{n=1}^{\tau_\alpha} |h(X_n)|\right)^2\right) < \infty$$

and

$$\gamma_h^2 = \pi(\alpha)\mathbb{E}_\alpha\left(\left(\sum_{n=1}^{\tau_\alpha} (h(X_n) - \mathbb{E}_\pi(h))\right)^2\right) > 0$$

then

$$\frac{1}{\sqrt{n}}\left(\sum_{i=1}^{n} (h(X_i) - \mathbb{E}_\pi(h))\right) \xrightarrow{d} N(0, \gamma_h^2).$$

3.3 The Metropolis-Hastings Algorithm

The *Metropolis-Hastings algorithm* belongs to the family of *Markov chain Monte Carlo (MCMC)* methods. It generates a Markov chain with a specific invariant distribution to which the chain converges. Therefore we can use the elements of the generated Markov chain from a certain *cutoff* index on as an approximate sample from the invariant distribution which in turn can be used to approximate integrals of the form (3.1). MCMC methods are therefore an alternative to the methods described in section 3.1. In the case of MCMC methods the theoretic argument for the convergence of the approximation is the *ergodic theorem 19.*

In this section we introduce the Metropolis-Hastings algorithm and establish conditions for its convergence. A more detailed discussion of the Metropolis-Hastings algorithm can be found in [14], chapter 6.

Given a distribution with density f that we want to simulate, the algorithm generates a Markov chain with invariant distribution f. The elements of the Markov chain are generated by drawing a sample from a conditional *proposal density* q and accepting this sample with a certain *acceptance probability.*

Algorithm 1 (Metropolis-Hastings)**.** *Given x_t the element of the Metropolis-Hastings Markov chain at time t,*

1. *Generate $Y_t \sim q(.|x_t)$.*
2. *Set the element of the chain at time $t+1$*

$$X_{t+1} = \begin{cases} Y_t & \text{with probability} \quad \rho(x_t, Y_t) \\ x_t & \text{with probability} \quad 1-\rho(x_t, Y_t), \end{cases}$$

where

$$\rho(x,y) = \min\left\{\frac{f(y)}{f(x)}\frac{q(x|y)}{q(y|x)}\right\}.$$

The algorithm is very flexible and allows a lot of variations based on the nature of the proposal density. Before we explore some variations of the Metropolis-Hastings algorithm, we summarize conditions which ensure the convergence of the algorithm. First we take a closer look at the transition kernel which will then allow us to use the theoretical results for Markov chains presented in section 3.2.

Definition 30. *Let (X_n) be a Markov chain with transition kernel K. The chain satisfies the* detailed balance condition *if there exists a function f such that*

$$K(y,x)f(y) = K(x,y)f(x) \quad \forall (x,y) \in \Omega \times \Omega.$$

Theorem 21. *Let (X_n) be a Markov chain with transition kernel K. If the chain satisfies the detailed balance condition with a probability density function f, then f is the invariant distribution (density) of the chain.*

Proof. Let $B \in \mathcal{F}$. Then

$$\int K(y,B)f(y)dy = \int\int_B K(y,x)f(y)dxdy =$$
$$\int\int_B K(x,y)f(x)dxdy = \int_B f(x)dx$$

which means that f is by definition 26 an invariant distribution of the chain. □

This result allows us to proof that the Markov chain generated by the Metropolis-Hastings algorithm has the target distribution f as an invariant distribution.

Theorem 22. *Let (X_n) be the Markov chain associated with the Metropolis-Hastings algorithm with the target distribution f. For each conditional proposal distribution g, such that*

$$\operatorname{supp} g \supseteq \operatorname{supp} f,$$

f is a stationary distribution of the chain (X_n).

Proof. The transition kernel for the Metropolis-Hastings algorithm is

$$K(x,y) = \rho(x,y)q(y|x) + (1 - r(x))\delta_x(y), \tag{3.8}$$

where $r(x) = \int \rho(x,y)q(y|x)dy$. To verify the detailed balance condition for (3.8) we can show that

$$\rho(x,y)q(y|x)f(x) = \rho(y,x)q(x|y)f(y)$$

and

$$(1 - r(x))\delta_x(y)f(x) = (1 - r(y))\delta_y(x)f(y).$$

Since K satisfies the detailed balance condition we can use theorem 21 to proof that f is an invariant distribution of the chain. □

Theorem 22 establishes that the Metropolis-Hastings algorithm is a valid approach to approximate samples from the target distribution with minimal restrictions on the conditional proposal density. The next theorem states the conditions under which the Markov chain from the Metropolis-Hastings algorithm can be used to approximate integrals of the form (3.1). The proof makes use of the ergodic theorem 19 and can be found in [14], theorem 6.2.5.

Theorem 23. *Let (X_t) be a Markov chain generated by the Metropolis-Hastings algorithm with transition kernel K. If the chain is μ_f-irreducible then:*

- $\lim_{T\to\infty} \frac{1}{T} \sum_{i=1}^{T} h(X_t) = \int h(x)f(x)dx \quad \mu_f - a.s. \quad \forall h \in L_1(\Omega, \mathcal{F}, \mu_f)$
- *If (X_t) is also aperiodic, then*

$$\lim_{n\to\infty} \left\| \int K^n(x,.)\mu(dx) - \mu_f \right\|_{TV} = 0$$

  for each initial distribution μ.

The next theorems state alternative conditions under which the conclusions of theorem 23 hold.

Theorem 24. *The conclusions of theorem 23 hold if the conditional proposal density q satisfies*

$$P\left(f(X_t)q(Y_t|X_t) \leq f(Y_t)q(X_t|Y_t)\right) < 1$$

and

$$q(x|y) > 0 \quad \forall (x,y) \in \operatorname{supp}(f)^2.$$

Theorem 25. *The conclusions of theorem 23 hold if the target density f is bounded and positive on every compact set in* $\operatorname{supp} f$ *and if there exists $\epsilon > 0$ and $\delta > 0$ such that*

$$q(y|x) > \epsilon \quad \text{if} \quad |x - y| < \delta.$$

We now describe two specializations of the Metropolis-Hastings algorithm which differ in the king of proposal distribution used to construct the Markov chain. A more detailed presentation can be found in [14], section 6.3.

Algorithm 2 (Independent Metropolis-Hastings)**.** *Given x_t the element of the Metropolis-Hastings Markov chain at time t,*

1. *Generate $Y_t \sim g(.)$*
2. *Set*

$$X_{t+1} = \begin{cases} Y_t & \text{with probability} \quad \min\left\{\frac{f(Y_t)}{f(x_t)}\frac{g(x_t)}{g(Y_t)}, 1\right\} \\ x_t & \text{otherwise.} \end{cases}$$

At each time step the generated proposal Y_t is independent of the current element x_t in the chain. A different approach is the *Random walk Metropolis-Hastings algorithm* where the proposal is simulated according to

$$Y_t = X_t + \epsilon_t$$

where ϵ_t is a random variable with distribution g, and therefore the conditional proposal density is of the form

$$q(y|x) = g(y - x).$$

Algorithm 3 (Random walk Metropolis-Hastings)**.** *Given x_t the element of the Metropolis-Hastings Markov chain at time t:*

1. *Generate $Y_t \sim g(y - x_t)$*
2. *Set*

$$X_{t+1} = \begin{cases} Y_t & \text{with probability} \quad \min\left\{\frac{f(Y_t)}{f(x_t)}, 1\right\} \\ x_t & \text{otherwise.} \end{cases}$$

The Metropolis-Hastings algorithm described so far worked with a fixed proposal density as input which determined the transition kernel and in turn the theoretical properties of the Markov chain. Therefore the proposal density has to be chosen carefully to suit the algorithms target density. The *Adaptive Rejection Metropolis Sampling (ARMS)* algorithm, developed by Gilks et al. [4] constructs a proposal density by adapting to the target density in a seperate step. However through this adaption the resulting Markov chain is time inhomogenous as the transition kernel is changed with each adaption. Therefore the theoretical results for Markov chains in section 3.2 do not apply. This problem can be circumvented by stopping the adaption after a certain number of steps and then fixing the proposal density. From then on the Markov chain is time homogenous and the theoretical results can be applied.

Before we describe the ARMS algorithm we introduce some notation. Let $f_1 \propto f$ be proportional to the target density f and set $h(x) = \log f_1(x)$. Further let $S_n = \{x_i : 0 \leq i \leq n+1\}$ be a set of points in $\operatorname{supp} f$. With $y = L_{i,i+1}(x)$ we

denote the equations for the lines between $(x_i, h(x_i))$ and $(x_{i+1}, h(x_{i+1}))$. We then set

$$\tilde{h}_n(x) = \begin{cases} L_{0,1}(x) & x < x_0 \\ \max\{L_{0,1}(x), L_{1,2}(x)\} & x_0 \leq x < x_1 \\ \max\{L_{i,i+1}(x), \min\{L_{i-1,i}(x), L_{i+1,i+2}(x)\}\} & x_i \leq x < x_{i+1}, \\ & i = 1, \ldots, n-1 \\ \max\{L_{n,n+1}(x), L_{n-1,n}(x)\} & x_n \leq x < x_{n+1} \\ L_{n,n+1}(x) & x_{n+1} \leq x \end{cases}$$

which determines the proposal density for the *Accept-Reject* part of the algorithm as $g_n \propto \exp(\tilde{h}_n)$. The instrumental density is

$$\psi_n(x) \propto \min\{f_1(x), \exp(\tilde{h}_n(x))\}.$$

Algorithm 4 (Adaptive Rejection Metropolis Sampling)**.** *With the notation introduced above and the value x_t of the Markov chain at time t:*

1. *Simulate $Y \sim g_n(.)$ and $U \sim U_{[0,1]}$ until*

$$U \leq \frac{f_1(Y)}{\exp(\tilde{h}_n(Y))}.$$

Set $S_{n+1} = S_n \cup \{Y\}$ if Y is rejected and the algorithm is in its burn-in phase.

2. *Set*

$$X_{t+1} = \begin{cases} Y & \text{with probability } \min\left\{\frac{f_1(Y)\psi_n(x_t)}{f_1(x_t)\psi_n(Y)}, 1\right\} \\ x_t & \text{otherwise.} \end{cases}$$

3.4 The Gibbs Sampler

The Metropolis-Hastings algorithm from the previous section did not take into account the particular form of the target distribution. In that sense it is a *generic* sampling algorithm. The *Gibbs sampler* that we describe in this section is more closely linked to the target distribution and requires explicit knowledge about it. The disadvantage from this necessity is countered by an improved performance over more general algorithms.

Gibbs sampling is by construction multi-dimensional with the target distribution f decomposed into individual parts $f = (f_1, \ldots, f_p)$, $p > 1$ such that the conditional components $f_i(.|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_p)$ can be simulated. The densities f_i of $X_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_p$ are called the *full conditionals*.

Algorithm 5 (Gibbs sampler)**.** *Given the value $x_t = (x_{t,1}, \ldots, x_{t,p})$ of the chain at time t, generate:*

1 $X_{t+1,1} \sim f_1(.|x_{t,2}, \ldots, x_{t,p})$

2 $X_{t+1,2} \sim f_2(.|x_{t+1,1}, x_{t,3}, \ldots, x_{t,p})$

$\vdots$

p $X_{t+1,p} \sim f_p(.|x_{t+1,1}, \ldots, x_{t+1,p-1})$.

A more general version of the Gibbs sampling algorithm can be constructed by *demarginalizing* the target distribution f and using a *completion* distribution g such that

$$\int g(x,z)dz = f(x).$$

For this, set $y = (x,z)$ and denote the conditional densities of $g(y) = g(y_1,\ldots,y_p)$ by

$$\begin{aligned} Y_1|y_2,\ldots,y_p &\sim g_1(.|y_2,\ldots,y_p)\\ Y_2|y_1,y_3,\ldots,y_p &\sim g_2(.|y_1,y_3,\ldots,y_p)\\ &\vdots\\ Y_p|y_1,\ldots,y_{p-1} &\sim g_p(.|y_1,\ldots,y_{p-1}). \end{aligned}$$

Algorithm 6 (Completion Gibbs sampler)**.** *Given the value $y_t = (y_{t,1},\ldots,y_{t,p})$ of the chain at time t, generate:*

1 $Y_{t+1,1} \sim g_1(.|y_{t,2},\ldots,y_{t,p})$

2 $Y_{t+1,2} \sim g_2(.|y_{t+1,1},y_{t,3},\ldots,y_{t,p})$

$\vdots$

p $Y_{t+1,p} \sim g_p(.|y_{t+1,1},\ldots,y_{t+1,p-1})$.

The completion distribution g should be chosen such that the full conditionals g_i can be easily simulated. With this construction the chain (Y_t) generated by the *Completion Gibbs sampler* is a Markov chain (however the subchain (X_t) is not necessarily a Markov chain) and therefore we can establish convergence properties for the algorithm similar to the ones for the Metropolis-Hastings algorithm.

Theorem 26. *If the Markov chain (Y_t) generated from the Gibbs sampler 6 is ergodic, then g is an invariant distribution of (Y_t) and f is the limiting distribution of the subchain (X_t).*

Theorem 27. *If the transition kernel*

$$\begin{aligned} K(y,y') =& g_1(y_1'|y_2,\ldots,y_p)g_2(y_2'|y_1,y_3,\ldots,y_p)\cdots\\ & g_p(y_p'|y_1,\ldots,y_{p-1}) \end{aligned}$$

associated with the Gibbs sampling algorithm 6 is absolute continuous with respect to the dominating measure, the generated Markov chain is Harris recurrent.

This result allows a proof of a convergence theorem similar to 23 for Metropolis-Hastings.

Theorem 28. *If the transition kernel associated with the Gibbs sampling algorithm 6 with the Markov chain (Y_t) is absolutely continuous with respect to the dominating measure, then:*

- *If* $h_1, h_2 \in L_1(\Omega, \mathcal{F}, \mu_g)$ *with* $\int h_2 d\mu_g \neq 0$, *then*

$$\lim_{T\to\infty} \frac{\sum_{t=1}^{T} h_1(Y_t)}{\sum_{t=1}^{T} h_2(Y_t)} = \frac{\int h_1 d\mu_g}{\int h_2 d\mu_g} \quad \mu_g - a.s.$$

- *If* (Y_t) *is also aperiodic, then*

$$\lim_{n\to\infty} \left\| \int K^n(y, .)\mu(dy) - \mu_g \right\|_{TV} = 0$$

for every initial distribution μ.

Similar to theorem 25 for the Metropolis-Hastings algorithm, Robert and Casella state an alternative condition on g such that theorem 28 holds. See [14], lemma 7.1.15 and corollary 7.1.16.

One requireemnt of the Gibbs sampling algorithms above is the ability to simulate samples directly from the full conditionals f_i or g_i respectively. If this requirement is not met for some $i \in \{1, \ldots, p\}$ we can customize algorithm 6 and insert a Metropolis-Hastings step and sample from an instrumental distribution q_i instead of g_i. This modification was suggested by Müller [11] and is called *Metropolis-within-Gibbs*.

Algorithm 7 (Metropolis-within-Gibbs)**.** *In the setup of algorithm 6, and* $(y_{t+1,1}, \ldots, y_{t+1,i-1}, y_{t,i}, \ldots, y_{t,p})$ *is given:*

1. *Simulate* $\tilde{Y}_i \sim q_i(.|y_{t+1,1}, \ldots, y_{t+1,i-1}, y_{t,i+1}, \ldots, y_{t,p})$
2. *Set*

$$Y_{t+1,i} = \begin{cases} \tilde{Y}_i & \text{with probability } \rho \\ y_{t,i} & \text{with probability } 1-\rho, \end{cases}$$

where

$$\rho = \min \left\{ \frac{\frac{g_i(\tilde{Y}_i|y_{t+1,1},\ldots,y_{t+1,i-1},y_{t,i+1},\ldots,y_{t,p})}{q_i(\tilde{Y}_i|y_{t+1,1},\ldots,y_{t+1,i-1},y_{t,i+1},\ldots,y_{t,p})}}{\frac{g_i(y_{t,i}|y_{t+1,1},\ldots,y_{t+1,i-1},y_{t,i+1},\ldots,y_{t,p})}{q_i(y_{t,i}|y_{t+1,1},\ldots,y_{t+1,i-1},y_{t,i+1},\ldots,y_{t,p})}}, 1 \right\}.$$

Chapter 4

MCMC for Bayesian Time Series Analysis

In this chapter we use the Metropolis-Hastings and Gibbs sampling algorithms to analyze a time series model from a Bayesian context. We choose a simple *AR(1)-process* with unknown variance and a conjugate prior which allows us to compare the numerical results from the MCMC algorithms to the analytical solution. We simulate different time series with differing distributions for the error variables which allows us to assess the robustness of the methods. In the next section 4.1 we describe the model we use for the time series and derive the closed form solution for the posterior distribution. In section 4.2 we present the MCMC methods used for the example, which are the Metropolis-Hastings algorithm and the Gibbs sampler. The results of these algorithms are presented in section 4.3 along with comparisons to the theoretical results. We end the chapter with some concluding remarks in section 4.4. We implemented the simulations in `C++` and generated the graphics in `R` using the `ggplot2` library. The source code of the simulations is listed in appendix B.

4.1 The AR(1)-process

For the example in this chapter we consider an AR(1) process on the index set $\mathbb{N}$. A discussion of higher order AR(p) processes from a Bayesian standpoint see [12]. Thus for a given starting value x_0 the process can be constructed through

$$X_t = \rho X_{t-1} + \epsilon_t, \quad t \in \mathbb{N}. \tag{4.1}$$

We assume the error process $(\epsilon_t)_{t\in\mathbb{N}}$ to be white noise (see section 1.5) with a probability density function $f_\epsilon(.|\theta)$ with parameter θ. The likelihood function of the process is then given by

$$L(\rho, \theta; x_0, \ldots, x_T) = \prod_{t=1}^{T} f_\epsilon(x_t - \rho x_{t-1}|\theta). \tag{4.2}$$

Given observations $x_0, \ldots, x_T$ from an AR(1)-process our goal is for a given prior distribution to obtain a posterior distribution for the parameter (ρ, θ),

which is a straightforward application of Bayes theorem. If we assume that the error variables ϵ_t are normally distributed with unknown precission τ then the normal-gamma distribution is a conjugate prior. With normally distributed error variables the likelihood of the process can be written as

$$\begin{aligned} L(\rho, \tau; x_0, \ldots, x_T) &= \prod_{t=1}^{T} \sqrt{\frac{\tau}{2\pi}} \exp\left(-\frac{\tau}{2}(x_t - \rho x_{t-1})^2\right) \\ &= \left(\frac{\tau}{2\pi}\right)^{\frac{T}{2}} \exp\left(-\frac{\tau}{2}\sum_{t=1}^{T}(x_t - \rho x_{t-1})^2\right). \end{aligned}$$

The normal-gamma prior with hyperparameter (m, d^2, a, b) has the density function

$$\pi(\rho, \tau) \propto \tau^{a-1} e^{-b\tau} \sqrt{\tau} \exp\left(-\frac{\tau}{2d^2}(\rho - m)^2\right).$$

Applying the likelihood and the prior to Bayes theorem yields the posterior distribution

$$\begin{aligned} \pi(\rho, \tau | x_0, \ldots, x_T) &\propto L(\rho, \tau; x_0, \ldots, x_T)\pi(\rho, \tau) \\ = \left(\frac{\tau}{2\pi}\right)^{\frac{T}{2}} \exp\left(-\frac{\tau}{2}\sum_{t=1}^{T}(x_t - \rho x_{t-1})^2\right) &\tau^{a-1} e^{-b\tau} \sqrt{\tau} \exp\left(-\frac{\tau}{2d^2}(\rho - m)^2\right). \end{aligned}$$

Setting $d^{*2} := \left(\frac{1}{d^2} + \sum_t x_{t-1}^2\right)^{-1}$ and $m^* := d^{*2}\left(\sum_t x_t x_{t-1} + \frac{m}{d^2}\right)$ we can write the posterior as

$$\begin{aligned} \pi(\rho, \tau | x_0, \ldots, x_T) \propto \tau^{a+\frac{T}{2}-1} \exp\left(-\tau\left(b + \frac{1}{2}\left(\frac{m^2}{d^2} - \frac{m^{*2}}{d^{*2}} + \sum_t x_t^2\right)\right)\right) \times \\ \tau^{\frac{1}{2}} \exp\left(-\frac{\tau}{2d^{*2}}(\rho - m^*)^2\right). \end{aligned}$$

From this we can see that the posterior distribution is again a normal-gamma distribution with hyperparameter (m^*, d^{*2}, a^*, b^*) where

$$\begin{aligned} d^{*2} &= \left(\frac{1}{d^2} + \sum_t x_{t-1}^2\right)^{-1}, \\ m^* &= d^{*2}\left(\sum_t x_t x_{t-1} + \frac{m}{d^2}\right), \\ a^* &= a + \frac{T}{2}, \\ b^* &= b + \frac{1}{2}\left(\frac{m^2}{d^2} - \frac{m^{*2}}{d^{*2}} + \sum_t x_t^2\right). \end{aligned}$$

4.2 MCMC Methods

In this section we describe the Metropolis-Hastings algorithm and the Gibbs sampler for the application of approximating the parameter posterior distribution of the AR(1)-process.

4.2.1 Metropolis-Hastings

In the Metropolis-Hastings algorithm the target density f is only used as a quotient in the expression for the acceptance probability. Therefore it only has to be known up to a proportional factor. Since $f(\rho,\tau) \propto L(\rho,\tau;x_0,\dots,x_T)\pi(\rho,\tau)$ we can write the acceptance probability as

$$\rho(\rho_x,\tau_x,\rho_y,\tau_y) = \min\left\{1, \frac{L(\rho_y,\tau_y;x_0,\dots,x_T)\pi(\rho_y,\tau_y)}{L(\rho_x,\tau_x;x_0,\dots,x_T)\pi(\rho_x,\tau_x)} \frac{q(\rho_x,\tau_x|\rho_y,\tau_y)}{q(\rho_y,\tau_y|\rho_x,\tau_x)}\right\}. \tag{4.3}$$

The question remaining is what proposal distribution q should be used. We choose an exponential distribution for the parameter τ and a normal distribution for the parameter ρ. To obtain the parameters for the proposal distribution we first caclulate the maximum likelihood estimates of ρ and τ which are

$$\hat{\rho} = \frac{\sum_t x_t x_{t-1}}{\sum_t x_{t-1}^2}$$

and

$$\hat{\tau} = \frac{T-1}{\sum_t (x_t - \hat{\rho}x_{t-1})^2}.$$

$\hat{\tau}$ will be our parameter for the proposal distribution of τ, therefore

$$\tau_{prop} \sim Ex_{\hat{\tau}}.$$

The variance of $\hat{\rho}$ is of order $(\sum_t x_{t-1}^2)^{-1}$ (see [16]) and therefore should be an appropriate value for the variance of the proposal distribution for ρ. We use a random walk design for the mean of the proposal distribution of ρ which gives

$$\rho_{prop,i} \sim N\left(\rho_{i-1}, \left(\sum_t x_{t-1}^2\right)^{-1}\right)$$

for the proposal distribution at iteration i of the parameter Markov chain.

With the proposal distribution we have everything needed to apply the Metropolis-Hastings algorithm.

4.2.2 Gibbs Sampler

For the application of the Gibbs sampling algorithm we first need to derive the full conditionals $\rho|\tau$ and $\tau|\rho$. A derivation for a more general process with constant drift can be found in [9]. We can rewrite $L(\rho,\tau;x_0,\dots,x_T)\pi(\rho,\tau)$ to get

$$\begin{aligned} f(\tau|\rho,x_0,\dots,x_T) &\propto L(\rho,\tau;x_0,\dots,x_T)\pi(\rho,\tau) \\ &\propto \tau^{\frac{T}{2}} \exp\left(-\frac{\tau}{2}\sum_{t=1}^{T}(x_t-\rho x_{t-1})^2\right)\tau^{a-1}e^{-b\tau}\sqrt{\tau}\exp\left(-\frac{\tau}{2d^2}(\rho-m)^2\right) \\ &\propto \tau^{a+\frac{T+1}{2}-1}\exp\left(-\frac{\tau}{2}\left(2b+\sum_{t=1}^{T}(x_t-\rho x_{t-1})^2+\frac{(\rho-m)^2}{d^2}\right)\right). \end{aligned}$$

Therefore $\tau|\rho \sim \gamma(\tilde{a}, \tilde{b})$ with

$$\tilde{a} = a + \frac{T+1}{2},$$
$$\tilde{b} = b + \frac{1}{2}\left(\sum_{t=1}^{T}(x_t - \rho x_{t-1})^2 + \frac{(\rho - m)^2}{d^2}\right).$$

For the full conditional $\rho|\tau$ we can again rewrite $L(\rho, \tau; x_0, \ldots, x_T)\pi(\rho, \tau)$ in the following way:

$$f(\rho|\tau, x_0, \ldots, x_T) \propto L(\rho, \tau; x_0, \ldots, x_T)\pi(\rho, \tau)$$
$$\propto \tau^{\frac{T}{2}} \exp\left(-\frac{\tau}{2}\sum_{t=1}^{T}(x_t - \rho x_{t-1})^2\right)\tau^{a-1}e^{-b\tau}\sqrt{\tau}\exp\left(-\frac{\tau}{2d^2}(\rho - m)^2\right)$$
$$\propto \exp\left(-\frac{\tau}{2}\left(\sum_t x_t^2 - 2\rho\sum_t x_t x_{t-1} + \rho^2\sum_t x_{t-1}^2 + \frac{1}{d^2}\left(\rho^2 - 2\rho m + m^2\right)\right)\right).$$

Setting $d'^2 := \left(\frac{1}{d^2} + \sum_t x_{t-1}^2\right)^{-1}$ and $m' := d'^2\left(\frac{m}{d^2} + \sum_t x_t x_{t-1}\right)$ we can simplify the last expression to get

$$f(\rho|\tau, x_0, \ldots, x_T) \propto \exp\left(-\frac{\tau}{2d'^2}\left(\rho - m'\right)^2\right)$$

and therefore $\rho|\tau \sim N\left(m', \frac{d'^2}{\tau}\right)$.

We now have everything needed to apply the Gibbs sampler 5 and the Metropolis-Hastings algorithm 1 to the simulated time series.

4.3 Simulation Results

In this section we present the results of the Metropolis-Hastings algorithm and the Gibbs sampler applied to the simulated time series of the AR(1)-process described in section 4.1. We simulated four different time series of length $T = 100$ with $\rho = 0.8$ and $\epsilon_t^{(1)} \sim N(0,1)$, $\epsilon_t^{(2)} \sim N(0,5)$, $\epsilon_t^{(3)} \sim t_1$, $\epsilon_t^{(4)} \sim t_3$, $t = 1, \ldots, T$. As prior distribution we chose a normal-gamma distribution $NG(m, d^2, a, b)$ with hyperparameters $m = 0$, $d^2 = 4$, $a = 1$ and $b = 1$. For the processes $N(0,1)$ and $N(0,5)$ the prior is conjugate to the model and the posterior hyperparameters are listed in table 4.1.

We also calculated the "posterior hyperparameters" for the processes t_1 and t_3, although the prior is not conjugate to the model in these cases, to assess the robustness of the methods against model misspecifications in the error distribution. These hyperparameters are also listed in table 4.1.

Next we describe the results of the individual MCMC methods in detail.

4.3.1 Metropolis-Hastings

We applied the Metropolis-Hastings algorithm as described in the previous section on all simulated time series which produced samples of size 50000 with a *burn-in* period of 10000. The sample means and variances along with the

	ρ		τ	
Process	m^*	d^{*2}	a^*	b^*
$N(0,1)$	0.743	0.003521	50.5	58.838
$N(0,5)$	0.806	0.000539	50.5	325.875
t_1	0.678	0.000104	50.5	2607.022
t_3	0.744	0.001059	50.5	154.386

Table 4.1: Theoretical posterior hyperparameters from the conjugate AR(1)-process models.

	Mean		Variance	
Process	Sim.	Theor.	Sim.	Theor.
$N(0,1)$	0.744	0.743	$4.19 \cdot 10^{-3}$	$4.10 \cdot 10^{-3}$
$N(0,5)$	0.808	0.806	$3.13 \cdot 10^{-3}$	$3.48 \cdot 10^{-3}$
t_1	0.682	0.678	$4.36 \cdot 10^{-3}$	$5.37 \cdot 10^{-3}$
t_3	0.744	0.744	$3.20 \cdot 10^{-3}$	$3.24 \cdot 10^{-3}$

Table 4.2: Means and variances of the simulated posterior samples for the parameter ρ from the Metropolis-Hastings algorithm.

	Mean		Variance	
Process	Sim.	Theor.	Sim.	Theor.
$N(0,1)$	0.857	0.858	$1.49 \cdot 10^{-2}$	$1.46 \cdot 10^{-2}$
$N(0,5)$	0.155	0.155	$4.78 \cdot 10^{-4}$	$4.76 \cdot 10^{-4}$
t_1	0.019	0.019	$7.38 \cdot 10^{-6}$	$7.43 \cdot 10^{-6}$
t_3	0.327	0.327	$2.12 \cdot 10^{-3}$	$2.12 \cdot 10^{-3}$

Table 4.3: Means and variances of the simulated posterior samples for the parameter τ from the Metropolis-Hastings algorithm.

theoretical values for the parameters ρ and τ are shown in tables 4.2 and 4.3 respectively.

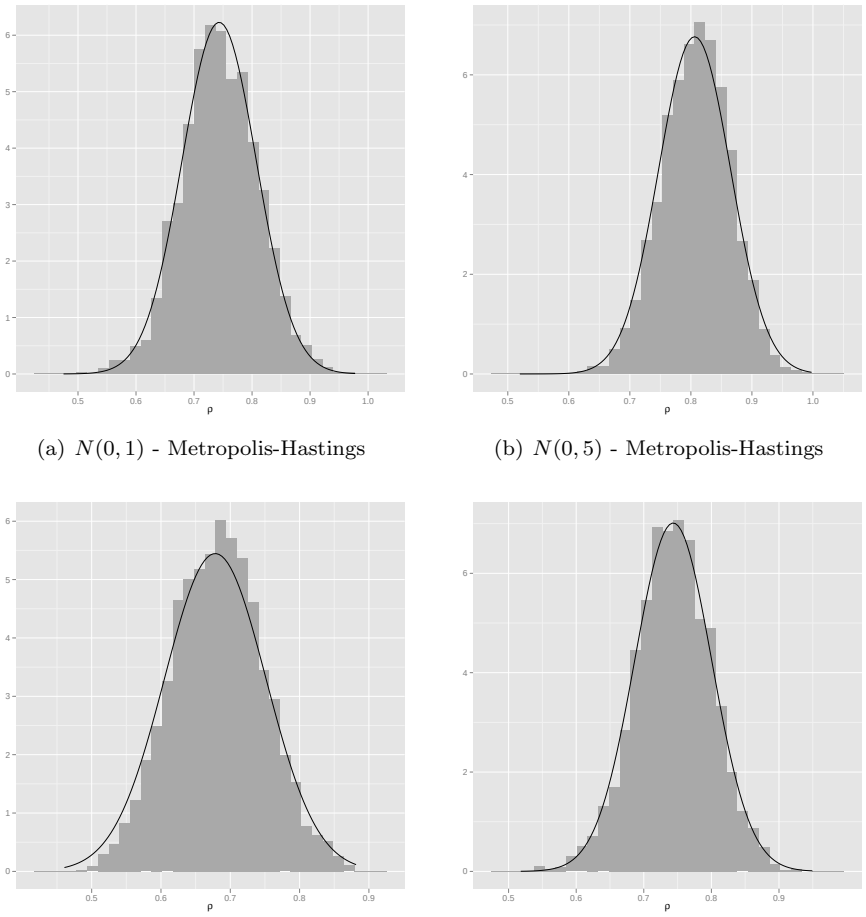
A measure for the effectiveness of the Metropolis-Hastings algorithm is the average acceptance rate. This is the average probability of proposals being accepted. These are listed in table 4.4. The fact that all acceptance rates for the simulations are below 20% shows that the Metropolis-Hastings algorithm performed very inefficiently in this setting, accepting less than 2 out of 10 proposals. The reason for this could be that the exponential distribution is not suited as a proposal distribution for the parameter τ. A gamma distributed proposal for τ would surely perform better if we choose the right hyperparameters for the proposal distribution. This however would mean that we have to use our knowledge about the "real" posterior distribution for the application of the Metropolis-Hastings algorithm which would make it less of a black-box algorithm that can be used with minimal knowledge of the target distribution being simulated.

Process	Acceptance Rate (%)
$N(0,1)$	13.6
$N(0,5)$	15.6
t_1	16.2
t_3	14.8

Table 4.4: Average acceptance rates for the Metropolis-Hastings algorithm.

Figures 4.1 and 4.2 display the histograms of the generated samples along with the densities of the theoretical posterior distributions. Overall it seems that the simulated posterior distribution fits relatively good to the theoretical posterior distribution. The quantile comparison plots in figures 4.3 and 4.4 confirm this belief except for the parameter ρ in the model with t_1 distributed errors. There the quantile-quantile points are S-shaped which indicates that the tails of the simulated distribution are thinner than the tails of the theoretical distribution. Besides that, the simulation results match with the theoretical results in the case of t distributed errors. In this sense the Metropolis-Hastings algorithm is robust against model misspecifications in the error distribution. This should be no surprise since the MCMC methods generate samples of this "theoretic" posterior distribution. The model misspecification does not influence the result on the level of the MCMC methods but on the expected posterior distribution which is implied by the choice of the model.

To assess the convergence of the simulated samples to the theoretical parameter we ran the algorithm multiple times, each time generating a new sample. From this set of samples we calculated the median, 10% and 90% percentiles of the sample mean. The evolution of these statistics with increasing sample size is plotted in figures 4.5 and 4.6. The sample mean converges quickly to the theoretical value for the parameter τ. Convergence for the parameter ρ is not quite as fast and in the case of t_1 distributed errors the evolution of the parameter mean is not stable, even for large sample sizes.



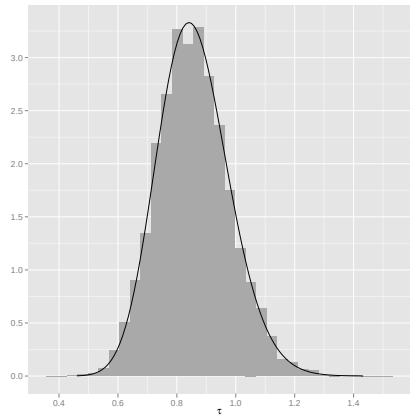
(a) $N(0,1)$ - Metropolis-Hastings

(b) $N(0,5)$ - Metropolis-Hastings

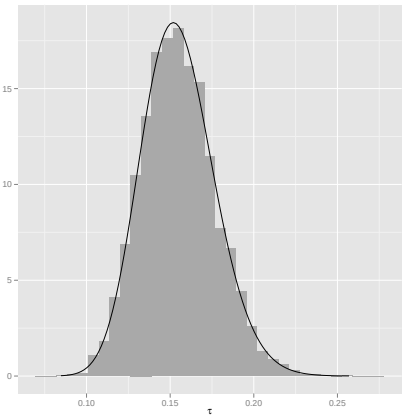
(c) t_1 - Metropolis-Hastings

(d) t_3 - Metropolis-Hastings

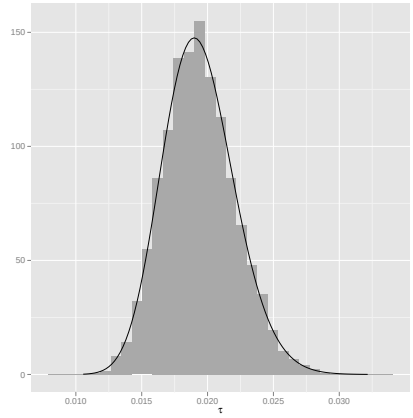
Figure 4.1: Histograms for the simulated posterior samples of the parameter ρ with the theoretical densities.



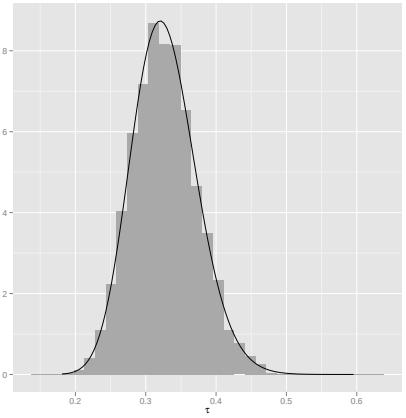
(a) $N(0,1)$ - Metropolis-Hastings



(b) $N(0,5)$ - Metropolis-Hastings

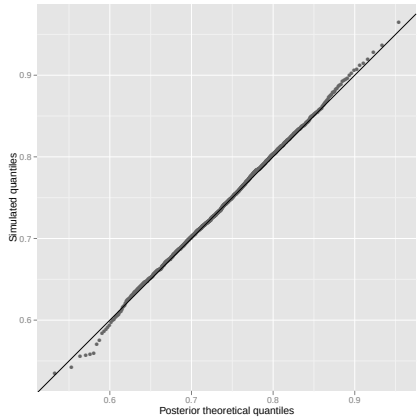


(c) t_1 - Metropolis-Hastings

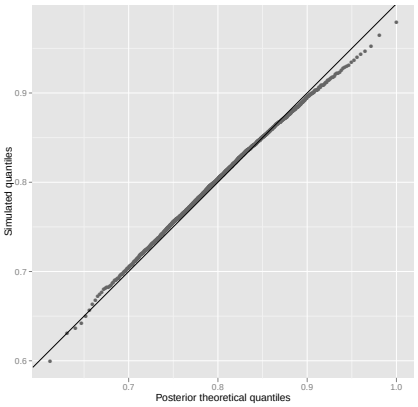


(d) t_3 - Metropolis-Hastings

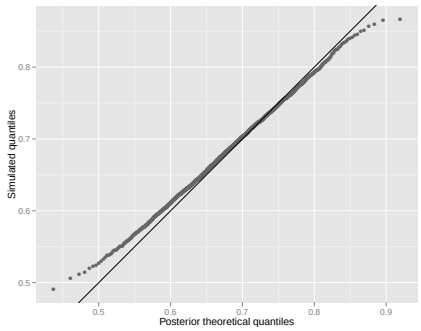
Figure 4.2: Histograms for the simulated posterior samples of the parameter τ with the theoretical densities.



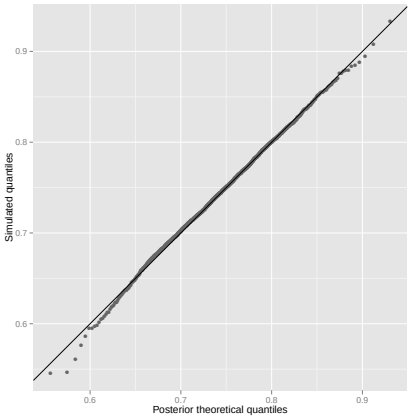
(a) $N(0,1)$ - Metropolis-Hastings



(b) $N(0,5)$ - Metropolis-Hastings

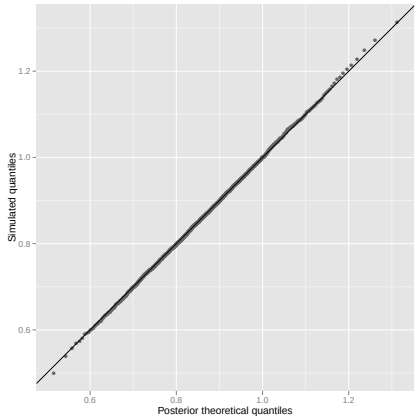


(c) t_1 - Metropolis-Hastings

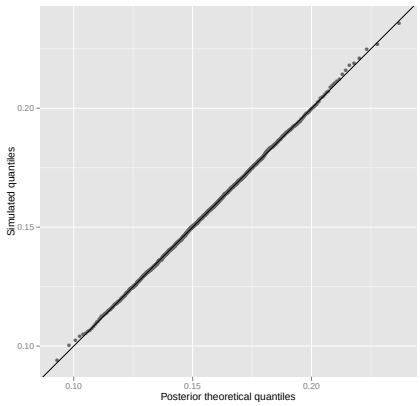


(d) t_3 - Metropolis-Hastings

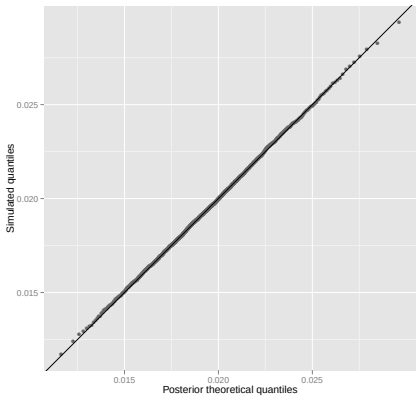
Figure 4.3: Quantile comparisons of the simulated posterior samples of the parameter ρ to the theoretical quantiles.



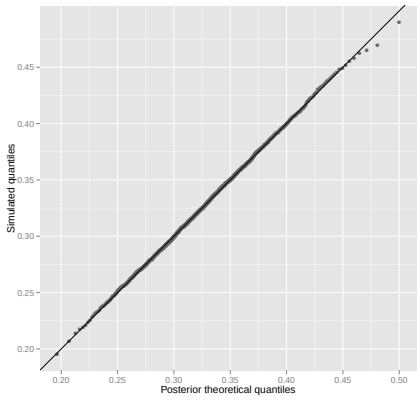
(a) $N(0,1)$ - Metropolis-Hastings



(b) $N(0,5)$ - Metropolis-Hastings

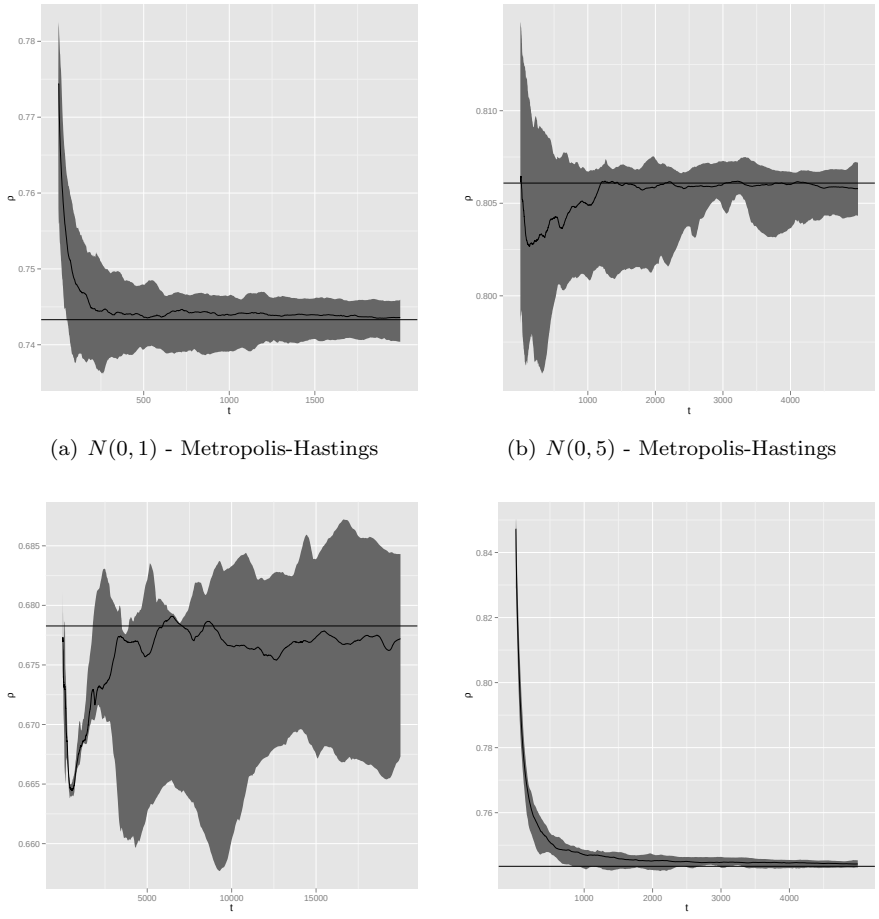


(c) t_1 - Metropolis-Hastings



(d) t_3 - Metropolis-Hastings

Figure 4.4: Quantile comparisons of the simulated posterior samples of the parameter ρ to the theoretical quantiles.



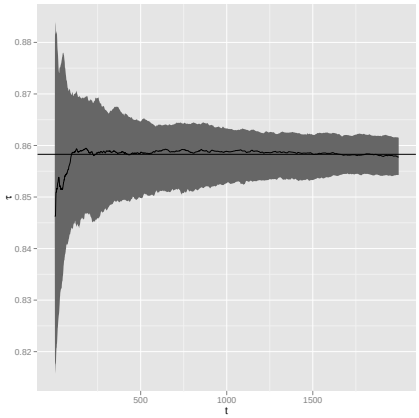
(a) $N(0,1)$ - Metropolis-Hastings

(b) $N(0,5)$ - Metropolis-Hastings

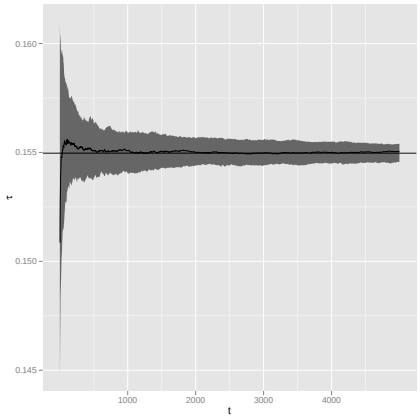
(c) t_1 - Metropolis-Hastings

(d) t_3 - Metropolis-Hastings

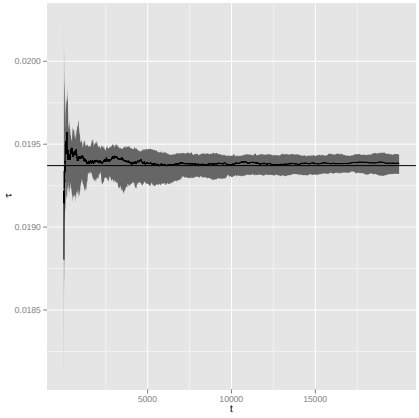
Figure 4.5: Evolution of the simulated sample means for the parameter ρ. The black trajectory is the median of the sample paths and the borders of the grey area are the 10% and 90% percentiles respectively. The horizontal line represents the theoretical posterior mean.



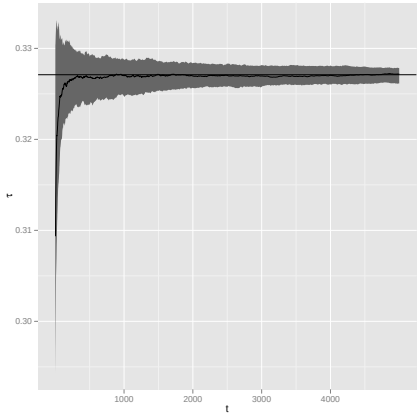
(a) $N(0,1)$ - Metropolis-Hastings



(b) $N(0,5)$ - Metropolis-Hastings



(c) t_1 - Metropolis-Hastings



(d) t_3 - Metropolis-Hastings

Figure 4.6: Evolution of the simulated sample means for the parameter τ. The black trajectory is the median of the sample paths and the borders of the grey area are the 10% and 90% percentiles respectively. The horizontal line represents the theoretical posterior mean.

4.3.2 Gibbs Sampler

We applied the Gibbs sampling algorithm, detailed in section 4.2, to the simulated processes with a sample size of 50000 and a *burn-in* period of 10000. Tables 4.5 and 4.6 list the sample means and variances along with the theoretical values for the parameters ρ and τ.

	Mean		Variance	
Process	Sim.	Theor.	Sim.	Theor.
$N(0,1)$	0.743	0.743	$4.20 \cdot 10^{-3}$	$4.10 \cdot 10^{-3}$
$N(0,5)$	0.805	0.806	$3.56 \cdot 10^{-3}$	$3.48 \cdot 10^{-3}$
t_1	0.677	0.678	$5.49 \cdot 10^{-3}$	$5.37 \cdot 10^{-3}$
t_3	0.742	0.744	$3.31 \cdot 10^{-3}$	$3.24 \cdot 10^{-3}$

Table 4.5: Means and variances of the simulated posterior samples for the parameter ρ from the Gibbs sampler.

	Mean		Variance	
Process	Sim.	Theor.	Sim.	Theor.
$N(0,1)$	0.858	0.858	$1.45 \cdot 10^{-2}$	$1.46 \cdot 10^{-2}$
$N(0,5)$	0.155	0.155	$4.73 \cdot 10^{-4}$	$4.76 \cdot 10^{-4}$
t_1	0.019	0.019	$7.39 \cdot 10^{-6}$	$7.43 \cdot 10^{-6}$
t_3	0.327	0.327	$2.11 \cdot 10^{-3}$	$2.12 \cdot 10^{-3}$

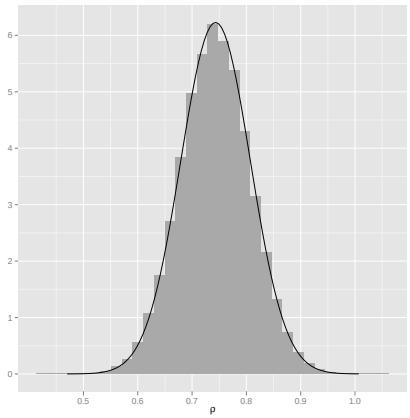
Table 4.6: Means and variances of the simulated posterior samples for the parameter τ from the Gibbs sampler.

The histograms of the generated samples along with the densities of the theoretical posterior distributions are shown in figures 4.7 and 4.8 and match the theoretical densities very well for all processes. This is confirmed by the quantile comparison plots which are shown in figures 4.9 and 4.10. Compared to the Metropolis-Hastings algorithm (see figures 4.1 and 4.2) the Gibbs sampler produces more accurate posterior samples even for t_1 distributed errors. This improved accuracy however is at the cost of having to know the full conditional distributions and the ability to generate samples from them.

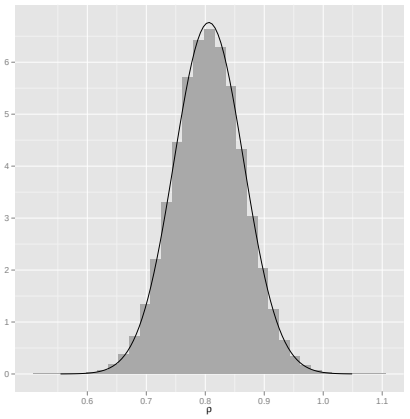
Figures 4.11 and 4.12 show the evolution of the arithmetic means of the simulated parameters. Again, compared to the parameter evolutions of the Metropolis-Hastings algorithm (see figures 4.5 and 4.6), the Gibbs sampler genereated parameter evolutions are much more stable and converge very fast for all processes.

4.4 Conclusions

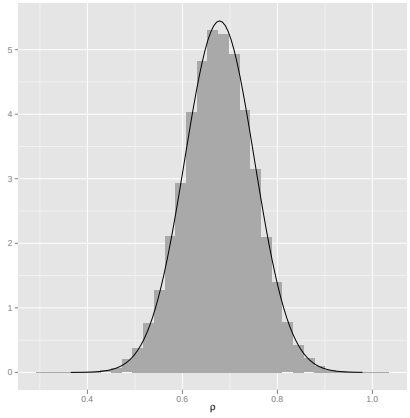
In this chapter we applied the Metropolis-Hastings algorithm and the Gibbs sampler to simulated time series of an AR(1) process. Under the assumption of normally distributed error variables the normal-gamma distribution is conjugate



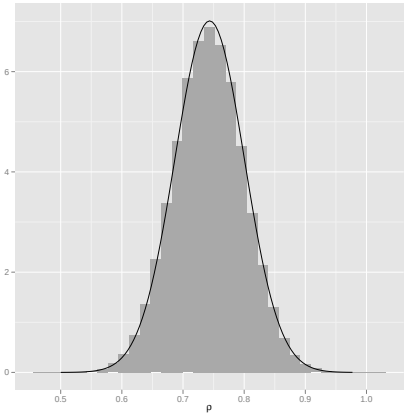
(a) $N(0, 1)$ - Gibbs sampler



(b) $N(0, 5)$ - Gibbs sampler

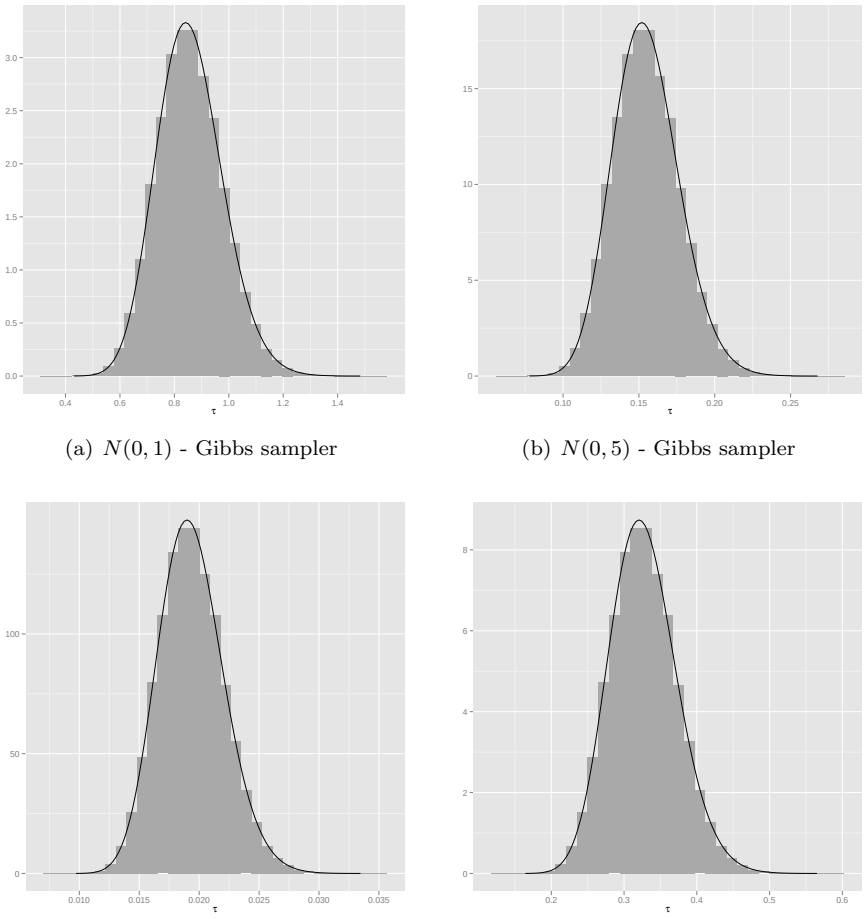


(c) t_1 - Gibbs sampler



(d) t_3 - Gibbs sampler

Figure 4.7: Histograms for the simulated posterior samples of the parameter ρ with the theoretical densities.



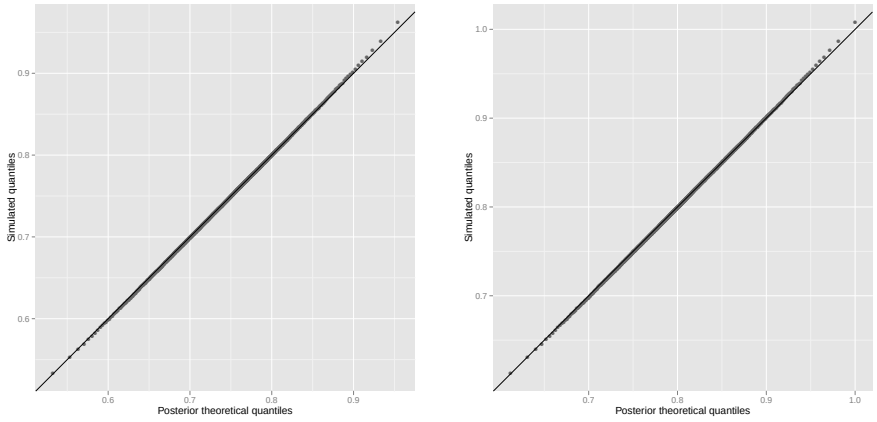
(a) $N(0,1)$ - Gibbs sampler

(b) $N(0,5)$ - Gibbs sampler

(c) t_1 - Gibbs sampler

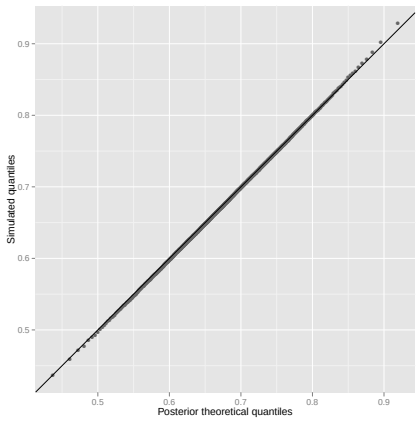
(d) t_3 - Gibbs sampler

Figure 4.8: Histograms for the simulated posterior samples of the parameter τ with the theoretical densities.

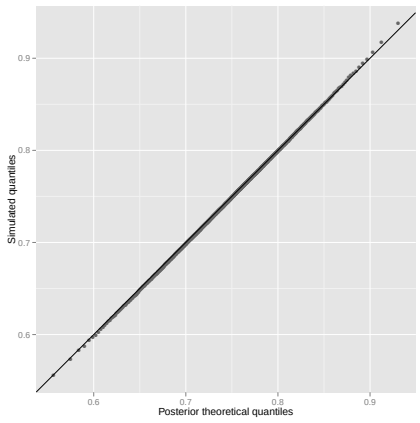


(a) $N(0,1)$ - Gibbs sampler

(b) $N(0,5)$ - Gibbs sampler

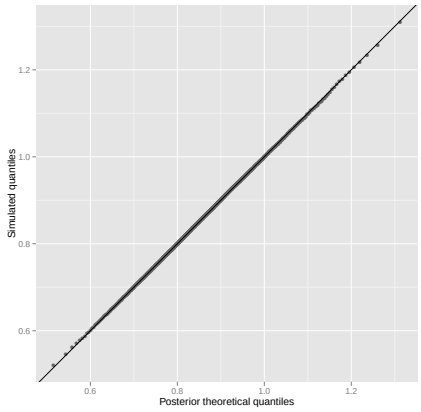


(c) t_1 - Gibbs sampler

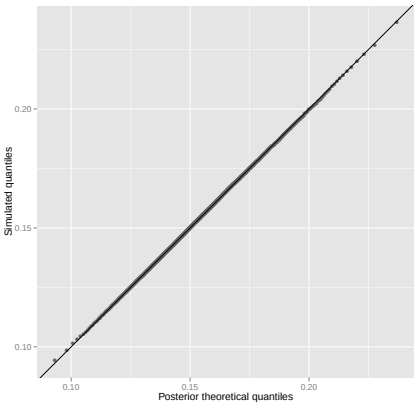


(d) t_3 - Gibbs sampler

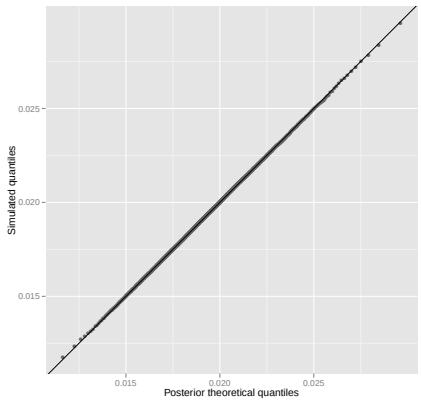
Figure 4.9: Quantile comparisons of the simulated posterior samples of the parameter ρ to the theoretical quantiles.



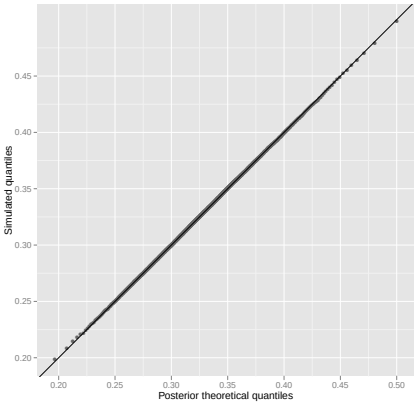
(a) $N(0,1)$ - Gibbs sampler



(b) $N(0,5)$ - Gibbs sampler

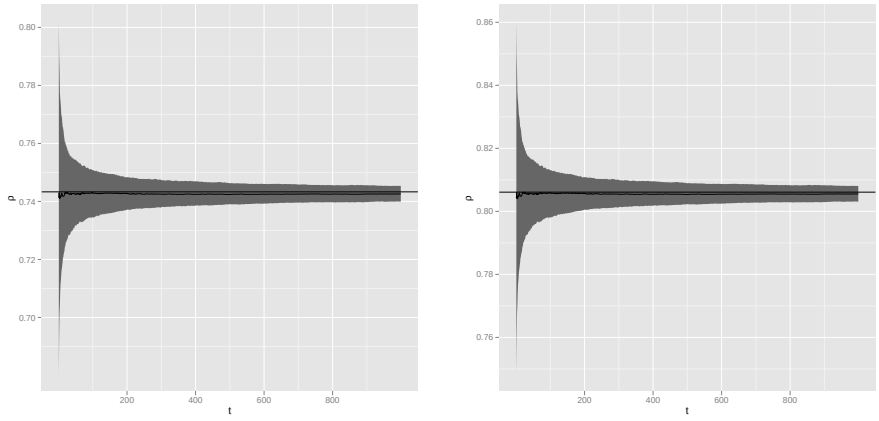


(c) t_1 - Gibbs sampler



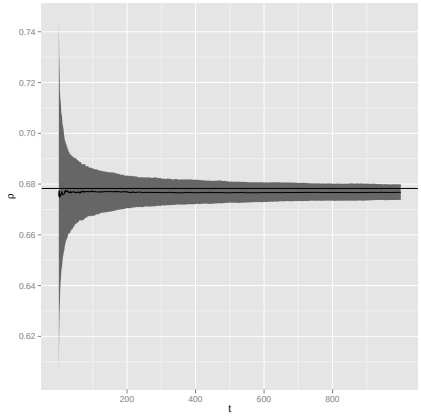
(d) t_3 - Gibbs sampler

Figure 4.10: Quantile comparisons of the simulated posterior samples of the parameter ρ to the theoretical quantiles.

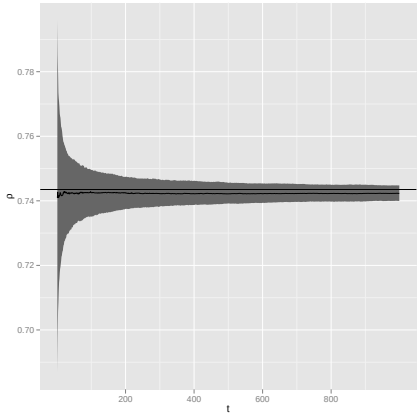


(a) $N(0,1)$ - Gibbs sampler

(b) $N(0,5)$ - Gibbs sampler

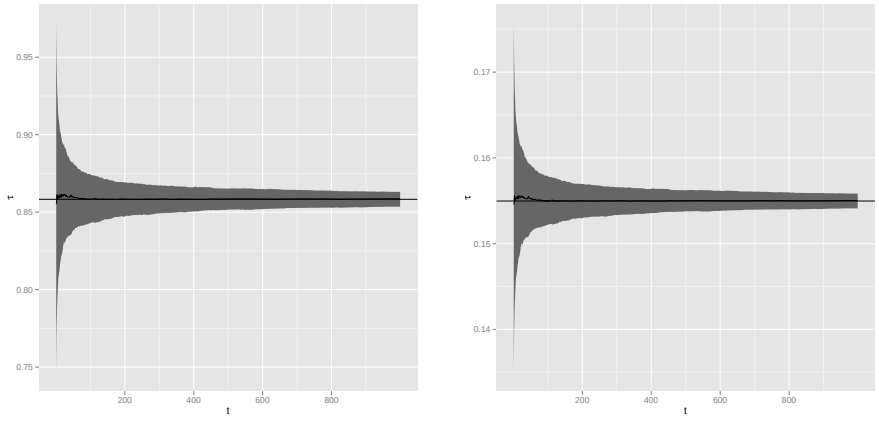


(c) t_1 - Gibbs sampler



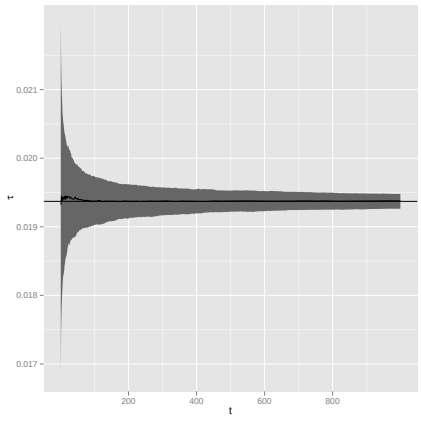
(d) t_3 - Gibbs sampler

Figure 4.11: Evolution of the simulated sample means for the parameter ρ. The black trajectory is the median of the sample paths and the borders of the grey area are the 10% and 90% percentiles respectively. The horizontal line represents the theoretical posterior mean.

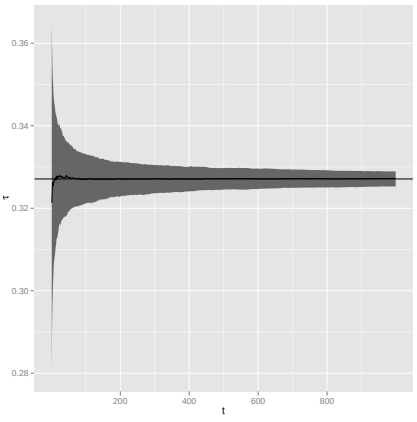


(a) $N(0,1)$ - Gibbs sampler

(b) $N(0,5)$ - Gibbs sampler



(c) t_1 - Gibbs sampler



(d) t_3 - Gibbs sampler

Figure 4.12: Evolution of the simulated sample means for the parameter τ. The black trajectory is the median of the sample paths and the borders of the grey area are the 10% and 90% percentiles respectively. The horizontal line represents the theoretical posterior mean.

to the model which enabled us to compare the results of the MCMC methods with the theoretical posterior distribution of the model.

Both algorithms managed to generate good approximations of the posterior distribution, except the Metropolis-Hastings algorithm in the case of a simulated time series with t_1 distributed errors (see figure 4.1(c)). Here the simulated posterior sample for the parameter ρ had too thin tails as can be seen in figure 4.3(c). A too small sample size and a suboptimal choice for the proposal distribution can explain this and is further supported by the low acceptance rates produced by the Metropolis-Hastings algorithm (see table 4.4).

Compared to the Metropolis-Hastings algorithm, the Gibbs sampler performed better for all simulated time series which is no surprise since it makes use of special knowledge of the posterior distribution – the full conditionals.

The parameter evolution plots (Figures 4.5, 4.6, 4.11 and 4.12), which display the median, 10% and 90% quantiles of the paths for the sample mean for multiple generated posterior samples, clearly show the difference of the stability and the convergence of the posterior mean between the Metropolis-Hastings algorithm and the Gibbs sampler.

To sum up, we can say that there is a certain performance and effectiveness tradeoff between algorithms that require explicit knowledge of the posterior distribution (e.g. the Gibbs sampler) and *black-box* algorithms that lack this requirement (e.g. the Metropolis-Hastings algorithm).

Appendix A

CO_2-Emission: Source Code

Listing A.1: Functions for the Bayesian Analysis of the DLM

```
require(expm)

# A model is a list with elements F, V, G and W
# that correspond to the respective objects in
# the DLM model.

dlm.forecast.prior <- function(model, a, R, steps=1) {
  k = steps
  G <- model$G
  a.k <- G%^%(k-1)%*%a
  R.k <- G%^%(k-1)%*%R%*%t(G%^%(k-1))
  if(k>1) {
    W <- model$W
    for(j in 2:k) {
      R.k <- R.k + G%^%(k-j)%*%W%*%t(G%^%(k-j))
    }
  }
  return(list(a=a.k, R=R.k))
}
dlm.forecast <- function(model, a, R, steps=1) {
  prior <- dlm.forecast.prior(model, a, R, steps)
  F <- model$F
  V <- model$V
  f <- t(F)%*%prior$a
  Q <- t(F)%*%prior$R%*%F + V
  return(list(f=as.numeric(f), Q=as.numeric(Q)))
}
dlm.posterior <- function(y, model, a, R) {
  forecast <- dlm.forecast(model, a, R, 1)
  Q <- forecast$Q
  f <- forecast$f
  F <- model$F
  A <- (R%*%F)/Q
  e <- y - f
  m <- a + A*e
  C <- R - A%*%t(A)*Q
```

```
  return(list(m=m, C=C))
}
dlm.updated.prior <- function(model, m, C) {
  G <- model$G
  W <- model$W
  a <- G%*%m
  R <- G%*%C%*%t(G) + W
  return(list(a=a, R=R))
}
dlm.walkforward <- function(data, dlm.mod) {
  T <- length(data)
  model <- list(F=t(dlm.mod$FF), V=dlm.mod$V, G=dlm.mod$GG,
                W=dlm.mod$W)

  m0 <- dlm.mod$m0
  C0 <- dlm.mod$C0
  m <- matrix(ncol=length(m0), nrow=T)
  C <- list()
  a <- matrix(ncol=length(m0), nrow=T)
  R <- list()
  f <- c()
  Q <- c()
  for(t in 1:T) {
    prior <- dlm.updated.prior(model, m0, C0)
    a[t,] <- prior$a
    R[[t]] <- prior$R
    posterior <- dlm.posterior(data[t], model, a[t,], R[[t]])
    m[t,] <- posterior$m
    C[[t]] <- posterior$C
    m0 <- m[t,]
    C0 <- C[[t]]
    forecast <- dlm.forecast(model, a[t,], R[[t]])
    f[t] <- forecast$f
    Q[t] <- forecast$Q
  }
  return(list(m=m, C=C, a=a, R=R, f=f, Q=Q))
}
```

Listing A.2: Bayesian Analysis of the CO_2 dataset with the DLM

```
require(dlm)
# The dlm package provides functionality
# for generating DLM models and MLE
# which we use to obtain the 'known' variance.

source('dlm.r')

data(co2)
data.ts <- ts(co2, start=start(co2),
              frequency=frequency(co2))

# Genereate the spectrum
spec <- spec.pgram(as.vector(co2), spans=c(3,3),
                   log='no')
```

```
spec.df <- data.frame(freq=spec$freq,
                      spec=spec$spec)

# Build the DLM model with the MLE for V and W
build.fun <- function(x) {
  mod <- dlmModPoly(order=2, m0=c(315,0),
                    C0=diag(c(5,1))) +
         dlmModSeas(12, C0=diag(1,11))
  V(mod) <- exp(x[1])
  W(mod) <- diag(c(exp(x[2:3]),rep(0,11)))
  return(mod)
}
fit <- dlmMLE(data.ts, parm=c(0,0,0), build=build.fun)
dlm.mod <- build.fun(fit$par)

# Walkforward analysis from previous listing
dlm.res <- dlm.walkforward(data.ts, dlm.mod)
res.df <- data.frame(co2=co2, f=dlm.res$f,
                     ci.u=dlm.res$f +
                     qnorm(0.975,sd=sqrt(dlm.res$Q)),
                     ci.l=dlm.res$f -
                     qnorm(0.975,sd=sqrt(dlm.res$Q)))
res.dec.df <- data.frame(co2=co2, trend=dlm.res$a[,1],
                         seasonal=dlm.res$a[,3])

f <- c()
Q <- c()
model <- list(F=t(dlm.mod$FF), V=dlm.mod$V, G=dlm.mod$GG,
              W=dlm.mod$W)
a <- dlm.res$a[dim(dlm.res$a)[1]-36,]
R <- dlm.res$R[[length(dlm.res$R)-36]]
for(k in 1:37) {
  forecast <- dlm.forecast(model, a, R, k)
  f <- c(f, forecast$f)
  Q <- c(Q, forecast$Q)
}

data.ts.v <- as.vector(data.ts)
data.ts.v.pre <- rep(NA, length(data.ts.v))
data.ts.v.post <- rep(NA, length(data.ts.v))
pre <- 1:(length(data.ts.v)-36)
post <- (length(data.ts.v)-36):length(data.ts.v)
data.ts.v.pre[pre] <- data.ts.v[pre]
data.ts.v.post[post] <- data.ts.v[post]
plot.ts(data.ts.v.pre, data.ts.v.post)
f.ts <- rep(NA, length(data.ts.v))
Q.ts <- rep(NA, length(data.ts.v))
f.ts[post] <- f
Q.ts[post] <- Q

ci.l <- f.ts - qnorm(0.975, sd=sqrt(Q.ts))
ci.u <- f.ts + qnorm(0.975, sd=sqrt(Q.ts))
fc.df <- data.frame(pre=data.ts.v.pre,
                    post=data.ts.v.post,
```

```
                    fc=f.ts, ci.u=ci.u, ci.l=ci.l)
```

Appendix B

MCMC-Methods: Source Code

Listing B.1: Model.hpp

```
#ifndef __MODEL__
#define __MODEL__

#include<gsl/gsl_rng.h>
#include<vector>

class Model {
public:
  typedef std::vector<double> param_vec;
  typedef std::vector<double> data_vec;

  Model() {}
  virtual ~Model() {}
};

class MHModel : public Model {
protected:
  const gsl_rng* rng;

public:
  MHModel(const gsl_rng* rng) : rng(rng) {}

  virtual double target_prop(const Model::param_vec& param,
    const Model::data_vec& data) const = 0;

  virtual double proposal_density(const Model::param_vec& y,
    const Model::param_vec& cond,
    const Model::data_vec& data) const = 0;
  virtual Model::param_vec proposal_generator(
    const Model::param_vec& cond,
    const Model::data_vec& data) const = 0;
};
```

```
class GibbsModel : public Model {
protected:
  const gsl_rng* rng;

public:
  GibbsModel(const gsl_rng* rng) : rng(rng) {}

  virtual Model::param_vec full_conditionals_generator(
    const Model::param_vec& current,
    const Model::data_vec& data) const = 0;
};

#endif // __MODEL__
```

Listing B.2: Simulator.hpp

```
#ifndef __SIMULATOR__
#define __SIMULATOR__

#include<cmath>
#include"Model.hpp"
#include"DataLogger.hpp"

class Simulator {
protected:
  Model* model;
  DataLogger* logger;
  const Model::data_vec& data;
  const Model::param_vec& initial_param;
  std::size_t sample_size;
  std::size_t burn_in;
  const gsl_rng* rng;

public:
  Simulator(Model* model,
            DataLogger* logger,
            const Model::param_vec& initial_param,
            const Model::data_vec& data,
            std::size_t sample_size,
            std::size_t burn_in,
            const gsl_rng* rng) :
    model(model), logger(logger),
    initial_param(initial_param),
    data(data),
    sample_size(sample_size),
    burn_in(burn_in),
    rng(rng) {}

  virtual void run() = 0;
};

class MHSimulator : public Simulator {
protected:
  Model::param_vec mh_step(
```

```
    const Model::param_vec& current_param);

  std::size_t acceptance_count;

public:
  MHSimulator(MHModel* model,
              DataLogger* logger,
              const Model::param_vec& initial_param,
              const Model::data_vec& data,
              std::size_t sample_size,
              std::size_t burn_in,
              const gsl_rng* rng) :
    Simulator(static_cast<Model*>(model), logger,
              initial_param,
              data,
              sample_size,
              burn_in,
              rng),
    acceptance_count(0) {}

  virtual void run();

  double get_acceptance_rate() const {
    return this->acceptance_count /
      double(this->sample_size);
  }
  std::size_t get_acceptance_count() const {
    return this->acceptance_count;
  }
};

Model::param_vec MHSimulator::mh_step(
const Model::param_vec& current_param) {
  MHModel* mh_model =
    dynamic_cast<MHModel*>(this->model);
  Model::param_vec new_param =
    mh_model->proposal_generator(current_param, this->data);
  double u = gsl_rng_uniform(this->rng);
  double post_prop_new =
    mh_model->target_prop(new_param, this->data);
  double post_prop_current =
    mh_model->target_prop(current_param, this->data);
  double prop_density_new =
    mh_model->proposal_density(new_param, current_param,
                               this->data);
  double prop_density_current =
    mh_model->proposal_density(current_param, new_param,
                               this->data);
  double rho = std::min(1., post_prop_new/post_prop_current *
                        prop_density_current /
                        prop_density_new);
  if(u <= rho) {
    this->acceptance_count++;
    return new_param;
```

```
  }
  else
    return current_param;
}

void MHSimulator::run() {
  Model::param_vec current_param = this->initial_param;
  for(std::size_t i = 0; i < this->sample_size; ++i) {
    current_param = this->mh_step(current_param);
    if(i >= this->burn_in)
      this->logger->log_param_vec(current_param);
  }
}

class GibbsSimulator : public Simulator {
public:
  GibbsSimulator(GibbsModel* model,
                 DataLogger* logger,
                 const Model::param_vec& initial_param,
                 const Model::data_vec& data,
                 std::size_t sample_size,
                 std::size_t burn_in,
                 const gsl_rng* rng) :
    Simulator(static_cast<Model*>(model), logger,
              initial_param,
              data,
              sample_size,
              burn_in,
              rng) {}

  virtual void run();
};

void GibbsSimulator::run() {
  Model::param_vec current_param =
    this->initial_param;
  for(std::size_t i = 0; i < this->sample_size; i++) {
    current_param = dynamic_cast<GibbsModel*>(this->model)->
      full_conditionals_generator(current_param, this->data);
    if(i >= this->burn_in)
      this->logger->log_param_vec(current_param);
  }
}

#endif // __SIMULATOR__
```

Listing B.3: DataLogger.hpp

```
#ifndef __DATA_LOGGER__
#define __DATA_LOGGER__

#include<string>
#include<vector>
#include<list>
```

```
#include<map>
#include<fstream>
#include<cmath>

#include <gsl/gsl_sort.h>
#include <gsl/gsl_statistics.h>

#include"Model.hpp"

class DataLogger {
public:
  DataLogger() {}

  virtual void log_param_vec(
    const Model::param_vec& param) = 0;
};

class MemoryLogger : public DataLogger {
protected:
  typedef std::list<Model::param_vec> param_list;
  typedef std::map<std::string, double> var_map;

  param_list parameters;
  var_map variables;

public:
  MemoryLogger() {}

  virtual void log_param_vec(
  const Model::param_vec& param) {
    this->parameters.push_back(param);
  }
};

class CSVLogger : public DataLogger {
protected:
  typedef std::map<std::string, double> var_map;

  std::ofstream file;
  var_map variables;

public:
  CSVLogger(const std::string& filename) :
    file(filename.c_str()) {}

  ~CSVLogger() {this->file.flush();this->file.close();}

  virtual void log_param_vec(const Model::param_vec& param) {
    for(Model::param_vec::const_iterator it = param.begin();
        it != param.end(); it++) {
      this->file << *it;
      if(it != (--param.end())) {
        this->file << ",";
      }
```

```
      else {
        this->file << "\n";
      }
    }
    this->file.flush();
  }
};

class TrajectoryLogger : public DataLogger {
protected:
  typedef std::vector<double**> grids_t;

  std::size_t p, tc, T;
  std::size_t curr_traj, curr_t;
  grids_t param_trajectories;
  std::ofstream file;

  grids_t setup_grids(std::size_t gc, std::size_t yc,
                      std::size_t xc) {
    grids_t grids(gc);
    for(std::size_t i = 0; i < gc; ++i) {
      grids[i] = new double*[yc];
      for(std::size_t t = 0; t < yc; ++t) {
        grids[i][t] = new double[this->T];
        for(std::size_t j = 0; j < this->T; ++j) {
          grids[i][t][j] = 0.0;
        }
      }
    }
    return grids;
  }

  void free_grids(grids_t& grids, std::size_t gc,
                  std::size_t yc) {
    for(std::size_t i = 0; i < gc; ++i) {
      for(std::size_t t = 0; t < yc; ++t) {
        delete[] grids[i][t];
      }
      delete[] grids[i];
    }
    grids.clear();
  }

public:
  TrajectoryLogger(std::size_t param_count,
                   std::size_t trajectory_count,
                   std::size_t sample_size,
                   const std::string& filename) :
    p(param_count), tc(trajectory_count),
    T(sample_size), file(filename.c_str()) {
    this->param_trajectories =
      this->setup_grids(this->p, this->tc, this->T);
    this->curr_traj = 0;
    this->curr_t = 0;
```

```
}

virtual ~TrajectoryLogger() {
  this->free_grids(this->param_trajectories,
                   this->p, this->tc);
  this->file.flush();
  this->file.close();
}

virtual void log_param_vec(const Model::param_vec& param) {
  if(this->curr_t >= this->T ||
     this->curr_traj >= this->tc) {
    return;
  }
  for(std::size_t i = 0; i < this->p; ++i) {
    this->param_trajectories[i][this->curr_traj]
      [this->curr_t] =
      param[i];
  }
  this->curr_t++;
}

void start_next_trajectory() {
  this->curr_traj++;
  this->curr_t = 0;
}

grids_t get_trajectory_stats() {
  grids_t stats = this->setup_grids(
    this->param_trajectories.size(), 3, this->T-1);
  double* curr_mean = new double[this->tc];
  for(std::size_t i = 0; i < this->p; ++i) {
    for(std::size_t t = 1; t < this->T; ++t) {
      for(std::size_t tr = 0; tr < this->tc; ++tr) {
        if(t == 1) {
          curr_mean[tr] = (
            this->param_trajectories[i][tr][t-1] +
            this->param_trajectories[i][tr][t]) / 2.0;
        } else {
          curr_mean[tr] = (
            t*curr_mean[tr] +
            this->param_trajectories[i][tr][t]) / (t+1);
        }
      }
      gsl_sort(curr_mean, 1, this->tc);
      double mean_q_10 =
        gsl_stats_quantile_from_sorted_data(
          curr_mean, 1, this->tc, 0.1);
      double mean_q_50 =
        gsl_stats_quantile_from_sorted_data(
          curr_mean, 1, this->tc, 0.5);
      double mean_q_90 =
        gsl_stats_quantile_from_sorted_data(
          curr_mean, 1, this->tc, 0.9);
```

```
        stats[i][0][t-1] = mean_q_10;
        stats[i][1][t-1] = mean_q_50;
        stats[i][2][t-1] = mean_q_90;
      }
    }
    return stats;
  }

  void write_to_file() {
    grids_t stats = this->get_trajectory_stats();
    for(std::size_t t = 0; t < this->T-1; ++t) {
      for(std::size_t i = 0; i < this->p; ++i) {
        this->file << stats[i][0][t] << ",";
        this->file << stats[i][1][t] << ",";
        this->file << stats[i][2][t];
        if(i < this->p-1) {
          this->file << ",";
        }
      }
      this->file << "\n";
    }

    this->free_grids(stats,
                     this->param_trajectories.size(), 3);
  }
};

#endif // __DATA_LOGGER__
```

Listing B.4: AR1.hpp

```
#ifndef __AR1__
#define __AR1__

#include<cmath>
#include<gsl/gsl_randist.h>

class GibbsAR1Model : public GibbsModel {
public:
  GibbsAR1Model(const gsl_rng* rng) : GibbsModel(rng) {}

  virtual Model::param_vec full_conditionals_generator(
  const Model::param_vec& param,
  const Model::data_vec& data) const {
    Model::param_vec theta(2);
    double a = 1.0;
    double b = 1.0;
    double m = 0.0;
    double d_2 = 4.0;
    std::size_t n = data.size();
    double ssq = 0.0;
    double ssq2 = 0.0;
    double ssq3 = 0.0;
    for(std::size_t t = 1; t < n; ++t) {
```

```
      ssq += std::pow(data[t] - param[0]*data[t-1], 2);
    }
    for(std::size_t t = 0; t < n; ++t) {
      ssq2 += std::pow(data[t], 2);
      ssq3 += data[t]*data[t-1];
    }
    double a_p = a + 0.5*(n);
    double b_p = b + 0.5 *
      (ssq + std::pow(param[0]-m, 2)/d_2);
    double d_2_p = 1.0 / (1./d_2 + ssq2);
    double m_p = ssq3 / ssq2;
    theta[1] = gsl_ran_gamma(this->rng, a_p, 1./b_p);
    theta[0] = m_p +
      gsl_ran_gaussian(this->rng,
                       std::sqrt(d_2_p / theta[1]));

    return theta;
  }
};

class MHAR1Model : public MHModel {
protected:
  mutable double tau_exp;
  mutable double d_2;

  void calc_a_b(const Model::data_vec& data) const {
    std::size_t T = data.size();
    double ssq1 = 0.0;
    double ssq2 = 0.0;
    for(std::size_t t = 1; t < T; t++) {
      ssq1 += data[t]*data[t-1];
      ssq2 += std::pow(data[t], 2.0);
    }
    double rho = ssq1 / ssq2;
    double ssq = 0.0;
    for(std::size_t t = 1; t < T; t++) {
      ssq += std::pow(data[t] - rho*data[t-1], 2.0);
    }
    this->tau_exp = (T-2) / ssq;
    this->d_2 = 1.0 / (ssq2);
  }

public:
  MHAR1Model(const gsl_rng* rng) : MHModel(rng) {
    this->tau_exp = 1e-99;
    this->d_2 = 1e-99;
  }

  virtual double target_prop(
  const Model::param_vec& param,
  const Model::data_vec& data) const {
    double a = 1.0;
    double b = 1.0;
    double m = 0.0;
```

```
    double d_2 = 4.0;
    std::size_t T = data.size();
    double ssq = 0.0;
    for(std::size_t t = 1; t < T; t++) {
      ssq += std::pow(data[t] - param[0]*data[t-1],
                      2.0);
    }
    return std::pow(param[1],
                    a+(T-2)/2.0) *
                    std::exp(-0.5*param[1] *
                      (ssq + 2*b +
                        std::pow(param[0]-m, 2)/d_2));
  }

  virtual double proposal_density(
  const Model::param_vec& y,
  const Model::param_vec& cond,
  const Model::data_vec& data) const {
    if(this->tau_exp <= 1e-99 || this->d_2 <= 1e-99) {
      this->calc_a_b(data);
    }
    return gsl_ran_exponential_pdf(y[1], this->tau_exp) *
      gsl_ran_gaussian_pdf(y[0] - cond[0],
                           std::sqrt(this->d_2));
  }

  virtual Model::param_vec proposal_generator(
  const Model::param_vec& cond,
  const Model::data_vec& data) const {
    if(this->tau_exp <= 1e-99 || this->d_2 <= 1e-99) {
      this->calc_a_b(data);
    }
    Model::param_vec theta(2);
    theta[1] = gsl_ran_exponential(this->rng, this->tau_exp);
    theta[0] = cond[0] +
      gsl_ran_gaussian(this->rng,
                       std::sqrt(this->d_2));
    return theta;
  }
};

void calc_rho_tau(const Model::data_vec& data,
                  double* rho, double* tau) {
  std::size_t T = data.size();
  double ssq1 = 0.0;
  double ssq2 = 0.0;
  for(std::size_t t = 1; t < T; t++) {
    ssq1 += data[t]*data[t-1];
    ssq2 += std::pow(data[t], 2.0);
  }
  *rho = ssq1 / ssq2;
  double ssq = 0.0;
  for(std::size_t t = 1; t < T; t++) {
    ssq += std::pow(data[t] - *rho*data[t-1], 2.0);
```

```
  }
  *tau = (T-2) / ssq;
}

#endif // __AR1__
```

Listing B.5: ar_mh.cpp

```
#include<vector>
#include<sstream>
#include<fstream>
#include<string>
#include<iostream>
#include<gsl/gsl_randist.h>

#include"Simulator.hpp"
#include"AR1.hpp"

int main(int argc, char** argv) {
  if(argc < 2) return -1;
  std::string data_filename(argv[1]);
  std::ifstream ifs(data_filename.c_str());
  std::cout << "Reading␣data␣from␣file␣" <<
    data_filename << "␣...";

  std::istringstream sstr;
  std::string buf;
  double val;
  Model::data_vec data;
  while(!std::getline(ifs, buf).eof()) {
    sstr.clear();
    sstr.str(buf);
    sstr >> val;
    data.push_back(val);
  }
  ifs.close();
  std::cout << "␣loaded␣" << data.size() <<
    "␣rows." << std::endl;

  std::size_t sample_size = 10000;
  std::size_t burn_in = 1000;
  if(argc >= 3) {
    sstr.clear();
    sstr.str(argv[2]);
    sstr >> sample_size;
  }
  if(argc >= 4) {
    sstr.clear();
    sstr.str(argv[3]);
    sstr >> burn_in;
  }

  std::cout << "Creating␣" << sample_size <<
    "␣samples" << std::endl;
```

```
  std::cout << "Burn␣in␣period␣is␣" <<
    burn_in << std::endl;

  gsl_rng_env_setup();
  const gsl_rng_type * T = gsl_rng_default;
  gsl_rng * rng = gsl_rng_alloc(T);
  Model::param_vec initial_param(2);
  initial_param[0] = 0.8;
  initial_param[1] = 0.0;

  MHAR1Model* model = new MHAR1Model(rng);
  std::string results_file(data_filename+"_mh");
  CSVLogger* logger = new CSVLogger(results_file);
  MHSimulator sim(model, logger, initial_param,
                  data, sample_size+burn_in, burn_in, rng);
  sim.run();

  std::cout << "Acceptances:␣" <<
    sim.get_acceptance_count() << std::endl;
  std::cout << "Accpetance␣rate:␣" <<
    sim.get_acceptance_rate() << std::endl;

  gsl_rng_free(rng);
  return 0;
}
```

Listing B.6: ar_gibbs.cpp

```
#include<vector>
#include<sstream>
#include<fstream>
#include<string>
#include<iostream>
#include<gsl/gsl_randist.h>

#include"Simulator.hpp"
#include"AR1.hpp"

int main(int argc, char** argv) {
  if(argc < 2) return -1;
  std::string data_filename(argv[1]);
  std::ifstream ifs(data_filename.c_str());
  std::cout << "Reading␣data␣from␣file␣" <<
    data_filename << "␣...";

  std::istringstream sstr;
  std::string buf;
  double val;
  Model::data_vec data;
  while(!std::getline(ifs, buf).eof()) {
    sstr.clear();
    sstr.str(buf);
    sstr >> val;
    data.push_back(val);
```

```
  }
  ifs.close();
  std::cout << "␣loaded␣" << data.size() <<
    "␣rows." << std::endl;

  std::size_t sample_size = 10000;
  std::size_t burn_in = 1000;
  if(argc >= 3) {
    sstr.clear();
    sstr.str(argv[2]);
    sstr >> sample_size;
  }
  if(argc >= 4) {
    sstr.clear();
    sstr.str(argv[3]);
    sstr >> burn_in;
  }

  std::cout << "Creating␣" << sample_size <<
    "␣samples" << std::endl;
  std::cout << "Burn␣in␣period␣is␣" <<
    burn_in << std::endl;

  gsl_rng_env_setup();
  const gsl_rng_type * T = gsl_rng_default;
  gsl_rng * rng = gsl_rng_alloc(T);
  Model::param_vec initial_param(2);
  initial_param[0] = 0.5;
  initial_param[1] = 1.;

  GibbsAR1Model* model = new GibbsAR1Model(rng);
  std::string results_file(data_filename+"_gibbs");
  CSVLogger* logger = new CSVLogger(results_file);
  GibbsSimulator sim(model, logger, initial_param,
                     data, sample_size+burn_in,
                     burn_in, rng);
  sim.run();

  gsl_rng_free(rng);
  return 0;
}
```

Listing B.7: ar_mh_param_evolution.cpp

```
#include<vector>
#include<sstream>
#include<fstream>
#include<string>
#include<iostream>
#include<gsl/gsl_randist.h>

#include"Simulator.hpp"
#include"AR1.hpp"
```

```
int main(int argc, char** argv) {
  if(argc < 2) return -1;
  std::string data_filename(argv[1]);
  std::ifstream ifs(data_filename.c_str());
  std::cout << "Reading␣data␣from␣file␣" <<
    data_filename << "␣...";

  std::istringstream sstr;
  std::string buf;
  double val;
  Model::data_vec data;
  while(!std::getline(ifs, buf).eof()) {
    sstr.clear();
    sstr.str(buf);
    sstr >> val;
    data.push_back(val);
  }
  ifs.close();
  std::cout << "␣loaded␣" << data.size() <<
    "␣rows." << std::endl;

  std::size_t sample_size = 10000;
  std::size_t sim_paths = 1000;
  std::size_t burn_in = 0;
  if(argc >= 3) {
    sstr.clear();
    sstr.str(argv[2]);
    sstr >> sample_size;
  }
  if(argc >= 4) {
    sstr.clear();
    sstr.str(argv[3]);
    sstr >> sim_paths;
  }

  std::cout << "Simulating␣" << sim_paths <<
    "␣trajectories" << std::endl;
  std::cout << "each␣with␣" << sample_size <<
    "␣samples." << std::endl;

  double rho;
  double tau;
  calc_rho_tau(data, &rho, &tau);

  gsl_rng_env_setup();
  const gsl_rng_type * T = gsl_rng_default;
  gsl_rng* rng = gsl_rng_alloc(T);

  double avg_acceptance = 0.0;
  std::string results_file(
    data_filename+"_mh_pe");
  TrajectoryLogger* logger =
    new TrajectoryLogger(
      2, sim_paths, sample_size, results_file);
```

```
  for(std::size_t i = 0; i < sim_paths; ++i) {
    Model::param_vec initial_param(2);
    initial_param[0] = rho;
    initial_param[1] = tau;

    MHAR1Model* model = new MHAR1Model(rng);
    MHSimulator sim(model, logger, initial_param,
                    data, sample_size+burn_in,
                    burn_in, rng);
    sim.run();
    avg_acceptance += sim.get_acceptance_rate();
    logger->start_next_trajectory();
    delete model;
  }

  logger->write_to_file();
  delete logger;

  std::cout << "Average␣accpetance␣rate:␣" <<
    avg_acceptance / sim_paths << std::endl;

  gsl_rng_free(rng);
  return 0;
}
```

Listing B.8: ar_gibbs_param_evolution.cpp

```
#include<vector>
#include<sstream>
#include<fstream>
#include<string>
#include<iostream>
#include<gsl/gsl_randist.h>

#include"Simulator.hpp"
#include"AR1.hpp"

int main(int argc, char** argv) {
  if(argc < 2) return -1;
  std::string data_filename(argv[1]);
  std::ifstream ifs(data_filename.c_str());
  std::cout << "Reading␣data␣from␣file␣" <<
    data_filename << "␣...";

  std::istringstream sstr;
  std::string buf;
  double val;
  Model::data_vec data;
  while(!std::getline(ifs, buf).eof()) {
    sstr.clear();
    sstr.str(buf);
    sstr >> val;
    data.push_back(val);
  }
```

```
  ifs.close();
  std::cout << "␣loaded␣" << data.size() <<
    "␣rows." << std::endl;

  std::size_t sample_size = 10000;
  std::size_t sim_paths = 1000;
  std::size_t burn_in = 0;
  if(argc >= 3) {
    sstr.clear();
    sstr.str(argv[2]);
    sstr >> sample_size;
  }
  if(argc >= 4) {
    sstr.clear();
    sstr.str(argv[3]);
    sstr >> sim_paths;
  }

  std::cout << "Simulating␣" << sim_paths <<
    "␣trajectories" << std::endl;
  std::cout << "each␣with␣" << sample_size <<
    "␣samples." << std::endl;

  double rho;
  double tau;
  calc_rho_tau(data, &rho, &tau);

  gsl_rng_env_setup();
  const gsl_rng_type * T = gsl_rng_default;
  gsl_rng * rng = gsl_rng_alloc(T);

  std::string results_file(data_filename+"_gibbs_pe");
  TrajectoryLogger* logger = new TrajectoryLogger(
    2, sim_paths, sample_size, results_file);
  for(std::size_t i = 0; i < sim_paths; ++i) {
    Model::param_vec initial_param(2);
    initial_param[0] = rho;
    initial_param[1] = tau;

    GibbsAR1Model* model = new GibbsAR1Model(rng);
    GibbsSimulator sim(model, logger, initial_param,
                       data, sample_size+burn_in,
                       burn_in, rng);
    sim.run();
    logger->start_next_trajectory();
    delete model;
  }

  logger->write_to_file();
  delete logger;

  gsl_rng_free(rng);
  return 0;
}
```

Bibliography

[1] K.B. Athreya and P. Ney. A new approach to the limit theory of recurrent markov chains. *Trans. Amer. Math. Soc.*, 245:493–501, 1978.

[2] P. Billingsley. *Probability and Measure*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2012.

[3] J. Elstrodt. *Maß- und Integrationstheorie*. Springer-Lehrbuch. Springer Verlag, 2011.

[4] W.R. Gilks, N.G. Best, and K.K.C. Tan. Adaptive rejection metropolis sampling within gibbs sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 44(4):455–472, 1995.

[5] J.D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[6] C. D. Keeling and T. P. Whorf. Mauna loa atmospheric co2 concentration.

[7] K.R. Koch. *Introduction to Bayesian Statistics*. Springer, 2007.

[8] N. Kusolitsch. *Maß- und Wahrscheinlichkeitstheorie: Eine Einführung*. Springer Verlag, 2011.

[9] H.D. Lopes. Bayesian econometrics lab 1. http://faculty.chicagobooth.edu/hedibert.lopes/teaching/BayesianEconometrics/lab1.html.

[10] S.P. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability*. Springer, 1993.

[11] P. Müller. Alternatives to the gibbs sampling scheme. Technical report, Institute of Statistics and Decision Sciences, Duke University, 1993.

[12] C. Neustädter. A bayesian contribution to time series analysis. Diploma thesis, Vienna UT, 2008.

[13] A. Pole, M. West, and J. Harrison. *Applied Bayesian forecasting and times series analysis*. Texts in statistical science. Chapman and Hall, 1994.

[14] C.P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer texts in statistics. Springer, 1999.

[15] C.P. Robert and G. Casella. *Introducing Monte Carlo Methods with R*. Use R! Springer, 2010.

[16] P. Schönfeld. *Methoden der Ökonometrie.* Number Bd. 1 in Vahlens Handbücher der Wirtschafts- und Sozialwissenschaften. Vahlen, 1969.

[17] M. West and J. Harrison. *Bayesian forecasting and dynamic models.* Springer series in statistics. Springer, 1989.