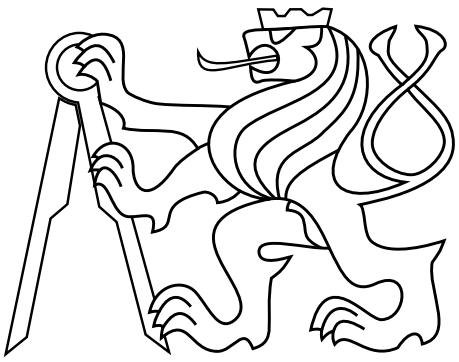
Master Thesis

Bayesian Parameter Estimation of State-Space Models with Intractable Likelihood

Bc. Tomáš Kala
Supervisor: Ing. Kamil Dedecius, PhD.

May 2019



Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kala** Jméno: **Tomáš** Osobní číslo: **434690**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Bioinformatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Bayesovské odhadování parametrů stavových modelů při nedostupné věrohodnostní funkci

Název diplomové práce anglicky:

Bayesian parameter estimation of state-space models with intractable likelihood

Pokyny pro vypracování:

Stavové modely představují velmi populární formalismus vhodný pro popis celé řady různých náhodných procesů, od časových řad po aplikace v teorii řízení. Pokud tyto modely neobsahují statické parametry, lze pro jejich odhad použít např. Kalmanův filtr a jeho varianty, dále particle filtraci aj. Pokud ovšem statické parametry obsahují, tyto filtry zpravidla nekonvergují a nezbývá, než přikročit k optimalizačním technikám typu maximalizace věrohodnosti či particle Markov chain Monte Carlo. Další komplikace nastávají, pokud navíc není věrohodnostní funkce pro pozorovanou veličinu dostupná, nebo je nepřesná či příliš komplikovaná. Diplomová práce je specificky zaměřena poslední zmíněnou problematiku.
Specifické pokyny
1. Seznamte se s metodami pro odhadování stavových modelů pomocí kalmanovské filtrace a sekvenční Monte Carlo filtrace. Nastudujte problematiku statických parametrů a jejich odhadu.
2. Proveďte rešerši ohledně využití daných metod v oblasti bioinformatiky, například v modelování buněčných procesů.
3. Seznamte se s metodami ABC - Approximate Bayesian Computation a jejich využití ve filtraci stavových modelů.
4. Navrhněte vhodný způsob odhadování statických parametrů stavových modelů s využitím metod ABC.
5. Experimentálně (na vhodném problému z oblasti molekulární biologie) a případně teoreticky ověřte vlastnosti navržené metody, diskutujte její vlastnosti a navrhněte další možné směry výzkumu.

Seznam doporučené literatury:

1] C. C. Drovandi, A. N. Pettitt, and R. A. McCutchan, “Exact and approximate Bayesian inference for low integer-valued time series models with intractable likelihoods,” Bayesian Anal., vol. 11, no. 2, pp. 325–352, 2016.
[2] S. Martin, A. Jasra, S. S. Singh, N. Whiteley, P. Del Moral, and E. McCoy, “Approximate Bayesian Computation for Smoothing,” Stoch. Anal. Appl., vol. 32, no. 3, pp. 397–420, 2014.
[3] T. B. Schön, A. Svensson, L. Murray, and F. Lindsten, “Probabilistic learning of nonlinear dynamical systems using sequential Monte Carlo,” Mech. Syst. Signal Process., vol. 104, pp. 866–883, May 2018.
[4] C. Andrieu, A. Doucet, and R. Holenstein, “Particle Markov chain Monte Carlo methods,” J. R. Stat. Soc. Ser. B (Statistical Methodol., vol. 72, no. 3, pp. 269–342, Jun. 2010.
[5] K. Dedecius, “Adaptive kernels in approximate filtering of state-space models,” Int. J. Adapt. Control Signal Process., vol. 31, no. 6, pp. 938–952, Jun. 2017.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Kamil Dedecius, Ph.D., ÚTIA AV ČR

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **04.02.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

Ing. Kamil Dedecius, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Abstract

State-space models (SSMs) are widely used to formalize partially-observed random processes found e.g. in biology, econometrics and signal processing. Given a sequence of observed variables, the interest is to infer a corresponding sequence of latent states assumed to have generated the observations. This procedure is known as filtering. When the SSM is parameterized by a static parameter in addition to the dynamic states, the inference must target both components. The problem then becomes considerably more complex, and the filters typically do not converge. Unless the SSM is linear and Gaussian, its likelihood is intractable, and straightforward inference of the static parameter is not possible. It has been shown that the particle filter can be used as an unbiased estimator of this likelihood even in non-linear models, but the method requires the SSM observation model to be specified as a probability density function. In applications, one is typically in possession of a means to simulate new observations, but not to evaluate their probabilities. Attempts to fit arbitrary probability distributions to the observations typically lead to the particle filter collapsing. Inspired by the techniques of Approximate Bayesian Computation (ABC), this thesis derives an ABC-based filter, which is able to estimate the likelihood even when the observation model is not probabilistic. The performance of the derived algorithm is first demonstrated on a simulation study. Next, the method is applied to a molecular biology problem describing a simplified prokaryotic auto-regulatory network.

Keywords: State-space model, particle filter, Approximate Bayesian Computation, auto-regulation.

Abstrakt

Stavové modely představují široce používaný formalismus pro popis částečně pozorovaných náhodných procesů vyskytujících se např. v biologii, ekonometrii a zpracování signálu. Cílem filtrace je odhadnout sekvenci skrytých stavů, o níž předpokládáme, že vygenerovala sekvenci pozorovaných náhodných veličin. Je-li stavový model navíc parametrizován statickým parametrem, je nutné ho zahrnout v inferenci. Celý proces se tím podstatně zkomplikuje, a filtrační algoritmy typicky nekonvergují. Až na případ lineárního Gaussovského stavového modelu není věrohodnostní funkce dostupná, a inference tak není snadná. Bylo ukázáno, že částicový filtr je možné použít jako nestranný odhad věrohodnosti i v nelineárním modelu. Tento odhad ovšem předpokládá, že model pozorování je dán jako hustota pravděpodobnosti. V aplikacích je typicky k dispozici simulace pozorovaných veličin ze skrytých stavů, ale ne vyhodnocení jejich pravděpodobností. Pokusy o modelování pravděpodobnostního rozdělení těchto pozorování pak často vedou ke kolapsu částicového filtru. Inspirováni technikami Approximate Bayesian Computation (ABC) odvodíme filtr schopný odhadnout věrohodnost i v případech, kdy model pozorování není zadán jako hustota pravděpodobnosti. Vyvinutý algoritmus je nejprve otestován v simulační studii. Následně je aplikován na problém z molekulární biologie, ve kterém se pokusíme modelovat zjednodušený autoregulační systém v prokaryotách.

Klíčová slova: Stavový model, částicový filtr, Approximate Bayesian Computation, autoregulace.

Author statement for graduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

signature

Acknowledgements

I would like to thank my supervisor Kamil Dedecius for being such a positive person. It was a real pleasure to work with him and I believe I have learned a great deal. Next, I want to thank my family for keeping me alive and well-fed. Finally, I thank my friends Lukáš and Petr for keeping me sane.

Contents

Chapter 1

Introduction

Probabilistic and statistical modelling arises in a wide variety of situations. Often, the measurements one uses to perform inference have been corrupted by an unknown error. In addition, one may not have access to a correct model for the particular situation — the "true" model is either unknown, or even impossible to formulate.

In the former case, one naturally assumes a random error associated with the observations, and attempts to infer an unknown parameter from the data while accounting for this randomness. The inference may take the form of a point estimate, confidence region, hypothesis test, etc.

In the latter case, one has no choice but to work with a given, although possibly simplified model, either because of insufficient domain knowledge, or for computational reasons. Some degree of uncertainty about the parameters of such a model is then introduced. It is often beneficial to think of these parameters as random variables themselves, in accordance with the Bayesian methodology (Robert, 2007). Such formulation allows to quantify one's prior beliefs about the parameter values, and then to update them upon receiving new observations.

In this thesis, we work with state-space models (SSMs) consisting of a sequence of observed random variables $\boldsymbol{y}_t$ indexed by discrete time $t = 1, \ldots, T$, which are assumed to be generated by a latent random process $\boldsymbol{x}_t$. The distribution of $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ is assumed to be parameterized by a static parameter $\boldsymbol{\theta}$. Our goal is to perform posterior inference about this parameter, given the observed sequence $\{\boldsymbol{y}_t\}_{t=1}^{T}$. Furthermore, we assume that the likelihood function of the SSM is intractable and must be estimated. This assumption is well-grounded, as the likelihood is only available in severely restricted cases to be discussed in Chapter 3, together with a formal definition of the SSM.

The contribution is twofold. First, we show how to apply the Approximate Bayesian Computation (ABC) methodology (Rubin et al., 1984; Pritchard et al., 1999) to obtain an estimate of the likelihood even under a misspecified model for the observed variables $\boldsymbol{y}_t$. Second, we use our results to model the genetic auto-regulation process in prokaryotes. In such a problem, the observation model is typically misspecified, as all attempts to describe such a complex system are necessarily simplified. The quote by the famous statistician George E. P. Box, *"all models are wrong, but some are useful"* (Box, 1979), comes to mind here.

The rest of the thesis is organized as follows. In Chapter 2, we review some of the related work. Literature on Markov Chain Monte Carlo (MCMC) methods is discussed, as well as their use in estimating the parameters of an SSM. We list several results dealing with inference in SSMs with intractable likelihoods, as these are relevant to this thesis. Literature on ABC methods is reviewed as well, along with papers describing how these could be applied to SSMs. Finally, we discuss the application of SSMs to bioinformatics, focusing on molecular biology.

In Chapter 3, we properly define the assumed form of a state-space model. We show how one would implement a sampler to approximately infer the static parameters given a sequence of observations. We also show that in this basic form, such sampler is unusable, since it relies on the evaluation of the likelihood function, which is intractable (up to certain special cases). We then describe how this likelihood can be estimated using the particle filter (Doucet et al., 2001) without affecting the asymptotic properties of the sampler.

Chapter 4 provides a description of the ABC framework, and how it can be applied to estimate the likelihood even under a misspecified observation model. We discuss the pros and cons of such approach compared to the particle filter described in Chapter 3.

Chapter 5 provides numerical studies, where we apply the model developed in Chapter 4 to several examples and compare it with the model utilizing the particle filter. This chapter also includes the prokaryotic auto-regulation study discussed above.

Finally, Chapter 6 concludes the thesis and discusses some possible directions to be investigated in the future.

Chapter 2

Background

Markov Chain Monte Carlo methods have been widely used for approximate inference in general probabilistic models. We first address some classical works devoted to these techniques, as well as their use in state-space models. We then move on to literature describing the inference in SSMs, starting with filtering and continuing to likelihood estimation. Afterwards, works related to the Approximate Bayesian Computation methodology are surveyed. The chapter is concluded by a section describing the use of the state-space models in bioinformatics, with a focus on problems arising in molecular biology and genetics.

2.1 Markov Chain Monte Carlo methods

Monte Carlo methods (Robert and Casella, 2005) form a large class of algorithms relying on random sampling to produce numerical results, allowing to approximately solve a vast amount of otherwise intractable problems. In statistical modelling, the expectation of some transformation of a random variable is typically of interest. This is approximated by the empirical mean of a transformed random sample generated according to this distribution. Often, the probability distribution of interest is too complex to sample exactly. Assuming that the probability density function of this distribution can be evaluated (at least up to a normalizing constant), Monte Carlo methods are able to output a random sample approximately distributed according to the true distribution. *Markov Chain* Monte Carlo (MCMC) methods (Brooks et al., 2011) employ a Markov chain designed so that its limiting distribution is the target. At least asymptotically, the samples are indeed distributed according to the desired distribution.

An attractive property of MCMC, as opposed to plain Monte Carlo, is that the transition distribution of such chain need not resemble the target distribution even closely, and that the problem is relatively unaffected by the dimensionality (MacKay, 2002). The downside is the difficulty to determine convergence — for how long a chain should be simulated in order to sufficiently reach the limiting distribution. In addition, one typically requires independent samples from the target distribution, which, however, the Markov chain samples are *not*. The Markov chain samples are usually "thinned out" by keeping every nth one to ensure their approximate independence.

Perhaps the best known MCMC algorithm is the Metropolis algorithm (Metropolis et al., 1953), later improved by Hastings (1970). In this algorithm, random samples are iteratively generated from the Markov chain transition distribution. Each sample is then compared with the previous one and accepted with a certain probability ensuring that the limiting distribution is indeed the target. The go-to references for (Markov Chain) Monte Carlo methods are Robert and Casella (2005) and Brooks et al. (2011). An appealing treatment of MCMC methods with applications in physics and machine learning can

be found in MacKay (2002).

Many MCMC algorithms have been proposed to solve a wide variety of problems. For our task, the Metropolis-Hastings algorithm is sufficient, since the main problem is in the likelihood estimation and not in designing the best sampler possible.

2.2 Parameter inference in state-space models

We assume that the state-space model (SSM) takes the form informally stated in Chapter 1, fully specified in the next chapter. If all the parameters of interest change in time, that is, the inference is about the latent process $\boldsymbol{x}_t$ given the observed sample $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$, one arrives at the task of state filtering.

If the transition distribution from state $\boldsymbol{x}_t$ to state $\boldsymbol{x}_{t+1}$ is linear in the states and corrupted by uncorrelated additive noise centered at 0, this task can be solved exactly by the Kalman filter (Kalman, 1960). The resulting filter is then optimal with respect to the mean squared error. A particularly nice overview of the Kalman filter connecting it with other linear statistical models is Roweis and Ghahramani (1999).

Once the state transition becomes non-linear, as is typically the case, one can use various generalizations of the Kalman filter, such as the extended Kalman filter (EKF), which locally linearizes the transition distribution and then applies the Kalman filter to it, or the unscented Kalman filter (Julier and Uhlmann, 1997). These methods come without any optimality guarantees, though. The EKF additionally works best under a very mild non-linearity, due to its first-order approximation.

In recent years, the particle filter (Doucet et al., 2001) has become a popular alternative due to its simple implementation, appealing asymptotic properties and the fact that it allows for the transition model to be arbitrarily non-linear. Since the particle filter is used later in Chapter 3, we postpone a more detailed description there.

If, on the other hand, some of the unknown parameters are static, the task becomes more complex. Blindly applying an MCMC algorithm or any other approximation is not possible, as the likelihood function, on which such algorithms typically depend, cannot be evaluated. The paper Andrieu et al. (2010) introduced the idea of using the particle filter to obtain an estimate of the likelihood, which has been shown by Del Moral (2004) to preserve the limiting distribution of the underlying Markov chain. The resulting algorithm is called *Marginal Metropolis-Hastings*. A more recent overview can be found in the tutorial by Schön et al. (2017).

2.3 Approximate Bayesian Computation

In its original formulation, the method of Approximate Bayesian Computation (ABC) provides a way to approximate the posterior distribution $p(\theta \mid y) \propto f(y \mid \theta)p(\theta)$, assuming that the prior $p(\cdot)$ is fully known, and that the likelihood $f(\cdot \mid \theta)$ can be sampled from, but not evaluated (Rubin et al., 1984; Pritchard et al., 1999). A more recent treatment of ABC methods can be found in Marin et al. (2012) or Lintusaari et al. (2017).

Principally, ABC works by simulating a sample $\tilde{\theta}$ from the prior, substituting it to the observation-generating model, and simulating pseudo-observations $\tilde{y}$. The generating model can be either the true likelihood, or some deterministic process – a differential equation, a chemical reaction, etc. The pseudo-observations are then compared to the real observations y, and if they are "similar enough", the sample $\tilde{\theta}$ is accepted. Otherwise, it is rejected. The posterior distribution of θ is given in terms of the accepted samples $\tilde{\theta}$. The above described variant is referred to as the accept-reject ABC, for obvious reasons.

In this thesis, we apply the ABC method in place of the particle filter to allow for inference about the static parameter θ when the observation likelihood is not available. In addition, the use of ABC allows for a possibly misspecified observation model of the SSM, which is often the case, as one may not possess the necessary domain knowledge or computational power needed for the real model. Such a situation has been considered by Jasra (2015), although only through the use of the accept-reject variant given above.

Since accepting a sufficient number of samples may take a long time, an idea is to measure the distance between the true and pseudo-observations through a kernel function. This formulation would not reject any samples — instead, previously rejected samples would get assigned low weights. This has been investigated by Dedecius (2017), along with a proposed way to automatically tune the kernel width. How to exactly apply the ABC method to our problem is addressed in Chapter 4 in detail.

2.4 Applications to molecular biology

Finally, we review works describing how the framework of SSMs and their parameter inference can be applied in the context of bioinformatics, more concretely, to problems of molecular biology and genetics.

The go-to reference for stochastic modelling in biology is Wilkinson (2011). It contains a broad overview of applications of various probabilistic models to examples from molecular biology and chemistry. Included is a description of the Gillespie algorithm Gillespie (1976, 1977) used to simulate chemical reactions, which we use in Chapter 5.

A recent application of SSMs to molecular biology can be found in Golightly and Wilkinson (2011), where the authors use the particle filter to approximate the unknown likelihoods of various biological models. We implement these examples in Chapter 5 and compare them with the ABC approximation.

The paper d'Alché Buc et al. (2007) models biological networks, such as gene regulatory networks or signalling pathways, by SSMs, and estimates their parameters. The static parameters of the model are viewed as dynamic states which, however, do not change in time. The unscented Kalman filter is then applied to estimate these "dynamic" parameters. Such approach is simple, as it does not require the use of MCMC algorithms, but comes without the appealing asymptotic properties of MCMC inference.

Wang et al. (2009), Sun et al. (2008) and Zeng et al. (2011) proceed in a similar fashion when estimating the parameters of various biochemical networks. The used models are only mildly non-linear, and so the extended Kalman filter is sufficient, again without any asymptotic guarantees of identifying the true parameters.

An interesting approach to learning the structure of a gene regulatory network from a gene expression time series can be found in Noor et al. (2012). First, the particle filter is applied to learn the hidden states of the network. Once these hidden states are known, the LASSO regression is applied to learn a sparse representation of the regulatory network, since each gene is assumed to interact only with a small number of other genes.

Chapter 3

Learning the parameters of a state-space model

This chapter describes the state-space model (SSM) formulation we are working with. In Section 3.1, we formally define the SSM and state our assumptions about the individual probability distributions.

In Section 3.2, we calculate the posterior distribution of the parameters of interest, and show that straightforward inference is not possible. Further on, we derive a sampler to approximate this distribution. This sampler is unusable, as it requires the evaluation of the intractable likelihood. Nevertheless, it is illustrative to compare it with the variant derived later.

To circumvent the likelihood evaluation, we introduce the particle filter in Section 3.3. This section gives the definition and some of the properties of the filter.

Finally, in Section 3.4 we show how to use the particle filter to estimate the likelihood, and argue that it does not affect the asymptotic properties of the sampler.

Most of this chapter is based on Andrieu et al. (2010) and Schön et al. (2017).

3.1 State-Space Model definition

The state-space model, often also called the hidden Markov model (HMM) assumes a sequence of latent states $\{\boldsymbol{x}_t\}_{t=0}^{\infty} \subseteq \mathbb{R}^{d_x}$ following a Markov chain, and a sequence of observed variables $\{\boldsymbol{y}_t\}_{t=1}^{\infty} \subseteq \mathbb{R}^{d_y}$. All involved distributions are parameterized by an unknown static parameter $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^d$.

For a fixed time $T \geq 1$, we use the shorthands $\boldsymbol{x}_{0:T} = \{\boldsymbol{x}_t\}_{t=0}^{T}$ and $\boldsymbol{y}_{1:T} = \{\boldsymbol{y}_t\}_{t=1}^{T}$ throughout the thesis.

The HMM formulation means that the joint distribution of $\boldsymbol{x}_{0:T}$ and $\boldsymbol{y}_{1:T}$ factorizes, for any $T \geq 1$, into

$$p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) = p(\boldsymbol{x}_0 \mid \boldsymbol{\theta}) \prod_{t=1}^{T} f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta}) g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}), \tag{3.1}$$

where $p(\boldsymbol{x}_0 \mid \boldsymbol{\theta})$ is the prior distribution over the initial state, $f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ is the transition distribution at time t and $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$ is the observation model at time t.

The factorization (3.1) can be written more clearly as

$$\begin{aligned} \boldsymbol{x}_0 \mid \boldsymbol{\theta} &\sim p(\boldsymbol{x}_0 \mid \boldsymbol{\theta}), \\ \boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta} &\sim f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \quad t = 1, \ldots, T, \\ \boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta} &\sim g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}), \quad t = 1, \ldots, T. \end{aligned}$$

Finally, in accordance with the Bayesian approach (Robert, 2007), we introduce a prior distribution π over the unknown parameter $\boldsymbol{\theta}$ quantifying our knowledge about $\boldsymbol{\theta}$ before having observed any data. This allows us to state the full joint distribution

$$p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{1:T}, \boldsymbol{\theta}) = p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta}). \tag{3.2}$$

The corresponding graphical model is depicted in Figure 3.1.

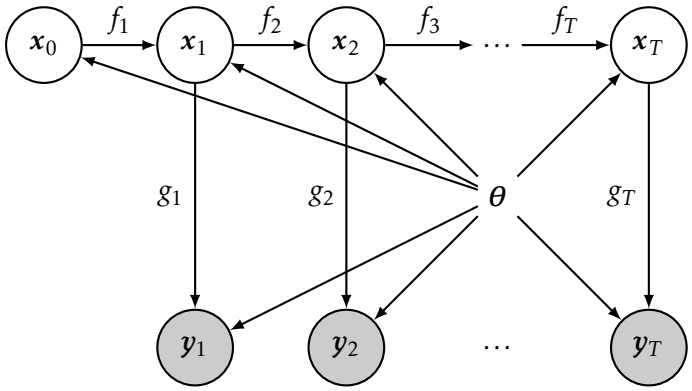


Figure 3.1: Graphical model describing the full joint distribution (3.2). The shaded nodes denote the observed variables, white nodes represent the latent variables.

3.2 Parameter inference

Given an observed sequence $\boldsymbol{y}_{1:T}$, Bayesian inference relies on the joint posterior density

$$p(\boldsymbol{\theta}, \boldsymbol{x}_{0:T} \mid \boldsymbol{y}_{1:T}) = \underbrace{p(\boldsymbol{x}_{0:T} \mid \boldsymbol{\theta}, \boldsymbol{y}_{1:T})}_{\text{State inference}} \underbrace{p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})}_{\text{Parameter inference}}. \tag{3.3}$$

Our primary goal is to infer the static parameter $\boldsymbol{\theta}$. From (3.3), it is clear that for state inference, one needs knowledge about $\boldsymbol{\theta}$, so even if the latent states $\boldsymbol{x}_{0:T}$ are of interest, knowledge about $\boldsymbol{\theta}$ is necessary.

Bayesian inference To perform Bayesian inference of $\boldsymbol{\theta}$, we express the posterior of $\boldsymbol{\theta}$ by applying the Bayes theorem:

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T}) = \frac{p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}}. \tag{3.4}$$

Evaluating the likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$ requires marginalising over $\boldsymbol{x}_{0:T}$:

$$p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) = \int p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x}_{0:T}, \tag{3.5}$$

where $p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$ is given in (3.1). Unless the SSM is linear and Gaussian, such $d_x(T+1)$-dimensional integral is intractable (Andrieu et al., 2010).

Inference under tractable likelihood assumption Let us first proceed as if the likelihood was tractable. We derive a sampler for $\boldsymbol{\theta}$ and note which component cannot be evaluated because of dependence on the intractable likelihood (3.5). Section 3.4 then describes the necessary modifications to allow circumventing the likelihood evaluation.

Often, the interest is not directly in the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$ itself, but in the expectation of some function ϕ w.r.t. this distribution, i.e., in

$$\mathbb{E}_{p(\cdot|\boldsymbol{y}_{1:T})}[\phi(\boldsymbol{\theta})] = \int \phi(\boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T}) \, \mathrm{d}\boldsymbol{\theta}. \tag{3.6}$$

We construct a Metropolis-Hastings sampler (Metropolis et al., 1953; Hastings, 1970) with target distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$. This gives us M samples approximately distributed according to this target, denoted $\boldsymbol{\theta}^{(m)}$, $m = 1, \ldots, M$. The expectation (3.6) is then approximated by the arithmetic mean

$$\frac{1}{M} \sum_{m=1}^{M} \phi(\boldsymbol{\theta}^{(m)}).$$

An appealing property of the Metropolis-Hastings algorithm is that such arithmetic mean almost surely converges to (3.6) as the number of samples grows (Robert and Casella, 2005), i.e.,

$$\frac{1}{M} \sum_{m=1}^{M} \phi(\boldsymbol{\theta}^{(m)}) \xrightarrow[M \to \infty]{a.s} \int \phi(\boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T}) \, \mathrm{d}\boldsymbol{\theta}.$$

Finally, we note that if one is interested in the distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$ itself, it can be recovered by the empirical distribution

$$\widehat{p}(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T}) = \frac{1}{M} \sum_{m=1}^{M} \delta_{\boldsymbol{\theta}^{(m)}}(\boldsymbol{\theta}),$$

where δ denotes the Dirac distribution. This estimate can be additionally smoothed using kernel methods (Wand and Jones, 1994).

Metropolis-Hastings algorithm The Metropolis-Hastings algorithm is described in Algorithm 1. Although well-known, it is included for comparison with the variant utilizing the particle filter introduced in Algorithm 5.

The algorithm constructs a Markov chain on the variable $\boldsymbol{\theta}$, whose transition distribution q is called the proposal distribution in this context. Starting from an initial state $\boldsymbol{\theta}_0$, candidate states $\boldsymbol{\theta}'$ are iteratively sampled according to $q(\cdot \mid \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the current state of the chain.

In the next step, the acceptance probability α is calculated in (3.7). This probability considers which of the two states $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ is more probable under the target distribution $p(\cdot \mid \boldsymbol{y}_{1:T}) \propto p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) \pi(\boldsymbol{\theta})$. Additionally, it allows the chain to "step back" and not move to the new state $\boldsymbol{\theta}'$ by comparing the probability of the two states under q, but in reverse direction. With probability α, the Markov chain then evolves into $\boldsymbol{\theta}'$; otherwise, it remains in the current state.

It can be shown (Robert and Casella, 2005) that the distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$ is the limiting distribution of such Markov chain. This means that with the number of transitions going to infinity, the sampled $\boldsymbol{\theta}$ are distributed according to our target distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$. To approximately reach this limiting distribution, a number of initial samples (called the burn-in period) is often discarded. In addition, one usually wants independent samples from the target distribution, which the samples from a Markov chain are *not*. In practice, only samples with a given spacing are kept to ensure their approximate independence; this is called thinning.

Similarly to the prior π, setting the proposal q is problem-dependent, and both distributions must be selected carefully. Diagnosing converge of the sampler is a notably difficult task, and one usually resorts to graphical tools to determine whether the sampled values have stabilized (Brooks et al., 2011). Some of such plots are given in Chapter 5.

Algorithm 1 Metropolis-Hastings

Input: Number of samples M, $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$.

1: Initialize $\boldsymbol{\theta}^{(0)}$.
2: **for** $m = 1$ **to** M **do**
3: Sample $\boldsymbol{\theta}' \sim q(\cdot \mid \boldsymbol{\theta}^{(m-1)})$.
4: Calculate the aceptance probability

$$\alpha = \min\left\{1, \frac{p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}')\pi(\boldsymbol{\theta}')}{p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}^{(m-1)})\pi(\boldsymbol{\theta}^{(m-1)})} \frac{q(\boldsymbol{\theta}^{(m-1)} \mid \boldsymbol{\theta}')}{q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(m-1)})}\right\}. \tag{3.7}$$

5: Sample $u \sim \mathcal{U}(0,1)$.
6: **if** $u \leq \alpha$ **then**
7: $\boldsymbol{\theta}^{(m)} \leftarrow \boldsymbol{\theta}'$ ▷ With probability α, accept the proposed sample.
8: **else**
9: $\boldsymbol{\theta}^{(m)} \leftarrow \boldsymbol{\theta}^{(m-1)}$ ▷ With probability $1 - \alpha$, reject the proposed sample.
10: **end if**
11: **end for**

Output: $\left\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}\right\}$

We see from Algorithm 1 that the acceptance probability (3.7) cannot be calculated, as it depends on the intractable likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$. In Section 3.4, we give a modified variant of the Metropolis-Hastings algorithm, where the likelihood is approximated using the particle filter. The derivation of this filter is the content of the next section.

3.3 The particle filter

The particle filter (Doucet et al., 2001) is a method for approximating the filtering distribution $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ using a finite number of samples called particles. The algorithm is also known as sequential Monte Carlo or sequential importance sampling. The latter name sheds some light on how the method works, and it is exactly through importance sampling that the particle filter is derived.

Importance sampling Here we briefly review the basic idea behind importance sampling. For a more thorough treatment, the reader is referred to MacKay (2002) or Robert and Casella (2005).

Consider a situation where the expectation of some function ϕ w.r.t. the distribution with density $p(\boldsymbol{x})$,

$$\Phi := \mathbb{E}_p[\phi(\boldsymbol{X})] = \int \phi(\boldsymbol{x}) p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{3.8}$$

is of interest. Assume that the integral is analytically intractable and that one cannot generate samples from $p(\boldsymbol{x})$ to approximate this expectation. Assume further that the density $p(\boldsymbol{x})$ can be evaluated, at least up to a multiplicative constant, i.e., that it takes the form

$$p(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{Z},$$

where Z is an unknown normalizing constant, and $p^*(\boldsymbol{x})$ can be evaluated. Such situation frequently arises in Bayesian statistics, where a posterior distribution of interest

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int p(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}}$$

is given in terms of the Bayes theorem. The normalizing constant in the denominator is often unavailable in analytic form. However, the numerator can be evaluated.

Next, we introduce a (typically simpler) distribution with density $q(\boldsymbol{x}) = \frac{q^*(\boldsymbol{x})}{Z_Q}$ s.t.

1. One can sample from q;
2. One can evaluate q^*;
3. $p(\boldsymbol{x}) > 0$ implies $q(\boldsymbol{x}) > 0$.

The expectation (3.8) can then be written as

$$\Phi = \int \phi(\boldsymbol{x}) \frac{q(\boldsymbol{x})}{q(\boldsymbol{x})} p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int \phi(\boldsymbol{x}) \underbrace{\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}}_{w^*(\boldsymbol{x})} q(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \mathbb{E}_q[\phi(\boldsymbol{X}) w^*(\boldsymbol{X})],$$

where $w^*(\boldsymbol{x})$ are called the importance weights. By defining $w(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{q^*(\boldsymbol{x})}$, Φ can be approximated by

$$\Phi \approx \widehat{\Phi} := \frac{\sum_{i=1}^N \phi(\boldsymbol{x}^{(i)}) w(\boldsymbol{x}^{(i)})}{\sum_{i=1}^N w(\boldsymbol{x}^{(i)})}, \quad \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)} \overset{iid}{\sim} q(\boldsymbol{x}).$$

We note that by using w instead of w^* and normalizing by the weights sum instead of the sample size N, we bypass the evaluation of Z and Z_Q, since they cancel out. The importance weights here account for correcting the discrepancy between the distribution $q(\boldsymbol{x})$ and the true distribution $p(\boldsymbol{x})$.

The estimator $\widehat{\Phi}$ converges to the true expectation Φ as $N \to \infty$. However, it is not necessarily unbiased (MacKay, 2002).

Sequential importance sampling (SIS) The SIS algorithm uses a set of weighted particles $\left\{\left(\boldsymbol{x}_t^{(i)}, w_t^{(i)}\right) : i = 1, \ldots, N\right\}$ to represent the filtering distribution $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$. To simplify notation, we write $w_t^{(i)}$ instead of $w_t(\boldsymbol{x}^{(i)})$ from now on. The empirical approximation to $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ is then

$$\widehat{p}(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta}) = \frac{\sum_{i=1}^N w_t^{(i)} \delta_{\boldsymbol{x}_t^{(i)}}(\boldsymbol{x}_t)}{\sum_{i=1}^N w_t^{(i)}}.$$

As the name suggests, the algorithm involves a sequential application of the importance sampling procedure with increasing time t.

Returning to the SSM (3.1), we consider the posterior distribution of a sequence of states $\boldsymbol{x}_{0:t}$ given a sequence of observations $\boldsymbol{y}_{1:t}$. By application of the Bayes theorem, we obtain the following recursive formula:

$$\begin{aligned} p(\boldsymbol{x}_{0:t} \mid \boldsymbol{y}_{1:t}) &\propto p(\boldsymbol{y}_t \mid \boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}) p(\boldsymbol{x}_{0:t} \mid \boldsymbol{y}_{1:t-1}) \\ &= g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}) p(\boldsymbol{x}_{0:t-1} \mid \boldsymbol{y}_{1:t-1}) \\ &= g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t) f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{0:t-1} \mid \boldsymbol{y}_{1:t-1}), \end{aligned}$$

where the equalities follow from the hidden Markov model independence assumptions. For clarity, we suppress the static parameter $\boldsymbol{\theta}$ from the conditioning.

For the target $p(\boldsymbol{x}_{0:t} \mid \boldsymbol{y}_{1:t})$, we introduce an importance sampling distribution $q(\boldsymbol{x}_{0:t} \mid \boldsymbol{y}_{1:t})$ and sample $\boldsymbol{x}_{0:t}^{(i)}$ from it. The importance weights are (up to normalization) given by

$$\begin{aligned} w_t^{(i)} &\propto \frac{p(\boldsymbol{x}_{0:t}(i) \mid \boldsymbol{y}_{1:t})}{q(\boldsymbol{x}_{0:t}^{(i)} \mid \boldsymbol{y}_{1:t})} \\ &\propto \frac{g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}) f_t(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{x}_{t-1}^{(i)}) p(\boldsymbol{x}_{0:t-1}^{(i)} \mid \boldsymbol{y}_{1:t-1})}{q(\boldsymbol{x}_{0:t}^{(i)} \mid \boldsymbol{y}_{1:t})}. \end{aligned} \tag{3.9}$$

By definition of the conditional probability and the hidden Markov model assumptions, we can write the importance sampling distribution as

$$q(\boldsymbol{x}_{0:t} \mid \boldsymbol{y}_{1:t}) = q(\boldsymbol{x}_t \mid \boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t}) q(\boldsymbol{x}_{0:t-1} \mid \boldsymbol{y}_{1:t-1}).$$

By substituting into (3.9), we obtain the following recursion:

$$\begin{aligned} w_t^{(i)} &\propto \frac{g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}) f_t(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{x}_{t-1}^{(i)})}{q(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{x}_{0:t-1}^{(i)}, \boldsymbol{y}_{1:t})} \frac{p(\boldsymbol{x}_{0:t-1}^{(i)} \mid \boldsymbol{y}_{1:t-1})}{q(\boldsymbol{x}_{0:t-1}^{(i)} \mid \boldsymbol{y}_{1:t-1})} \\ &\propto \frac{g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}) f_t(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{x}_{t-1}^{(i)})}{q(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{x}_{0:t-1}^{(i)}, \boldsymbol{y}_{1:t})} w_{t-1}^{(i)}. \end{aligned} \tag{3.10}$$

Evidently, updating the ith weight when transitioning from time $t-1$ to t is a relatively simple task involving only multiplication by the first fraction in (3.10).

The sequential importance sampling algorithm is summarized in Algorithm 2. This is almost the particle filter; there are still two issues to be addressed, though. First, the problem of weight degeneracy discussed in the next paragraph. Second, the choice of the importance sampling distribution $q(\boldsymbol{x})$ addressed later.

Algorithm 2 Sequential Importance Sampling

Input: Number of particles N, current parameter value $\boldsymbol{\theta}$, $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$.

1: Sample $\boldsymbol{x}_0^{(i)} \sim p(\cdot \mid \boldsymbol{\theta}), \quad i = 1, \ldots, N.$ ▷ Initialize N particles.
2: $w_0^{(i)} \leftarrow \frac{1}{N}, \quad i = 1, \ldots, N.$ ▷ Initialize uniform weights.
3: **for** $t = 1$ **to** T **do**
4: Sample $\boldsymbol{x}_t^{(i)} \sim q(\cdot \mid \boldsymbol{x}_{0:t-1}^{(i)}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta}), \quad i = 1, \ldots, N.$ ▷ Sample N new particles.
5: Set $w_t^{(i)} \propto \frac{g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}, \boldsymbol{\theta}) f_t(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{\theta})}{q(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{x}_{0:t-1}^{(i)}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})} w_{t-1}^{(i)}, \quad i = 1, \ldots, N.$ ▷ Update the weights as per (3.10).
6: **end for**

Resampling A serious problem preventing the use of the SIS algorithm is that the weights degenerate over time. At each time step, the variance of the weights reduces (Doucet et al., 2001). This means that the (normalized) weights always converge to a situation where a single weight is 1 and the others are 0.

To alleviate this, the following resampling step is introduced.

The normalized importance weights are interpreted as a probability vector of a categorical distribution. The particles are then resampled (sampled with replacement) according to this distribution. This effectively selects a population of "strong individuals" for the next time step.

Algorithm 3 is known as multinomial resampling. There are other, more sophisticated, approaches, such as stratified resampling (Douc and Cappe, 2005), which come at the cost of increased complexity.

Algorithm 3 Multinomial resampling

Input: Importance weights $w_t^{(1)},\dots,w_t^{(N)}$, particles $\boldsymbol{x}_t^{(1)},\dots,\boldsymbol{x}_t^{(N)}$.

1: $\widetilde{w}_t^{(i)} \leftarrow \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}}, \quad i = 1,\dots,N.$ ▷ Normalize weights.

2: Sample a_i s.t. $\mathbb{P}(a_i = j) = \widetilde{w}_t^{(j)}, \quad i,j = 1,\dots,N.$ ▷ Sample indices with replacement.

3: $w_t^{(a_i)} \leftarrow \frac{1}{N}, \quad i = 1,\dots,N.$ ▷ Reset weights.

Output: Resampled particles $\boldsymbol{x}_t^{(a_1)},\dots,\boldsymbol{x}_t^{(a_N)}$ and weights $w_t^{(a_1)},\dots,w_t^{(a_N)}$.

The particle filter The remaining step is the choice of the importance sampling distribution $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$. Obviously, the more similar this distribution is to the target $p(\boldsymbol{x}_{0:t} \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$, the closer approximation we obtain.

The particle filter arises when the transition distribution $f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ is chosen as the importance distribution, that is, when

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta}) = f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta}).$$

The importance weights (3.10) then simplify into

$$w_t^{(i)} \propto g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}) w_{t-1}^{(i)}. \tag{3.11}$$

The particle filter is summarized in Algorithm 4. The algorithm is called *bootstrap* particle filter, due to resemblance of the resampling step to the non-parametric bootstrap (Efron, 1979). By being defined in terms of importance sampling, the algorithm inherits the appealing asymptotic properties.

Algorithm 4 Bootstrap particle filter

Input: Number of particles N, current parameter value $\boldsymbol{\theta}$, $\{\boldsymbol{y}_1,\dots,\boldsymbol{y}_T\}$.

1: Sample $\boldsymbol{x}_0^{(i)} \sim p(\cdot \mid \boldsymbol{\theta}), \quad i = 1,\dots,N.$ ▷ Initialize N particles.

2: $w_0^{(i)} \leftarrow \frac{1}{N}, \quad i = 1,\dots,N.$ ▷ Initialize uniform weights.

3: **for** $t = 1$ **to** T **do**

4: Sample $\boldsymbol{x}_t^{(i)} \sim f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{\theta}), \quad i = 1,\dots,N.$ ▷ Sample N new particles.

5: Set $w_t^{(i)} \propto g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}, \boldsymbol{\theta}) w_{t-1}^{(i)}, \quad i = 1,\dots,N.$ ▷ Update the weights as per (3.11).

6: Resample $\boldsymbol{x}_t^{(i)}$ and reset $w_t^{(i)}$ using Algorithm 3, $\quad i = 1,\dots,N.$

7: **end for**

3.4 Using the particle filter to estimate the likelihood

As mentioned in Section 3.3, the particle filter is typically used to approximate the filtering distribution $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$. This will be utilized to provide a tractable approximation to the likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$ such that the limiting distribution of the Metropolis-Hastings Markov chain remains unaffected. This section describes how it is done and gives the resulting variant of the sampler

Likelihood estimate in general Suppose that we are in possession of an estimator $\widehat{z}$ of the likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$. As such, it necessarily depends on $\boldsymbol{y}_{1:T}$ and $\boldsymbol{\theta}$. Since we aim to use the particle filter to calculate $\widehat{z}$, the estimator also depends on the importance weights calculated using random samples $\boldsymbol{x}_t^{(i)}$. This makes the estimator a random variable with

some distribution denoted $\psi(z \mid \boldsymbol{\theta}, \boldsymbol{y}_{1:T})$. It is not necessary to have this distribution available, as it is later shown to cancel out in the Metropolis-Hastings acceptance ratio.

We now return to our model (3.4) and introduce $\widehat{z}$ as an auxiliary variable, along with our variable of interest $\boldsymbol{\theta}$. This changes the target distribution from $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$ to

$$\psi(\boldsymbol{\theta}, z \mid \boldsymbol{y}_{1:T}) = p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})\psi(z \mid \boldsymbol{\theta}, \boldsymbol{y}_{1:T}) = \frac{p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})}{p(\boldsymbol{y}_{1:T})}\psi(z \mid \boldsymbol{\theta}, \boldsymbol{y}_{1:T}). \tag{3.12}$$

In theory, we could now construct a Metropolis-Hastings algorithm with $\psi(\boldsymbol{\theta}, z \mid \boldsymbol{y}_{1:T})$ as the target, instead of $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$ as was the case in Algorithm 1. However, this would not solve our problem, since calculating the acceptance ratio still requires the calculation of the likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$, as (3.12) makes clear.

Instead, we define a new target distribution over $(\boldsymbol{\theta}, \widehat{z})$ by replacing the likelihood in (3.12) by its estimate $\widehat{z}$:

$$\pi(\boldsymbol{\theta}, z \mid \boldsymbol{y}_{1:T}) \coloneqq \frac{z\pi(\boldsymbol{\theta})}{p(\boldsymbol{y}_{1:T})}\psi(z \mid \boldsymbol{\theta}, \boldsymbol{y}_{1:T}). \tag{3.13}$$

There are of course some conditions imposed on $\pi(\boldsymbol{\theta}, z \mid \boldsymbol{y}_{1:T})$ for it to be useful:

1. $\pi(\boldsymbol{\theta}, z \mid \boldsymbol{y}_{1:T})$ must be non-negative for all $(\boldsymbol{\theta}, z)$;
2. $\pi(\boldsymbol{\theta}, z \mid \boldsymbol{y}_{1:T})$ must integrate to 1;
3. the marginal distribution of $\pi(\boldsymbol{\theta}, z \mid \boldsymbol{y}_{1:T})$ for $\boldsymbol{\theta}$ must be the original target $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$.

The first two conditions simply state that π is a valid probability distribution. The third condition ensures that by constructing a Metropolis-Hastings algorithm with π as the target, the original target distribution is preserved once the auxiliary variables are marginalised out. All three conditions are satisfied if $\widehat{z}$ is a non-negative unbiased estimator of the likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$. This is shown as follows.

1. Non-negativity of π follows from the assumed non-negativity of the estimator $\widehat{z}$ and validity of the distributions in (3.13).

2, 3. Assume that $\widehat{z}$ is an unbiased estimate of $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$, i.e., that $\mathbb{E}_\psi[\widehat{z}] = p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$. Consider now the marginal of π for $\boldsymbol{\theta}$:

$$\begin{aligned}\int \pi(\boldsymbol{\theta}, z \mid \boldsymbol{y}_{1:T})\,\mathrm{d}z &= \frac{\pi(\boldsymbol{\theta})}{p(\boldsymbol{y}_{1:T})}\int z\psi(z \mid \boldsymbol{\theta}, \boldsymbol{y}_{1:T})\,\mathrm{d}z \\ &= \frac{\pi(\boldsymbol{\theta})}{p(\boldsymbol{y}_{1:T})}\mathbb{E}_\psi[\widehat{z}] \\ &= \frac{\pi(\boldsymbol{\theta})}{p(\boldsymbol{y}_{1:T})}p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) \\ &= p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T}),\end{aligned} \tag{3.14}$$

the original target distribution. This satisfies condition 3. For condition 2, we simply integrate (3.14) w.r.t. $\boldsymbol{\theta}$, which results in unity due to $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$ being a valid probability distribution.

Acceptance ratio computation Given the new target distribution π, we can now construct a Metropolis-Hastings algorithm on the joint space of $(\boldsymbol{\theta}, z)$.

This means that the proposed samples are now given as $(\boldsymbol{\theta}', z') \sim \psi(\cdot, \cdot \mid \boldsymbol{y}_{1:T})$. In practice, this is done by first sampling $\boldsymbol{\theta}' \sim q(\cdot \mid \boldsymbol{\theta}^{(m-1)})$, and then $\widehat{z'} \sim \psi(\cdot \mid \boldsymbol{\theta}', \boldsymbol{y}_{1:T})$. The acceptance ratio can now be computed as

$$\begin{aligned}
\alpha &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}', z' \mid \boldsymbol{y}_{1:T})}{\pi(\boldsymbol{\theta}^{(m-1)}, z^{(m-1)} \mid \boldsymbol{y}_{1:T})} \frac{q(\boldsymbol{\theta}^{(m-1)} \mid \boldsymbol{\theta}')\psi(z^{(m-1)} \mid \boldsymbol{\theta}^{(m-1)}, \boldsymbol{y}_{1:T})}{q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(m-1)})\psi(z' \mid \boldsymbol{\theta}', \boldsymbol{y}_{1:T})}\right\} \\
&= \min\left\{1, \frac{z'\pi(\boldsymbol{\theta}')\psi(z' \mid \boldsymbol{\theta}', \boldsymbol{y}_{1:T})}{z^{(m-1)}\pi(\boldsymbol{\theta}^{(m-1)})\psi(z^{(m-1)} \mid \boldsymbol{\theta}^{(m-1)}, \boldsymbol{y}_{1:T})} \frac{q(\boldsymbol{\theta}^{(m-1)} \mid \boldsymbol{\theta}')\psi(z^{(m-1)} \mid \boldsymbol{\theta}^{(m-1)}, \boldsymbol{y}_{1:T})}{q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(m-1)})\psi(z' \mid \boldsymbol{\theta}', \boldsymbol{y}_{1:T})}\right\} \\
&= \min\left\{1, \frac{z'\pi(\boldsymbol{\theta}')}{z^{(m-1)}\pi(\boldsymbol{\theta}^{(m-1)})} \frac{q(\boldsymbol{\theta}^{(m-1)} \mid \boldsymbol{\theta}')}{q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(m-1)})}\right\}.
\end{aligned}$$

Since (3.14) shows that the marginal of π for $\boldsymbol{\theta}$ is the original target $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$, all we need to do is to discard the sampled $\widehat{z}^{(m)}$ and keep only $\boldsymbol{\theta}^{(m)}$ when running Metropolis-Hastings on the joint space of $(\boldsymbol{\theta}, z)$.

Calculating the estimate using the particle filter Finally, we describe how exactly is the particle filter used as an estimator of $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$.

First, we decompose the likelihood into a product of simpler distributions, which are then marginalised over the corresponding hidden state:

$$\begin{aligned}
p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) &= \prod_{t=1}^{T} p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \\
&= \prod_{t=1}^{T} \int p(\boldsymbol{y}_t, \boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x}_t \\
&= \prod_{t=1}^{T} \int p(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x}_t.
\end{aligned} \tag{3.15}$$

Using the particles $\left\{\boldsymbol{x}_t^{(i)}\right\}_{i=1}^{N}$, we plug in the empirical approximation to $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$, $\widehat{p}(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{x}_t^{(i)}}(\boldsymbol{x}_t)$, into (3.15), obtaining

$$\begin{aligned}
p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) &\approx \prod_{t=1}^{T} \int p(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) \left[\frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{x}_t^{(i)}}(\boldsymbol{x}_t)\right] \mathrm{d}\boldsymbol{x}_t \\
&= \prod_{t=1}^{T} \frac{1}{N}\sum_{i=1}^{N} \int p(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) \delta_{\boldsymbol{x}_t^{(i)}}(\boldsymbol{x}_t) \, \mathrm{d}\boldsymbol{x}_t \\
&= \prod_{t=1}^{T} \frac{1}{N}\sum_{i=1}^{N} p(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}, \boldsymbol{\theta})
\end{aligned}$$

due to linearity of the integral and properties of the Dirac distribution.

In $p(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}, \boldsymbol{\theta})$, we recognize the particle filter weights $w_t^{(i)}$ defined in (3.11). This allows us to finally define the likelihood estimate as

$$\widehat{z} := \prod_{t=1}^{T} \frac{1}{N}\sum_{i=1}^{N} w_t^{(i)}. \tag{3.16}$$

This estimator is obviously non-negative due to construction of the weights. The proof that it is also unbiased (and therefore also integrates to unity) is more involved and the reader is referred to Del Moral (2004) for the original proof.

Finally, we describe the resulting variant of the Metropolis-Hastings algorithm employing the likelihood estimate (3.16). This algorithm, called marginal Metropolis-Hastings, was introduced by Andrieu et al. (2010). Compared to Algorithm 1, all components of this algorithm can be evaluated. Due to construction of the estimator $\widehat{z}$, the marginal of the limiting distribution of Algorithm 5 is the original target $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$.

Algorithm 5 Marginal Metropolis-Hastings

Input: Number of samples M, $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$.

1: Initialize $\boldsymbol{\theta}^{(0)}$.
2: Run Algorithm 4 with $\boldsymbol{\theta}^{(0)}$ to obtain the weights $w_{0,t}^{(i)}, \quad t = 1, \ldots, T,\ i = 1, \ldots, N$.
3: Calculate $\widehat{z}^{(0)}$ according to (3.16) using $w_{0,t}^{(i)}$.
4: **for** $m = 1$ **to** M **do**
5: Sample $\boldsymbol{\theta}' \sim q(\cdot \mid \boldsymbol{\theta}^{(m-1)})$.
6: Run Algorithm 4 with $\boldsymbol{\theta}'$ to obtain the weights $w_{m,t}^{(i)}, \quad t = 1, \ldots, T,\ i = 1, \ldots, N$.
7: Calculate $\widehat{z}'$ according to (3.16) using $w_{m,t}^{(i)}$.
8: Calculate the aceptance probability

$$\alpha = \min\left\{1, \frac{\widehat{z}'\pi(\boldsymbol{\theta}')}{\widehat{z}^{(m-1)}\pi(\boldsymbol{\theta}^{(m-1)})} \frac{q(\boldsymbol{\theta}^{(m-1)} \mid \boldsymbol{\theta}')}{q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(m-1)})}\right\}.$$

9: Sample $u \sim \mathcal{U}(0, 1)$.
10: **if** $u \leq \alpha$ **then**
11: $\left(\boldsymbol{\theta}^{(m)}, \widehat{z}^{(m)}\right) \leftarrow (\boldsymbol{\theta}', \widehat{z}')$ ▷ With probability α, accept the proposed sample.
12: **else**
13: $\left(\boldsymbol{\theta}^{(m)}, \widehat{z}^{(m)}\right) \leftarrow \left(\boldsymbol{\theta}^{(m-1)}, \widehat{z}^{(m-1)}\right)$ ▷ With probability $1 - \alpha$, reject the proposed sample.
14: **end if**
15: **end for**

Output: $\left\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}\right\}$

Chapter 4

Approximate Bayesian Computation

We are now in possession of a sampler able to approximate the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$ even when the model likelihood is not tractable. As such, the sampler can be used in general non-linear SSMs, as long as the observation model $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$ is a well-defined probability density. This requirement can be relaxed by introducing the method of Approximate Bayesian Computation (ABC). The algorithm derived in this chapter utilizes the ABC framework to approximate the SSM likelihood even when the observation model is misspecified or given only as a deterministic mapping $\boldsymbol{x}_t \mapsto \boldsymbol{y}_t$. This allows to infer $\boldsymbol{\theta}$ even when $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$ is not given in terms of a probability density function.

We first motivate the use of ABC methods in our problem in Section 4.1. Then, in Section 4.2, we describe the method in general and discuss some limitations. Section 4.3 introduces ABC to our state-space model framework and addresses some potential issues through kernel functions. Finally, in Section 4.4, we summarize how exactly is the ABC method used in our model, and provide an alternative variant of the Metropolis-Hastings algorithm which relies on ABC instead of the particle filter to estimate the likelihood.

4.1 Motivation

In the previous chapter, we derived a way to bypass the likelihood function evaluation when calculating the Metropolis-Hastings acceptance ratio. The method relies on the particle filter to calculate a set of weights $w_t^{(i)} \propto g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}, \boldsymbol{\theta})$, where $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(i)}, \boldsymbol{\theta})$ is the observation model defined in (3.1). These weights are used to estimate the likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$ as given in (3.16). However, calculating the weights in such way requires full knowledge of this observation model.

In practice, one may not have access to a correct observation model in the form of a probability density $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$. Instead, only a model of the process which generates an observation $\boldsymbol{y}_t$ from the latent state $\boldsymbol{x}_t$ may be available. This generative process may take the form of a differential equation, chemical reaction, simulation, etc. One is then in possession of a mean to generate an observation, but not to evaluate how probable it is. By attempting to fit an arbitrary probability distribution to this generative model, an error is necessarily introduced. The particle filter weights might then not reflect reality, and would lead to incorrect results when using such misspecified model for $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$.

As an alternative way to approximate the likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$, we can utilize our knowledge of the generative process $\boldsymbol{x}_t \mapsto \boldsymbol{y}_t$ to simulate a number of pseudo-observations $\boldsymbol{u}_t$. A surrogate for the observation density $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$ is then calculated by evaluating the closeness of these pseudo-observations to the true measurement $\boldsymbol{y}_t$. Intuitively, if a large number of the simulated observations fall close to $\boldsymbol{y}_t$, we would expect the true probability density to be high in that region. By bypassing the evaluation of $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$,

inference can proceed even without knowing the observation model density. This is exactly the idea behind the approximate Bayesian computation methodology; it is discussed in Section 4.2.

Unfortunately, such approximation comes at a price. In Chapter 3, it has been shown that using the particle filter does not introduce any approximation error, since the likelihood estimate is unbiased and leaves the limiting distribution of the Metropolis-Hastings Markov chain intact. This is not the case when applying ABC methods, see Section 4.3.

4.2 ABC in general

Before describing how to apply ABC to state-space models, we first summarize the underlying ideas. The ABC method is introduced in the context of general Bayesian inference under a misspecified likelihood function. Later on, we build on these foundations when applying ABC to our SSM framework.

One thing to note is that ABC has traditionally been applied to estimate the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ for some parameter $\boldsymbol{\theta}$ and observation $\boldsymbol{y}$. The method is first considered with this application in mind, and in Section 4.3, we describe how use it to estimate the SSM likelihood instead.

Approximate Bayesian Computation The methodology of ABC dates back to Rubin et al. (1984), where a procedure using simulated psedudo-observations to approximate the posterior distribution was first described. Lately, ABC methods have gained popularity in modelling biological processes (Pritchard et al., 1999). A more recent review can be found in Marin et al. (2012); Lintusaari et al. (2017).

In its classical formulation, ABC provides a way to approximate an intractable posterior $p(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$ by introducing an auxiliary variable $\boldsymbol{u}$. The posterior approximation is then constructed by integrating over this variable and considering only values sufficiently close to the true measurement (Jasra et al., 2012). It takes the form of

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) \approx p^{\epsilon}(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{\int \mathbb{I}_{\mathcal{A}_{\epsilon,\boldsymbol{y}}}(\boldsymbol{u}) p(\boldsymbol{u} \mid \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{u}}{\int_{\mathcal{A}_{\epsilon,\boldsymbol{y}}} \mathrm{d}\boldsymbol{u}}, \tag{4.1}$$

where $\mathbb{I}_{\mathcal{A}_{\epsilon,\boldsymbol{y}}}$ is the indicator function of a set $\mathcal{A}_{\epsilon,\boldsymbol{y}} = \left\{\boldsymbol{u} \in \mathbb{R}^{d_y} : \rho(\boldsymbol{u}, \boldsymbol{y}) \leq \epsilon\right\}$ and $\rho : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$ is a metric, typically the Euclidean distance.

The motivation behind (4.1) is that such integral can be approximated by randomly sampling from the likelihood $p(\cdot \mid \boldsymbol{\theta})$ without needing to evaluate it. This way, the likelihood can exist only conceptually, and we are able to simulate samples $\boldsymbol{u}$ from a model reflecting some real-world process, without considering the underlying probability density.

The hyper-parameter $\epsilon \geq 0$ controls how far the auxiliary variable $\boldsymbol{u}$ can be from the true measurement $\boldsymbol{y}$ to be considered similar. Clearly, if we set $\epsilon = 0$, the integral becomes $p(\boldsymbol{y} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$, and we recover the true posterior. In general, the smaller ϵ, the better approximation is obtained, though at the cost of increased computational complexity.

To avoid the curse of dimensionality, a summary statistic $\boldsymbol{s} : \mathbb{R}^{d_y} \to \mathbb{R}^p$, $1 \leq p < d_y$ is often introduced. Instead of comparing $\rho(\boldsymbol{u}, \boldsymbol{y}) \leq \epsilon$, one then compares $\rho(\boldsymbol{s}(\boldsymbol{u}), \boldsymbol{s}(\boldsymbol{y})) \leq \epsilon$ (assuming that the metric has been redefined to $\rho : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$).

It can be shown that if $\boldsymbol{s}$ is a sufficient statistic for the parameter $\boldsymbol{\theta}$, the probability density $p^{\epsilon}(\boldsymbol{\theta} \mid \boldsymbol{y})$ converges to $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ as $\epsilon \to 0$ (Jasra, 2015). However, it is typically impossible to find such statistic outside of the exponential family of distributions. Otherwise, using a statistic that is not sufficient introduces an additional approximation error.

Basic version of the ABC simulation We now give a basic variant of a sampling-based approximation to $p^{\epsilon}(\boldsymbol{\theta} \mid \boldsymbol{y})$. In the spirit of (4.1), Algorithm 6 performs rejection sampling by comparing whether a sampled $\boldsymbol{u}$ is in $\mathcal{A}_{\epsilon,\boldsymbol{y}}$ or not. After describing the algorithm, we discuss some limitations of this basic approach.

Algorithm 6 ABC Rejection Algorithm

Input: Number of samples M, observation $\boldsymbol{y}$, metric ρ, maximum distance ϵ.

$i \leftarrow 1$
while $i \leq M$ **do**
  Sample $\boldsymbol{\theta}' \sim \pi(\cdot)$. ▷ Sample from the prior.
  Simulate $\boldsymbol{u}$ from $p(\cdot \mid \boldsymbol{\theta}')$. ▷ Simulate a pseudo-observation.
  if $\rho(\boldsymbol{u}, \boldsymbol{y}) \leq \epsilon$ **then**
    $\boldsymbol{\theta}^{(i)} \leftarrow \boldsymbol{\theta}'$ ▷ Accept the proposed sample.
    $i \leftarrow i + 1$
  end if
end while

Output: Accepted samples $\left\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}\right\}$.

ABC rejection iteratively samples parameters $\boldsymbol{\theta}'$ from the prior, plugs them into the likelihood $p(\cdot \mid \boldsymbol{\theta}')$, and simulates pseudo-observations $\boldsymbol{u}$. These are then compared to the true measurement $\boldsymbol{y}$ using the metric ρ. If the proposed parameter $\boldsymbol{\theta}'$ gave rise to a pseudo-observation similar enough to the true $\boldsymbol{y}$ (i.e., $\boldsymbol{u} \in \mathcal{A}_{\epsilon,\boldsymbol{y}}$), the parameter is kept under the assumption that the true data are likely under $\boldsymbol{\theta}'$. The ABC approximation is then given in terms of the accepted samples $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}$ as the empirical distribution

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) \approx \frac{1}{M} \sum_{i=1}^{M} \delta_{\boldsymbol{\theta}^{(i)}}(\boldsymbol{\theta}).$$

Setting a low value of ϵ increases the approximation accuracy, at the cost of increased rejection rate. On the other hand, setting ϵ too large causes the algorithm to accept more often, but leads to simulating pseudo-measurements dissimilar to $\boldsymbol{y}$ and, in turn, incorrect $\boldsymbol{\theta}^{(i)}$. Setting a suitable value of ϵ is therefore the main difficulty when using ABC. Several approaches are discussed by Jasra et al. (2012); Jasra (2015). One particular way (Dedecius, 2017) is used in Section 4.3 in the context of SSMs.

There are many improvement to the basic ABC of Algorithm 6, discussed for instance by Marin et al. (2012). In particular, more sophisticated sampling approaches relying again on MCMC are described. This is not an issue relevant to the SSM framework, as the samples are generated in a different fashion, given in the next section.

4.3 ABC in SSMs

Next, we describe how exactly is the ABC methodology applied in the context of SSMs.

Section 4.2 states that the typical use case of ABC arises in cases where we have knowledge about the data-generating process, but are unable to evaluate the probability of such data. In the context of SSMs, this translates into knowing how the observed values $\boldsymbol{y}_t$ have been generated from the latent states $\boldsymbol{x}_t$, but being unable to evaluate the density $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$. This prevents us from calculating the importance weights w_t through the particle filter, which relies on the availability of $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$.

Toni et al. (2009) and Jasra et al. (2012) describe how a filter could be constructed using the ABC approximation. Additionally, Jasra (2015) applies this filter in the context of SSMs. We first discuss the construction of this filter. Afterwards, we address a particular limitation of this approach through kernel functions.

Filter construction through ABC In SSMs, the dimensionality of the observation space is typically low; the observations are often scalar quantities. It is then not necessary to consider any summary statistics.

Jasra et al. (2012) consider a modification of the particle filter (Algorithm 4) which simulates pseudo-observations according to the observation model, and calculates the importance weights based on their closeness to the true measurements. The pseudocode is given in Algorithm 7.

Algorithm 7 ABC-based filter

Input: Number of particles N, current parameter value $\boldsymbol{\theta}$, maximum distance ϵ, $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$.

1: Sample $\boldsymbol{x}_0^{(i)} \sim p(\boldsymbol{x}_0 \mid \boldsymbol{\theta}), \quad i = 1, \ldots, N.$ ▷ Initialize N particles.
2: $w_0^{(i)} \leftarrow \frac{1}{N}, \quad i = 1, \ldots, N.$ ▷ Initialize uniform weights.
3: **for** $t = 1$ **to** T **do**
4: Sample $\boldsymbol{x}_t^{(i)} \sim f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{\theta}), \quad i = 1, \ldots, N.$ ▷ Sample N new particles.
5: Simulate $\boldsymbol{u}_t^{(i)}$ from $g_t(\boldsymbol{u}_t \mid \boldsymbol{x}_t^{(i)}, \boldsymbol{\theta}), \quad i = 1, \ldots, N.$ ▷ Simulate N pseudo-observations.
6: Set $w_t^{(i)} \propto \mathbb{I}_{\mathcal{A}_{\epsilon, \boldsymbol{y}_t}}(\boldsymbol{u}_t^{(i)}) w_{t-1}^{(i)}, \quad i = 1, \ldots, N.$
7: Resample $\boldsymbol{x}_t^{(i)}$ and reset $w_t^{(i)}$ using Algorithm 3, $\quad i = 1, \ldots, N.$
8: **end for**

The algorithm proceeds similarly to Algorithm 4 except for the way the weights are computed. Instead of evaluating the unavailable density $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$ at the true observation $\boldsymbol{y}_t$, a pseudo-observation $\boldsymbol{u}_t$ is simulated. The weight is then set to a non-zero value if $\boldsymbol{u}_t \in \mathcal{A}_{\epsilon, \boldsymbol{y}_t}$, and 0 otherwise. It may seem that the weights for the same particle i necessarily collapse to 0 after a number of time steps due to the recursive multiplication in step 6. However, step 7 resets the weights uniformly after resampling, so such collapse does not occur.

Analogously to Section 3.4, the weights are then used to approximate the likelihood $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$. According to Jasra (2015), the estimate is given by

$$\widehat{z} = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} \frac{w_t^{(i)}}{\int_{\mathcal{A}_{\epsilon, \boldsymbol{y}_t}} \mathrm{d}\boldsymbol{u}}. \tag{4.2}$$

The integral in the denominator essentially normalizes the weights $w_t^{(i)}$ to be equal to the probability density of $\mathcal{U}(\boldsymbol{y}_t; \epsilon)$, the uniform distribution in a sphere centered at $\boldsymbol{y}_t$ with radius ϵ given in terms of the metric ρ.

Bias The use of this ABC filter introduces bias to the parameter inference. Recall that in Section 3.4, we required $\widehat{z}$ to be an unbiased estimator of $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$. This was the case when the weights were calculated according to $w_t^{(i)} \propto g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t) w_{t-1}^{(i)}$. This unbiasedness is not preserved here, since $\widehat{z}$ estimates

$$\begin{aligned} p^{\epsilon}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) &= \int p^{\epsilon}(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x}_{0:T} \\ &= \int p(\boldsymbol{x}_0 \mid \boldsymbol{\theta}) \prod_{t=1}^{T} f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta}) g_t^{\epsilon}(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x}_{0:T} \end{aligned}$$

(compare the inside of the integral with (3.1)), as noted by Jasra (2015). Here the approximate observation density is given by the ABC form

$$g_t^{\epsilon}(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) = \frac{\int \mathbb{I}_{\mathcal{A}_{\epsilon, \boldsymbol{y}_t}}(\boldsymbol{u}) g_t(\boldsymbol{u} \mid \boldsymbol{x}_t, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{u}}{\int_{\mathcal{A}_{\epsilon, \boldsymbol{y}_t}} \mathrm{d}\boldsymbol{u}},$$

similarly to (4.1). This essentially means that by plugging (4.2) into the Metropolis-Hastings acceptance ratio, the limiting distribution of the underlying Markov chain becomes $p^\epsilon(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T}) \propto p^\epsilon(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$, instead of the correct $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$. In general, this bias cannot be dealt with, and is a price to pay for using the incorrect observation model.

An interesting way to address this deficiency has been proposed by Fearnhead and Prangle and by Wilkinson (2013). The authors note that one uses data $\boldsymbol{y}_{1:T}$ assumed to have been generated according to (3.1), $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$, but fitting the ABC approximation $p^\epsilon(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$. In an attempt to bring the data closer the model being really fitted, the authors use a sequence of perturbed observations $\boldsymbol{z}_t = \boldsymbol{y}_t + \boldsymbol{v}$, $\boldsymbol{v} \sim \mathcal{U}(\mathbf{0}; \epsilon)$ which denotes the uniform distribution in a sphere given by ρ, with radius ϵ and centered at the origin. It is proved that if $\boldsymbol{\theta}$ is estimated according to maximum likelihood, the estimate is consistent when using the perturbed sequence $\boldsymbol{z}_{1:T}$. This approach is called the Noisy ABC.

Use of kernel functions A limitation of Algorithm 6 and Algorithm 7 lies in the use of the indicator function $\mathbb{I}_{\mathcal{A}_{\epsilon,\boldsymbol{y}}}$. There are two problems:

1. A practical one; it may happen that no samples are accepted and the output is null in the case of Algorithm 6, or too many weights become zero in the case of Algorithm 7 and the filter collapses.

2. A more fundamental one, the simulated pseudo-observations $\boldsymbol{u}_t$ are all assigned equal weights, regardless of how far they lie from the true measurement $\boldsymbol{y}_t$. Intuitively, a pseudo-observation closer to the true $\boldsymbol{y}_t$ should be assigned a higher weight than one which is further away.

Both issues can be mitigated by considering a general kernel function in place of the indicator $\mathbb{I}_{\mathcal{A}_{\epsilon,\boldsymbol{y}}}$. Let a kernel of width ϵ centered at $\boldsymbol{y}$ and evaluated at $\boldsymbol{u}$ be denoted by $\kappa(\boldsymbol{u}; \boldsymbol{y}, \epsilon)$. In machine learning, kernel functions are often taken *proportional* to some symmetric probability density function (Hastie et al., 2001). However, as we aim to replace the indicator function $\mathbb{I}_{\mathcal{A}_{\epsilon,\boldsymbol{y}}}$ by such kernel, we require the kernel to be properly normalized (integrate to unity) to mirror the normalization in (4.2).

With the kernel function, we can write $w_t^{(i)} \propto \kappa(\boldsymbol{u}_t; \boldsymbol{y}_t, \epsilon) w_{t-1}^{(i)}$. The likelihood estimate (4.2) becomes

$$\widehat{z} = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} w_t^{(i)} \tag{4.3}$$

due to the kernel being normalized. This way, the weights are no longer uniform but reflect the distance of $\boldsymbol{u}_t$ from $\boldsymbol{y}_t$. There is also no risk of the filter collapsing due to majority of the weights becoming zero.

Introducing the kernel function to the weights computation is in principle similar to using importance sampling rather than simple rejection sampling (MacKay, 2002). Instead of accepting/rejecting the generated samples based on whether they match some criterion, they are all accepted and weighted according to *how well* they match it.

With the kernel functions in mind, we describe a procedure for automatic tuning of the kernel width ϵ, which has been ignored so far. The adopted method has been derived in a one-dimensional setting, i.e., $\boldsymbol{u}, \boldsymbol{y} \in \mathbb{R}$. We correspondingly denote $\boldsymbol{y}$ by y and $\boldsymbol{u}$ by u. When considering multivariate observations, the kernels are applied coordinate-wise (assuming independence between the components of $\boldsymbol{y}$ or $\boldsymbol{u}$, respectively).

Before describing the kernel tuning procedure, we give several examples of commonly used kernel functions and comment on their usage. The kernels are shown in Figure 4.1.

1. **Gaussian kernel** The Gaussian kernel takes the form

$$\kappa(u;y,\epsilon) = \frac{1}{\sqrt{2\pi\epsilon^2}} \exp\left\{-\frac{(u-y)^2}{2\epsilon^2}\right\}.$$

   It is one of the most-commonly used kernel functions.

2. **Cauchy kernel** The Cauchy kernel takes the form

$$\kappa(u;y,\epsilon) = \frac{1}{\pi\epsilon\left[1+\left(\frac{u-y}{\epsilon}\right)^2\right]}.$$

   As opposed to the Gaussian distribution, the Cauchy distribution has heavier tails, which make it suitable for situations with potentially distant observations. This kernel typically assigns non-trivial probability even to distant pseudo-observations, preventing the filter from collapsing under outliers.

3. **Uniform kernel** The uniform kernel takes the form

$$\kappa(u;y,\epsilon) = \begin{cases} \frac{1}{2\epsilon}, & y-\epsilon < u < y+\epsilon; \\ 0, & \text{otherwise.} \end{cases}$$

   Using this kernel, we recover the standard ABC which accepts u if $u \in \mathcal{A}_{\epsilon,y} = \left\{u : \left|u-y\right| < \epsilon\right\}$. If we are in a situation with multivariate observations $\boldsymbol{y}_t$ and apply the uniform kernel coordinate-wise, the set of accepted samples coincides with the standard ABC using $\rho(\boldsymbol{u},\boldsymbol{y}) = \max_{i=1}^{d_y} \left|u_i - y_i\right|$, the L_∞ distance.

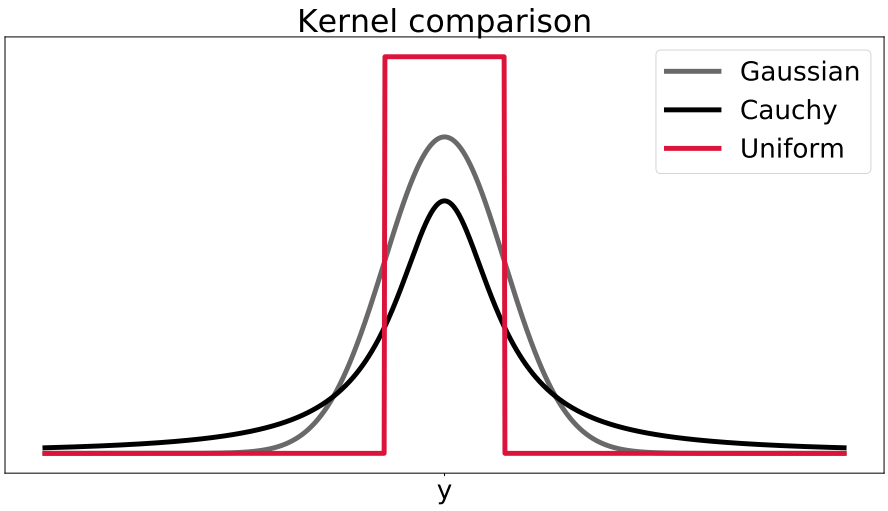


Figure 4.1: The Gaussian, Cauchy and uniform kernel functions centered at $y = 0$ with width $\epsilon = 3$.

The Noisy ABC procedure described above is then naturally generalized by perturbing the observations by samples from a given kernel, instead of the uniform distribution in a ball.

Kernel width tuning In this paragraph, we address the issue of tuning the kernel width ϵ. Since the previous section has shown that the indicator function $\mathbb{I}_{\mathcal{A}_{\epsilon,y}}$ is recovered by a particular kernel, the method also applies to the standard ABC formulation.

A careful setting of ϵ is necessary. When the kernel width is too low, most of the N generated pseudo-observations are assigned with low probabilities, and the importance weights are close to zero. On the other hand, when the width is too high, even outlying pseudo-observations are assigned a non-trivial probability and shift the filter to incorrect values. In addition, the kernel becomes flat and close to the uniform variant. A manual setting of ϵ is thus a non-obvious task without any guidelines in the data. Additionally, the width should somehow reflect the filter evolution over time, since all observations $\boldsymbol{y}_t$ are different and may require different kernel widths.

In this thesis, we adopt a procedure described by Dedecius (2017), which is briefly reviewed below. More details can be found in the original paper.

The method is based on the idea that the true observation model $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$ should cover a given number of generated pseudo-observations $\boldsymbol{u}_t^{(i)}, i = 1, \dots, N$ by a $100p\%$ high probability region (p-HPR), where $p \in (0, 1)$ is a given constant. If this is true, the pseudo-observations can be expected to describe the distribution $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$ sufficiently-well. As this distribution is not known, it is approximated by the kernel κ evaluated at $\boldsymbol{u}_t^{(i)}, i = 1, \dots, N$.

As given in (4.3), the kernel is evaluated at each pseudo-observation $\boldsymbol{u}_t^{(i)}, i = 1, \dots, N$ while centered at $\boldsymbol{y}_t$. We then need to tune the width at time t, denoted ϵ_t, so that a given fraction $\frac{\alpha}{N}$ of the pseudo-observations is covered by the p-HPR of the kernel. For the procedure to work, the kernel function κ must be invariant under translation and scaling, i.e., belong to the location-scale family of distributions. Many popular kernels including the three discussed above, belong to this family.

The tuning procedure involves two steps:

1. Identify $u_t^{[\alpha]}$, the αth closest pseudo-observation to y_t.
2. Center the kernel κ at y_t and set the width ϵ_t so that

$$\left| \int_{y_t}^{u_t^{[\alpha]}} \kappa(u_t; y_t, \epsilon_t) \, \mathrm{d}u_t \right| = \frac{p}{2}, \tag{4.4}$$

   meaning that $u_t^{[\alpha]}$ lies at the boundary of the p-HPR of $\kappa(\cdot; y_t, \epsilon_t)$.

These two steps are visualized in Figure 4.2. In the case of multidimensional $\boldsymbol{y}_t$ and $\boldsymbol{u}_t$, this procedure is performed coordinate-wise.

The meaning of equation (4.4) is that $u_t^{[\alpha]}$ is either the $\frac{1-p}{2}$-quantile or the $\frac{1+p}{2}$-quantile of $\kappa(\cdot; y_t, \epsilon_t)$, depending on whether $u_t^{[\alpha]} \leq y_t$ or $u_t^{[\alpha]} \geq y_t$. If we restrict ourselves to symmetric kernels, we may get rid of this case division by exploiting kernel symmetry.

Let F denote the cumulative distribution function of the kernel $\kappa(\cdot; 0, 1)$ centered at 0 with width $\epsilon = 1$. Let its quantile function be denoted by F^{-1}. From κ belonging to the location-scale family, we get that the quantile function of a general kernel $\kappa(\cdot; y_t, \epsilon_t)$ is

$$Q(\beta) = y_t + \epsilon_t F^{-1}(\beta), \quad \beta \in (0, 1). \tag{4.5}$$

As (4.4) and the assumed kernel symmetry require $\left| u_t^{[\alpha]} \right|$ to be the $\frac{1+p}{2}$-quantile of $\kappa(\cdot; y_t, \epsilon_t)$, we can substitute $u_t^{[\alpha]}$ for $Q(\beta)$ in (4.5) and solve for ϵ_t, obtaining

$$\epsilon_t = \frac{\left| u_t^{[\alpha]} - y_t \right|}{F^{-1}(\frac{1+p}{2})}, \tag{4.6}$$

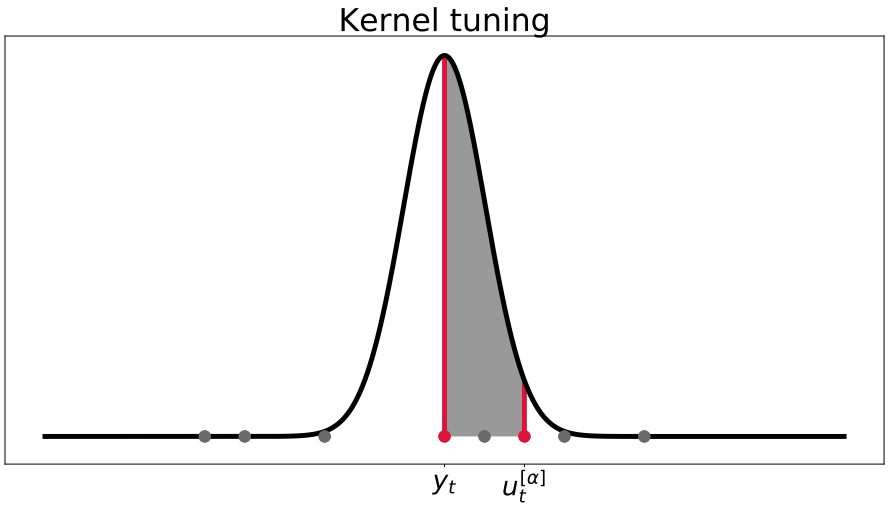


Figure 4.2: Visualization of the kernel tuning procedure. In this picture, $\alpha = 2$, $p = 0.95$, and the kernel is Gaussian. Plotted is the kernel along with a number of pseudo-observations $u_t^{(i)}$ and the true measurement y_t. Equation (4.4) states that the shaded area has volume $p/2 = 0.475$.

where the absolute value comes from the kernel being symmetric, so it is irrelevant whether we consider pseudo-observations lower or greater than the true observation y_t. The quantile function F^{-1} is uniquely associated with each kernel, and the only free parameters are α and p.

4.4 Likelihood estimate through ABC

The main contribution of this thesis is the utilization of the ABC framework to make inference possible even in SSMs with an unknown observation model. We have already derived all the necessary elements needed to state our main result. It remains to put them together by formalizing the entire inference process using ABC as a likelihood estimator.

Algorithm 8 presents a modification of the particle filter which uses the ABC approximation as a surrogate for the unknown observation model. Compared to the particle filter, our formulation is applicable even when the observation model $g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$ is not given as a probability density. Unlike the basic ABC filter described in Algorithm 7, our variant employs kernel functions to measure observation similarity, reflecting the fact that the closer a pseudo-measurement is to the true observation, the higher weight it should be assigned. In addition, we account for automatic tuning of the kernel widths by adapting them so that they cover a sufficient number of the simulated pseudo-measurements.

Finally, Algorithm 9 reformulates the marginal Metropolis-Hastings according to the ABC methodology. Under the new formulation, the estimator $\widehat{z}$ of $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})$ is constructed by evaluating (4.3) on a set of importance weights calculated by Algorithm 8. The result is used to compute the acceptance probability which again controls whether a proposed $\boldsymbol{\theta}'$ and the corresponding $\widehat{z}'$ are accepted or not.

Algorithm 8 ABC-based filter with automatic kernel tuning

Input: Number of particles N, current parameter value $\boldsymbol{\theta}$, HPR p, number of covered pseudo-observations α, $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$.

1: Sample $\boldsymbol{x}_0^{(i)} \sim p(\cdot \mid \boldsymbol{\theta}), \quad i = 1, \ldots, N.$ ▷ Initialize N particles.
2: $w_0^{(i)} \leftarrow \frac{1}{N}, \quad i = 1, \ldots, N.$ ▷ Initialize uniform weights.
3: **for** $t = 1$ **to** T **do**
4: Sample $\boldsymbol{x}_t^{(i)} \sim f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{\theta}), \quad i = 1, \ldots, N.$ ▷ Sample N new particles.
5: Simulate $\boldsymbol{u}_t^{(i)}$ from $g_t(\cdot \mid \boldsymbol{x}_t^{(i)}), \quad i = 1, \ldots, N.$ ▷ Simulate N pseudo-observations.
6: Identify $\boldsymbol{u}_t^{[\alpha]}$. ▷ Find the αth closest pseudo-observation to $\boldsymbol{y}_t$.
7: $\epsilon_t \leftarrow \frac{\left| u_t^{[\alpha]} - y_t \right|}{F^{-1}(\frac{1+p}{2})}$ ▷ Set the kernel width at time t according to (4.6).
8: Set $w_t^{(i)} \propto \kappa(\boldsymbol{u}_t^{(i)}; \boldsymbol{y}_t, \epsilon_t) w_{t-1}^{(i)}, \quad i = 1, \ldots, N.$
9: Resample $\boldsymbol{x}_t^{(i)}$ and reset $w_t^{(i)}$ using Algorithm 3, $\quad i = 1, \ldots, N.$
10: **end for**

Algorithm 9 Marginal Metropolis-Hastings with ABC filter

Input: Number of samples M, $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$.

1: Initialize $\boldsymbol{\theta}^{(0)}$.
2: Run Algorithm 8 with $\boldsymbol{\theta}^{(0)}$ to obtain the weights $w_{0,t}^{(i)}, \quad t = 1, \ldots, T, \; i = 1, \ldots, N.$
3: Calculate $\widehat{z}^{(0)}$ according to (4.3) using $w_{0,t}^{(i)}$.
4: **for** $m = 1$ **to** M **do**
5: Sample $\boldsymbol{\theta}' \sim q(\cdot \mid \boldsymbol{\theta}^{(m-1)})$.
6: Run Algorithm 8 with $\boldsymbol{\theta}'$ to obtain the weights $w_{m,t}^{(i)}, \quad t = 1, \ldots, T, \; i = 1, \ldots, N.$
7: Calculate $\widehat{z}'$ according to (4.3) using $w_{m,t}^{(i)}$.
8: Calculate the aceptance probability

$$\alpha = \min\left\{1, \frac{\widehat{z}' \pi(\boldsymbol{\theta}')}{\widehat{z}^{(m-1)} \pi(\boldsymbol{\theta}^{(m-1)})} \frac{q(\boldsymbol{\theta}^{(m-1)} \mid \boldsymbol{\theta}')}{q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(m-1)})}\right\}.$$

9: Sample $u \sim \mathcal{U}(0, 1)$.
10: **if** $u \leq \alpha$ **then**
11: $\left(\boldsymbol{\theta}^{(m)}, \widehat{z}^{(m)}\right) \leftarrow (\boldsymbol{\theta}', \widehat{z}')$ ▷ With probability α, accept the proposed sample.
12: **else**
13: $\left(\boldsymbol{\theta}^{(m)}, \widehat{z}^{(m)}\right) \leftarrow \left(\boldsymbol{\theta}^{(m-1)}, \widehat{z}^{(m-1)}\right)$ ▷ With probability $1 - \alpha$, reject the proposed sample.
14: **end if**
15: **end for**

Output: $\left\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}\right\}$

Chapter 5

Applications

Having investigated the theoretical properties of our method, it remains to empirically assess its performance. For the first test scenario, we have selected the Lotka-Volterra model described in Section 5.3. This is a relatively simple problem whose purpose is to validate our conclusions from the theoretical analysis. The second example is a simplified model for a prokaryotic auto-regulatory network addressed in Section 5.4. In both scenarios, the particle filter-based approach is compared with our ABC approximation under various model misspecifications.

The considered problems require simulating stochastic reactions in order to propagate the system states through time. This is done with the help of the Gillespie algorithm, which is first described in Section 5.2.

5.1 Implementation notes

All of the experiments described below have been implemented in Python 3.6.5. The performance-critical parts were additionally written in Cython to obtain C-like performance. The only additional dependencies are NumPy 1.14.3, SciPy 1.2.1, Matplotlib 2.2.2 and Statsmodels 0.9.0. The experiments have been performed on a standard laptop computer.

5.2 Preliminary: the Gillespie algorithm

The Gillespie algorithm (Gillespie, 1976, 1977) is used to simulate a stochastic process describing the time evolution of a system of reactions. The discussion given here follows Wilkinson (2011).

Time evolution of a reaction system Consider a system consisting of u species $\mathcal{X}_1, \ldots, \mathcal{X}_u$ and v reactions $\mathcal{R}_1, \ldots, \mathcal{R}_v$. The species can describe literal animal species, as is the case in the Lotka-Volterra model in Section 5.3, or individual molecule types, as in Section 5.4. The reactions describe the interactions between these species through time.

Let the number of molecules (or individuals, in case of animal species) of the species $\mathcal{X}_i$ at time t be denoted by $X_{i,t}$, and let $\boldsymbol{X}_t = (X_{1,t}, \ldots, X_{u,t})^\intercal$. Additionally, let the number of reactions of type $\mathcal{R}_i$ which occurred in a time window $(0, t]$ be denoted by $R_{i,t}$, and let $\boldsymbol{R}_t = (R_{1,t}, \ldots, R_{v,t})^\intercal$. The evolution of the system from time 0 to time t is described by the equation

$$\boldsymbol{X}_t - \boldsymbol{X}_0 = \mathbb{S}\boldsymbol{R}_t, \tag{5.1}$$

where $\mathbb{S} \in \mathbb{R}^{u \times v}$ is called the stoichiometry matrix of the system, and describes the difference in the number of molecules of each species after each reaction occurs. To gain

insight into the meaning of $\mathbb{S}$, it is instructive to write it as

$$\mathbb{S} = \mathbb{P}_{\text{post}} - \mathbb{P}_{\text{pre}},$$

where the element (i, j) of $\mathbb{P}_{\text{pre}}$ denotes the number of molecules of $\mathcal{X}_i$ before a reaction of type $\mathcal{R}_j$ takes place, and the element (i, j) of $\mathbb{P}_{\text{post}}$ describes the same quantity *after* it takes place. Equation (5.1) can then be written as

$$\boldsymbol{X}_t = \boldsymbol{X}_0 + \left(\mathbb{P}_{\text{post}} - \mathbb{P}_{\text{pre}}\right)\boldsymbol{R}_t,$$

and describes the net gain in the number of molecules of each species given their initial numbers, and accounting for their increase/decrease when a number of reactions of each type occurs.

In addition, each reaction $\mathcal{R}_i$ has a stochastic rate constant c_i and a rate law (also called the hazard function) $h_i(\boldsymbol{X}_t, c_i)$ associated with it. The interpretation of the hazard function is such that $h_i(\boldsymbol{X}_t, c_i)\mathrm{d}t$ is the probability of a reaction of type $\mathcal{R}_i$ occurring in a time interval $(t, t + \mathrm{d}t]$, conditionally on the system being in state $\boldsymbol{X}_t$. Such a situation is described by an exponential distribution – the time to the event of a reaction of type $\mathcal{R}_i$ occurring, assuming no other reaction is taking place, is distributed according to $\mathcal{E}xp\,(h_i(\boldsymbol{X}_t, c_i))$. This is however a convenient simplification, since multiple reactions are typically occurring at the same time.

The Gillespie algorithm In a system with v reactions and their hazard functions $h_i(\boldsymbol{X}_t, c_i)$, the hazard of *some* reaction occurring is

$$h_0(\boldsymbol{X}_t, \boldsymbol{c}) = \sum_{i=0}^{v} h_i(\boldsymbol{X}_t, c_i),$$

where $\boldsymbol{c} = (c_1, \ldots, c_v)^\mathsf{T}$. The time to the next reaction is then distributed according to $\mathcal{E}xp\,(h_0(\boldsymbol{X}_t, \boldsymbol{c}))$. The particular reaction type is a random variable with a categorical distribution $\mathcal{C}at\left(\widetilde{h}_1(\boldsymbol{X}_t, c_1), \ldots, \widetilde{h}_v(\boldsymbol{X}_t, c_v)\right)$, where $\widetilde{h}_i(\boldsymbol{X}_t, c_i) = \dfrac{h_i(\boldsymbol{X}_t, c_i)}{h_0(\boldsymbol{X}_t, \boldsymbol{c})}$.

With the above in mind, the Gillespie algorithm can now be formulated, and is given in Algorithm 10. Its purpose is to simulate the state evolution (5.1) for a given time horizon T while accounting for the randomness in the time until a reaction of a particular type takes place. For the purpose of this algorithm, denote the columns of the stoichiometry matrix $\mathbb{S}$ by $\boldsymbol{S}^i$, $\quad i = 1, \ldots, v$.

The algorithm is usually the bottleneck of most simulations, and must be implemented carefully; otherwise, the simulation becomes unacceptably slow. The final time t is at the output as well, since it may exceed the horizon T. If the algorithm is run consecutively during a simulation, the interest is to follow the previous run by starting at its final time t.

5.3 Lotka-Volterra model

5.3.1 Problem description

The first considered problem is the Lotka-Volterra model (Lotka, 1909; Volterra, 1928). The system describes a simplified time interaction of a population consisting of a predator and prey species. Denoting the prey species by $\mathcal{X}_1$ and the predator species by $\mathcal{X}_2$, the system can be described by the reactions

$$\mathcal{R}_1: \quad \mathcal{X}_1 \to 2\mathcal{X}_1, \tag{5.2}$$

$$\mathcal{R}_2: \quad \mathcal{X}_1 + \mathcal{X}_2 \to 2\mathcal{X}_2, \tag{5.3}$$

$$\mathcal{R}_3: \quad \mathcal{X}_2 \to \emptyset. \tag{5.4}$$

Algorithm 10 Gillespie algorithm

Input: Time horizon T, rate constants $\boldsymbol{c} = (c_1, \dots, c_v)^\intercal$, initial molecule numbers $\boldsymbol{X}_0$.

$t \leftarrow 0$
$\boldsymbol{X}_t \leftarrow \boldsymbol{X}_0$
while $t \leq T$ **do**
  Calculate $h_i(\boldsymbol{X}_t, c_i), \quad i = 1, \dots, v.$
  $h_0(\boldsymbol{X}_t, \boldsymbol{c}) \leftarrow \sum_{i=1}^{v} h_i(\boldsymbol{X}_t, c_i)$
  Calculate $\widetilde{h}_i(\boldsymbol{X}_t, c_i) = \dfrac{h_i(\boldsymbol{X}_t, c_i)}{h_0(\boldsymbol{X}_t, \boldsymbol{c})}, \quad i = 1, \dots, v.$
  Sample $\mathrm{d}t \sim \mathcal{E}xp\left(h_0(\boldsymbol{X}_t, \boldsymbol{c})\right)$. ▷ Simulate the time to the next reaction.
  Sample $i \sim \mathcal{C}at\left(\widetilde{h}_1(\boldsymbol{X}_t, c_1), \dots, \widetilde{h}_v(\boldsymbol{X}_t, c_v)\right)$. ▷ Simulate the reaction type.
  $\boldsymbol{X}_{t+\mathrm{d}t} \leftarrow \boldsymbol{X}_t + \boldsymbol{S}^i$ ▷ Update the state according to the reaction i.
  $t \leftarrow t + \mathrm{d}t$
end while

Output: Final state $\boldsymbol{X}_t$, final time t.

Equation (5.2) describes the reproduction of the prey species. Equation (5.3) describes the interaction between the predator and the prey where a predator consumes an individual of the prey species and produces an offspring. Equation (5.4) describes the extinction of the predator species when no prey is present.

The state of the system at time t is $\boldsymbol{X}_t = (X_{1,t}, X_{2,t})^\intercal$. The stoichiometry matrix is given by

$$\mathbb{S} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix},$$

and the hazard functions vector is $\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{c}) = (c_1 X_1, c_2 X_1 X_2, c_3 X_2)^\intercal$ (Golightly and Wilkinson, 2011). Although simple to describe, this model is analytically intractable (Wilkinson, 2011).

For the inference problem, we consider the unknown parameters to be $\boldsymbol{\theta} = (c_1, c_2, c_3)^\intercal$, and the state at time t to be $\boldsymbol{x}_t = (X_{1,t}, X_{2,t})^\intercal$. Since the rate constants c_1, c_2, c_3 are by definition positive, we are working in the log space to avoid restricting ourselves to positive support distributions. The model is specified by the following:

$$\begin{aligned} p(x_{1,0} \mid \boldsymbol{\theta}) &= \mathcal{P}o(50), \\ p(x_{2,0} \mid \boldsymbol{\theta}) &= \mathcal{P}o(100), \\ f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta}) &\text{ is simulated using Algorithm 10 }, \\ g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) &= \mathcal{N}_2\left(\boldsymbol{x}_t, 10^2\right). \end{aligned}$$

The parameter prior and the Metropolis-Hastings proposal are additionally given by

$$\begin{aligned} \pi(\log c_i) &= \mathcal{U}(-7, 2), \quad i = 1, 2, 3, \\ q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}) &= \mathcal{N}_3(\boldsymbol{\theta}, 0.01\mathbb{I}_3). \end{aligned}$$

The initial parameters are $\boldsymbol{\theta}^{(0)} = (1, 0.005, 0.6)^\intercal$. We simulate a sequence of 16 observations $\boldsymbol{y}_t$ using the Gillespie algorithm with parameters $\boldsymbol{\theta}^{(0)}$. The inference is started in the correct parameters and applied on a short sequence only; its purpose is only to demonstrate that the algorithm is able to identify these parameters.

We apply the marginal Metropolis-Hastings algorithm depending on the particle filter as well as the one utilizing ABC methods. Both algorithms are ran for $M = 50000$ samples with $N = 100$ particles. Additionally, the number of covered pseudo-observations in the ABC formulation is $\alpha = 90$ and the volume of the p-HPR is $p = 0.95$.

When using the ABC method, we simulate pseudo-observations from the observation model g_t without the noise term. That is, we simulate $\boldsymbol{u}_t = \boldsymbol{x}_t$, deterministically.

5.3.2 Inference using the particle filter

To start, we consider inference using Algorithm 5, i.e., using the particle filter to approximate the likelihood.

Correctly specified observation model At first, we assume the correct model specification. This means that the observations $\boldsymbol{y}_t$ have been corrupted by Gaussian noise, and the observation model g_t is Gaussian as well. In this situation, the particle filter is expected to perform well, as all assumptions have been met.

In Figure 5.1, Figure 5.2 and Figure 5.3, the results for the parameters c_1, c_2 and c_3, respectively, are shown. For each parameter, we show (in this order) the trace plot, the autocorrelation plot, and the histogram of sampled values. We use no burn-in period, as the inference starts in the correct values, but apply thinning of 100, i.e., keep every 100th sample. This ensures relatively uncorrelated samples, as is clear from the auto-correlation plots. The acceptation rate of the Metropolis-Hastings algorithms moves around 20 %.

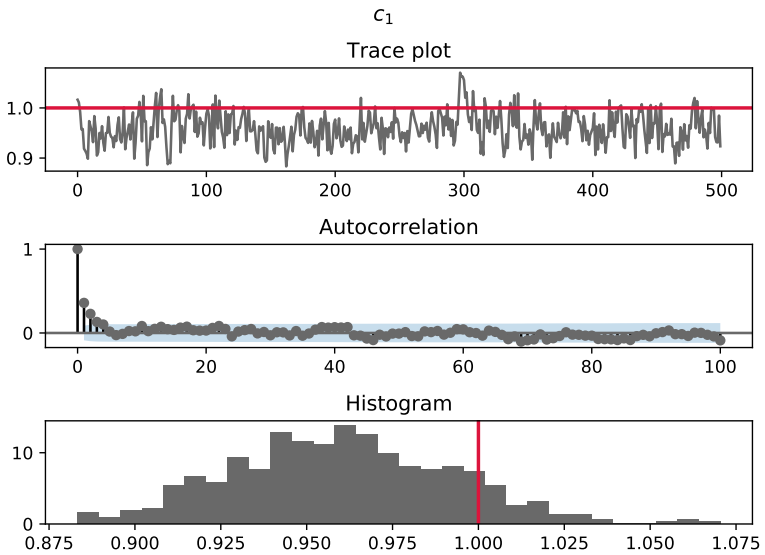


Figure 5.1: Particle filter-based inference of the parameter c_1 in the Lotka-Volterra model. Uses Gaussian noise and a Gaussian observation model. The true value is shown in red.

In all cases, the correct parameters are well-covered by the sampled values, while allowing for some degree of variance. The histograms provide estimated posterior distributions of the individual parameters. The sampled values can be used to provide point estimates or credible intervals for the true parameters.

Misspecified observation model Next, we keep the Gaussian observation model, but corrupt the observation sequence by a Cauchy noise with scale 10. Arguably, this scale is quite high, but is used to match the scale of the Gaussian noise from the previous section. The heavy-tailed Cauchy distribution allows sampling distant noise terms, and corrupts the observation sequence $\boldsymbol{y}_t$ much more severely. The Gaussian observation model g_t then assigns probability close to zero to these values, and the filter collapses. This is clear from Figure 5.4, Figure 5.5 and Figure 5.6.

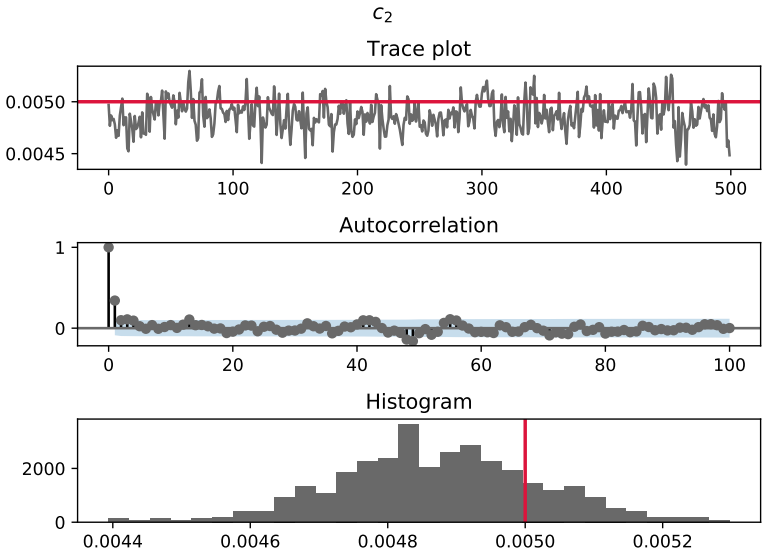


Figure 5.2: Particle filter-based inference of the parameter c_2 in the Lotka-Volterra model. Uses Gaussian noise and a Gaussian observation model. The true value is shown in red.

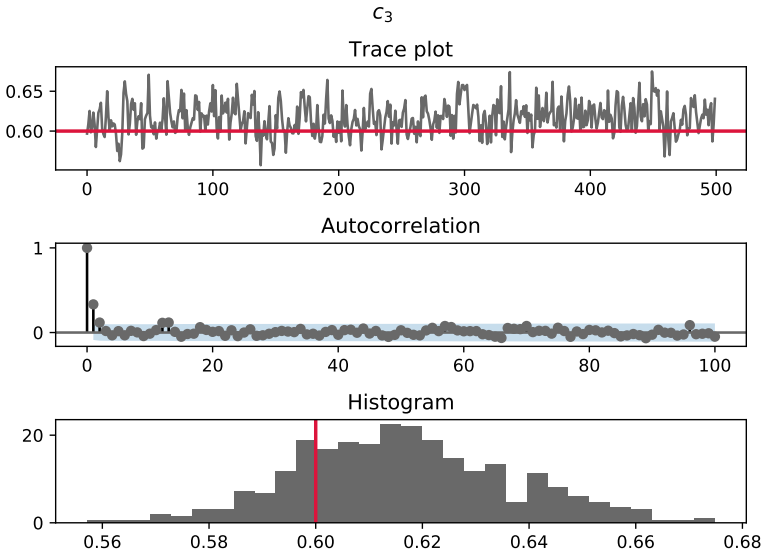


Figure 5.3: Particle filter-based inference of the parameter c_3 in the Lotka-Volterra model. Uses Gaussian noise and a Gaussian observation model. The true value is shown in red.

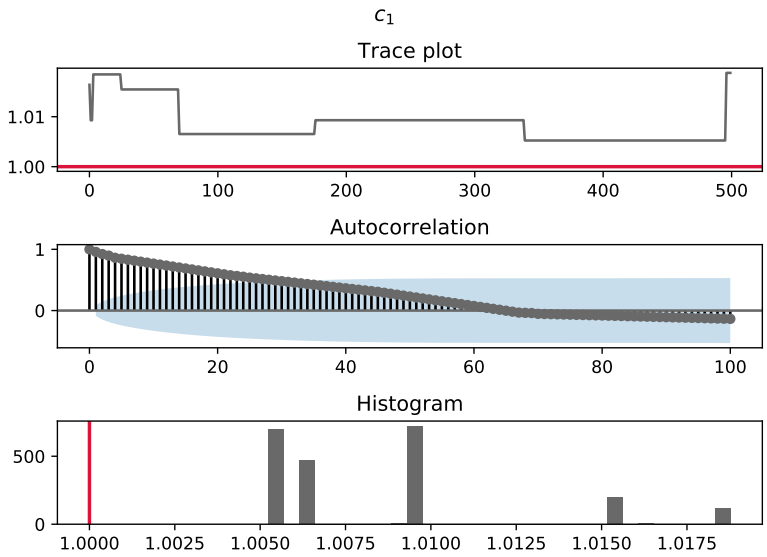


Figure 5.4: Particle filter-based inference of the parameter c_1 in the Lotka-Volterra model. Uses Cauchy noise and a Gaussian observation model. The true value is shown in red.

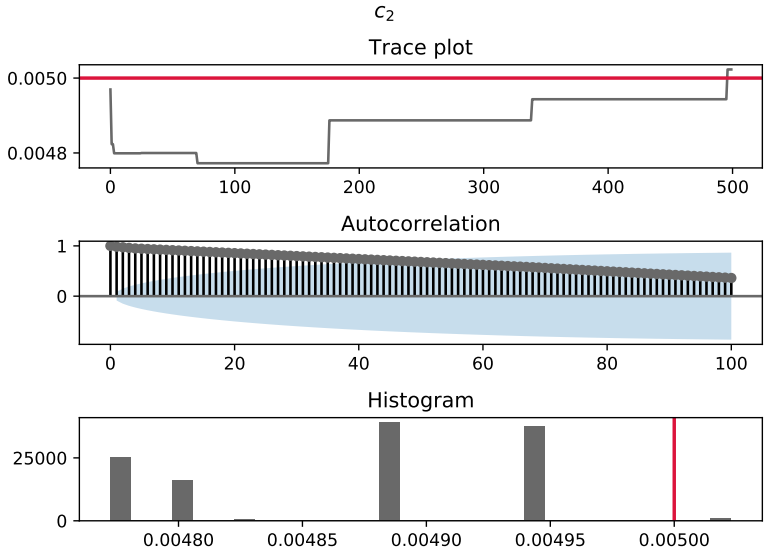


Figure 5.5: Particle filter-based inference of the parameter c_2 in the Lotka-Volterra model. Uses Cauchy noise and a Gaussian observation model. The true value is shown in red.

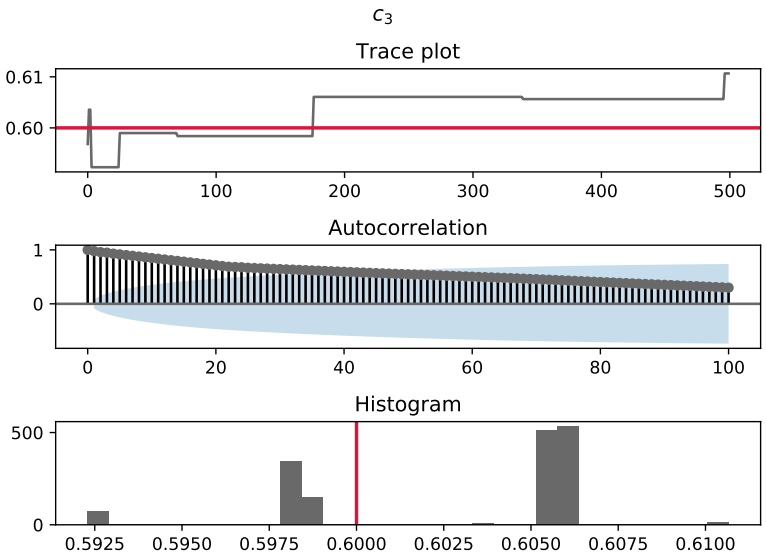


Figure 5.6: Particle filter-based inference of the parameter c_3 in the Lotka-Volterra model. Uses Cauchy noise and a Gaussian observation model. The true value is shown in red.

Clearly, Cauchy noise corrupts the sequence too much, and the results are poor. The accepted parameters are almost constant, and the posterior distribution is not even remotely-well approximated. This is an expected behavior, since the particle filter is known not to perform well under model misspecification.

5.3.3 Inference using ABC

Next, we apply Algorithm 9, the variant of the Metropolis-Hastings algorithm depending on ABC.

Gaussian noise, Gaussian kernel At first, we again corrupt the sequence y_t by a Gaussian noise, as indicated above. We then run Algorithm 9 with a Gaussian kernel to infer about the parameters θ. The results are shown in Figure 5.7, Figure 5.8 and Figure 5.9.

Compared to the results obtained by running the particle filter-based inference with a correct observation model, the results are slightly worse in the case of c_1, as the true value is in a region with a lower probability. Somewhat worse result is to be expected though, since the ABC methods provide only an approximation. Otherwise, the results are comparable to those utilizing the particle filter.

Cauchy noise, Gaussian kernel Next, we again corrupt the observation sequence by the heavy-tailed Cauchy noise. First, we keep the Gaussian kernel to calculate the importance weights. The results are in Figure 5.10, Figure 5.11 and Figure 5.12.

Compared to the particle filter with a misspecified observation model, the filter does not collapse at all. Instead, it remains stable, and the results resemble those obtained from a particle filter assuming a correct observation model, or those given by the previous ABC use-case.

This shows the strength of the ABC approximation – even under a heavy-tailed noise such as the Cauchy one, using a set of simulated pseudo-observations with a suitable kernel function allows the likelihood estimate to remain stable even under a severe model misspecification.

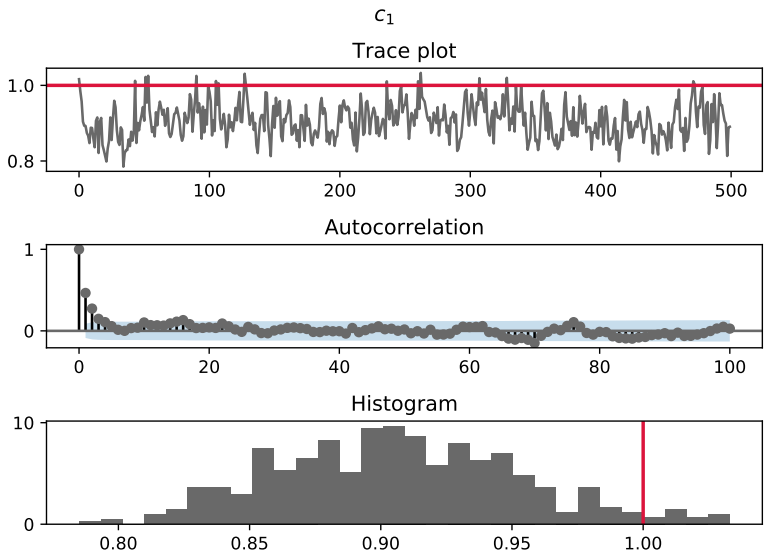


Figure 5.7: ABC-based inference of the parameter c_1 in the Lotka-Volterra model. Uses Gaussian noise and a Gaussian kernel. The true value is shown in red.

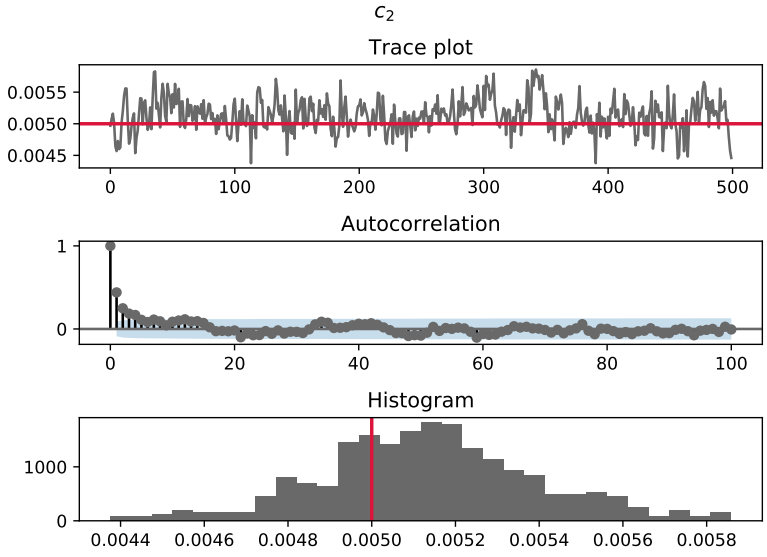


Figure 5.8: ABC-based inference of the parameter c_2 in the Lotka-Volterra model. Uses Gaussian noise and a Gaussian kernel. The true value is shown in red.

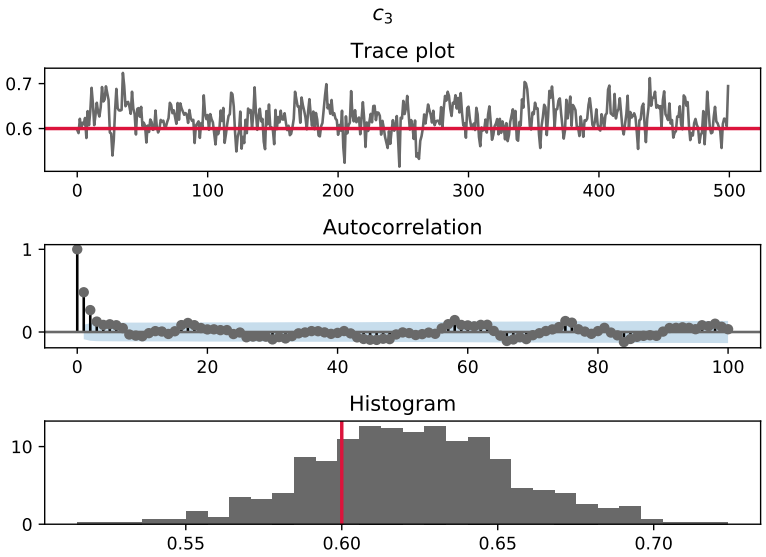


Figure 5.9: ABC-based inference of the parameter c_3 in the Lotka-Volterra model. Uses Gaussian noise and a Gaussian kernel. The true value is shown in red.

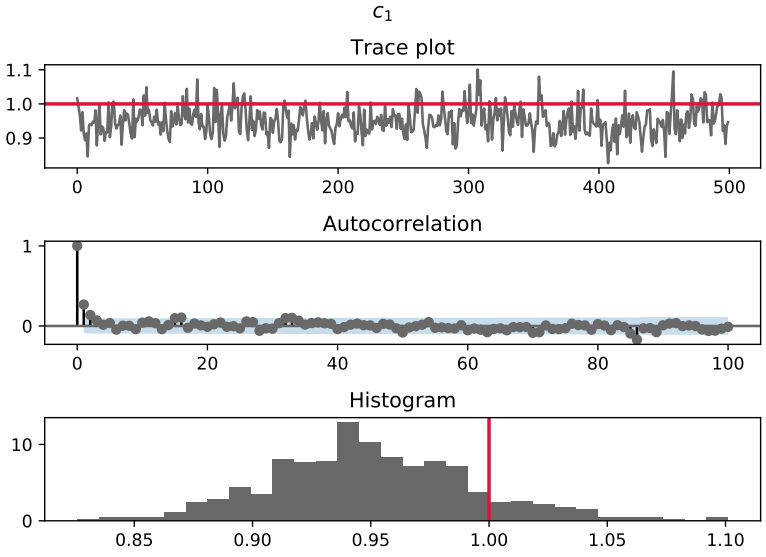


Figure 5.10: ABC-based inference of the parameter c_1 in the Lotka-Volterra model. Uses Cauchy noise and a Gaussian kernel. The true value is shown in red.

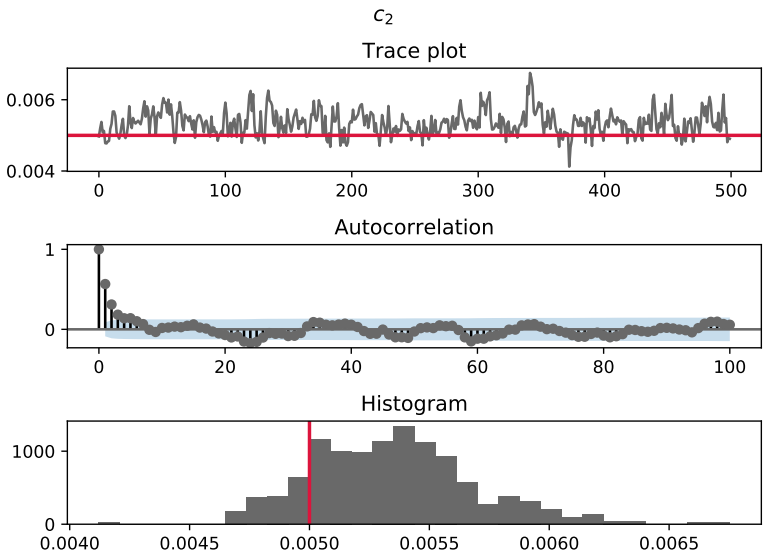


Figure 5.11: ABC-based inference of the parameter c_2 in the Lotka-Volterra model. Uses Cauchy noise and a Gaussian kernel. The true value is shown in red.

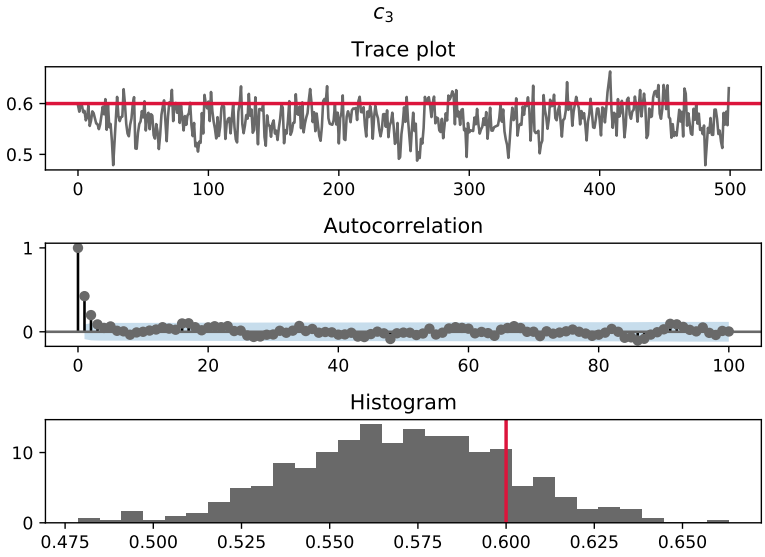


Figure 5.12: ABC-based inference of the parameter c_3 in the Lotka-Volterra model. Uses Cauchy noise and a Gaussian kernel. The true value is shown in red.

Cauchy noise, Cauchy kernel Finally, we repeat the same experiment as in the previous section, but use a Cauchy kernel instead of the Gaussian one. The results are very similar to those obtained using a Gaussian kernel, indicating that the filter is fairly robust to kernel choice. This agrees with the conclusion provided by Dedecius (2017).

The Cauchy kernel might be preferable to the Gaussian one for computational reasons – its quantile function (required for kernel width tuning) can be calculated without resorting to numerical approximations.

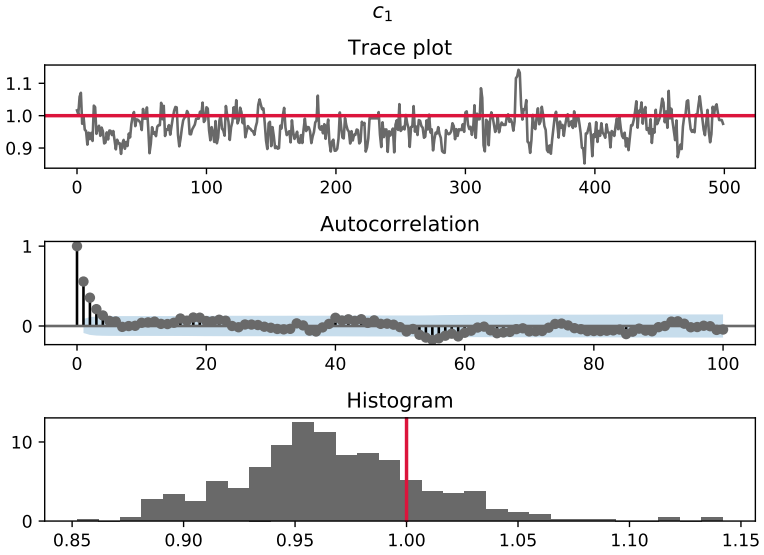


Figure 5.13: ABC-based inference of the parameter c_1 in the Lotka-Volterra model. Uses Cauchy noise and a Cauchy kernel. The true value is shown in red.

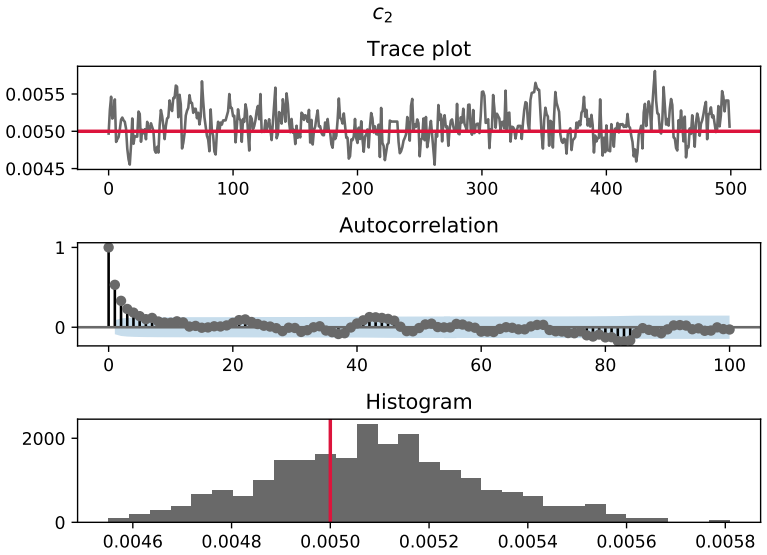


Figure 5.14: ABC-based inference of the parameter c_2 in the Lotka-Volterra model. Uses Cauchy noise and a Cauchy kernel. The true value is shown in red.

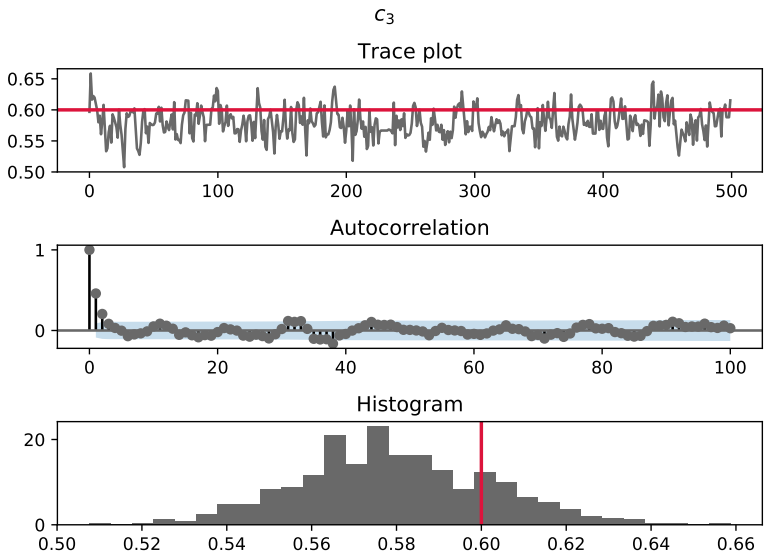


Figure 5.15: ABC-based inference of the parameter c_3 in the Lotka-Volterra model. Uses Cauchy noise and a Cauchy kernel. The true value is shown in red.

5.3.4 Experiment conclusion

The Lotka-Volterra simulation study demonstrated the performance of our algorithm and confirmed our expectations about the differences between the particle filter and our ABC-based method. As indicated in Chapter 3, the particle filter-based inference using the correct observation model gives the best results. The sampled values well cover the true parameter and have a reasonable degree of variance. On the other hand, when the observation model is misspecified, the filter completely collapses, and the inference becomes unreliable.

Application of the ABC framework to the correctly specified model introduces some bias which was theoretically justified in Chapter 4. The method achieves a slightly worse precision than the particle filter, although the results are still sensible. ABC mainly excels under a misspecified model and prevents the filter from collapsing, while still achieving results comparable to the correctly parameterized particle filter. In addition, the ABC filter is not particularly sensitive to the kernel choice. These points make our method a superior choice when the true observation model is not known, as it allows for stable inference at a similar computational cost.

5.4 Prokaryotic auto-regulation model

5.4.1 Problem description

The second considered model comes from Golightly and Wilkinson (2005, 2011), and describes a simplified prokaryotic autoregulatory gene network. The discussion below follows these two papers.

The model assumes that the dimer of a protein P, denoted P_2, represses the transcription of its coding gene by binding to a regulatory region in the gene. This transcription begins by binding of RNA-polymerase to the promoter of the gene. As the RNA-polymerase moves along the gene, it transcripts the gene into mRNA. The transcription is repressed by binding of P_2 to regions od the gene called operators. The repression and

transcription can be described in a simplified way by the following reactions:

$$\begin{aligned}\mathcal{R}_1: &\quad DNA + P_2 \to DNA \cdot P_2\\ \mathcal{R}_2: &\quad DNA \cdot P_2 \to DNA + P_2\\ \mathcal{R}_3: &\quad DNA \to DNA + RNA.\end{aligned}$$

The entire process of translating the mRNA into the protein P is summarized by the single reaction

$$\mathcal{R}_4: \quad RNA \to RNA + P.$$

The reversible dimerization of P is described by

$$\begin{aligned}\mathcal{R}_5: &\quad 2P \to P_2\\ \mathcal{R}_6: &\quad P_2 \to 2P.\end{aligned}$$

The remaining two reactions describe mRNA and protein degradation

$$\begin{aligned}\mathcal{R}_7: &\quad RNA \to \emptyset\\ \mathcal{R}_8: &\quad P \to \emptyset.\end{aligned}$$

The state of the system at time t is $\boldsymbol{X}_t = (RNA_t, P_t, (P_2)_t, (DNA \cdot P_2)_t, DNA_t)^\intercal$. The stoichiometry matrix is given by

$$\mathbb{S} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1 & -2 & 2 & 0 & -1\\ -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0\\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{5.5}$$

and the hazard function vector is (Golightly and Wilkinson, 2011)

$$\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{c}) = \left(c_1 P_2 DNA, c_2 DNA \cdot P_2, c_3 DNA, c_4 RNA, \frac{c_5 P(P-1)}{2}, c_6 P_2, c_7 RNA, c_8 P \right)^\intercal.$$

The matrix (5.5) has linearly dependent rows – we can add the last two rows together to produce a zero vector. Given the variable ordering in $\boldsymbol{X}_t$, this implies that

$$DNA \cdot P_2 + DNA = k,$$

where k is a constant. This is equation is called a conservation law, and the constant k denotes the number of copies of the particular gene in the genome. From this conservation law, it follows that $DNA \cdot P_2 = k - DNA$, which we substitute into the system described above. This results in a system of four variables only, denoted $\boldsymbol{X}_t = (RNA_t, P_t, (P_2)_t, DNA_t)^\intercal$.

This substitution simplifies the stoichiometry matrix (5.5) into

$$\mathbb{S} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1 & -2 & 2 & 0 & -1\\ -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0\\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

and the hazard function vector into

$$\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{c}) = \left(c_1 P_2 DNA, c_2 (k - DNA), c_3 DNA, c_4 RNA, \frac{c_5 P(P-1)}{2}, c_6 P_2, c_7 RNA, c_8 P \right)^\intercal. \tag{5.6}$$

For the inference problem, the unknown parameters are $\boldsymbol{\theta} = (c_1, c_2, c_3, c_4, c_7, c_8)^\intercal$. The parameters c_5 and c_6 are assumed to be known, exactly as in Golightly and Wilkinson (2011). The latent state at time t is $\boldsymbol{x}_t = (RNA_t, P_t, (P_2)_t, DNA_t)^\intercal$. We again work in the log-space, as the rate constants are positive by definition. The model described by Golightly and Wilkinson (2011) assumes the initial state to be known $\boldsymbol{x}_0 = (8, 8, 8, 5)^\intercal$, $k = 10$, $c_5 = 0.1$, $c_6 = 0.9$, and that

$$f_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{\theta}) \text{ is simulated using Algorithm 10 ,}$$
$$g_t(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) = \mathcal{N}\left(P_t + 2(P_2)_t, 2^2\right).$$

This means we are only able to observe the protein concentrations, subject to error. The problem becomes considerably more challenging compared to the Lotka-Volterra model, where we observed the entire state vector (up to noise). Here we only observe a noisy linear combination of two elements of the latent state, which makes the inference much more difficult. The biological interpretation is that we are not able to distinguish the protein monomers from its dimers.

The parameter prior and the Metropolis-Hastings proposal are additionally given by

$$\pi(\log c_i) = \mathcal{U}(-7, 2), \quad i = 1, 2, 3, 4, 7, 8,$$
$$q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}) = \mathcal{N}_6(\boldsymbol{\theta}, 0.08\mathbb{I}_6).$$

The initial parameters are sampled from the prior distributions. We simulate the data $\boldsymbol{y}_t$ using Algorithm 10 and obtain a sequence of $T = 237$ observations $\boldsymbol{y}_t \sim \mathcal{N}\left(P_t + 2(P_2)_t, 2^2\right)$.

The marginal Metropolis-Hastings is run again for $M = 50000$ samples, this time with $N = 200$ particles. We use a burn-in period of 10000 and thin the samples by 25. The ABC parameters are $\alpha = 180$ and $p = 0.95$. The pseudo-observations are again simulated from g_t without the noise term.

5.4.2 Inference using the particle filter

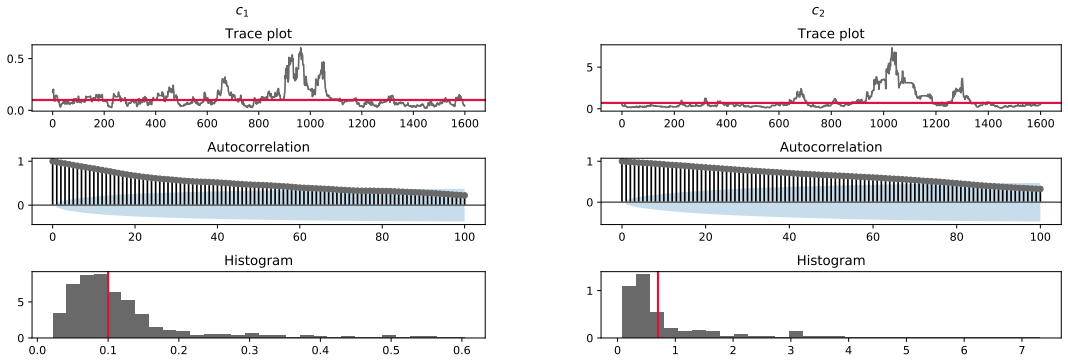
We again start by applying the particle filter-based algorithm. At first, we examine the results under a well-specified model. Next, we use a misspecified model by once again corrupting the input sequence by the Cauchy noise.

Correctly specified observation model At first, we consider the model exactly as described above, and perform inference using the particle filter. This is a well-specified model, since the observation model g_t coincides with the noise on the data sequence $\boldsymbol{y}_{1:T}$. The results are shown in Figure 5.16, Figure 5.17 and Figure 5.18.

Most of the parameters (apart from c_4) are well-covered by the posterior samples, agreeing with the results of Golightly and Wilkinson (2011). Even after thinning, we see that the samples are highly correlated. The original authors did not provide any autocorrelation plots, so the results cannot be compared. Having such correlated samples means that calculating empirical estimates would be unreliable, as they typically require independent observations. This issue could be addressed by considering more complex proposal distributions or samplers.

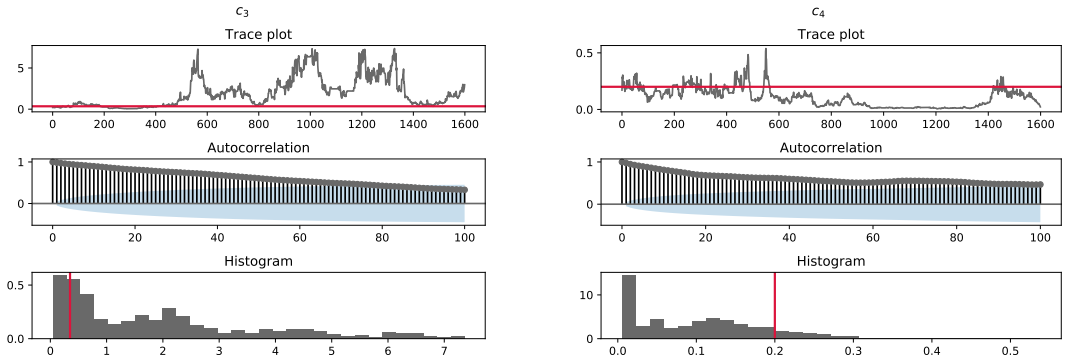
Misspecified observation model Next, we examine the filter behavior under a misspecified observation model. We still use g_t as described above (i.e., Gaussian), but corrupt the input sequence $\boldsymbol{y}_{1:T}$ by a Cauchy noise instead of Gaussian, keeping the scale parameter at 2. The results are shown in Figure 5.19, Figure 5.20 and Figure 5.21.

The results are (as expected) similar to the same situation in the Lotka-Volterra study. The particle filter completely collapses under a misspecified model and is unusable for any statistical inference.



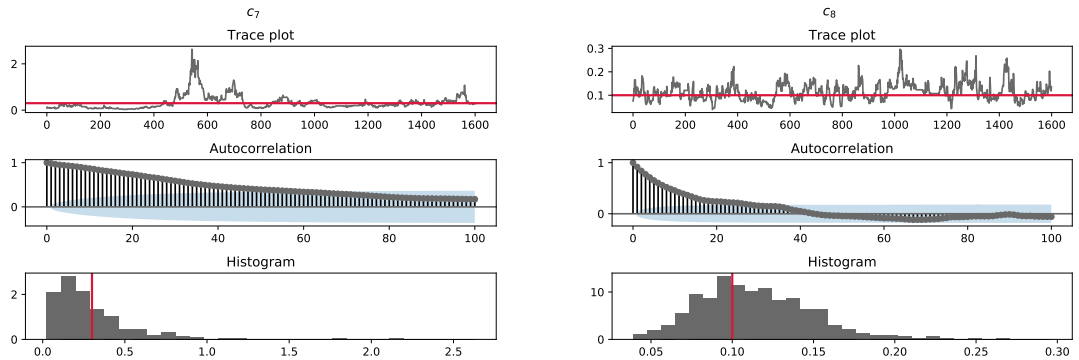
(a) c_1 (b) c_2

Figure 5.16: Particle filter-based inference of the parameters c_1 and c_2 in the auto-regulation model. Uses Gaussian noise and a Gaussian observation model. The true values are shown in red.



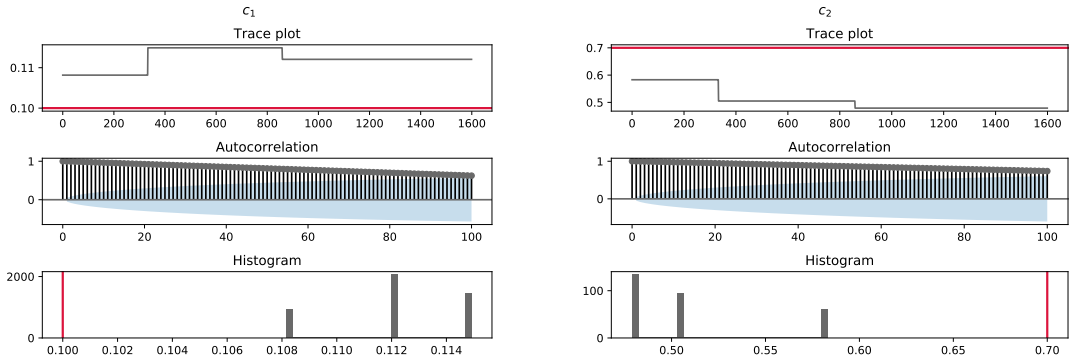
(a) c_3 (b) c_4

Figure 5.17: Particle filter-based inference of the parameters c_3 and c_4 in the auto-regulation model. Uses Gaussian noise and a Gaussian observation model. The true values are shown in red.



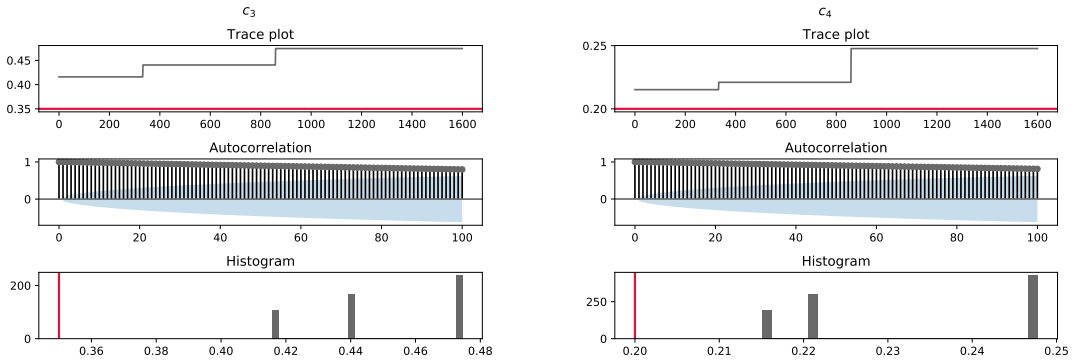
(a) c_7 (b) c_8

Figure 5.18: Particle filter-based inference of the parameters c_7 and c_8 in the auto-regulation model. Uses Gaussian noise and a Gaussian observation model. The true values are shown in red.



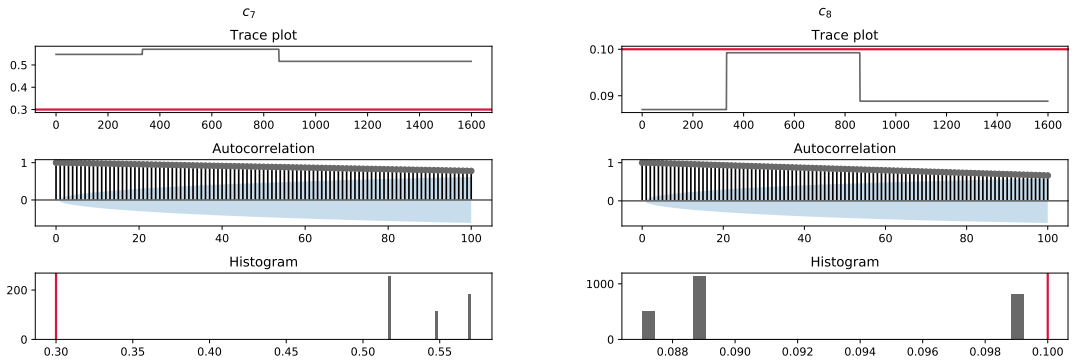
(a) c_1 (b) c_2

Figure 5.19: Particle filter-based inference of the parameters c_1 and c_2 in the autoregulation model. Uses Cauchy noise and a Gaussian observation model. The true values are shown in red.



(a) c_3 (b) c_4

Figure 5.20: Particle filter-based inference of the parameters c_3 and c_4 in the autoregulation model. Uses Cauchy noise and a Gaussian observation model. The true values are shown in red.



(a) c_7 (b) c_8

Figure 5.21: Particle filter-based inference of the parameters c_7 and c_8 in the autoregulation model. Uses Cauchy noise and a Gaussian observation model. The true values are shown in red.

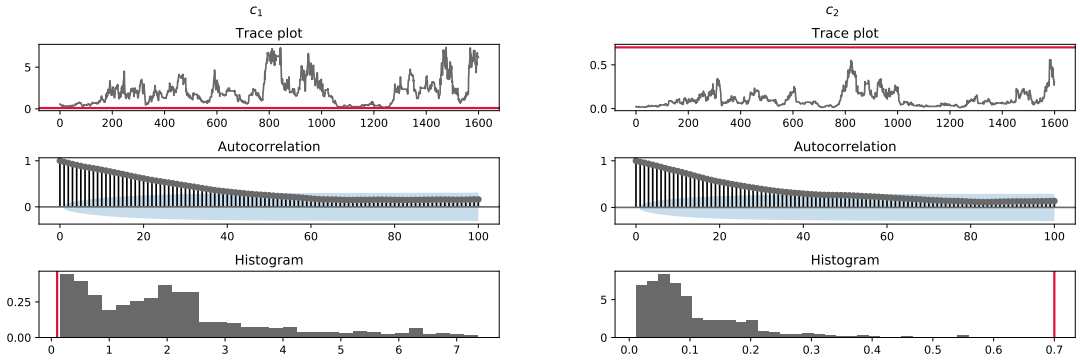
5.4.3 Inference using ABC

Next, we apply the marginal Metropolis-Hastings algorithm depending on the ABC methods.

Gaussian noise, Gaussian kernel At first, we use an input sequence corrupted by a Gaussian noise, as described above, and use ABC with a Gaussian kernel. The results are shown in Figure 5.22, Figure 5.23 and Figure 5.24.

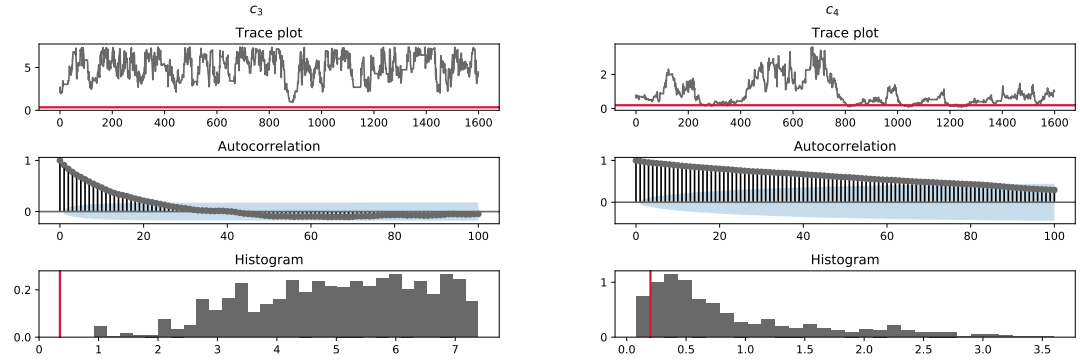
Unfortunately, it appears that the ABC approximation introduces too much bias to the inference, as only two parameters (c_4 and c_7) are correctly identified. In addition, the samples for c_8 are not as far off, judging from the x-axis labels. Samples for the remaining parameters have diverged from the true values. As was the case with the particle filter, there is a strong autocorrelation present in the sampled values.

Some indications given in Wilkinson (2011) state that ABC (though applied in a different context than SSMs) together with the simple Gillespie algorithm can lead to poor results. To obtain better estimates, more sophisticated reaction simulators should be considered for the complex problems studied in molecular biology.



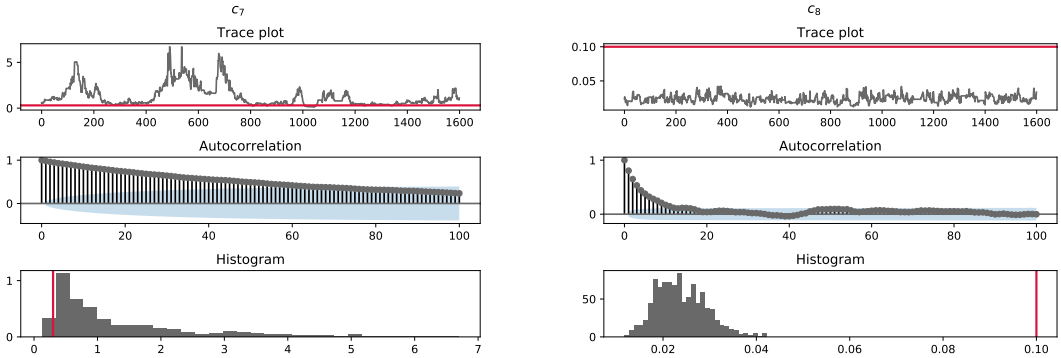
(a) c_1 (b) c_2

Figure 5.22: ABC-based inference of the parameters c_1 and c_2 in the auto-regulation model. Uses Gaussian noise and a Gaussian kernel. The true values are shown in red.



(a) c_3 (b) c_4

Figure 5.23: ABC-based inference of the parameters c_3 and c_4 in the auto-regulation model. Uses Gaussian noise and a Gaussian kernel. The true values are shown in red.



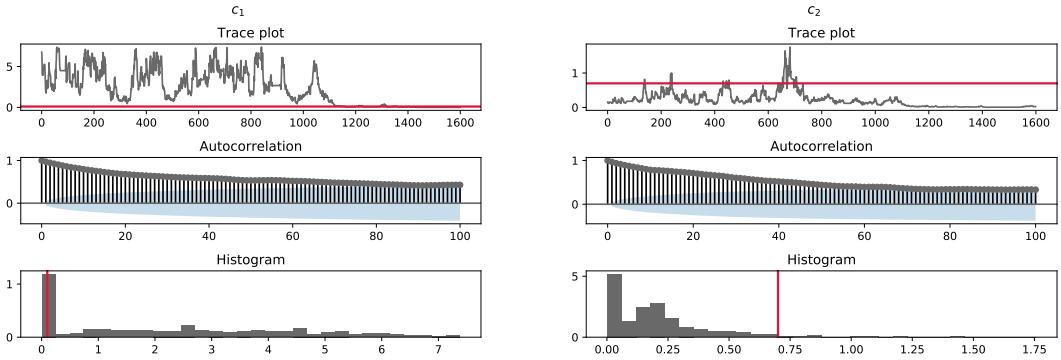
(a) c_7 (b) c_8

Figure 5.24: ABC-based inference of the parameters c_7 and c_8 in the auto-regulation model. Uses Gaussian noise and a Gaussian kernel. The true values are shown in red.

Cauchy noise, Gaussian kernel Next, we again apply the heavy-tailed Cauchy noise to the input sequence, but keep using the Gaussian kernel. We are interested to see whether the filter collapses, as was the case in particle filter-based inference under a misspecified model. The results are shown in Figure 5.25, Figure 5.26 and Figure 5.27.

The situation is similar to the previous experiment with Gaussian noise. The parameters c_1, c_4 and to a lesser degree, c_8, are identified. The samples of the other parameters diverge.

Unlike the particle filter, our method again does not collapse under model misspecification. It is still a better choice if we do not have access to the correct model, as it manages to identify at least some parameters.

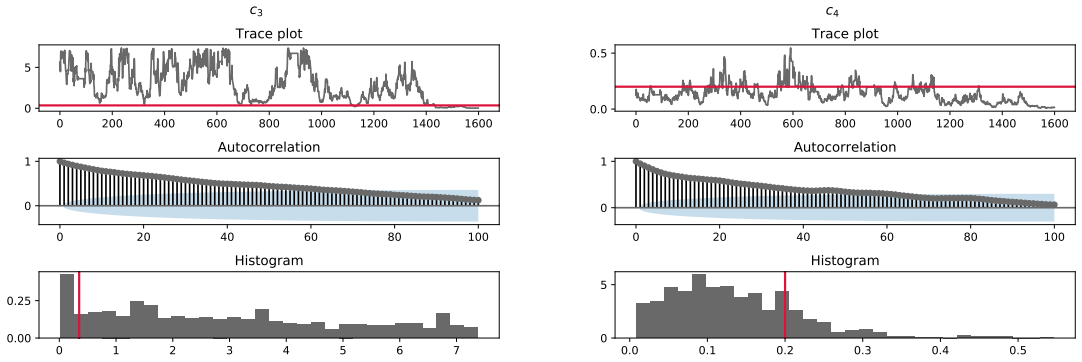


(a) c_1 (b) c_2

Figure 5.25: ABC-based inference of the parameters c_1 and c_2 in the auto-regulation model. Uses Cauchy noise and a Gaussian kernel. The true values are shown in red.

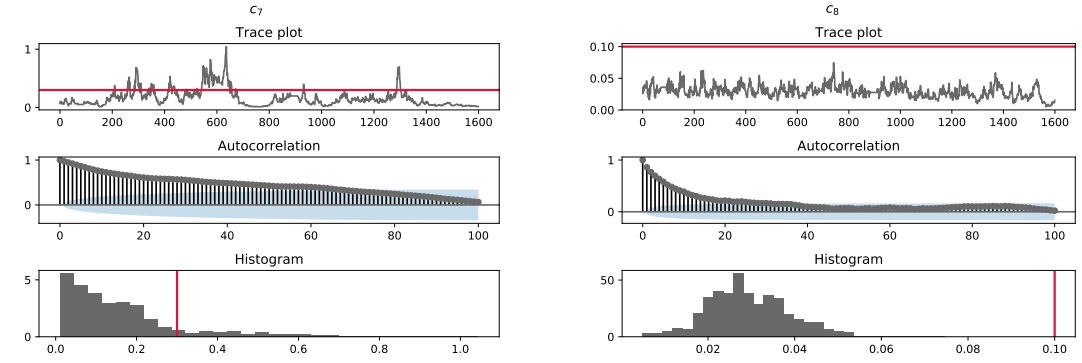
Cauchy noise, Cauchy kernel Finally, we are interested to see what happens when we keep the Cauchy noise on the input sequence but use the Cauchy kernel. The results are shown in Figure 5.28, Figure 5.29 and Figure 5.30.

As expected, the filter does not collapse. There is still a notable identification problem, similarly to the the previous two experiments – a simple change in the kernel function cannot significantly improve the estimate quality. Compared to the Gaussian kernel experiment, the sampled values are somewhat more spread out.



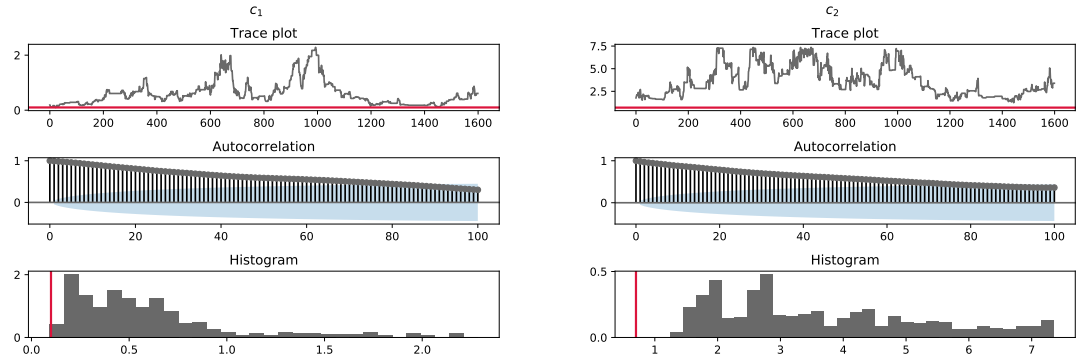
(a) c_3 (b) c_4

Figure 5.26: ABC-based inference of the parameters c_3 and c_4 in the auto-regulation model. Uses Cauchy noise and a Gaussian kernel. The true values are shown in red.



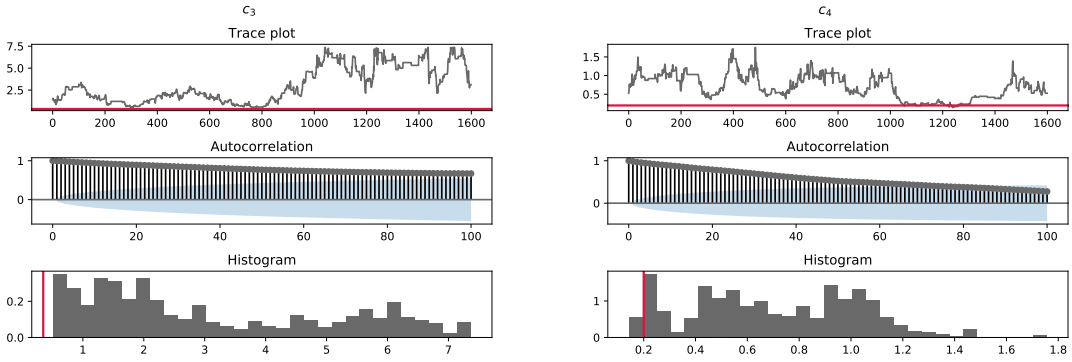
(a) c_7 (b) c_8

Figure 5.27: ABC-based inference of the parameters c_7 and c_8 in the auto-regulation model. Uses Cauchy noise and a Gaussian kernel. The true values are shown in red.



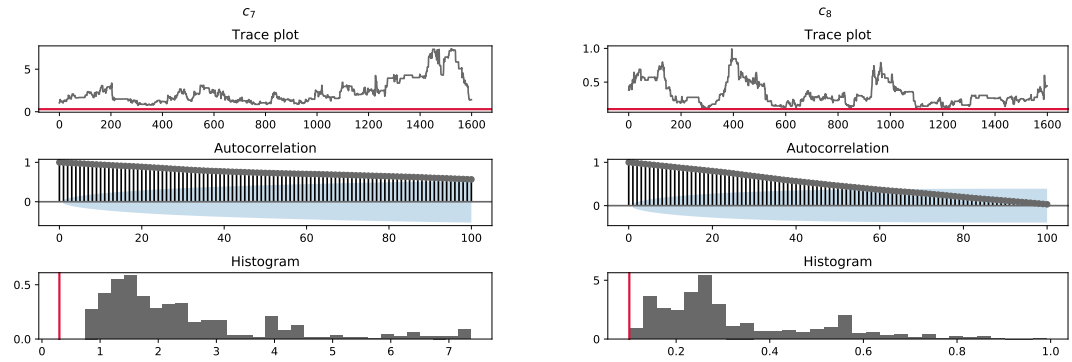
(a) c_1 (b) c_2

Figure 5.28: ABC-based inference of the parameters c_1 and c_2 in the auto-regulation model. Uses Cauchy noise and a Cauchy kernel. The true values are shown in red.



(a) c_3 (b) c_4

Figure 5.29: ABC-based inference of the parameters c_3 and c_4 in the auto-regulation model. Uses Cauchy noise and a Cauchy kernel. The true values are shown in red.



(a) c_7 (b) c_8

Figure 5.30: ABC-based inference of the parameters c_7 and c_8 in the auto-regulation model. Uses Cauchy noise and a Cauchy kernel. The true values are shown in red.

5.4.4 Experiment conclusion

We applied our algorithm to a simplified representation of a prokaryotic autoregulatory network. As in the Lotka-Volterra study, the latent spaces were again simulated using the Gillespie algorithm. In this experiment, inference was much more difficult, as the observed sequence $\boldsymbol{y}_t$ was only a one-dimensional linear combination of two of the four elements of the state vector $\boldsymbol{x}_t$.

The particle filter managed to well identify the static parameter under a correct observation model. When this model was misspecified, the filter completely collapsed and the results became unusable, as was expected.

Our ABC-based method introduced too much bias and encountered identification problems in about half of the unknown parameters. We attempted to increase the number of particles N and carefully tune the scale of the proposal distribution, but the results remained similar. Still, the ABC filter does not collapse even when applied to a sequence corrupted by heavy-tailed noise. Our method should be considered when we suspect that the observation model is unknown, as it remains stable and identifies at least a subset of the parameters.

One thing to note is due to the form of the hazard function (5.6), the parameters c_2 and c_3, and c_4 and c_7 are correlated, as they appear together with the same nucleic acids. This may introduce additional difficulties with parameter identification. Another possible cause for the unsatisfactory results is the simple Gillespie algorithm. It is likely that using a more complex simulation mechanism would lead to better results; this is a possibility left for future work.

Chapter 6

Conclusion and future work

This thesis deals with static parameter inference in state-space models (SSMs). We approach the problem using the probabilistically consistent and versatile Bayesian framework. This involves formulating a prior density and inferring a posterior distribution of the static parameter given an observed sequence. The inference process is complicated by the intractable likelihood of the state-space model, which prevents the application of standard Bayesian methods.

The state-of-the-art approach to the problem is to approximate the unknown posterior distribution using Markov Chain Monte Carlo (MCMC) sampling from the static parameter space and employing a "nested" sequential Monte Carlo filter – i.e., a particle filter – for the likelihood evaluation. In particular, this likelihood serves to calculate the MCMC acceptance probability. This is known as the particle Markov Chain Monte Carlo (PMCMC) algorithm. The particle filter can be proved to preserve the asymptotic properties of the sampler, even though the likelihood is only approximated.

A common drawback of this approach is the assumptions of a fully specified data-generating mechanism in the form of a probability density function. In applications, this assumption is often violated and calls for approximations. In this thesis, a novel method inspired by the recently developed approximate Bayesian computation (ABC) filters is proposed. While it preserves the MCMC part of the algorithm, it replaces the particle filter by an adaptive ABC filter. The likelihood is then approximated by applying this ABC filter to each sample from the static parameter space. Compared to the particle filter, this method does not require the data-generating model to be probabilistic and instead allows for deterministic functions commonly occurring in practice.

The resulting algorithm does not introduce any additional computational complexity over the particle filter. Unlike PMCMC, our method does not collapse under a misspecified observation model of the SSM and remains stable even when the observed sequence is corrupted by a heavy-tailed noise. Under a known data-generating model, our algorithm necessarily performs worse than the particle filter, since it brings an additional level of approximation. The bias of the ABC method is only mild in the simulation study and the results are comparable to the particle filter. It is more notable in the much more complex autoregulation model but arguably, the Gillespie algorithm used to simulate reactions is too simplistic. It is likely that employing a more complex simulator would represent the biological process more faithfully and allow for more precise inference.

In future work, generalizations of the adaptive kernel tuning to multiple dimensions should be considered. The tuning procedure utilized in this thesis has been derived in context of one-dimensional kernels and applied coordinate-wise in a multidimensional setting. As a consequence, the individual observation vector elements are assumed independent. More reliable likelihood estimates and, in turn, closer representation of the static parameter posterior could be obtained by exploiting the observation dependencies.

If applications to more complicated biological systems was of interest, one should study more sophisticated ways of simulating the latent states. The Gillespie algorithm is still a naïve method, and fails to cover the details present in systems as complex as those found in molecular biology.

Bibliography

C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods (with discussion). *Journal of the Royal Statistical Society, Series B*, 72:1–33, 01 2010.

G. E. Box. Robustness in the strategy of scientific model building. In *Robustness in statistics*, pages 201–236. Elsevier, 1979.

S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

F. d'Alché Buc, M. Quach, and N. Brunel. Estimating parameters and hidden variables in non-linear state-space models based on ODEs for biological networks inference. *Bioinformatics*, 23(23):3209–3216, 12 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm510. URL https://doi.org/10.1093/bioinformatics/btm510.

K. Dedecius. Adaptive kernels in approximate filtering of state-space models. *International Journal of Adaptive Control and Signal Processing*, 31(6):938–952, 2017. doi: 10.1002/acs.2739. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/acs.2739.

P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems With Applications*, volume 100. 05 2004. ISBN 0387202684. doi: 10.1007/978-1-4684-9393-1.

R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. pages 64 – 69, 10 2005. ISBN 953-184-089-X. doi: 10.1109/ISPA.2005.195385.

A. Doucet, A. Smith, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Information Science and Statistics. Springer New York, 2001. ISBN 9780387951461. URL https://books.google.cz/books?id=uxX-koqKtMMC.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

P. Fearnhead and D. Prangle. Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474. doi: 10.1111/j.1467-9868.2011.01010.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2011.01010.x.

D. T. Gillespie. A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *Journal of Computational Physics*, 22:403–434, Dec. 1976. doi: 10.1016/0021-9991(76)90041-3.

D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977. doi: 10.1021/j100540a008. URL https://doi.org/10.1021/j100540a008.

A. Golightly and D. J. Wilkinson. Bayesian inference for stochastic kinetic models using a diffusion approximation. *Biometrics*, 61(3):781–788, 2005.

A. Golightly and D. J. Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle markov chain monte carlo. *Interface focus*, 1:807–20, 12 2011. doi: 10.1098/rsfs.2011.0047.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. doi: 10.1093/biomet/57.1.97. URL http://biomet.oxfordjournals.org/cgi/content/abstract/57/1/97.

A. Jasra. Approximate bayesian computation for a class of time series models. *International Statistical Review*, 83(3):405–435, 2015. doi: 10.1111/insr.12089. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12089.

A. Jasra, S. S. Singh, J. S. Martin, and E. McCoy. Filtering via approximate bayesian computation. *Statistics and Computing*, 22(6):1223–1237, Nov 2012. ISSN 1573-1375. doi: 10.1007/s11222-010-9185-0. URL https://doi.org/10.1007/s11222-010-9185-0.

S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. pages 182–193, 1997.

R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering (ASME)*, 82D:35–45, 01 1960. doi: 10.1115/1.3662552.

J. Lintusaari, M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Fundamentals and recent developments in approximate bayesian computation. *Systematic biology*, 66(1): e66–e82, 2017.

A. J. Lotka. Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, 14(3):271–274, 1909. doi: 10.1021/j150111a004. URL https://doi.org/10.1021/j150111a004.

D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002. ISBN 0521642981.

J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6): 1087–1092, 1953.

A. Noor, E. Serpedin, M. N. Nounou, and H. N. Nounou. Inferring gene regulatory networks via nonlinear state-space models and exploiting sparsity. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9:1203–1211, 2012.

J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular biology and evolution*, 16(12):1791–1798, 1999.

C. Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387212396.

S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Comput.*, 11(2):305–345, Feb. 1999. ISSN 0899-7667. doi: 10.1162/089976699300016674. URL http://dx.doi.org/10.1162/089976699300016674.

D. B. Rubin et al. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, 12(4):1151–1172, 1984.

T. Schön, A. Lindholm, L. Murray, and F. Lindsten. Probabilistic learning of nonlinear dynamical systems using sequential monte carlo. *Mechanical Systems and Signal Processing*, 03 2017. doi: 10.1016/j.ymssp.2017.10.033.

X. Sun, L. X. Jin, and M. Xiong. Extended kalman filter for estimation of parameters in nonlinear state-space models of biochemical networks. *PLoS ONE*, 3:1220 – 4, 2008.

T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. H. Stumpf. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society, Interface*, 6 31:187–202, 2009.

V. Volterra. Variations and Fluctuations of the Number of Individuals in Animal Species living together. *ICES Journal of Marine Science*, 3(1):3–51, 04 1928. ISSN 1054-3139. doi: 10.1093/icesjms/3.1.3. URL https://doi.org/10.1093/icesjms/3.1.3.

M. Wand and M. Jones. *Kernel Smoothing*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1994. ISBN 9780412552700. URL https://books.google.cz/books?id=GTOOi5yE008C.

Z. Wang, X. Liu, Y. Liu, J. Liang, and V. Vinciotti. An extended kalman filtering approach to modeling nonlinear dynamic gene regulatory networks via short gene expression time series. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 6:410–9, 07 2009. doi: 10.1109/TCBB.2009.5.

D. Wilkinson. *Stochastic Modelling for Systems Biology, Second Edition*. Chapman & Hall/CRC Mathematical and Computational Biology. Taylor & Francis, 2011. ISBN 9781439837726. URL https://books.google.cz/books?id=G3BaHtBrW68C.

R. Wilkinson. Approximate bayesian computation (abc) gives exact results under the assumption of model error. *Statistical applications in genetics and molecular biology*, 12: 1–13, 05 2013. doi: 10.1515/sagmb-2013-0010.

N. Zeng, Z. Wang, Y. Li, M. Du, and X. Liu. Inference of nonlinear state-space models for sandwich-type lateral flow immunoassay using extended kalman filtering. *IEEE Transactions on Biomedical Engineering*, 58:1959–1966, 2011.

Appendix A

Attached files

The attached files contain the source codes and input data necessary to run the experiments. The main scripts used to run the simulations come with a command line interface containing a help message listing the accepted arguments. The attachment structure is specified in Table A.1.

File	Description
`/data`	Serialized input data for the two experiments.
`auto_regulation.py`	Script to run the autoregulation experiment.
`auto_regulation_routines.pyx`	Additional routines writen in Cython.
`lotka_volterra.py`	Script to run the Lotka-Volterra experiment.
`lotka_volterra_routines.pyx`	Additional routines written in Cython.
`mcmc.py`	Implementation of the inference methods.
`utils.py`	Miscellaneous utility functions.

Table A.1: List of attached files.