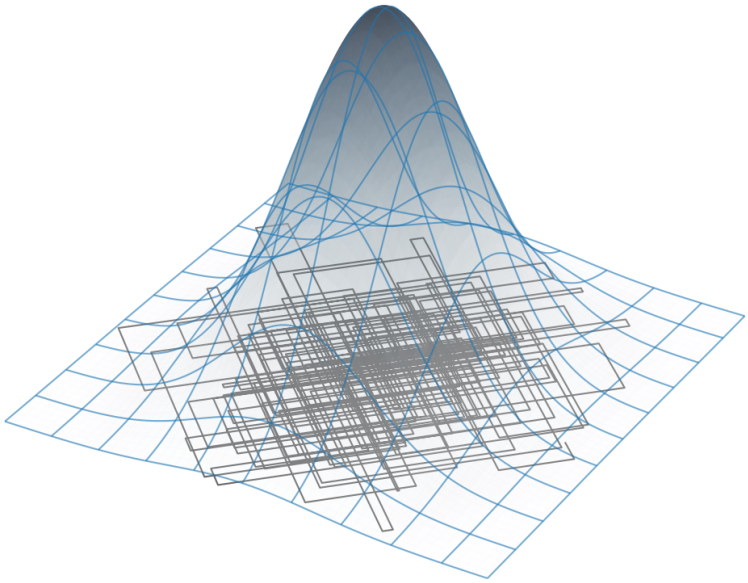


Fast Bayesian Inference with Piecewise Deterministic Markov Processes

Estimation of Latent Parameters of an Adversarial Missile

Master's thesis in Physics

KARL HAMMAR

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Fast Bayesian Inference with Piecewise Deterministic Markov Processes

Estimation of Latent Parameters of an Adversarial Missile

KARL HAMMAR

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2023

Fast Bayesian Inference with Piecewise Deterministic Markov Processes
Estimation of Latent Parameters of an Adversarial Missile
KARL HAMMAR

Supervisor: Adam Andersson, Saab
Supervisor: Benjamin Svedung Wettervik, Saab
Examiner: Moritz Schauer, Department of Mathematical Sciences, Chalmers

Master's Thesis 2023
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Zig-Zag sampling of a multivariate Gaussian distribution.

Typeset in LaTeX
Gothenburg, Sweden 2023

Fast Bayesian Inference with Piecewise Deterministic Markov Processes
Estimation of Latent Parameters of an Adversarial Missile
KARL HAMMAR
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Piecewise Deterministic Markov Processes (PDMPs) present a recent class of samplers for Bayesian inference. In this thesis, PDMP samplers are employed to sample state and latent parameters of an adversarial missile, described by an SDE. An approximate method for fast sampling is developed for this problem, and the performance of two different PDMP samplers, the Zig-Zag sampler, and the bouncy particle sampler, are compared. We find that the approximations needed for the methods to be competitive have a small impact on accuracy and that the method has the potential to be useful in real-world applications. Additionally, an approach for sampling from target models which may experience discontinuous jumps is developed. Using a particular trajectory realization of one such model we show that the method works as expected. This bears importance for the sampling of parameters related to maneuvering target types, where jump dynamics are relevant for target modeling.

Keywords: Bayesian inference, Stochastic process, Piecewise Deterministic Markov Process, State estimation.

Acknowledgements

First I would like to thank my supervisors at Saab, Adam Andersson and Benjamin Svedung Wettervik. Your expertise has guided me throughout this project and your enthusiasm has made the journey oh-so joyful. You have truly helped me in more ways than you imagine, and for this I thank you. I would like to thank my academic supervisor at Chalmers, Moritz Schauer, for giving advice about the project at large and PDMPs in particular. It is always a pleasure to meet you. To all the incredible people who I have met at Saab, thank you for providing me with laughter, friendship, and much-needed breaks from work. To my partner, Anna Manarczyk, thank you for your love and support. Lastly, I would like to thank my family, Lars, Johanna, Emma, and Anna Hammar, for your support and encouragement.

Karl Hammar, Gothenburg, April 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order.

ACF	Autocorrelation function
BPS	Bouncy particle sampler
EM	Euler-Maruyama
ESS	Effective sample size
MAE	Mean absolute error
MAP	Maximum a posteriori probability
MH	Metropolis-Hastings
ODE	Ordinary differential equation
PDMP	Piecewise Deterministic Markov Process
RK4	Fourth order Runge-Kutta method
SDE	Stochastic differential equation
SZZ	Sticky Zig-Zag sampler
UKF	Unscented Kalman filter
ZZ	Zig-Zag sampler

Nomenclature

Below is the nomenclature of mathematical symbols and operations that are used throughout this thesis.

$\mathbb{N}_0$	Natural numbers including 0
$\mathbb{R}_+$	Positive real numbers, $[0, \infty)$
$[a..b]$	Integers between a and b, $\{x \in \mathbb{Z} : a \leq x \leq b\}$
$\langle a, b \rangle$	Scalar product between euclidean vectors a and b
$(a)^+$	$\max(0, a)$
$C^n(E)$	Functions on E that are n times differentiable
$B(E)$	Bounded real valued functions on E
$B_\mathrm{b}(E)$	Bounded Borel-measurable function on E
$\mathcal{B}_E$	The Borel σ-algebra on E
A^c	Set complement of A
$\mathbb{1}_A(\cdot)$	Indicator function on A
0_d	Zero-vector of length d
I_d	Identity matrix of size $d \times d$
$\mathcal{U}(a, b)$	Uniform distribution on $[a, b]$
$\mathcal{N}(\mu, \mathbf{\Sigma})$	Gaussian distribution with mean μ and covariance $\mathbf{\Sigma}$

Contents

1

Introduction

Inference plays an important role in both everyday situations and in science. It can be defined as the task of gaining knowledge about some system based on data generated by it. In science this is often deliberate and explicit, we perform experiments to gather data which are used to learn quantitative results about some phenomena. A large part of our understanding of the world is thus based on different kinds of inference.

This thesis concerns the task of state estimation for an adversarial missile. At its heart, this is an inference task. Data are given in the form of radar measurements of the missile (target), and we want to infer information about parameters related to its state. Specifically, we are interested in drag-model parameters and the kinematic state of the missile. One of the main motivations for wanting to learn these parameters is that extrapolation of target states outside the set of supporting measurements can be done with greater accuracy with this information. This is important for the Point of origin and Point of impact problems, where one extrapolates data to figure out from where a missile was launched, and where it might land, respectively. Information about the point of origin/impact is crucial for the ability to apply counter-measures, or take cover.

The Bayesian framework is especially relevant for these problems since it provides a full probability distribution for the specified variables in the posterior. Compared to a point estimate, this is a rich description. Furthermore, prior knowledge can be incorporated in a straightforward way. Unfortunately, Bayesian posteriors can often be challenging to obtain and usually require the use of sampling methods that produce approximations of the posterior. These methods are often computationally heavy. The challenge, therefore, lies in being able to sample the posterior with sufficient accuracy and on timescales fast enough such that the obtained information is still useful for the actor who makes measurements of the missile.

Piecewise Deterministic Markov Processes (PDMPs) provide a family of promising methods to sample probability distributions. Contrary to standard samplers such as the Metropolis-Hastings sampler, PDMPs do not require a proposal distribution which simplifies their construction. They also sample the target space in continuous rather than discrete time, which has been shown to speed up convergence [1]. Because of this, we aim to examine if and how PDMPs can be used to efficiently sample probability distributions related to the state estimation problem for an adversarial missile.

1.1 Modelling targets with SDEs

In this thesis, we use stochastic target models to describe different types of missiles. Why is that? Missiles are, as far as we are concerned, deterministic in nature. They follow the classical laws of mechanics, which means that their motion can be deduced from the forces which act upon them such as thrust, gravity, drag, wind, and other pressure differentials. If the target is agile, some algorithm or human may influence these forces, but ultimately there is nothing *random* about them. For a distant observer, however, many of these forces are difficult or impossible to know. It is therefore not realistic to model the target perfectly.

Since no model is a perfect representation of reality, we are always left with *some* amount of model mismatch no matter how we choose our model, and we need a way to handle this. One way to do so that is adopted in this thesis is to model our targets as a stochastic differential equation (SDE). An SDE can informally be described as an ODE with some added noise. The noise can essentially be of any type that we please, and it is practical to choose it in a way that represents the model aspects that we were not able to capture explicitly. Examples of this could include unmodeled vibrations (perhaps with some continuous noise) or control inputs (perhaps with some discrete and discontinuous noise). We usually refer to the noise term(s) in a stochastic model as process noise. One way to interpret it is as an uncertainty in the model which represents the degree of model mismatch.

1.2 Outline

This thesis is structured as follows: In Chapter 2 some general theory on Markov processes is presented, which unifies our target models (SDEs) and PDMPs under a common framework. In Chapter 3 the tools we use to solve the Bayesian inference problem are presented. This includes specific PDMP samplers and a Bayesian filtering algorithm. Chapter 4 considers practical aspects of using PDMPs to sample probability distributions. A significant challenge is how to choose computational bounds for PDMP simulation. In Chapter 5 target and measurement models are described. Here, we also present a method for sampling from an SDE with jumps. Our results are presented in Chapter 6. In Chapter 7 a discussion is held about our results and possible future research areas and Chapter 8 concludes our results.

Before continuing, we recommend the reader who is unacquainted with (or the reader who needs to refresh their memory on) measure theory and stochastic processes to read Appendix A.1–A.3. This theory lays the foundation for the following chapters, especially Chapter 2 and 3.

2

Markov processes

A Markov process is a stochastic process that satisfies the *Markov property*. The Markov property states that given all the information up to an arbitrary time, the only information needed to predict the future evolution of the process beyond this time is the information of the current state. The Markov property is often valid in physical systems and Markov processes therefore lend themselves well as models in engineering applications. The theory of Markov processes is rich and well-developed. A natural way to represent models of physical phenomena is by stochastic differential equations, whose solutions turn out to be Markov processes. In Sections 2.1–2.3 some general theory for Markov processes is presented. In Section 2.4 important examples of Markov processes, all used in this thesis, are presented. These are diffusion processes, pure jump processes (including PDMPs as a special case), and jump-diffusion processes. Finally in Section 2.5, an analytic solution to a linear SDE is obtained, and numerical approximation schemes are presented for non-linear SDEs.

2.1 Markov processes and their characterizations

Let us start by making the Markov property of stochastic processes precise. The Markov property is concerned with information encoded in the filtration on the probability space on which a process lives. Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbf{P})$ be a filtered probability space on which a stochastic process $(X_t)_{t\geq 0}$, adapted to $(\mathcal{F}_t)_{t\geq 0}$ lives. For ease of reading, $(X_t)_{t\geq 0}$ is often simply denoted X. We say that X is a *Markov process* if it satisfies the Markov property,

$$\mathbf{E}[f(X_t)|\mathcal{F}_s] = \mathbf{E}[f(X_t)|X_s], \quad 0 \leq s \leq t < \infty, \tag{2.1}$$

for any $f \in B_{\mathrm{b}}(E)$, where $B_{\mathrm{b}}(E)$ denotes the bounded and Borel-measurable functions on a state space E. When conditioning on X_s in the expected value, we more precisely mean the σ-algebra generated by X_s, i.e., $\sigma(X_s)$. Equation (2.1) says that out of all the information contained in $\mathcal{F}_s$, only the state of the process at the latest instant is needed to determine the future. Stated differently, given the present, the future is independent of the past. Markov processes are often referred to as *memoryless* for this reason.

Markov processes constitute a rather large class of processes, and the question then arises of how they can be characterized and understood. An instructive question is the following: *Given the state X_t of a Markov process X at time t, what is the*

state or distribution of states at some later time $t + s$? It turns out that there are multiple interconnected ways to answer this question, and that a Markov process can be characterized in multiple (equivalent) ways. In the following, we present three common characterizations, those are the *operator semigroup*, the *transition probabilities*, and the *infinitesimal generator*.

We start with defining the operator semigroup[1] for a Markov process. Let $B(E)$ be the space of bounded functions $E \to \mathbb{R}$ on a state-space E, equipped with the supremum norm

$$\|f\| = \sup_{x \in E} |f(x)|.$$

Given a function $f \in B(E)$ and $x \in E$, define a family of operators $(T_t)_{t \geq 0}$ as

$$(T_t f)(x) = \mathbf{E}[f(X_t)|X_0 = x], \quad 0 \leq t < \infty. \tag{2.2}$$

The operator T_t finds the expected value of some function $f \in B(E)$ of X_t at a time t, given the state $X_0 = x$ at time 0. This results in a new function $T_t f \in B(E)$. The family $(T_t)_{t \geq 0}$ is often called the operator (or transition) semigroup since it can be shown to satisfy the semigroup property

$$T_{s+t} = T_s T_t, \ s, t \geq 0, \quad T_0 = \mathrm{id},$$

and is a closed operation on $B(E)$. In order to make the connection to the Markov process explicit, consider the special choice $f = \mathbb{1}_A, A \in E$ which yields

$$(T_t \mathbb{1}_A)(x) = \mathbf{E}[\mathbb{1}_A(X_t)|X_0 = x] = \mathbf{P}(X_t \in A|X_0 = x). \tag{2.3}$$

Equation (2.3) nicely relates to a family of *transition probabilities*, which are defined next. Let $\mathcal{B}_E$ denote the Borel σ-algebra on E. For $0 \leq t < \infty$, $A \in \mathcal{B}_E$ and $x \in E$ we define the transition probability

$$p_t(x, A) = (T_t \mathbb{1}_A)(x).$$

By considering all $t \geq 0$, a family of transition probability functions $(p_t)_{t \geq 0}$, indexed by the time variable t, is formed. For each t, a transition probability function specifies the probability to transition from x to A in time t, i.e., given that the process is now in state x, what is the probability of being in the set A a time t later. Just like the operator semigroup in (2.2), the family of transition probabilities characterizes the Markov process. Note in particular that $p_0(x, A) = \mathbb{1}_A(x)$. It can be seen that $(p_t)_{t \geq 0}$ is also a semigroup with convolution as its operator.

The final way in which we characterize a Markov process is by means of its *infinitesimal generator*. The infinitesimal generator can be thought of as the derivative of the operator semigroup with respect to its time parameter at $t = 0$. Given the operator semigroup, the generator is obtained as its derivative. Conversely, given

[1] A semigroup is a set S with an associative operator $\cdot : S \times S \to S$. It has all the properties of a group except the existence of an inverse.

the generator, the semigroup can be obtained or generated. The generator, therefore, provides an equivalent way to represent a Markov process. The infinitesimal generator $\mathcal{A}$ of the semigroup $(T_t)_{t\geq 0}$ is the operator with domain

$$\mathcal{D}(\mathcal{A}) = \left\{ f \in B(E) : \lim_{t\downarrow 0} \frac{T_t f - f}{t} \text{ exists} \right\}$$

and action

$$\mathcal{A}f = \lim_{t\downarrow 0} \frac{T_t f - f}{t}, \quad f \in \mathcal{D}(\mathcal{A}).$$

Given $f \in \mathcal{D}(\mathcal{A})$ we have by the semigroup property of T and the definition of $\mathcal{A}$ that

$$\partial_t (T_t f) = \lim_{h\downarrow 0} \frac{T_{t+h} f - T_t f}{h} = \lim_{h\downarrow 0} \frac{T_h T_t f - T_t f}{h} = \mathcal{A} T_t f \tag{2.4}$$

Thus defining $u(t,x) := (T_t f)(x)$ we have that

$$\partial_t u = \mathcal{A}u, \ t > 0; \quad u(0,x) = f(x), \ x \in E. \tag{2.5}$$

This means that given the generator $\mathcal{A}$, the transition semigroup $(T)_{t\geq 0}$ is obtained by solving the abstract linear differential equation (2.5). This equation is called Kolmogorov's Backward equation. As exemplified below, in applications the generator $\mathcal{A}$ is a differential operator or integro-differential operator (an operator with both derivatives and integrals). The Kolmogorov backward equation is thus a partial differential equation or partial integro-differential equation.

The operator semigroup $(T_t)_{t\geq 0}$, the corresponding family of transition probabilities $(p_t)_{t\geq 0}$ and the infinitesimal generator $\mathcal{A}$ all fully represent the same process X. Without proof (Proposition 14.10 in [2] for details), we give the following result for $f \in \mathcal{D}(\mathcal{A})$ and $t \geq 0$:

$$T_t f - f = \int_0^t \mathcal{A} T_t f \mathrm{d}s = \int_0^t T_t \mathcal{A} f \mathrm{d}s.$$

The first equality is simply the integral form of (2.4) whereas in the second equality, a factor T_t has been taken outside the limit in the same equation.

2.2 Invariant measures

Given a Markov process X, a question one could ask is the following: does the distribution of X converge in some suitable sense as time tends to infinity? If it does converge, then what does it converge to, and can it converge in multiple ways? The concepts needed to answer these questions are those of *invariant measures* and *ergodicity*.

Given that the distribution of a process does converge in a suitable sense, we call the limiting distribution an invariant or stationary distribution (or measure). There

is of course no guarantee that such a distribution exists. Formally, a measure π is said to be invariant (or stationary) if

$$\pi(x) = \int_E \pi(y) p_t(\mathrm{d}y, x), \quad \text{for all } x \in E, t > 0,$$

where $(p_t)_{t\geq 0}$ is the family of transition probabilities for the process. Thus if the process starts in the stationary distribution π, the process will remain there for any $t > 0$. That is what we expect in the stationary state.

One way to prove that a particular process admits a particular invariant measure is by the infinitesimal generator. The following theorem provides a link between the infinitesimal generator and the invariant measure.

Theorem 2.2.1. *Let $D \subset \mathcal{D}(\mathcal{A})$ be a subset such that for any $f \in \mathcal{D}(\mathcal{A})$ there exists $f_n \in D$ such that $f_n \to f$ and $\mathcal{A}f_n \to \mathcal{A}f$. A probability measure μ on E is an invariant measure if and only if*

$$\int_E \mathcal{A}f \mathrm{d}\mu = 0 \quad \textit{for all } f \in D.$$

Proof sketch for 2.2.1. Assume that μ is an invariant measure and $f \in \mathcal{D}(\mathcal{A})$, then

$$\int_E \mathcal{A}f \mathrm{d}\mu = \int_E \lim_{t\downarrow 0} \frac{T_t f - f}{t} \mathrm{d}\mu = \lim_{t\downarrow 0} \frac{1}{t} \left(\int_E T_t f \mathrm{d}\mu - \int_E f \mathrm{d}\mu \right). \tag{2.6}$$

In the first step, the definition of the generator was used. In the second equality, we have taken the limit outside the integral. This is allowed by dominated convergence since the integrand is a bounded function. By definition,

$$\int_E T_t f \mathrm{d}\mu = \mathbf{E}[(T_t f)(X_\infty)] = \mathbf{E}[\mathbf{E}[f(X_t)|X_0 = X_\infty]],$$

where X_∞ is a random variable distributed according to the stationary distribution μ. Due to stationarity, we have

$$\mathbf{E}[\mathbf{E}[f(X_t)|X_0 = X_\infty]] = \mathbf{E}[\mathbf{E}[f(X_s)|X_0 = X_\infty]], \quad s, t \geq 0.$$

In particular this hold for $t = 0$, and since $T_0 = \mathrm{id}$, we have $\int_E T_t f \mathrm{d}\mu = \int_E f \mathrm{d}\mu$, and hence the limit in (2.6) converges to 0. The proof for the converse is omitted and is found in [3] (Theorem 3.37). □

Given that an invariant measure does exist, an important question is whether or not it is unique, or if the process can converge towards different invariant measures for different initial values or by random chance. This is the essence of ergodicity. Proving uniqueness of an invariant distribution is often tricky, and must be done on a process-by-process basis. Unfortunately, the technical definition of ergodicity is out of the scope of this thesis and we suggest the reader to think about ergodicity in terms of uniqueness of invariant measures, which is sufficient for our purposes.

2.3 Markov kernels

Markov kernels are used frequently throughout this thesis, specifically they are utilized to specify jump transitions in the construction of PDMPs in Chapter 3, but they are also important in the theory of Markov process as a whole. Given this reliance on Markov kernels, a definition is appropriate.

Formally, a Markov kernel is a map from one measurable space to another. If $(E, \mathcal{B}_E)$ and $(E', \mathcal{B}'_E)$ are two measurable spaces, a map $Q : E \times \mathcal{B}'_E \to [0, 1]$ is a Markov kernel if it satisfies the following criteria:

1. For fixed $x \in E$, $Q(x, \cdot)$ is a probability measure on $(E', \mathcal{B}'_E)$.
2. For fixed $A' \in \mathcal{B}'_E$, $Q(\cdot, A')$ is $\mathcal{B}_E$-measurable.

This definition is more general than what is needed in this thesis, as we only deal with Markov kernels within a single measurable space. The first requirement states roughly that for any state $x \in E$, Q describes a probability distribution from which a new state is picked. The second requirement is that of measurability, which is needed when integrating (for example in (2.11)).

We have in fact already seen an example of Markov kernels. The family of transition probabilities $(p_t)_{t\geq 0}$ fulfill the above requirements and are Markov kernels. We ask the reader to pay special attention to the difference between Markov kernels used to specify a full Markov process, which we designate $(p_t)_{t\geq 0}$, and Markov kernels used to specify jumps, designated Q. In the latter case, the Markov kernels are only used as a component of a Markov process, while transition probabilities $(p_t)_{t\geq 0}$ specify a Markov process completely.

2.4 Examples of Markov processes

We next present some examples of Markov processes that are used in this thesis. It turns out that both our target models and Piecewise Deterministic Markov processes fit within the framework of Markov processes, which is quite elegant. In Section 2.4.1 diffusion processes are introduced. For our purposes, these are solutions to stochastic differential equations, used as target models. In Section 2.4.2 jump processes are introduced, which is where Piecewise Deterministic Markov processes fit into the Markov process framework. Finally in Section 2.4.3 we present jump-diffusion processes where the concepts of diffusion and jumps are merged.

2.4.1 Diffusion processes

An important class of Markov processes is that given by solutions to *stochastic differential equations* (SDEs). Stochastic differential equations generalize the concept of ordinary differential equations to the realm of probability theory and stochastic processes. A stochastic differential equation is formed by the inclusion of a stochastic process in one or more terms of an ODE. These stochastic components are often referred to as *noise*, which can be of different types. The most common type of noise is *Gaussian noise*, which corresponds to adding a Brownian motion to an ODE.

The resulting processes are called *diffusion processes*. Consider a Brownian motion W adapted to a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbf{P})$. A diffusion SDE on differential form then reads

$$\mathrm{d}X_t = \mu(X_t)\mathrm{d}t + \sigma(X_t)\mathrm{d}W_t, \quad t > 0, X_0 = x_0, \tag{2.7}$$

where μ is a *drift* coefficient and σ is a *diffusion* coefficient. In general, the coefficients are explicitly dependent on the time t, but we restrict ourselves to the time-homogeneous case. For $\sigma = 0$ we recognize an ordinary differential equation. If σ is constant, we say that the noise is *additive* and if σ depends on X_t we say the the noise is *multiplicative*.

The solution to an SDE can informally be understood as a function that satisfies the integral form of the SDE. It turns out that there are actually two notions of the solutions to an SDE, the *strong* and *weak* notion. A strong solution implies a weak solution, but the converse is not true. We only define the strong solution concept, following [4].

Definition 2.4.1. *A stochastic process $(X_t)_{t\geq 0}$ is a strong solution to the diffusion SDE in* (2.7) *if*

$$X_t = X_0 + \int_0^t \mu(X_s)\mathrm{d}s + \int_0^t \sigma(X_s)\mathrm{d}W_s \tag{2.8}$$

exists and is satisfied for all $t > 0$.

Here, $\int_0^t \sigma(X_s)\mathrm{d}W_s$ is an Itô integral, defined in Appendix A.3. The strong solution is a functional of the noise realization, i.e., given a noise realization, the strong solution reconstructs the exact path of the SDE while satisfying the integral equation (2.8).

2.4.2 Jump processes

Jump processes are a class of Markov processes that are characterized by discontinuous jumps. The archetypal jump process is the Poisson process. A counting process $(N_t)_{t\geq 0}$ is a Poisson process with rate Λ if it satisfies:

1. $N_0 = 0$,
2. N_t has independent increments, i.e. $N_t - N_s$ is independent of $N_s - N_r$ for any $0 \leq r < s < t$,
3. N_t is integer-valued and the number of jumps in a time period of length τ is Poisson($\Lambda\tau$)-distributed.

A realization of a Poisson process is shown in Figure 2.1. Note that the function is continuous from the right with existing left limits. This type of function is of fundamental importance for jump processes and has been given a name: *càdlàg*. The name comes from the French description of their properties, *continue à droite, limite à gauche*.

The *compound Poisson process* is a generalization of the Poisson process which allows jumps of different sizes. Given a Poisson process $(N_t)_{t\geq 0}$, the compound Poisson process $(M_t)_{t\geq 0}$ is defined as

$$M_t = \sum_{i=1}^{N_t} D_i, \quad t \geq 0,$$

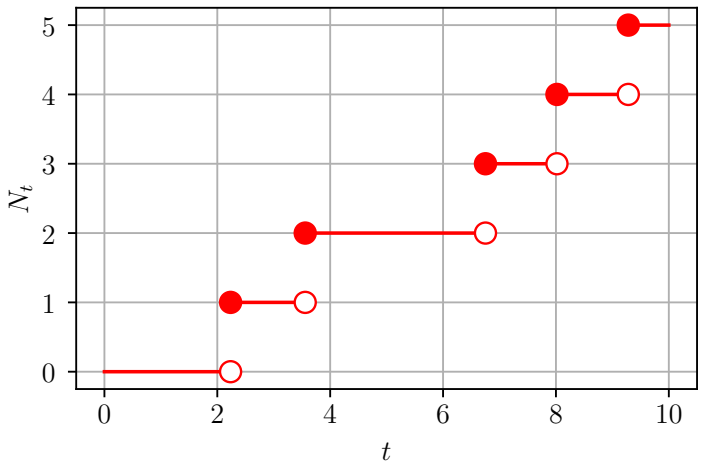


Figure 2.1: Example of a realization of a Poisson process with $\Lambda = 1$.

where D_i are independent and identically distributed random variables. A compound Poisson process retains the event times of the regular Poisson process but modifies the jump magnitudes.

2.4.2.1 Integral with respect to compound Poisson process

If we want to use jump processes in modeling, it would be convenient if we could put compound Poisson processes into the framework of SDEs. In order to do so, let $(D_n), n = 1, 2, \ldots$ be a sequence of independent and identically distributed random variables with distribution π and let (t_n) be a sequence of event times from a standard Poisson process with rate Λ. The *counting measure* $N(\mathrm{d}t, \mathrm{d}x)$ is then defined as

$$N(\mathrm{d}t, \mathrm{d}x) = \sum_{n=1}^{\infty} \delta_{(t_n, D_n)}(\mathrm{d}t, \mathrm{d}x).$$

It is naturally a random measure and every outcome of it represents an infinite collection of jumps situated in space and time. For any time $t \geq 0$ and increment Δt, the interval $[t, t + \Delta t]$ almost surely contains a finite number of jumps. Given a càdlàg process X on the state space $E = \mathbb{R}^d$, we can define the integral

$$\int_{\mathbb{R}^d} G(X_{t-}, y) N(\mathrm{d}t, \mathrm{d}y), \tag{2.9}$$

where G is some function. With the choice $G(X_{t-}, y) = y$ the compound Poisson process itself is obtained which means that (2.9) is a generalization of the compound Poisson process. We have used $t-$ to indicate the left limit, which is required when working with càdlàg processes, as left and right limits may be different.

2.4.2.2 Piecewise Deterministic Markov Processes

Piecewise Deterministic Markov Processes constitute a class of Markov processes that are useful for their ability to sample certain probability measures. In this thesis,

multiple such processes are used, and since they are examples of jump processes, we motivate their introduction in this framework.

A PDMP can be described as deterministic dynamics, specified as an ODE, accompanied by stochastic jumps. The jump dynamics consists of two parts, a state-dependent *event-rate* $\lambda : E \to \mathbb{R}^+$, which specifies the rate at which jumps occur, and a state-dependent transition kernel Q, which specifies jump distributions. During time intervals of no jumps, the deterministic dynamics are described by the ODE

$$\mathrm{d}X_t = \mu(X_t)\mathrm{d}t, \quad t > 0, \tag{2.10}$$

with initial condition given from the last jump transition. A function X_t is a solution to (2.10) if and only if it satisfies

$$\mathrm{d}f(X_t) = \mathfrak{X}f(X_t)\mathrm{d}t = \sum_{i=1}^{d} \mu_i(X_t)\partial_i f(X_t)\mathrm{d}t, \quad t > 0,$$

for all $f : \mathbb{R}^d \to \mathbb{R}$ such that $f \in C^\infty(\mathbb{R}^d)$ [2]. The operator $\mathfrak{X}$ is often called a vector field and describes the deterministic dynamics of the PDMP given that no jumps occur.

A PDMP can thus be thought of as the following three components:

1. The ODE: The process follows deterministic dynamics on the state space E, corresponding to the vector field $\mathfrak{X}$.
2. The events: Events are generated from an inhomogeneous Poisson process with a state-dependent intensity $\lambda : E \to \mathbb{R}^+$.
3. The transition kernel: At event times, the state of the process is modified according to the transition kernel (a Markov kernel) Q. These state changes are discontinuous jumps.

The triplet $(\mathfrak{X}, \lambda, Q)$ is referred to as the local characteristics of a PDMP, and is enough to characterize it. When we introduce specific PDMPs in Chapter 3, we do so by specifying their local characteristics.

Equivalently, a PDMP can be specified by its infinitesimal generator. The infinitesimal generator for a PDMP is sometimes useful when one wants to prove results about them. Unfortunately, the limited domain of the (usual) generator is often too small to do this and we therefore define the *extended generator*. The extended generator $\mathfrak{A}$ has a larger domain than the infinitesimal generator $\mathcal{A}$, and coincides with $\mathcal{A}$ on $\mathcal{D}(\mathcal{A})$. If results can be proven with the extended generator, they will carry over to the process we are interested in. The definition of the extended generator is now given.

Definition 2.4.2. *The extended generator $\mathfrak{A}$ is an operator which acts on functions $f : E \to \mathbb{R}$ as $\mathfrak{A}f$. Its domain $\mathcal{D}(\mathfrak{A})$ is the set of functions f for which $\mathfrak{A}f$ is integrable for every $x \in E$ and*

$$C_t^f = f(X_t) - f(X_0) - \int_0^t \mathfrak{A}f\mathrm{d}s$$

is a local martingale.

The extended generator of a PDMP with local characteristics $(\mathfrak{X}, \lambda, Q)$ is given in [2], and reads

$$\mathfrak{A}f(x) = \mathfrak{X}f(x) + \lambda(x)\int_E (f(y) - f(x))Q(x, \mathrm{d}y). \tag{2.11}$$

The first term describes deterministic dynamics and the second term describes jumps.

2.4.3 Jump-diffusion processes

Next, we consider processes with drift, diffusion, and jumps. The SDE is similar to the diffusion SDE (2.7) but with the addition of the jump integral. It reads

$$\mathrm{d}X_t = \mu(X_{t-})\mathrm{d}t + \sigma(X_{t-})\mathrm{d}W_t + \int_{\mathbb{R}^d} G(X_{t-}, y)N(\mathrm{d}t, \mathrm{d}y), \quad t > 0, X_0 = x_0, \tag{2.12}$$

and has generator

$$\mathcal{A}f(x) = \sum_i^d \mu_i \partial_i f(x) + \frac{1}{2}\sum_{i,j=1}^d (\sigma\sigma^\mathsf{T})_{i,j}\partial_i\partial_j f(x) + \Lambda\int_{\mathbb{R}^d}(f(x + G(x, y)) - f(x))\pi(\mathrm{d}y) \tag{2.13}$$

where π is the measure for the jump magnitude and Λ is the jump rate [5]. It can be shown that strong solutions for (2.12) can be obtained by interlacing solutions the inter-jump processes [5]. Interlacing is essentially the construction of a solution by solving the segments between jumps and then piecing together the small segments. Uniqueness of such solutions can also be shown.

It is an interesting exercise to compare the (extended) infinitesimal generator of a PDMP in (2.11) with the generator which arises from the jump-diffusion SDE here. First, we note that the drift part is identical. We also see that PDMPs are not diffusive processes as the diffusion term is missing. It is evident that both processes contain jumps. We see that PDMPs have a state-dependent rate in contrast to the jump-diffusion SDE which we have defined. Notice also that π in (2.13) can be written as a Markov kernel as $Q'(x, \mathrm{d}y) = \pi(\mathrm{d}y)$. Finally, consider the choice $G(x, y) = y - x$. With this choice, the integral in the generator for our SDE simply reads

$$\int_{\mathbb{R}^d}(f(y) - f(x))Q'(x, \mathrm{d}y),$$

which makes the comparison with the PDMP generator look even more similar. A PDMP can in fact be written on differential form, but the correct description requires some further details, and we refer the interested reader to [2].

2.5 Analytical and approximative SDE solutions

Proving existence and uniqueness of solutions to stochastic differential equations can be done under various assumptions, but finding explicit analytical solutions is only possible for a limited set of equations. One type of SDE which can be solved analytically is the linear type. We give the solution to one particular linear SDE

most relevant to this thesis in Section 2.5.1. While SDEs in many realistic cases are not linear, being able to solve the linear case exactly is useful for approximate schemes based on linearization. One such approach is outlined in Section 2.5.2. Here we also describe the Euler-Maruyama scheme, which is a standard algorithm to simulate an SDE.

2.5.1 Analytical solution to an important SDE

The simplest SDE is linear,

$$\mathrm{d}X_t = \mathbf{A}X_t\mathrm{d}t + B\mathrm{d}W_t,\ t > 0; \quad X_0 = x_0, \tag{2.14}$$

with drift matrix $\mathbf{A}$ and diffusion coefficient B. In the interest of solving an SDE of importance for this thesis, we consider the specific system of a particle moving in one dimension with random acceleration. The state is $X = (p, v)$ with p and v being position and velocity respectively. The SDE for this system reads

$$\mathrm{d}X_t = \mathbf{A}X_t\mathrm{d}t + B\mathrm{d}W_t, \quad \mathbf{A} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 1 \end{pmatrix},\ t > 0; \quad X_0 = x_0.$$

Let $Y_t = \exp(-\mathbf{A}t)X_t$. Since the matrix exponential follows the usual rules of differentiation we have

$$\mathrm{d}Y_t = -\mathbf{A}\exp(-\mathbf{A}t)X_t\mathrm{d}t + \exp(-\mathbf{A}t)\mathrm{d}X_t.$$

By substitution, the SDE for Y_t is reads

$$\mathrm{d}Y_t = \exp(-\mathbf{A}t)B\mathrm{d}W_t,$$

and after integration, we get the solution

$$Y_t = Y_0 + \int_0^t \exp(-\mathbf{A}s)B\mathrm{d}W_s.$$

We transform back to X (noticing $X_0 = Y_0$) and obtain

$$X_t = \exp(\mathbf{A}t)X_0 + \int_0^t \exp(\mathbf{A}(t-s))B\mathrm{d}W_s. \tag{2.15}$$

Here we recall the definition of the matrix exponential, $\exp(\mathbf{M}) = \sum_{i=1}^{\infty} \mathbf{M}^k/k!$, and make use of the fact that $\mathbf{A}$ is nilpotent[2] to show

$$\exp(\mathbf{A}(t-s)) = \begin{pmatrix} 1 & t-s \\ 0 & 1 \end{pmatrix}.$$

The integral term in (2.15) has a deterministic integrand and stochastic integrator and it is thus an Itô integral. Since the integrand is deterministic and linear, the solution to (2.15) is a Gaussian with mean $\exp(\mathbf{A}t)X_0$ and covariance

$$\mathrm{Cov}[X_t] = \mathbf{E}\left[\left(\int_0^t \exp(\mathbf{A}(t-s))B\mathrm{d}W_s\right)\left(\int_0^t \exp(\mathbf{A}(t-s))B\mathrm{d}W_s\right)^{\mathsf{T}}\right].$$

[2] A matrix $\mathbf{A}$ is nilpotent if $\mathbf{A}^k = \mathbf{0}$ for some $k \in \mathbb{N}_{\not<}$.

The expected value is calculated using the Itô isometry [4],

$$\begin{aligned}\mathrm{Cov}[X_t] &= \mathbf{E}\left[\int_0^t (\exp(\mathbf{A}(t-s))B)(\exp(\mathbf{A}(t-s))B)^\mathsf{T}\mathrm{d}s\right] \\ &= \int_0^t \left(\begin{pmatrix}1 & t-s \\ 0 & 1\end{pmatrix}\begin{pmatrix}0 \\ 1\end{pmatrix}\right)\left(\begin{pmatrix}1 & t-s \\ 0 & 1\end{pmatrix}\begin{pmatrix}0 \\ 1\end{pmatrix}\right)^\mathsf{T}\mathrm{d}s \\ &= \int_0^t \begin{pmatrix}(t-s)^2 & t-s \\ t-s & 1\end{pmatrix}\mathrm{d}s \\ &= \begin{pmatrix}t^3/3 & t^2/2 \\ t^2/2 & t\end{pmatrix}.\end{aligned}$$

The full solution (in distribution) for X is thus given as

$$X_t \sim \mathcal{N}\left(\begin{pmatrix}1 & t \\ 0 & 1\end{pmatrix}x_0, \begin{pmatrix}t^3/3 & t^2/2 \\ t^2/2 & t\end{pmatrix}\right), \quad t \geq 0,$$

where the initial condition $X_0 = x_0$ has been used. The mean of the solution is linear in t, i.e., the particle mean moves with constant velocity given by x_0. This is expected as the stochastic force acting on the particle has an expected value of 0. We also note that the covariance grows faster for the higher-order variable, the position. Of course, in practice not all SDEs are linear, and in those cases, we often need to resort to numerical approximations.

2.5.2 Numerical approximations

In many realistic cases analytical solutions to SDEs cannot be readily obtained. Instead, one must resort to approximations. In the following, two approaches are explained. The first is the Euler-Maruyama method. The Euler-Maruyama method can be used to solve an SDE to arbitrary accuracy, given a noise realization. However, it can not be used to calculate the covariance of the solution without repeated experiments of different noise realizations. The second method is based on linearization around the corresponding ODE obtained by removing the noise. This method gives an approximation of the covariance matrix that we need and is incredibly fast. However, there is no discretization parameter, which means that the method cannot be made arbitrarily accurate. It should be noted that there are plenty of other numerical schemes for solving the SDE available, [6] provides a good reference.

2.5.2.1 The Euler-Maruyama scheme

The *Euler-Maruyama* (EM) method presents a way to simulate realizations of the process defined by an SDE. The method is analogous to the explicit Euler method for ODEs. A suitably small time discretization parameter δt is selected, and the differential equation is then solved incrementally with the noise term simulated from a random number generator. Given the state $X_0 = x_0$ at time t_0, the state X_N at time $t_0 + N\delta t$ is calculated recursively as

$$X_{n+1} = X_n + \mu(X_n)\delta t + \sigma(X_n)\epsilon_{\delta t}, \quad n = 0, 1, 2, \ldots, N-1$$

where $\epsilon_{\delta t} \sim \mathcal{N}(0_d, \delta t I_d)$. The SDE is thus simulated by using small time increments and doing the approximation of freezing the coefficient with their left side values (at X_n). In order to obtain an (empirical) distribution of the state at some later time T, a large number of such simulations must be performed. This is not always feasible in applications for efficiency reasons and some other method must therefore be employed.

2.5.2.2 A noise linearization approximation

For the inference task defined in the next chapter, it is evident that we need a way to solve a nonlinear SDE quickly. We therefore introduce an approach to find an approximate solution to an SDE which is very fast. We refer to it as the noise linearization method or approximation. First, the stochastic part of the SDE (2.7) is omitted, resulting in the ODE

$$\mathrm{d}\hat{X}_t = \mu(\hat{X}_t)\mathrm{d}t,\ t > 0; X_0 = x_0. \tag{2.16}$$

This ODE can for example be solved numerically with a Runge-Kutta method. We define $\Delta_t = X_t - \hat{X}_t$, and consider the process (2.7) minus (2.16),

$$\mathrm{d}\Delta_t = \Big(\mu(X_t) - \mu(\hat{X}_t)\Big)\mathrm{d}t + \sigma(X_t)\mathrm{d}W,\ t > 0; \Delta_0 = 0.$$

By Taylor expansion of μ around the ODE solution $\hat{X}$ we have that

$$\mathrm{d}\Delta_t = \Big(\mathbf{J}_\mu(\hat{X}_t)\Delta_t + \mathcal{O}((X_t - \hat{X}_t)^2)\Big)\mathrm{d}t + \sigma(X_t)\mathrm{d}W.$$

Here $\mathbf{J}_\mu$ denotes the Jacobian of μ. In the next step we replace $\mathbf{J}_\mu(\hat{X}_t)$ with $\mathbf{J}_\mu(X_0)$ and $\sigma(X_t)$ with $\sigma(X_0)$ and discard the remainder. This results in a new, linear SDE

$$\mathrm{d}\tilde{\Delta}_t = \mathbf{J}_\mu(X_0)\tilde{\Delta}_t\mathrm{d}t + \sigma(X_0)\mathrm{d}W$$

which is on the form of (2.14). This SDE is solved analytically, following the steps in 2.5.1. Since $\tilde{\Delta}_0 = 0$, the expected value of this SDE is 0 for all $t \geq 0$.

We remark that the matrix $\mathbf{A} = \mathbf{J}_\mu(X_0)$ may not be nilpotent in many realistic cases. For example, $\mathbf{A}$ is not nilpotent when one of the variables couples to itself. In such cases, the exponential can be calculated by diagonalization of $\mathbf{A} = \mathbf{C}\mathbf{D}\mathbf{C}^{-1}$ where $\mathbf{D}$ is a diagonal matrix,

$$\exp(\mathbf{A}) = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k}{k!} = \mathbf{C}\exp(\mathbf{D})\mathbf{C}^{-1}.$$

If the matrix is not diagonalizable, similar results can sometimes be obtained by considering the Jordan Matrix decomposition $\mathbf{A} = \mathbf{S}\mathbf{J}\mathbf{S}^{-1}$. Finally, the approximated solution to X is given by

$$X_t \approx \hat{X}_t + \tilde{\Delta}_t, \quad t \geq 0,$$

where $\hat{X}$ is obtained using some numerical ODE solver. Since $\mathbf{E}[\tilde{\Delta}] = 0$ for all $t \geq 0$, the mean is given purely from $\hat{X}$, while the covariance is given purely from $\tilde{\Delta}$. Finally, we remark this method is only accurate for either small times or small noise.

3

Bayesian inference for SDEs

The main topic of this thesis is that of fast parameter estimation for stochastic differential equations, i.e., sampling posterior distributions for parameters that govern the dynamics of an SDE. Many common algorithms designed to sample posterior distributions are based on the simulation of discrete-time Markov processes, i.e., Markov chains. These algorithms rely on Markov processes which admit the posterior distribution as their invariant measure. The Metropolis-Hastings algorithm (MH) is probably the most well-known algorithm in this category. Metropolis-Hastings is simple to implement and can often be an efficient sampling scheme, however, it can be difficult to select the proposal distribution in a way that achieves efficient sampling [7]. Other popular samplers based on discrete-time Markov chains include Hamiltonian Monte-Carlo [7], [8] and the Gibbs sampler [9].

In this thesis, we employ PDMP samplers, which are a class of samplers based on the simulation of PDMPs that admit the posterior distribution as its invariant measure. These samplers differ from traditional samplers in that they sample the posterior in continuous time rather than discrete time. PDMP-samplers are *non-reversible* and have *momentum*, which are properties that sometimes give them favorable convergence properties [1]. In contrast to MH, there is no proposal distribution that needs to be designed as the dynamics of a PDMP are well defined by the posterior measure alone.

In Sections 3.1–3.3 three different PDMPs are introduced, the Zig-Zag sampler, the sticky Zig-Zag sampler, and the bouncy particle sampler. In Section 3.4 we describe our inference task generally in relation to an arbitrary target and sensor model. Here we also present the unscented Kalman filter, which is used for likelihood evaluation.

3.1 The Zig-Zag sampler

The Zig-Zag sampler (ZZ) is a sampler based on the simulation of a Zig-Zag process. Formally, we differentiate between the Zig-Zag sampler, which is the algorithm used to sample from a probability measure, and the Zig-Zag process, which is the stochastic process upon which the sampler is based. Sometimes these terms are used interchangeably. The Zig-Zag process is aptly named since the paths produced by it look like it "zig-zags" around in the sample space. An example of this is shown in Figure 3.1.

Let us now give a detailed introduction to the Zig-Zag sampler. We do this following

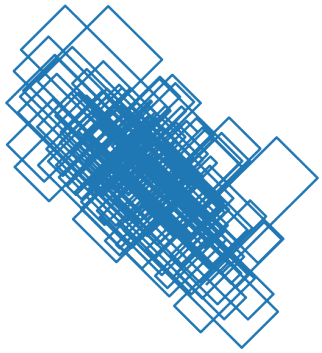


Figure 3.1: An example realization of the Zig-Zag process, sampling from a Gaussian distribution in two dimensions.

the presentation in [1] giving relevant definitions, propositions, and proofs, where they contribute towards understanding. The objects defined in this presentation are then related to the local characteristics of the PDMP. For other PDMPs, we are satisfied by giving the definitions and some intuition about their dynamics.

Define the state space $E = \mathbb{R}^d \times \{-1, +1\}^d$ and denote elements $(\xi, \theta) \in E$ where $\xi \in \mathbb{R}^d$ and $\theta \in \{-1, +1\}^d$. In the context of sampling from a probability distribution, ξ represents the states that are being sampled while θ represents a velocity vector. Each variable ξ_i has thus been equipped with a velocity θ_i. We denote the PDMP as $(\Xi_t, \Theta_t)_{t \geq 0}$.

As explained in Section 2.4.2.2, a PDMP consists of two main components, deterministic dynamics, and jumps. The deterministic dynamics of the Zig-Zag process is that of moving ξ in the direction of the velocity vector θ. This is summarized by a set of ODEs

$$\mathrm{d}\xi_i = \theta_i \mathrm{d}t, \quad i = 1, \ldots, d. \tag{3.1}$$

At discrete and random times, the process jumps. A jump corresponds to a velocity flip for one of the state variables, defined formally with the functions $F_1, \ldots, F_d$,

$$(F_i[\theta])_j = \begin{cases} \theta_j & i \neq j, \\ -\theta_j & i = j. \end{cases}$$

The rate at which these event occur are specified by d inhomogeneous Poisson processes with rates $\lambda_1(\xi, \theta), \ldots, \lambda_d(\xi, \theta)$. These rates are functions of both state and velocity and must be selected in such a way as to achieve the correct invariant distribution for the process. They are thus related to the desired invariant measure that we wish to sample from. If the desired invariant measure is μ, then we define $\Psi(\xi) = -\log(\mu(\xi))$. Proposition 3.1.1 relates the event rates λ_i to Ψ. It is important for the proof of Theorem 3.1.1.

Proposition 3.1.1. *Suppose that* $\Psi \in C^1(\mathbb{R}^d)$ *satisfies*

$$\int_{\mathbb{R}^d} \exp(-\Psi(\xi)) \mathrm{d}\xi < \infty. \tag{3.2}$$

A continuous function $\lambda : E \to \mathbb{R}_+^d$ *satisfies*

$$\lambda_i(\xi,\theta) - \lambda_i(\xi, F_i[\theta]) = \theta_i \partial_{\xi_i}\Psi(\xi) \text{ for all } (\xi,\theta) \in E, i = 1,\dots,d \tag{3.3}$$

if and only if there exists a positive function $\gamma : E \to \mathbb{R}_+^d$ *such that for all* i *and* $(\xi,\theta) \in E$, $\gamma_i(\xi,\theta) = \gamma_i(\xi, F_i[\theta])$ *and*

$$\lambda_i(\xi,\theta) = (\theta_i \partial_{\xi_i}\Psi(\xi))^+ + \gamma_i(\xi,\theta). \tag{3.4}$$

The function γ is not unique and is in practise usually set to 0, as that is expected to give the fastest convergence [1]. The rates λ_i with $\gamma_i = 0$ are called the *canonical rates*. The proof for Proposition 3.1.1 is now given.

Proof of Proposition 3.1.1. Assume the existence of γ satisfying (3.4). Relation (3.3) is verified by substitution, more precisely

$$\begin{aligned}
\lambda_i(\xi,\theta) - \lambda_i(\xi, F_i[\theta]) &= \left[(\theta_i \partial_{\xi_i}\Psi(\xi))^+ + \gamma_i(\xi,\theta)\right] - \left[(F_i[\theta_i]\partial_{\xi_i}\Psi(\xi))^+ + \gamma_i(\xi, F_i[\theta])\right] \\
&= (\theta_i \partial_{\xi_i}\Psi(\xi))^+ - (-\theta_i \partial_{\xi_i}\Psi(\xi))^+ \\
&= (\theta_i \partial_{\xi_i}\Psi(\xi))^+ - (\theta_i \partial_{\xi_i}\Psi(\xi))^- \\
&= \theta_i \partial_{\xi_i}\Psi(\xi).
\end{aligned}$$

To prove the converse, let $\lambda : E \to \mathbb{R}_+^d$ be a function that satisfies (3.3). Define the function

$$\gamma_i(\xi,\theta) = \lambda_i(\xi,\theta) - (\theta_i \partial_{\xi_i}\Psi(\xi))^+ \tag{3.5}$$

for $i = 1,\dots,n$ and $(\xi,\theta) \in E$. This trivially satisfies (3.4). We now show that with this definition $\gamma_i(\xi,\theta) - \gamma_i(\xi, F_i[\theta]) = 0$ for all ξ and θ. To prove this, we use (3.3) and (3.5). We see that

$$\begin{aligned}
\gamma_i(\xi,\theta) - \gamma_i(\xi, F_i[\theta]) &= \left[\lambda_i(\xi,\theta) - (\theta_i \partial_{\xi_i}\Psi(\xi))^+\right] - \left[\lambda_i(\xi, F_i[\theta]) - (F_i[\theta_i]\partial_{\xi_i}\Psi(\xi))^+\right] \\
&= \lambda_i(\xi,\theta) - \lambda_i(\xi, F_i[\theta]) - \left[(\theta_i \partial_{\xi_i}\Psi(\xi))^+ - (-\theta_i \partial_{\xi_i}\Psi(\xi))^-\right] \\
&= \lambda_i(\xi,\theta) - \lambda_i(\xi, F_i[\theta]) - \theta_i \partial_{\xi_i}\Psi(\xi) \\
&= 0.
\end{aligned}$$

We need to show that γ is positive, i.e., that $\gamma : E \to \mathbb{R}_+^d$. To do so, pick any $(\xi,\theta) \in E$. Suppose that $\gamma_i(\xi,\theta) < 0$ and $\theta_i \partial_{\xi_i}\Psi(\xi) \le 0$. Then

$$\lambda_i(\xi,\theta) = \gamma_i(\xi,\theta) + \underbrace{(\theta_i \partial_{\xi_i}\Psi(\xi))^+}_{=0} < 0,$$

which is a contradiction since $\lambda_i(\xi,\theta) \ge 0$. Now suppose $\theta_i \partial_i \Psi(\xi) > 0$. Then

$$\begin{aligned}
\lambda_i(\xi, F_i[\theta]) &= \gamma_i(\xi, F_i[\theta]) + (F_i[\theta_i]\partial_{\xi_i}\Psi(\xi))^+ \\
&= \gamma_i(\xi,\theta) + \underbrace{(\theta_i \partial_{\xi_i}\Psi(\xi))^-}_{=0} < 0.
\end{aligned}$$

This is also a contradiction. Thus, $\gamma_i(\xi,\theta) \ge 0$ for all $(\xi,\theta) \in E$, which concludes the proof. □

Next, we present Theorem 3.1.1, which guarantees that the Zig-Zag process with rates λ_i admits the correct invariant distribution. We let μ_0 denote the Lebesgue measure on E, satisfying $\mathrm{d}\mu_0(\xi,\theta) = \mathrm{d}\xi$.

Theorem 3.1.1. *Suppose that $\Psi \in C^1(\mathbb{R}^d)$ satisfies (3.2) and that $\lambda : E \to \mathbb{R}^d_+$ satisfies (3.3). Assume further that μ is a probability measure on E such that*

$$\frac{\mathrm{d}\mu}{\mathrm{d}\mu_0}(\xi,\theta) = \frac{\exp(-\Psi(\xi))}{\int_E \exp(-\Psi(x))\mathrm{d}x}, \quad (\xi,\theta) \in E. \tag{3.6}$$

Then the Zig-Zag process $(\Xi_t, \Theta_t)_{t\geq 0}$ has invariant distribution μ.

Theorem 3.1.1 directly relates μ to Ψ by a Radon-Nikodym derivative (see Appendix A.1). It states that the Zig-Zag process with switching rates λ_i, as defined in Proposition 3.1.1, indeed has μ as its invariant distribution. The condition that $\Psi \in C^1(\mathbb{R}^d)$ places some restrictions on the type of distributions that can be sampled. For example, uniform distributions seem to pose problems since they have discontinuities at their ends. However, in practice, these problems can often be overcome as is explained in Section 4.4.

In order to give the proof for Theorem 3.1.1, the extended generator of the Zig-Zag process is needed. It is given in [10] and reads

$$\mathcal{L}f(\xi,\theta) = \sum_{i=1}^{d} \Big(\theta_i \partial_{\xi_i} f(\xi,\theta) + \lambda_i(\xi,\theta)\Big(f(\xi, F_i[\theta]) - f(\xi,\theta)\Big)\Big), \quad (\xi,\theta) \in E. \tag{3.7}$$

Proof of Theorem 3.1.1. Referring to Theorem 2.2.1, we can prove the correct invariant measure by showing that $\int_E \mathcal{L}f \mathrm{d}\mu = 0$. The generator for the Zig-Zag process can by linearity be written as $\mathcal{L} = L_1 + \cdots + L_d$, and we only need to show $\int_E L_i f \mathrm{d}\mu = 0$ for one of these. We first use (3.6) to obtain

$$\int_E L_i f(\xi,\theta)\mathrm{d}\mu(\xi,\theta) = \sum_{\theta\in\{-1,+1\}^d} \int_{\mathbb{R}^d} L_i f(\xi,\theta)\frac{1}{Z}\exp(-\Psi(\xi))\mathrm{d}\xi.$$

By the definition of the generator (3.7) we have

$$\frac{1}{Z}\sum_{\theta\in\{-1,+1\}^d} \int_{\mathbb{R}^d} \Big[\theta_i \partial_{\xi_i} f(\xi,\theta) + \lambda_i(\xi,\theta)\Big(f(\xi,F_i[\theta]) - f(\xi,\theta)\Big)\Big]\exp(-\Psi(\xi))\mathrm{d}\xi. \tag{3.8}$$

Let us first consider the second term. We note that the sum can be taken inside the integral and that the summation over θ can be written as a sum of *pairs*, $(\theta, F_i[\theta]) \in \{-1,+1\}^d \times \{-1,+1\}^d$. In other words, both θ and $F_i[\theta]$ are included in the sum in (3.12), and treating these together turns out to be useful. Such a pair reads

$$\lambda_i(\xi,\theta)\Big(f(\xi,F_i[\theta]) - f(\xi,\theta)\Big) + \lambda_i(\xi,F_i[\theta])\Big(f(\xi,\theta) - f(\xi,F_i[\theta])\Big).$$

By Proposition 3.1.1 and the trick $\max(0,-a)-\max(0,a)=-a$, we have

$$\begin{aligned}
&\lambda_i(\xi,\theta)\Big(f(\xi,F_i[\theta])-f(\xi,\theta)\Big)+\lambda_i(\xi,F_i[\theta])\Big(f(\xi,\theta)-f(\xi,F_i[\theta])\Big)\\
&\quad=\Big((\theta_i\partial_{\xi_i}\Psi(\xi))^+ +\gamma(\xi,\theta))\Big)\Big(f(\xi,F_i[\theta])-f(\xi,\theta)\Big)\\
&\qquad+\Big((-\theta_i\partial_{\xi_i}\Psi(\xi))^+ +\gamma(\xi,F_i[\theta]))\Big)\Big(f(\xi,\theta)-f(\xi,F_i[\theta])\Big)\\
&\quad=\Big[(\theta_i\partial_{\xi_i}\Psi(\xi))^+-(-\theta_i\partial_{\xi_i}\Psi(\xi))^+\Big]f(\xi,F_i[\theta])\\
&\qquad+\Big[-(\theta_i\partial_{\xi_i}\Psi(\xi))^+ +(-\theta_i\partial_{\xi_i}\Psi(\xi))^+\Big]f(\xi,\theta)\\
&\quad=(\theta_i\partial_{\xi_i}\Psi(\xi))f(\xi,F_i[\theta])\\
&\qquad+((F_i[\theta])_i\partial_{\xi_i}\Psi(\xi))^+f(\xi,\theta)\\
&\quad=\Big(\lambda(\xi,\theta)-\lambda(\xi,F_i[\theta])\Big)f(\xi,F_i[\theta])\\
&\qquad+\Big(\lambda(\xi,F_i[\theta])-\lambda(\xi,\theta)\Big)f(\xi,\theta).
\end{aligned}\tag{3.9}$$

Inside the sum over θ, we have the identity

$$\lambda_i(\xi,\theta)\Big(f(\xi,F_i[\theta])-f(\xi,\theta)\Big)=\Big(\lambda(\xi,F_i[\theta])-\lambda(\xi,\theta)\Big)f(\xi,\theta),\tag{3.10}$$

by symmetry of the terms in (3.9). By use of partial integration on the first term and (3.10) on the second term in (3.8) we obtain

$$\frac{1}{Z}\sum_{\theta\in\{-1,+1\}^d}\int_{\mathbb{R}^d}\Big[\theta_i\partial_{\xi_i}\Psi(\xi,\theta)+\lambda_i(\xi,F_i[\theta])-\lambda_i(\xi,\theta)\Big]f(\xi,\theta)\exp(-\Psi(\xi))\mathrm{d}\xi,$$

which is 0 by (3.3). The integral $\int_E \mathcal{L}f\mathrm{d}\mu$ is therefore 0, which means that μ is an invariant measure. □

The final result needed is that of ergodicity. Ideally one would like to prove ergodicity for the canonical rates $\gamma=0$, however, this turns out to be difficult. We can intuitively understand why $\gamma>0$ makes ergodicity easier to prove. If the rate is always positive, then the process for any $(\xi,\theta)\in E$ has a nonzero probability to transition to any velocity $\theta\in\{-1,+1\}^d$. The following theorem establishes ergodicity under the simplifying assumption $\gamma>0$.

Theorem 3.1.2. *Assume $\lambda_i(\xi,\theta)>0$ for all $i=1,\dots,d$ and $(\xi,\theta)\in E$. Then the Zig-Zag process has at most one invariant distribution.*

If the invariant distribution is unique, then there is no way that the same process can converge to different distributions. For the proof of Theorem 3.1.2 we refer to the supplementary material in [1]. For an ergodicity proof under weaker assumptions, see [10].

To finalize the introduction to the Zig-Zag process, we relate the above concepts to the local characteristics of the Zig-Zag process. The deterministic dynamics of the process given in (3.1) which is equivalently specified using the vector field as

$$\mathfrak{X}=\sum_{i=1}^d\theta_i\partial_{\xi_i}.$$

The intensity λ satisfies

$$\lambda(\xi, \theta) = \sum_{i=1}^{d} \lambda_i(\xi, \theta),$$

and the kernel Q is given by

$$Q((\xi, \theta), \mathrm{d}x) = \sum_{i=1}^{d} \frac{\lambda_i(\xi, \theta)}{\lambda(\xi, \theta)} Q_i^{\mathrm{refl}}((\xi, \theta), \mathrm{d}x).$$

Here, $Q_i^{\mathrm{refl}}((\xi, \theta), \mathrm{d}x) = \delta_{(\xi, F_i(\theta))}(\mathrm{d}x)$ which means that the Markov kernel Q is not stochastic, but corresponds to deterministic velocity flips for the Zig-Zag process.

3.2 The sticky Zig-Zag sampler

One of the shortcomings of the Zig-Zag sampler and other standard PDMP samplers is that they fail to sample measures that contain atoms, i.e., measures where some points have a positive probability. *Sticky* samplers are intended to solve this problem and are designed to sample from measures of the form

$$\mu(\mathrm{d}\xi) = C \exp(-\Psi(\xi)) \prod_{i=1}^{d} \left(\mathrm{d}\xi_i + \frac{1}{\kappa_i} \delta_0(\mathrm{d}\xi_i) \right), \tag{3.11}$$

where $C > 0$ and $\kappa_1, \ldots, \kappa_d > 0$ are constants, Ψ is some integrable and differentiable function, and δ_0 is a Dirac measure in zero. This measure can arise in the context of Bayesian inference if a *spike-and-slab* type prior

$$u(\mathrm{d}\xi) = \prod_{i=1}^{d} (w_i \pi_i(\xi_i) \mathrm{d}\xi_i + (1 - w_i) \delta_0(\mathrm{d}\xi_i))$$

is used. The prior u consists of continuous probability measures π_i (slabs) and Dirac measures in zero (spikes). The weights $w_i \in [0, 1]$ specify the probability that each variable ξ_i is nonzero. Each atom is here centered at 0 to simplify notation, but generalizations to non-centered atoms are trivial. In this context some of the quantities in (3.11) are known, specifically Ψ is given as

$$\Psi(\xi) = C_\Psi - \log(\ell(\xi)) - \sum_{i=1}^{d} \log(\pi_i(\xi_i)),$$

where $\ell(\xi)$ is the likelihood and C_Ψ is a normalization constant. The constants κ_i are given as

$$\kappa_i = \frac{w_i}{1 - w_i} \pi_i(0), \quad i = 1, \ldots, d,$$

given that the atoms are located at 0 [11].

In order to sample from (3.11), a process that has it as its invariant measure must be constructed. The authors of [11] show that this can be done with a PDMP if *stickiness* is introduced. The main idea of sticky PDMPs is to freeze variables for an exponentially distributed time whenever they hit an atom. This way, significant

probability mass can be ascribed to a single number. While a variety of PDMP samplers can be made sticky, we focus on the sticky Zig-Zag sampler (SZZ).

We start by defining the state space of the sticky Zig-Zag sampler. Define $\bar{\mathbb{R}} = (-\infty, 0^-] \cup [0^+, \infty)$ where we distinguish between the left and right limits of 0. The state space of the sticky Zig-Zag sampler is $E = \bar{\mathbb{R}}^d \times \{-1, +1\}^d$ with states $(\xi, \theta) \in E$, where ξ and θ are the positions and velocities of the particles respectively. When the process reaches a 0 in a variable ξ_i from either the left or the right, that variable is *frozen* for an exponentially distributed time $\sim \text{Exp}(\kappa_i)$. During this time, the other variables continue their dynamics as before (unless they also encounter freezing events). When ξ_i unfreezes, it continues with the same velocity as before, but on the other side of the 0. The distinction between the negative and positive 0's are required to keep the process Markovian. If there was only one 0, there would be no way of differentiating between a variable that is frozen, and one which was just unfrozen. Keep in mind that (in the theoretical formulation) the velocity component of the variable must remain unchanged during the freezing, otherwise one would need to look in the past to find its velocity when resuming the dynamics.

Let us introduce some notation, following that of [11]. Denote by $\mathfrak{F}_i$ the set of states with ξ_i being in the frozen state,

$$\mathfrak{F}_i = \{(\xi, \theta) \in E : \xi_i = 0^-, \theta_i = 1 \text{ or } \xi_i = 0^+, \theta_i = -1\}.$$

Let $\alpha(\xi, \theta)$ be the index set which denotes the indices of the *active* (that is, non-frozen) variables,

$$\alpha(\xi, \theta) = \{i \in \{1, \dots, d\} : (\xi, \theta) \notin \mathfrak{F}_i\}.$$

The frozen variables are denoted similarly as $\alpha(\xi, \theta)^\mathsf{c} = \{1, \dots, d\} \setminus \alpha(\xi, \theta)$. Define the map $T_i : \mathfrak{F}_i \to E$ which unfreezes the state $(\xi, \theta) \in \mathfrak{F}_i$ as

$$T_i(\xi, \theta) = \begin{cases} (\xi[i : 0^+], \theta) & \text{if } \xi_i = 0^-, \theta_i = +1, \\ (\xi[i : 0^-], \theta) & \text{if } \xi_i = 0^+, \theta_i = -1, \end{cases}$$

where $\xi[i : y]$ is convenient notation to replace the i'th index of ξ with y,

$$(\xi[i : y])_k = \begin{cases} \xi_i & \text{if } i \neq k, \\ y & \text{if } i = k. \end{cases}$$

For the sticky Zig-Zag sampler, there are two relevant Markov kernels. The first is the velocity reflection kernel Q^{refl}, which remains unchanged from the ordinary Zig-Zag process. The second is unique to the sticky variant and is the unfreezing kernel Q^{unfr}. It corresponds to a Dirac measure at $T_i(\xi, \theta)$. The corresponding Poisson clock has intensity κ_i. In order to characterize the sticky Zig-Zag process by its local characteristics, write

$$\lambda(\xi, \theta) = \lambda^{\text{refl}}(\xi, \theta) + \lambda^{\text{unfr}}(\xi, \theta),$$

where $\lambda^{\text{refl}}(\xi, \theta) = \sum_{i \in \alpha(\xi, \theta)} \lambda_i^{\text{refl}}(\xi, \theta)$ with $\lambda_i^{\text{refl}}(\xi, \theta) = (\theta_i \partial_{\xi_i} \Psi(\xi, \theta))^+$ is the reflection rate and $\lambda^{\text{unfr}}(\xi, \theta) = \sum_{i \in \alpha(\xi, \theta)^\mathsf{c}} \lambda_i^{\text{unfr}}$ with $\lambda_i^{\text{unfr}} = \kappa_i$ is the unfreezing rate. The full

Markov kernel Q can then be summarized as

$$Q((\xi,\theta),\mathrm{d}x) = \sum_{i\in\alpha(\xi,\theta)} \frac{\lambda_i^{\mathrm{refl}}(\xi,\theta)}{\lambda(\xi,\theta)} Q_i^{\mathrm{refl}}((\xi,\theta),\mathrm{d}x) + \sum_{i\in\alpha(\xi,\theta)^{\mathsf{c}}} \frac{\lambda_i^{\mathrm{unfr}}(\xi,\theta)}{\lambda(\xi,\theta)} Q_i^{\mathrm{unfr}}((\xi,\theta),\mathrm{d}x).$$

The vector field $\mathfrak{X}$ is almost the same as for the regular Zig-Zag, with the difference that frozen coordinates remain, well, frozen.

$$\mathfrak{X} = \sum_{i\in\alpha(\xi,\theta)} \theta_i \partial_{\xi_i}.$$

The dynamics of the sticky Zig-Zag process can thus be summarized as:

1. Until an event happens, the process moves along deterministic continuous paths specified by its velocity θ. Frozen coordinates (those in $\alpha(\xi,\theta)^{\mathsf{c}}$) do not move during this time.
2. If a coordinate i reaches 0 during the deterministic dynamics, it gets frozen and added to frozen set $\alpha(\xi,\theta)^{\mathsf{c}}$.
3. Events happen according to d independent Poisson clocks. Each variable has its own clock, which is either of the unfreeze or reflection type. When an event is generated from one of the clocks, the state is updated in accordance with the corresponding Markov kernel, which is either an unfreezing event or a reflection event.

3.3 The bouncy particle sampler

The bouncy particle sampler (BPS), is a PDMP sampler which is similar to the Zig-Zag sampler in that the deterministic dynamics are given by a constant velocity. However, in contrast to the Zig-Zag sampler, the bouncy particle sampler only has one Poisson clock specifying the reflection times and at the reflection times the velocity is reflected against the gradient of the likelihood. This means that while there is only one Poisson clock for the reflections instead of d, the full gradient needs to be evaluated, which makes that operation d times more expensive. For this reason, performance comparisons between the Zig-Zag sampler and the bouncy particle sampler are interesting. We now define the bouncy particle process in terms of its local characteristics.

Define the state space $E = \mathbb{R}^d \times \mathbb{R}^d$ with elements $(\xi,\theta) \in E$ corresponding to states and velocities respectively. The deterministic dynamics are identical to the Zig-Zag process, i.e.,

$$\mathfrak{X} = \sum_{i=1}^{d} \theta_i \partial_{\xi_i}$$

which corresponds to moving with constant velocity θ such that $\mathrm{d}\xi = \theta \mathrm{d}t$. As before, there is a Poisson clock with inhomogeneous intensity

$$\lambda_{\mathrm{bounce}}(\xi,\theta) = (\langle \nabla_\xi \Psi(\xi,\theta), \theta \rangle)^+,$$

which defines the rate of bounce events. The Markov kernel for a bounce is specified as $Q_{\text{bounce}}((\xi,\theta),\mathrm{d}x) = \delta_{(\xi,R(\xi,\theta))}(\mathrm{d}x)$, where

$$R(\xi,\theta) = \theta - 2\frac{\langle \nabla_\xi \Psi(\xi,\theta),\theta\rangle}{\|\Psi(\xi,\theta)\|^2}\Psi(\xi,\theta).$$

The geometric interpretation of R is that of reflecting the velocity θ against the gradient of Ψ. While the PDMP described above can be shown to have the correct invariant distribution, it is not ergodic. This can be seen in Figure 3.2. The process does converge to an invariant measure, but it can clearly be seen that it is not the Gaussian measure intended as the process never visits the center of the distribution. In order to rectify this situation, *refreshment* events are introduced. In general,

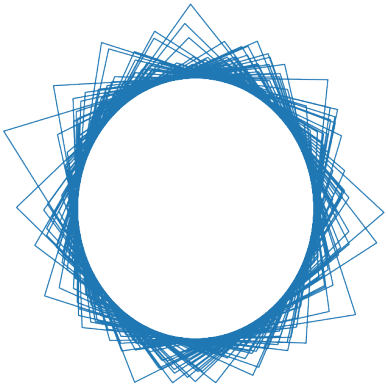


Figure 3.2: Sampling of a Gaussian distribution in two variables without velocity refreshments. The process is clearly not ergodic.

there is a variety of ways to define refreshments. For example, one could modify the bounce kernel and introduce some randomness there, as is done in [12]. The standard refreshment scheme is defined as follows. Introduce a second Poisson clock with a constant rate λ_{ref}. Each time a refreshment event is generated, the velocity component of the state is refreshed, according to a Markov kernel $Q_{\text{ref}}((\xi,\theta),\mathrm{d}x) = (\delta_\xi, \mathcal{N}(0_d, I_d))(\mathrm{d}x)$. The total rate is thus

$$\lambda(\xi,\theta) = \lambda_{\text{bounce}}(\xi,\theta) + \lambda_{\text{ref}}$$

with corresponding Markov kernel

$$Q((\xi,\theta),\mathrm{d}x) = \frac{\lambda_{\text{bounce}}(\xi,\theta)}{\lambda(\xi,\theta)}Q_{\text{bounce}}((\xi,\theta),\mathrm{d}x) + \frac{\lambda_{\text{ref}}}{\lambda(\xi,\theta)}Q_{\text{ref}}((\xi,\theta),\mathrm{d}x).$$

3.4 Bayesian filtering and the marginal posterior

In this thesis, we are given a series of noisy measurements $Z_{0:N}$ of some underlying process, defined as an SDE. The goal is to sample from posteriors related to this

process. It could be latent parameters β of the SDE or the states $X_{0:N}$ themselves. Constructing a posterior for both, we obtain

$$\begin{aligned} p(X_{0:N}, \beta | Z_{0:N}) &\propto p(Z_{0:N} | X_{0:N}, \beta) p(X_{0:N}, \beta) \\ &= \underbrace{\Pi_{k=0}^{N} p(Z_k | X_k, \beta)}_{\text{Measurements}} \underbrace{\Pi_{k=1}^{N} p(X_k | X_{k-1}, \beta)}_{\text{Dynamics (very rich prior)}} \underbrace{p(X_0, \beta)}_{\text{Prior}}. \end{aligned} \tag{3.12}$$

Here, the state is defined for every measurement time. Sampling from this distribution can be challenging because the dimensionality grows without restriction as new measurements are made. For this reason, combined with the fact that we are mostly interested in latent parameters, we choose to marginalize out all states except the first one. This significantly reduces the dimensionality of the posterior, which after the marginalization reads

$$p(X_0, \beta | Z_{0:N}) \propto p(Z_{0:N} | X_0, \beta) p(X_0, \beta).$$

The trade-off for this is that the likelihood is now more expensive to calculate. One way to do it is with a *filtering algorithm*. One of the most well-known filters is the Kalman filter [13]. The Kalman filter is an iterative procedure based on two steps, the *prediction* and *update* steps. In the prediction step, an estimate of the current state is used to predict the next state, given some model dynamics. In our case, those dynamics are encoded as an SDE. In the update step, the prediction is blended with a noisy state measurement at this later time to produce a new state estimate. The blending between the prediction and measurement is done in matrix form with the *Kalman gain* K, which is constructed in a way that can be shown to be optimal for linear systems. Unfortunately, the Kalman filter only works well for linear systems. There are multiple extensions and variations of the Kalman filter designed to work better in nonlinear settings, and one of those is the *Unscented Kalman Filter* (UKF) presented next. An example of how a filter can be used for state estimation is shown in Figure 3.3.

3.4.1 The Unscented Kalman Filter

Given an estimate (prior) of the state X_i of some system at time i, the filtering task is to estimate the distribution for X_{i+1}, i.e., the state at time $i+1$, given a noisy measurement $Z_{i+1} = h(X_{i+1})$. In general, both X_i and X_{i+1} may have arbitrary distributions, and both system dynamics and the measurement function h may be nonlinear. The filtering problem is therefore often difficult to solve exactly and the UKF is one attempt to do so in an approximate manner.

The UKF is an approximate Bayesian filter based on the so-called *unscented transform*, which is an approximate way to propagate distributions through nonlinear functions. The unscented transform is used in both the prediction and update stages of the filter. In the prediction step, the unscented transform starts from the assumption that the prior $\tilde{X}_i$ is a Gaussian, i.e., $\tilde{X}_i \sim \mathcal{N}(\hat{X}_i, \hat{\mathbf{P}}_i)$. A set of so-called σ-points are then strategically selected from the distribution, which are deterministically fed through the nonlinearity. A new Gaussian $\tilde{X}_{i+1|i} \sim \mathcal{N}(\hat{X}_{i+1|i}, \hat{\mathbf{P}}_{i+1|i})$ is then fitted to the propagated σ-points. The idea behind this transformation is that

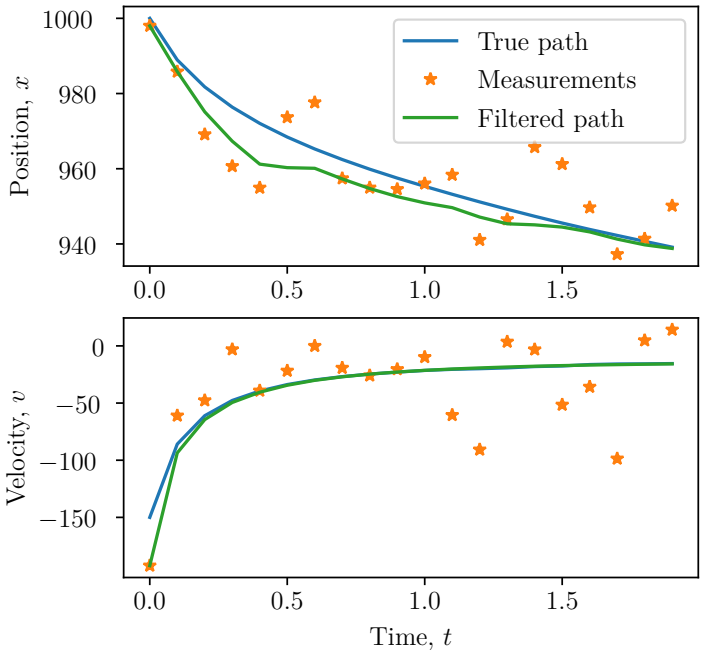


Figure 3.3: Filtering with the UKF.

a few strategically selected σ-points may be enough to capture the general behavior of the nonlinear function. Of course, this is not always the case.

In the update step, the unscented transform is used again. Here the prior $\tilde{X}_i$ is replaced with the prediction $\tilde{X}_{i+1|i}$ and the nonlinearity is now the measurement function instead of the system dynamics. We next present the unscented filtering algorithm step by step. The unscented Kalman filter can be constructed in a few slightly different ways, and the following presentation roughly follows [14].

Let $\tilde{X}_i \sim \mathcal{N}(\hat{X}_i, \hat{\mathbf{P}}_i)$ denote an estimate of state i. We use the tilde to remind ourselves that this is an approximation of the true state. The objective is to first calculate the prediction $\tilde{X}_{i+1|i}$, which is then used in the update step to obtain $\tilde{X}_{i+1|i+1}$. To calculate the prediction, consider $2d+1$ σ-points (d-dimensional vectors) $S_{-d}, \ldots, S_d$ where d is the state dimensionality. Let each σ-point have associated weights $w^a_{-d}, \ldots, w^a_d$ satisfying

$$1 = \sum_{j=-d}^{d} w_j^a,$$
$$\mathbf{E}[\tilde{X}_i] = \sum_{j=-d}^{d} w_j^a S_j,$$

and weights $w^c_{-d}, \ldots, w^c_d$ satisfying

$$1 = \sum_{j=-d}^{d} w_j^c,$$
$$\mathbf{E}[(\tilde{X}_i)_\ell (\tilde{X}_i)_m]] = \sum_{j=-d}^{d} w_k^c (S_j)_\ell (S_j)_m \quad \text{for all } \ell, m = 1, \ldots, d.$$

These restrictions still leave some freedom of choice for the σ-points. One alternative is to select them symmetrically as

$$\begin{aligned} S_0 &= \hat{X}_i, \\ S_{\pm j} &= S_0 \pm \alpha\sqrt{\kappa}\mathbf{L}_j, \quad j = 1, \ldots, d, \\ w_0^a &= \frac{\alpha\kappa - d}{\alpha^2\kappa}, \\ w_0^c &= w_0^a + 1 - \alpha^2 + \beta, \\ w_{\pm j}^a &= w_{\pm j}^c = \frac{1}{2\alpha^2\kappa}, \quad j = 1, \ldots, d, \end{aligned}$$

where α, κ and β are design parameters with (for example) values $\alpha = \kappa = 1$ and $\beta = 2$. The vector $\mathbf{L}_j$ is the j-th column of the Cholesky factorization of $\hat{\mathbf{P}}_i = \mathbf{L}\mathbf{L}^\mathsf{T}$.

With σ-points defined, the prediction stage can be performed. First, the σ-points are propagated through a nonlinearity, here denoted f. This function can take any form but often corresponds to the numerical solution of an ODE. Weighted sums are then computed to obtain the prediction for the next state $\tilde{X}_{i+1|i} \sim \mathcal{N}(\hat{X}_{i+1|i}, \hat{\mathbf{P}}_{i+1|i})$ with

$$\begin{aligned} X_{S_j} &= f(S_j), \quad j = -d, \ldots, d, \\ \hat{X}_{i+1|i} &= \sum_{j=-d}^{d} w_j^a X_{S_j}, \\ \hat{\mathbf{P}}_{i+1|i} &= \sum_{j=-d}^{d} w_j^c (X_{S_j} - \hat{X}_{i+1|i})(X_{S_j} - \hat{X}_{i+1|i})^\mathsf{T} + \mathbf{Q}_i, \end{aligned}$$

where $\mathbf{Q}_i$ is the covariance of the process noise.

In the update step, a new set of σ-points $S'_{-d}, \ldots, S'_d$ are created from $\tilde{X}_{i+1|i}$. These new σ-points are transformed to measurement space through the (possibly non-linear) measurement function h. The random variable that is obtained is called the *innovation* and represents the predicted measurement, given the previous state estimate. The innovation is $\tilde{Z}_{i+1|i} \sim \mathcal{N}(\hat{Z}_{i+1|i}, \hat{\mathbf{S}}_{i+1|i})$ with

$$\begin{aligned} Z_{S'_j} &= h(S'_j), \quad j = -d, \ldots, d, \\ \hat{Z}_{i+1|i} &= \sum_{j=-d}^{d} w_j^a Z_{S'_j}, \\ \hat{\mathbf{S}}_{i+1|i} &= \sum_{j=-d}^{d} w_j^c (Z_{S'_j} - \hat{Z}_{i+1|i})(Z_{S'_j} - \hat{Z}_{i+1|i})^\mathsf{T} + \mathbf{R}_{i+1}, \end{aligned}$$

where $\mathbf{R}_{i+1}$ is the covariance of the measurement noise. Next, the cross-covariance and Kalman gain are computed as

$$\begin{aligned} \mathbf{C} &= \sum_{j=-d}^{d} w_j^c (X_{S_j} - \hat{X}_{i+1|i})(Z_{S'_j} - \hat{Z}_{i+1|i})^\mathsf{T}, \\ \mathbf{K} &= \mathbf{C}(\hat{\mathbf{S}}_{i+1|i})^{-1}, \end{aligned}$$

and an estimate $\tilde{X}_{i+1|i+1} \sim \mathcal{N}(\hat{X}_{i+1|i+1}, \hat{\mathbf{P}}_{i+1|i+1})$ is finally obtained for the next state, with mean and covariance given as

$$\begin{aligned}\hat{X}_{i+1|i+1} &= \hat{X}_{i+1|i} + \mathbf{K}(Z_{i+1} - \hat{Z}_{i+1|i}),\\ \hat{\mathbf{P}}_{i+1|i+1} &= \hat{\mathbf{P}}_{i+1|i} - \mathbf{K}\hat{\mathbf{S}}_{i+1|i}\mathbf{K}^{\mathsf{T}},\end{aligned}$$

where Z_{i+1} is the measurement corresponding to time $i+1$.

Note that the filtering procedure does not utilize information about previous states or measurements. It can therefore easily be used in an iterative manner in order to reconstruct a series of states, given a series of state measurements. The filter thus has the Markov property.

3.4.1.1 Likelihood evaluation

In this thesis, we are interested in using the UKF to calculate the likelihood of some measurement data given initial state X_0 and latent parameters β. For this problem, the likelihood reads

$$p(Z_{0:N}|X_0, \beta) = p(Z_0|X_0) \prod_{i=0}^{N-1} p(Z_{k+1}|Z_{1:i+1}, X_0, \beta),$$

where the probabilistic chain rule has been used for factorization. The likelihood of the first measurement, $p(Z_0|X_0)$ is simply

$$p(Z_0|X_0) = \mathcal{N}(h(X_0), \mathbf{R}_k),$$

while the expression for $p(Z_{i+1}|Z_{1:i-1}, X_0, \beta)$ is approximated recursively with the UKF as

$$p(Z_{i+1}|Z_{0:i}, X_0, \beta) \approx \mathcal{N}(\hat{Z}_{i+1|i}, \hat{\mathbf{S}}_{i+1|i}).$$

In order to calculate the likelihood of an arbitrary measurement Z_{i+1}, the innovation $\tilde{Z}_{i+1|i}$ is thus simply compared to the actual measurement. The explicit formula for the negative log-likelihood reads

$$\frac{1}{2}\log((2\pi)^d \det(\hat{\mathbf{S}}_{i+1|i})) + \frac{1}{2}(Z_{i+1} - \hat{Z}_{i+1|i})^{\mathsf{T}}(\hat{\mathbf{S}}_{i+1|i})^{-1}(Z_{i+1} - \hat{Z}_{i+1|i}),$$

and the total negative log-likelihood is then just a sum of the terms on this form plus the likelihood for the first measurement Z_0.

3.4.1.2 Filtering conditioned on jumps

In this thesis we handle SDEs that contain discontinuous jumps. The approach we take to the jumps is to try to figure out when they happen in a sampling algorithm, such that we can condition on them in the likelihood evaluation. We must therefore adapt the filtering algorithm to the case when it is conditioned on jumps. First, we note that the jumps only play a part in the prediction stage of the filter. The prediction stage is adapted as follows: If there is not a jump before the next measurement time, a normal predict and update step is preformed. If there are one or

more jumps before the next measurement, an intermediate state is predicted, with time corresponding to the time to when the next jump happens. The state is then modified according to the jump size. A new predict step is then done and the procedure is repeated until there are no more jumps. After a final prediction step, the update step is preformed.

4

Practical aspects of PDMPs

This chapter considers practical aspects of using PDMPs to sample probability distributions. One of the main difficulties of implementing a PDMP sampler is the generation of event times from inhomogeneous Poisson processes. This can be addressed by *Poisson thinning*, which is outlined in Section 4.1. Another challenge for practical applications of PDMPs is that the sampling needs to be sufficiently fast. One possible method towards this goal is the use of approximate bound models, discussed in Sections 4.1.1–4.1.3. Another way in which the sampling can be sped up is by means of preconditioning, which is explained in Section 4.2. In Section 4.3 automatic differentiation is presented as an alternative to manual derivation of likelihood functions. Finally, in Section 4.4 we explain how PDMPs can be represented in a computer and propose methods to overcome the issues that arise when sampling variables with non-global support.

4.1 Poisson thinning

One challenge that arises when implementing a PDMP is that of simulating the first event time τ from an inhomogeneous Poisson process with rate $\lambda(t)$. The first event time τ is a random variable with survival function

$$\mathbf{P}(\tau > t) = \exp\bigg(-\int_0^t \lambda(s)\mathrm{d}s \bigg).$$

For some functions $\lambda(t)$, the survival function can be analytically inverted, allowing efficient simulation of event times. One such example is if $\lambda(t)$ is a linear function, but unfortunately, such cases are rare and the rates encountered in this thesis are not on such a form. A common way to overcome this problem is by *Poisson thinning*, presented in Proposition 4.1.1 [15].

Proposition 4.1.1 (Poisson thinning)**.** *Let $\lambda : \mathbb{R}_+ \to \mathbb{R}_+$ and $\tilde{\lambda} : \mathbb{R}_+ \to \mathbb{R}_+$ be two functions such that $\tilde{\lambda}(t) \geq \lambda(t)$ for $t \geq 0$. Let $\tilde{\tau}_1, \tilde{\tau}_2, \ldots$ be an infinite sequence of event times generated from the inhomogeneous Poisson process with rate $\tilde{\lambda}(t)$. If $\tilde{\tau}_i$ is removed from the sequence with probability $1 - \lambda(\tilde{\tau}_i)/\tilde{\lambda}(\tilde{\tau}_i)$ for $i \in \mathbb{N}$, then the resulting sequence $\tau_1, \tau_2, \ldots$ are event times sampled from an inhomogeneous Poisson process with rate $\lambda(t)$.*

Thinning is thus the process of simulating events from a Poisson process with rate $\tilde{\lambda}(t)$ satisfying $\tilde{\lambda}(t) \geq \lambda(t)$, and then removing events in such a way that the remaining events can be considered generated from a Poisson process with rate $\lambda(t)$. We

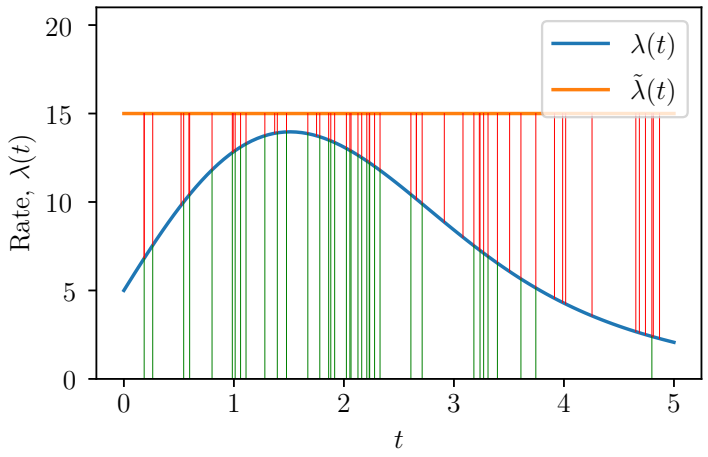


Figure 4.1: Thinning of $\lambda(t)$ using the constant bound $\tilde{\lambda}(t)$. Vertical red lines correspond to event proposals and green lines correspond to accepted events.

call $\tilde{\lambda}(t)$ a *computational bound*. In order for thinning to be useful, $\tilde{\lambda}(t)$ needs to be constructed in such a way that it is easy to simulate events from. An example of an inhomogeneous Poisson process that has been simulated with thinning is shown in Figure 4.1.

Proposition 4.1.1 formulates Poisson thinning for an infinite sequence of events, but in the simulation of a PDMP we are only interested in the first event time. Hence, we do not generate the full sequence $\tau_1, \tau_2, \ldots$, but rather only generate τ_1 and perform thinning on that event proposal. The thinning procedure can then be considered as an acceptance/rejection step which is repeated until an event is accepted.

4.1.1 Computational bounds

The design of computational bounds $\tilde{\lambda}$ can affect both the efficiency and accuracy of a sampling algorithm. In this section, we discuss some aspects of computational bounds relevant to their design and applicability.

A computational bound is informally said to be *tight* when $\tilde{\lambda}$ is close to λ. Tight bounds are usually preferred since they are associated with a high acceptance probability in the Poisson thinning step. However, tighter bounds are often challenging to obtain and can require extensive computational effort. Finding suitable bounds is often a trade-off between computational efficiency and tightness. Another characterization of a bound is that of whether it is exact or approximate. In order for a simulated PDMP to admit the correct invariant distribution, we require that $\tilde{\lambda}(t) \geq \lambda(t)$ for all $t \geq 0$. Any bound $\tilde{\lambda}$ that fulfills this is said to be an *exact bound*. In contrast, a bound which violates this inequality is said to be *approximate*.

For some problems, reasonably tight and exact bounds can be found analytically. Such cases include when $\lambda(t)$ is globally bounded, when it is Lipschitz continuous, and when a dominating Hessian can be found [1]. Unfortunately, analytical bounds can suffer from some practical drawbacks. One problem is that they require explicit

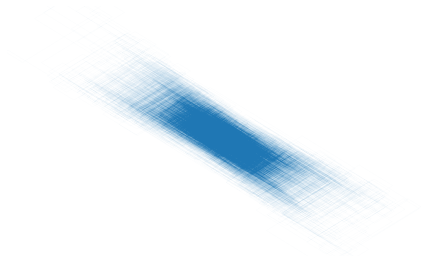


Figure 4.2: Example of inaccurate sampling of a Gaussian with high degree of bound violations.

derivation of Ψ in all relevant variables and for each derivative, a suitable bound must be found, which can be a tedious and error-prone task. For these reasons, analytical bounds will not be explored further in this thesis.

A PDMP $(\Xi_t, \Theta_t)_{t\geq 0}$ on the state space E is associated with one or more event rates. Such an event rate $\lambda(t)$ can be expressed as a function of state rather than time as $\lambda(t) = m(\Xi_t, \Theta_t)$. Based on this we introduce a computational bound $M(x) \geq m(x)$ for all $x \in E$. If we have $M(x') \geq m(x')$ for all $x' \in A$, where $A \subset E$ is a set with probability close to 1 with regards to the posterior measure, then we argue that this bound may not need to be exact. The reason for this is that it may not be necessary to bound m in regions of the state-space where the probability of visiting is very small. This suggests that an approximate bound can be a feasible approximation. There are many ways to construct approximate bound models. In this thesis, we use two variants. In Section 4.1.2 a model-based approach is outlined, and in Section 4.1.3 a bound which uses local information about the rate is described.

Approximate bounds inherently violate the assumptions used to prove that a PDMP has the correct invariant distribution. This leads to the introduction of a bias, meaning that samples will be generated from a distribution that has been to some degree perturbed from the correct one. An exaggerated example of this is shown in Figure 4.2. Intuitively, bound violation events mean that the process will be more inclined to move into improbable regions of the state space than it should, and the posterior is therefore broadened and smoothed out. One way to quantify this problem is by the *bound violation rate*, i.e., how often do we find $M < m$ in proportion to the total number of proposed events during sampling? In order to achieve sufficiently accurate sampling, a low bound violation rate is expected to be a necessary, but not sufficient criterion.

4.1.2 Constant bound model

Let us now introduce the first of two approximate bound models. The model is presented for a single Poisson clock but it is straightforward to duplicate the model for multiple variables. At event times $\tau_1, \tau_2, \ldots$, samples $m(\Xi_{\tau_1}, \Theta_{\tau_1}), m(\Xi_{\tau_2}, \Theta_{\tau_2}), \ldots$ which we denote $m_1, m_2, \ldots$ are obtained. They are used as data for the regression model

$$m_i = \beta_0 + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, c^2),$$

where β_0 and c are unknown parameters. The variance c^2 can be estimated by calculating the variance of the samples. A Gaussian prior $\mathcal{N}(\mu, \sigma^2)$ is used for β_0 leading to a posterior of the form

$$-\log(p(\beta_0|\{\tilde{m}_i\})) = \sum_i \frac{1}{2}\frac{(\tilde{m}_i - \beta_0)^2}{c^2} + \frac{1}{2}\frac{(\mu - \beta_0)^2}{\sigma^2} + const.$$

The posterior is Gaussian with mean $\hat{\beta}_0$ and variance $\hat{\sigma}^2$. The estimates of $\hat{\beta}_0$ and $\hat{\sigma}$ can be updated iteratively whenever a new sample is obtained. Given a new sample m_i and current estimates of $\hat{\beta}_0$ and $\hat{\sigma}$, the updated estimate for the mean $\hat{\beta}_0^{\text{new}}$ is

$$\hat{\beta}_0^{\text{new}} = \left(\frac{m_i}{c^2} + \frac{\hat{\beta}_0}{\hat{\sigma}^2}\right) \Big/ \left(\frac{1}{c^2} + \frac{1}{\hat{\sigma}^2}\right),$$

The updated estimate of the standard deviation $\hat{\sigma}^{\text{new}}$ is

$$\hat{\sigma}^{\text{new}} = 1 \Big/ \sqrt{\frac{1}{c^2} + \frac{1}{\hat{\sigma}^2}}$$

The regression model is used to construct the bound

$$M = \hat{\beta}_0 + k(c + \hat{\sigma}).$$

Here, the parameter k is a "safety factor" used to tune the probability of bound violation. For $k = 0$, the acceptance rate is large, but so is the bound violation probability, implying a large bias. Large values of k are expected to lead to a smaller bias but reduced computational efficiency. We call this the *constant bound model* and note that it is a simplified version of the model in [16].

4.1.3 Quasi-local bound model

An alternative way to construct a bound model is to make it *local*, i.e., a function of the local curvature of the log-likelihood function. Given the most recent event proposal time τ, we can, taking inspiration from [17], construct such a bound by using a first-order Taylor expansion of m as

$$\begin{aligned} \tilde{\lambda}(t) &= a + b(t - \tau), \quad t \geq \tau, \\ a &= (m(\Xi_\tau, \Theta_\tau))^+ + \Gamma, \\ b &= \gamma|\partial_t m(\Xi_\tau, \Theta_\tau))|, \end{aligned}$$

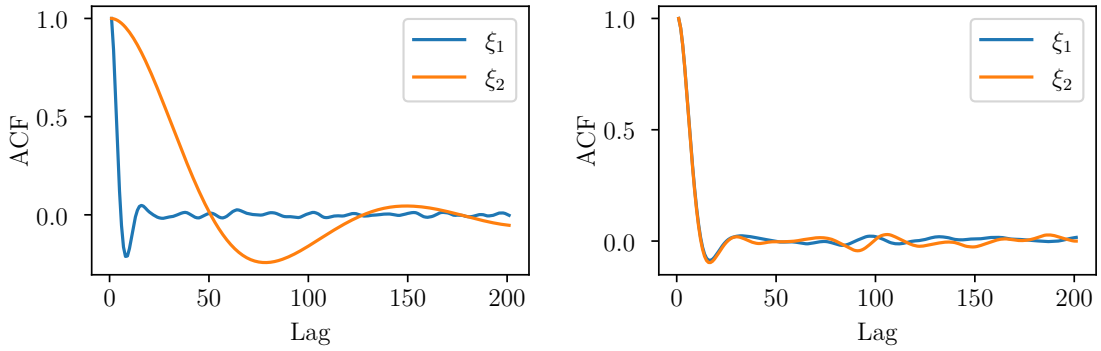
where t is the PDMP time and $\Gamma \geq 0$ and $\gamma \geq 1$ are model parameters. The coefficients of the model are updated at every event proposal. We call this the *quasi-local bound model* because it uses local information in the sense of the derivative of m but there is no maximum time t_{exp} after which the model. Since there is no expiration time, the model may be approximate for many realistic likelihoods. However, we note that it is possible to make a local bound model exact. One such solution is presented in [18], where a minimization approach is used in order to create local and exact upper bound valid for some specified time. A downside of this approach is

that it requires minimization of m, which may be computationally expensive. This approach could be interesting if likelihood evaluations are cheap, which is not the case in this thesis.

The intuition behind the quasi-local bound model is that there exists a linear function $\tilde{\lambda}(t) = a + bt$ which locally bounds λ, presuming that it is a locally Lipschitz continuous function. However, such a bound is only exact for an unknown expiry time t_{exact}. Since this model does not expire after t_{exact} it is an approximate model and bound violations are possible. In order to maintain control over the bound violation rate, the slope of the bound model is multiplied with a constant factor $\gamma > 1$ such that the bound initially has a steeper slope than λ. Additionally, Γ is used to maintain a minimum event proposal rate to avoid problems when $a \approx 0$ and $b \approx 0$. Alternatively, one could explore the use of an expiration time t_{exp} on the bound, after which time the bound model is updated regardless of whether there have been any event proposals or not.

4.2 Preconditioning

PDMPs extend the sample space with a velocity component for each variable. Together, these velocity components make up a velocity vector for the process. If this velocity vector is poorly aligned with the shape of the distribution, then the sampler may reflect very often in some variables while reflecting very seldom in others. Consider sampling with the Zig-Zag sampler from a Gaussian distribution with a badly conditioned covariance matrix, for instance, $\mathbf{\Sigma} = \text{Diag}(1, 10)$. Although the components are independent, the rates at which new independent samples are being obtained differ quite drastically between the variables. Since the rate at which new independent samples are obtained is limited by the slowest of the state components, this indicates an inefficiency. The aim of preconditioning is to recondition the problem in a way that makes it more suitable for simulation with a PDMP. An example of this is shown in Figure 4.3.



(a) Without preconditioning. **(b)** With preconditioning.

Figure 4.3: Comparison of autocorrelation functions (ACF) with and without preconditioning. In this example a gaussian distribution with $\mathbf{\Sigma} = \text{Diag}(1, 10)$ is sampled with the Zig-Zag sampler.

In [19] the use of linear transformation matrices for preconditioning is explored, and we follow this precedent. We use a linear transformation matrix $\mathbf{L}^\mathsf{T}$ given from a Cholesky factorization $\mathbf{\Sigma}^{-1} = \mathbf{L}\mathbf{L}^\mathsf{T}$, where $\mathbf{\Sigma}$ is the covariance matrix of the posterior. This choice is based on the following observation: If a random variable X has a multivariate Gaussian distribution, then its log likelihood function is proportional to $X^\mathsf{T}\mathbf{\Sigma}^{-1}X$. We write

$$X^\mathsf{T}\mathbf{\Sigma}^{-1}X = X^\mathsf{T}\mathbf{L}\mathbf{L}^\mathsf{T}X = (\mathbf{L}^\mathsf{T}X)^\mathsf{T}(\mathbf{L}^\mathsf{T}X)$$

and observe that the transformed variables $\mathbf{L}^\mathsf{T}X$ are normally distributed with the identity matrix as their covariance matrix. They are thus independent and have variance 1.

In practice, the posterior is often not Gaussian and therefore poorly represented by its covariance matrix. In this case, preconditioning by means of a linear matrix may not perform optimally, however, it turns out to still be useful.

An additional complication is that $\mathbf{\Sigma}$ is a priori unknown which means that we cannot directly calculate $\mathbf{L}^\mathsf{T}$. The solution adopted in this thesis is to estimate $\mathbf{\Sigma}$ using an empirical distribution obtained during a burn-in period. An alternative could be to obtain a maximum a posteriori probability (MAP) estimate of the posterior and calculate the local curvature, the Hessian, at the MAP estimate. For a Gaussian distribution, the Hessian at the MAP is related to the covariance as $\mathbf{H} = \mathbf{\Sigma}^{-1}$.

4.3 Automatic differentiation

The simulation of a PDMP requires evaluations of derivatives of the negative log-likelihood Ψ. In the Zig-Zag sampler, partial derivatives $\partial_i\Psi$ are required, while the bouncy particle sampler requires the gradient $\nabla\Psi$ to be evaluated. The local bound model requires second-order derivatives of Ψ. In realistic settings, it is often impractical to derive these manually, as the likelihood may include complicated user-defined functions such as filters, transformations, and control flow. For these reasons, we seek a way to evaluate derivatives in an automated manner.

There are three main approaches available: *symbolic differentiation, numeric differentiation* and *automatic differentiation*. Symbolic differentiation is essentially using known derivation rules to obtain an explicit expression of the derivative. This approach may be suitable in some cases, especially if $\Psi(\xi)$ can be expressed somewhat simply as an analytic function. However, it is not practically viable for more complicated likelihood functions. In contrast, numeric differentiation uses two (or more) nearby points to approximate the derivative as

$$\partial_i\Psi(\xi) \approx \frac{\Psi(\xi + \mathbf{e}_i h) - \Psi(\xi)}{h},$$

for some small h. This approach, although general, has the problem of choosing a suitable value of h. If h is too big, the approximation error can be large, and if h is too small, numerical instability may lead to errors. Achieving numerical precision can therefore be tricky.

Automatic differentiation circumvents issues of both symbolic and numeric differentiation. The intuition behind automatic differentiation is that any function evaluated on a computer is a composition of elementary functions for which exact derivation rules are known. Consider the function

$$y = (f_N \circ \cdots \circ f_1)(x).$$

By recursively evaluating,

$$w_k = f_k(w_{k-1}), \quad k = 1, \ldots, N; w_0 = x,$$

y is given as w_N. Similarly, the chain rule allows the evaluation of the derivative as

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}f_N(w_{N-1})}{\mathrm{d}w_{N-1}} \cdots \frac{\mathrm{d}f_2(w_1)}{\mathrm{d}w_1} \frac{\mathrm{d}f_1(w_0)}{\mathrm{d}x}.$$

Since all expressions for derivatives are exact, the calculation gives exact results up to numerical precision. There are two modes of automatic differentiation; *forward mode*, where the chain rule is traversed from f_1 to f_N (inside to outside) and *reverse mode*, where it is traversed from f_N to f_1 (outside to inside). Forward mode is more straightforward, as it coincides with the order of operations performed to evaluate y. Forward mode is appropriate for functions $f : \mathbb{R}^M \to \mathbb{R}^N$ where $M \ll N$, as one pass is needed for each partial derivative. The opposite is true for reverse mode automatic differentiation [20]. In the context of PDMP sampling, partial derivatives and gradients need to be evaluated which makes the forward mode preferable.

4.4 Practical considerations

When simulating a PDMP, its trace, $(\Xi_t, \Theta_t)_{t \geq 0}$, is obtained. In order to store this continuous quantity in a computer, so-called *skeleton points* are often used. Skeleton points are points that are generated every time an event is proposed, regardless if it is a reflection proposal, a velocity refreshment, an unfreezing event, or otherwise. In particular, this means that skeleton points are generated every time the velocity changes direction. Since the dynamics between velocity jumps are deterministic, skeleton points can be used to reconstruct the exact path of the process. Skeleton points specify the time, position, and velocity of the process and come as a triplet (t, ξ, θ), where t is the PDMP-time, ξ is the state and θ is the velocity. In practice, simulating a PDMP thus corresponds to obtaining a sequence of skeleton points $(T_n, \Xi_n, \Theta_n)_{n \in \mathbb{N}_0}$.

Another practical problem when sampling with PDMPs is that of making sure that a sampled variable is reflected when it reaches the edge of its support. This problem can arise when sampling from a probability distribution with non-global support such as the exponential distribution which has support on $\mathbb{R}_+$, or the uniform distribution with support on a closed interval. We use two ways to address these problems: Manual reflection events and variable transformations. Manual reflection events are hard-coded reflections of the velocity component once the variable reaches the edge of its support. One example where this approach is suitable is when

sampling a variable that has a uniform prior. The other approach uses a transformation to transform a variable with non-global support into a variable with global support. One example of such a transformation is $x \to \exp(x)$, which can be used to transform a variable with support on $\mathbb{R}_+$ to a variable with support on $\mathbb{R}$.

5

Models

In this chapter, we present the target and sensor models used in this thesis. In Section 5.1, a sensor model, intended to capture the geometry of a typical target-sensor scene for a radar sensor is defined. In Section 5.2 we define the first target model. This model consists of an SDE that is intended to represent targets that travel on ballistic trajectories. This class of targets includes for example artillery and mortars. In Section 5.3 we present a second target model that includes jumps in the acceleration variable. This model is one-dimensional and somewhat simpler in order to focus on the complexities that arise when sampling distributions related to SDEs that include jumps.

In Section 5.4 we define the posterior to be sampled for the ballistic model and present the SDE approximations used in order to make sampling feasible. Finally in Section 5.5 we define a posterior related to the acceleration jump-drag model, which includes times and magnitudes of jumps. A novel approach for sampling from such a posterior using a binomial approximation is developed. The approximation results in a posterior that is suitably sampled with the sticky Zig-Zag sampler.

5.1 Measurement model

We consider a measurement model (sensor) that produces target measurements of the following four quantities: *range, radial velocity, elevation angle* and *azimuth angle*. In Figure 5.1 the geometry of the target-sensor scene is shown.

Each measurement made by the sensor is modeled as a random variable, here denoted Z, that depends on the state of the target X. It is specified by a distribution $p(Z|X) = \mathcal{N}(h(X), \mathbf{R})$, where h is a measurement function and $\mathbf{R}$ is a covariance matrix that represents measurement noise. Assuming that the sensor is placed at the origin and measures a state $X = [\vec{p}, \vec{v}] = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$, $h(X)$ takes the form:

$$h(X) = \begin{pmatrix} \|\vec{p}\| \\ \vec{v} \cdot \vec{p}/\|\vec{p}\| \\ \arcsin(z/\|\vec{p}\|) \\ \arcsin(y/\sqrt{x^2 + y^2}) \end{pmatrix},$$

and $\mathbf{R}$ can be specified arbitrarily. If multiple measurements are made, the noise is assumed to be uncorrelated between them. The intention of this model is to capture typical scales and geometries of a radar-target scene while avoiding details about the radar sensor.

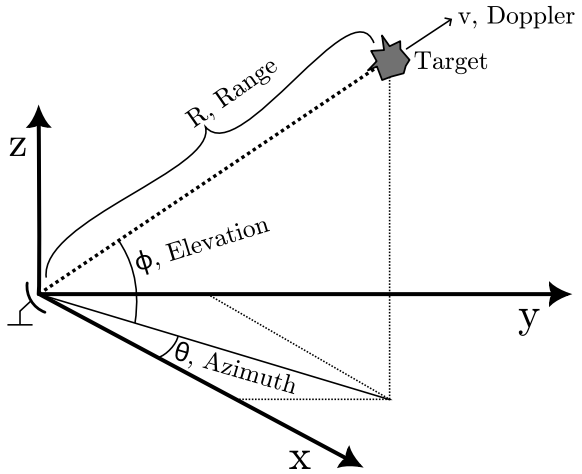


Figure 5.1: Illustration of the target-sensor scene. A sensor at the origin measures the range, radial velocity, azimuth angle, and elevation angle of a target.

5.2 Ballistic model

The ballistic target model is defined as an SDE for a 3-dimensional position and velocity state $X = [\vec{p}, \vec{v}] = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$:

$$\mathrm{d}\begin{pmatrix}\vec{p}\\ \vec{v}\end{pmatrix} = \begin{pmatrix}\vec{v}\\ -f_\mathrm{d}(\|\vec{v}\|)\|\vec{v}\|^2\hat{e}_\parallel - g\hat{e}_z\end{pmatrix}\mathrm{d}t + \begin{pmatrix}0_{3\times 1}\\ \sigma_\parallel\hat{e}_\parallel\mathrm{d}W_t^\parallel + \sigma_{\perp_1}\hat{e}_{\perp_1}\mathrm{d}W_t^{\perp_1} + \sigma_{\perp_2}\hat{e}_{\perp_2}\mathrm{d}W_t^{\perp_2}\end{pmatrix} \tag{5.1}$$

where $\hat{e}_\parallel, \hat{e}_{\perp_1}, \hat{e}_{\perp_2}$ are basis vectors in a target-aligned reference frame defined in Figure 5.2, $\hat{e}_z$ is the absolute up direction, $W_t^\parallel, W_t^{\perp_1}, W_t^{\perp_2}$ are Brownian motions, $\sigma_{\perp_1}, \sigma_{\perp_2}$ and $\sigma_\parallel$ are diffusion coefficients, $f_\mathrm{d}(\|\vec{v}\|)$ is a velocity-dependent drag coefficient and g is the gravitation. The transformation between the target-aligned and the absolute coordinate system is a simple rotation, details of which can be found in Appendix C.1.

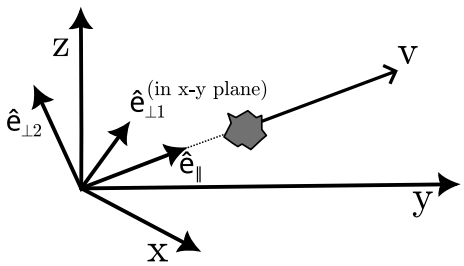


Figure 5.2: Target-aligned coordinate system. The first basis vector is aligned with the velocity $\hat{e}_\parallel = \vec{v}/\|\vec{v}\|$. The two other vectors are $\hat{e}_{\perp_1} = \vec{z}\times\hat{e}_\parallel/\|\vec{z}\times\hat{e}_\parallel\|$ where $\vec{z}$ is the absolute up direction and $\hat{e}_{\perp_2} = \hat{e}_\parallel \times \hat{e}_{\perp_1}$. Basis vector $\hat{e}_{\perp_1}$ is parallel with the $x-y$ plane and $\hat{e}_{\perp_2}$ is the up-direction in the target aligned system.

The drag coefficient $f_\mathrm{d}(\cdot)$ is inspired by [21] and is defined as

$$f_\mathrm{d}(v) = A + \frac{Ck}{\sqrt[4]{\left(\left(\frac{v}{v_0}\right)^2 - 1\right)^2 + k^4}\left(1 + \exp\left(\frac{-8(v/v_0 - (1-\delta/2))}{\delta}\right)\right)},$$

where v_0 is the speed of sound (assumed constant $343\,\mathrm{m\,s^{-1}}$) and A, C, k and δ are parameters. It is shown in Figure 5.3 with the parameters used in this thesis. We use this form on the drag coefficient because it captures general drag characteristics that are found empirically in subsonic, transonic, and supersonic regimes.

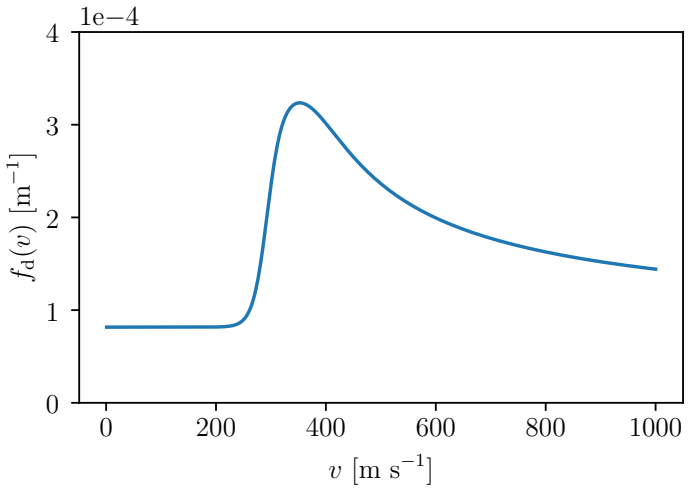


Figure 5.3: Example of drag coefficient as a function of velocity. Parameters used: $A = 10/350^2$ m^{-1}, $C = 30/350^2$ m^{-1}, $k = 0.7, \delta = 0.3$.

5.3 Acceleration jump-drag model

We now define a one-dimensional model for a target which undergoes jumps in its acceleration variable, which we call the acceleration jump-drag model. The state includes position, velocity, and acceleration and is described by a jump-diffusion SDE,

$$\mathrm{d}\begin{pmatrix} x_t \\ \dot{x}_t \\ \ddot{x}_t \end{pmatrix} = \begin{pmatrix} \dot{x}_{t-} \\ -A\,|\dot{x}_{t-}|\dot{x}_{t-} + \ddot{x}_{t-} \\ 0 \end{pmatrix} \mathrm{d}t + \begin{pmatrix} 0 \\ 0 \\ \sigma_\mathrm{process} \end{pmatrix} \mathrm{d}W_t + \int_{\mathbb{R}^d} \begin{pmatrix} 0 \\ 0 \\ y - \ddot{x}_{t-} \end{pmatrix} N(\mathrm{d}t, \mathrm{d}y), \tag{5.2}$$

where A and σ_process are drag and diffusion coefficients, respectively. The last term is a jump integral as defined in Section 2.4.2.1 with rate Λ and jump distribution $\gamma = \mathcal{N}(0, \sigma^2_\mathrm{jump})$. Note that jumps are modeled in such a way that the distribution γ does not correspond to increments of the state, but rather we consider γ as the distribution of accelerations that the process jumps *to*. This is achieved by $y - \ddot{x}_{t-}$ in the integrand.

Models of this type are compelling for targets that have control inputs. The control can for example be in the form of a rocket motor or active aerodynamic surfaces. Hypersonic missiles present one example where this may be relevant.

5.4 Sampling the ballistic model

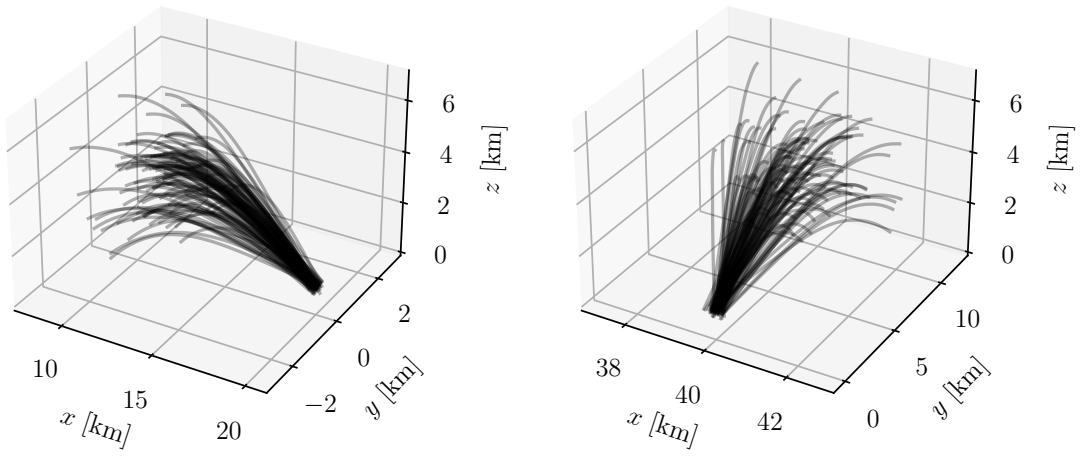
Regarding the ballistic model, we are in this thesis interested in sampling the distribution

$$p(X_0, \beta | Z_{0:N}) \propto p(Z_{0:N} | X_0, \beta) p(X_0, \beta),$$

given noisy measurements $Z_{0:N}$ of a process realization. Here, X_0 denotes the state at the first measurement time and β are the drag coefficient parameters, $\beta = [A, C, k, \delta]$. The evaluation of the likelihood $p(Z_{0:N}|X_0, \beta)$ is done with an unscented Kalman filter. The SDE is approximated by making the target-aligned basis vectors constant within each solution interval. This implies a separation of dimensions which simplifies calculations. We call this the *static rotation approximation* and it will be verified along with the UKF approximations in Chapter 6. The SDE that is obtained after the static rotation approximation is solved with the noise linearization approach outlined in Section 2.5.2.2.

5.4.1 Evaluation scenario

Two different test cases, corresponding to two different priors for initial conditions in the ballistic SDE are evaluated. In the first test case (Case 1) the target is heading towards the radar at an initial distance of 20 km. In the second test case (Case 2) the target is at twice the distance from the observer and is traveling roughly orthogonally to the radar. Figure 5.4 shows 100 sample trajectories from either test case.



(a) Case 1. **(b)** Case 2.

Figure 5.4: 100 trajectory realizations from Case 1 in (a) and Case 2 in (b). The radar sensor is placed at the origin.

The trajectories in Figure 5.4 have been simulated with drag parameters and initial state sampled from the prior distribution shown in Table 5.1. Note that rather than specifying the variables A, C, k and δ directly, a prior is set on their logarithms. This avoids problems in the sampling algorithm when they approach the edge of their support, as explained in Section 4.4.

Table 5.1: Definition of mean and standard deviation for the Gaussian priors used in each respective test case.

Variable	μ (Case 1)	μ (Case 2)	σ (Case 1 & 2)
$\log(A)$	$\log(10/350^2)$	$\log(10/350^2)$	0.3
$\log(C)$	$\log(30/350^2)$	$\log(30/350^2)$	0.3
$\log(k)$	$\log(0.7)$	$\log(0.7)$	0.3
$\log(\delta)$	$\log(0.3)$	$\log(0.3)$	0.3
x	20000	40000	100
y	0	0	100
z	1000	1000	100
$\dot{x}$	-866	0	100
$\dot{y}$	0	866	100
$\dot{z}$	500	500	100

Isotropic process noise is used with $1 = \sigma_{\perp_1} = \sigma_{\perp_2} = \sigma_{\parallel}$. This is a simplified choice and for real scenarios $\sigma_{\parallel} \gg \sigma_{\perp_{1,2}}$ might be a more realistic choice. The sensor model described in 5.1 is used to make 30 target measurements with 1 s between each measurement. The measure measurement noise matrix $\mathbf{R}$ is assumed to be diagonal (uncorrelated noise) with entries given in Table 5.2.

Table 5.2: Uncertainties in radar measurements.

Measurement	Standard deviation
Range	20 m
Radial velocity	$2\,\mathrm{m\,s^{-1}}$
Azimuth	0.2 °
Elevation	0.2 °

5.5 Sampling the acceleration jump-drag model

Our objective when sampling with the acceleration jump-drag model is to find a posterior distribution of jump times and magnitudes $(\tau_i, \Delta_i)_{i\in[1..K]}$ given some data, i.e.

$$p((\tau_i, \Delta_i)_{i\in[1..K]}|Z_{0:N}), \tag{5.3}$$

where K is a discrete and unknown number of jump times and $Z_{0:N}$ are $N+1$ uniformly spaced noisy measurements of the acceleration jump-drag process. In the

following, we present a novel approach to sample from such a distribution with the sticky Zig-Zag sampler.

One of the fundamental problems when trying to find jump times and magnitudes is that the number of jumps K is unknown. Our solution to this problem is based on approximating the underlying Poisson process with a Binomial process. The Binomial process has a fixed number of events, which allows us to use a sample space of fixed dimensionality. Furthermore, the binomial process can be shown to be equal to the Poisson process in the limit where the number of events goes to infinity while the mean is kept constant. This suggests that a binomial process with matched first moment could be a suitable approximation for the Poisson process. A detailed description now follows.

We introduce a state space consisting of *variable pairs* $(\tau_i, M_i), i = 1, \ldots, s$, where $\tau_i \in [0, T]$ represents the time of a jump and M_i represents the jump size. Let Λ be the rate of the homogeneous compound Poisson process which we are approximating. The unknown number of events K generated in time T is distributed according to the Poisson distribution with parameter ΛT, and the expected number of events is $\mathbf{E}[K] = \Lambda T$. The number of variable pairs s is chosen as $s \gg \mathbf{E}[K]$, such that the probability that there are more jumps than we have sampling variables is sufficiently low.

Since the Poisson process is homogeneous, the event times which are generated will be uniformly distributed on $[0, T]$. This tells us that a uniform prior should be selected for the time variables τ_i. For the magnitude variables M_i, identical slab-and-spike priors on the form

$$u_i(x) = w\pi(x) + (1 - w)\delta_0(x)$$

are used, where $\pi(x)$ is the prior distribution of jump magnitudes, given that a jump has occurred and δ_0 is the Dirac measure at 0. We say that a variable pair with non-zero jump magnitude is *active*. The number of active variables N are distributed according to the binomial distribution $B(s, w)$. Its mean is $\mathbf{E}[N] = sw$. A reasonable way to choose w would be to match the expected number of active variables to the expected number of jumps as $\mathbf{E}[K] = \mathbf{E}[N]$. This also guarantees that the Poisson limit alluded to above is exact, and we obtain a simple formula for the weights,

$$w = \frac{\Lambda T}{s}.$$

We have now specified a state space of s identical variable pairs with priors. The magnitude variables have spike and slab priors, and the time variables have uniform priors. Sampling with the sticky Zig-Zag sampler is therefore suitable.

A nice property of this method is that the computational burden during sampling does not scale significantly with the number of variable pairs used. This is because as more variable pairs are introduced, the probability that they are in the frozen state, where they have essentially 0 computational burden, is increased.

Finally, to avoid confusion we note that the prior distribution π is not the same as γ in Section 5.3. The prior $\pi(x)$ models *increments* in the state at jump times, whereas

$\gamma(x)$ is the distribution of accelerations that the process may jump to. This means that there is a slight mismatch between the process that generates the trajectory and the sampling method. This is nothing strange, as there is already a model mismatch between the Poisson process used to model the jumps and the true jumps, which correspond to control inputs at chosen times throughout the trajectory.

5.5.1 Evaluation scenario

To evaluate the ability to sample jump times and magnitudes, we use a single realization of the acceleration jump-drag SDE, i.e., equation (5.2). The SDE is simulated with a jump intensity $\Lambda = 0.1$ (one jump expected every 10 seconds) and a jump distribution with standard deviation $\sigma_{\text{jump}} = 100$. The SDE parameters are $A = 10/350^2$ and $\sigma_{\text{process}} = 1$. A total of 20 position and velocity states are measured with 1 s between each measurement. The measurement noise is Gaussian and has a standard deviation of 200 m in the position component, and $20\,\text{m}\,\text{s}^{-1}$ in the velocity component.

The specific SDE realization used is shown in Figure 5.5. A filtering solution that does not account for jumps is also shown. We see that while the position estimates look reasonable, the velocity and especially acceleration estimates are unsatisfactory. It should however be noted that the comparison with the filter solution is somewhat unfair, as any filtering solution which is not conditioned on the jumps should include the jump noise in the covariance matrix, which it currently does not.

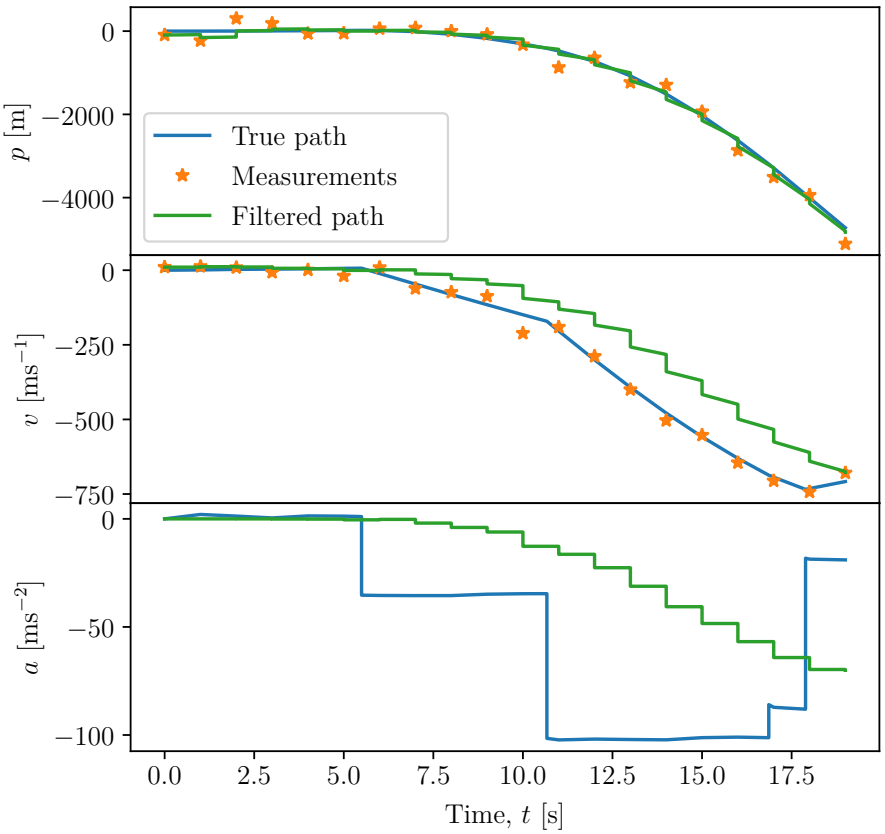


Figure 5.5: A realization of the acceleration jump-drag SDE with noisy position and velocity measurements. The filtering solution is poor since it has not been conditioned on the jumps.

6

Results

In this chapter, PDMP samplers are used to sample posteriors related to the models outlined in Chapter 5. In order to be able to evaluate and compare their performance, we first describe our selected performance metrics in Section 6.1. This is followed by results for the ballistic model in Section 6.2, where the Zig-Zag sampler with constant bounds is compared to the bouncy particle sampler with both constant and quasi-local bounds. In section 6.3 the sticky Zig-Zag sampler is used to find unknown jump times and magnitudes in the acceleration jump-drag model.

6.1 Evaluation

In this section, we describe the performance metrics used to evaluate our selected PDMP sampling algorithms. In Section 6.1.1, a method to evaluate accuracy which encompasses both sampling and likelihood approximations is presented. This method is used to verify that all our approximations taken together do not result in a significant bias. In Sections 6.1.2–6.1.3 two different notions of efficiency are presented. These are important for comparisons of the different sampling algorithms as well as absolute performance in terms of computation time.

6.1.1 Prior reconstruction accuracy

In order for sampling to be practical, approximations are used. In this thesis, approximations are used in two areas, approximate likelihood evaluation and approximate bounds for PDMP simulation. These can both lead to a biased posterior. For this reason, there is a deviation between the true posterior $p(\beta, X_0|Z_{0:N})$, and the distribution which we can realistically sample $\tilde{p}(\beta, X_0|Z_{0:N})$. We seek a way to verify that this deviation is sufficiently small for our purposes.

Consider a prior on the sampled parameters $p(\beta, X_0)$. This can be rewritten as marginalization over all measurement realizations as

$$p(\beta, X_0) = \int p(\beta, X_0|Z_s)p(Z_s)\mathrm{d}Z_s.$$

Consider the following two approximations of the right-hand side: First, discretize the integral with a sum over distinct measurement realizations Z_k. Second, replace $p(\beta, X_0|Z_k)$ with an approximation $\tilde{p}(\beta, X_0|Z_k)$, obtained through sampling. This

results in the an approximation $\tilde{p}(\beta, X_0)$ of $p(\beta, X_0)$ which reads

$$\tilde{p}(\beta, X_0) = \frac{1}{K} \sum_{k=1}^{K} \tilde{p}(\beta, X_0 | Z_k), \tag{6.1}$$

where Z_k are measurement sequences from the true prior distribution $p(\beta, X_0)$. For large K the discretization error tends to zero, and the difference between $p(\beta, X_0)$ and $\tilde{p}(\beta, X_0)$ is dominated by the sampling approximation. If the sampling error is small, this should result in a small difference between the two distributions. This can be verified in multiple ways and we choose to do so visually.

We remark that an accurate reconstruction of the prior, by marginalizing the approximate posterior is a good indication of an accurate posterior. It is a convenient substitute for comparing the posterior itself as the latter is not known analytically. An alternative would be to use some exact sampling scheme to study the posterior directly.

6.1.2 Effective sample size

Next, we give the first of two ways to quantify sampling efficiency based on *effective sample size* (ESS). Effective sample size is a way to quantify the number of independent samples in a correlated sequence of samples, where each sample individually has the same probability distribution. It is based on the *autocorrelation function* (ACF), which quantifies the degree of correlation in a sequence of samples $s_1, \ldots, s_N$ as

$$\rho(l) = \frac{\sum_{n=1+l}^{N} (s_n - \bar{s})(s_{n-l} - \bar{s})}{\sum_{n=1}^{N} (s_n - \bar{s})^2},$$

where l is the *lag*, i.e., the distance between the samples that are being compared, and $\bar{s}$ denotes the mean of the sequence $s_1, \ldots, s_N$. While it can be instructive to plot the autocorrelation as a function of lag, there is often a need to summarize the information. ESS is one way to achieve this, and is calculated as

$$\text{ESS} = \frac{N}{1 + 2 \sum_{\ell=1}^{\ell_{max}} \rho(\ell)},$$

with ℓ_{max} corresponding the the first point at which $\rho(\ell) < 0.05$ [7]. While ESS can be used as an efficiency metric by itself, it can be useful to define a metric that is independent of N. For this purpose, we define *correlation length* as N/ESS. Correlation length should be thought of as the number of correlated samples needed before the first and last sample in the sequence can be considered independent. It is an important measure of sampling efficiency because it dictates the number of sampling iterations needed to obtain a new independent sample.

Since PDMP samplers are continuous in time, the application of these concepts is not direct. An alternative formulation of ESS for continuous sample paths is presented in [1]. Here the PDMP-trace is partitioned into batches which are then integrated. We empirically found that results from this approach were similar to using the skeleton points as input to the autocorrelation function, i.e., treating the

skeleton points as samples. Since the results were similar and the skeleton point method avoids the complication of choosing the batch size parameter, that method is used. Additionally, this makes the interpretation of correlation length as a number of loop iterations direct.

6.1.3 Convergence of expectations

For some applications, complete convergence of the posterior in terms of Kullback–Leibler divergence or some other distance metric might not be necessary. Instead, the convergence of certain expectations with respect to the sample trace might be sufficient. For our task, relevant expectations include means and variances of marginals of the posterior. The mean of a marginal posterior distribution $\hat{\mu}_i$ for variable i can be estimated as

$$\hat{\mu}_i(t) = \frac{1}{t}\int_0^t (\Xi_s)_i \mathrm{d}s, \tag{6.2}$$

and the variance $\hat{\sigma}_i^2$ can be estimated as

$$\hat{\sigma}_i^2(t) = \frac{1}{t}\int_0^t (\Xi_s)_i^2 \mathrm{d}s - \hat{\mu}_i(t)^2, \tag{6.3}$$

where t is the time parameter in the PDMP trajectory. We note that these expectations only capture select aspects of the posterior, covariances between variables are for example not captured (although they could be if the above formulae are adapted).

6.2 Ballistic model

For the ballistic model, we evaluate the Zig-Zag sampler with constant bounds, the bouncy particle sampler with constant bounds, and the bouncy particle sampler with quasi-local bounds for each of the two test cases. The principal reason for not using the quasi-local bound model with the Zig-Zag sampler is that there are individual reflection rates for each of the variables. This means that the velocity vector in the Zig-Zag process is not aligned with the partial derivatives $\partial_{\xi_i}\Psi$ used in the bounds for each respective variable.

First, in Section 6.2.1 we describe and analyze the approximations made in the likelihood evaluation. Having verified our approximations, the posterior is sampled. This starts with the initialization of the preconditioning matrix, presented in Section 6.2.2. After an appropriate preconditioning matrix has been estimated the real sampling phase begins. Results are presented in Section 6.2.3 in terms of prior reconstruction accuracy and efficiency for each of the test cases and PDMP variants.

6.2.1 Approximate likelihood evaluation

The purpose of this section is to describe the method by which the likelihood is evaluated for the ballistic model and discuss the approximations that have been made.

Likelihood evaluation for the ballistic model is done with an unscented Kalman filter, with system dynamics given by the ballistic SDE in (5.1). Since this SDE is difficult to solve efficiently, the static rotation approximation introduced in Section 5.4 is used to approximate the SDE. This approximation effectively decouples the velocity-aligned dimension from the other two, creating three independent SDEs. The SDE in the velocity direction is solved with the noise linearization method outlined in Section 2.5.2.2 while the orthogonal (drag-free) directions are already linear and are thus analytically solved by the calculation performed in Section 2.5.1. Details about the solution to noise linearized SDE can be found in appendix C.2. Finally, a fourth-order Runge-Kutta ODE solver (RK4) is used to solve the nonlinear propagation problem.

Next, we seek to verify or motivate the approximations that have been made, starting with the static rotation approximation and the noise linearization approach. In order to understand this approximation we solve the exact ballistic SDE using parameters from Section 5.4.1 with Euler-Maruyama simulations and compare the mean and covariance of the resulting distribution obtained with our method. For the Euler-Maruyama method we perform 100000 simulations from the initial state $X_0 = [0, 0, 0, 1000/\sqrt{2}, 0, 1000/\sqrt{2}]$ and a time step of 1 ms. Due to the fine discretization and large number of simulations we can consider this as a benchmark. In the noise linearization method, the RK4 method solves the ODE using a single step. In both cases, the total time step is 1 s.

The mean μ_{EM} and covariance Σ_{EM} obtained using Euler-Maruyama simulations are given by

$$\mu_{\text{EM}} = \begin{pmatrix} 659.647 \\ 0.001 \\ 654.96 \\ 615.288 \\ 0.001 \\ 606.114 \end{pmatrix}, \quad \Sigma_{\text{EM}} \begin{pmatrix} 0.296 & -0.001 & -0.008 & 0.425 & -0.002 & -0.014 \\ \text{sym.} & 0.298 & 0.001 & -0.001 & 0.432 & 0.002 \\ \text{sym.} & \text{sym.} & 0.291 & -0.014 & 0.002 & 0.418 \\ \text{sym.} & \text{sym.} & \text{sym.} & 0.850 & -0.003 & -0.026 \\ \text{sym.} & \text{sym.} & \text{sym.} & \text{sym.} & 0.867 & 0.002 \\ \text{sym.} & \text{sym.} & \text{sym.} & \text{sym.} & \text{sym.} & 0.840 \end{pmatrix}.$$

We compare this with the mean μ_{approx} and covariance Σ_{approx} given from solving the approximated SDE with the noise linearization approach,

$$\mu_{\text{approx}} = \begin{pmatrix} 659.595 \\ 0.0 \\ 654.904 \\ 615.296 \\ 0.0 \\ 606.121 \end{pmatrix}, \quad \Sigma_{\text{approx}} = \begin{pmatrix} 0.309 & 0.0 & -0.025 & 0.452 & 0.0 & -0.048 \\ \text{sym.} & 0.333 & 0.0 & 0.0 & 0.5 & 0.0 \\ \text{sym.} & \text{sym.} & 0.309 & -0.048 & 0.0 & 0.452 \\ \text{sym.} & \text{sym.} & \text{sym.} & 0.905 & 0.0 & -0.095 \\ \text{sym.} & \text{sym.} & \text{sym.} & \text{sym.} & 1.0 & 0.0 \\ \text{sym.} & \text{sym.} & \text{sym.} & \text{sym.} & \text{sym.} & 0.905 \end{pmatrix}.$$

We first note that the approximated mean μ_{approx} agrees well with μ_{EM} since it is well within one standard deviation (obtained from the EM covariance). This implies that a single Runge-Kutta step is enough. The covariance matrices generally look similar as well, but there are some surprising larger deviations. The fifth diagonal element, corresponding to the variance of $\dot{y}$ is interesting to understand. Since the noise coefficient is 1, the simulation is run for 1 second and the drift term is 0 in tangential directions, one would expect the variance to be 1, but we find the

value 0.867. The discrepancy comes from the fact that noise and gravity rotate the coordinate system such that a small part of the drag force gets projected on the initially orthogonal components. The projected force is restoring drag force approximately proportional to v, which explains why the variance is smaller than 1. If the experiment is repeated with lower velocity, then the variance of $\dot{y}$ is indeed found to be closer to 1.

We also perform the corresponding simulation in one dimension. In this case, there are no orthogonal directions and the "projected drag" effect can therefore not exist. We perform 10000 Euler-Maruyama simulations using a time-step of 1 ms and obtain the mean μ_{EM} and covariance Σ_{EM}

$$\mu_{\text{EM}} = \begin{pmatrix} 932.81 \\ 869.95 \end{pmatrix}, \quad \Sigma_{\text{EM}} = \begin{pmatrix} 0.284 & 0.407 \\ \text{sym.} & 0.820 \end{pmatrix}.$$

The noise linearization method results in the mean μ_{approx} and covariance Σ_{approx} given as

$$\mu_{\text{approx}} = \begin{pmatrix} 932.74 \\ 869.96 \end{pmatrix}, \quad \Sigma_{\text{approx}} = \begin{pmatrix} 0.284 & 0.404 \\ \text{sym.} & 0.811 \end{pmatrix}.$$

In this case, the difference between the means and covariances are very small and we therefore conclude that the error in the covariance matrix above is mainly a result of the static rotation error.

The static rotation approximation can also lead to an angular error, i.e., an error in the direction in which the target aligned basis vector $\hat{e}_{\parallel}$ is pointing. This error is proportional to $1/v$, as the impulse needed to rotate the velocity vector by a fixed amount grows linearly with the target velocity. Since ballistic target speeds are typically high, this error is assumed insignificant on small timescales (1 s).

Next, we consider the use of the UKF. One approximation made by the UKF is that state distributions can be represented by a Gaussian distribution before and after propagation through a nonlinear function. This assumption is now tested. We do this by propagating an initially Gaussian distribution through the ballistic nonlinearity with the Euler-Maruyama algorithm and inspecting the obtained distribution. The initial distribution is centered at $[0, 0, 0, 1000/\sqrt{2}, 0, 1000/\sqrt{2}]$ with independent components which all have a standard deviation of 100. The exact SDE (5.1) with drag parameters as above is simulated 10000 times with a time discretization parameter $\delta t = 1\,\text{ms}$ for 1 s. The resulting distribution is shown in Figure 6.1. We notice that with the selected variance in the initial state, the distribution looks roughly Gaussian, and we use this as validation for the Gaussian approximation. This approximation is however not valid for long simulation times or for very large variances in the initial distribution.

In conclusion, the approximations that we have made seem appropriate for our purposes. The biggest errors seem to come from the static rotation approximation. However, we do not believe that this small error in the process noise is enough to create any significant bias, especially since we have presented a "worst case" scenario. One possible way to reduce the static rotation error would be to split up the solution interval into multiple smaller parts.

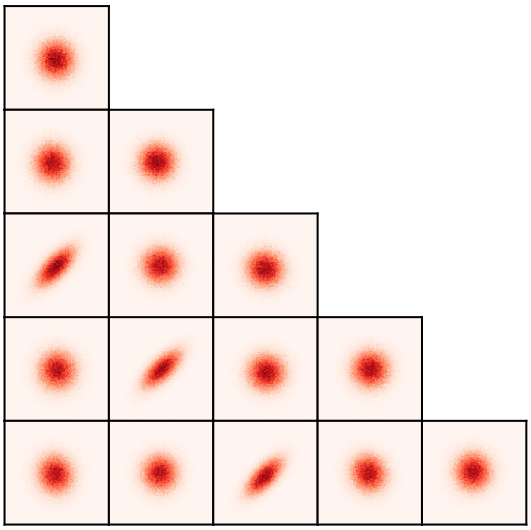


Figure 6.1: Corner plot of distribution obtained by simulation of (5.1) for 1 s with 10000 Euler-Maruyama samples from an initially Gaussian distribution. The distribution remains approximately Gaussian.

6.2.2 Initialization of preconditioning matrix

Before performing any actual sampling, a burn-in or initialization phase is conducted. The purpose of the burn-in phase is mainly to find a suitable preconditioning matrix $\mathbf{L}$. The Zig-Zag sampler is used for the burn-in phase for all experiments. The state is initialized with its true value, i.e., the value that was used for trajectory realization. This is done to avoid the complication of having to select a suitable initial state, however, we suspect that a MAP estimate should work well as an initialization state when this information is not available. The constant bound model is used with a Gaussian prior with mean $\mu = 500$ and standard deviation $\sigma = 500$. The safety factor for the bound is $k = 5$.

A total of 20000 burn-in steps are performed for Case 1, and 40000 steps are used for Case 2, where a "step" corresponds to one loop iteration, or equivalently, one skeleton point. The sampler is first initialized with the identity matrix as the preconditioning matrix. Then, every 1000 steps the preconditioning matrix is updated using an estimate of the posterior covariance matrix calculated from the second half of the thus-far obtained samples (skeleton points).

Figure 6.2 depicts a realization of the burn-in phase for Case 1. Vertical lines indicate steps where the preconditioning matrix has been updated. Notice that after each preconditioning step, the variables traverse the sample space quicker and quicker until it explores the space at a similar speed in all variables. It is particularly noticeable for the first three variables, A, C, and k. This indicates that the burn-in phase has been successful.

In Figure 6.3 corner plots of the posterior distribution are shown for Case 1 and 2. The densities plotted are of the quantities sampled after application of the preconditioning matrix, i.e. $Y = \mathbf{L}^\mathsf{T} X$. If the posterior were a true Gaussian, then Y would

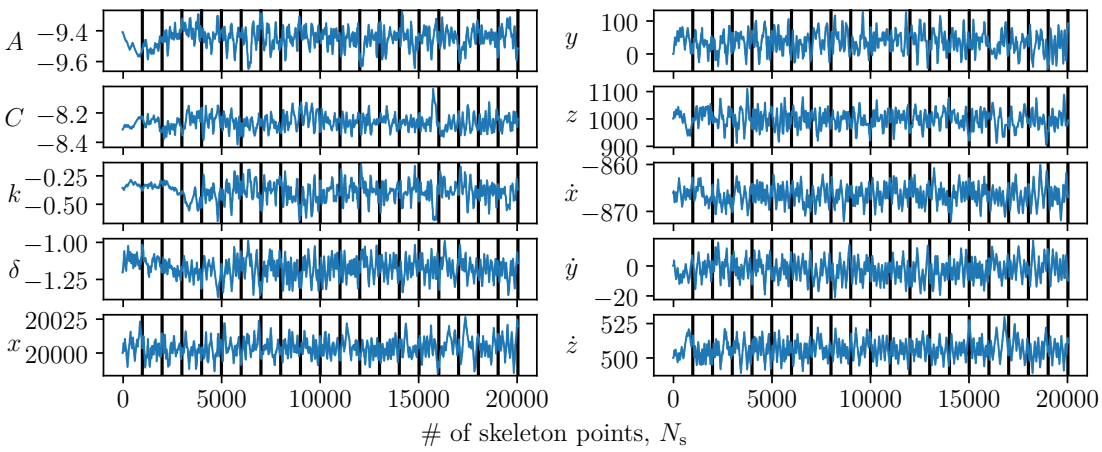
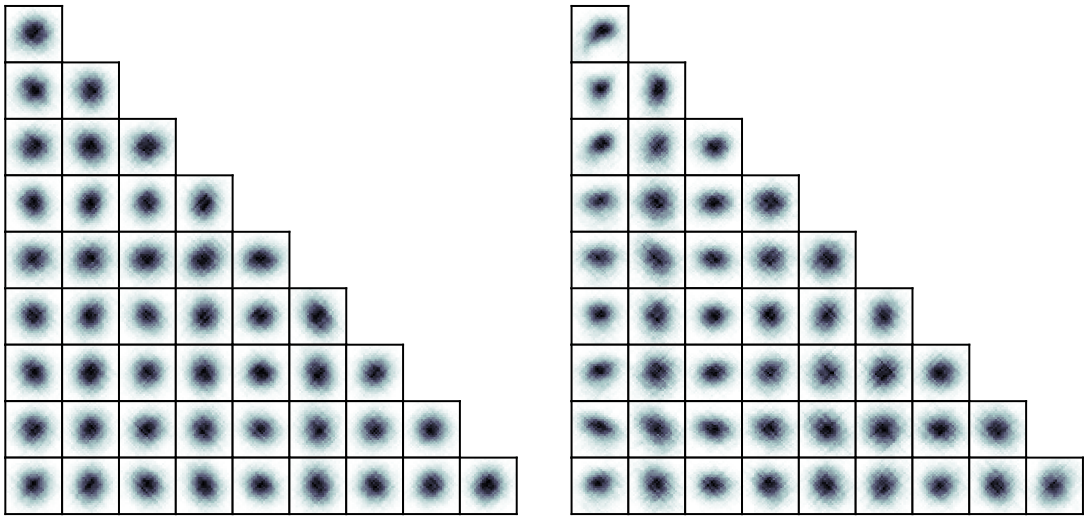


Figure 6.2: Example of the burn-in phase for test case 1. Plotted are the traces for each respective variable. Vertical lines indicate updates of preconditioning matrix, which at each time is estimated from the second half of the thus far obtained data.

have the identity matrix as covariance matrix, and all the covariances would be 1. In order to get a sense of how far off from the ideal Gaussian case the distributions are, the densities are plotted in windows of constant size. We observe that both Case 1 and Case 2 look quite Gaussian. It is a bit difficult to notice, but we can see that in Case 2 the first two variables (top left) seem to have a banana-shaped marginal distribution.

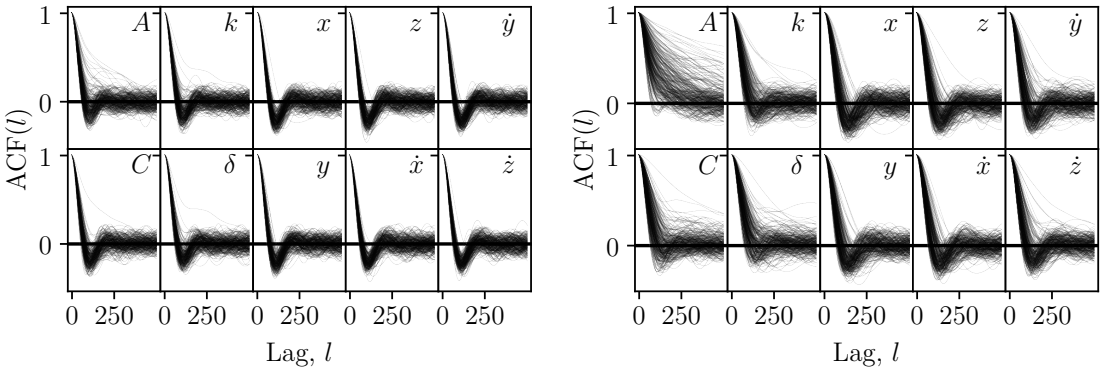


(a) Case 1. **(b)** Case 2.

Figure 6.3: Corner plot of the posterior for the preconditioned variables $Y = \mathbf{L}^{\mathsf{T}} X$ for a realization of Case 1 and Case 2.

In Figure 6.4 the autocorrelation function is shown for a large number of trajectories for both test cases. Notice that the autocorrelation functions indicate longer

correlation lengths, especially for the first variable of test case 2. This is the same variable that we noted had a non-trivial covariance with respect to especially the second variable in Figure 6.3. This indicates that there exists a non-linear effect that the preconditioning was not able to solve. Figure 6.4 also verifies that, apart



(a) Test case 1. **(b)** Test case 2.

Figure 6.4: Autocorrelation functions for each respective variable. Each line corresponds to one of 300 trajectory realizations. The Zig-Zag sampler with constant bounds was used.

from a few outliers, the sampler is able to obtain new independent samples. This means that we can move on to the sampling phase.

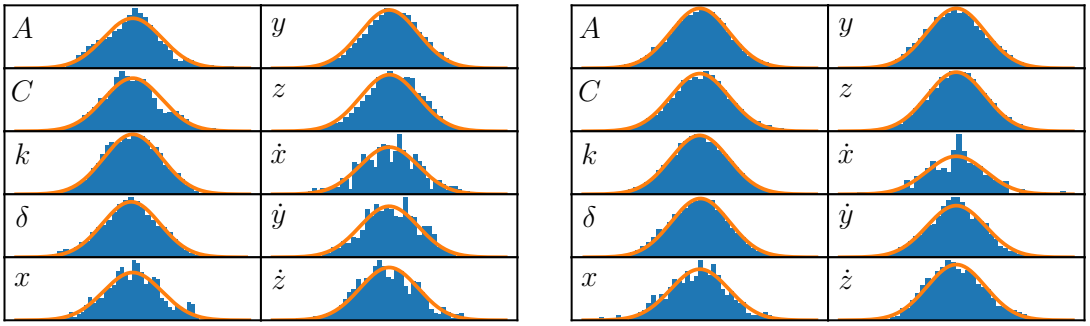
6.2.3 Sampling the posterior

After the burn-in phase has concluded and a preconditioning matrix $\mathbf{L}$ has been learned, the real sampling phase begins. For each configuration of a sampler, we run 300 tests. Each test entails a trajectory realization from the prior distributions defined in Table 5.1, initialization following Section 6.2.2, and then running the sampler for 15000 steps both for Case 1 and 2.

Where the constant bound model is used, the same parameters as during burn-in are used, i.e., $k = 5$ as safety factor, and a Gaussian prior on β_0 with $\mu = 500$ and $\sigma = 500$. The quasi-local bound model is used with the parameters $\gamma = 1.5$ and $\Gamma = 1$. Independent velocity refreshments are used for the bouncy particle sampler with a rate $\lambda_{\text{ref}} = 100$.

6.2.3.1 Prior reconstruction accuracy

Next, we use the methods explained in Section 6.1.1 to verify that all our approximations taken together do not lead to a significant bias. In Figure 6.5 the estimated sum in (6.1) is compared to the prior which was actually sampled, i.e., the Gaussian prior in Table 5.1. There is good agreement between the two for test cases 1 and 2 both, indicating no visible bias. This is also true for the bouncy particle sampler variants which are found in appendix D.1.



(a) Case 1. **(b)** Case 2.

Figure 6.5: Comparison between the prior and its sampled approximation (see Section 6.1.1). A total of 300 trajectory realizations were used and 30 independent samples were selected from each. The Zig-Zag sampler with constant bounds was used.

6.2.3.2 Efficiency

First, we evaluate the samplers in terms of their efficiency. The presented results are for the sampling phase only. The first way in which the efficiency is quantified is by time per effective sample. This metric is calculated by considering the total time taken for the sampling, divided by effective sample size. In table 6.1 and 6.2, the Zig-Zag sampler with constant bounds, the bounce particle sampler with constant bounds, and the bouncy particle sampler with local bounds are compared for test case 1 and 2 respectively. The entries with *best* or *worst* attached correspond to statistics related to the slowest (worst) and fastest (best) variable in each run (not to be confused with the best or worst results overall).

Table 6.1: Case 1. Averages and standard deviations calculated from 300 trajectory realizations.

	ZZ	BPS (const.)	BPS (local)
# skeleton points	15000	15000	15000
# derivatives of Ψ	14999	14970 ± 10	14927 ± 8
# gradients of Ψ	0	1274 ± 212	3039 ± 56
# velocity refreshments	0	29 ± 10	72 ± 8
Acceptance rate	0.1187 ± 0.0029	0.0851 ± 0.0142	**0.2036 ± 0.0038**
Bound violation rate	0.0011 ± 0.0004	0.0003 ± 0.0003	**0.0000 ± 0.0001**
Correlation length (best)	74 ± 5	30 ± 6	**12 ± 1**
Correlation length (worst)	107 ± 54	38 ± 10	**15 ± 3**
Total time	**92 ± 34 s**	127 ± 23 s	370 ± 46 s
Total time / ESS (best)	0.45 ± 0.17 s	**0.25 ± 0.05 s**	0.30 ± 0.00 s
Total time / ESS (worst)	0.66 ± 0.24 s	**0.32 ± 0.06 s**	0.37 ± 0.01 s

Table 6.2: Case 2. Averages and standard deviations calculated from 300 trajectory realizations.

	ZZ	BPS (const.)	BPS (local)
# skeleton points	15000	15000	15000
# derivatives of Ψ	14999	14981 ± 8	14940 ± 10
# gradients of Ψ	0	1098 ± 206	3387 ± 155
# velocity refresh	0	18 ± 8	59 ± 10
Acceptance rate	0.1128 ± 0.0047	0.0733 ± 0.0137	**0.2267 ± 0.0103**
Bound violation rate	0.0027 ± 0.0009	**0.0013 ± 0.0010**	0.0033 ± 0.0023
Correlation length (best)	103 ± 20	49 ± 16	**15 ± 2**
Correlation length (worst)	310 ± 253	92 ± 69	**35 ± 19**
Total time	**91 ± 32 s**	118 ± 24 s	356 ± 43 s
Total time / ESS (best)	0.62 ± 0.22 s	0.39 ± 0.08 s	**0.36 ± 0.04 s**
Total time / ESS (worst)	1.88 ± 0.65 s	**0.72 ± 0.15 s**	0.83 ± 0.10 s

In addition to time per effective sample, the algorithms are compared on the basis of acceptance rate. Here, it seems like the bouncy particle sampler with local bounds performs best. It is able to achieve an acceptance rate above 20% while keeping the bound violation rate at a similar level to the other variants. However, the local model is slower due to having to calculate a second-order derivative at each event proposal while the samplers with constant bounds only need to calculate a first-order derivative. We also observe that bound violation rates seem rather low with our selected bound parameters. Less conservative bounds may lead to higher acceptance rates and improved efficiency, but may at the same time increase bias. The relation between bound violation rate and posterior bias would need to be better understood if more aggressive bound parameters are to be used.

The second way in which performance is evaluated is by the convergence of the expectations defined in Section 6.1.3. In order to evaluate these metrics, a long run of 300000 steps is made with each PDMP variant. Additionally, a run that only samples the Gaussian prior is made. The traces from these runs was batched into 60 and 30 sections respectively for test case 1 and 2. In each batch, the mean and variance are estimated according to (6.2)–(6.3). These estimates are functions of the number of skeleton points used to estimate the quantities. The error in these estimates is quantified by the mean absolute error (MAE), which is compared to the true mean μ_{true} and variance σ^2_{true}. The true mean and variance are calculated using the concatenation of all batches. Figure 6.6 shows an example of how the mean estimate $\hat{\mu}$ converges towards μ_{true} as a function of the number of skeleton points. The variance is calculated analogously. As more skeleton points are included in the estimate of $\hat{\mu}$ and $\hat{\sigma}$, the should converge towards the true values μ_{true} and σ_{true}. We evaluate convergence efficiency of $\hat{\mu}$ based on how many skeleton points are needed for its MAE to go below 20% of the true standard deviation σ_{true}. Similarly, we evaluate convergence efficiency of $\hat{\sigma}$ based on the number of skeleton points needed

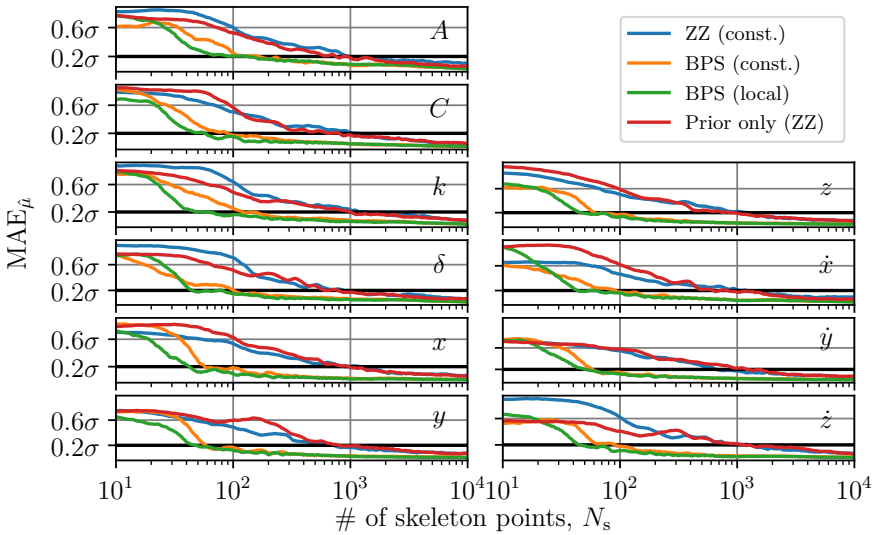


Figure 6.6: MAE of $\hat{\mu}$ for Case 1 shown as a function of skeleton points. The y-axis is scaled with the standard deviation for each respective variable.

for the MAE of $\hat{\sigma}$ to be within 20% of σ_{true}. These results are shown in Table 6.3–6.4.

Table 6.3: Case 1. The number of skeleton points needed for the MAE of the estimated quantities $\hat{\mu}$ and $\hat{\sigma}$ to go below 20% of the true values μ_{true} and σ_{true}. Calculated from on long run batched into 60 pieces.

		ZZ	BPS (const.)	BPS (local)	Prior (ZZ)
$\mathrm{MAE}(\hat{\mu}) < 0.2\sigma_{\text{true}}$	(best)	710 (5.3 s)	60 **(0.6 s)**	46 (1.3 s)	729
	(worst)	1922 (8.8 s)	203 **(1.9 s)**	135 (3.6 s)	1132
$\mathrm{MAE}(\hat{\sigma}) < 0.2\sigma_{\text{true}}$	(best)	231 **(1.7 s)**	342 (3.2 s)	108 (2.9 s)	283
	(worst)	700 **(5.2 s)**	997 (9.2 s)	572 (15.6 s)	448

Table 6.4: Case 2. The number of skeleton points needed for the MAE of the estimated quantities $\hat{\mu}$ and $\hat{\sigma}$ to go below 20% of the true values μ_{true} and σ_{true}. Calculated from on long run batched into 30 pieces.

		ZZ	BPS (const.)	BPS (local)	Prior (ZZ)
$\mathrm{MAE}(\hat{\mu}) < 0.2\sigma_{\text{true}}$	(best)	898 (6.4 s)	105 (1.1 s)	45 **(1.0 s)**	729
	(worst)	3308 (23.6 s)	801 (8.3 s)	79 **(1.8 s)**	1132
$\mathrm{MAE}(\hat{\sigma}) < 0.2\sigma_{\text{true}}$	(best)	439 (3.1 s)	226 **(2.4 s)**	193(4.3 s)	283
	(worst)	3138 (22.4 s)	933 **(10.3 s)**	905 (20.2 s)	448

From this data, we make some interesting observations. First, we notice that the means converge quite quickly while the variance takes longer to converge. This is

especially true for the bouncy particle sampler variants, where the mean convergence time is comparable to the time taken to generate a new independent sample. For the variance, we see that the Zig-Zag sampler is the fastest for test case 1. One possible explanation for this is that the refreshment rate is too low for the bouncy particle sampler. With a low refreshment rate, the particle bounces in "circles" (as exemplified in Figure 3.2). It therefore makes sense that the mean would converge quickly, whereas the variance would take longer to converge, due to the refreshments events that are needed to explore the full space.

6.3 Acceleration jump-drag model

In this section, we use the sticky Zig-Zag sampler to find jump times and magnitudes in a realization of the acceleration jump-drag SDE (5.2), i.e., sampling the posterior defined by equation (5.3). We use the specific trajectory generated in Figure 5.5 to evaluate the PDMP sampling strategy outlined in Section 5.3. The specific trajectory realization is chosen because it contains multiple jumps of varying magnitude, two of which are very close in time.

In section 6.2.1 we describe and analyze the approximations made in the likelihood evaluation. For the acceleration jump-drag model, an important challenge is how to find a good initial state, this is done during a burn-in phase presented in Section 6.3.2. Finally, the posterior is sampled and the results are presented in Section 6.3.3.

6.3.1 Approximate likelihood evaluation

Similarly to the ballistic model, the likelihood must be evaluated in an approximate manner and we next describe how this is done and validate some of the approximations which it entails. The likelihood in the acceleration jump-drag model is $p(Z_{0:N}|(\tau_i, \Delta_i)_{i\in[1..K]}, X_0)$, i.e., the probability of the data, given jump times, jump magnitudes and the initial state. Jump times and magnitudes are part of the state space, while the initial state X_0 is not. The initial state X_0 is instead taken to be the first measurement Z_0 and with 0 acceleration. Likelihood evaluation is done with an unscented Kalman filter which has been adapted to handle jumps following Section 3.4.1.2. Since we have conditioned on jumps, the system dynamics are given by the jump-free part of the acceleration jump-drag SDE (5.2). This SDE is solved approximately with the noise linearization method presented in 2.5.2.2. The analytic expressions obtained for the covariance can be found in Appendix C.3. The RK4 ODE solver is used with a number of steps N calculated as

$$N = 1 + \left\lfloor \frac{Av^2\Delta T}{200} \right\rfloor$$

in order to avoid large errors for large velocities.

We next discuss some of the approximations that this likelihood evaluation entails, beginning with the Gaussian assumption of the UKF. To get an idea about this approximation we again propagate an initially Gaussian state through the nonlinearity (jump-free acceleration jump-drag SDE) and observe the resulting distribution. The

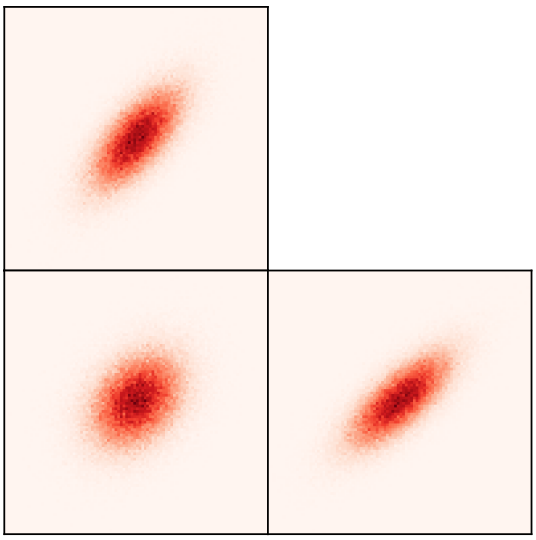


Figure 6.7: Corner plot of the initial distribution propagated 1 s with 100000 Euler-Maruyama samples, resulting in a new distribution.

initial distribution has mean $[x, \dot{x}, \ddot{x}] = [0, 1000, 0]$, and each variable is uncorrelated with a standard deviation of 100. A total of 100000 particles are propagated 1 s forwards with Euler-Maruyama using $\delta t = 1$ ms. The propagated density is shown in Figure 6.7. The obtained distribution looks roughly Gaussian.

The noise linearization approximation is tested in a similar fashion. We use the same initial state $[x, \dot{x}, \ddot{x}] = [0, 1000, 0]$ and propagate it forward 1 second with both Euler-Maruyama and the noise linearization approach. The distribution at the next time is obtained using 10000 Euler-Maruyama simulations with a time-step of 0.1 ms has mean μ_{EM} and covariance Σ_{EM} given as

$$\mu_{\text{EM}} = \begin{pmatrix} 961.282 \\ 924.529 \\ -0.001 \end{pmatrix}, \quad \Sigma_{\text{EM}} = \begin{pmatrix} 0.0454 & 0.1119 & 0.1598 \\ \text{sym.} & 0.2953 & 0.4737 \\ \text{sym.} & \text{sym.} & 0.9968 \end{pmatrix}.$$

Due to the fine discretization and large number of simulations we can use this as a benchmark. Using the noise linearization method, the distribution at the next time has mean μ_{approx} and covariance Σ_{approx} given as

$$\mu_{\text{approx}} = \begin{pmatrix} 961.274 \\ 924.528 \\ 0 \end{pmatrix}, \quad \Sigma_{\text{approx}} = \begin{pmatrix} 0.0457 & 0.1123 & 0.1601 \\ \text{sym.} & 0.2955 & 0.4739 \\ \text{sym.} & \text{sym.} & 1.0000 \end{pmatrix}.$$

We see that the covariance matrices look very similar and the mean approximation by the ODE solver is very good, giving an error much smaller than one standard deviation. Having verified that our approximations are sound, we move on to sampling the posterior.

6.3.2 Initialization of state

The purpose of the burn-in phase for the acceleration jump-drag model is mainly to find a "good" initialization state, i.e., a set of jumps that explain the data well. Initialization is done with the sticky Zig-Zag sampler using the constant bound model. The Gaussian prior on β_0 is selected with mean μ at 500 and a standard deviation σ of 500. A safety factor $k = 5$ was used in the constant bound model. We use 20 variable pairs to represent possible jumps which results in a 40-dimensional sampling space. The problem is not preconditioned in the sense explained in Section 4.2. The principal reason for this is that the jump posterior is highly multimodal, as any time-magnitude variable pair may explain any of the jumps, and a Gaussian approximation would therefore be nonsensical. Instead, the velocities of the time variables in the sticky Zig-Zag sampler are scaled with a factor 0.01, as that is found to yield better results.

The burn-in phase consists of 10^5 steps, which take 189 s to perform. The average acceptance rate during the burn-in phase was 5.3% and the bound violation rate was 0.05%. The obtained trace is shown along with the locations of the true jumps in Figure 6.8. We see that the sampler is able to recover the jump times and magnitudes of the real jumps from an initially jump-less state. It is interesting to note that some jumps seem to be explored by different variable pairs (presumably at different times, although this cannot be seen in this figure). This highlights the fact that each pair is able to represent any of the jumps. The line at 0 jump magnitude shows that most of the variables are in the frozen state with magnitude 0. Unfreezing events can also be seen in various places close to this line, but in most cases, the variable is quickly reflected back to a frozen state again.

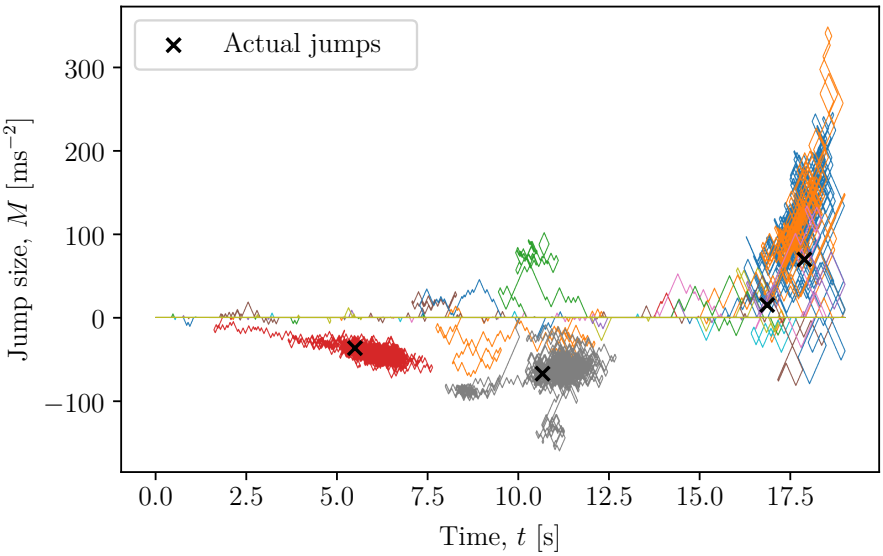


Figure 6.8: Trace of the burn-in phase. Each color corresponds to a distinct variable pair.

6.3.3 Sampling the posterior

When the burn-in phase is completed, a real sampling phase consisting of 10^5 steps can begin. The sampler is initialized in the last state obtained from the burn-in phase and the time variables are once again scaled with a factor 0.01. The bound model is also reset and the parameters used are the same as during burn-in.

Due to the strong multimodality of the posterior for each variable pair, achieving convergence in any strict sense of the word is not feasible. What we really care about is some weak sense of convergence, where the full *set* of variable pairs, taken together, converge towards something. It does not matter whether it is the first or last variable pair that explains a particular jump, only that a jump is represented in the estimate. We visualize the posterior by a superposition of jump time and magnitude pairs in Figure 6.9. Jumps of size 0 are not visualized. The sampling

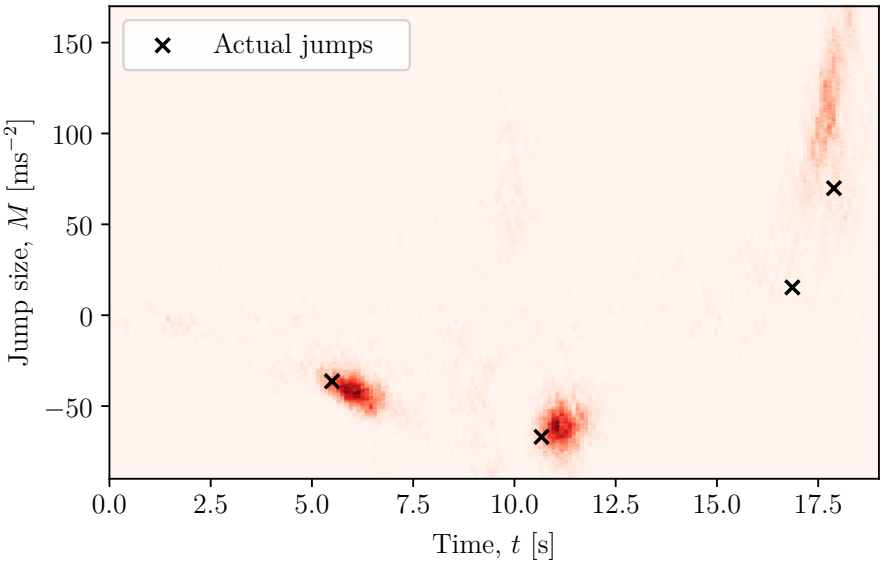


Figure 6.9: Obtained posterior for each variable pair plotted in the same graph.

phase took 202 s, had an acceptance rate of 6.0% and a bound violation rate of 0.07%. We observe that the markers indicating true jumps seem to agree quite well with the posterior for the first two jumps, while the sampler seems to prefer one large jump to explain the third and fourth jumps. This makes sense since they are quite close to each other in time. We also note the large uncertainty of the magnitude of this last jump. This is to be expected since the likelihood penalty of introducing a large jump at the end is low since this jump only affects the data which comes after it. By manually comparing the likelihoods of conditioning on the true jumps, and the suggested jumps by the sampler, it turns out that the likelihood is actually higher for the merged jumps, so it is not surprising that the sampler prefers this solution.

6.3.3.1 Filtering conditioned on jumps

The filter can now be run conditioned on the jumps obtained from the sampling phase. This is done in Figure 6.10. The filtered solution conditioned on the jumps is much closer to the truth than the unconditioned solution we had in Figure 5.5. This indicates that we can use the samples obtained to perform conditioned filtering.

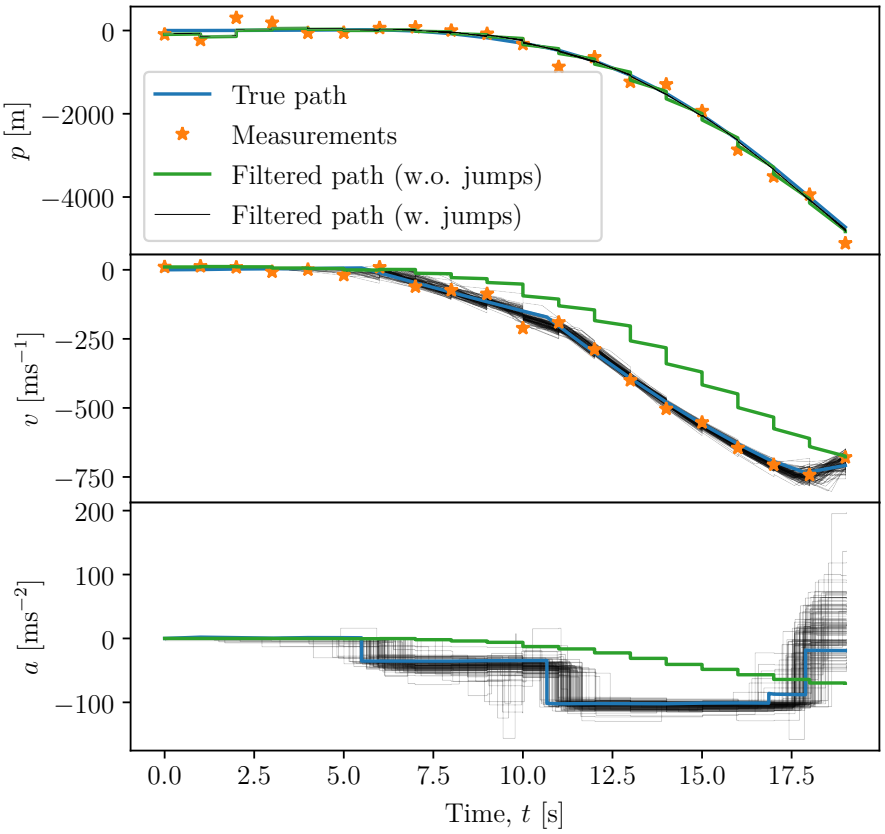


Figure 6.10: Filtering conditioned jumps (every 1000th skeleton point from the sampling trace) compared to the unconditioned filter.

7

Discussion

Throughout the thesis, we have made various approximations guided by achieving reasonably good accuracy. However, it may be the case that the soft quantities (expected values) that we are interested in do not require this level of accuracy. This opens up the discussion about how performance can be improved further.

In the following chapter, a discussion is held regarding approximations and other choices made in the thesis. In Section 7.1 the choice of state space is discussed. In Section 7.2 various ways to achieve speed increases including SDE approximation and choice of filtering algorithm are discussed. With the ultimate goal of achieving fast sampling, each approximation considered is a trade-off between speed and accuracy. Finally in Section 7.3 we suggest some areas of future research.

7.1 State definition

Throughout this thesis, we have selected to marginalize the target states for most of the measurement sequence and evaluate the likelihood with a filter. Filtering involves approximations and can make likelihood evaluation computationally heavy. An alternative approach is to include target states at *all* measurement times in the sampling space, i.e., sample the full posterior defined by (3.12). This would lead to a much larger sampling space as it scales with the number of measurements, but likelihood evaluation would be fast as no filtering is needed. Additionally, the likelihood would be sparse which means that partial derivatives with respect to state variables only depend on a subset of the data, i.e., state measurements of the state variable itself and its neighbors. This property can be exploited by the Zig-Zag sampler to make likelihood evaluation even faster [1]. Unfortunately, we found this approach to be impractical due to problematic correlations between states separated in time. In one dimension the correlations were manageable, but in three dimensions we were unable to achieve convergence. For this reason, we believe that the filtering approach used in this thesis is a better option.

7.2 Achieving fast Bayesian inference

An important goal of this thesis is to achieve *fast* sampling. Our results are promising, but there is yet a performance gap before the methods can be used in an online setting. The majority of the sampling time is spent evaluating (derivatives of) the

likelihood, and any choices made therein, therefore, have the potential to impact the sampling speed in a drastic way. We therefore next discuss possible choices that would enable faster likelihood evaluation.

In terms of time, a large part of the likelihood evaluation is spent in the prediction stage of the UKF, propagating σ-points through the non-linearity. This corresponds to solving an ODE with the RK4 method. However, it may be the case that there are regimes where it suffices with a lower-order method. If that is the case, then the computational time can be reduced here.

The filter evaluates process noise covariance using the noise linearization approach in Section 2.5.2.2. This approach seems fast, but alternatives could be explored. Many alternatives are available in [6]. It would be interesting to test how well a simple SDE linearization performs, replacing both the time-intensive Runge-Kutta solver and the calculation of the process noise covariance. This seems like a rough approximation, and if the accuracy is not satisfactory using this method, one could perhaps afford to reduce the time step of the approximation by introducing intermediate steps between measurements. Another possible way to cut down on the computation time is to use a cheaper filtering algorithm, such as the extended Kalman filter.

Another way to reduce computation time is to reduce the *number* of likelihood evaluation needed. The number of evaluations needed is related to which PDMP variant is used, and with what computational bounds. From the results in Section 6.2 we noted that bouncy particle sampler variants generally perform better than the Zig-Zag sampler for the ballistic model. Here, it would be interesting to extend the analysis to include the Boomerang sampler, which is another example of a PDMP which has been claimed to outperform the bouncy particle sampler in some cases [22].

There is also the possibility of tuning bound models. As discussed in Section 4.1.1, the parameters of approximate bound models should be optimized to achieve a trade-off between a high acceptance rate and a low bound violation rate. If the acceptance rate can be improved without introducing significant bias in the quantities we are interested in, then performance can be improved due to the reduced number of likelihood evaluations.

We also note that there are likely cases where rough approximations during the burn-in phase are acceptable. One example where this could be relevant is for the acceleration jump-drag model, where rough approximations during the burn-in phase seem acceptable since we are only looking to find a reasonable initial state.

We also make a note on code implementation. All experiments have been carried out using the Julia programming language. It is highly probable that implementation in a lower-level language, with low-level code optimization in mind, will give a considerable performance bump. We also note that multiple instances of the Zig-Zag algorithm can be run in parallel to obtain independent samples more quickly, given that the user has access to multiple processing cores.

7.3 Future work

There are multiple ways to build on the work from this thesis. First, it would be interesting to implement and test some of the suggestions to improve sampling speed from Section 7.2. Here, we also add that another way to speed up sampling could be a surrogate likelihood-gradient method with some neural net to approximate the filter. This has the advantage that the time complexity does not scale with the number of measurements.

Another possible area of future work is to develop target and measurement models which mimic reality more closely. Improvements of target models could include modeling of fuel and engine, use of non-isotropic or multiplicative process noise, or an atmospheric model where air density is a function of height to name a few. The measurement model could be improved by letting the measurement noise be a function of the state. For a radar sensor, this is particularly relevant as the measurement uncertainties are dependent on the distance to the target. More complex models may be a better representation of reality than simpler ones, but may also lead to a more time-consuming likelihood evaluation. Mitigating this such that models do not become practically unfeasible should be a core part of future work in this area.

Another interesting area of future work is to continue the work on jump sampling. We have shown that our binomial approximation approach is feasible for one realization of a particular SDE, but it would be interesting to extend this to different SDEs and also investigate the different properties of the approach (such as its accuracy). Any work made in this area should probably involve the development of new bound models that are better adapted to the multimodal nature of the posterior, as the constant bound model used here achieves a relatively low acceptance rate. For the purpose of missile tracking, it would be particularly interesting to take the acceleration jump-drag model to three dimensions. This would significantly extend the envelope of possible threats that we are able to investigate.

8

Conclusion

In this thesis, we set out to explore how PDMP samplers can be employed to sample Bayesian posteriors related to the state estimation problem of an adversarial missile. We find a method that is able to achieve this with reasonable efficiency, utilizing a combination of approximate bounds and preconditioning. We also find that automatic differentiation is useful for taking derivatives in an efficient and fool-proof way.

In terms of results, two classes of targets are explored. The first is the class of objects on a ballistic trajectory, modeled with a diffusion SDE. For this model, our results indicate that the bouncy particle sampler samples the posterior more effectively than the Zig-Zag sampler both in terms of time per independent sample and the time needed for the convergence of expectations. The performance achieved is promising and shows that PDMP methods are a viable option to obtain these distributions. However, the performance is still prohibitive for online applications. We suggest multiple possible ways in which the sampling could be made faster, including code optimizations, more aggressive approximations for the SDE and likelihood evaluation, and improvements to the sampling scheme in terms of bound model and PDMP dynamics. It seems plausible that with these optimizations taken together, or the use of a surrogate likelihood-gradient method, PDMPs can be realistic for use in online settings.

The second class of targets explored are targets that may undergo jumps in their acceleration state, modeled with a jump-diffusion SDE. We develop a method to sample jump times and magnitudes from realizations of jump-diffusion processes with the sticky Zig-Zag sampler based on a binomial approximation. We find that for our jump-diffusion SDE, we can obtain good estimates of jump times and magnitudes with this method. Exploring this method further and for a wider range of jump-diffusion SDEs would be an interesting area of further research.

Bibliography

[1] J. Bierkens, P. Fearnhead, and G. Roberts, "The zig-zag process and super-efficient sampling for bayesian analysis of big data," *Annals of Statistics*, vol. 47, Jul. 2016. DOI: 10.1214/18-AOS1715.

[2] M. Davis, *Markov Models & Optimization* (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). Taylor & Francis, 1993, ISBN: 9780412314100.

[3] T. Liggett, *Continuous Time Markov Processes: An Introduction* (Graduate studies in mathematics). American Mathematical Society, 2010, ISBN: 9780821849491.

[4] F. Klebaner, *Introduction to Stochastic Calculus with Applications* (Introduction to Stochastic Calculus with Applications). Imperial College Press, 2005, ISBN: 9781860945557.

[5] D. Applebaum, B. Bollobas, U. of Cambridge, *et al.*, *Lévy Processes and Stochastic Calculus* (Cambridge Studies in Advanced Mathematics). Cambridge University Press, 2004, ISBN: 9780521832632.

[6] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations* (Institute of Mathematical Statistics Textbooks). Cambridge University Press, 2019, ISBN: 9781108693448.

[7] S. Brooks, A. Gelman, G. Jones, and X. Meng, *Handbook of Markov Chain Monte Carlo* (Chapman & Hall/CRC Handbooks of Modern Statistical Methods). CRC Press, 2011, ISBN: 9781420079425.

[8] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987, ISSN: 0370-2693. DOI: 10.1016/0370-2693(87)91197-X.

[9] G. Casella and E. George, "Explaning the gibbs sampler," *The American Statistician*, vol. 46, pp. 167–174, Aug. 1992. DOI: 10.1080/00031305.1992.10475878.

[10] J. Bierkens, G. O. Roberts, and P.-A. Zitt, "Ergodicity of the zigzag process," *The Annals of Applied Probability*, vol. 29, no. 4, Aug. 2019. DOI: 10.1214/18-aap1453.

[11] J. Bierkens, S. Grazzi, F. van der Meulen, and M. Schauer, "Sticky pdmp samplers for sparse and local inference problems," *Statistics and Computing*, vol. 33, 2021.

[12] M. Vincent, "Bayesian inverse problems with neural generative priors," M.S. thesis, Chalmers University, 2022.

[13] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960, ISSN: 0021-9223. DOI: 10.1115/1.3662552.

[14] S. Sarkka, "On unscented kalman filtering for state estimation of continuous-time nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1631–1641, 2007. DOI: 10.1109/TAC.2007.904453.

[15] P. A. W. Lewis and G. S. Shedler, "Simulation of nonhomogeneous poisson processes by thinning," *Naval Research Logistics Quarterly*, vol. 26, no. 3, pp. 403–413, 1979. DOI: https://doi.org/10.1002/nav.3800260304.

[16] A. Pakman, D. Gilboa, D. Carlson, and L. Paninski, "Stochastic bouncy particle sampler," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Aug. 2017, pp. 2741–2750.

[17] M. Schauer, S. Grazzi, and C. Scherrer, "Mschauer/zigzagboomerang.jl: V0.13.1," Aug. 2022. DOI: 10.5281/zenodo.7038467.

[18] A. Corbella, S. E. F. Spencer, and G. O. Roberts, *Automatic zig-zag sampling in practice*, 2022. DOI: 10.48550/ARXIV.2206.11410.

[19] A. Bertazzi and J. Bierkens, *Adaptive schemes for piecewise deterministic monte carlo algorithms*, 2020. DOI: 10.48550/ARXIV.2012.13924.

[20] A. Griewank *et al.*, "On automatic differentiation," *Mathematical Programming: recent developments and applications*, vol. 6, no. 6, pp. 83–107, 1989.

[21] P. Krus and A. Abdallah, "Modelling of transonic and supersonic aerodynamics for conceptual design and flight simulation," Oct. 2019, pp. 30–34. DOI: 10.3384/ecp19162003.

[22] J. Bierkens, S. Grazzi, K. Kamatani, and G. O. Roberts, "The boomerang sampler," *ArXiv*, vol. abs/2006.13777, 2020.

[23] S. Shreve, *Stochastic Calculus for Finance II: Continuous-Time Models* (Springer Finance Textbooks v. 11). Springer, 2004, ISBN: 9780387401010.

[24] G. Folland, *Real Analysis: Modern Techniques and Their Applications* (A Wiley-Interscience publication). Wiley, 1999, ISBN: 9780471317166.

A

Theory

The aim of this chapter is to give the reader an elementary introduction to, or a refreshment of, concepts from measure theory and stochastic calculus needed for the thesis. The presentation is intended to define the needed constructions and give some level of intuition about them, without going into detail and only stating the needed results. For a more complete theory, the reader is suggested to consult [23] for a pleasant introduction, or [4] and [24] for a deeper treatment.

A.1 Measure-theoretic probability theory

Definition A.1.1. *Let $\mathcal{P}(X)$ be the power-set (the set of all subsets) of some abstract set X. Then, $\mathcal{A} \subseteq \mathcal{P}(X)$ is a σ-algebra if it satisfies:*

1. $\Omega \in \mathcal{A}$,
2. *If* $A \in \mathcal{A}$, *then* $A^{\mathsf{c}} \in \mathcal{A}$,
3. *If* $A_1, A_2, \cdots \in \mathcal{A}$, *then* $\bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$.

An element $A \in \mathcal{A}$ is called a *measurable set*.

In general, it is possible to construct many σ-algebras from the same set X. For example, the power-set $\mathcal{P}(X)$ is always a σ-algebra (the largest), so is $\{\varnothing, X\}$ (the smallest). Usually, we would want to have as large of a σ-algebra as possible, so that we can measure as much as possible (we want $\mathcal{A}$ to contain many measurable sets). However, it turns out that the power set itself is too large to construct a useful integration theory. This is connected to the Banach-Tarsky paradox, which (assuming the axiom of choice) implies the existence of sets with non-unique volume, and in turn, pathological non-unique integrals. A rich σ-algebra whose sets have unambiguous volume is the Borel σ-algebra. While smaller than the power set, it contains all the sets whose volumes we intuitively would be interested in. We recall that a topological space is a set in which the open sets (the topology) are defined.

Definition A.1.2. *For a topological space E, the Borel σ-algebra $\mathcal{B}(E)$ is the σ-algebra that is generated by the open sets of E, i.e., $\mathcal{B}(\Omega)$ is the smallest σ-algebra that contains all open sets of E.*

The construction of $\mathcal{B}(E)$ can be thought of in the following way: First include all of the open sets of E, and then adds all of the sets (but no more) needed in order to satisfy the requirement that they constitute a σ-algebra.

The tuple $(X, \mathcal{A})$ is called a *measurable space*. A *measure* is defined on a measurable space. A measure is a generalization of the concept of length, area, or volume to abstract sets X. The normal generalized volume measure is called the *Lebesgue measure* and is ordinarily denoted μ_0, but we can also consider others like the trivial measure (assigns 0 to every set), counting measure (count number of elements) or the Dirac measure (1 only when a specific element is in the set). The formal definition of a measure is given below:

Definition A.1.3. *A measure is a set function $\mu : \mathcal{A} \to [0, \infty]$ on a measurable space $(X, \mathcal{A})$ that satisfies the following:*

1. $\mu(\varnothing) = 0$,
2. $\mu\Big(\bigcup_{i=1}^{\infty} A_i\Big) = \sum_{i=1}^{\infty} \mu(A_i)$, *for all* $A_1, A_2, \cdots \in \mathcal{A} : A_j \cap A_k = \varnothing, j \neq k$.

A measurable space $(X, \mathcal{A})$ can be *equipped* with a measure and is then called a *measure space* (note the difference between measurable and measure, only the latter has a measure attached). It is usually written as a triplet $(X, \mathcal{A}, \mu)$.

Definition A.1.4. *A measure μ on a measurable space $(X, \mathcal{A})$ is a probability measure if it additionally satisfies:*

1. $\mu(X) = 1$.

A *probability space* is a measure space, often denoted $(\Omega, \mathcal{F}, \mathbf{P})$, where the measure $\mathbf{P}$ is a probability measure. For probability spaces, we interpret Ω as the set of all possible outcomes of a trial. Ω is called the *sample space*. The σ-algebra $\mathcal{F}$ is called the *event space* and the set of *events* that we may consider. One interpretation is that $\omega \in \Omega$ is a *seed*, that gives rise to a set of events. Using a dice as an example, Ω corresponds to the outcomes $\{1, 2, 3, 4, 5, 6\}$, while the events are things like: *the outcome is a prime number* or *the outcome is 4*.

Given a measure space $(X, \mathcal{A}, \mu)$ we next outline the construction of the integral: A function $f : X \to \mathbb{R}$ is called *simple* if it has the form $f(x) = \sum_{k=1}^{N} \alpha_k \mathbf{1}_{A_k}(x)$, where $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$ and $A_1, \ldots, A_N \in \mathcal{A}$ are disjoint sets. For a simple function, an integral is defined with respect to the measure μ as $\int f(x)\mu(dx) = \sum_{k=1}^{N} \alpha_k \mu(A_k)$. If $f : X \to \mathbb{R}$ is not simple but sufficiently regular, e.g., continuous, then $\int f(x)\mu(dx) = \lim_{k\to\infty} \int f_k(x)\mu(dx)$, where f_k are simple functions such that $f_k \to f$ as $k \to \infty$ in the suitable sense. The most general class of functions for which this holds are called *measurable functions*. More precisely, $f \colon (X, \mathcal{A}) \to (R, \mathcal{B}(R))$ is measurable if for every set $B \in \mathcal{B}(R)$ the set $A = \{x \in X : f(x) \in B\}$ belongs to $\mathcal{A}$. In the case of probability spaces we call a function $Y : X \to \mathbb{R}$ a random variable if it is measurable. On probability spaces integration corresponds to expectation, and since random variables are measurable, their expectations are well-defined (but possibly infinite).

One of the advantages of using measures in the theory of probability is that the outcomes can be of both discrete (atomic) and non-discrete (non-atomic) nature. It is straightforward to define a measure for a random variable that takes the value 0 with probability 0.5 and is taken from $\mathcal{N}(0, 1)$ with probability 0.5. In this case, we

call 0 an *atom*. The measure for this example reads

$$\mu(\mathrm{d}x) = \frac{1}{2}\frac{1}{\sqrt{2\pi}}\exp(-x^2/2)\mathrm{d}x + \frac{1}{2}\delta_0(\mathrm{d}x),$$

where δ is the *Dirac measure* defined as

$$\delta_x(A) = \mathbb{1}_A(x) = \begin{cases} 1, & x \in A, \\ 0, & x \notin A, \end{cases} \qquad A \in \mathcal{F}.$$

It is possible to define multiple measures on the same space. We have already touched on the Lebesgue measure and probability measures. The Radon-Nikodym theorem allows us to relate probability measures to each other.

Theorem A.1.1 (Radon–Nikodym theorem)**.** *Let $(\Omega, \mathcal{F})$ be a measurable space with two σ-finite*[1] *measures μ and ν. If ν is absolutely continuous*[2] *with respect to μ, then there exists a function $f : X \to [0, \infty)$, such that*

$$\nu(A) = \int_A f\mathrm{d}\mu, \text{ for any } A \subseteq \mathcal{F}.$$

The function f is called the *Radon-Nikodym derivative* and is often written $\mathrm{d}\nu/\mathrm{d}\mu$. The Radon–Nikodym theorem makes the connection between measures and probability distributions clear. For a probability measure $\mu_p(A)$ and the Lebesgue measure $\mu_0(\mathrm{d}x) = \mathrm{d}x$ for some $A \in \mathcal{F}$ we have the relation

$$\mu_p(A) = \int_A \mathrm{d}\mu_p = \int_A \frac{\mathrm{d}\mu_p}{\mathrm{d}\mu_0}\mathrm{d}\mu_0 = \int_A p(x)\mathrm{d}x$$

where $p(\cdot)$ is the probability density for the measure $\mu_p(A)$.

A.2 Stochastic processes

Definition A.2.1. *Given a probability space $(\Omega, \mathcal{F}, \mathbf{P})$, an ordered index set T and a measurable state space (S, Σ), a stochastic process is defined as a measurable function $X : \Omega \times T \to S$. The stochastic process is often denoted $X = (X_t)_{t\in T}$.*

In this thesis, we assume $T = \mathbb{R}_+$. Since X depends on $\omega \in \Omega$, we say that each ω corresponds to a *realization* of the process. A more intuitive interpretation may be obtained if we include ω explicitly: $X(\omega) = (X_t(\omega))_{t\in T}$. We can then consider $X_t(\omega)$ a function of t for each ω.

In order to gain some intuition, let us consider Brownian motion (done in more detail below). In this case, Ω would be something like the set of all continuous functions with $f(0) = 0$, T would be $\mathbb{R}^+$ and S would be $\mathbb{R}$. However, when we work with actual processes we usually do not need to fully specify the probability space etc., but define the properties of the process in (S, Σ)-space.

We now turn our attention to the important concept of filtrations.

[1] A measure μ is called σ-finite if $\mu(A) < \infty$, for any $A \in \mathcal{A}$.

[2] The measure ν is absolutely continuous with respect to μ if $\mu(A) = 0$ implies $\nu(A) = 0$.

Definition A.2.2. *Given a measure space $(\Omega, \mathcal{F}, \mathbf{P})$, a filtration $(\mathcal{F}_t)_{t\geq 0}$ is an increasing collection of sub-σ-algebras of $\mathcal{F}$, indexed by $t \geq 0$. Thus for each $0 \leq s \leq t$ we have $\mathcal{F}_s \subseteq \mathcal{F}_t$.*

The space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbf{P})$ is referred to a *filtered probability space*. Filtrations are an important concept for stochastic processes. Let X_t denote some stochastic process. The most important filtration is the *natural filtration* $\mathcal{F}_t^X = \sigma(X_s; 0 \leq s \leq t)$. It is the smallest σ-algebra making all the random variables X_s, $s \in [0, t]$ measurable.

Filtrations should be understood as information. For a stochastic process X_t adapted to a filtration $(\mathcal{F}_t)_{t\geq 0}$, the σ-algebra $\mathcal{F}_t$ contains all the information about X_s for any $s \leq t$. Without presenting the technical definition of conditional expectations, it holds for a $\mathcal{F}_t$-measurable stochastic process X that

$$\mathbf{E}[X_t|\mathcal{F}_t] = X_t, \text{ for all } t \geq 0.$$

Since $F_s \subseteq F_t$ for $s \leq t$ it also holds that

$$\mathbf{E}[X_s|\mathcal{F}_t] = X_s, s \leq t.$$

The natural filtration $\mathcal{F}^X$ can thus be seen as an entity that accumulates information about the history of the process. Finally, we say that a stochastic process is adapted to a filtration $\mathcal{F}$ if X_t is $\mathcal{F}_t$-measurable for every $t \geq 0$.

An important class of processes is *martingales*. These are processes that have no tendency to either rise or fall. We next present the formal definition.

Definition A.2.3. *A stochastic process X_t adapted to a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, P)$ is called a martingale if*

$$\mathbf{E}[X_t|\mathcal{F}_s] = X_s.$$

Let us now give the definition of a *stopping time*.

Definition A.2.4. *Let $\tau : \Omega \to \mathbb{R}_+$ be a random variable on the filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, P)$. τ is said to be a stopping time if*

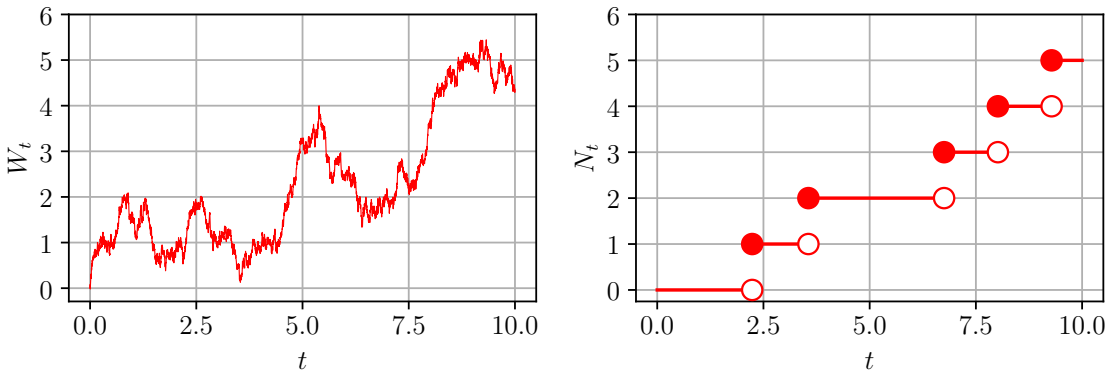
$$\{\tau < t\} \in \mathcal{F}_t \text{ for each } t \geq 0.$$

Intuitively a stopping time can be thought of as a random variable for which one can answer the question: given the information up until this point, has the stopping time happened or not? One example of a stopping time is the first time that a process reaches a particular value, say the first time $X_t \geq 10$. Depending on the process, this may happen relatively quickly, or perhaps never.

Next, we define (roughly) the concept of a *local martingale* using stopping times. An adapted stochastic process X on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, P)$ is said to be a local martingale if there exists a sequence of increasing stopping times τ_n such that $\lim_{n\to\infty} \tau_n = \infty$ and that for every n, $X_{\min(t,\tau_n)}$ is a martingale.

A.2.1 Brownian motion and the Poisson process

In this thesis, two processes are of fundamental importance. They are Brownian motion and the Poisson process. These two are of very different character, as can be seen in Figure A.1. Brownian motion (sometimes called the Wiener process) is continuous, but with small changes happening all the time, whereas the Poisson process stays constant for longer periods of time, until a discontinuous jump happens. The Poisson process is one example of a so-called *jump process*.



(a) **(b)**

Figure A.1: Brownian motion in (a) and the Poisson process in (b).

Let us now define Brownian motion formally.

Definition A.2.5. *A stochastic process $(W_t)_{t\geq 0}$ is a Brownian motion if it satisfies:*

1. $W_0 = 0$,
2. *W has independent increments, i.e., $W_t - W_s$ is independent of $W_s - W_r$ for any $0 \leq r < s < t$,*
3. $W_{t+s} - W_t \sim \mathcal{N}(0, s)$,
4. *W_t is continuous in t with probability 1.*

Let us also give the formal definition of the Poisson process.

Definition A.2.6. *The counting process $\{N_t, t \geq 0\}$ is a Poisson process with rate Λ if it satisfies:*

1. $N_0 = 0$,
2. *N_t has independent increments, i.e. $N_t - N_s$ is independent of $N_s - N_r$ for any $0 \leq r < s < t$,*
3. *N is integer-valued and the number of jumps in a time period of length τ is Poisson$(\Lambda\tau)$-distributed.*

A.3 Stochastic integration

In the following, we provide (roughly) the definition of the stochastic integral on an interval $[0, T]$. Indefinite generalizations are possible, but not presented here. The stochastic integral, also called Itô integral, is an integral where both integrand and integrator may be stochastic processes. The stochastic integral is first defined for *simple predictable processes*. By using the so-called *Itô isometry*, we are able to then conduct an argument that extends the stochastic integration concept to non-simple functions.

Definition A.3.1. *Let $0 = t_0 < t_1 < t_2 < \cdots < t_K = T$ be a partition on $[0, T]$. A simple predictable process f is a stochastic process on the form*

$$f_t = \sum_{k=0}^{K-1} \alpha_k \mathbb{1}_{[t_k, t_{k+1})}(t),$$

where α_k are $\mathcal{F}_{t_k}$-measurable and square integrable $\mathbb{R}^d$-valued random variables.

For simple processes f we define the stochastic integral from 0 to T, $I_T(f)$, with respect to a Brownian motion W on a probability space $(\Omega, \mathcal{F}, \mathbf{P})$ as

$$I_T(f) = \int_0^T f_t \, \mathrm{d}W_t = \sum_{k=0}^{K-1} \alpha_k (W_{t_{k+1}} - W_{t_k})$$

For simple processes, the Itô isometry

$$\mathbf{E}[\|I_T(f)\|^2] = \mathbf{E}\left[\int_0^T \|f_t\|^2 \mathrm{d}t \right],$$

can be shown. We now introduce *Lebesgue spaces*, L^p-spaces, for the special case $p = 2$. We define a space of L^2-integrable functions equipped with the corresponding L^2-norm. The definition is stated for a general measure space.

Definition A.3.2. *Let $(X, \mathcal{A}, \mu)$ be a measure space. We define the space of square integrable functions $L^2(X; \mathbb{R}^d)$ as*

$$L^2_\mu(X; \mathbb{R}^d) = \left\{ f : X \to \mathbb{R}^d : f \text{ is measurable and } \left(\int_X \|f\|^2 \mathrm{d}\mu \right)^{\frac{1}{2}} < \infty \right\}$$

with norm

$$\|f\|_{L^2_\mu(X;\mathbb{R}^d)} = \left(\int_X \|f\|^2 \mathrm{d}\mu \right)^{\frac{1}{2}}.$$

It can be shown that $L^2_\mu(X; \mathbb{R}^d)$ is *complete*, meaning that every Cauchy sequence in $L^2_\mu(X; \mathbb{R}^d)$ converges to a limit in $L^2_\mu(X; \mathbb{R}^d)$.

In terms of L^2-norms, the Itô isometry reads

$$\|I_T(f)\|^2_{L^2(\Omega;\mathbb{R}^d)} = \|f\|^2_{L^2(\Omega \times [0,T];\mathbb{R}^d)}$$

where we have dropped the dependence on the measure in the notation. It understood that Ω has measure $\mathbf{P}$ and $\Omega \times [0,T]$ has measure $\mathbf{P} \times \lambda$, where λ is Lebesgue measure. We now wish to extend the stochastic integration concept to a larger class of so-called predictable processes. The Itô isometry is central to this extension.

Let S denote the class of all simple predictable stochastic processes. Let L^2_{pred} denote the *closure* of S in the $L^2(\Omega \times [0,T]; \mathbb{R}^d)$-norm. Since L^2_{pred} is the closure of S, it holds that for every $g \in L^2_{\text{pred}}$, there exists a Cauchy sequence $(g^n)_{n\in\mathbb{N}} \subset S$ such that

$$\lim_{n\to\infty} \|g - g^n\|_{L^2(\Omega,[0,T];\mathbb{R}^d)} = 0.$$

The difference of any two simple processes is also a simple process. The Itô isometry then yields

$$\lim_{m,n\to\infty} \|g^m - g^n\|^2_{L^2(\Omega\times[0,T];\mathbb{R}^d)} = \lim_{m,n\to\infty} \|I_T(g^m) - I_T(g^n)\|^2_{L^2(\Omega;\mathbb{R}^d)} = 0.$$

This means that $(I_T(g^n))_{n\in\mathbb{N}}$ is also a Cauchy sequence. Since $L^2(\Omega;\mathbb{R}^d)$ is a complete space there exists a limit $I_T(g) := \lim_{n\to\infty} I_T(g^n) \in L^2(\Omega;\mathbb{R}^d)$. This limit defines the stochastic integral for functions $g \in L^2_{\text{pred}}$ and we can for simplicity write it as

$$\int_0^T g_t \mathrm{d}W_t := I_T^{(g)}.$$

Since L^2_{pred} really is an extension of S, we have now successfully defined the stochastic integral for non-simple functions $f \in L^2_{\text{pred}}$. The space of functions L^2_{pred} is rich, for instance, it can be shown that solutions to SDEs whose coefficients are Lipschitz continuous are in L^2_{pred}.

B

Algorithms

In this chapter we present algorithms for the Zig-Zag sampler, the sticky Zig-Zag sampler, and the bouncy particle sampler in Sections B.1–B.3 respectively. The algorithms are schematic representations and not efficient implementations.

B.1 The Zig-Zag algorithm

Algorithm 1 Zig-Zag sampler

function ZIG-ZAG(ξ_0, θ_0)
  $(T_0, \Xi_0, \Theta_0) \leftarrow (0, \xi_0, \theta_0)$
  for $k \leftarrow 1$ to N_{iter} **do**
    for $i \leftarrow 1$ to d **do**
      $\lambda_i(t) \leftarrow m_i(\Xi_{k-1} + \Theta_{k-1}t, \Theta_{k-1})$
      Let $\tilde{\lambda}_i$ denote a computational upper bound for λ_i
      Simulate τ_i such that $\mathbf{P}(\tau_i > t) = \exp(-\int_0^t \tilde{\lambda}_i(s)\mathrm{d}s)$
    end for
    $\tau \leftarrow \min((\tau_i)_{i\in[1..d]})$
    $j \leftarrow \operatorname{argmin}_{i\in[1..d]}((\tau_i)_{i\in[1..d]})$
    $T_k \leftarrow T_{k-1} + \tau$
    $\Xi_k \leftarrow \Xi_{k-1} + \Theta_{k-1}\tau$
    if $u \sim \mathcal{U}(0,1) < \lambda_j(\tau)/\tilde{\lambda}_j(\tau)$ **then**
      $\Theta_k \leftarrow F_j[\Theta_{k-1}]$
    else
      $\Theta_k \leftarrow \Theta_{k-1}$
    end if
  end for
  return $(T_i, \Xi_i, \Theta_i)_{i\in[0..N_{\text{iter}}]}$
end function

B.2 The sticky Zig-Zag algorithm

Algorithm 2 Sticky Zig-Zag sampler

function STICKY-ZIG-ZAG(ξ_0, θ_0)
  $(T_0, \Xi_0, \Theta_0) \leftarrow (0, \xi_0, \theta_0)$
  $\alpha \leftarrow$ The set of active variables
  $\alpha^{\mathsf{c}} \leftarrow [1..d] \setminus \alpha$
  for $k \leftarrow 1$ to N_{iter} **do**
    for $i \leftarrow 1$ to d **do**
      $\lambda_i(t) \leftarrow m_i(\Xi_{k-1} + \Theta_{k-1}t, \Theta_{k-1})$
      Let $\tilde{\lambda}_i$ denote a computational upper bound for λ_i
      Simulate τ_i^{refl} such that $\mathbf{P}(\tau_i > t) = \exp(-\int_0^t \tilde{\lambda}_i(s)\mathrm{d}s)$
      Simulate $\tau_i^{\text{unfreeze}} \sim \text{Exp}(1/\kappa_i)$
    end for
    $\tau^{\text{refl}} \leftarrow \min((\tau_i^{\text{refl}})_{i\in\alpha})$
    $\tau^{\text{freeze}} \leftarrow$ Time until next active variable freezes
    $\tau^{\text{unfreeze}} \leftarrow \min((\tau_i^{\text{unfreeze}})_{i\in\alpha^{\mathsf{c}}})$
    if $\tau^{\text{refl}} < \min(\tau^{\text{freeze}}, \tau^{\text{unfreeze}})$ **then** ▷ Reflection event
      $T_k \leftarrow T_{k-1} + \tau^{\text{refl}}$
      $\Xi_k \leftarrow \Xi_{k-1} + \Theta_{k-1}^{\text{active}}\tau^{\text{refl}}$
      $j \leftarrow \text{argmin}_{i\in\alpha}((\tau_i^{\text{refl}})_{i\in\alpha})$
      if $u \sim \mathcal{U}(0,1) < \lambda_j(\tau^{\text{refl}})/\tilde{\lambda}_j(\tau^{\text{refl}})$ **then**
        $\Theta_k \leftarrow F_j[\Theta_{k-1}]$
      else
        $\Theta_k \leftarrow \Theta_{k-1}$
      end if
    else if $\tau^{\text{freeze}} < \tau^{\text{unfreeze}}$ **then** ▷ Freezing event
      $T_k \leftarrow T_{k-1} + \tau^{\text{freeze}}$
      $\Xi_k \leftarrow \Xi_{k-1} + \Theta_{k-1}^{\text{active}}\tau^{\text{freeze}}$
      $\Theta_k \leftarrow \Theta_{k-1}$
      $j \leftarrow$ Index of the variable that freezes
      $\alpha \leftarrow \alpha \setminus \{j\}$
      $\alpha^{\mathsf{c}} \leftarrow \alpha^{\mathsf{c}} \cup \{j\}$
    else ▷ Unfreezing event
      $T_k \leftarrow T_{k-1} + \tau^{\text{unfreeze}}$
      $\Xi_k \leftarrow \Xi_{k-1} + \Theta_{k-1}^{\text{active}}\tau^{\text{unfreeze}}$
      $\Theta_k \leftarrow \Theta_{k-1}$
      $j \leftarrow \text{argmin}_{i\in\alpha^{\mathsf{c}}}((\tau_i^{\text{unfreeze}})_{i\in\alpha^{\mathsf{c}}})$
      $\alpha \leftarrow \alpha \cup \{j\}$
      $\alpha^{\mathsf{c}} \leftarrow \alpha^{\mathsf{c}} \setminus \{j\}$
    end if
  end for
  return $(T_i, \Xi_i, \Theta_i)_{i\in[0..N_{\text{iter}}]}$
end function

B.3 The bouncy particle algorithm

Algorithm 3 Bouncy particle sampler

function BOUNCY-PARTICLE(ξ_0, θ_0)
  $(T_0, \Xi_0, \Theta_0) \leftarrow (0, \xi_0, \theta_0)$
  for $k \leftarrow 1$ to N_{iter} **do**
    $\lambda(t) \leftarrow m(\Xi_{k-1} + \Theta_{k-1}t, \Theta_{k-1})$
    Let $\tilde{\lambda}$ denote a computational upper bound for λ
    Simulate τ^{refl} such that $\mathbf{P}(\tau > t) = \exp(-\int_0^t \tilde{\lambda}(s)\mathrm{d}s)$
    Simulate $\tau^{\text{refresh}} \sim \text{Exp}(\lambda_{\text{ref}})$
    if $\tau^{\text{refl}} < \tau^{\text{refresh}}$ **then** ▷ Reflection event
      $T_k \leftarrow T_{k-1} + \tau^{\text{refl}}$
      $\Xi_k \leftarrow \Xi_{k-1} + \Theta_{k-1}\tau^{\text{refl}}$
      if $u \sim \mathcal{U}(0,1) < \lambda(\tau^{\text{refl}})/\tilde{\lambda}(\tau^{\text{refl}})$ **then**
        $\Theta_k \leftarrow \Theta_{k-1} - 2\frac{\langle \Theta_{k-1}, \nabla\Psi(\Xi_k, \Theta_{k-1})\rangle}{\|\nabla\Psi(\Xi_k, \Theta_{k-1})\|^2}\nabla\Psi(\Xi_k, \Theta_{k-1})$
      end if
    else ▷ Refresh event
      $T_k \leftarrow T_{k-1} + \tau^{\text{refresh}}$
      $\Xi_k \leftarrow \Xi_{k-1} + \Theta_{k-1}\tau^{\text{refresh}}$
      $\Theta_k \leftarrow$ Sample from $\mathcal{N}(0_d, I_d)$
    end if
  end for
  return $(T_i, \Xi_i, \Theta_i)_{i\in[0..N_{\text{iter}}]}$
end function

C

Calculations

C.1 Rotation matrices in the ballistic model

In this section, we derive the rotation matrices between the internal and absolute reference used for the ballistic model. Define the pitch angle θ_p as the angle between the nose of the target and the ground. Define the yaw angle θ_y as the angle between the heading of the target and the x-axis of the absolute coordinate system. Roll is not relevant since the target is modeled as a point. A rotation matrix between the absolute and internal coordinate systems can then be defined as

$$R(\theta_p, \theta_y) = R_{\text{pitch}}(\theta_p) R_{\text{yaw}}(\theta_y),$$

where the R_{pitch} and R_{yaw} are defined as

$$R_{\text{pitch}}(\theta_p) = \begin{pmatrix} \cos(\theta_p) & 0 & \sin(\theta_p) \\ 0 & 1 & 0 \\ -\sin(\theta_p) & 0 & \cos(\theta_p) \end{pmatrix}, \quad R_{\text{yaw}}(\theta_y) = \begin{pmatrix} \cos(\theta_y) & -\sin(\theta_y) & 0 \\ \sin(\theta_y) & \cos(\theta_y) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Trigonometric functions of θ_p and θ_y can be calculated from the velocity vector $\vec{v} = [v_x, v_y, v_z]$ as

$$\begin{aligned} \cos(\theta_p) &= \sqrt{\frac{v_x^2 + v_y^2}{v_x^2 + v_y^2 + v_z^2}}, & \sin(\theta_p) &= \frac{-v_z}{\sqrt{v_x^2 + v_y^2 + v_z^2}}, \\ \cos(\theta_y) &= \frac{v_x}{\sqrt{v_x^2 + v_y^2}}, & \sin(\theta_y) &= \frac{v_y}{\sqrt{v_x^2 + v_y^2}}, \end{aligned}$$

with the special case $\cos(\theta_y) = 1, \sin(\theta_y) = 0$ if $\sqrt{v_x^2 + v_y^2} = 0$.

C.2 Process covariance in the ballistic model

In this section, the process noise covariance of using the noise linearization approximation on the ballistic SDE is derived. The SDE for the state $X = [x, \dot{x}]$ reads

$$\mathrm{d}\begin{pmatrix} x \\ \dot{x} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ -f_d(\dot{x})|\dot{x}|\dot{x} \end{pmatrix} \mathrm{d}t + \begin{pmatrix} 0 \\ \sigma_{\text{process}} \end{pmatrix} \mathrm{d}W_t$$

where $[0, \sigma_{\text{process}}] = B$ is the noise coefficient. The Jacobian of the drift coefficient is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 0 & a \end{pmatrix},$$

where $a = \partial_{\dot{x}}(-f_d(\dot{x})|\dot{x}|\dot{x})$, which yields a linear SDE on the form of (2.14). Let $\mathbf{A} = \mathbf{C}\mathbf{D}\mathbf{C}^{-1}$ be the diagonalization of $\mathbf{A}$, with

$$\mathbf{C} = \begin{pmatrix} 1 & \frac{1}{a} \\ 0 & 1 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 0 & 0 \\ 0 & a \end{pmatrix}, \quad \mathbf{C}^{-1} = \begin{pmatrix} 1 & -\frac{1}{a} \\ 0 & 1 \end{pmatrix}.$$

We then have

$$\exp(\pm\mathbf{A}t) = \mathbf{C}\exp(\pm\mathbf{D}t)\mathbf{C}^{-1} = \mathbf{C}\begin{pmatrix} 1 & \frac{1}{a}(\exp(\pm at) - 1) \\ 0 & \exp(\pm at) \end{pmatrix}\mathbf{C}^{-1}.$$

The solution is now slightly different from the presentation in 2.5.1, as we will wait with transforming back until the end. The integrand in the Itô isometry, $(\exp(-\mathbf{A}t)B)(\exp(-\mathbf{A}t)B)^{\mathsf{T}}$, reads

$$\sigma^2_{\text{process}}\begin{pmatrix} \frac{1}{a^2}(\exp(-ta) - 1)^2 & \frac{1}{a}(\exp(-ta) - 1)\exp(-ta) \\ \text{sym.} & (\exp(-ta))^2 \end{pmatrix},$$

which is then integrated from 0 to T, yielding

$$\mathbf{\Sigma} = \begin{pmatrix} I_{1,1} & I_{1,2} \\ \text{sym.} & I_{2,2} \end{pmatrix}$$

with entries

$$\begin{aligned} I_{1,1} &= -\sigma^2_{\text{process}}(-2aT + \exp(-2aT) - 4\exp(-aT) + 3)/(2a^3), \\ I_{1,2} &= -\sigma^2_{\text{process}}(\exp(-2aT)(\exp(aT) - 1)^2)/(2a^2), \\ I_{2,2} &= -\sigma^2_{\text{process}}(\exp(-2aT) - 1)/(2a). \end{aligned}$$

Finally transforming back to X, we obtain

$$\text{Cov}[X] = \exp(\mathbf{A}T)\mathbf{\Sigma}\exp(\mathbf{A}T)^{\mathsf{T}}.$$

The result of this product is too long to be stated here.

C.3 Process covariance in the acceleration jump-drag model

In this section the process noise covariance of using the noise linearization approximation on the acceleration jump-drag SDE is derived. The SDE for the state $X = [x, \dot{x}, \ddot{x}]$ reads

$$\mathrm{d}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ -A|\dot{x}|\dot{x} + \ddot{x} \\ 0 \end{pmatrix}\mathrm{d}t + \begin{pmatrix} 0 \\ 0 \\ \sigma_{\text{process}} \end{pmatrix}\mathrm{d}W_t$$

where $[0, 0, \sigma_{\text{process}}] = B$ is the noise coefficient. The Jacobian of the drift coefficient is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & a & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

where $a = -2A|\dot{x}|$, which yields a linear SDE on the form of (2.14). Let $\mathbf{A} = \mathbf{S}\mathbf{J}\mathbf{S}^{-1}$ be the Jordan matrix decomposition of $\mathbf{A}$, with

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & \frac{1}{a} \\ 0 & 1 & 1 \\ 0 & -a & 0 \end{pmatrix}, \quad \mathbf{J} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & a \end{pmatrix}, \quad \mathbf{S}^{-1} = \begin{pmatrix} 1 & -\frac{1}{a} & -\frac{1}{a^2} \\ 0 & 0 & -\frac{1}{a} \\ 0 & 1 & \frac{1}{a} \end{pmatrix}.$$

We then have

$$\exp(\pm\mathbf{A}t) = \mathbf{S}\exp(\pm\mathbf{J}t)\mathbf{S}^{-1} = \mathbf{S}\begin{pmatrix} 1 & \pm t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \exp(\pm at) \end{pmatrix}\mathbf{S}^{-1}.$$

The solution is now slightly different from the presentation in 2.5.1, as we will wait with transforming back until the end. The integrand in the Itô isometry, $(\exp(-\mathbf{A}t)B)(\exp(-\mathbf{A}t)B)^{\mathsf{T}}$, reads

$$\sigma^2_{\text{process}} \begin{pmatrix} \left(\frac{\exp(-at)+at-1}{a^2}\right)^2 & \left(\frac{\exp(-at)+at-1}{a^2}\right)\left(\frac{\exp(-at)-1}{a}\right) & \left(\frac{\exp(-at)+at-1}{a^2}\right) \\ \text{sym.} & \left(\frac{\exp(-at)-1}{a}\right)^2 & \left(\frac{\exp(-at)-1}{a}\right) \\ \text{sym.} & \text{sym.} & 1 \end{pmatrix},$$

which is then integrated from 0 to T, yielding

$$\mathbf{\Sigma} = \begin{pmatrix} I_{1,1} & I_{1,2} & I_{1,3} \\ \text{sym.} & I_{2,2} & I_{2,3} \\ \text{sym.} & \text{sym.} & I_{3,3} \end{pmatrix}$$

with entries

$$\begin{aligned} I_{1,1} &= \sigma^2_{\text{process}}(2(aT-1)^3 - 3\exp(-2aT) - 12aT\exp(-aT) + 5)/(6a^5), \\ I_{1,2} &= -\sigma^2_{\text{process}}(\exp(-2aT)(\exp(aT)(aT-1)+1)^2)/(2a^4), \\ I_{1,3} &= \sigma^2_{\text{process}}(aT(aT-2) - 2\exp(-aT) + 2)/(2a^3), \\ I_{2,2} &= -\sigma^2_{\text{process}}(-2aT + \exp(-2aT) - 4\exp(-aT) + 3)/(2a^3), \\ I_{2,3} &= -\sigma^2_{\text{process}}(aT + \exp(-aT) - 1)/a^2, \\ I_{3,3} &= \sigma^2_{\text{process}}T. \end{aligned}$$

Finally transforming back to X, we obtain

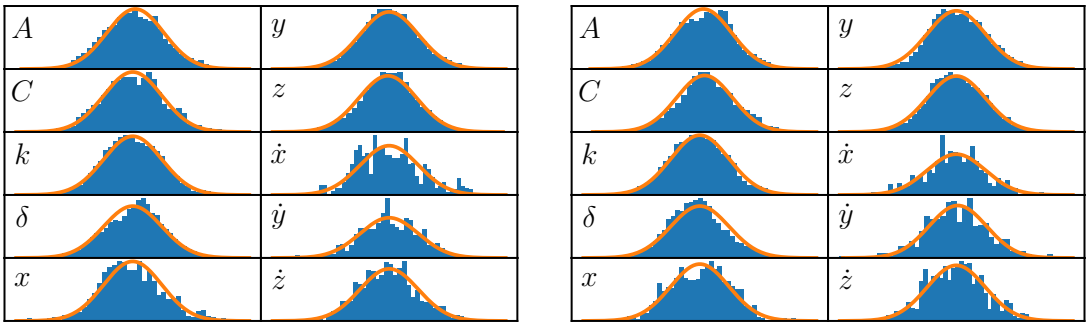
$$\text{Cov}[X] = \exp(\mathbf{A}T)\mathbf{\Sigma}\exp(\mathbf{A}T)^{\mathsf{T}}.$$

The result of this product is too long to be stated here.

D

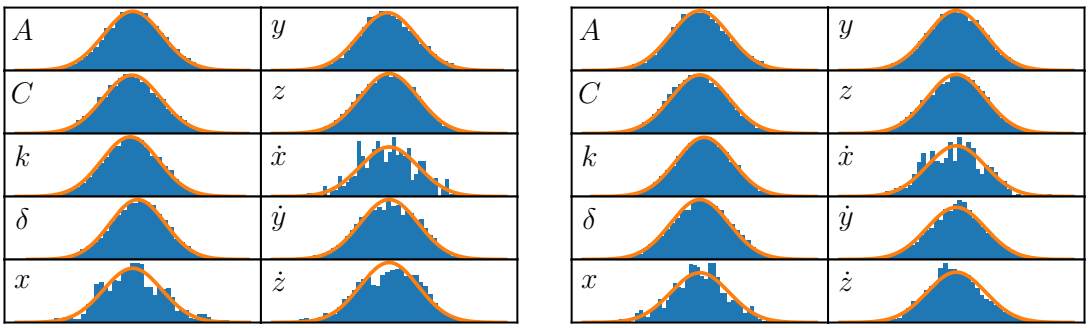
Figures

D.1 Prior reconstruction accuracy



(a) Test case 1, BPS (const.). **(b)** Test case 2, BPS (const.).

Figure D.1: Comparison between the prior and the sampled approximation thereof, see Section 6.1.1. A total of 300 trajectory realizations were used and 30 independent samples were selected from each.



(a) Test case 1, BPS (local). **(b)** Test case 2, BPS (local).

Figure D.2: Comparison between the prior and the sampled approximation thereof, see Section 6.1.1. A total of 300 trajectory realizations were used and 30 independent samples were selected from each.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY