Probabilistic learning of nonlinear dynamical systems using sequential Monte Carlo

Thomas B. Schön, Andreas Svensson, Lawrence Murray, Fredrik Lindsten

Department of Information Technology, Uppsala University, Sweden.

Abstract

Probabilistic modeling provides the capability to represent and manipulate *uncertainty* in data, models, predictions and decisions. We are concerned with the problem of learning probabilistic models of dynamical systems from measured data. Specifically, we consider learning of probabilistic nonlinear state-space models. There is no closed-form solution available for this problem, implying that we are forced to use approximations. In this tutorial we will provide a self-contained introduction to one of the state-of-the-art methods—the particle Metropolis–Hastings algorithm—which has proven to offer a practical approximation. This is a Monte Carlo based method, where the particle filter is used to guide a Markov chain Monte Carlo method through the parameter space. One of the key merits of the particle Metropolis–Hastings algorithm is that it is guaranteed to converge to the "true solution" under mild assumptions, despite being based on a particle filter with only a finite number of particles. We will also provide a motivating numerical example illustrating the method using a modeling language tailored for sequential Monte Carlo methods. The intention of modeling languages of this kind is to open up the power of sophisticated Monte Carlo methods—including particle Metropolis–Hastings—to a large group of users without requiring them to know all the underlying mathematical details.

Keywords: Probabilistic modelling, nonlinear dynamical systems, system identification, parameter estimation, Bayesian methods, Metropolis–Hastings, sequential Monte Carlo, particle filter.

Email addresses: thomas.schon@it.uu.se (Thomas B. Schön), andreas.svensson@it.uu.se (Andreas Svensson), lawrence.murray@it.uu.se (Lawrence Murray), fredrik.lindsten@it.uu.se (Fredrik Lindsten)

1. Introduction

The true value of measured data arises once it has been analyzed and some kind of knowledge has been extracted from this analysis. The analysis often relies on the combination of a mathematical model and the measured data. The model is a compact representation—set of assumptions—of some phenomenon of interest, and establishes a link between that phenomenon and the data, which is expected to provide some insight. The knowledge we seek is typically a function of some unknown variables or parameters in the model. However, any reasonable model will be *uncertain* when making statements about unobserved variables, and uncertainty therefore plays a fundamental role in modeling. Probabilistic modeling allows the representation and manipulation of uncertainty in data, models, decisions and predictions. The capability to mathematically represent and manipulate uncertainty—which is essential to the development throughout this tutorial—is provided by probability theory. A good introduction to the ideas underlying probabilistic modeling in contemporary machine learning is provided by [1] and from a system identification point of view we recommend [2].

Throughout this tutorial we are concerned with the problem of learning probabilistic models of nonlinear dynamical systems from measured data, which is sometimes referred to as the nonlinear system identification problem. These models provide an interpretable representation from which it is possible to extract the knowledge we seek, compared to doing so directly from the data. The model is thus a natural middle ground between the measured data and that knowledge. More specifically we are concerned with the nonlinear state-space model (SSM)

$$x_t = f(x_{t-1}, u_t, v_t, \theta), \tag{1a}$$
$$y_t = g(x_t, u_t, \theta) + e_t, \tag{1b}$$
$$x_0 \sim p(x_0 \,|\, \theta), \tag{1c}$$
$$\theta \sim p(\theta), \tag{1d}$$

where $y_t \in \mathsf{Y}$ denotes the observed output and u_t denotes a known input signal. In the interest of concise notation we will, without loss of generality, suppress the input signal u_t. The unknown variables are the state $x_t \in \mathsf{X}$ describing the system's evolution over time and the static parameters θ. Furthermore, $p(\theta)$ denotes the prior assumptions about θ. The stochastic variables v_t and e_t encode noise, commonly referred to as process noise and measurement noise, respectively. Finally, the functions f and g encode the dynamics and the measurement equation, respectively.

The first step towards extracting knowledge from a set of measured data $y_{1:T} = \{y_1, \ldots, y_T\}$ is to learn a probabilistic model of the form (1) by computing the conditional distribution of the unknown $x_{0:T}$ and θ conditioned on $y_{1:T}$, denoted $p(x_{0:T}, \theta \,|\, y_{1:T})$. This provides a useful representation which is typically much closer to the knowledge we seek than the measured data itself. Once we have a representation of this conditional distribution, it can be used to compute more specific quantities that typically constitute the end result of the analysis. To mention just a few examples of such quantities we have mean values, variances or estimates of some tail probability.

The key challenge is that, in general, there is no closed form expression available for $p(x_{0:T}, \theta \,|\, y_{1:T})$, so we must resort to approximations. We will focus on approximations based on Monte Carlo sampling which, while being costly to compute, have the appealing property of converging to the true solution as the amount of computations increases. Over the last decade, these approximations have evolved rapidly, so that we now have computationally feasible solutions available.

This naturally brings us to the aim of this tutorial, which is to provide a gentle introduction to probabilistic learning of nonlinear dynamical systems, and to introduce in some detail one of the current state-of-the-art methods to do so. This method relies on the systematic combination of two Monte Carlo algorithms, where a sequential Monte Carlo algorithm is used to compute a good proposal distribution for a Markov chain Monte Carlo (MCMC) algorithm. Hence, in terms of methods, this tutorial is focused on introducing one particular solution rather than surveying all available methods (see e.g. [3, 4] for recent accounts of that sort). However, the key ideas discussed in this tutorial (in the context of the specific method) are in fact central to many other state-of-the-art Monte Carlo learning methods as well. For example the particle filter itself, and the fact that it is capable of producing an unbiased estimate of the likelihood, are also of more general interest. In the accompanying paper [5], we show how probabilistic modeling, implemented via a new

algorithm of the type outlined in this tutorial, is used to solve one of the challenging benchmark problems in this special issue with promising results. We will also hint at the tailored software that is being developed to make these mathematical tools available to a much wider audience without a thorough knowledge of Monte Carlo methods. Such software allows the user to focus entirely on the modeling problem and leave the computational learning problem to the software.

In Section 2 we introduce probabilistic nonlinear state-space models in more detail. The model as it is stated in (1) clearly incorporates uncertainty due to the presence of the noise sources v_t and e_t, as well as uncertainty in the initial state x_0 and in the parameters θ. In probability theory, uncertainty is represented using random variables and in Section 2 the probabilistic nature of the model will be made even more explicit when we represent it as a joint distribution $p(x_{0:T}, \theta, y_{1:T})$ of all the random variables present in the model. The basic Monte Carlo idea is then introduced in Section 3 together with an explanation of how this idea can be used to learn the conditional distribution of the parameters given the measurements $p(\theta \mid y_{1:T})$. The idea is developed further in Section 4, where it becomes clear that we also need information about the unknown state variables, resulting in the introduction of the sequential Monte Carlo method (a.k.a. the particle filter) to estimate the state variables. In Section 5, the particle filter is used inside the MCMC method introduced in Section 3. The basic particle filter construction from Section 4 can be improved in several ways and in Section 6 we discuss some of the most important developments in this direction. The resulting method is then illustrated using a nonlinear spring-damper system in Section 7. Finally we conclude with a discussion in Section 8.

2. Probabilistic modelling of dynamical systems

We will refer to the joint distribution of all observed (here $y_{1:T}$) and unobserved (here $x_{0:T}$ and θ) variables as the *full probabilistic model*, which in our present setting amounts to $p(x_{0:T}, \theta, y_{1:T})$. The idea of using the mathematics of probability to represent and manipulate uncertainty is commonly referred to as Bayesian statistics [6]. In order to write down the full probabilistic model for (1) let us start by noticing that for many models we can express the conditional distribution of y_t given x_t and θ as

$$p(y_t \mid x_t, \theta) = p_{e_t}(y_t - g(x_t, \theta), \theta), \tag{2}$$

where $p_{e_t}(\cdot)$ denotes the distribution of the measurement noise e_t. Looking back at (1b) with the assumption that x_t and θ are both known we see that the only randomness stems from the measurement noise e_t. Hence, the conditional distribution of y_t given x_t and θ is dictated by $p_{e_t}(\cdot)$, which explains (2). In a similar way, the dynamical equation (1a) implicitly defines a conditional probability distribution $p(x_t \mid x_{t-1}, \theta)$ describing the state evolution.[1] Hence, the SSM in (1) can be expressed as

$$x_t \mid x_{t-1}, \theta \sim p(x_t \mid x_{t-1}, \theta), \tag{3a}$$
$$y_t \mid x_t, \theta \sim p(y_t \mid x_t, \theta), \tag{3b}$$
$$\theta \sim p(\theta) \tag{3c}$$

An important and useful aspect of the SSM is that the current state x_t contains all information about the past that is needed to make predictions into the future. Formally this aspect is captured by the *Markov property* stating that $p(x_{t+1} \mid x_{0:t}) = p(x_{t+1} \mid x_t)$. Using conditional probabilities we can factor the full probabilistic model as

$$p(x_{0:T}, \theta, y_{1:T}) = p(y_{1:T} \mid x_{0:T}, \theta) \underbrace{p(x_{0:T} \mid \theta) p(\theta)}_{\text{prior distribution}}, \tag{4}$$

where $p(y_{1:T} \mid x_{0:T}, \theta)$ describes the distribution of the data. The so-called *prior distribution* $p(x_{0:T}, \theta) = p(x_{0:T} \mid \theta) p(\theta)$ represents our initial assumptions about the unknown states and parameters. It is instructive

[1] The attentive reader might wonder why the noise is assumed to be additive in (1b) but not in (1a). The reason is that the methods that we will consider require the density function $p(y_t \mid x_t, \theta)$ to be available for point-wise evaluation. The additivity assumption implies that this density can be expressed as in (2). The methods do not, however, require $p(x_t \mid x_{t-1}, \theta)$ to be available for point-wise evaluation.

to rewrite the model (4) slightly in order to more clearly see that it is just a different way of representing (1). Let us first continue the use of conditional probabilities in order to further decompose $p(y_{1:T} \mid x_{0:T}, \theta)$ into

$$\begin{aligned} p(y_{1:T} \mid x_{0:T}, \theta) &= p(y_T \mid y_{1:T-1}, x_{0:T}, \theta) p(y_{1:T-1} \mid x_{0:T}, \theta) \\ &= p(y_T \mid x_T, \theta) p(y_{1:T-1} \mid x_{0:T-1}, \theta), \end{aligned} \tag{5}$$

where we also made use of the conditional independence of the observations given the current state, as well as the Markov property of the state. Moreover we can rewrite the prior distribution over the states $p(x_{0:T} \mid \theta)$ by noting that

$$p(x_{0:T} \mid \theta) = p(x_T \mid x_{0:T-1}, \theta) p(x_{0:T-1} \mid \theta) = p(x_T \mid x_{T-1}, \theta) p(x_{0:T-1} \mid \theta), \tag{6}$$

where we made use of conditional probabilities in the first equality and the Markov property in the second equality. Repeated use of (5) and (6) results in

$$p(x_{0:T}, \theta, y_{1:T}) = \left(\prod_{t=1}^{T} \underbrace{p(y_t \mid x_t, \theta)}_{\text{observation}} \right) \underbrace{\left(\prod_{t=1}^{T} \underbrace{p(x_t \mid x_{t-1}, \theta)}_{\text{dynamics}} \right) \underbrace{p(x_0 \mid \theta)}_{\text{initial}} \underbrace{p(\theta)}_{\text{param.}}}_{\text{prior}} \tag{7}$$

explicitly showing how the "engineering standard" SSM, as it is formulated in (1), relates to the full probabilistic model formulation.

Starting from our model (7) (or (1)), the problem we set out to solve is how to compute the distribution of the unobserved variables conditioned on the observed variables, $p(x_{0:T}, \theta \mid y_{1:T})$. This distribution is referred to as the *posterior* distribution. Using conditional probability it separates into the so-called state and parameter inference problems according to

$$p(x_{0:T}, \theta \mid y_{1:T}) = \underbrace{p(x_{0:T} \mid \theta, y_{1:T})}_{\text{state inf.}} \underbrace{p(\theta \mid y_{1:T})}_{\text{param. inf.}}. \tag{8}$$

In solving the parameter inference problem—which is the focus of this tutorial—it is helpful to start with

$$p(\theta \mid y_{1:T}) = \frac{p(y_{1:T} \mid \theta) p(\theta)}{p(y_{1:T})} = \frac{p(y_{1:T} \mid \theta) p(\theta)}{\int p(y_{1:T} \mid \theta) p(\theta) \mathrm{d}\theta}. \tag{9}$$

The target for the inference algorithms we eventually will derive will mainly be $p(\theta \mid y_{1:T})$, and for this reason we refer to it as the *target distribution* and denote it by $\pi(\cdot)$. The end user may not be interested in the target distribution per se, but rather in some test function $\varphi(\theta)$ evaluated over it. These can be evaluated by solving integrals of the form

$$I[\varphi] = \mathbb{E}\left[\varphi(\theta) \mid y_{1:T}\right] = \int \varphi(\theta) p(\theta \mid y_{1:T}) \mathrm{d}\theta. \tag{10}$$

A common test function is just the identity function $\varphi(\theta) = \theta$, so that the integral (10) provides an estimate of the expected value of θ conditioned on the data $y_{1:T}$, i.e. $\mathbb{E}\left[\theta \mid y_{1:T}\right]$. Another example is the indicator function $\varphi(\theta) = I(\theta > \vartheta)$, for some threshold value ϑ, which provides an estimate of a tail probability, perhaps important in modelling extreme events. Other test functions or combinations of them yield the covariance and higher moments, or estimates of domain-specific utility or loss.

From (9), it is clear that in order to compute the posterior distribution of the unknown parameters we first need to compute the *data distribution* $p(y_{1:T} \mid \theta)$. This can be expressed in terms of the joint distribution $p(x_{0:T}, y_{1:T} \mid \theta)$ by averaging it (marginalizing) over all possible values for the state variables $x_{0:T}$

$$p(y_{1:T} \mid \theta) = \int p(y_{1:T}, x_{0:T} \mid \theta) \mathrm{d}x_{0:T}. \tag{11}$$

Besides being useful to us later, this indicates the important fact that the state inference problem is inherent in the parameter inference problem when we are dealing with SSMs. Note that if we instead model the

unknown parameters θ as deterministic variables to be estimated using maximum likelihood, we still have to deal with the high-dimensional integral in (11), since this will be used in computing the so-called likelihood function. The likelihood function is defined as the data distribution evaluated for the particular measurement sequence $y_{1:T}$ that we have available, and it is thus not a probability distribution, but rather a deterministic function of the unknown variables θ. Hence, both Bayesian and maximum likelihood approaches will benefit from the methodology presented in this paper.

We will, however, reuse an important term from the maximum likelihood literature, and simply refer to the value of the data distribution $p(y_{1:T} \mid \theta)$ evaluated at $y_{1:T}$ and θ as *the likelihood*. Furthermore, we have used—and will continue to use—$p(\cdot)$ to denote a probability density function of variables that in some way describes the model or can be induced from the model (1). Distributions describing random variables that are not directly related to the model are instead denoted using characters from the Greek alphabet. These variables will for example arise as parts of the (stochastic) algorithms that we will derive in the coming sections.

3. Solving the parameter inference problem

The parameter inference problem amounts to computing the posterior distribution $p(\theta \mid y_{1:T})$, and commonly also to evaluate the integral (10) with respect to this distribution for some test function $\varphi(\theta)$. The posterior $p(\theta \mid y_{1:T})$ is in general[2] not available in closed form, and consequently there are typically no closed-form solutions to the integral (10) either. We can, however, construct an estimator of it using the Monte Carlo idea, which is introduced in Section 3.1. More specifically we will make use of a so-called pseudo-marginal MCMC method— introduced in Section 3.3—which itself employs an estimate. However, before we describe the pseudo-marginal idea we will first introduce the basic MCMC idea itself in Section 3.2.

3.1. The Monte Carlo idea

Monte Carlo integration provides approximate solutions to integrals of the form

$$c = \mathbb{E}\left[h(\xi)\right] \triangleq \int_{\Xi} h(\xi) p_{\xi}(\xi) \mathrm{d}\xi, \tag{12}$$

where $\xi \in \Xi$ and $\xi \sim p_{\xi}(\xi)$. A classical choice to evaluate the integral is to take N equally-spaced grid points $\{\xi^n\}_{n=1}^N$ on some interval $[a, b]$, and compute (for the scalar case)

$$\widehat{c} = \frac{b-a}{N} \sum_{n=1}^{N} h(\xi^n) p(\xi^n) \tag{13}$$

as an estimate of c. This is the usual Riemann sum. In Monte Carlo integration, the idea is to instead choose N *random* sample points $\{\xi^n\}_{n=1}^N$ from the probability distribution $p_{\xi}(\xi)$ and compute

$$\widehat{c} = \frac{1}{N} \sum_{n=1}^{N} h(\xi^n). \tag{14}$$

Clearly $\widehat{c}$ is now itself random. There are two advantages to this approach. The first is that, while a fixed grid can give an arbitrarily bad estimate, it can be proven that the expectation of the random estimator $\widehat{c}$ from the Monte Carlo integration (14) is exactly c, i.e., $\mathbb{E}\left[\widehat{c}\right] = c$, and we say that $\widehat{c}$ is an *unbiased* estimator of c. Secondly, if the dimension of Ξ is D, the error of the Riemann sum (13) scales as $\mathcal{O}(N^{-1/D})$, while that of the Monte Carlo estimate (14) scales as $\mathcal{O}(N^{-1/2})$ [see e.g. 8, 9]. That is, for more than a few dimensions, the Monte Carlo estimator exhibits smaller error.

[2]For certain special cases such as the autoregressive model with only Gaussian noise, closed-form solutions exist. This corresponds to Bayesian linear regression [6, 7].

3.2. The Markov chain Monte Carlo idea

A Markov chain $\{\theta[m]\}_{m\geq 0}$ is a stochastic process with the property that the next state $\theta[m+1]$ of the process depends on the current state $\theta[m]$, but, given this, it is conditionally independent of all previous states $\theta[0:m-1]$, recall the Markov property introduced in Section 2. It is completely specified by an initial distribution and a transition kernel which defines the (stochastic) transition from $\theta[m]$ to $\theta[m+1]$. The SSM (1) is a popular and useful example of a Markov chain $\{x_t\}_{t\geq 0}$ with initial distribution $p(x_0)$ and transition kernel given by $p(x_t \,|\, x_{t-1})$. A Markov chain is said to have an *equilibrium distribution* if the marginal distribution of the samples generated by the chain, $\{\theta[m]\}_{m=0}^{M}$, converges to this equilibrium distribution asymptotically as $M \to \infty$. That is, for large m, the samples $\theta[m]$ will be distributed roughly to the equilibrium distribution regardless of the distribution of $\theta[0]$. The idea underlying MCMC is to simulate a Markov chain which is designed in such a way that its equilibrium distribution coincides with the target distribution, here taken to be $\pi(\theta) = p(\theta \,|\, y_{1:T})$. Our interest lies in Markov chains which are structured such that as $M \to \infty$ we have that

$$\frac{1}{M}\sum_{m=0}^{M} \varphi(\theta[m]) \xrightarrow{\text{a.s.}} \int \varphi(\theta) p(\theta \,|\, y_{1:T}) \mathrm{d}\theta, \tag{15}$$

where $\xrightarrow{\text{a.s.}}$ denotes almost sure convergence. The finite and explicit sum in (15) provides an approximation of the intractable integrals *if* there is a way of constructing the underlying Markov chain $\{\theta[m]\}_{m\geq 0}$. The construction of such a Markov chain is indeed possible and one way is to use the following two-step procedure. In the first step a new *candidate* state θ' is proposed by generating a sample from the so-called *proposal distribution* $q(\theta \,|\, \theta[m])$, i.e. $\theta' \sim q(\theta \,|\, \theta[m])$. The proposal distribution depends on the current state $\theta[m]$ and its form is decided by the user. A common choice is a so-called Gaussian random walk, where a candidate sample is obtained by simply adding a realization from a certain Gaussian random variable to the current state of the Markov chain.

In the second step we must choose whether the next state $\theta[m+1]$ of the Markov chain should be the candidate sample θ' just proposed, or a repeat of the current state $\theta[m]$. It can be shown that if we chose the candidate sample as the next state with probability

$$\alpha = \min\left(1, \frac{\pi(\theta')}{\pi(\theta[m])}\frac{q(\theta[m] \,|\, \theta')}{q(\theta' \,|\, \theta[m])}\right) = \min\left(1, \frac{p(y_{1:T} \,|\, \theta')p(\theta')}{p(y_{1:T} \,|\, \theta[m])p(\theta[m])}\frac{q(\theta[m] \,|\, \theta')}{q(\theta' \,|\, \theta[m])}\right), \tag{16}$$

the resulting Markov chain will have the target distribution as its equilibrium distribution. The second equality in (16) is due to (9). For natural reasons α in (16) is referred to as the *acceptance probability*; with probability α the new state in the Markov chain is chosen as $\theta[m+1] = \theta'$ (the candidate sample is accepted) and with probability $1-\alpha$ the candidate sample is rejected and the Markov chain remains in the same state as in the previous iteration, i.e. $\theta[m+1] = \theta[m]$. The samples $\{\theta[m]\}_{m=0}^{M}$ from the resulting Markov chain constitutes an empirical approximation

$$\widehat{\pi}(\theta) = \frac{1}{M}\sum_{m=0}^{M} \delta_{\theta[m]}(\theta) \tag{17}$$

of the posterior distribution. Here, $\delta_{\theta[m]}(\theta)$ denotes the (Dirac) point-mass distribution at $\theta = \theta[m]$. The algorithm of first proposing a candidate state and then accepting or rejecting this as the next state of the Markov chain is referred to as the *Metropolis–Hastings (MH) algorithm*, introduced by [10, 11]. By now there are many textbook style introductions available, see e.g. [8, 9].

The intractable integral (10) becomes a tractable finite sum simply by inserting (17) into (10), resulting in

$$\widehat{I}[\varphi] \triangleq \int \varphi(\theta)\widehat{\pi}(\theta)\mathrm{d}\theta = \frac{1}{M}\sum_{m=0}^{M}\int \varphi(\theta)\delta_{\theta[m]}(\theta)\mathrm{d}\theta = \frac{1}{M}\sum_{m=0}^{M}\varphi(\theta[m]), \tag{18}$$

where the last equality follows from the properties of the Dirac point-mass distribution. The estimator $\widehat{I}[\varphi]$ defined above is well-behaved in the sense that under certain non-trivial conditions it obeys both a law of

large numbers and a central limit theorem, see e.g. [8, 12] for a thorough treatment. The law of large numbers tells us that the estimator is consistent as $M \to \infty$, whereas the central limit theorem tells us that the error is approximately Gaussian with a standard deviation that decreases with the typical Monte Carlo rate of $1/\sqrt{M}$. A nice historical account of MCMC is provided in [13].

3.3. Using unbiased estimates within Metropolis–Hastings

The problem preventing us from implementing the Metropolis–Hastings algorithm to infer $p(\theta \,|\, y_{1:T})$ for a general state-space model is that we cannot compute the acceptance probability α given in (16), since there is no closed-form expression available for the likelihood $p(y_{1:T} \,|\, \theta)$. However, what if we have an estimate of the likelihood, can we then use this estimate instead of the exact likelihood in the acceptance probability and still end up with a valid algorithm? ("Valid", here, meaning that the method converges in the sense of (15).) The answer to this highly nontrivial question is actually *yes*—under certain conditions on the likelihood estimate (described below). The result is referred to as an *exact approximation*, since we obtain an *exact* Metropolis–Hastings algorithm, in the sense that the equilibrium distribution of the Markov chain remains the target distribution of interest, despite the fact that we make use of an approximation of the likelihood when evaluating the acceptance probability.

Let us now explain and prove that this actually works. We start by assuming that we have access to an estimator $\widehat{z}$ of the likelihood $p(y_{1:T} \,|\, \theta)$. The estimator naturally depends on the observed data $y_{1:T}$ and the model parameter θ (since these variables determine the value of the estimand). Furthermore, we assume that the estimator depends on some additional random variables—as we shall see below, these random variables typically come from yet another Monte Carlo procedure which is used to compute the estimate. Consequently, the estimator $\widehat{z}$ is itself a *random variable* and it has some distribution $\psi(z \,|\, \theta, y_{1:T})$ depending on θ and $y_{1:T}$. Furthermore, we will not require ψ to be available on closed form (as we will see its density will actually cancel in the acceptance rate, so we do not need to evaluate it), but it exists conceptually nevertheless.

Next, we introduce the random variable $\widehat{z}$ (i.e., the estimator of the likelihood) as an *auxiliary variable* in the model. Auxiliary variables—while being of no particular interest on their own—are commonly introduced in statistical models in order to simplify the inference for some other variable of interest. In our case, the variable of interest is the model parameter θ. Thus, we consider the joint distribution for (θ, z) given by

$$\psi(\theta, z \,|\, y_{1:T}) = p(\theta \,|\, y_{1:T})\psi(z \,|\, \theta, y_{1:T}) = \frac{p(y_{1:T} \,|\, \theta)p(\theta)\psi(z \,|\, \theta, y_{1:T})}{p(y_{1:T})}. \tag{19}$$

We note that the original target distribution $p(\theta \,|\, y_{1:T})$ is (by construction) obtained by marginalizing the joint distribution $\psi(\theta, z \,|\, y_{1:T})$ with respect to the auxiliary variable $\widehat{z}$.

If we now were to construct a Metropolis–Hastings algorithm for both the parameters θ and the auxiliary variable $\widehat{z}$ it would suffer from the same problem as before, since the (intractable) likelihood still appears in the expression (19). To overcome this difficulty we will make use of the following trick: we *define* a new joint target distribution over $(\theta, \widehat{z})$ by simply replacing the intractable likelihood $p(y_{1:T} \,|\, \theta)$ in (19) with its estimator $\widehat{z}$

$$\pi(\theta, z \,|\, y_{1:T}) := \frac{z\, p(\theta)\psi(z \,|\, \theta, y_{1:T})}{p(y_{1:T})}. \tag{20}$$

For this distribution to be useful for our purposes we have three requirements that need to be fulfilled. The first two are formal in nature (stating that $\pi(\theta, z \,|\, y_{1:T})$ is a valid probability distribution): *(i)* that $\pi(\theta, z \,|\, y_{1:T})$ has to be everywhere non-negative, and *(ii)* that it integrates to 1. The third requirement stems from the auxiliary variable construction, stating *(iii)* that $\pi(\theta, z \,|\, y_{1:T})$ has to preserve the property that its marginal distribution for θ coincides with the target distribution of interest: $\int \pi(\theta, z \,|\, y_{1:T}) dz = p(\theta \,|\, y_{1:T})$. As we shall see next, all of these requirements are fulfilled if $\widehat{z}$ is a *non-negative* and *unbiased* estimate of $p(y_{1:T} \,|\, \theta)$.

The first requirement—non-negativity of π—follows directly from the assumed non-negativity of $\widehat{z}$. For the second and third requirements, we consider the integral

$$\int \pi(\theta, z \,|\, y_{1:T}) \mathrm{d}z = \frac{p(\theta)}{p(y_{1:T})} \int z\, \psi(z \,|\, \theta, y_{1:t}) \mathrm{d}z. \tag{21}$$

However, since $\psi(z \mid \theta, y_{1:T})$ is nothing but the distribution of the estimator $\widehat{z}$, the integral in the expression above is just the mean of $\widehat{z}$. Hence, due to the assumed unbiasedness of $\widehat{z}$

$$\int \pi(\theta, z \mid y_{1:T}) \mathrm{d}z = \frac{p(\theta)}{p(y_{1:T})} p(y_{1:T} \mid \theta) = p(\theta \mid y_{1:T}). \tag{22}$$

Thus, we see that the marginal distribution of π with respect to θ is $p(\theta \mid y_{1:T})$ as required, which also implies that π is a properly normalized probability distribution (it integrates to 1, which can be seen by further integrating both sides of (22) with respect to θ).

We will now construct a standard Metropolis–Hastings algorithm with $\pi(\theta, z \mid y_{1:T})$ as its target distribution. Note that we will work on the joint space of θ and $\widehat{z}$, and the joint proposal will be a two-step procedure which first samples θ' from $q(\theta \mid \theta[m])$, and thereafter samples $\widehat{z}'$ from $\psi(z \mid \theta', y_{1:T})$. More compactly, we can write this as $(\theta', \widehat{z}') \sim q(\theta' \mid \theta[m])\psi(z' \mid \theta', y_{1:T})$. Inserting all these expressions into the acceptance rate (16), we obtain

$$\begin{aligned}
\alpha &= \min\left(1, \frac{\pi(\theta', z' \mid y_{1:T})}{\pi(\theta[m], z[m] \mid y_{1:T})} \frac{q(\theta[m] \mid \theta')\psi(z[m] \mid \theta[m], y_{1:T})}{q(\theta' \mid \theta[m])\psi(z' \mid \theta', y_{1:T})}\right) \\
&= \min\left(1, \frac{z' p(\theta')\psi(z' \mid \theta', y_{1:T})}{z[m]p(\theta[m])\psi(z[m] \mid \theta[m], y_{1:T})} \frac{q(\theta[m] \mid \theta')\psi(z[m] \mid \theta[m], y_{1:T})}{q(\theta' \mid \theta[m])\psi(z' \mid \theta', y_{1:T})}\right) \\
&= \min\left(1, \frac{z' p(\theta')}{z[m]p(\theta[m])} \frac{q(\theta[m] \mid \theta')}{q(\theta' \mid \theta[m])}\right).
\end{aligned} \tag{23}$$

The development above is summarize in Algorithm 1.

Algorithm 1 pseudo-marginal Metropolis–Hastings

1: **Initialisation** ($m = 0$)**:** Set $\theta[0]$ arbitrarily and sample $\widehat{z}[0] \sim \psi(z \mid \theta[0], y_{1:T})$.
2: **for** $m = 1$ **to** M **do**
3: Sample $\theta' \sim q(\theta \mid \theta[m-1])$.
4: Sample $\widehat{z}' \sim \psi(z \mid \theta', y_{1:T})$.
5: With probability

$$\alpha = \min\left(1, \frac{\widehat{z}' p(\theta')}{\widehat{z}[m-1]p(\theta[m-1])} \frac{q(\theta[m-1] \mid \theta')}{q(\theta' \mid \theta[m-1])}\right), \tag{24}$$

set $\{\theta[m], \widehat{z}[m]\} \leftarrow \{\theta', \widehat{z}'\}$ (accept the candidate samples) and with probability $1 - \alpha$ set $\{\theta[m], \widehat{z}[m]\} \leftarrow \{\theta[m-1], \widehat{z}[m-1]\}$ (reject the candidate samples).
6: **end for**

This idea of making use of a non-negative and unbiased estimate of the likelihood within a Metropolis–Hastings algorithm is referred to as the *pseudo-marginal approach*. It was first introduced in 2003 [14] and it has later been generalized and thoroughly analysed, see [15, 16]. Note that it is only required that we can simulate from $\psi(z \mid \theta, y_{1:T})$, we never have to evaluate it point-wise.

The pseudo-marginal algorithm will converge towards the correct posterior distribution $p(\theta \mid y_{1:T})$ under weak assumptions, as long as the nonnegativity and unbiasedness conditions hold for the estimate $\widehat{z}$ (convergence, here, means that the distribution of $\theta[m]$ converges to $p(\theta \mid y_{1:T})$ as $m \to \infty$). This is not to say that the precision of the estimate $\widehat{z}$ is of no interest, however. Indeed, if the variance in the estimate $\widehat{z}$ is overly large, this will result in the algorithm converging slowly. In practice it is therefore important to keep the variance in the estimate $\widehat{z}$ as low as possible to obtain an efficient solution. We return to this in Section 5.

4. Computing an unbiased estimate of the likelihood using the particle filter

In order to use Algorithm 1 for identification of a state-space model, we need a way to construct an *unbiased* and *non-negative* estimate of the likelihood $p(y_{1:T} \mid \theta)$ for any value of θ. Specifically, as the algorithm considers a candidate value θ' for the model parameters, we need to compute an estimate $\widehat{z} \approx p(y_{1:T} \mid \theta')$ that is, intuitively speaking, used to determine whether or not θ' is a promising candidate.

The likelihood is only a function of the data $y_{1:T}$ and the parameters θ, and not the state variables $x_{0:T}$. Thus, to compute $p(y_{1:T} \mid \theta)$ we need to marginalize out all possible trajectories $x_{0:T}$. We can write this as

$$p(y_{1:T} \mid \theta) = \int_{\mathsf{X}^{T+1}} p(y_{1:T} \mid x_{0:T}, \theta) p(x_{0:T} \mid \theta) \mathrm{d}x_{0:T}. \tag{25}$$

In general, this integral admits no closed-form solution[3]. We can, however, construct an unbiased and non-negative estimator of it using the Monte Carlo idea. We will in Section 4.1 detail a naive construction of such an estimator. That construction serves as a motivation for the particle filter, which is then introduced in Section 4.2. It provides a practically useful computation of a non-negative and unbiased estimator of the likelihood for a nonlinear state-space model (1).

4.1. Monte Carlo integration for the likelihood

The problem of solving (25) is a special case of (12) where $\xi = x_{0:T}$, $p_\xi(\xi) = p(x_{0:T} \mid \theta)$ and $h(\xi) = p(y_{1:T} \mid x_{0:T}, \theta)$. A first attempt to solve this problem is to use the basic Monte Carlo approach as presented in Section 3.1 and proceed as follows:

1. Generate N samples of the state trajectory $x_{0:T}^n \sim p(x_{0:T} \mid \theta)$ for $n = 1, \ldots, N$. In practice, this can be done by simulating the system dynamics: $x_0^n \sim p(x_0 \mid \theta)$, then $x_1^n \sim p(x_1 \mid x_0^n, \theta)$, then $x_2^n \sim p(x_2 \mid x_1^n, \theta)$, etc. up to T, for $n = 1, \ldots, N$.
2. Compute an estimate of the likelihood as $\widehat{z} := \frac{1}{N} \sum_{n=1}^N p(y_{1:T} \mid x_{0:T}^n, \theta) = \frac{1}{N} \sum_{n=1}^N \prod_{t=1}^T p(y_t \mid x_t^n, \theta)$.

This results in a basic Monte Carlo estimate $\widehat{z}$ which is indeed unbiased and non-negative so it is a valid approach. In practice, however, the variance of this $\widehat{z}$ is too large to be of any practical use, as we will illustrate in Figure 1. The reason is that many samples $\{x_{0:T}^n\}_{n=1}^N$ are likely to be drawn in a place where they contribute very little to the estimate due to the high dimension of the space $\mathsf{X}^{(T+1)}$.

To mitigate this problem the structure of the state-space model can be leveraged. Specifically, we factorise the high-dimensional integral into a product of T lower-dimensional integrals:

$$p(y_{1:T} \mid \theta) = \prod_{t=1}^T p(y_t \mid y_{1:t-1}, \theta) = \prod_{t=1}^T \int_{\mathsf{X}} p(y_t \mid x_t, \theta) p(x_t \mid y_{1:t-1}, \theta) \mathrm{d}x_t. \tag{26}$$

We can then perform Monte Carlo integration for each integral, one at a time, and take the product of the estimates to obtain an overall estimate. The advantage here is that it is typically much easier to handle these lower-dimensional integrals than it is to handle the preceding higher-dimensional integral. The drawback, however, is that it is typically not possible to simulate directly from $p(x_t \mid y_{1:t-1}, \theta)$. To tackle this issue we will instead make use of an algorithm known as the *particle filter*, which we introduce next.

4.2. Introducing the particle filter

The particle filter sequentially generates samples $\{x_t^n\}_{n=1}^N$ for $t = 1, 2, \ldots$, such that the samples with index t are approximately distributed according to $p(x_t \mid y_{1:t-1}, \theta)$. What differs from the basic Monte Carlo procedure outlined above (where each $x_{0:T}^n$ was drawn independently) is that all the samples are allowed to interact between iterations. Note that the particle filter is most often derived as a way of approximating the filter distribution $p(x_t \mid y_{1:t}, \theta)$. However, we can equally-well derive it as an algorithm for approximation of the predictor distribution $p(x_t \mid y_{1:t-1}, \theta)$, since that is what we need to estimate the likelihood. To derive the particle filter it is convenient to introduce the notion of an *empirical approximation* of a probability distribution. An empirical distribution with N samples $\{x_t^n\}_{n=1}^N$ approximating the distribution $p(x_t \mid y_{1:t-1}, \theta)$ looks as

$$\widehat{p}(x_t \mid y_{1:t-1}, \theta) = \frac{1}{N} \sum_{n=1}^N \delta_{x_t^n}(x_t), \tag{27}$$

[3]For certain (indeed important) special cases such as linear models with only Gaussian noise, closed-form solutions exists, obtained via the Kalman filter. Complete details on that special case—including how to use Kalman filters to implement a Metropolis–Hastings algorithm—is provided in [3].

where $\delta_{x_t^n}(x_t)$ is again a (Dirac) point-mass distribution at x_t^n. Note that if we plug this empirical distribution into an integral with respect to $p(x_t \,|\, y_{1:t-1}, \theta)$ we recover the normal Monte Carlo estimator of that integral. For instance, for the integral in (26) we get:

$$\int_{\mathsf{X}} p(y_t \,|\, x_t, \theta) p(x_t \,|\, y_{1:t-1}, \theta) \mathrm{d}x_t \approx \int_{\mathsf{X}} p(y_t \,|\, x_t, \theta) \left[\frac{1}{N} \sum_{n=1}^{N} \delta_{x_t^n}(x_t) \right] \mathrm{d}x_t = \frac{1}{N} \sum_{n=1}^{N} p(y_t \,|\, x_t^n, \theta). \tag{28}$$

Let us now consider how the samples—or particles as they are often called—$\{x_t^n\}_{n=1}^N$ can be generated sequentially. When $t = 1$ we have, by using the convention $y_{1:0} = \emptyset$, that $p(x_1 \,|\, y_{1:0}, \theta) = p(x_1 \,|\, \theta)$ and we can thus sample directly from the initial distribution $x_0^n \sim p(x_0 \,|\, \theta)$ and simulate the system dynamics to obtain $x_1^n \sim p(x_1 \,|\, x_0^n)$ for $n = 1, \ldots, N$. For any consecutive time step it holds that each target distribution $p(x_{t+1} \,|\, y_{1:t}, \theta)$ can be constructed from the previous, $p(x_t \,|\, y_{1:t-1}, \theta)$, in the following way. First, by Bayes' theorem it follows that

$$p(x_t \,|\, y_{1:t}, \theta) \propto p(y_t \,|\, x_t, \theta) p(x_t \,|\, y_{1:t-1}, \theta). \tag{29}$$

Next, by incorporating the state at time $t+1$ (according to the dynamical model) and marginalising over x_t we have

$$p(x_{t+1} \,|\, y_{1:t}, \theta) = \int p(x_{t+1}, x_t \,|\, y_{1:t}, \theta) \mathrm{d}x_t = \int p(x_{t+1} \,|\, x_t, \theta) p(x_t \,|\, y_{1:t}, \theta) \mathrm{d}x_t. \tag{30}$$

The two equations above are often referred to as the measurement update and time update, respectively, as the former takes the t^{th} measurement y_t into account and the latter propagates the distribution forward in time according to the system dynamics (cf. the two steps of the Kalman filter).

The particle filter makes use of Equations (29) and (30) to sample approximately from $p(x_{t+1} \,|\, y_{1:t}, \theta)$ based on the existing samples from $p(x_t \,|\, y_{1:t-1}, \theta)$. First, we plug the empirical approximation (27) into (29). This gives rise to an empirical approximation of the distribution $p(x_t \,|\, y_{1:t}, \theta)$, but where each particle is assigned a *weight* w_t^n according to the multiplicative factor $w_t^n = p(y_t \,|\, x_t^n, \theta)$. The unknown normalising constant in (29) (hidden in the proportionality sign) is not needed since we can simply normalise the weights of the empirical distribution, which becomes

$$\widehat{p}(x_t \,|\, y_{1:t}, \theta) = \sum_{n=1}^{N} \frac{w_t^n}{\sum_{j=1}^{N} w_t^j} \delta_{x_t^n}(x_t). \tag{31}$$

Next, we note that by (30) it is *conceptually* possible to sample from $p(x_{t+1} \,|\, y_{1:t}, \theta)$ by first sampling from $p(x_t \,|\, y_{1:t}, \theta)$ and then simulating the system dynamics forward by sampling from the transition density $p(x_{t+1} \,|\, x_t, \theta)$. To make this practical, the particle filter plugs the empirical approximation (31) into (30), resulting in a *weighted* empirical approximation. Thus, we simulate N particles from the empirical distribution (31). This simply amounts to sampling with replacement from among the existing particles $\{x_t^n\}_{n=1}^N$, with probabilities given by the normalized weights. Let the resulting particles be denoted by $\{\bar{x}_t^n\}_{n=1}^N$. Then we propagate these particles forward in time by sampling $x_{t+1}^n \sim p(x_{t+1}^n \,|\, \bar{x}_t^n, \theta)$ for $n = 1, \ldots, N$. This completes one time step of the particle filter, and the procedure described above can now be repeated for time $t+2$, $t+3$, etc.

The procedure of sampling from the empirical distribution (31) is commonly referred to as *resampling*. This is a key ingredient of the particle filter which, intuitively, exchanges the weight of each particle for a number of copies among the N chosen. Particles with relatively small weights tend to be discarded, while particles with relatively large weights tend to be replicated several times.

We summarize this by Algorithm 2, the *bootstrap particle filter*, which was introduced independently by [17, 18, 19].

From each empirical approximation $\widehat{p}(x_t \,|\, y_{1:t-1}, \theta)$, it is straightforward to perform Monte Carlo integration of (26) by substituting the empirical approximation $\widehat{p}(x_t \,|\, y_{1:t-1}, \theta)$ in place of each $p(x_t \,|\, y_{1:t-1}, \theta)$ as in (28). This results in an estimate of the likelihood given by

$$\widehat{z} := \prod_{t=1}^{T} \left[\frac{1}{N} \sum_{n=1}^{N} p(y_t \,|\, x_t^n, \theta) \right] = \prod_{t=1}^{T} \left[\frac{1}{N} \sum_{n=1}^{N} w_t^n \right]. \tag{32}$$

Algorithm 2 Bootstrap particle filter (all operations are for $n = 1, \ldots, N$)

1: **Initialization:**
2: Sample $x_0^n \sim p(x_0 \,|\, \theta)$ and propagate $x_1^n \sim p(x_1 \,|\, x_0^n, \theta)$.
3: Compute $w_1^n = p(y_1 \,|\, x_1^n, \theta)$.
4: **for** $t = 2$ **to** T **do**
5: **Resampling:** Sample a_t^n with $\mathbb{P}(a_t^n = j) \propto w_{t-1}^j$ and set $\bar{x}_{t-1}^n = x_{t-1}^{a_t^n}$.
6: **Propagation:** Sample $x_t^n \sim p(x_t \,|\, \bar{x}_{t-1}^n, \theta)$.
7: **Weighting:** Compute $w_t^n = p(y_t \,|\, x_t^n, \theta)$.
8: **end**

(Note that the weights w_t^n appearing in the above expression are not normalised.) This estimator is obviously non-negative, since each term w_t^n is non-negative. What is less obvious to see is that the estimator also satisfies the requirement of being unbiased, which means that it can be used within Algorithm 1 to obtain a valid pseudo-marginal Metropolis–Hastings method. For a proof of this claim we refer the interested reader to [20, 21].

We summarize this section by Figure 1, where we have implemented the approach proposed in the beginning of Section 4.1 ('vanilla Monte Carlo'), as well as the bootstrap particle filter, to illustrate the different properties of the likelihood estimate $\widehat{z}$ for both approaches. As can be clearly seen, the variance of $\widehat{z}$ obtained from the particle filter is much smaller than the variance obtained by the vanilla Monte Carlo approach. Nevertheless, both approaches provide an unbiased estimate. It should be noted that in more challenging situations the skewness of the distribution of $\widehat{z}$ can be much more extreme (both for the vanilla Monte Carlo approach and for the particle filter if using to few particles) in the sense that $\mathbb{P}(\widehat{z} < p(y_{1:T} \,|\, \theta)) \approx 1$ despite the fact that the estimate is unbiased.

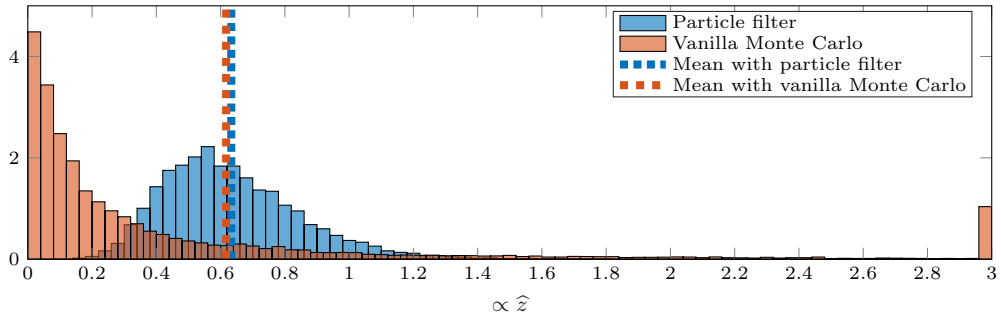

Figure 1: Estimation of the likelihood $p(y_{1:T} \,|\, \theta)$ in the spring-damper example (Section 7) for θ set to the true value. The histograms describe how 10 000 independent samples obtained by vanilla Monte Carlo integration (orange, as described in Section 4.1) and the particle filter (blue, as described in Section 4.2) are distributed, both using $N = 256$. Both approaches provide unbiased estimates (the means, dotted lines, are indeed essentially the same), but the particle filter has significantly smaller variance and less heavy tails than the vanilla Monte Carlo integration (the rightmost orange bin contains all samples ≥ 3; in fact the biggest orange sample obtained was as large as 100.) All estimates have been rescaled by a constant factor for clarity of presentation.

5. Using the particle filter estimate within pseudo-marginal MH

The non-negative and unbiased likelihood estimator described in Section 4.2 can now be used inside the pseudo-marginal Metropolis–Hastings algorithm from Section 3.3. The end result is a standard Metropolis–Hastings algorithm operating on the non-standard joint space of the model parameters θ and the auxiliary variable $\widehat{z}$. The target distribution for our new algorithm is given by $\pi(\theta, z \,|\, y_{1:T})$, which was defined in (20). The result is referred to as *particle Metropolis–Hastings (PMH)*—summarized in Algorithm 3. It was introduced by Andrieu et al. in 2010 [22] (under the name particle *marginal* Metropolis–Hastings).

The PMH algorithm will thus produce a Markov chain $\{\theta[m], \widehat{z}[m]\}_{m=0}^M$ on the joint space of the parameters θ and the auxiliary variable $\widehat{z}$. Since the posterior distribution $p(\theta \,|\, y_{1:T})$ is obtained as a marginal of $\pi(\theta, z \,|\, y_{1:T})$ we can—by construction—obtain samples from $p(\theta \,|\, y_{1:T})$ by extracting the sub-chain $\{\theta[m]\}_{m=0}^M$

Algorithm 3 Particle Metropolis–Hastings (PMH)

Initialisation ($m = 0$)**:** Set $\theta[0]$ arbitrarily and run Algorithm 2 to obtain $\widehat{z}[0]$.
for $m = 1$ **to** M **do**
  Sample $\theta' \sim q(\theta \,|\, \theta[m-1])$.
  Sample $\widehat{z}' \sim \psi(z \,|\, \theta', y_{1:T})$ by running Algorithm 2 once and compute (32).
  Compute the acceptance probability

$$\alpha_m = \min\left(1, \frac{\widehat{z}' p(\theta')}{\widehat{z}[m-1] p(\theta[m-1])} \frac{q(\theta[m-1] \,|\, \theta')}{q(\theta' \,|\, \theta[m-1])}\right), \tag{33}$$

  Sample ω_m uniformly over $[0, 1]$.
  if $\omega_m < \alpha_m$ **then**
    Set $\{\theta[m], \widehat{z}[m]\} \leftarrow \{\theta', \widehat{z}'\}$ (accept the candidate samples).
  else
    Set $\{\theta[m], \widehat{z}[m]\} \leftarrow \{\theta[m-1], \widehat{z}[m-1]\}$ (reject the candidate samples).
  end
end

from $\{\theta[m], \widehat{z}[m]\}_{m=0}^{M}$. Hence, the samples of the auxiliary variable are simply ignored. It is worth noting that the samples of the states $x_{0:T}$ produced by the particle filters that are executed as parts of Algorithm 3 provide a competitive solution to the problem of estimating the joint smoothing distribution $p(x_{0:T} \,|\, y_{1:T})$.

As noted in Section 3, the variance of the estimate $\widehat{z}$ will affect the convergence speed of the algorithm. Specifically, if the variance is very large, then the method tends to get "stuck" for many consecutive iterations, not accepting any proposed moves. The reason for this is that large variance in the estimate is often related to a large skewness as well, in the sense that with high probability $\widehat{z} < p(y_{1:T} \,|\, \theta)$. However, since the estimate is unbiased, this implies that $\widehat{z}$ sometimes (but rarely) must take on values $\widehat{z} \gg p(y_{1:T} \,|\, \theta)$. This is illustrated by Figure 1 where the vanilla Monte Carlo estimate is heavily skewed. This is a problem when $\widehat{z}$ is used in Algorithm 3 since if most of the probability mass is concentrated on $\widehat{z} \ll p(y_{1:T} \,|\, \theta)$, the acceptance probability α_m will tend to be small and the sampler can get stuck for many iterations. For the method to work satisfactorily we therefore need to ensure that the variance in $\widehat{z}$ is not overly large, which in turn implies that we need to use a sufficiently large number N of particles in the underlying particle filter. What "sufficiently" means is problem dependent, but as a rule-of-thumb N should scale linearly with T.

A possible improvement of Algorithm 3 is offered by noting that the particle filter output can also be used to compute estimates of the likelihood gradient and Hessian. These estimates can be used to construct better proposal distributions q for θ, with the potential to explore the parameter space similarly to the way in which Newton's optimization algorithm is exploring the parameter space. This can lead to significant improvements compared to the standard random walk proposal that is commonly employed, see [23]. The classic variance reduction technique of positively correlating two estimators has been applied also to the particle Metropolis–Hastings algorithm by introducing positive correlations between subsequent likelihood estimators [24].

6. Variance reduction methods for the particle filter

As illustrated by Figure 1, the bootstrap particle filter results in a substantial improvement in the likelihood estimate over the vanilla Monte Carlo approach. Since both approaches result in unbiased estimates, this improvement essentially stems from a reduction in variance. Various improvements on the bootstrap particle filter can similarly be used to further reduce the variance of the estimate of the likelihood (while maintaining unbiasedness). Indeed, many such improvements have been proposed in the literature over the past quarter century. For the interested reader we review a few key ideas below. Additional improvements and details can be found in e.g. [25, 26, 27].

6.1. Low-variance resampling

Firstly, we note that the resampling step of the particle filter—while being of key importance to its stability—is a random procedure. As such it inevitably introduces some additional variance in the estimate of the likelihood. This additional variance can be reduced by using standard variance reduction techniques,

e.g. stratification, when sampling from the empirical distribution (31). This still results in a valid particle filter, as long as the resampling method used is itself unbiased. More precisely, when sampling from the empirical distribution (31) we require that the expected number of copies of each particle is proportional to its weight,

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N} I(\bar{x}_t^n = x_t^i)\right] = \frac{w_t^i}{\sum_{n=1}^{N} w_t^n}, \qquad i = 1, \ldots, N, \tag{34}$$

where $I(\cdot)$ is an indicator function. Several low-variance resampling methods satisfying this unbiasedness condition are reviewed in [28, 29].

Low-variance resampling can (and should) always be used when implementing the particle filter. Other variance reduction techniques, however, are more model-specific and can be used only for certain classes of state-space models. We describe two such possible improvements below.

6.2. Conditioning on the current measurement

The first such improvement is a technique which aims to take the current measurement y_t into account when simulating the particles $\{x_t^n\}_{n=1}^N$ at time t. The intuition behind this is that the measurement y_t often contains valuable information about the state of the system at time t, which can help to simulate these particles in a "good" region of the state space. To this end we assume that the distribution for the system dynamics *conditional* on the current observation—i.e.

$$p(x_t \,|\, x_{t-1}, y_t, \theta) = \frac{p(y_t \,|\, x_t, \theta) p(x_t \,|\, x_{t-1}, \theta)}{p(y_t \,|\, x_{t-1}, \theta)}, \tag{35}$$

—is available in closed form. This is not always the case, but for some highly relevant state-space models it is indeed available. One important example is when the state dynamics $p(x_t \,|\, x_{t-1}, \theta)$ are Gaussian (but with a possibly nonlinear dependence on x_{t-1}) and the measurement equation is linear and Gaussian, $p(y_t \,|\, x_t, \theta) = \mathcal{N}(y_t \,|\, Cx_t, R)$, for the (non state dependent) matrices C and R. We shall further assume that the normalization factor $p(y_t \,|\, x_{t-1}, \theta)$ in (35) can be evaluated point-wise, but this typically follows from the aforementioned assumption.

The effect of simulating particles x_t^n conditionally on y_t is that we obtain an (unweighted) empirical approximation of the *filtering distribution* instead of the one-step predictive distribution as in (28). That is, we sequentially obtain the empirical approximations

$$\widehat{p}(x_t \,|\, y_{1:t}, \theta) = \frac{1}{N}\sum_{n=1}^{N} \delta_{x_t^n}(x_t), \tag{36}$$

for $t = 0, \ldots, T$. Note the conditioning on y_t on the left-hand side in (36).

Our main object of interest—the likelihood—can be expressed in terms of these filtering distributions using a similar factorisation as in (26), but where we choose to marginalise over x_{t-1} instead of x_t,

$$p(y_{1:T} \,|\, \theta) = \prod_{t=1}^{T} \int_{\mathsf{X}} p(y_t \,|\, x_{t-1}, \theta) p(x_{t-1} \,|\, y_{1:t-1}, \theta) \mathrm{d}x_{t-1}. \tag{37}$$

To see how we can sequentially obtain the empirical distributions (36) we start by combining the measurement and time update equations, (29) and (30), with the expression (35),

$$\begin{aligned} p(x_t \,|\, y_{1:t}, \theta) &\propto \int p(y_t \,|\, x_t, \theta) p(x_t \,|\, x_{t-1}, \theta) p(x_{t-1} \,|\, y_{1:t-1}, \theta) \mathrm{d}x_{t-1} \\ &= \int p(x_t \,|\, x_{t-1}, y_t, \theta) p(y_t \,|\, x_{t-1}, \theta) p(x_{t-1} \,|\, y_{1:t-1}, \theta) \mathrm{d}x_{t-1}. \end{aligned} \tag{38}$$

Proceeding in a similar way as for the bootstrap particle filter presented above, we plug in the "current" empirical approximation of the filtering distribution (i.e., from time $t-1$) into the above integral. In doing so, the factor $p(y_t \,|\, x_{t-1}, \theta)$ enters as a weight on the particles at time $t-1$. Thus, to sample a new set of

particles $\{x_t^n\}_{n=1}^N$ approximately from the filtering distribution at time t, we *(i)* resample the particles at time $t-1$ with probabilities proportional to $\nu_t^n := p(y_t \mid x_{t-1}^n, \theta)$, yielding $\{\bar{x}_{t-1}^n\}_{n=1}^N$ and *(ii)* propagate these particles to time t by sampling $x_t^n \sim p(x_t \mid \bar{x}_{t-1}^n, y_t, \theta)$ for $n = 1, \ldots, N$.

Finally, by (37), the estimate of the likelihood is given by

$$z := \prod_{t=1}^{T} \left[\frac{1}{N} \sum_{n=1}^{N} \nu_t^n \right]. \tag{39}$$

As for the bootstrap particle filter, it holds that this estimator is non-negative and unbiased. However, it is often the case in practice that (39) has much lower variance than its bootstrap particle filter counterpart (32).

The particle filter described above is often referred to as the *fully adapted auxiliary particle filter*. The term "adapted" here refers to the fact that we adapt the sampling distributions, both in the resampling step and when simulating new particles, to the information provided by the current measurement y_t. "Fully" refers to the fact that we do so by using the exact conditional distribution in (35). As mentioned above, however, this conditional distribution is only available in closed form for a restricted class of models. More generally it is possible to use an approximation $q(x_t \mid x_{t-1}, y_t, \theta) \approx p(x_t \mid x_{t-1}, y_t, \theta)$ for simulating new particles, and similarly an approximation $q(y_t \mid x_{t-1}, \theta) \approx p(y_t \mid x_{t-1}, \theta)$ when computing the resampling weights. It turns out that it is still possible to obtain a valid particle filter implementation, yielding unbiased estimates of the likelihood, where the approximations q are used as *proposal distributions* within an importance sampling framework. We refer to [30, 25] for details. A complete textbook-style introduction along the lines of this section is available in [31].

6.3. Rao-Blackwellization

One important (and rather common) class of state-space models for which another variance reduction technique is possible are the so-called *conditionally linear Gaussian (CLG)* state-space models. These models are characterised by having a substructure that can be identified as being linear and Gaussian. Therefore, this substructure is analytically tractable using Kalman filtering techniques, which can be leveraged when running a particle filter for the whole model. This is done in a way which resembles the method of *Rao-Blackwellization* of statistical estimators, and the resulting particle filters are therefore commonly referred to as Rao-Blackwellized particle filters (RBPFs).

To be more specific, let the state variable x_t be partitioned into two components $x_t = (x_t^{\mathrm{n}}, x_t^{\mathrm{l}})$ where we use the superscripts n and l for nonlinear and linear, respectively. Then, a CLG model can be defined as a state-space model where the *conditional stochastic process* $\{(x_t^{\mathrm{l}}, y_t) \mid x_{0:t}^{\mathrm{n}}\}_{t \geq 0}$ follows a (time-inhomogeneous) linear Gaussian state-space model. We can thus identify whether or not a specific model under study is CLG by "pretending" that the nonlinear state component $\{x_t^{\mathrm{n}}\}_{t \geq 0}$ is observed. If the model then can be viewed as a linear Gaussian state-space model, then the original model is CLG.

To give an example, consider the mixed linear/nonlinear model

$$x_{t+1}^{\mathrm{n}} = f^{\mathrm{n}}(x_t^{\mathrm{n}}) + A^{\mathrm{n}}(x_t^{\mathrm{n}}) x_t^{\mathrm{l}} + v_t^{\mathrm{n}}, \tag{40a}$$

$$x_{t+1}^{\mathrm{l}} = f^{\mathrm{l}}(x_t^{\mathrm{n}}) + A^{\mathrm{l}}(x_t^{\mathrm{n}}) x_t^{\mathrm{l}} + v_t^{\mathrm{l}}, \tag{40b}$$

$$y_t = g(x_t^{\mathrm{n}}) + C(x_t^{\mathrm{n}}) x_t^{\mathrm{l}} + e_t, \tag{40c}$$

where $x_t^{\mathrm{n}} \in \mathbb{R}^{d_{\mathrm{n}}}$, $x_t^{\mathrm{l}} \in \mathbb{R}^{d_{\mathrm{l}}}$, and $y_t \in \mathbb{R}^{d_y}$. The model is specified in terms of the (possibly nonlinear) functions $f^{\mathrm{n}} : \mathbb{R}^{d_{\mathrm{n}}} \mapsto \mathbb{R}^{d_{\mathrm{n}}}$, $A^{\mathrm{n}} : \mathbb{R}^{d_{\mathrm{n}}} \mapsto \mathbb{R}^{d_{\mathrm{n}} \times d_{\mathrm{l}}}$, $f^{\mathrm{l}} : \mathbb{R}^{d_{\mathrm{n}}} \mapsto \mathbb{R}^{d_{\mathrm{l}}}$, $A^{\mathrm{l}} : \mathbb{R}^{d_{\mathrm{n}}} \mapsto \mathbb{R}^{d_{\mathrm{l}} \times d_{\mathrm{l}}}$, $g : \mathbb{R}^{d_{\mathrm{n}}} \mapsto \mathbb{R}^{d_y}$, and $C : \mathbb{R}^{d_{\mathrm{n}}} \mapsto \mathbb{R}^{d_y \times d_{\mathrm{l}}}$ (where we have suppressed the dependence on the model parameter θ for brevity). Furthermore, the process noises v_t^{n} and v_t^{l}, as well as the measurement noise e_t are all assumed to be Gaussian. To see that this model is indeed CLG, pretend that $\{x_t^{\mathrm{n}}\}_{t \geq 0}$ is observed. Then, all the functions just listed (which specify the model) are known constants and the model is reduced to a linear Gaussian state-space model. Note that the dynamical equation (40a) is viewed as a measurement equation in this "pretend" linear Gaussian state-space model.

The RBPF exploits the structure of a CLG model by simulating particles representing the nonlinear state

x_t^{n}, while simultaneously tracking the linear state x_t^{l} using a separate Kalman filter for each particle.[4] We do not go in to the details on the RBPF implementation here, but instead refer the interested reader to [32, 27]. Note that CLG models can come in many different forms (hence the rather abstract definition above). However, the variance reduction offered by Rao-Blackwellization can in many cases be substantial, so it is well worth the effort to investigate whether or not a given model under study can be viewed as CLG—thus opening up for using an RBPF—before running a bootstrap particle filter on this model. Automatic Rao-Blackwellization is a recent topic of research in probabilistic programming [33].

7. Numerical illustrations: Learning a nonlinear spring-damper system

In this section, we will walk through a basic but illustrative example of probabilistic learning in an applied situation. The code used in the example is available in the appendix and also via the first author's homepage. For a more challenging numerical example, we refer to, e.g., [5]. The nonlinear spring-damper system that is used to exemplify the method is illustrated in Figure 2 and it is modeled via a spring force F_{s} and a damper force F_{d} according to

$$F_{\mathrm{s}} = -ks^p, \tag{41a}$$

$$F_{\mathrm{d}} = -f_{\mathrm{c}}\mathrm{sign}(\dot{s}) - c_0\dot{s}, \tag{41b}$$

where k denotes the spring coefficient and c_0 denotes the damper coefficient. Furthermore $\dot{s}$ denotes the derivative of the displacement s with respect to time. The spring will be linear for $p = 1$, while $p < 1$ results in a nonlinear spring. Furthermore, $f_{\mathrm{c}} = 0$ gives a linear damper, and the nonlinear damper $f_{\mathrm{c}} > 0$ is motivated in [34]. Via the use of force balance $\sum F = m\ddot{s}$ and a forward-Euler discretization with sampling time T_{s} we obtain the following discrete-time state-space model

$$x_{t+1}^1 = x_t^1 + T_{\mathrm{s}}x_t^2, \tag{42a}$$

$$x_{t+1}^2 = x_t^2 + \tfrac{T_{\mathrm{s}}}{m}(-f_{\mathrm{c}}\mathrm{sign}(x_t^2) - c_0x_t^2 - k(x_t^1)^p) + v_t, \tag{42b}$$

$$y_t = x_t^1 + e_t, \tag{42c}$$

where x_t^1 represents $s(t)$ and x_t^2 represents $\dot{s}(t)$, and we have also included measurement noise e_t and process noise v_t (e.g. an external unknown force). From this model, $T = 1\,000$ data points were simulated with an initial known mass displacement $s(0) = 0.5$, shown in Figure 3. Furthermore, the sampling time was $T_{\mathrm{s}} = 0.1$, the mass $m = 8$, and the noise distributions $e_t \sim \mathcal{N}(0, 0.1^2)$ and $v_t \sim \mathcal{N}(0, 0.01^2)$, which we assume is known, whereas the remaining parameters $\theta = (k, p, f_c, c_0)^{\mathsf{T}}$ are assumed unknown. Their true values, as used in the simulation, are as follows: $k^\star = 2.16$, $p^\star = 0.58$, $f_c^\star = 0.01$ and $c_0^\star = 0.71$ (see also Figure 4).

Let us now specify the priors to be used. The parameters f_{c}, c_0 and k all have to be non-negative by construction, implying that any reasonable prior must be non-negative as well. The Gamma distribution—which we will denote by $\mathcal{G}$—is a reasonable choice. More specifically, since f_{c} is likely to be small we choose $f_{\mathrm{c}} \sim \mathcal{G}(2, 0.01)$, c_0 on the other hand can take on slightly larger values, so its prior is chosen as $\mathcal{G}(2, 1)$. For the spring coefficient k the prior is chosen as $\mathcal{G}(4, .3)$. Finally, p is by physical laws restricted to the interval $0 \leq p \leq 1$, which motivates a uniform prior $\mathcal{U}(0, 1)$. These priors for the parameters are shown in Figure 4.

With the model definition $p(\theta, x_{0:T}, y_{1:T})$ in place we can now start to learn θ. We did this by generating 10 000 samples from $p(\theta \,|\, y_{1:T})$ using PMH as described in Algorithm 3. The result is shown in Figure 4 (together with the true values and the priors). As the proposal in PMH, we used a random walk based on a Gaussian distribution with rather small variance. This takes a few minutes on an ordinary desktop computer. As we have mentioned the PMH algorithm will also provide samples from the state smoothing distribution $p(x_{0:T} \,|\, y_{1:T})$, which we show in Figure 5.

8. Discussion

Probabilistic modeling is about uncertain representations of data and knowledge using probability distributions and how to actually compute these representations by inference and learning algorithms. We have

[4] To be more precise, the RBPF maintains particles representing the entire history of the nonlinear state $x_{0:t}^{\mathrm{n}}$ and conditional on these histories the model is linear subject to Gaussian noise.

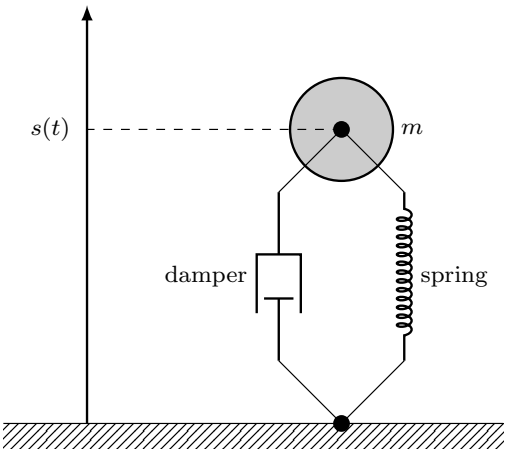

Figure 2: The spring-damper system.

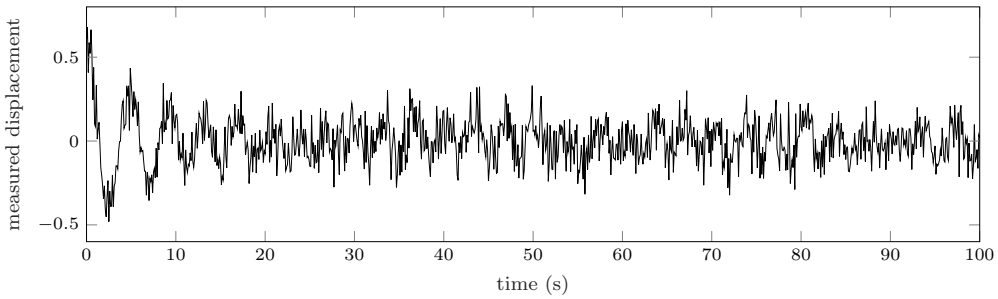

Figure 3: Data simulated from the spring-damper system, which will be used to learn the parameter values θ. As clearly can be seen, there is a significant amount of stochastic noise present in the system, and the data record is not very large.

in this tutorial explained a solution to the problem of computing the posterior distribution $p(\theta \mid y_{1:T})$ of the unknown parameters θ in a nonlinear state-space model (1) using approximate solutions that converge to that posterior distribution as we make use of more computational power. This solution involved the use of a particle filter inside a standard Metropolis–Hastings algorithm, where the particle filter was used to compute non-negative and unbiased estimates of the likelihood which in turn enabled the computation of the relevant acceptance probabilities. There are fairly general software implementations available via the modeling language LibBi [35] which was used to implement the example. We will end this tutorial by briefly reflecting upon some recent developments on probabilistic *representations* and how to carry out the *computations* required to learn and make use of these representations.

The development of new probabilistic models describing nonlinear dynamical phenomena is a highly active and exciting area of research right now. One of the key lessons from modern machine learning is that flexible models often provide the best results [1]. The two dominating approaches for creating flexible models are Bayesian non-parametrics [36] and deep learning [37], both of which can be combined with the state-space model to create flexible and useful models. The most popular Bayesian non-parametric model is arguably the Gaussian process [38] which offers a probabilistic distribution over functions. The Gaussian process construction thus offers a rather natural probabilistic model for the nonlinear functions $f(\cdot)$ and $g(\cdot)$ in (1). Such a construction was developed in [39] and the required learning algorithms were recently enhanced in [40] based on new model approximations. When using flexible models it is important to have a principled way of trading-off their flexibility and the actual fit to data. An emerging way of developing flexible models is provided by probabilistic programming where probabilistic models are represented using computer programs, which was in fact used to implement and solve the example we gave in this tutorial.

The development of new and more capable approximations based on sequential Monte Carlo methods is progressing rapidly. As a first clear trend we mention algorithms scaling to higher-dimensional models, where many of the standard algorithms struggle. Relevant developments in this direction involve localization

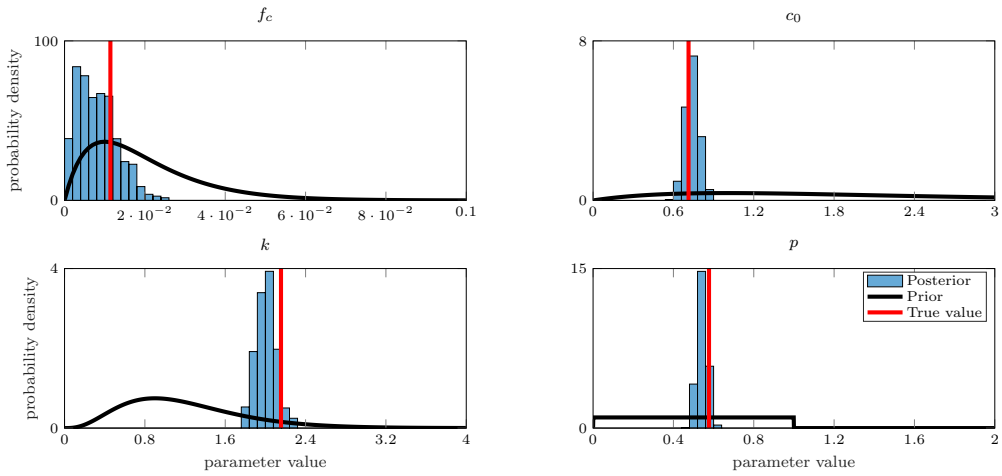

Figure 4: The marginals of the posterior distribution (blue histograms) for the parameters θ, i.e., what can be said about the parameters given measurements $y_{1:T}$ in Figure 3, the priors (black curves) and the model (41) illustrated using vertical lines at the true parameter values.

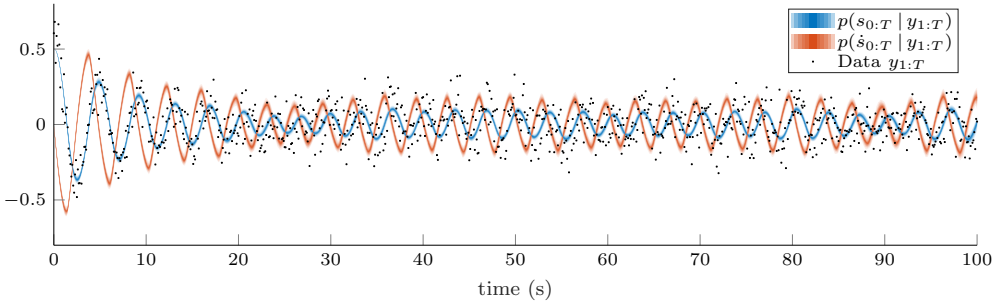

Figure 5: The measurements (dots) alongside with the computed state smoothing distributions, i.e., $p(s_{0:T} \mid y_{1:T})$ (shaded blue) and $p(\dot{s}_{0:T} \mid y_{1:T})$ (shaded orange).

strategies [41], the so-called location particle smoother [42] and the discussion in [43]. Furthermore, the nested SMC method [44] allows us to *exactly approximate* the locally optimal proposal, and significantly extend the class of models for which we can perform efficient inference using SMC. This development also opens up approximate inference in nonlinear spatio-temporal state-space models [45, 46].

A probabilistic graphical model [47, 48, 49] makes use of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to represent the relationships between the random variables used in the model. Each random variable is represented as a vertex/node in the graph and the set of all vertices is denoted by $\mathcal{V}$. The set of edges $\mathcal{E}$ contain elements $(i, j) \in \mathcal{E}$ connecting the two nodes $(i, j) \in \mathcal{V} \times \mathcal{V}$. As an example we provide the directed probabilistic graphical model of the SSM (3) in Figure 6. From this figure it is clear that the SSM is a very specific instance of a probabilistic graphical model, which leads to the rather natural question of whether we can also make use of SMC for inference and learning in more general graphical models. The answer to this question is yes and one key is to introduce a *sequential decomposition* of the graphical model. This sequential decomposition can then be exploited by SMC, since SMC is in fact applicable to any problem that amounts to learning a sequence of probability distributions defined on a sequence of spaces of increasing dimension. See e.g., [50, 51, 52] for developments in this direction. For a tutorial style introduction to the use of SMC for inference in probabilistic graphical models we refer to [53]. This ends our discussion on the second trend, namely the use of SMC type algorithms for inference in more general probabilistic models.

The third trend is the composition of two or more existing algorithms into new and more capable algorithms. We have in this tutorial presented the particle Metropolis–Hastings algorithm—which is one instantiation of this trend—corresponding to a particular combination of the particle filter and the Metropolis–Hastings algorithm. It belongs to the growing family of exact approximation algorithms. The particle Gibbs

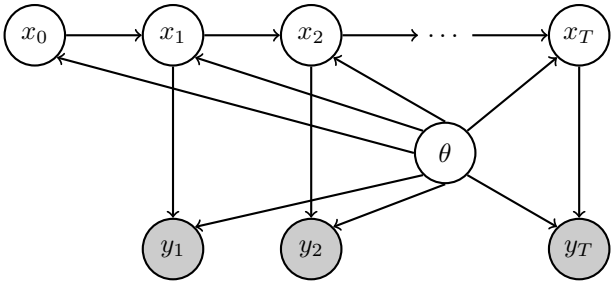

Figure 6: Probabilistic graphical model representing the SSM introduced in (3) for T measurements.

sampler [22] makes use of the so-called conditional particle filter to generate samples from the posterior distribution. The SMC^2 sampler which was independently derived by [54, 55] is similar in spirit to the particle Metropolis–Hastings algorithm in that it makes use of a particle filter within another sampling algorithm. More specifically it makes use of an SMC sampler [56] over parameters θ that in turn has many internal particle filters over the state x. The particle filter can also be composed with another particle filter to produce interesting and capable algorithms. The idea of coupling two particle filters is developed in [57] and the resulting construction can for example be used to build a competitive solution to the nonlinear state smoothing problems.

Acknowledgement

This research is financially supported by the Swedish Research Council via the projects *Probabilistic modeling of dynamical systems* (contract number: 621-2013-5524), *Learning of Large-Scale Probabilistic Dynamical Models* (contract number: 2016-04278) and *NewLEADS - New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079), and the Swedish Foundation for Strategic Research (SSF) via the project *ASSEMBLE* (contract number: RIT15-0012).

References

[1] Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, May 2015.

[2] V. Peterka. Bayesian system identification. *Automatica*, 17(1):41–53, 1981.

[3] T. B. Schön, F. Lindsten, J. Dahlin, J. Wågberg, A. C. Naesseth, A. Svensson, and L. Dai. Sequential Monte Carlo methods for system identification. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, Beijing, China, October 2015.

[4] N. Kantas, A. Doucet, S. S. Singh, J. M. Maciejowski, and N. Chopin. On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351, 2015.

[5] A. Svensson, F. Lindsten, and T. B. Schön. Learning nonlinear state-space models with highly informative observations: a tempered sequential Monte Carlo solution. *Mechanical Systems and Signal Processing (MSSP)*, 2017. Accepted for publication.

[6] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. CRC Press, third edition, 2013.

[7] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York, USA, 2006.

[8] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag New York, 2004.

[9] A. B. Owen. *Monte Carlo theory, methods and examples*. 2013. Book draft.

[10] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machine. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[11] W. K. Hastings. Monte Carlo simulation methods using Markov Chains and their applications. *Biometrica*, 57:97–109, 1970.

[12] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Cambridge University Press, 2009.

[13] C. Robert and G. Casella. A short history of Markov chain Monte Carlo: subjective recollections from incomplete data. *Statistical Science*, 26(1):102–115, 2011.

[14] M. A. Beaumont. Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160, 2003.

[15] C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

[16] C. Andrieu and M. Vihola. Convergence properties of pseudo-marginal Markov chain Monte Carlo algorithms. *The Annals of Applied Probability*, 25(2):1030–1077, 2015.

[17] L. Stewart and P. McCarty. The use of Bayesian belief networks to fuse continuous and discrete information for target recognition and discrete information for target recognition, tracking, and situation assessment. In *Proceedings of SPIE Signal Processing, Sensor Fusion and Target Recognition*, volume 1699, pages 177–185, 1992.

[18] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

[19] G. Kitagawa. A Monte Carlo filtering and smoothing method for non-Gaussian nonlinear state space models. In *Proceedings of the 2nd US-Japan joint Seminar on Statistical Time Series Analysis*, pages 110–131, 1993.

[20] P. Del Moral. *Feynman-Kac formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Applications. Springer, New York, USA, 2004.

[21] M. K. Pitt, R. dos Santos Silva, R. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.

[22] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society. Series B (Methodology)*, 72(2):1–33, 2010.

[23] J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, January 2015.

[24] G. Deligiannidis, A. Doucet, and M. K. Pitt. The correlated pseudo-marginal method. Technical report, arXiv preprint arXiv:1511.04992, 2015.

[25] A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovskii, editors, *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press, Oxford, UK, 2011.

[26] O. Cappé, S. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[27] A. Doucet, S. J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[28] R. Douc, O. Cappé, and E. Moulines. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, Zagreb, Croatia, September 2005.

[29] J. D. Hol, T. B. Schön, and F. Gustafsson. On resampling algorithms for particle filters. In *Proceedings of the Nonlinear Statistical Signal Processing Workshop*, Cambridge, UK, September 2006.

[30] M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[31] T. B. Schön and F. Lindsten. Learning of dynamical systems – Particle filters and Markov chain methods. Draft manuscript, 2017.

[32] R. Chen and J. S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society. Series B (Methodology)*, 62(3):493–508, 2000.

[33] L. M. Murray, D. Lundén, J. Kudlicka, D. Broman, and T. B. Schön. Delayed sampling and automatic Rao–Blackwellization of probabilistic programs. Technical report, arXiv:1708.07787, 2017.

[34] B. F. Spencer, S. J. Dyke, M. K. Sain, and J. D. Carlson. Phenomenological model for magnetorheological dampers. *Journal of engineering mechanics*, 123(3), 1997.

[35] L. M. Murray. Bayesian state-space modelling on high-performance hardware using LibBi. *Journal of Statistical Software*, 67(10):1–36, 2015.

[36] Z. Ghahramani. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering*, 371(20110553), 2013.

[37] Y. LeCun, Y. Bengio, and Hinton G. Deep learning. *Nature*, 521:436–444, 2015.

[38] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[39] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. In *Advances in Neural Information Processing Systems (NIPS) 26*, Lake Tahoe, NV, USA, December 2013.

[40] A. Svensson and T. B. Schön. A flexible state space model for learning nonlinear dynamical systems. *Automatica*, 80:189–199, June 2017.

[41] P. Rebeschini and R. van Handel. Can local particle filters beat the curse of dimensionality? *Annals of Applied Probability*, 25(5):2809–2866, 2015.

[42] J. Briggs, M. Dowd, and R. Meyer. Data assimilation for large-scale spatio-temporal systems using a location particle smoother. *Environmetrics*, 24(2):81–97, 2013.

[43] P. M. Djuric and M. F. Bugallo. Particle filtering for high-dimensional systems. In *IEEE 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, St. Martin, France, 2013.

[44] A. C. Naesseth, F. Lindsten, and T. B. Schön. Nested sequential Monte Carlo methods. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, July 2015.

[45] C. K. Wikle. Modern perspectives on statistics for spatio-temporal data. *WIREs Computational Statistics*, 7(1):86–98, 2015.

[46] N. Cressie and C. K. Wikle. *Statistics for spatio-temporal data*. Wiley, 2011.

[47] M. I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, 2004.

[48] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

[49] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[50] A. C. Naesseth, F. Lindsten, and T. B. Schön. Sequential Monte Carlo for graphical models. In *Advances in Neural Information Processing Systems (NIPS) 27*, Montreal, Quebec, Canada, December 2014.

[51] A. Beskos, D. Crisan, A. Jasra, K. Kamatani, and Y. Zhou. A stable particle filter for a class of high-dimensional state-space models. *Advances in Applied Probability*, 49(1):24–48, 2017.

[52] F. Lindsten, A. M. Johansen, A. C. Naesseth, B. Kirkpatrick, T. B. Schön, J. Aston, and A. Bouchard-Côté. Divide-and-Conquer with sequential Monte Carlo. *Journal of Computational and Graphical Statistics (JCGS)*, 26(2):445–458, 2017.

[53] A. Doucet and A. Lee. *Sequential Monte Carlo methods*, chapter in Handbook of Graphical Models. 2017.

[54] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos. SMC^2: an efficient algorithms for sequential analysis of state-space models. *Journal of the Royal Statistical Society, Series B*, 75(3):397–426, June 2013.

[55] A. Fulop and J. Li. Efficient learning via simulation: A marginalized resample-move approach. *Journal of Econometrics*, 176(2):146–161, October 2013.

[56] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society. Series B (Methodology)*, 63(3):411–436, 2006.

[57] P. E. Jacob, F. Lindsten, and T. B. Schön. Smoothing with couplings of conditional particle filters. Technical report, arXiv:1701.02002, 2017.

Appendix A. Implementation in LibBi

The numerical example was implemented using LibBi ([35], `www.libbi.org`). The code needed to reproduce the examples will be available via the first authors homepage. Apart from simulating data, the LibBi code for implementing the model (41) and run the PMH learning algorithm is provided below.

First, the following content is placed in a file called `Damper.bi`.

```
model Damper {
  /* Specifying all variables in the problem */
  const T_s = .1, m = 2           /* Known constants */
  param f_c, c_0, k, p            /* Unknown variables */
  noise v                         /* Process noise */
  state s, sdot, s_old, sdot_old  /* State variables */
  obs y

  sub parameter {
    /* Specifying all priors for the unknown parameters  */
    f_c ~ gamma(2, 0.01)
    c_0 ~ gamma(2, 1)
    k ~ gamma(4, 0.3)
    p ~ uniform(0, 1)
  }

  sub proposal_parameter {
    /* Specifying the random walk proposal used in PMH */
    f_c ~ truncated_gaussian(f_c, 1.0e-3, 0.0)
    c_0 ~ truncated_gaussian(c_0, 1.0e-2, 0.0, 1.0)
    k ~ truncated_gaussian(k, 1.0e-2, 0.0)
    p ~ truncated_gaussian(p, 1.0e-2, 0.0, 1.0)
  }

  sub initial {
    /* Specifying the inital state variables */
    s <- 0.5
    sdot <- 0.0
  }

  sub transition(delta = 1) {
    /* Describe how the states in the model evolves at each time step. The variables s_old and sdot_old is only
          for temprary storage of the previous state values. */
    v ~ gaussian(0.0, 1.0e-2)

    s_old <- s
    sdot_old <- sdot

    s <- s_old + T_s*(sdot_old)
    sdot <- sdot_old + (T_s/m)*(-f_c*((sdot_old < 0) ? -1 : 1) - c_0*sdot_old - k*((s_old < 0) ? -1 : 1)*pow(
         abs(s_old), p)) + v
  }

  sub observation {
    /* Describe how the measurements relates to the states */
    y ~ gaussian(s, 1.0e-1)
  }
}
```

Once the file `Damper.bi` is in place, the following command can be run from the command line:

```
libbi sample --target posterior --model-file Damper.bi --nsamples 10000 --nparticles 256 --end-time 1000 --
    sampler mh --obs-file data/obs.nc --output-file results/posterior.nc
```

which will produce 10 000 samples from $p(\theta \mid y_{1:1000})$, where $y_{1:1000}$ is found in the file `data/obs.nc`. The samples are generated by the PMH algorithm and recorded in the file `results/posterior.nc` which can be loaded into, e.g., Python, Matlab or R for further processing.