Approximate Manifold Sampling

Robust Bayesian Inference for Machine Learning

By

MAURO CAMARA ESCUDERO

School of Mathematics
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Science.

JANUARY 2024

Word count: 40598

Abstract

Efficient sampling from probability densities concentrated around a lower-dimensional submanifold is crucial in numerous applications arising in machine learning, statistics, and statistical physics. This task is particularly challenging due to the extreme anisotropy and high-dimensionality of the problem, and the correlation between the variables. We propose a novel family of bespoke MCMC algorithms designed to sample efficiently from these densities and show their computational superiority to general purpose and specialized samplers. Furthermore, we contribute to the development of integrator and Markov snippets, which are a particular class of general-purpose sequential algorithms for Bayesian inference and machine learning that can leverage the geometry of the space with integrators, is highly robust to the choice of the step size and the number of integration steps, and naturally lends itself to parallelisation. Building on these foundations, we present a sequential algorithm that is particularly well-suited to approximate manifold sampling.

Dedication and acknowledgements

First of all, I would like to express immense gratitude to my supervisors, Christophe and Mark, for their unwavering mentorship and support throughout these three years and a half, both on a professional and personal level. When I have been guilty of sending emails and Slack messages at ungodly hours of the night, you have made me feel welcome and have gone above and beyond, and for that, I am forever grateful.

To my examiners, Anthony and Matthew thank you for the interesting points you raised during the examination. Your insightful feedback and thoughtful questions have greatly enriched my research. I deeply appreciate your time and effort in reviewing my work and for contributing to my academic growth.

A mia mamma, a mio papà, e a mio fratello, per il vostro instancabile sostegno, incoraggiamento incessante, ed amore incondizionato. Grazie per aver creduto in me durante questo cammino ed avermi insegnato l'arte della perseveranza. Nonostante la distanza, mi avete sempre fatto sentire a casa, ve ne sono profondamente grato. Voglio anche ringraziare calorosamente la mia (gigantesca) famiglia esterna e Valentina: il vostro supporto ha arricchito ogni passo di quest'avventura.

Jeremy, I feel privileged to have met you, going to dance events together, our distanced walks during the pandemic, and all our fun adventures have kept me going and I want you to know I do not take that for granted.

To all my dancing friends, thank you for the fun classes, weekends, competitions and performances. I never thought dancing would become such a big part of my life, and I owe this, in no small part, to Matt and Sarah for introducing me to the dance in such a fun, lighthearted way: your classes have often been the highlight of my weeks throughout these years. Mellis, from teaching me my first dance choreography, to becoming fellow dancers and friends at weekenders, I am grateful for all your words of encouragement both on and off the dance floor, as well as for being such a caring friend and checking in with me during this journey. Claire, you have been an incredible friend and dance partner, I cannot wait to catch you on the dance floor after submission. You are brilliant and an academic inspiration, wherever your PhD search will lead you, I have no doubt you will be successful and meet people who will be as caring and supportive as you have been to me.

Aoife, I want to thank you whole-heartedly for your friendship, unfaltering helping hand, restoring my love of reading and fostering my addiction to Mexican paletas.

Peter, Emma and Simone, thank you for your friendship and fun times in Senate House towards the end of this journey. You have made the writing much more enjoyable and I am beyond grateful for your friendship.

Alex, I was fortunate enough to meet you and I want to thank you warmly for introducing me to the Bristol music scene, our impromptu micro-adventures, and just for your general silliness, all of which, have been a blessing.

Andrea, you have lured me into the Sci-Fi world, brought out my inner child, and together we have had more adventures than I can count, which have made my time in Bristol that much more enjoyable. I am forever grateful to you for teaching me so much about how to be a good friend.

CoSInES and Bayes4Health friends and colleagues, I don't think my PhD experience would have been the same without so many friends at conferences, workshops and masterclasses. Particularly, I want to thank everyone who has put the outstanding effort in organising the masterclasses, which have been not only a great source for learning from the best in the field, but also invaluable opportunities to make friends. I will not forget the fun board game nights and delicious dinners organised by Andi and Sam. Lorenzo and Estevao, thank you for humbling me at the gym, you both put an unsparing amount of hard work in your research and exercising, which is incredibly inspiring. Filippo, I hope you have not been preparing for the Brasilian Jiu-Jitsu match at the next masterclass, I sure know I haven't.

Beth, thank you for initiating me to the secret art of chicken Adobo, and for teaching me how to (fail to) make sushi in a fast-paced environment: after that, the PhD was a walk in the park.

To the gang in Verona: Ale, Alice, Lopre, Matteo, Mela, Napo, Ponzi, Savi, and Sofia, I am grateful for the encouragement you gave me before deciding to come to the UK several years ago and, most importantly, I am thankful that, to this day, we make an effort to meet frequently and keep that bond alive. We are now all spread out, going our own ways, but I feel very fortunate that when we meet, it feels like no time has passed.

Sofiia, I am forever grateful for your love, support and for the way you make me feel at home. Your unwavering belief in me, your kindness, and your strength have been a constant source of inspiration. Thank you for making every moment together so special.

I want to thank the admin staff, for their exceptional care, support, and hard work in organising fun away days, hackathons, and much more, as well as always offering a helping hand in times of need.

Finally, I wanna thank me [44]

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

TABLE OF CONTENTS

List of Tables

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

In this chapter, we review some of the standard algorithms to sample from probability distributions, focusing on Metropolis-Hastings, Importance Sampling, and Sequential Monte Carlo. These will constitute the building blocks for the contributions of this thesis.

1.1 Contributions

The first contribution of this thesis is the development of a parametrized family of sampling algorithms that are efficient at sampling from filamentary distributions, i.e. probability measures that are highly concentrated around a lower-dimensional manifold. We show on a range of problems that they are greatly superior to general-purpose algorithms such as Random Walk Metropolis and Hamiltonian Monte Carlo [16, 47], which become extremely inefficient when we want to reduce bias. We demonstrate that when taking the computational cost and the bias into consideration, our proposed algorithm is often superior to exact manifold sampling algorithms.

The second contribution of this thesis, is contributing to the development of the theory and methodology of Markov and Integrator Snippets [154], specifically in the context of sampling from filamentary distributions. We show that Markov Snippets have several appealing properties and demonstrate empirically their superiority to standard SMC samplers.

1.2 Thesis Organization

This chapter introduces standard sampling algorithms and relevant notation, both of which will form useful references for the rest of the thesis. Chapter 2 introduces the problem of approximate manifold sampling and presents THUG, our proposed algorithm for this task. Chapter 3 reviews

and extends the theory on Markov Snippets, describes several of its appealing properties, and introduces a novel sequential algorithm for approximate manifold sampling.

1.3 Reproducibility and Code

The examples in this thesis are all coded in Python 3.10.4 [142], and the code will be made available here. Three separate code implementations will be available. The first one is written in such a way as to maximise readability on first try, and its aim is to provide fellow researchers with a place to check their understanding of the algorithms presented in this thesis, without having to browse many files and/or classes. This code is purposely not optimised nor follows Don't Repeat Yourself (DRY) principles, in fact, repetition and redundancy are used to increase readability. The second implementation can be used to reproduce the experiments and plots in this thesis. The third implementation's aim is to maximise computational efficiency and usability, and follows (when helpful) standard Object-Oriented Programming guidelines. Finally, here is a statistically-themed playlist I have redacted to help you read through this thesis[1].

1.4 Notation

- **Sets**: Sans-Serif capital letters will denote sets $\mathsf{A}, \mathsf{B}, \mathsf{C}$ except for the standard Euclidean space $\mathbb{R}$ and sigma-algebras, which will be denoted using Calligraphic capital letters such as $\mathscr{F}, \mathscr{X}, \mathscr{Y}$, and we write $\mathscr{B}(\mathbb{R})$ for the Borel sigma algebra on $\mathbb{R}$. We write $\mathsf{A}^n := \mathsf{A} \times \mathsf{A} \times \cdots \times \mathsf{A}$ for the n^{th} Cartesian product of a set, and $\mathscr{A}^{\otimes n}$ for the minimal product sigma-algebra on A^n. For a set A we write $|\mathsf{A}|$ for its cardinality. The set of natural numbers will be denoted by $\mathbb{N} = \{1, 2, \ldots\}$, the set of integers as $\mathbb{Z}$ and the set of non-negative integers as $\mathbb{Z}_+$. Given a set A, its power set will be written as $\mathscr{P}(\mathsf{A})$. Given two real numbers $a, b \in \mathbb{R}$ with $a < b$, we use standard conventions for intervals meaning that square brackets include the endpoints, and parenthesis do not. For any two integers $n, m \in \mathbb{Z}$ with $n > m$, we write $[\![m, n]\!] := \{m, m+1, \ldots, n\}$ for the set of integers from m to n, including the endpoints. For brevity, when $m = 1$ and $n > 0$, we just write $[\![n]\!] = \{1, \ldots, n\}$.

- **Matrices and Vectors**: Roman capital letters will denote matrices $\mathsf{N}, \mathsf{T}, \mathsf{J}$ and we reserve the symbols I_n and $0_{n \times m}$ for the n-dimensional identity matrix and the $n \times m$ rectangular zero matrix respectively. We denote the transpose of a matrix A as $\mathsf{A}^\top$ and the absolute value of its determinant as $|\det \mathsf{A}|$. For any vector $x \in \mathbb{R}^n$ we write $\|x\|$ for its Euclidean norm.

- **Calculus**: Given a function $f : \mathbb{R}^n \to \mathbb{R}^m$ we denote its Jacobian matrix as either $\mathsf{J}_f(x)$ or J_x, when f is clear from context. When $m > 1$, the second derivative of f can represented by a

[1] If you are wary, as you should be, about clicking shortened links, you should navigate to the following address https://maurocamaraescudero.netlify.app/code

third-order tensor, which can be thought of as a linear map that takes in a vector in $\mathbb{R}^n$ and outputs an $m \times n$ matrix. We call this linear map the Hessian tensor of f and we denote it as $\mathrm{H}[x](v,w)$ for $x,v,w \in \mathbb{R}^n$. Notice that it reduces to the Hessian matrix H_x or $\mathrm{H}(x)$ when $m = 1$. For $f : \mathbb{R}^n \to \mathbb{R}$ we write ∇f for its gradient which we consider as a column vector $\nabla f(x) = \mathrm{J}_f(x)^\top$. To specify the variable of differentiation, we sometimes write $\nabla_x f(x)$. In the context of ODEs, if $x : [0, +\infty) \to \mathbb{R}^n$ is a function of time, then we write $\dot{x}$ and $\ddot{x}$ for its first and second derivative with respect to time. We often indicate composition of a function with itself using exponents, meaning that $f^2 = f \circ f$, $f^3 = f \circ f \circ f$ and so on.

- **Measure Theory and Probability**: For a measure π and a measurable set A we write $\pi(\mathsf{A}) = \int_{\mathsf{A}} d\pi$ for the size of A under π. We write $\pi \ll \mu$ for two measures on the same measurable space such that π is absolutely continuous with respect to μ, and denote their Radon-Nikodym derivative by $\frac{d\pi}{d\mu}$ or $\frac{\pi(dx)}{\mu(dx)}$. We do not settle for one or the other because we use them to emphasize different things. The first one is used when the Radon-Nikodym theorem allows for simplifications in downstream calculations, the second is used to emphasize that it is a function, especially when it will be composed with a different one, e.g. via a change of variables. If a measure π admits a density with respect to the Lebesgue measure, we shall denote its density with the same symbol, this is a standard abuse of notation that simplifies exposition. Similarly, if K is a Markov kernel and $K(x, dy)$ admits a density on the second argument with respect to the Lebesgue measure, we denote it with the same symbol but lower case $k(y \mid x)$. Given a measure π and a function ψ, we write π^ψ for the pushforward of π by ψ. The Dirac-delta measure at $X \in \mathsf{X}$ will be denoted as $\delta_X(dx)$. The expectation and variance operators are $\mathbb{E}[\cdot]$ and $\mathbb{V}[\cdot]$ respectively. When $X : \mathsf{A} \to \mathsf{B}$ is a random variable from a probability space $(\mathsf{A}, \mathscr{A}, P)$ to a measurable space $(\mathsf{B}, \mathscr{B})$, we write P_X for its distribution on the latter space. In this context, we shall often write P as $\mathbb{P}$ to allow for the standard notation $P_X(\cdot) = \mathbb{P}(X \in \cdot)$ that interprets $\mathbb{P}$ as the probability operator. We write $\sigma(X)$ and $\sigma(\mathscr{C})$ for the sigma-algebra generated by a random variable X and a collection of sets $\mathscr{C}$ respectively. If $(\mathsf{A}, \mathscr{A})$ and $(\mathsf{B}, \mathscr{B})$ are two measurable spaces, we write $(\mathsf{A} \times \mathsf{B}, \mathscr{A} \otimes \mathscr{B})$ for their product, where $\mathsf{A} \times \mathsf{B}$ denotes the Euclidean product and $\mathscr{A} \otimes \mathscr{B}$ is the minimal sigma algebra that it generates. When $\mathscr{A} = \mathscr{B}$, we simply write $\mathscr{A}^{\otimes 2}$. For a probability measure μ on $(\mathsf{Z}, \mathscr{Z})$ and a Markov kernel $M : \mathsf{Z} \times \mathscr{Z} \to [0,1]$ we write $\mu \otimes M$ for the joint probability measure $\mu(dz)M(z, dz')$, and we generally write $\mu \otimes M(dz, dz')$. When M is just a probability measure ν, then $\mu \otimes \nu$ denotes the product measure. We also define $\mu \oplus M(dz, dz')$ as $\mu(dz')M(z', dz')$, meaning that the $\oplus$ operator denotes the joint measure but with the order of the variables of integration reversed. This is helpful when writing Radon-Nikodym derivatives such as $\frac{\mu(dz')L(z', dz)}{\mu(dz)M(z, dz')}$ succinctly as $\frac{d\mu \oplus L}{d\mu \otimes M}$. The marginal of the joint $\mu \otimes M(dz, dz')$ in z' is denoted μM as usual.

- **Miscellaneous**: $\mathcal{N}(\mu, \Sigma)$ indicates a normal distribution with mean μ and covariance Σ, and when we wish to specify the variable of interest, we sometimes write $\mathcal{N}(dx \mid \mu, \Sigma)$ for

the distribution. Consequently, $\mathcal{N}(x \mid \mu, \Sigma)$ denotes the density with respect to a suitable Lebesgue measure. For the Kronecker delta we use the same symbol as the Dirac delta i.e. δ_{ij}.

1.5 Ordinary Monte Carlo

A common task in Bayesian statistics is that of estimating expectations of a *test function* $h : \mathsf{X} \to \mathbb{R}$ with respect to a probability measure π on a measurable space $(\mathsf{X}, \mathscr{X})$

$$\mathbb{E}[h] = \int h \, d\pi. \tag{1.1}$$

It is common to refer to π as the *target* distribution and in most cases of interest $(\mathsf{X}, \mathscr{X}) = (\mathbb{R}^d, \mathscr{B}(\mathbb{R}^d))$ for $d \in \mathbb{N}$. If sampling from π directly and independently is possible, then the problem is trivial and upon obtaining N independent and identically-distributed (IID) samples $X^{(1)}, \ldots, X^{(N)}$, $\mathbb{E}[h]$ can be approximated using ordinary Monte Carlo

$$\mathbb{E}[h] \approx \hat{h}_{\mathrm{MC}} := \frac{1}{N} \sum_{i=1}^{N} h(X^{(i)}) \qquad \text{where } X^{(i)} \sim \pi.$$

This corresponds to constructing an *empirical distribution* $\widehat{\pi}^N$ on $(\mathsf{X}, \mathscr{X})$

$$\widehat{\pi}^N(dx) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X^{(i)}}(dx),$$

substituting it for π in (1.1), exchanging the integral sign and the finite summation, and using the standard property of a Dirac-delta measure

$$\int h(x) \delta_X(dx) = h(X).$$

1.6 Markov Chain Monte Carlo

1.6.1 General Markov Chain Theory

In practice it is often not possible to obtain exact and independent samples from π. A popular alternative is to use Markov Chain Monte Carlo (MCMC) which provides dependent samples that are approximately distributed according to π. A full description of the theory and methodology behind MCMC is beyond the scope of this thesis, and we refer the reader to [24] and references therein, and instead focus on few important concepts that will aid understanding of the rest of the thesis.

MCMC algorithms construct a time-homogeneous Markov Chain with transition kernel P that has π as its equilibrium distribution, meaning that the distribution of the states of the chain approaches π with arbitrary accuracy as the Markov Chain progresses, independently of the distribution of the initial state

$$\lim_{n \to \infty} P^n(x, \mathsf{A}) = \pi(\mathsf{A}) \qquad \text{for } \pi\text{-almost all } x.$$

Standard Markov Chain theory suggests a strategy for checking that π is an equilibrium distribution. Informally, if P can visit all states (irreducible), is aperiodic, and leaves π invariant,

$$\pi P = \pi, \tag{1.2}$$

then P is positive recurrent π is both the unique invariant distribution and the equilibrium distribution of P [139]. Furthermore, the convergence happens for all points when P is positive Harris recurrent, meaning that we have probability one of visiting every region with positive mass, even if we start from a region of measure-zero. In this case, the two scenarios are equivalent [139].

1.6.2 Metropolis-Hastings

Perhaps the most common strategy for constructing a MCMC with π as its equilibrium distribution is using the Metropolis-Hastings (MH) framework [68, 100]. Equation (1.2) is known as *global balance* and can be very difficult to establish in practice. A stronger condition known as *detailed balance* is often easier to verify [4] and requires P to be π-reversible. Let $(\mathsf{X}, \mathscr{X})$ be a measurable space, $P : \mathsf{X} \times \mathscr{X} \to [0,1]$ be a Markov kernel and π the target distribution on $(\mathsf{X}, \mathscr{X})$, then P is π-reversible if

$$\int_{\mathsf{A}} \pi(dx) P(x, \mathsf{B}) = \int_{\mathsf{B}} \pi(dx) P(x, \mathsf{A}) \qquad \mathsf{A}, \mathsf{B} \in \mathscr{X}. \tag{1.3}$$

Intuitively, this states that the probability of starting from a state in A and moving to a state in B, is the same as starting from B and ending in A, on average.

MH kernels are designed to satisfy detailed balance, and hence have π as their equilibrium distribution. At an intuitive level, the algorithm starts at a state $x \in \mathsf{X}$ and proposes a new state y, which is always accepted if this has higher density than x under π and is accepted with some probability, that enforces detailed-balance, when this has lower density instead. For the more formal construction, we follow [139, 140]. The MH algorithm requires an additional Markov kernel $Q : \mathsf{X} \times \mathscr{X} \to [0,1]$ known as transition or proposal kernel and a measurable acceptance probability function $a : \mathsf{X} \times \mathsf{X} \to [0,1]$. Then, we define the MH kernel P as

$$P(x, dy) = a(x,y) Q(x, dy) + \left[\int (1 - a(x,y)) Q(x, dy) \right] \delta_x(dy). \tag{1.4}$$

[140] gives sufficient and necessary conditions on Q and a so that P leaves π invariant. By far the most popular choice for $a(x,y)$, which gives the name to this class of methods, is the Hastings acceptance probability function

$$a_{\mathrm{MH}}(x,y) = \min \left\{ 1, \frac{\pi(dy) Q(y, dx)}{\pi(dx) Q(x, dy)} \right\}. \tag{1.5}$$

The proof that this choice implies $\pi P = \pi$ is a standard result, so we refer the reader to [4, 139]. This is not the only possible choice of acceptance function and in fact detailed balance is preserved

as long as $a(x,y)$ can be written as a function of the ratio

$$r(x,y) := \frac{\pi(dy)Q(y,dx)}{\pi(dx)Q(x,dy)} \tag{1.6}$$

that satisfies $a(r) = ra(1/r)$. The Hastings acceptance function corresponds to the choice $a(r) = \min(1,r)$ but another alternative is the Barker acceptance function [9, 69] which corresponds to $a(r) = r/(1+r)$, in which case we get

$$a_{\mathrm{B}}(x,y) = \frac{\pi(dy)Q(y,dx)}{\pi(dx)Q(x,dy) + \pi(dy)Q(y,dx)}. \tag{1.7}$$

Estimates obtained using a_{MH} have smaller asymptotic variance, for the same proposal kernel [110, 140], which helps explaining why the acceptance function (1.5) has been much more common in the literature. Recently, however, the Barker acceptance function has been successfully used in the context of Markov Jump processes [69, 93, 146]. For the rest of this thesis, whenever we refer to a Metropolis-Hastings algorithm, we shall implicitly assume that it uses the Hastings acceptance function, unless explicitly specified otherwise.

1.6.2.1 Expected Squared Jump Distance

The Expected Squared Jump Distance (ESJD) is a common metric used to assess the performance of a MH algorithm. It is defined as the expected value of the squared distance between jumps of the Markov Chain, that is

$$\mathbb{E}_{\pi\otimes P}[\|Y - X\|^2],$$

where P is the MH kernel defined in Equation (1.4), X is the random variable for the current state of the chain, and Y is the next state of the chain. Therefore, the definition of ESJD assumes that the chain is at stationarity

$$X_0, X_1, \ldots, X_N \sim \pi(dx_0) \bigotimes_{n=1}^{N} P(x_{n-1}, dx_n)$$

although this metric is very often used to assess the performance of a Markov Chain that has not necessarily reached that stage but rather $X_0 \sim \nu$, for some initial distribution ν. Given a set of approximate samples from π, denoted $\{X_0, X_1, \ldots, X_N\}$ one can estimate this expectation directly

$$\mathrm{ESJD} \approx \frac{1}{N}\sum_{n=1}^{N} \|X_n - X_{n-1}\|^2, \tag{1.8}$$

but the disadvantage of this approach is that we gain no information about the expectation from rejections and for acceptances, we only use their squared distance but neglect how likely that state was to be proposed in the first place. An alternative, more popular, estimator is found in Appendix A.3

$$\mathrm{ESJD} \approx \frac{1}{N}\sum_{n=1}^{N} \|X^*_{n-1} - X_{n-1}\|^2 a_{\mathrm{MH}}(X_{n-1}, X^*_{n-1}), \tag{1.9}$$

where now $X^*_{n-1} \sim Q(X_{n-1}, \cdot)$ is the proposal at iteration $n = 1, \dots, N$. Notice that the squared distances are weighted by the acceptance probabilities, and states of the Markov Chain inform this estimate, independently of whether they were rejected or not (unless their acceptance probability is zero). Throughout the thesis, we will use this estimator of the ESJD.

1.6.2.2 Random Walk Metropolis

The Random Walk Metropolis (RWM) sampler is a special instance of the Metropolis-Hastings algorithm where the proposal kernel is symmetric, meaning

$$\int_{\mathsf{A}} Q(x, \mathsf{B})\lambda(dx) = \int_{\mathsf{B}} Q(y, \mathsf{A})\lambda(dy) \qquad \forall\, \mathsf{A}, \mathsf{B} \in \mathscr{X},$$

for a given reference measure λ. Consequently, the proposal kernel disappears from the ratio, which simplifies to

$$r(x, y) = \frac{\pi(dy)/\lambda(dy)}{\pi(dx)/\lambda(dx)}.$$

The most common scenario is when $(\mathsf{X}, \mathscr{X}) = (\mathbb{R}^d, \mathscr{B}(\mathbb{R}^d))$, λ is the Lebesgue measure on it, and the proposal kernel is a d-dimensional Gaussian distribution with covariance matrix $\Sigma > 0$, and centered at its first argument

$$Q(x, dy) = \mathcal{N}(dy \mid x, \Sigma).$$

From a computational perspective, a proposal is generated from $x \in \mathbb{R}^d$ by first sampling standard normal perturbations $v \sim \mathcal{N}(0, \mathrm{I}_d)$ and then setting

$$y = x + \mathrm{L}v \qquad \text{where } \mathrm{L}\mathrm{L}^\top = \Sigma. \tag{1.10}$$

A common choice is to set $\Sigma = \sigma^2 \mathrm{I}_d$ for some scaling $\sigma > 0$. For a target consisting of a product of IID components, [56] found that setting $\sigma^2 = s^2/d$ for $s > 0$ leads to the components of the chain to converge to the appropriate Langevin diffusion (which is the corresponding limit of a rescaled continuous-time RWM), and the optimal scaling s can be tuned to reach the optimal acceptance rate 0.234. Further work, has expanded this to much more general target distributions [151], and the proof relies on maximising the Expected Squared Jump Distance (ESJD), which happens when $s > 0$ is such that we achieve the optimal rate 0.234.

1.6.2.3 Metropolis-Hastings with involutions

Equation (1.10) expresses a sample from $Q(x, dx)$ as a deterministic transformation of x and a random perturbation $v \sim \mathcal{N}(0, \mathrm{I}_d)$. As initially noted by [139] and then further developed in [4], it can be very convenient to think of and construct MH algorithms where the proposal is obtained via a deterministic function of the current state of the Markov chain, and some auxiliary variables. More formally, let $(\mathsf{V}, \mathscr{V})$ be another measurable space, $R : \mathsf{X} \times \mathscr{V} \to [0, 1]$ be a regular conditional distribution, and let their product measure on $(\mathsf{X} \times \mathsf{V}, \mathscr{X} \otimes \mathscr{V})$ be

$$\mu(dx, dv) = \pi(dx)R(x, dv).$$

Clearly π is the marginal of μ in the x component, since $R(x, dy)$ integrates to one for any $x \in \mathsf{X}$. Rather than performing MH with π as target, we can instead perform it on the product μ. Suppose that the current state of the chain is $Z = (X, V)$, obtained by first sampling $X \sim \pi$ and then $V \sim R(X, \cdot)$, we can imagine a proposal mechanism where the new state of the chain is obtained by applying an *involution* $\psi : \mathsf{X} \times \mathsf{V} \to \mathsf{X} \times \mathsf{V}$ to Z, which means that the proposal kernel is $\delta_{\psi(z)}(dz')$. The MH acceptance function (this also works for more acceptance functions, such as those satisfying $a(r) = ra(1/r)$ as before) then becomes

$$a_{\mathrm{MH}}(z, z') = \min\left\{1, \frac{\mu^{\psi}(dz)}{\mu(dz)}\right\}.$$

It most cases of interest, $\mathsf{X} = \mathsf{V} = \mathbb{R}^d$, both π and $R(x, \cdot)$ admit a density with respect to the Lebesgue measure and ψ is a diffeomorphism in which case

$$a_{\mathrm{MH}}(z, z') = \min\left\{1, \frac{\mu(\psi^{-1}(z))}{\mu(z)} \left|\det \mathrm{J}_{\psi^{-1}}(z)\right|\right\}.$$

A MH algorithm targeting μ and using the acceptance probability above will leave π invariant [4]. This framework is not only useful for showing correctness of existing algorithms (by manually constructing μ and ψ, the acceptance ratio exists under mild conditions) but it is also helpful in designing new algorithms since all that is required is identifying a product distribution that has marginal π and an involution that maps (x, v) to (x', v'), where x' is the proposed state. For instance, RWM is very easy to fit into this framework. The product distribution is simply

$$\mu(dx, dv) = \pi(dx)\mathcal{N}(dv \mid 0, \mathrm{I}_d),$$

the involution is $\psi_{\mathrm{RWM}}(x, v) = (x + \mathrm{L}v, -v)$ and we notice that the determinant of the Jacobian of ψ_{RWM} is one, and hence the acceptance function becomes

$$a_{\mathrm{MH,\ RWM}}(x, v) = \min\left\{1, \frac{\pi(x + \mathrm{L}v)\mathcal{N}(-v \mid 0, \mathrm{I}_d)}{\pi(x)\mathcal{N}(v \mid 0, \mathrm{I}_d)}\right\} = \min\left\{1, \frac{\pi(x + \mathrm{L}v)}{\pi(x)}\right\}.$$

1.6.2.4 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) is a MH algorithm where the proposals are generated by integrating a Hamiltonian system constructed in such a way that the invariant distribution of P is our desired target π. Since the proposals are generated by integrating a system of ODEs, it is natural to express it in terms of the framework introduced in the previous subsection. In this subsection, we provide context regarding how the HMC involution arises in physics and statistics, and how it relates to subsection 1.6.2.3. This introduction to HMC borrows heavily from [124] and [82], and uses concepts in [4]. For $d \in \mathbb{N}$, let $\mathsf{X}, \mathsf{V} \subset \mathbb{R}^d$ be non-empty, open and connected subsets, and $\mathsf{Z} = \mathsf{X} \times \mathsf{V}$ their Cartesian product, known as *phase space*. We call $x \in \mathsf{X}$ the *position* variables and $v \in \mathsf{V}$ the *momentum* variables and assume they both depend on time $t \in [0, \infty)$. Any

C^2 function $H : \mathsf{Z} \to \mathbb{R}$ is called a *Hamiltonian*, and the following autonomous system of ODEs is called a *Hamiltonian System*

$$\begin{aligned} \dot{x} &= \nabla_v H(x,v) \\ \dot{v} &= -\nabla_x H(x,v). \end{aligned} \tag{1.11}$$

Given an initial configuration at time zero $z(0) = (x_0, v_0)$, a *solution* is a function $z(t)$ that satisfies both the initial condition and the system above. It is easy to show that the Hamiltonian is constant along its solutions. In physics, $H(x,v)$ is interpreted as the total energy of a system of d particles with positions x and momenta v, and decomposed into *potential energy* $V(x)$ and *kinetic energy* $K(x,v)$

$$H(x,v) = V(x) + K(x,v).$$

For the scope of this short introduction, we shall focus on the standard scenario where the kinetic energy does not depend on the position $K(x,v) = K(v)$, although position-dependent formulations can often bring advantages in high-dimensional settings [59]. In this case, the system (1.11) is *separable* and can be integrated using the Leapfrog algorithm (or more precisely, the Generalized Velocity Verlet [82]). Given a step size $\delta > 0$ and a pair $z_n = (x_n, v_n)$, the Leapfrog integrator is a function $\phi_\delta : \mathsf{X} \times \mathsf{V} \to \mathsf{X} \times \mathsf{V}$ that maps $(x_n, v_n) \longmapsto (x_{n+1}, v_{n+1})$ according to the following mechanism

$$\begin{aligned} v_{n+1/2} &= v_n - \frac{\delta}{2} \nabla_x V(x_n) \\ x_{n+1} &= x_n + \delta \nabla_v K(v_{n+1/2}) \\ v_{n+1} &= v_{n+1/2} - \frac{\delta}{2} \nabla_x V(x_{n+1}). \end{aligned}$$

Of course, the display above returns only the next step in the discretization, (x_{n+1}, v_{n+1}), but in practice we would repeat this procedure for a number of *Leapfrog steps* $L \in \mathbb{N}$ so as to reach a certain integration time $T = \delta L$, an operation that we denote as ϕ_δ^L. Coming back to the task of estimating expectations with respect to π, choose a specific form for the energy functions

$$\begin{aligned} V(x) &= -\log \pi(x) \\ K(v) &= -\log \mathcal{N}(v \mid 0, \mathrm{I}_d). \end{aligned}$$

Whenever the covariance matrix of the kinetic energy is the identity, one can indifferently refer to v as the velocity or momentum variable. Three properties of this integrator make it very appealing to be used in the context of MH. Firstly, this integrator is *symmetric*, meaning that if we apply ϕ_δ^L and reverse the velocity, then we can come back exactly to our starting pair by apply ϕ_δ^L one more time. More formally, if we define $\sigma(x,v) = (x,-v)$, then $\psi_{L,\delta}^{\mathrm{HMC}} := \sigma \circ \phi_\delta^L$ is an involution. The second important property is that the integrator is symplectic, which implies it is also volume preserving and hence has unit absolute Jacobian determinant. These two properties together tell us that $\psi_{L,\delta}^{\mathrm{HMC}}$ is a valid candidate for proposing a new state of the Markov chain with product density

$$\mu(x,v) = \pi(x)\mathcal{N}(v \mid 0, \mathrm{I}_d) = \exp(-H(x,v)),$$

which has π as marginal. Being valid, does not mean that it is necessarily a *good* proposal mechanism. However, the Leapfrog integrator is a second order integrator for the solutions of the Hamiltonian, which means that H will be approximately constant along the discretized trajectories, for small step sizes, and hence the MH acceptance ratio is likely to be high.

1.7 Importance Sampling

An alternative to Markov Chain-based sampling is importance-based sampling, which uses the discrepancy between a target and a proposal distribution to construct weighted samples that approximate the target.

1.7.1 Normalized Importance Sampling

Let μ and η be two target probability distributions on $(\mathsf{X}, \mathscr{X})$ with $\eta \gg \mu$. Importance Sampling (IS) is a strategy to estimate an expectation of a test functions $h : \mathsf{X} \to \mathbb{R}$ with respect to μ using samples from η, and it consists of two steps:

1. *Change of variables*: rewrite the expectation using a change of measure

$$\mu(h) = \eta\left(h \frac{d\mu}{d\eta}\right) \tag{1.12}$$

2. *Empirical approximation*: substitute η with an empirical approximation $\hat{\eta}^N$

$$\hat{\eta}^N(dx) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X^{(i)}}(dx) \qquad \text{where} \qquad X^{(i)} \sim \eta(dx),$$

   where the samples $X^{(i)}$ are typically assumed independent in simple examples, although in more complicated settings they can have strong dependence and be obtained in a sequential fashion [120].

Overall, $\mu(h)$ can be estimated with the following IS estimate:

$$\mu(h) \approx \frac{1}{N}\sum_{i=1}^{N} h(X^{(i)}) w(X^{(i)}) \quad \text{where} \quad w = \frac{d\mu}{d\eta} \quad \text{and} \quad X^{(i)} \sim \eta(dx).$$

We call $w(X^{(i)})$ the importance weights. Notice, as usual, that by setting $h = \mathbb{1}_{\mathsf{A}}$ for any measurable set $\mathsf{A} \in \mathscr{X}$ we recover the empirical approximation for the target

$$\hat{\mu}^N(dx) = \frac{1}{N}\sum_{i=1}^{N} w(X^{(i)}) \delta_{X^{(i)}}(dx).$$

A common assumption is the existence of a dominating Lebesgue measure $\mathrm{Leb}_{\mathsf{X}} \gg \eta \gg \mu$. It may seem that one requires $\mathrm{Leb}_{\mathsf{X}}$ and η to be equivalent in order to derive the importance sampling weights by leveraging the fact that

$$\frac{\mathrm{Leb}_{\mathsf{X}}(dx)}{\eta(dx)} = \left(\frac{\eta(dx)}{\mathrm{Leb}_{\mathsf{X}}(dx)}\right)^{-1}$$

whenever $\eta \equiv \text{Leb}_{\mathsf{X}}$. It turns out that this is not necessary, and as long as all measures are σ-finite then the set of points where the density of μ is positive but the density of η is zero (both densities are with respect to Leb_{X}) has measure zero under Leb_{X}, and therefore (see Problem 32.6, part c in [18])

$$w(x) = \frac{d\mu}{d\eta}(x) = \mathbb{1}_{\{y \in \mathsf{X} \,:\, \eta(y) > 0\}}(x) \frac{\mu(x)}{\eta(x)}.$$

In the expression above, we have used the convention (see Section 1.4) that $\mu(x)$ and $\eta(x)$ denote densities with respect to Leb_{X}.

1.7.2 Self-Normalised Importance Sampling

The same ideas hold when one can only compute densities up to a normalizing constant, with the caveat that one needs to estimate the unknown ratio of normalizing constants. This can be done, once more, using IS, but leads to a biased estimate for $\mu(h)$, as we shall see next. Assume the existence of sigma-finite measures $\tilde{\mu}$ and $\tilde{\eta}$ that assign to each measurable set $\mathsf{A} \in \mathscr{X}$ a size proportional to that of μ and η respectively

$$\begin{aligned} \tilde{\mu}(\mathsf{A}) &= Z_\mu \mu(\mathsf{A}) && \forall\, \mathsf{A} \in \mathscr{X} \\ \tilde{\eta}(\mathsf{A}) &= Z_\eta \eta(\mathsf{A}) && \forall\, \mathsf{A} \in \mathscr{X} \end{aligned}$$

for two constants $Z_\mu, Z_\eta \in \mathbb{R}_+$ known as *normalizing constants*, whose definitions can be found by setting $\mathsf{A} = \mathsf{X}$ above

$$\begin{aligned} Z_\mu &= \int_{\mathsf{X}} \tilde{\mu}(dx) \\ Z_\eta &= \int_{\mathsf{X}} \tilde{\eta}(dx). \end{aligned}$$

One often has only access to the densities of these measures rather than the original ones

$$\begin{aligned} \tilde{\mu}(x) &= Z_\mu \mu(x) \\ \tilde{\eta}(x) &= Z_\eta \eta(x), \end{aligned}$$

which means that to evaluate the weights $w(x)$ one needs to know the ratio of normalizing constants Z_η / Z_μ (or equivalently, the normalizing constant of the Radon-Nikodym derivative $d\mu/d\eta$)

$$w(x) = \frac{Z_\eta}{Z_\mu} \cdot \frac{\tilde{\mu}(x)}{\tilde{\eta}(x)}.$$

Luckily this ratio can be estimated using the change of measure (1.12) and by choosing h to be the constant function 1

$$\frac{Z_\mu}{Z_\eta} = \eta\left(\frac{\tilde{\mu}}{\tilde{\eta}}\right)$$

One can then use samples $X^{(1)},\dots,X^{(N)}$ from η to estimate the normalizing constant

$$\frac{Z_\mu}{Z_\eta} \approx \frac{1}{N}\sum_{i=1}^{N} w(X^{(i)}) \qquad \text{where } w(X^{(i)}) = \frac{\tilde{\mu}(X^{(i)})}{\tilde{\eta}(X^{(i)})},$$

and plug this estimate into the change of measure above, leading to the following biased estimate

$$\mu(h) \approx \sum_{i=1}^{N} h(X^{(i)})W(X^{(i)}) \qquad \text{where} \qquad W(X^{(i)}) := \frac{w(X^{(i)})}{\sum_{j=1}^{N} w(X^{(j)})}$$

are called the *normalized weights*, and this estimate is sometimes known as *self-normalized IS*. Clearly, the empirical approximation of the target in this case becomes

$$\hat{\mu}^N(dx) = \sum_{i=1}^{N} W(X^{(i)})\delta_{X^{(i)}}(dx).$$

1.7.3 Effective Sample Size

The Effective Sample Size (ESS) [91] is a metric used when assessing the efficacy of an importance sampling estimator . It was originally derived in the unpublished note [80] using the Delta method, and then later popularised by [90]. Given a self-normalized importance sampling estimator

$$\hat{h}_{\text{IS}} = \frac{\sum_{i=1}^{N} h(X^{(i)})w(X^{(i)})}{\sum_{i=1}^{N} w(X^{(i)})} \qquad w(X^{(i)}) = \frac{\tilde{\mu}(X^{(i)})}{\tilde{\eta}(X^{(i)})} \qquad X^{(i)} \sim \eta,$$

the ESS is a way to compare the variance of this estimator with the variance of that one would obtain by performing ordinary Monte Carlo $\mathbb{V}[\hat{h}_{\text{MC}}]$

$$\text{ESS} = N\frac{\mathbb{V}[\hat{h}_{\text{MC}}]}{\mathbb{V}[\hat{h}_{\text{IS}}]}. \tag{1.13}$$

The ESS indicates the number of equivalent IID samples from the target that would return an estimator with the same variance. As shown in the Appendix A.2, using the Delta method [119] one can find a reasonable approximation to $\mathbb{V}[\hat{h}_{\text{IS}}]$ that allows one to estimate the ESS without knowledge of the test function, since

$$\mathbb{V}[\hat{h}_{\text{IS}}] \approx \mathbb{V}[\hat{h}_{\text{MC}}](1+\mathbb{V}[W]),$$

where $\mathbb{V}[W]$ is the variance of the normalized weights, which leads to the usual expression

$$\text{ESS} \approx \frac{\left(\sum_{i=1}^{n} w(X^{(i)})\right)^2}{\sum_{i=1}^{N} w(X^{(i)})^2}, \tag{1.14}$$

as shown in the Appendix A.2.

1.8 Sequential Monte Carlo Samplers

Sequential Monte Carlo (SMC) samplers [38, 39, 45, 46] are an alternative to MCMC samplers for targeting a distribution μ on $(\mathsf{X}, \mathscr{X})$. The idea is to sample from a base distribution ν, chosen by the practitioner, and then use a combination of IS and resampling to approximate a sequence of distributions $\{\mu_n : n \in [\![P]\!]\}$ interpolating between a relatively simple probability measure μ_1, whose density we can evaluate up to a normalizing constant, and our actual target distribution $\mu_P = \mu$. Common implementations of SMC samplers adaptively choose the interpolating distributions μ_n and terminate once a stopping criterion is reached, whereas the original algorithm targeted a predetermined sequence of target distributions. For the time being, we shall assume the latter implementation for simplicity.

1.8.1 Kernels and Extended Distributions

Let $M_n : \mathsf{X} \times \mathscr{X} \to [0,1]\}$ for $n \in [\![2, P]\!]$ be a collection of Markov kernels termed *forward kernels*, and let $L_n : \mathsf{X} \times \mathscr{X} \to [0,1]$ for $n \in [\![1, P-1]\!]$ be termed *backward kernels*. For each n define the following augmented distributions on $(\mathsf{X}^n, \mathscr{X}^{\otimes n})$

$$\mu_n(dx_{1:n}) = \mu_n(dx_n) \prod_{k=1}^{n-1} L_k(x_{k+1}, dx_k) \qquad n \in [\![P]\!] \tag{1.15}$$

$$\eta_n(dx_{1:n}) = \nu(dx_1) \prod_{k=1}^{n-1} M_{k+1}(x_k, dx_{k+1}) \qquad n \in [\![P]\!] \tag{1.16}$$

which we call the *joint target* and the *joint proposal* at iteration n, and by construction $\eta_1 = \nu$. Notice that $\mu_n(dx_n)$ is a marginal of $\mu_n(dx_{1:n})$ as can be seen by integrating out $x_{1:n-1}$ and using the fact that $\{L_{n-1}\}$ are *probability* kernels. In addition, notice that by (improperly) defining $h_{1:n}(x_{1:n}) = h(x_n)$ we see that expectations of test functions are preserved, in the sense that

$$\mu_n(h_{1:n}) = \int_{\mathsf{X}^n} h_{1:n}(x_{1:n}) \mu_n(dx_{1:n}) = \int_{\mathsf{X}} h(x_n) \mu_n(dx_n) = \mu_n(h_n). \tag{1.17}$$

This consideration and the marginalization property discussed above, should give us a sense of confidence that the extended distributions are "compatible" with their marginals. We shall require the following two conditions

$$\mu_n(dx_{1:n}) \ll \eta_n(dx_{1:n}) \qquad \forall\, n \in [\![P]\!],$$

$$\mu_n(dx_n) L_{n-1}(x_n, dx_{n-1}) \ll \mu_{n-1}(dx_{n-1}) M_n(x_{n-1}, dx_n) \qquad \forall\, n \in [\![P]\!],$$

as well as assume that we either know Z_ν, or we can compute the normalized density $\nu(x)$.

1.8.2 Extended Importance Sampling and Incremental Weights

Consider standard IS using $\eta_n(dx_{1:n})$ as importance distribution and $\mu_n(dx_{1:n})$ as target. The importance weights can be written recursively

$$\begin{aligned} w_n(x_{1:n}) &= \frac{\mu_n(dx_{1:n})}{\eta_n(dx_{1:n})} \\ &= \omega_n(x_{n-1}, x_n) w(x_{1:n-1}), \end{aligned}$$

where we have defined the *incremental weights* as

$$\omega_n(x_{n-1}, x_n) = \frac{\mu_n(dx_n) L_{n-1}(x_n, dx_{n-1})}{\mu_{n-1}(dx_{n-1}) M_n(x_{n-1}, dx_n)}. \tag{1.18}$$

The exact form of these weights depend on two factors: the type of forward and backward kernel, and the strength of our assumptions. For our purposes, we shall always assume that the incremental weights are well-defined Radon-Nikodym derivatives

$$\mu_n(dx_n) L_{n-1}(x_n, dx_{n-1}) \ll \mu_{n-1}(dx_{n-1}) M_n(x_{n-1}, dx_n), \qquad \forall\, n \in [\![2, P]\!].$$

A common assumption is that all four measures in Equation (1.18) are absolutely continuous with respect to the Lebesgue measure $\mathrm{Leb}_{\mathsf{X}}$, thus leading to

$$\omega_n(x_{n-1}, x_n) = \frac{\mu_n(x_n) \ell_{n-1}(x_n, x_{n-1})}{\mu_{n-1}(x_{n-1}) m_n(x_{n-1}, x_n)},$$

where ℓ_{n-1} and m_n are the *normalized* Radon-Nikodym derivatives of L_{n-1} and M_n respectively. Similarly to the considerations for IS, one typically has access only to unnormalized density functions of the target and the kernels, but all the canonical choices of the latter are such that one does not need to worry about the normalizing constants of either the forward or backward kernels. This is because one can choose the kernels so that they disappear from ω_n, as we will see in section 1.8.6, and thus we shall disregard their normalizing constants.

$$\begin{aligned} w_n(x_{1:n}) &= \frac{\tilde{\mu}_n(x_n) \prod_{k=1}^{n-1} \ell_k(x_{k+1}, x_k)}{\nu(x_1) \prod_{k=1}^{n-1} m_{k+1}(x_k, x_{k+1})} \cdot \frac{1}{Z_{\mu_n}} \\ &= \frac{\tilde{\mu}_n(x_n) \ell_{n-1}(x_n, x_{n-1})}{\tilde{\mu}_{n-1}(x_n) m_n(x_{n-1}, x_n)} \cdot \frac{\tilde{\mu}_{n-1}(x_{n-1}) \prod_{k=1}^{n-2} \ell_k(x_{k+1}, x_k)}{\nu(x_1) \prod_{k=1}^{n-2} m_{k+1}(x_k, x_{k+1})} \cdot \frac{1}{Z_{\mu_n}} \\ &= \frac{1}{Z_{\mu_n}} \cdot \tilde{\omega}_n(x_{n-1}, x_n) w_{n-1}(x_{1:n-1}), \end{aligned} \tag{1.19}$$

where we have defined the unnormalized incremental weight

$$\tilde{\omega}_n(x_{n-1}, x_n) = \frac{\tilde{\mu}_n(x_n) \ell_{n-1}(x_n, x_{n-1})}{\tilde{\mu}_{n-1}(x_{n-1}) m_n(x_{n-1}, x_n)}.$$

In this extended framework, since no resampling is performed, one could simply estimate the final normalizing constant $Z_{\mu_P} = Z_\mu$

$$\begin{aligned} Z_\mu &= \int_{\mathsf{X}} \mu_P(dx_P) \\ &= \int_{\mathsf{X}^P} \mu_P(dx_P) \prod_{k=1}^{P-1} L_k(x_{k+1}, dx_k) \\ &= \int_{\mathsf{X}^P} w_P(x_{1:P}) \eta_P(dx_{1:P}) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} w_P(X_{1:P}^{(i)}) \qquad\qquad X_{1:P}^{(i)} \sim \nu(dx_1) \prod_{k=1}^{P-1} M_{k+1}(x_k, dx_{k+1}). \end{aligned}$$

If one designs the kernels correctly, one does not need to assume that the corresponding random measures are absolutely continuous with respect to $\mathrm{Leb}_{\mathsf{X}}$ since they disappear from the ratio. To aid exposition, we shall refer to this setting as the *standard assumptions*.

Definition 1.1 (Standard Assumptions)**.** We refer to the standard assumptions whenever the targets distributions $\mu_1, \ldots, \mu_P$ admit densities with respect to $\mathrm{Leb}_{\mathsf{X}}$ that can be computed only up to a normalizing constant, and the Markov kernels M_n and L_{n-1} are chosen in such a way as to disappear from ω_n.

As long as the Radon-Nikodym derivative of the incremental weights exist, one could compute the paths $X_{1:P}^{(i)}$ either in parallel or sequentially. The sequential implementation closely resembles and helps with understanding that of an SMC sampler, so we detail it in Algorithm 1 for the case in which we can evaluate normalized densities $\mu_1(x_1)$ and $\nu(x_1)$.

Algorithm 1: Extended Importance Sampling

1 Sample from base distribution $X_1^{(i)} \sim \nu$ for $i \in [\![N]\!]$ and compute initial weights

$$w_1(X_1^{(i)}) = \frac{\mu_1(X_1^{(i)})}{\nu(X_1^{(i)})} \qquad i \in [\![N]\!].$$

2 **for** $n = 1, \ldots, P-1$ **do**

3 Extend particles: sample $X_{n+1}^{(i)} \sim M_{n+1}(X_n^{(i)}, \cdot)$ and set $X_{1:n+1}^{(i)} = (X_1^{(i)}, \ldots, X_{n+1}^{(i)})$.

4 Compute incremental weights

$$\omega_{n+1}(X_n^{(i)}, X_{n+1}^{(i)}) = \frac{d\mu_{n+1} \oplus L_n}{dM_{n+1} \otimes \mu_n}(X_n^{(i)}, X_{n+1}^{(i)}).$$

5 Update weights: set $w_{n+1}(X_{1:n+1}^{(i)}) = \omega_{n+1}(X_n^{(i)}, X_{n+1}^{(i)}) w_n(X_{1:n}^{(i)})$.

6 **end**

7 Return weighted particles $\left\{\left(X_{1:P}^{(i)}, w_P(X_{1:P}^{(i)})\right) : i \in [\![N]\!]\right\}$.

As typical for IS, Algorithm 1 is well-known to suffer from particle-degeneracy issues [31] due to vanishing weights. Resampling mechanisms are a common strategy used to counteract this

problem and when introduced within Algorithm 1, lead to an SMC sampler. Before introducing resampling, however, we need to talk about a very useful tool in studying SMC samplers: Feynman-Kac formula.

1.8.3 Feynman-Kac Formula

At the beginning of this chapter we assumed $\mu_n(dx_{1:n}) \ll \eta_n(dx_{1:n})$ for any $n \in [\![P]\!]$, hence by the Radon-Nikodym theorem we can write the extended target as a change of measure from the extended proposal

$$\mu_n(dx_{1:n}) = w_1(x_1)\left[\prod_{k=2}^{n}\omega_k(x_{k-1},x_k)\right]\eta_n(dx_{1:n}) \qquad n \in [\![2,P]\!].$$

It is also possible to write the proposal distribution recursively

$$\eta_n(dx_{1:n}) = \eta_{n-1}(dx_{1:n-1})M_n(x_{n-1},dx_n) \qquad n \in [\![2,P]\!].$$

In turn, these two formulas can be used to write a recursion formula for the extended targets

$$\begin{aligned}\mu_n(dx_{1:n}) &= w_1(x_1)\left[\prod_{k=2}^{n}\omega_k(x_{k-1},x_k)\right]\eta_n(dx_{1:n}) \\ &= \omega_n(x_{n-1},x_n)\left\{w_1(x_1)\left[\prod_{k=2}^{n-1}\omega_k(x_{k-1},x_k)\right]\eta_{n-1}(dx_{1:n-1})\right\}M_n(x_{n-1},dx_n) \\ &= \omega_n(x_{n-1},x_n)\mu_{n-1}(dx_{1:n-1})M_n(x_{n-1},dx_n)\end{aligned}$$

This expression is known as the Feynman-Kac formula and can be understood as a two-step procedure to go from $\mu_{n-1}(dx_{1:n-1})$ to $\mu_n(dx_{1:n})$.

1. *Extension*: from $\mu_{n-1}(dx_{1:n-1})$ on $(\mathsf{X}^{n-1},\mathscr{X}^{\otimes n-1})$ to $\mu_{n-1}(dx_{1:n-1})M_n(x_{n-1},dx_n)$ on $(\mathsf{X}^{n},\mathscr{X}^{\otimes n})$

2. *Change of Measure*: from $\mu_{n-1}(dx_{1:n-1})M_n(x_{n-1},dx_n)$ to $\mu_n(dx_{1:n})$ via $\omega_n(x_{n-1},x_n)$

It is more convenient to write this formula to go from $\mu_{n-1}(dx_{n-1})$ to $\mu_n(dx_{n-1:n})$ for two reasons. Firstly, due to the sequential structure of the algorithm, one only needs to show correctness inductively and since the first step is a simple importance sampling step, all that remains to show is that the samples at the end of each iteration are a good approximation to the interpolating target μ_n. Secondly, one typically performs resampling at each iteration, making the other terms in the Feynman-Kac formula redundant. Simply integrate out $x_{1:n-2}$ from the formula above, and use the fact that the extended target $\mu_{n-1}(dx_{1:n-1})$ has $\mu_{n-1}(dx_{n-1})$ as marginal

$$\mu_n(dx_{n-1:n}) = \omega_n(x_{n-1},x_n)\mu_{n-1}(dx_{n-1})M_n(x_{n-1},dx_n). \tag{1.20}$$

1.8.4 Resampling

The literature on Importance Resampling (IR) and Sequential Importance Resampling (SIR) is rich in information about the advantages of resampling in counteracting particle degeneracy, which are of fundamental importance in SMC samplers too. However one important advantage specific to this scenario, if resampling is performed at each step, is that after performing resampling the Feynman-Kac formula (1.20) now allows us to estimate expectations and normalizing constants with simplicity. Firstly, notice that using Feynman-Kac formula expectations $\mu_n(h)$ can be approximate as follows:

$$\begin{aligned}\mu_n(h) &= \int_{\mathsf{X}^2} h(x_{n-1:n})\omega_n(x_{n-1},x_n)\mu_{n-1}(dx_{n-1})M_n(x_{n-1},dx_n)\\ &\approx \int_{\mathsf{X}^2} h(x_{n-1:n})\omega_n(x_{n-1},x_n)\hat{\mu}^N_{n-1}(dx_{n-1})M_n(x_{n-1},dx_n)\end{aligned}$$

If one could sample from $\hat{\mu}^N_{n-1}(dx_{n-1})M_n(x_{n-1},dx_n)$ then expectations could be approximated using only the incremental weights $\omega_n(x_{n-1},x_n)$, rather than the full weights $w_n(x_{1:n})$. This is exactly what resampling allows us to do.

At the beginning of the $(n-1)^{\text{th}}$ iteration (excluding the very first iteration) of Algorithm 1, a set of weighted particles $\{(X^{(i)}_{1:n-1}, w_{n-1}(X^{(i)}_{1:n-1})) : i \in [\![N]\!]\}$ will be available to approximate $\mu_{n-1}(dx_{1:n-1})$

$$\mu_{n-1}(dx_{1:n-1}) \approx \hat{\mu}^N_{n-1}(dx_{1:n-1}) := \frac{1}{N}\sum_{i=1}^N w_{n-1}(X^{(i)}_{1:n-1})\delta_{X^{(i)}_{1:n-1}}(dx_{1:n-1}).$$

Resampling consists of sampling N times from $\hat{\mu}^N_{n-1}(dx_{1:n-1})$

$$\check{X}^{(i)}_{1:n-1} \sim \hat{\mu}^N_{n-1}(dx_{1:n-1}) \qquad i \in [\![N]\!], \tag{1.21}$$

thus obtaining a new unweighted sample approximating $\mu_{n-1}(dx_{1:n-1})$

$$\mu_{n-1}(dx_{1:n-1}) \approx \frac{1}{N}\sum_{i=1}^N \delta_{\check{X}^{(i)}_{1:n-1}}(dx_{1:n-1}).$$

As before, we have assumed that one can compute the weights without worrying about the normalizing constants of the target or the Markov kernels. Under *standard assumptions*, the particle approximation will use the *normalized weights*

$$\hat{\mu}^N_{n-1}(dx_{1:n-1}) := \sum_{i=1}^N W_{n-1}(X^{(i)}_{1:n-1})\delta_{X^{(i)}_{1:n-1}}(dx_{1:n-1}).$$

Resampling can be performed in many ways and we refer to [31] for more details, however it always comes down to sampling a set of ancestor indices $A^{(i)}_{n-1} \in [\![N]\!]$ proportionally to the corresponding weights

$$\mathbb{P}(A^{(i)}_{n-1} = j) \propto w_{n-1}(X^{(j)}_{1:n-1}),$$

and then setting $\breve{X}_{1:n-1}^{(i)} = X_{1:n-1}^{A_{n-1}^{(i)}}$. By performing resampling before using the forward kernel, we are effectively able to obtain a particle approximation of $\hat{\mu}_{n-1}^N(dx_{n-1})M_n(x_{n-1}, dx_n)$ (and sample from it) and therefore expectations can be estimated as follows

$$\mu_n(h) \approx \frac{1}{N}\sum_{i=1}^{N} h(\breve{X}_{n-1}^{(i)})\omega_n(\breve{X}_{n-1}^{(i)}, X_n^{(i)}) \qquad\qquad (\breve{X}_{n-1}^{(i)}, X_n^{(i)}) \sim \hat{\mu}_{n-1}^N \otimes M_n,$$

where ω_n would need to be normalized when using *standard assumptions*. The original SMC sampler, for the scenario where resampling is performed at each step, is given in Algorithm 2.

Algorithm 2: SMC sampler

Sample from base distribution $X_1^{(i)} \sim \nu$ for $i \in [\![N]\!]$ and compute initial weights

$$w_1(X_1^{(i)}) = \frac{\tilde{\mu}_1(X_1^{(i)})}{\nu(X_1^{(i)})} \qquad i \in [\![N]\!].$$

for $n = 1, \ldots, P-1$ **do**

  Normalize weights

$$W_n(X_{1:n}^{(i)}) = \frac{w_n(X_n^{(i)})}{\sum_{j=1}^{N} w_n(X_{1:n}^{(j)})}$$

  Sample $A_n^{(i)}$ such that $\mathbb{P}(A_n^{(i)} = j) = W_n(X_{1:n}^{(j)})$ and set $\breve{X}_{1:n}^{(i)} = X_{1:n}^{A_n^{(i)}}$ for $i \in [\![N]\!]$.

  Sample $X_{n+1}^{(i)} \sim M_{n+1}(\breve{X}_n^{(i)}, \cdot)$ and set $X_{1:n+1}^{(i)} = (\breve{X}_{1:n}^{(i)}, X_{n+1}^{(i)})$.

  Compute (incremental) weights

$$\omega_{n+1}(\breve{X}_n^{(i)}, X_{n+1}^{(i)}) = \frac{d\mu_{n+1} \oplus L_n}{dM_{n+1} \otimes \mu_n}(\breve{X}_n^{(i)}, X_{n+1}^{(i)}).$$

  Set $w_{n+1}(X_{1:n+1}^{(i)}) = \omega_{n+1}(\breve{X}_n^{(i)}, X_{n+1}^{(i)})$.

end

Return weighted particles $\left\{\left(X_{1:P}^{(i)}, w_P(X_{1:P}^{(i)})\right) : i \in [\![N]\!]\right\}$.

When performing resampling, the algorithm is intrinsically sequential and therefore the normalizing constants of the interpolating distributions do not cancel out as they did in Equation (1.19), but the ratio of consecutive normalizing constants can be estimated as in IS. At iterations in which resampling is performed the ratio $Z_{\mu_{n-1}}/Z_{\mu_n}$ can be estimated as the total sum of the incremental weights, otherwise of the previous weights multiplied by the incremental weights, as described in [38].

1.8.5 Distributions involved

1.8.5.1 Without resampling

When no resampling is performed, and thus we are performing extended IS as in Algorithm 1, the marginal distribution of particles $X_n^{(i)}$ is

$$\eta_n(dx_n) = \int_{x_{1:n-1} \in \mathsf{X}^{n-1}} \nu(dx_1) \prod_{k=1}^{n-1} M_{k+1}(x_k, dx_{k+1}) \qquad n \in [\![P]\!], \tag{1.22}$$

and we call this the *marginal proposal*. Naturally, as long as $(\mathsf{X}^{n-1}, \mathscr{X}^{\otimes(n-1)})$ is *nice* space, there exists a Markov probability kernel $\eta_{1:n-1|n} : \mathsf{X} \times \mathscr{X}^{\otimes(n-1)} \to [0,1]$ such that

$$\eta_n(dx_{1:n}) = \eta_n(dx_n)\eta_{1:n-1|n}(x_n, dx_{1:n-1}), \tag{1.23}$$

and we call this the *conditional proposal*.

1.8.5.2 With resampling

Suppose that up to iteration $n-1$ we have not performed resampling, and therefore $X_{n-1}^{(i)} \sim \eta_n(dx_n)$. If we decide to perform resampling at the next iteration, then the distribution of the resampled particles will be like the one shown in Equation (1.21)

$$\check{X}_{1:n-1}^{(i)} \sim \hat{\mu}_{n-1}^N(dx_{1:n-1}) \qquad i \in [\![N]\!]. \tag{1.24}$$

The SMC sampler then proceeds by using the forward kernel to move these particles

$$X_n^{(i)} \sim M_n(\check{X}_{n-1}^{(i)}, dx_n),$$

which means that the marginal distribution of $X_n^{(i)}$ will be

$$\hat{\mu}_{n-1}^N M_n(dx_n) = \int_{x_{1:n-1} \in \mathsf{X}^{n-1}} \hat{\mu}_{n-1}^N(dx_{1:n-1}) M_n(x_{n-1}, dx_n),$$

which makes sense because this means that they are *approximately* distributed according to $\mu_{n-1}M_n$, and we shall use this remark in the next section to help us choose good backward kernels.

Finally, if the last time that resampling was performed was at iteration $m < n$, then the marginal proposal distribution at iteration n will be

$$X_n^{(i)} \sim \int_{x_{1:n-1} \in \mathsf{X}^{n-1}} \hat{\mu}_{m-1}^N(dx_{1:m-1}) \prod_{k=m}^{n} M_k(x_{k-1}, dx_k).$$

1.8.6 Popular choices for forward and backward kernels

As mentioned earlier, the presence of forward and backward kernels in the expression for the incremental weights means that these need to be chosen carefully. All popular choices of M_n and L_{n-1} are such that their densities disappear from the ratio. There seems to be a lot of confusion in the literature about the form of the incremental weights for some popular choices of M_n and L_{n-1}, hence we aim to make some clarity here.

1.8.6.1 Optimal Backward Kernel

It is possible to find an expression for the backward kernel that minimizes the variance of the weights w_n in IS, i.e. when no resampling is performed. This is termed the *optimal backward kernel* and although it is typically impossible to use, it can provide guidance in choosing good backward kernels. As shown in Appendix A.5.1, the optimal backward kernel is

$$L_{n-1}^{\text{opt}}(x_n, dx_{n-1}) = \frac{\eta_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\eta_n(dx_n)} \tag{1.25}$$

where $\eta_n(dx_n)$ is defined as in equation (1.22). The IS weights are then given by

$$w_n(x_{1:n}) = \frac{\mu_n(dx_n)}{\eta_n(dx_n)},$$

This backward kernel is impractical to use because one cannot in general evaluate $\eta_n(dx_n)$ due to the intractability of the integral.

1.8.6.2 Near-Optimal Backward Kernel

The near-optimal backward kernel [39] is often a more practical choice than L_{n-1}^{opt}

$$L_{n-1}^{\text{near-opt}}(x_n, dx_{n-1}) = \frac{\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\mu_{n-1}M_n(dx_n)}, \tag{1.26}$$

and as shown in Appendix A.5.2 it leads to the following incremental weights

$$\omega_n(x_{n-1}, x_n) = \frac{d\mu_n}{d\mu_{n-1}M_n}(x_n), \tag{1.27}$$

notice that in this case the incremental weights only depend on x_n and not x_{n-1}. Most commonly, this choice is combined with the assumption that M_n leaves μ_{n-1} invariant leading to $\omega_n(x_{n-1}, x_n) = d\mu_n/d\mu_{n-1}(x_n)$.

1.8.6.3 Reversible triplet

Another choice is to let (μ_n, M_n, L_{n-1}) to be a reversible triplet (see Theorem D.4), meaning that

$$\mu_n(dx_{n-1})M_n(x_{n-1}, dx_n) = \mu_n(dx_n)L_{n-1}(x_n, dx_{n-1}),$$

which implies that M_n leaves μ_n invariant, and L_{n-1} is the reversal kernel. With this choice, the incremental weights become

$$\omega_n(x_{n-1}, x_n) = \frac{\mu_n(dx_n)L_{n-1}(x_n, dx_{n-1})}{\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)} = \frac{\mu_n(dx_{n-1})}{\mu_{n-1}(dx_{n-1})}. \tag{1.28}$$

This is perhaps the most popular choice in applications. We denote it as

$$L_{n-1}^{\text{rev}}(x_n, dx_{n-1}) = \frac{\mu_n(dx_{n-1})M_n(x_{n-1}, dx_n)}{\mu_n(dx_n)}.$$

It's popularity is due to the fact that the weights are easy to compute and allows using custom Metropolis-Hastings kernels as forward kernels.

1.8.6.4 M_n leaves μ_{n-1} invariant

Another choice commonly used in the literature is M_n leaving μ_{n-1} invariant. This choice is often confused either with L_{n-1}^{opt} or with L_{n-1}^{rev}. The backward kernel is

$$L_{n-1}(x_n, dx_{n-1}) = \frac{\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\mu_{n-1}(dx_n)}.$$

and as shown in Appendix A.5.3 the incremental weights are

$$\omega(x_{n-1}, x_n) = \frac{\mu_n(x_n)}{\mu_{n-1}(x_n)}. \tag{1.29}$$

1.8.6.5 Another near-optimal backward kernel

One may wish to minimize the variance of the incremental weights. Recall that the incremental weights are defined as a Radon-Nikodym derivative and as such, this has minimum variance, with respect to the distribution of $x_{n-1:n}$, whenever it is constant. The backward kernel

$$L_{n-1}^{\text{near-opt}}(x_n, dx_{n-1}) = \frac{\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\mu_n(dx_n)}.$$

does the job, as shown in Appendix A.5.4, and leads to constant weights.

1.8.7 Estimating acceptance probability

In a standard SMC sampler with a MH kernel, the acceptance probability is straightforward to estimate. Consider Algorithm 2 where the forward kernel consists of $T \in \mathbb{Z}_+$ steps of a RWM kernel, meaning that for each particle $\check{X}_n^{(i)}$ we use the MH accept-reject mechanism T times and thus obtain binary flags $a_{n,i,k} \in \{0,1\}$ for $i \in [\![N]\!]$ and $k \in [\![T]\!]$, with $a_{n,i,k} = 0$ indicating that the i^{th} particle rejected the k^{th} proposal, and $a_{n,i,k} = 1$ indicating an acceptance instead. The average acceptance probability for the SMC sampler at step n can then be estimated as

$$\hat{a}_n = \frac{1}{NT}\sum_{i=1}^{N}\sum_{k=1}^{T} a_{n,i,k}, \tag{1.30}$$

which can be used both to diagnose the performance of the algorithm and to devise adaptation strategies for the hyperparameters of the MH kernel, such as the step size, in the spirit of [5].

1.8.8 Estimating ESJD

Suppose that our SMC sampler's forward kernel consists of $T \in \mathbb{Z}_+$ steps of a MH kernel, meaning that for each particle $\check{X}_n^{(i)}$ we generate a Markov Chain and set X_{n+1} to the final state of this chain. It is possible to estimate the ESJD of each of the N chains as

$$\text{ESJD}_{n,i} \approx \frac{1}{T}\sum_{k=1}^{T}\left\|\check{X}_{n,k-1}^{(i),*} - \check{X}_{n,k-1}^{(i)}\right\|^2 a_{\text{MH}}(\check{X}_{n,k-1}^{(i)}, \check{X}_{n,k-1}^{(i),*}),$$

where $\check{X}_{n,k}^{(i)}$ denotes the k^{th} state of the Markov chain generated starting from $\check{X}_n^{(i)}$ and $\check{X}_{n,k}^{(i),*}$ denotes the proposal. A measure of expected travelled distance for the SMC sampler is then given by the average ESJD across all particles (which we refer simply to as the ESJD of the SMC algorithm)

$$\text{ESJD}_n \approx \frac{1}{NT} \sum_{i=1}^{N} \sum_{k=1}^{T} \left\| \check{X}_{n,k-1}^{(i),*} - \check{X}_{n,k-1}^{(i)} \right\|^2 a_{\text{MH}}(\check{X}_{n,k-1}^{(i)}, \check{X}_{n,k-1}^{(i),*}).$$

Importantly, notice that estimating the ESJD using the framework developed in subsection 1.6.2.1 is sensible in this setting because the forward kernel is a MH kernel that leaves the current target invariant, that is we are using a reversible triplet (μ_n, M_n, L_{n-1}). Although our initial particles in Algorithm 2 are distributed according to ν and not π, assuming that the chain has reached stationarity is a common assumption and can be sensible after an adequate burn in for a well-mixing chain. Nonetheless, the proof in Appendix A.3 relies on the approximation

$$\mathbb{E}_{\pi \otimes P}[\|X - Y\|^2] = \mathbb{E}_{\pi \otimes Q}[\|X - Y\|^2 a(X,Y)] \approx \mathbb{E}_{\nu \otimes Q}[\|X - Y\|^2 a(X,Y)],$$

which, mixing aside, is only sensible when P leaves the current target invariant. For instance, when using the near-optimal backward kernel 1.26 and a forward kernel that does not leave μ_n invariant, one cannot estimate the ESJD in this manner.

CHAPTER 2

APPROXIMATE MANIFOLD SAMPLING

In this chapter, we explain the problem of sampling from a distributions concentrated around a submanifold and how it arises in various applications. We then introduce exact and approximate manifold sampling algorithms.

2.1 Introduction

2.1.1 Motivation and Aim

Sampling from probability distributions defined on a lower-dimensional manifold is of great importance in Molecular Dynamics (MD) [87], Approximate Bayesian Computation (ABC) [63], Bayesian Inverse Problems (BIP) [8], diffusion models [64], and topological statistics [40]. Initially developed in MD, improved and corrected versions of the sampling algorithms have been proposed [153]. These sampling algorithms can, however, be extremely computationally expensive as they require two calls to optimization routines at each step.

The aim of this thesis is to avoid these costly operations by developing sampling algorithms for a relaxation of the problem. Namely, we sample from a filamentary distribution [96], i.e. one supported on a small neighbourhood around the manifold and thus we trade-off a small bias for a much more efficient algorithm.

2.1.2 Applications

The task of sampling from a probability distribution highly concentrated around a manifold naturally arises in many fields of statistics. In this section, we focus on two particular examples: ABC and BIP.

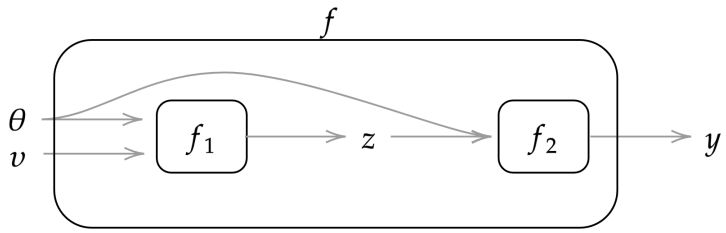

Figure 2.1: A simulator is a type of data-generating process where model parameters and random seeds combine to generate latent variables, which then can interact themselves with the parameters to generate the data.

Example 2.1.2.1 (Approximate Bayesian Computation)**.** *Consider performing Bayesian inference for a statistical model with parameters $\theta \in \Theta$ given some observed data $y^* \in \mathcal{Y}$, where the data generating process is known to be a simulator. A simulator is typically a piece of computer code that takes as input the model parameters θ and a set of random seeds $\upsilon \in \Upsilon$, and outputs data as a deterministic function of them, i.e. $y = f(\theta, \upsilon)$. Commonly, within the data-generating process some intermediary variables known as latent or instrumental variables are produced $z = z(\theta, \upsilon)$ and may or may not have physical meaning. Due to the complex structure of the data-generating process, the likelihood function is typically intractable, meaning it cannot be evaluated. However, given a parameter θ it is straightforward to sample from the model by first sampling $\upsilon \sim p(\upsilon)$ and then running the simulator forward; the data-generating process is illustrated in Figure 2.1. This set up leads to what is known as simulation-based or likelihood-free inference. Given a prior density $p(\theta)$, the restriction of $p(\theta)p(\upsilon)$ to the manifold*

$$\mathcal{M} = \left\{(\theta, \upsilon) \in \Theta \times \Upsilon : f(\theta, \upsilon) = y^*\right\}.$$

defines a density $p(\theta, \upsilon \mid y^*)$ such that the marginal $p(\theta \mid y^*)$ coincides with the posterior obtained with a tractable likelihood. ABC [127] is a relaxation of exact likelihood-free inference, and corresponds to sampling from the following approximate posterior

$$\eta_\epsilon(\theta, \upsilon \mid y^*) \propto p(\theta)p(\upsilon)k_\epsilon(y^* - f(\theta, \upsilon)),$$

where k_ϵ is a smoothing kernel [127] that concentrates $p(\theta)p(\upsilon)$ around $\mathcal{M}$.

Example 2.1.2.2 (Bayesian Inverse Problems)**.** *Bayesian inference in complex observational models is fundamental in many areas of science. The data is the output of a deterministic forward function $F : \Theta \to \mathcal{Y}$ perturbed by some noise, typically zero-centered Gaussian with scale $\sigma > 0$*

$$y = F(\theta) + \upsilon \qquad \text{where} \qquad \upsilon \sim \mathcal{N}(0, \sigma^2 \mathrm{I}),$$

as shown in Figure 2.2. Having observed data $y^* \in \mathcal{Y}$, and with a prior over the parameters $p(\theta)$, the true parameter posterior distribution $p_\sigma(\theta \mid y^*) \propto p(\theta)p_\sigma(y^* \mid \theta)$ is naturally concentrated

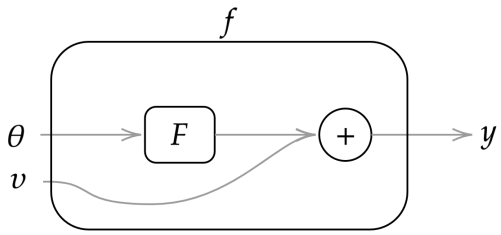

Figure 2.2: The data-generating process of a Bayesian inverse problem. Contrary to ABC, no latent variables are generated during the process, but observational noise is added to the output of F.

around the manifold

$$\mathcal{M}_0 = \{\theta \in \Theta : F(\theta) = y^*\}$$

for small values of $\sigma > 0$*, and it is an example of a filamentary distribution. Considering also the noise variables* $v \in \mathcal{Y}$ *then the true augmented posterior distribution* $p(\theta, v \mid y^*)$ *is naturally defined on the manifold*

$$\mathcal{M}_\sigma = \{(\theta, v) \in \Theta \times \mathcal{Y} : F(\theta) + v = y^*\},$$

independently of the value of σ.

Example 2.1.2.3 (Mixture of Gaussians around Ellipse)**.** *Consider a mixture of two Gaussians* $\pi(x)$ *that concentrates gradually around an ellipse, that is the* c*-level set of* $f(x) = x^\top Ax$ *for a positive semi-definite matrix* A*. This is shown in Figure 2.3. As the mixture concentrates more*

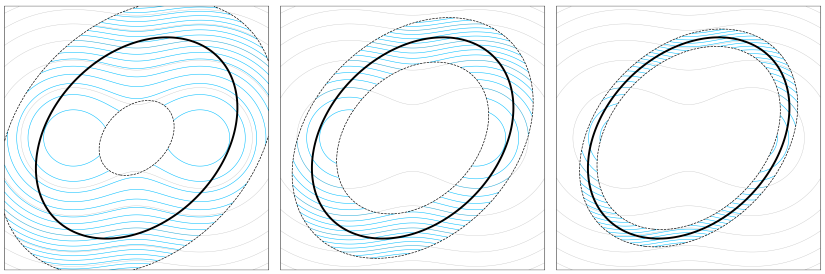

Figure 2.3: Grey: contours of the unconstrained mixture. Black: ellipse. Light-blue: contours of the filamentary density.

tightly around the ellipse, sampling becomes harder for general-purpose samplers. Figure 2.4 shows 500 *samples of RWM (red) and THUG (green), our proposed algorithm. [15] investigated the optimal scaling of RWM on ridged densities and showed that a vanishing acceptance probability is to be preferred, leading to larger jumps that are increasingly rarer. We run RWM for three different step sizes, achieving three different acceptance probabilities. Due to the anisotropy of the*

problem RWM struggles to sample the target appropriately for all step sizes. In contrast, THUG can use a much larger step size and achieves better performance.

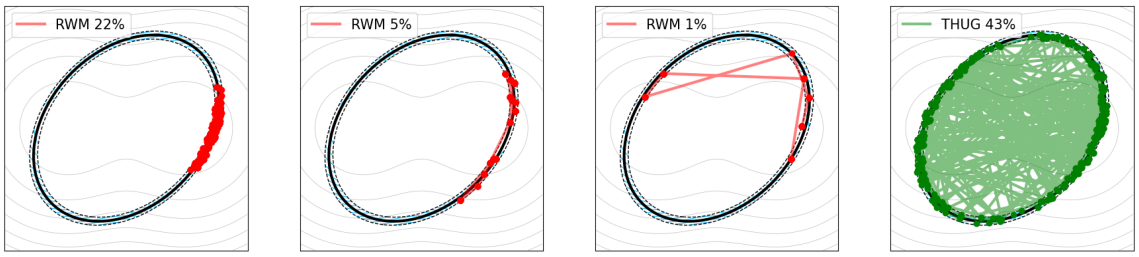

Figure 2.4: 500 samples of RWM (red) for different step sizes, versus 500 samples of THUG. Step sizes for RWM $\delta \in \{0.2, 0.7, 4.0\}$ are chosen to give different acceptance probabilities over $10k$ samples. Bigger step sizes lead to larger, but rarer moves [15]. Step size for THUG was 1.0 and was chosen to give about 40% acceptance probability.

2.2 Background on Exact Manifold Sampling

2.2.1 Geometry Fundamentals

The pre-image of any regular value $c \in \mathbb{R}^m$ of a smooth function $f : \mathbb{R}^n \to \mathbb{R}^m$ with $n > m$ is a smooth manifold of dimension $n - m$

$$\mathscr{M} := \{x \in \mathbb{R}^n : f(x) = c\}.$$

Notice that it is always possible to rewrite the function so that $c = 0$, hence we will assume that throughout. At any point $x \in \mathscr{M}$ the $m \times n$ Jacobian matrix of f, denoted J_x, defines two orthogonal subspaces

$$\begin{aligned} \mathscr{N}_x &= \{\mathrm{J}_x^\top \lambda : \lambda \in \mathbb{R}^m\} \\ \mathscr{T}_x &= \{v \in \mathbb{R}^n : \mathrm{J}_x v = 0\}, \end{aligned}$$

i.e. the normal and the tangent space, which have dimensions m and $n - m$ respectively, and decompose the ambient space as $\mathbb{R}^n = \mathscr{N}_x \oplus \mathscr{T}_x$. If J_x is full row-rank $\mathrm{J}_x \mathrm{J}_x^\top$ is invertible, and the following two projection matrices are well-defined

$$\begin{aligned} \mathrm{N}_x &= \mathrm{J}_x^\top (\mathrm{J}_x \mathrm{J}_x^\top)^{-1} \mathrm{J}_x \\ \mathrm{T}_x &= \mathrm{I}_n - \mathrm{N}_x \end{aligned} \tag{2.1}$$

and linearly project any vector $v \in \mathbb{R}^n$ onto $\mathscr{N}_x$ and $\mathscr{T}_x$, which we denote $v^\perp$ and $v^\parallel$ respectively.

2.2.2 Exact Manifold Sampling

Let π be a distribution on $\mathbb{R}^n$ with density $\pi(x)$ with respect to the Lebesgue measure. Suppose that due to the nature of the problem at hand, one wishes to sample from π restricted onto $\mathscr{M}$, then the Co-Area [40, 50, 81] formula tells us that this is a new distribution η with density

$$\eta(x) \propto \pi(x) |\det \mathrm{J}_x \mathrm{J}_x^\top|^{-1/2}, \tag{2.2}$$

with respect to the Hausdorff measure $\mathscr{H}^{n-m}$ on $\mathscr{M}$. In the MD literature η is known as the conditional probability measure on $\mathscr{M}$, in contrast to the restriction of π onto $\mathscr{M}$. We shall refer to η as either a manifold or constrained measure, and π as an unconstrained measure. MH algorithms to sample from constrained measures require more care than traditional algorithms since ensuring that the constraint and detailed-balance are satisfied is not as straightforward. At their core, they have a proposal and an acceptance step just as standard algorithms do, but also have an additional step to check the overall reversibility. For ease of exposition in the main body of this thesis we will focus on a constrained version of the RWM algorithm, Constrained RWM (C-RWM), algorithm as first suggested by [153], however HMC versions have also been developed and the algorithm is carefully detailed in [8] and in Appendix B.2.

2.2.3 Constrained Random Walk Metropolis

2.2.3.1 Proposal Step

The current state of the Markov chain $x \in \mathscr{M}$ is perturbed with a standard multivariate normal sample on $\mathscr{T}_x$ and then projected onto $\mathscr{M}$. More precisely, we generate a direction $v \in \mathscr{T}_x$ by linearly projecting $v \sim \mathcal{N}(0, \mathrm{I}_n)$ onto the tangent space $v = \mathrm{T}_x v$, and then we move to $y = x + v$ (see Appendix C.5 for a discussion on different tangent sampling methods). Typically y will lie outside of $\mathscr{M}$ so a non-linear projection will be necessary, i.e. find $\lambda \in \mathbb{R}^m$ such that $x' = y + \mathrm{J}_x^\top \lambda$ lies on $\mathscr{M}$. This is a non-linear root-finding problem and, for example, can be solved with Newton or symmetric-Newton solvers [8, 153]. If the optimizer fails to converge, we automatically reject. The proposal step is illustrated on the left panel of Figure 2.5.

2.2.3.2 Reversibility Check

The non-linear projection step to bring y onto $\mathscr{M}$ could have multiple solutions based on different initializations of λ, however not all of these solutions would necessarily satisfy detailed balance. In fact, reversibility is satisfied only if, when running the proposal step backwards from x', we would end up at x, within some tolerance $\rho > 0$ in practice. Whether or not this is the case for the solution found by the optimizer is unknown a-priori, hence a check is needed. To do so, we first need to find the total displacement from x to x' and then project it onto $\mathscr{T}_{x'}$. The total displacement is given by the initial velocity plus the projection vector found by the optimizer, i.e. $x' - x = v + \mathrm{J}_x^\top \lambda$. Projecting this onto $\mathscr{T}_{x'}$ simply amounts to multiplying by $T_{x'}$, hence $v' = \mathrm{T}_{x'}(v + \mathrm{J}_x^\top \lambda)$. Next, we

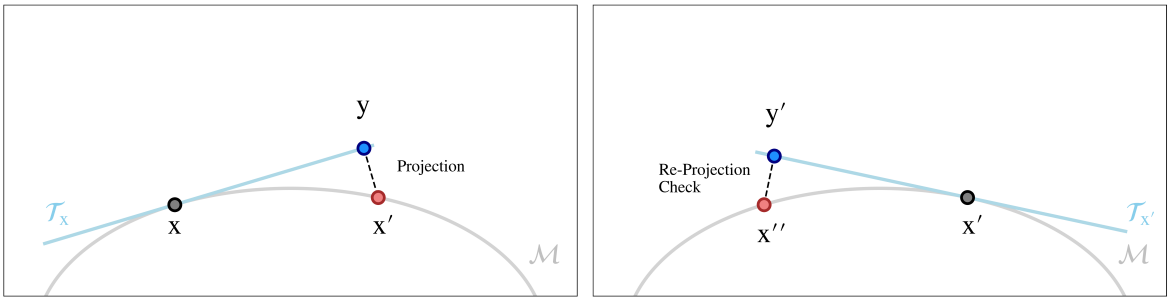

Figure 2.5: Left: The proposal mechanism of constrained samplers involves sampling on the tangent space and projecting onto $\mathcal{M}$. Right: Reversibility check of constrained samplers requires a second projection step. The proposal x' is only accepted if both projection steps are successful, as well as the Metropolis accept-reject step.

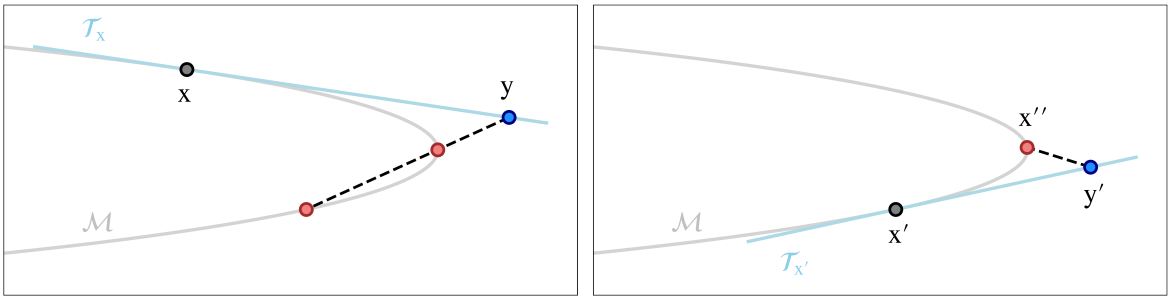

Figure 2.6: Left: The projection step in C-RWM could have multiple solutions. Right: Choosing the wrong solution may lead to a non-reversible sampling mechanism, thus it is necessary to check reversibility manually. (Note: axes have been rescaled unevenly to display the projected solutions clearly)

project $y' = x' + v'$ onto $\mathcal{M}$, i.e. we aim to find $\lambda' \in \mathbb{R}^m$ such that $x'' = y' + \mathrm{J}_{x'}^\top \lambda'$ lies on $\mathcal{M}$. If either the optimizer fails to converge or $\|x - x''\| > \rho$, we reject immediately. The need for a reversibility check is illustrated in Figure 2.6.

2.2.3.3 Acceptance Step

This is a straightforward MH accept-reject step: we accept x' with probability $a(x,x')$ otherwise stay at x,

$$a(x,x') = \min\left\{1, \exp\left(\log\eta(x') - \log\eta(x) - \frac{1}{2}\|v'\|^2 + \frac{1}{2}\|v\|^2\right)\right\}.$$

One step of C-RWM is given in Algorithm 3.

Algorithm 3: Constrained Random Walk Metropolis

Proposal Step:

1 *Tangent move*: $y = x + v$ where $v = \mathrm{T}_x \nu$ with $\nu \sim \mathcal{N}(0, \mathrm{I}_n)$.

2 *Projection*: set $x' = y + \mathrm{J}_x^\top \lambda$ where λ such that $f(x') = 0$. If it fails - reject.

Reversibility Check:

3 *Tangent Move*: $y' = x' + v'$ where $v' = \mathrm{T}_{x'}(v + \mathrm{J}_x^\top \lambda)$

4 *Projection*: $x'' = y' + \mathrm{J}_{x'}^\top \lambda'$ where λ' such that $f(x'') = 0$. If it fails - reject.

5 *Check*: If $\|x - x''\| > \rho$ reject x', otherwise continue to MH step.

Acceptance Step:

6 *Metropolis-Hastings*: Accept x' with probability $\alpha(x, x')$ otherwise stay at x.

Typically one switches the order of the reversibility and acceptance step to avoid performing a second non-linear projection when the sample will be rejected anyway. Nonetheless, the main cost of the algorithm are the two calls to the optimization routines used for projection and re-projection. The aim of this thesis is to by-pass these costly operations by accepting a small bias.

2.3 Approximate Manifold Sampling

2.3.1 Filamentary Distributions

Sampling from η is costly because the constraint has to be enforced twice using optimization routines. With the aim of avoiding those expensive operations, we relax the problem by instead sampling from a distribution η_ϵ defined on the ambient space $\mathbb{R}^n$ that interpolates between the unconstrained measure π and the constrained target η. The new target distribution η_ϵ depends on a bandwidth parameter $\epsilon > 0$ that determines the degree of closeness between η_ϵ and η. Loosely, we require

$$\begin{aligned} \eta_\epsilon &\longrightarrow \pi \qquad \text{as} \qquad \epsilon \to \infty \\ \eta_\epsilon &\longrightarrow \eta \qquad \text{as} \qquad \epsilon \to 0, \end{aligned}$$

in some sense. For our purposes, we require that expectations of suitably well-behaved test functions φ with respect to η_ϵ converge to expectations with respect to η, as $\epsilon \to 0^+$

$$\lim_{\epsilon \to 0^+} \int_{\mathbb{R}^n} \varphi(x) \eta_\epsilon(x) dx = \int_{f^{-1}(\{y^*\})} \varphi(x) \eta(x) \mathcal{H}^{n-m}(dx).$$

This is made more precise and proved under mild conditions in Appendix C. Sampling from η_ϵ for positive but small values of ϵ will lead to a small bias, but we note that in practice, particularly in Likelihood-free Inference and Bayesian Inverse Problems, practitioners are often willing to trade-off computational efficiency for a reasonable bias.

There are many ways of constructing η_ϵ but in practice these distributions naturally arise in a common form: their density with respect to the Lebesgue measure is the product between $\pi(x)$

and a smoothing kernel k_ϵ (more precisely, an approximation to the identity, see Definition C.1)

$$\eta_\epsilon(x) \propto \pi(x) k_\epsilon(y^* - f(x)), \tag{2.3}$$

such as Uniform, Gaussian or Epanechnikov [128]. Let us stress an important fact. Whereas $\eta(x)$ is a density with respect to the Hausdorff measure $\mathscr{H}$ on $\mathscr{M}$, both $\pi(x)$ and $\eta_\epsilon(x)$ are densities with respect to the Lebesgue measure on $\mathbb{R}^n$. Fundamentally, we are concentrating $\pi(x)$ around $\mathscr{M}$ using the kernel. Nonetheless, a careful reader might wonder whether expectations with respect to a distribution with density of the form (2.3) does indeed converge to expectations with respect to η on $\mathscr{M}$. It turns out that this the case, and it is shown in Theorem C.3.

The distribution η_ϵ is the prototypical example of a filamentary distribution for $\epsilon > 0$ small, and sampling from it using general-purpose sampling algorithms such as RWM and HMC is still very challenging. In this chapter, we develop a family of sampling algorithms tailored to sampling efficiently from η_ϵ while being computationally cheaper than constrained samplers when interest is in sampling from η.

2.3.2 An efficient integrator of constrained dynamics

This section describes geometrical tools that are crucial for the understanding of our sampler, and details why they are useful for sampling from a filamentary distribution η_ϵ. Similar to C-RWM and Constrained HMC (C-HMC) algorithms, THUG uses an auxiliary velocity variable $v \in \mathbb{R}^n$ to move across the state space, thanks to a properly defined mapping. More precisely, in THUG, the position and the velocity are updated using an efficient mechanism which maintains the position close to $\mathscr{M}$. The THUG bounce plays a central role.

Similar to the movement of a billiard ball hitting the cushion of a pool table, we call a bounce the composition of three simple operations: a straight line movement to a new position called the bounce point, a reflection of the direction of motion, and finally another straight line movement in this new direction. It is straightforward to confirm our physical intuition that this operation is time-reversible and volume-preserving.

Theorem 2.1 (Bounce Properties)**.** *Let* R *be any orthogonal matrix, and* $\delta > 0$ *be a step size. Define the bounce operation with step size* δ *and orthogonal matrix* R *as* $\mathbb{B}_{\mathrm{R},\delta}(x,v) = (x + (\delta/2)v + (\delta/2)\mathrm{R}v, \mathrm{R}v)$*. Then for any* R *and* δ*, the bounce is*

- *Time-reversible, i.e.* $\phi \circ \mathbb{B}_{\mathrm{R},\delta}$ *is an involution where* $\phi(x,v) = (x,-v)$*.*
- *Volume-preserving, i.e.* $|\det(\mathrm{J}_{\mathbb{B},\delta})| = 1$ *where* $\mathrm{J}_{\mathbb{B},\delta}$ *is the Jacobian matrix of* $\mathbb{B}_{\mathrm{R},\delta}$*.*

These two properties are satisfied by any bounce mechanism, but our algorithm uses a particular choice for R that under reasonable assumptions guarantees that its repeated application stays close to $\mathscr{M}$. The reflection matrix for the THUG bounce is

$$\mathrm{R} = \mathrm{I}_n - 2\mathrm{N}_{x+(\delta/2)v}, \tag{2.4}$$

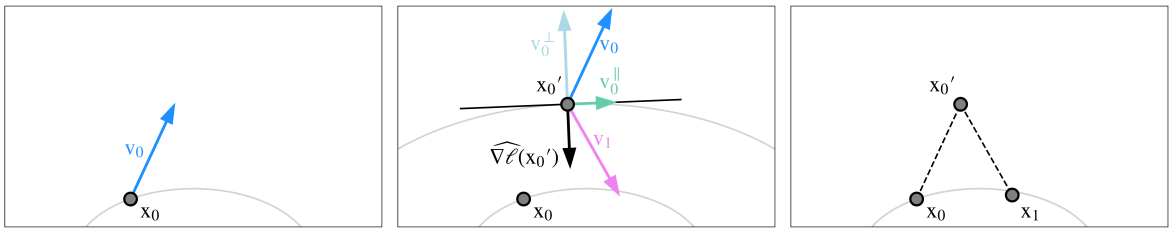

Figure 2.7: Bounce mechanism of the Hug kernel with $\delta = 2$ (so that half steps give $\delta/2 = 1$) on a bivariate Gaussian target with log-density $\ell(x)$. Left: A random direction $v_0 \sim \mathcal{N}(0, \mathrm{I}_n)$ is sampled. Middle: Velocity v_0 can be split into two components, one perpendicular and one parallel to the hyperplane defined by the gradient, but only the perpendicular one is flipped. Right: Take half a step in the new direction.

which fundamentally reflects the velocity off $\mathcal{T}_{x+(\delta/2)v}$, the tangent plane at the bounce point $x + (\delta/2)v$. For the rest of this paper, we shall write $\mathbb{B}_{\mathrm{THUG},\delta}$ for the THUG bounce. Intuitively, this makes sense: moving along the normal plane would lead to the largest change in f, so by bouncing off the tangent plane we are going in the opposite direction and approximately trying to minimize the change in f, thus remaining close to $\mathcal{M}$. The theorem below makes this more precise, and its proof, adapted from [94], is given in Appendix C.7.

Theorem 2.2 (Bounce Precision)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a smooth function, and for each $x \in \mathbb{R}^n$ let J_x denote its $m \times n$ Jacobian matrix, $\mathrm{H}[x]$ its Hessian order-three tensor, and $\mathrm{N}_x = \mathrm{J}_x^\top (\mathrm{J}_x \mathrm{J}_x^\top)^{-1} \mathrm{J}_x$ be the projection matrix defined in Subsection 2.2.1. Assume that the Hessian is bounded above by $\beta \in (0, \infty)$ and is γ-Lipschitz. Then the THUG bounce $\mathbb{B}_{THUG,\delta}$ repeated $B \in \mathbb{Z}_+$ times starting at $x_0, v_0 \in \mathbb{R}^n$ satisfies*

$$\|f(x_B) - f(x_0)\| \leq \frac{\delta^2 \|v_0\|^2}{8} (2\beta + \gamma \|T v_0\|)$$

where $T = B\delta$ is the total integration time.

A particular example of the THUG bounce mechanism has appeared before in the context of the Bouncy Particle Sampler [22, 126] and has found application in the Hug kernel of [94].

Example 2.3.2.1 (Hug Kernel)**.** *When $f(x) = \log \pi(x)$, the THUG bounce in Equation (2.4) reduces to the bounce mechanism of the Hug kernel with reflection matrix*

$$\mathrm{R} = \mathrm{I}_n - 2 \frac{\nabla \log \pi(x) \nabla \log \pi(x)^\top}{\|\nabla \log \pi(x)\|^2}.$$

The authors used this particular bounce to remain close to a given contour of their target distribution - we extend the use of this idea to arbitrary manifolds. Its application is illustrated in Figure 2.7.

As shown in the Appendix, it turns out that the THUG bounce can be interpreted as an explicit second order integrator for the dynamic of a particle moving with constant speed and centripetal acceleration on $\mathscr{M}$

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -\mathrm{J}_x^\top (\mathrm{J}_x \mathrm{J}_x^\top)^{-1} \mathrm{H}(v,v)[x], \end{aligned} \tag{2.5}$$

with the important distinction that the THUG discretization does not enforce the constraints. Remarkably, although the dynamic requires the velocity to lie on the tangent space at all times, the properties of Theorems 2.1 and 2.2 are satisfied regardless. The elastic collision of the THUG bounce works well when the curvature of the level set at the bounce point is similar to the curvature of $\mathscr{M}$ in nearby regions (see Appendix C.3 to see how the THUG bounce approximates curvature information). As a consequence, we expect larger movements orthogonal to $\mathscr{M}$, leading to farther level sets, to make the THUG bounce less effective. This intuition turns out to have a mathematical interpretation. Given $\alpha \in [0,1)$ we define the squeezing matrix $\mathrm{T}_{\alpha,x}$ and the squeezing operator $\mathbb{T}_\alpha$ as

$$\mathrm{T}_{x,\alpha} = \mathrm{I}_n - \alpha \mathrm{N}_x \qquad \text{and} \qquad \mathbb{T}_\alpha(x,v) = (x, \mathrm{T}_{\alpha,x} v),$$

where $\mathrm{T}_{\alpha,x}$ is a dampened version of the true projection matrix T_x that projects a vector onto $\mathscr{T}_x$. The next theorem tells us that squeezing the velocity before performing the THUG bounce allows us to improve the constant in Theorem 2.2.

Theorem 2.3 (Squeezed Bounce Precision)**.** *Suppose that in Theorem 2.2 we squeeze the velocity by a factor of $\alpha \in [0,1)$ before applying the THUG bounce $B \in \mathbb{Z}_+$ times, i.e. $(x'_B, v'_B) = \mathbb{B}^B_{THUG,\delta} \circ \mathbb{T}_\alpha(x_0, v_0)$, and denote by $\mathscr{B}_0$ the bound on the RHS of the inequality. Then*

$$\|f(x'_B) - f(x_0)\| \leq \mathscr{B}_0 - \frac{\alpha(2-\alpha)\delta^2 \|v_0^\perp\|^2}{8}(2\beta + \gamma \|T v_0\|) =: \mathscr{B}_\alpha,$$

and $\mathscr{B}_\alpha < \mathscr{B}_0$ whenever $\|v_0^\perp\| > 0$.

2.3.3 Tangential Hug

The main idea of THUG is to use $\mathbb{B}_{\mathrm{THUG},\delta} \circ \mathbb{T}_\alpha$ as a mapping to update (x,v) where $v \sim \mathscr{N}(0, \mathrm{I}_n)$. Figure 2.8 illustrates the link between the sampling mechanisms of C-RWM and THUG: (a) we replace a tangent velocity with a velocity drawn from a normal distribution with small variance in the direction orthogonal to the tangent plane, and (b) replace the costly non-linear projection with a computationally cheaper bounce. The result of Theorem 2.3 suggests that taking α close to one is optimal, following closely the C-RWM, but, as we shall see in Section 2.4.1, the choice of α is not straightforward when this mapping is embedded within a MH sampler. Further we note that in practice we have found, mostly empirically, that squeezing the velocity brings substantial improvements compared to what the theorem suggests, in particular when η_ϵ is very concentrated

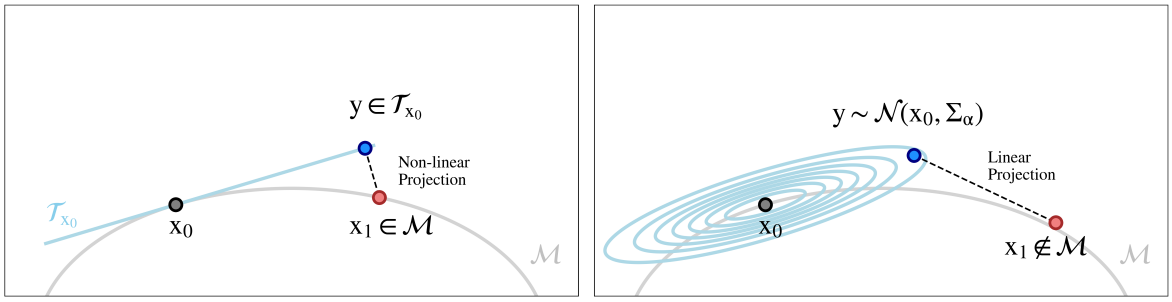

Figure 2.8: Left: dynamics of constrained samplers requires sampling on the tangent space and doing an expensive non-linear projection with an optimization routine. Right: our proposed algorithm samples from a multivariate normal aligned with the tangent space and uses a cheap linear projection that brings us close to $\mathcal{M}$.

on $\mathcal{M}$. Additional discussion as to why this might be the case is given in the Appendix, whereas experimental proof of this fact is given in Section 2.4.

The mapping $\mathbb{B}_{\text{THUG},\delta} \circ \mathbb{T}_\alpha$ possesses interesting properties, but is not time-reversible, therefore making its direct use within a valid MH algorithm difficult. This issue can be addressed easily by unsqueezing the velocity at the end of the trajectory. For $\alpha \in [0,1)$ the unsqueezing matrix $\mathrm{T}_{x,\alpha}^{-1}$ and the unsqueezing operator $\mathbb{T}_\alpha^{-1}$ are defined as

$$\mathrm{T}_{x,\alpha}^{-1} = \mathrm{I}_n + \frac{\alpha}{1-\alpha}\mathrm{N}_x \qquad \text{and} \qquad \mathbb{T}_\alpha^{-1}(x,v) = (x, \mathrm{T}_{x,\alpha}^{-1} v).$$

Overall the full THUG proposal mapping is given by

$$\mathbb{T}_\alpha^{-1} \circ \mathbb{B}_{\text{THUG},\delta} \circ \mathbb{T}_\alpha,$$

and one iteration of THUG is summarized in Algorithm 4.

The LinearProjection(J_x, v) operation is a function that computes the projection $\mathrm{J}_x^\top(\mathrm{J}_x\mathrm{J}_x^\top)\mathrm{J}_x v$, and can be implemented in multiple ways, but we found that using the QR-decomposition struck a good balance between efficiency and stability. If $\mathrm{QR} = \mathrm{J}_x^\top$ is the full QR-decomposition of $\mathrm{J}_x^\top$ then the first m columns of Q, denoted $\tilde{\mathrm{Q}} = \mathrm{Q}_{:,1:m}$, form a basis for the normal space (see Appendix C.4), and thus the projection can be computed as

$$\text{LinearProjection}(\mathrm{J}_x, v) = \tilde{\mathrm{Q}}\tilde{\mathrm{Q}}^\top v.$$

2.3.4 No Free Lunch: the trade-off in acceptance probability

While the bounce is a unitary operation for the velocity component, composition with $\mathrm{T}_{x_0,\alpha}$ and $\mathrm{T}_{x_B,\alpha}^{-1}$ breaks this property and the term $\Delta = \|v_B\|^2 - \|v_0\|^2$ is not zero and now appears in the acceptance ratio, in contrast with the scenario where $\alpha = 0$. The following Theorem quantifies how the change in norm of the velocities depend on the parameters $\alpha \in [0,1)$ and $\delta > 0$.

Algorithm 4: THUG (One Iteration)

INPUT :

- $x_0 \in \mathbb{R}^n$ current state of the Markov chain.
- $\alpha \in [0,1)$ squeezing parameter.
- $B \in \mathbb{N}$ number of bounces.
- $\delta > 0$ step size per bounce (total integration time is $T = B\delta$).
- $\mathrm{J}_x : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ function returning Jacobian evaluated at the input.
- LinearProjection(J, v) function to project v onto the row space of J.
- $\ell(x) = \log \eta_\epsilon(x)$ target log-density function.

Sample auxiliary: $v_0 \sim \mathcal{N}(0, \mathrm{I}_n)$. Set $(x, v) = (x_0, v_0)$.
Squeeze: $v \longleftarrow v - \alpha$LinearProjection$(\mathrm{J}_x, v)$
for $b = 1, \ldots, B$ **do**
  Move: $x \longleftarrow x + (\delta/2)v$
  Bounce: $v \longleftarrow v - 2$LinearProjection(J_x, v)
  Move: $x \longleftarrow x + (\delta/2)v$
end
Unsqueeze: $v \longleftarrow v + (\alpha/(1-\alpha))$LinearProjection$(\mathrm{J}_x, v)$
Accept-Reject: With probability $a = \exp(\ell(x) - \ell(x_0) - \|v\|^2/2 + \|v_0\|^2/2)$ accept x, otherwise stay at x_0.

Theorem 2.4 (Change in Velocity Norm)**.** *Let v_B be the final velocity obtained after $B \in \mathbb{Z}_+$ THUG bounces of step size δ starting from x_0 with velocity v_0 which is squeezed to w_0 by a factor of $\alpha \in [0,1)$. The change in norm squared $\Delta_v = \|v_B\|^2 - \|v_0\|^2$ has order*

$$\Delta_v = \mathcal{O}\left(\delta \frac{\alpha(2-\alpha)}{(1-\alpha)^2}\right).$$

The theorem above tells us that, unfortunately, the change in Δ_v depends linearly on δ. However, we have found in practice that the improvement in precision in ℓ brought by large values of α more than compensates the decrease in acceptance probability due to Δ_v when the filamentary distribution is particularly tight. In other words, squeezing the velocity is important in hard problems, as we show in subsection 2.4.1 of the Numerical Experiments.

2.3.5 THUG on a sphere is exact

We briefly note that THUG performs exact manifold sampling when the manifold is a d-dimensional sphere. Indeed the update in that case would be

$$x_1 = x_0 + \delta \mathrm{T}_{1/2} v_0 \qquad \text{where} \quad \mathrm{T}_{1/2} = \mathrm{I}_d - \frac{x_{1/2} x_{1/2}^\top}{\|x_{1/2}\|^2} \qquad \text{and} \quad x_{1/2} = x_0 + \frac{\delta}{2} v_0,$$

and it is easy to show that $\|x_1\| = \|x_0\|$. Notice that a Stereographic Projection Sampler [152] update would perform

$$x_1 = \frac{x_0 + \delta \mathrm{T}_0 v_0}{\|x_0 + \delta \mathrm{T}_0 v_0\|} \qquad \text{where} \quad \mathrm{T}_0 = \mathrm{I}_0 - \frac{x_0 x_0^\top}{\|x_0\|^2}$$

2.4 Numerical Experiment

2.4.1 Empirical analysis of the trade-off in acceptance probability

Let $(x_0^{(i)}, v_0^{(i)})$ be the i^{th} and current state of the Markov Chain, and denote by $(x_B^{(i)}, v_B^{(i)})$ the proposal obtained using the THUG dynamic with B bounces. Then, the acceptance probability is

$$\begin{aligned} a_i = a(x_0^{(i)}, v_0^{(i)}) &= \min\left\{1, \exp\left(\log \eta_\epsilon(x_B^{(i)}) - \log \eta_\epsilon(x_0^{(i)}) + \|v_0^{(i)}\|^2/2 - \|v_B^{(i)}\|^2/2\right)\right\} \\ &= \min\{1, \exp(\Delta\ell_i + \Delta k_i)\}, \end{aligned}$$

where we define $\Delta\ell_i := \log \eta_\epsilon(x_B^{(i)}) - \log \eta_\epsilon(x_0^{(i)})$ as the change in "potential energy", and $\Delta k_i := \|v_0^{(i)}\|^2/2 - \|v_B^{(i)}\|^2/2$ as the change in "kinetic energy". This decomposition is useful because it makes the dependence of the acceptance probability on the change in target density and the change in velocity-norm induced by α more explicit. In turn, Theorems 2.3 and 2.4 quantify how these two quantities change as a function of α and δ yet, on their own, they are not enough for a clear picture. This section is concerned with the empirical exploration of $\Delta\ell$ and Δk , where we can truly see that the trade-off is beneficial even on toy problems.

Example 2.4.1.1 ($\Delta\ell$ and Δk in a 2D ellipse problem)**.** *In the spirit of Example 2.1.2.3, we consider a 2D ellipse obtained as the level set of $f(x) = x^\top \Sigma^{-1} x$ where*

$$\Sigma = diag(1, 0.1).$$

The covariance matrix above is chosen as to create two areas of large curvature at the vertices of the major axis. We place a uniform prior on a sufficiently large rectangle containing the ellipse and use a Gaussian kernel as the likelihood, to concentrate the posterior around the ellipse. The resulting filamentary distribution is proportional to the Gaussian kernel and has density

$$\eta_\epsilon(x) \propto \frac{1}{\epsilon} \exp\left(-\frac{\|f(x) - y\|^2}{2\epsilon^2}\right),$$

and we sample from it using THUG for a specific $\epsilon > 0$ and a fixed $\delta > 0$. We want to choose a value of ϵ that is challenging enough so that using $\alpha > 0$ brings an improvement over $\alpha = 0$. Given ϵ, we choose δ so that the acceptance probabilities for both α values are above 5% and the improvement

is noticeable. In order to choose ϵ, we compute the average ESJD with $N = 1000$ samples, over a grid of δ and α values

$$ESJD = \frac{1}{N}\sum_{i=1}^{N} a_i \|x_B^{(i)} - x_0^{(i)}\|^2,$$

see Figure 2.9, and we choose a value of ϵ where the ESJD improves with α. In this case, we choose $\epsilon = 0.001$ although all except $\epsilon = 0.1$ would have also been sufficiently discriminatory for α and hence suitable. We then choose $\delta = 0.1$ since it is in a region of high ESJD for $\epsilon = 0.001$, and since the acceptance probabilities for $\alpha = 0$ and $\alpha = 0.99$ are 30% and 52% respectively. During the

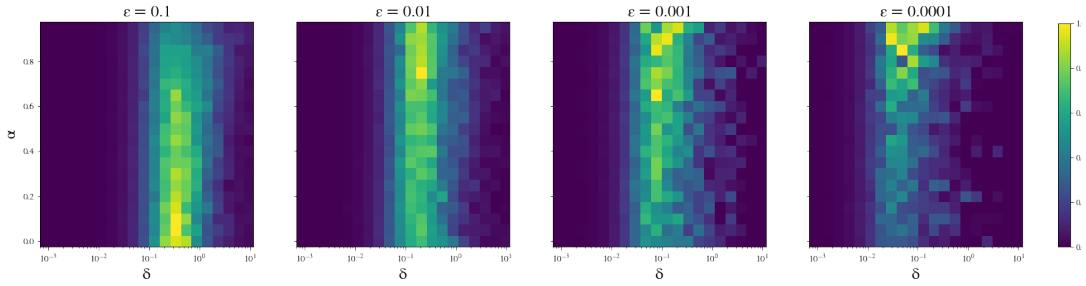

Figure 2.9: ESJD with $N = 1000$ computed over a grid of α,δ values, for different values of ϵ. For each plot, we map the ESJD independently into $[0,1]$ to use a common colorbar. For large values of ϵ, it is convenient to use $\alpha = 0.0$, but as ϵ increases, larger values of α are required. Based on these results, we choose $\epsilon = 0.001$ and $\delta = 0.1$.

sampling procedure we record the values of $\{\Delta\ell_i\}_{i=1}^N$ and $\{\Delta k_i\}_{i=1}^N$ and use them to generate three different tables. The first table aims to show that by increasing the value of α one is able to increase the proportion of samples with $\Delta\ell$ larger than a certain value, effectively shifting the distribution of $\Delta\ell$ over samples, and thus, possibly, enhancing the overall acceptance probability. With this in mind, we find values $L_p \in \mathbb{R}$ such that p% of samples obtained with $\alpha = 0$ have $\Delta\ell \geq L_p$, and we focus on $p \in \{25\%, 50\%, 75\%, 90\%\}$. For various values of α, we then compute the proportions p_α of samples with $\Delta\ell \geq L$. This is summarised in Table 2.1, where we can see that the proportion of samples p_α is larger than p_0 for all $\alpha > 0$ considered, hinting at a considerable improvement in acceptance probability due to a better ability to stay in high-density regions.

	$\Delta\ell \geq -0.18$	$\Delta\ell \geq -4.99$	$\Delta\ell \geq -44.97$	$\Delta\ell \geq -220.67$
$\alpha = 0$	25	50	75	90
$\alpha = 0.5$	35.24	64.01	86.82	95.97
$\alpha = 0.9$	47.97	74.06	89.23	96.04
$\alpha = 0.99$	47.04	72.45	88.12	96.07

Table 2.1: Proportions of samples with $\Delta\ell \geq L_p$ for different values of α, with $\epsilon = 0.001$ and $\delta = 0.1$.

The second table is very similar to the first one, but considers the change in kinetic energy instead. Since $\Delta k = 0$ whenever $\alpha = 0$, here we find the value K_p such that p% of samples obtained

with the largest α have $\Delta k \geq K_p$, and then use this to compute p_α for the other values of α. This is shown in Table 2.2.

	$\Delta k \geq 0.01$	$\Delta k \geq -0.32$	$\Delta k \geq -13.08$	$\Delta k \geq -610.08$
$\alpha = 0$	0	100	100	100
$\alpha = 0.5$	35.56	74.30	99.31	100.00
$\alpha = 0.9$	30.59	61.08	91.31	99.66
$\alpha = 0.99$	25.00	50.00	75.00	90.00

Table 2.2: Equivalent to Table 2.1 for Δk, however now the values K_p are obtained for $\alpha = 0.99$.

We can see the degrading effect of using $\alpha \neq 0$ on the kinetic energy, which will affect the acceptance probability. In order to see how the trade-off resolves, we therefore need Table 2.3, which reports the proportions of samples with $\Delta\ell + \Delta k \geq A_p$, where A_p is found based on $\alpha = 0$. This table does indeed show how the trade-off is advantageous for $\alpha \neq 0$ and indeed leads to more acceptances.

	$\Delta\ell + \Delta k \geq -0.18$	$\Delta\ell + \Delta k \geq -4.99$	$\Delta\ell + \Delta k \geq -44.97$	$\Delta\ell + \Delta k \geq -220.67$
$\alpha = 0$	25	50	75	90
$\alpha = 0.5$	33.44	63.53	86.81	95.97
$\alpha = 0.9$	40.25	70.11	87.86	95.73
$\alpha = 0.99$	37.69	63.90	78.66	86.86

Table 2.3: Equivalent to Tables 2.1 and 2.2 for $\Delta\ell + \Delta k$.

Example 2.4.1.2 ($\Delta\ell$ and Δk in a 10D ellipsoid problem)**.** *We now consider a* 10*-dimensional ellipsoid, again obtained as the level set of a* 10*-dimensional multivariate Gaussian with diagonal covariance matrix*

$$\Sigma = diag(0.1, 1, \ldots, 1).$$

We use a similar strategy to choose the ϵ and δ values to use, however in high-dimensions ESJD is biased towards settings that give low acceptance probability but large jumps, similarly to [15]. Consequently, by looking exclusively at the equivalent plots as those in Figure 2.9 one might choose a ϵ, δ pair with acceptance probabilities below 1%*. Instead, we also take into account the acceptance probability and only choose a pair that gives at least* 5% *acceptance probability for all values of α. This is because in practice, we have found that THUG performs best with larger acceptance probabilities. Tables 2.4, 2.5, and 2.6 show the results for $\epsilon = 0.001$ and $\delta = 0.1$, whereas Tables 2.7, 2.8, and 2.9 show the results for a smaller value of ϵ and δ. In both cases, we can see that the bulk of the change in acceptance ratio is driven by the change in potential energy, thus indicating that trade-off is advantageous for these values of ϵ.*

	$\Delta\ell \geq -3.51$	$\Delta\ell \geq -31.69$	$\Delta\ell \geq -160.98$	$\Delta\ell \geq -504.59$
$\alpha = 0$	25	50	75	90
$\alpha = 0.5$	30.46	59.06	83.07	94.30
$\alpha = 0.9$	33.93	63.63	85.19	94.82
$\alpha = 0.99$	36.44	64.09	84.40	94.20

Table 2.4: Proportions of samples with $\Delta\ell \geq L_p$ for increasing α, with $\epsilon = 0.001$ and $\delta = 0.1$.

	$\Delta k \geq 0.08$	$\Delta k \geq -0.29$	$\Delta k \geq -2.53$	$\Delta k \geq -11.95$
$\alpha = 0$	0	100	100	100
$\alpha = 0.5$	21.37	81.97	98.14	99.99
$\alpha = 0.9$	25.07	76.92	94.98	99.52
$\alpha = 0.99$	25.00	50.00	75.00	90.00

Table 2.5: Proportions of samples with $\Delta k \geq K_p$ for increasing α, with $\epsilon = 0.001$ and $\delta = 0.1$.

	$\Delta\ell + \Delta k \geq -3.51$	$\Delta\ell + \Delta k \geq -31.69$	$\Delta\ell + \Delta k \geq -160.98$	$\Delta\ell + \Delta k \geq -504.59$
$\alpha = 0$	25	50	75	90
$\alpha = 0.5$	30.38	59.02	83.06	94.29
$\alpha = 0.9$	33.82	63.58	85.16	94.81
$\alpha = 0.99$	35.46	63.65	84.05	94.02

Table 2.6: Proportions of samples with $\Delta\ell + \Delta k \geq A_p$ for increasing α, with $\epsilon = 0.001$ and $\delta = 0.1$.

	$\Delta\ell \geq -0.61$	$\Delta\ell \geq -6.27$	$\Delta\ell \geq -35.60$	$\Delta\ell \geq -147.64$
$\alpha = 0$	25	50	75	90
$\alpha = 0.5$	32.44	61.14	83.10	93.83
$\alpha = 0.9$	37.61	64.78	84.27	94.14
$\alpha = 0.99$	38.93	65.72	84.55	94.35

Table 2.7: Proportions of samples with $\Delta\ell \geq L_p$ for increasing α, with $\epsilon = 0.0001$ and $\delta = 0.03$.

	$\Delta k \geq 0.04$	$\Delta k \geq -0.00$	$\Delta k \geq -0.10$	$\Delta k \geq -0.48$
$\alpha = 0$	0	100	100	100
$\alpha = 0.5$	22.68	54.60	81.25	94.69
$\alpha = 0.9$	24.82	52.80	77.53	91.79
$\alpha = 0.99$	25.00	50.00	75.00	90.00

Table 2.8: Proportions of samples with $\Delta k \geq K_p$ for increasing α, with $\epsilon = 0.0001$ and $\delta = 0.03$.

	$\Delta\ell + \Delta k \geq -0.61$	$\Delta\ell + \Delta k \geq -6.27$	$\Delta\ell + \Delta k \geq -35.60$	$\Delta\ell + \Delta k \geq -147.64$
$\alpha = 0$	25	50	75	90
$\alpha = 0.5$	32.42	61.13	83.09	93.83
$\alpha = 0.9$	37.36	64.77	84.24	94.14
$\alpha = 0.99$	38.77	65.62	84.54	94.35

Table 2.9: Proportions of samples with $\Delta\ell + \Delta k \geq A_p$ for increasing α, with $\epsilon = 0.0001$ and $\delta = 0.03$.

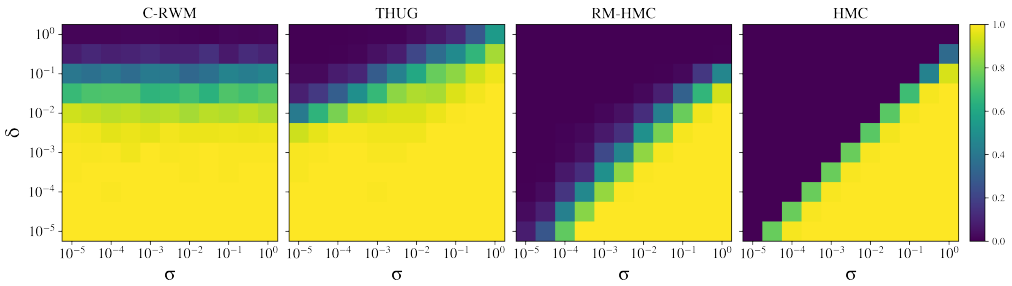

Figure 2.10: Average Acceptance Probability for varying values of observational noise $\sigma > 0$ and step size $\delta > 0$. THUG, HMC, and RM-HMC are used to sample from the true posterior $p_\sigma(\theta \mid y^*)$, whereas C-RWM is used to sample from $p_\sigma(\theta, v \mid y^*)$.

2.4.2 Bayesian Inverse Problem Example

Consider a BIP as laid out in Example 2.1.2.2, with a two-dimensional parameter $\theta = (\theta_0, \theta_1)$ having prior $p(\theta) = \mathcal{N}(\theta \mid 0, \mathrm{I}_2)$, forward function $F(\theta) = \theta_1^2 + 3\theta_0^2(\theta_0^2 - 1)$, and a normal prior over the noise $p(v) = \mathcal{N}(v \mid 0, \sigma^2)$. We observe data $y^* = 1.0$, and consider inference on the true parameter posterior distribution

$$p_\sigma(\theta \mid y^*) \propto p(\theta)\mathcal{N}(y^* \mid F(\theta), \sigma^2). \tag{2.6}$$

For small values of σ, this is a filamentary distribution and we sample from it using HMC, Riemann-Manifold HMC (RM-HMC) and THUG. To compare this with C-RWM we follow [8] and lift the distribution onto $\xi = (\theta, v)$ by defining the function $f_\sigma(\theta, v) = F(\theta) + v$, with Jacobian $\mathrm{J}_{\theta,v}$, and thus targeting the lifted posterior

$$p_\sigma(\theta, v \mid y^*) \propto p(\theta)p(v)\left|\det \mathrm{J}_{\theta,v}\mathrm{J}_{\theta,v}^\top\right|^{-1/2} \tag{2.7}$$

where k_ϵ is a Gaussian kernel. Clearly $p_\sigma(\theta \mid y^*)$ is the θ-marginal of the lifted distribution so one can sample from (2.7) and discard the v samples.

2.4.2.1 Acceptance Probability

We sample from the true posterior (2.6) using THUG, HMC and RM-HMC. Figure 2.10 shows how the acceptance probability changes as we decrease the noise scale around $\mathscr{M}_0$. Similarly to [8], the results are averaged across 10 runs of 50 samples each. All algorithms use $B = 20$ integration steps/bounces per iteration. The squeezing parameter for THUG is kept at $\alpha = 0.0$. For the smallest noise level, THUG is able to maintain a non-zero acceptance probability with a step size three orders of magnitude larger than HMC and RM-HMC. As one would expect, since C-RWM is used to sample on $\mathscr{M}_\sigma$, there is no dependence of the acceptance probability on the noise level.

2.4.2.2 Computational Cost

We study which algorithm is computationally more convenient for different values of $\sigma > 0$. We run 12 chains of $N = 2500$ samples for each algorithm using $B = 20$ integration steps, a fixed step size of $\delta = 0.1$, giving a total integration time of $\tau = B\delta$. In addition to C-RWM, THUG and HMC we also run an HMC sampler whose total integration time is fixed to τ but whose step size decreases proportionally to the noise scale σ. Figure 2.11 is a log-log plot showing the minimum (bulk) ESS across all components divided by total runtime for THUG, HMC and C-RWM. This metric is popular in studying the computational efficiency of different algorithms and has been used in the context of C-HMC [8] and the original Hug algorithm [94]. We found the minESS to be a more reliable measure of performance than the ESS computed over a chain of log-density values. Standard HMC quickly becomes inefficient as the noise scale decreases, and it is thus not an appropriate sampler for this type of problems. When its step size decreases in proportion to σ, we can see that the ESS stays at a relatively high level at a cost of an exponential increase in runtime. THUG (with $\alpha = 0.0$) is able to maintain an order of magnitude larger ESS-per-second for the first three noise scales. For smaller noise scales we notice a degradation in raw ESS, leading to less efficiency compared to C-RWM. On top of the natural degradation of the integrator, this is because this problem is very low-dimensional and therefore the projection operations in C-RWM are very cheap. As we will see in the G-and-K experiment, as dimensionality increases, these projection operations become extremely inefficient. Additional results for different values of τ are given in Appendix C.8.3.

2.4.2.3 Choosing the value of α

In the two experiments above we have kept $\alpha = 0.0$ and, in general, it is not clear how one should choose the value of this parameter. A natural approach is to embed the THUG kernel into an SMC sampler [38, 39] and adaptively choose the value of α. The strategy that we use here is as follows: at the i^{th}-round we adapt α with the aim of achieving a target acceptance probability a^*, by using the current estimated acceptance probability $\hat{a}_i$ across particles. The update is then done in the spirit of [5]

$$\tau_{i+1} = \tau_i - \gamma_{i+1}\left(\hat{a}_i - a^*\right),$$

where $\tau_i = \text{logit}(\alpha_i)$ and (γ_i) is a sequence of step sizes. In our case we have chosen $a^* = 0.3$ and, for stability purposes, we clip the value of α_i between $(\alpha_{\min}, \alpha_{\max}) = (0.01, 0.999)$. To illustrate how filamentary distributions can naturally be constructed, for this experiment we target the following approximate lifted distribution

$$p_{\epsilon,\sigma}(\theta, v \mid y^*) \propto p(\theta)p(v)k_\epsilon(y^* - f_\sigma(\theta, v)),$$

which converges to the true lifted distribution as $\epsilon \to 0$. All SMC samplers start with 5000 particles initialized from the prior, and use multinomial resampling at each step. The new

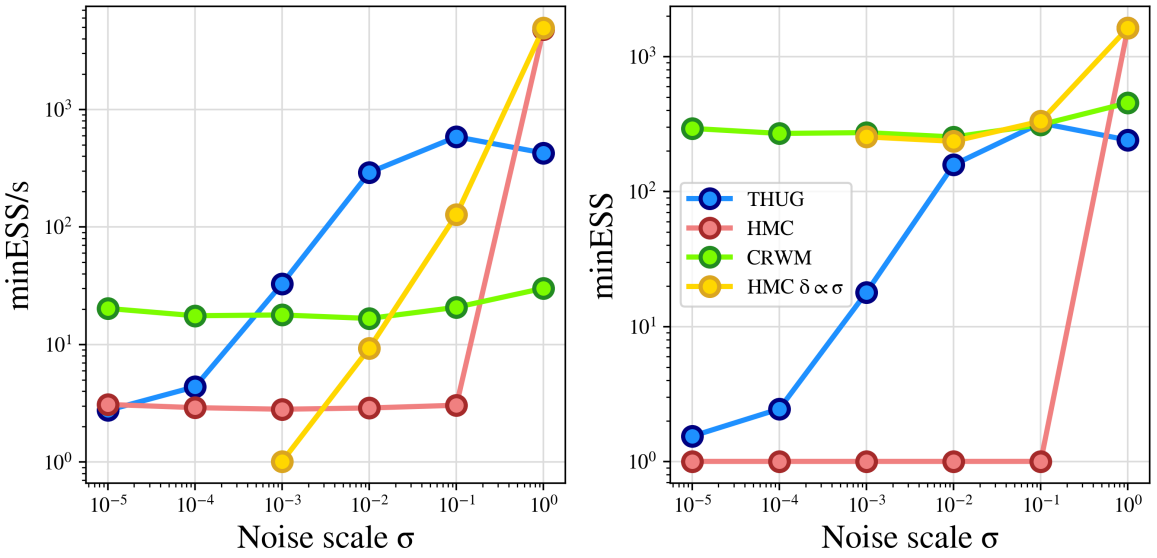

Figure 2.11: Left: minESS over total runtime (in seconds) for THUG, HMC and C-RWM. HMC $\delta \propto \sigma$ uses the same integration time as the other algorithms but decreases δ proportionally to σ. THUG, HMC and HMC $\delta \propto \sigma$ target the true posterior $p_\sigma(\theta \mid y^*)$ whereas C-RWM targets the lifted posterior $p(\theta, v \mid y^*)$. Right: raw ESS as a function of noise scale.

tolerance ϵ_i, at each step, is chosen according to the number of unique particles[1]. The SMC sampler stops either after reaching $\epsilon \leq 1 \times 10^{-10}$, after having performed 200 iterations, or if the acceptance probability drops below $a_{\text{ter}} = 0.01$. We consider three scenarios.

The first experiment consists in keeping the step size δ fixed and adapting α as detailed above. Figure 2.12 shows various metrics for three SMC samplers: one with a Random Walk kernel, one with a THUG kernel with $\alpha = 0.0$ fixed, and a second THUG kernel with adaptive α (denoted αTHUG). Each column displays the metrics for runs of varying step sizes $\delta \in \{0.01, 0.1, 0.5, 1.0\}$. Noticeably, both THUG and αTHUG strongly outperform SMC-RWM across all step sizes, and adapting α brings important improvements achieving up to three and seven orders of magnitude smaller ϵ than SMC-THUG and SMC-RWM respectively. For the smallest step size, SMC-THUG is effectively sampling from the lifted posterior since ϵ is in the order of machine precision.

In the second experiment we additionally adapt the step size δ based on the estimated acceptance probability as for α, above. Although one should justify the appropriateness of this joint optimization procedure mathematically, we shall leave this for future work and simply notice that it does seem to converge and allows to sample from extremely tight filamentary distributions, as shown in Figure 2.13.

Finally, in the last experiment of this subsection we combine RWM and αTHUG to form a black-box SMC sampler for increasingly concentrating posteriors that exploits the benefits of both

[1]One could either use the number of unique particles or the ESS. We track both and use the classical definition of ESS for sequential importance sampling algorithms.

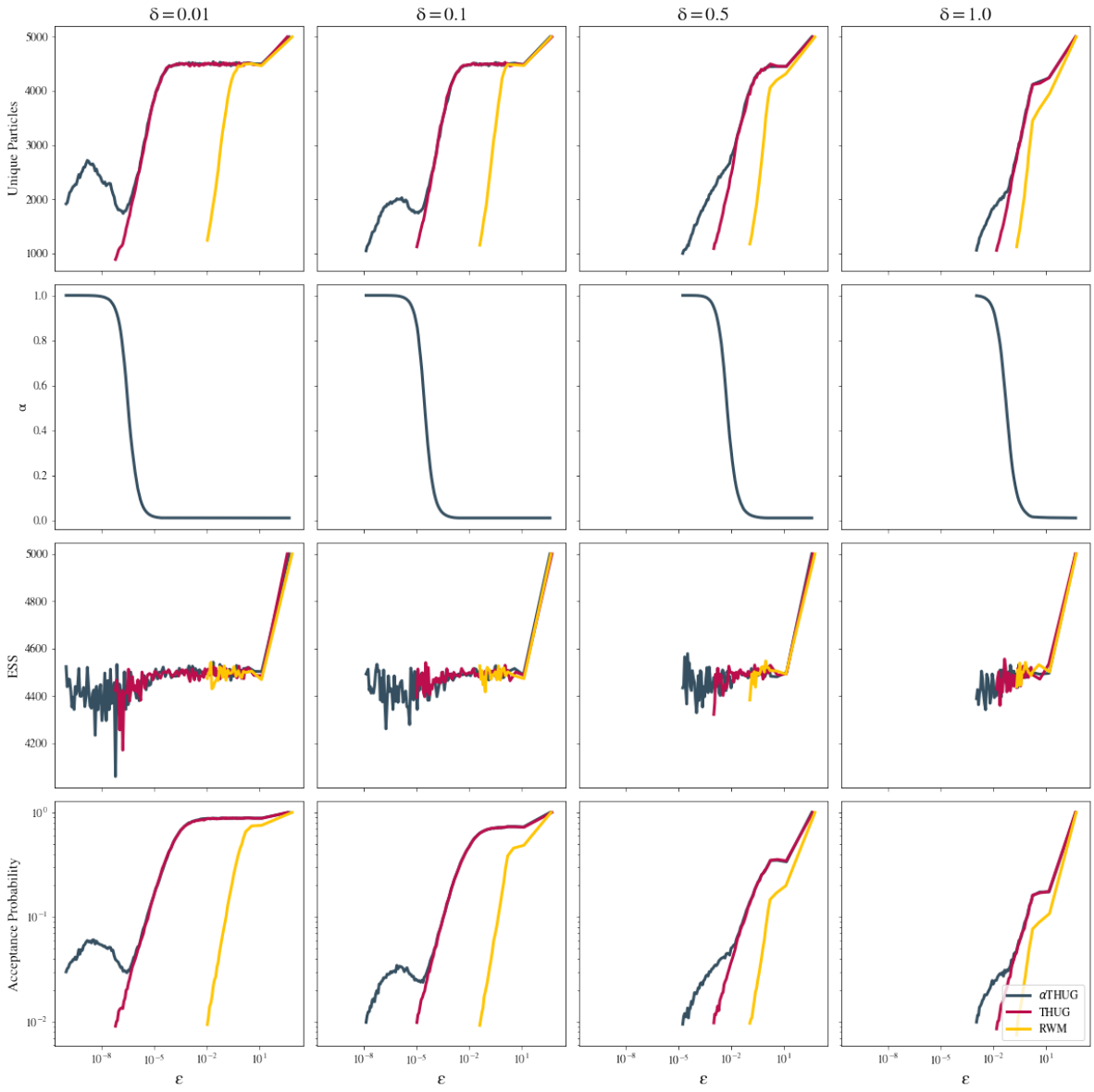

Figure 2.12: Comparison of SMC-RWM, SMC-THUG and SMC-αTHUG for different (fixed) step sizes. THUG/αTHUG are able to achieve much tighter approximate posterior distributions and provide improvements across all metrics.

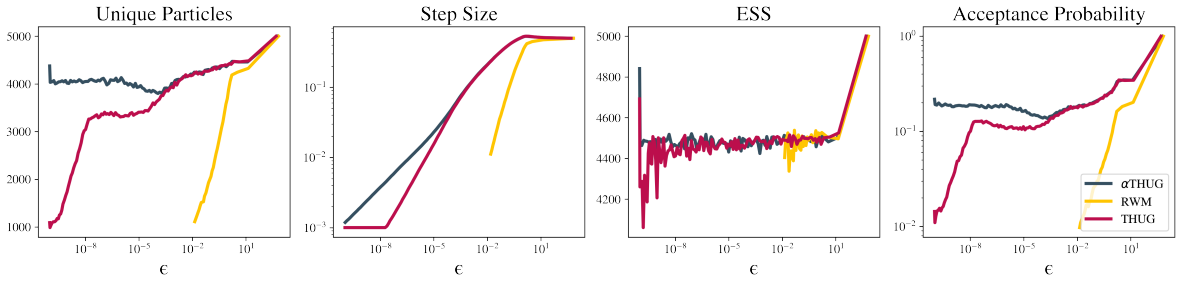

Figure 2.13: It is possible to adapt α and δ in tandem to achieve machine precision.

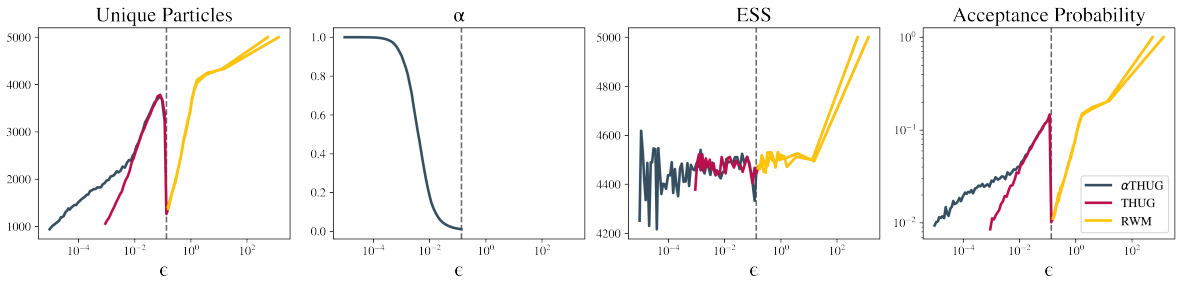

Figure 2.14: Black-box SMC sampler for increasingly concentrating posteriors starts with a RWM kernel and switches to a THUG kernel once the target is tight enough around the manifold. Dotted line represents when the switch between RWM and THUG/αTHUG happens.

kernels. Indeed, while Random Walk performs best in the early stages of the SMC sampler, i.e. when $p_{\epsilon,\sigma}$ is not a filamentary distribution, it performs poorly once ϵ is small enough. Conversely, THUG performs best when the target has a filamentary structure and poorly when $p_{\epsilon,\sigma}$ is not particularly tight around $\mathcal{M}_\sigma$. Therefore, we run an SMC sampler with a RWM kernel and once the acceptance probability gets close to a_{ter}, we switch to either THUG or αTHUG. More precisely, when $\hat{a}_i \leq \iota a_{\text{ter}}$, with $\iota \geq 1.0$, we change the mutation kernel to be THUG. The results are shown in Figure 2.14.

2.4.3 G-and-K Distribution

The task of inferring the parameters of the G-and-K quantile distribution is a standard benchmark problem in ABC. Suppose that some data $y^* \in \mathbb{R}^m$ has been observed and that it is known that the data-generating process consists of a differentiable function parametrized by $\theta = (a, b, g, k)$ that takes as input a set of normally distributed latent variables $z \sim \mathcal{N}(0, \mathrm{I}_m)$. This transformation is shown below and it is the quantile function of the G-and-K distribution, which is often used to model skewed or heavy-tailed data [112]

$$y = a + b\left(1 + 0.8\frac{1-e^{-gz}}{1+e^{-gz}}\right)(1+z^2)^k z. \tag{2.8}$$

Simulating data is straightforward: given θ one samples $z \sim p(z)$ and feeds it through the transformation above. On the other hand, evaluating the likelihood function exactly is not possible. However, [117] and [13] noticed that it is still possible to evaluate it with high precision by numerically inverting the g-and-k quantile function, albeit at a high computational cost. [112] has implemented a RWM sampler using this technique, which we use to obtain the ground truth against which we compare our numerical results.

In our experiments, we choose a product of four independent Uniform$(0, 10)$ distributions as the prior distribution for θ, however we reparametrize the problem in terms of an unconstrained parameter $\vartheta \sim \mathcal{N}(0, \mathrm{I}_4)$ (see Appendix E.1 for additional details). We generate the observed data using $\theta_0 = (3, 1, 2, 0.5)$ as the true parameter value, and choose a Gaussian kernel. The augmented ABC posterior distribution over $\xi = (\theta, z)$ is then a filamentary distribution for small values of the tolerance ϵ.

Figure 2.15 shows the minimum (bulk) ESS for THUG, αTHUG and C-RWM computed across four chains of 1000 samples each, for different values of $B \in \{1, 10, 50\}$, and for different dimensions of the data/latent variables $m \in \{50, 100, 200\}$. We study the behavior of THUG for three values of $\alpha \in \{0.0, 0.9, 0.99\}$ for tolerances $\epsilon \in \{10^0, \ldots, 10^{-8}\}$ almost up to machine precision. Light grey, grey, and dark grey colors represent different number of bounces/leapfrog steps per sample i.e. $B = 1, 10, 50$ respectively. There are four different markers (cross, circle, star, and triangle) representing the four algorithms examined in this experiment, i.e. C-RWM, THUG, αTHUG with $\alpha = 0.9, 0.99$ respectively. Since C-RWM samples on the manifold directly, its computational cost is independent of the value of ϵ, although it is important to note that due to numerical error the sampler is actually sampling from a filamentary distribution whose tightness around $\mathcal{M}$ depends on the tolerance for the non-linear projection, which we set to 10^{-12}, and assume negligible. The left-most figure shows the results for $m = 50$ and we can see that, while THUG is generally more efficient than C-RWM, the margin is small. However, as the dimensionality increases the cost of projecting onto the manifold becomes too prohibitive and C-RWM is only able to keep a positive acceptance probability with $B = 1$ steps, for the fixed step size of $\delta = 0.01$. THUG is able to achieve an order of magnitude better minESS/runtime while using a step size up to 50 times larger. Let us analyse the results in the figure above in regards to THUG. For a fixed value of m, we can see that larger values of B lead to smaller minESS/time values, while at the same time allowing to reach smaller tolerance values ϵ. Additionally, the larger α, the smaller ϵ we can reach. As a result, one can sample from filamentary distributions close to machine precision with the correct choice of B and α. The fact that more bounces lead to a decrease in minESS/time may seem counter-intuitive, but we conjecture this is due to the particular structure of the problem whereby the manifold is four-dimensional independently of the value of m, and thus proportionally minuscule compared to the ambient space, especially for large m. For a fixed value of B, a larger α often leads to a larger minESS/time, thus allowing more efficient sampling. Again, we can see the surprising effectiveness of the squeezing parameter α, allowing THUG to achieve extremely small tolerance

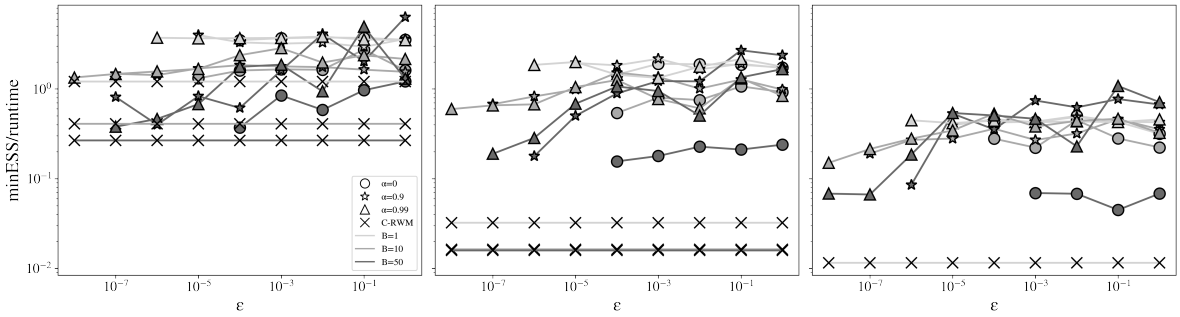

Figure 2.15: C-RWM vs αTHUG/THUG on the G-and-K example for $m = 50$ (left), $m = 100$ (center), and $m = 200$ (right). The cost of C-RWM is constant across different ϵs because it samples on the manifold directly.

values.

Figure 2.16 shows kernel density estimates of 10 000 samples for THUG, αTHUG (with $\alpha = 0.99$), RWM, and C-RWM on the G-and-K problem with $m = 50$. We run all algorithms (except C-RWM) for a sequence of decreasing tolerances $\epsilon \in \{10, 5, 1, 0.1, 0.01, 0.001, 0.0001\}$ keeping their step-size fixed, and only display the KDE of the samples obtained for the lowest possible ϵ with positive acceptance probability (for C-RWM, we simply use the samples obtained by sampling from the posterior on the manifold). For a fair comparison, we set the total integration time of THUG/αTHUG, and C-RWM to be the same ($T = 0.2$) as well as the number of bounces and leapfrog steps ($B = L = 5$). Furthermore, we set the RWM step-size to T/B (not to T). Noticeably, the posterior distribution sampled by THUG/αTHUG is comparable to the one sampled from C-RWM but comes at a much cheaper cost, especially for high dimensional problems, as shown in the right-most plot in Figure 2.15.

2.4.4 Lotka-Volterra

The predator-prey Lotka-Volterra (LV) model is another common benchmark in the ABC literature [62, 99] and appears in various formulations. The one we adopt here is as an Euler-Maruyama discretization of the system of Stochastic Differential Equations (SDEs)

$$\begin{aligned} dr &= (z_1 r - z_2 r f)dt + dn_r, \\ df &= (z_4 r f - z_3 f)dr + dn_f, \end{aligned} \tag{2.9}$$

where r and f represent the prey and predator populations respectively, n_r and n_f are zero mean unit-variance white noise processes, and $z = (z_1, z_2, z_3, z_4)$ are parameters governing the evolution of the system. In accordance with previous literature [62], we choose the true parameter values that lead to stable simulated trajectories with high probability, i.e. we set $z = (0.4, 0.005, 0.05, 0.001)$. We perform $N_s = 100$ discretization steps of Equations (2.9)

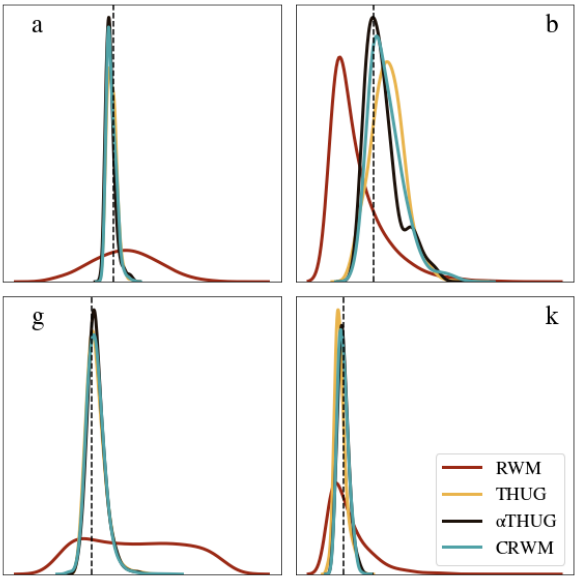

Figure 2.16: G-and-K example: kernel density estimates of 10 000 samples obtained via THUG, αTHUG ($\alpha = 0.99$), RWM, and C-RWM. Dotted lines are the components of θ_0, the true parameter value.

with initial conditions $r_0 = f_0 = 100$ and step size $\tau = 1.0$, and the generated data is $2N_s$-dimensional and consists of the prey and predator population counts at the discretization times $y^* = [r_1, f_1, \ldots, r_{N_s}, f_{N_s}]$. To enforce positivity of the parameters z, we assign a product of independent and identically distributed log-normal priors to each component

$$p(z_i) = \text{LogNormal}(z_i \mid -2, 1),$$

and then reparametrize the system to have standard normal priors. Overall, the constraint function for an input $\xi \in \mathbb{R}^{4+2N_s}$ is given in Algorithm 5. The manifold and filamentary posterior distributions targeted in this experiments are therefore

$$p_\epsilon(\xi \mid y^*) \propto \exp\left(-\frac{1}{2}\xi^\top \xi\right) k_\epsilon(y^* - F(\xi))$$
$$p(\xi \mid y^*) \propto \exp\left(-\frac{1}{2}\xi^\top \xi\right) \left|\mathrm{J}_\xi \mathrm{J}_\xi^\top\right|^{-1/2},$$

where k_ϵ is chosen as a Gaussian kernel and J is the Jacobian matrix of the constraint function F defined by Algorithm 5.

Analogously to the previous section, Figure 2.17 shows the minimum ESS across components normalised by total running time for THUG, αTHUG, and C-RWM, for a fixed value of $\delta = 0.01$ and a sequence of tolerances $\epsilon \in \{10^0, \ldots, 10^{-6}\}$. As a term of comparison, RWM achieved around 0.1% acceptance probability for $\epsilon = 10$, and did not manage to get a single acceptance in multiple runs of 10 000 samples with $\epsilon = 1.0$. The conclusions we can take are similar to the ones for

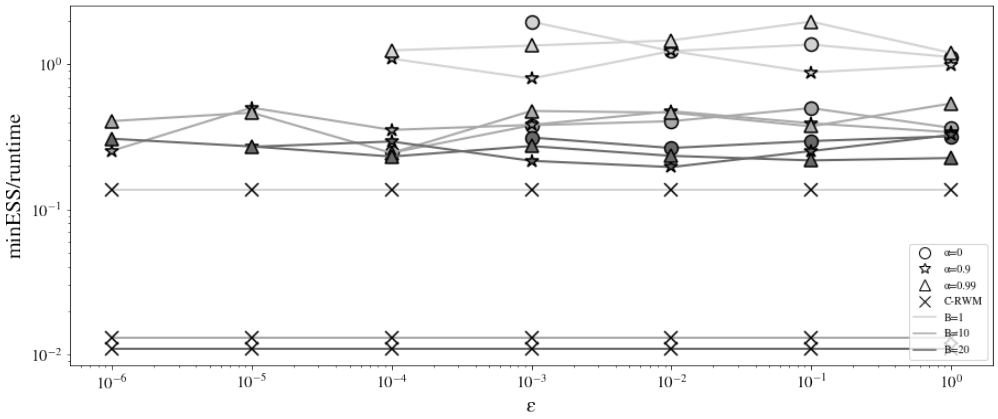

Figure 2.17: minESS/runtime for THUG (circle), αTHUG ($\alpha = 0.9$ star, $\alpha = 0.99$ triangle), and C-RWM (cross) for the LV model with $N_s = 100$.

the G-and-K example: larger values of α and of B allow THUG to sample from extremely tight filamentary distributions, while keeping a cheaper computational cost with respect to C-RWM.

Algorithm 5: Lotka-Volterra Constraint Function

$z \longleftarrow \exp(\xi_{1:4} - 2)$

$u \longleftarrow \xi_{5:2N_s+4}$.

for $s = 1, \ldots, N_s$ **do**

  $r_s \longleftarrow r_{s-1} + \tau(z_1 r_{s-1} - z_2 r_{s-1} f_{s-1}) + \sqrt{\tau}\sigma_r u_{2s-1}$

  $f_s \longleftarrow f_{s-1} + \tau(z_4 r_{s-1} f_{s-1} - z_3 f_{s-1}) + \sqrt{\tau}\sigma_f u_{2s}$

end

OUTPUT: $[r_1, f_1, \ldots, r_{N_s}, f_{N_s}] - y^*$

2.5 Related Work

2.5.1 Literature on Constrained Integration

2.5.1.1 Direct Integration

[107] is the first work that we are aware of where a set of equations of motions with holonomic constraints was integrated, albeit using standard numerical integrators. As noted a few years later by one of the authors [123], using general-purpose integrators lead to the constraints only being satisfied at initial times, and to the numerical trajectory to deviate from the true one for later times. To avoid this, [123] developed the SHAKE algorithm, which is a Verlet integrator [145] with additional steps to enforce the constraints at each step. In the same year [141] proposed to replace the Verlet integrator with a predictor-corrector one, since its explicit use of the velocity variables seemed to improve the overall precision of the integrator and widen

its applicability. A few years later, [2] developed the well-known RATTLE integrator which uses the velocity counterpart of the Verlet integrator. Although RATTLE and SHAKE are equivalent on the position steps and half-time velocity steps [83], only RATTLE is symplectic thanks to its ability to enforce the velocity constraint. Convergence results for the SHAKE algorithm were first derived by [10]. Higher order symplectic integrators for Hamiltonian systems based on Partitioned Runge-Kutta integrators have also been studied in two simultaneous works [118] and [75]. An alternative projection method was suggested in [65] and it was shown to better preserve the geometric properties of the flow of the ODE system. In particular their symmetric projection method shares some similarities with our bounce mechanism as it can use the midpoint gradient to perform the projection. Further advances on these topics can be found in [82] and [86].

2.5.1.2 Indirect Integration

Equations of motion with holonomic constraints form a Differential Algebraic Equations (DAEs) system of index three, which, through differentiation of the constraints, can be reduced to an unconstrained *underlying ODE*. There has been work integrating the underlying ODE or its state-space formulation [6, 7, 26, 54], but typically indirect integration is preferred due to its better stability properties for longer integration times [84]. Alternatively, [36] developed third-order numerical integrators whose steps automatically satisfy the constraints. As shown in the Appendix, THUG can be thought of as an integrator for the underlying ODE.

2.5.2 Literature on Constrained Sampling

We summarise the literature on constrained sampling in two broad categories based on whether or not the suggested algorithms use a MH acceptance step. Metropolis-adjusted algorithms with reversibility checks tend to be preferred in the statistics community since they sample from the target and are unbiased. Metropolis-unadjusted samplers based on the discretization of SDEs constrained on manifolds have a long history in the MD community since the step size $\delta > 0$ is assumed to be sufficiently small to induce a negligible bias, since reversibility is approximately preserved [66, 82].

2.5.2.1 Metropolis-Adjusted

[40] is perhaps the most well-known work in the statistics literature that considers the task of sampling a measure supported on a manifold and uses a MH correction step. [85] developed a sampler based on an overdamped Langevin dynamics with projection steps to satisfy the constraints, and for other SDE-based samplers with MH correction steps we refer the reader to the references therein. [153] was the first to notice that a reversibility check was needed when using constrained samplers in order to satisfy detailed balance, and proposed an unbiased version of the constrained random walk algorithm. [87] generalized this work to an unbiased

constrained Hamiltonian Monte Carlo sampler, and later generalized the reversibility check when multiple solutions to the non-linear projection can be computed [88]. [25] generalized C-HMC and developed a family of constrained sampling algorithms that use a guidance Hamiltonian to propose a new state of the Markov chain, and an acceptance Hamiltonian to perform the accept-reject step, essentially decoupling the proposal and acceptance mechanisms. As discussed in the Appendix, THUG also benefits from this decoupling by using a constrained Hamiltonian for its proposal step and accepting based on the filamentary distribution. Finally, let us remark that RM-HMC[59] was developed to exploit the local geometric structure of space when it exhibits strong correlations while sampling in the ambient space, but [8] showed that, although for a specific choice of metric RM-HMC and C-HMC can be considered equivalent in BIP, its implicit numerical integration scheme leads to much greater instability than using the RATTLE integrator.

2.5.2.2 Unadjusted Samplers

One class of unadjusted samplers consists in the discretization of SDEs constrained onto a submanifold. [33] discretized a diffusion having a distribution of the form (2.2) as its invariant distribution and applied this to free energy computations in MD, although the correctness of this approach was only proved rigorously in [34]. We refer to [143] for a second-order integration of constrained Langevin dynamics and [86] (and references therein) for a collection of related results. The second class of unadjusted samplers consists in the discretization of SDE with an additional soft-constraint keeping the dynamic in a neighbourhood of $\mathscr{M}$. The notion of a soft constraint has received some attention in the MD literature and we refer to [86] for an introduction. Typically the soft constraint is imposed with a strong drift term [49, 55, 76] corresponding to a Gaussian smoothing kernel and therefore unadjusted integration of such diffusions corresponds to sampling from a filamentary distribution. [34] determine an SDE with soft constraint that, as the tolerance goes to zero, converges to an SDE ergodic with respect to η, see Appendix C and Remark 2.3 in their paper for a discussion on rigidly and softly constrained dynamic. More recently, this line of work was generalised by [155] to a family of reversible diffusions ergodic with respect to different manifold and filamentary distributions, among which η and η_ϵ, and developed appropriate integrators. In particular, they develop a multi-scale numerical method to integrate softly-constrained SDEs, and provide a very insightful discussion on Metropolis-adjusted methods such as [87] and [85], and Metropolis-unadjusted (see Remark 8 in their paper). Based on a conjecture in the previous paper, [125] extended this work by integrating non-reversible diffusions and showed their improved asymptotic variance.

2.5.3 Literature on Filamentary Distributions

Although the work discussed in 2.5.2.2 comprised samplers targeting filamentary distributions, here we summarise the literature which has approximate manifold sampling as its explicit aim. [15] studied the asymptotic behavior of the RWM algorithm on ridged densities, that is

densities that are highly concentrated around a submanifold, in the limit of vanishing noise. They show that for submanifolds of unit codimension the usual strategy of adapting the step-size to control the acceptance probability is sub-optimal and instead a better strategy is to take larger steps and let the acceptance probability approach zero. However, when the submanifold is of much smaller dimensionality than the ambient space, then the optimal strategy is to use a position-dependent proposal and propose moves with smaller scaling in the directions orthogonal to the submanifold. [92] developed "The Barker Proposal": a novel gradient-based method that has good high-dimensional scaling properties but is more robust to tuning and has better ergodicity than MALA/HMC algorithms. [96] show that choosing the Markov kernel in a locally-informed way is preferable to a random scan approach for sparse and filamentary distributions which concentrates around submanifolds or low-dimensional structures. Finally, [122] developed a robust RWM sampler using approximate Hessian information and study its behavior for concentrating posteriors around a linear manifold.

2.5.4 Literature on applications of exact manifold sampling

[63] was the first work we are aware of, applying C-HMC to ABC. [8] followed this line of work but instead applied these techniques to BIP, whereas [64] worked on diffusion models.

2.5.5 Literature on Likelihood-free Inference

The origins of likelihood-free inference can be traced back to the work of [41] and [121], although it was the need for efficient and accurate inference in population genetics that spurred a number of algorithms such as Rejection-ABC [53, 115, 135, 147], its variants [12], and MCMC-ABC [97] which will become the workhorses of likelihood-free inference. Importance-based alternatives soon became popular such as Population Monte Carlo ABC [11, 129, 130]. The two main drawbacks where the prohibitive cost of the algorithm and a strong weight degeneracy when successive distributions are too different from each other. To solve the latter issue [111] incorporated a Partial Rejection Control mechanism [91] in the forward kernel. The second problem was addressed in [39] by using an MCMC kernel which lowered the computational complexity to be linear in the number of particles, although leading to higher-variance weights, and proposed an adaptive strategy for the tolerance schedule based on the ESS. [13] introduced an alternative strategy where the tolerance is chosen based on the number of unique particles. More recently, [48] proposed to use Delayed-Acceptance into the MCMC kernel of the SMC-ABC scheme with tolerance schedule similar to the latter work. [37] introduced a novel way to recycle the intermediary states explored by each particle and showed that it can bring improvements whenever the performance of the algorithm decreases with iterations, such as in rare event problems.

2.5.6 Literature on leveraging latent structure in ABC

[23] advocates for leveraging the additional information that is often possible to extract from a simulator, and the first hint of this idea can be found in [3], which has since spurred a wide array of papers in this topic. Coupled ABC [103] identifies the simulator as a deterministic function of the parameters and the latent variables. Hamiltonian ABC [98] builds upon advances in Stochastic Gradient Hamiltonian Dynamic algorithms [30, 43, 148] but the gradients are estimated using [133], whose variance is reduced by fixing the random seeds and using a Gaussian synthetic likelihood [149]. Optimization Monte Carlo [99] advances a population of weighted particles by finding the optimal parameter value for a given latent variable, sampled from the prior, that reconstructs the data up to a tolerance. It presents two main drawbacks. The first one is that it can lead to latent variables for which there is no such optimal parameter [63, 74], and the second one is that the weights can be unstable in regions where the constraint function is flat. [74] proposed Robust Optimization Monte Carlo to tackle these issues, and related methods were also introduced in the econometrics literature [51, 52]. In Automatic Variational ABC the authors lower bound the marginal likelihood using the ELBO inequality, as done in Variational Inference [19], replace the true likelihood with the ABC likelihood and leverage the change of variables formula and the reparametrization trick [78] to propagate gradients through an expectation with respect to the distribution of the latent variables, in other words they perform Stochastic Variational Inference in an ABC framework. [114] take advantage of rare-event techniques [28] to design a Pseudo-Marginal MH algorithm such that at each iteration, the ABC likelihood is estimated using a rare event SMC method with a Pseudo-Marginal Slice Sampling [101] forward kernel. Distilled Importance Sampling [113] aims to learn a normalizing flow approximating the ABC joint posterior over the parameters and the latent variables in a sequential fashion.

CHAPTER 3

MARKOV AND INTEGRATOR SNIPPETS

3.1 Setup, Notation and Contributions

Similarly to Hamiltonian Monte Carlo (HMC), we introduce an auxiliary measurable space $(\mathsf{V}, \mathscr{V})$ with distribution $\varpi(dv)$ and instead consider expectations with respect to the product probability measure $\mu = \pi \otimes \varpi$ on the product space $(\mathsf{Z}, \mathscr{Z}) = (\mathsf{X} \times \mathsf{V}, \mathscr{X} \otimes \mathscr{V})$ with reference measure v. We shall refer to μ as our target distribution. We sometimes abuse notation and write h for both a function on $(\mathsf{X}, \mathscr{X})$ or $(\mathsf{Z}, \mathscr{Z})$, which will be clear from context. Similarly to Sequential Monte Carlo (SMC), we construct a sequence of target distributions $\{\mu_n = \pi_n \otimes \varpi_n\}$ for $n \in [\![0, P]\!]$, with $\mu_P = \mu$, and μ_0 being easy to sample from. Let $\{\psi_n : \mathsf{Z} \to \mathsf{Z} : n \in [\![0, P]\!]\}$ be a collection of invertible functions, $N \in \mathbb{Z}_+$ be the number of particles, and $T \in \mathbb{Z}_+$ be the integration time. Before introducing the interacting particle system, we develop a new importance-based update.

3.1.1 Contributions to Markov and Integrator Snippets

Markov Snippets for sampling from a general target distribution were introduced in [154]. In this PhD thesis, we offer several general clarifications of the theory by phrasing them in terms of a sequential version of Pushforward Importance Sampling (PISA). We also develop Markov Snippets for a mixture of Markov kernels and consequently Integrator Snippets with a mixture of integrators. We provide several new metrics to evaluate the performance of these algorithms, including the Expected Squared Jump Distance. We suggest strategies to use these metrics to adapt the step size and the number of integration steps. Additionally, we study their behavior on filamentary distributions and develop GHUMS, as well as analyse several of the PISA algorithm in toy examples to understand its strengths and weaknesses.

3.2 Introduction

3.2.1 Understanding pushforward distributions

Suppose we have a measure μ on $(\mathsf{Z}, \mathscr{Z})$ and a measurable function ψ mapping Z to a separate space Y, for which a valid sigma algebra $\mathscr{Y}$ is available. A new measure μ^{ψ} on $(\mathsf{Y}, \mathscr{Y})$ can be defined by assigning to each set $\mathsf{B} \in \mathscr{Y}$ the same size that μ would assign to the preimage $\psi^{-1}(\mathsf{B})$.

Definition 3.1 (Pushforward Measure)**.** Let $(\mathsf{Z}, \mathscr{Z})$ and $(\mathsf{Y}, \mathscr{Y})$ be measurable spaces, $\psi : \mathsf{Z} \to \mathsf{Y}$ be $\mathscr{Z}/\mathscr{Y}$-measurable, and μ be a measure on $(\mathsf{Z}, \mathscr{Z})$. The function $\mu^{\psi} : \mathscr{Y} \to [0, +\infty]$ defined as

$$\mu^{\psi}(\mathsf{B}) := \mu(\psi^{-1}(\mathsf{B})) = \mu\left(\{z \in \mathsf{Z} : \psi(z) \in \mathsf{B}\}\right) \qquad \forall \mathsf{B} \in \mathscr{Y}$$

is also a measure and we refer to it as pushforward or image measure of μ by ψ.

The interpretation is that if $Z \sim \mu$ then $Y \sim \mu^{\psi}$ where $Y = \psi(Z)$. If ψ is an invertible function and $(\mathsf{Y}, \mathscr{Y}) = (\mathsf{Z}, \mathscr{Z})$ then the pushforward of μ by ψ^{-1} is defined analogously as the following probability distribution on $(\mathsf{Z}, \mathscr{Z})$

$$\mu^{\psi^{-1}}(\mathsf{A}) := \mu(\psi(\mathsf{A})) = \mu\left(\{\psi(z) \, : \, z \in \mathsf{A}\}\right) \qquad \forall \mathsf{A} \in \mathscr{Z}.$$

Similarly, the interpretation is that if $Z \sim \mu$ then $\psi^{-1}(Z) \sim \mu^{\psi}$. Since ψ is invertible, $\mu = (\mu^{\psi})^{\psi^{-1}}$ and $(\mu^{\psi^{-1}})^{\psi}$ therefore the reverse implications also hold

$$Z \sim \mu^{\psi} \implies \psi^{-1}(Z) \sim \mu$$
$$Z \sim \mu^{\psi^{-1}} \implies \psi(Z) \sim \mu.$$

Suppose $(\mathsf{Z}, \mathscr{Z}) = (\mathbb{R}^d, \mathscr{B}(\mathbb{R}^d))$ and that $\mu(dz), \mu^{\psi}(dz)$ and $\mu^{\psi^{-1}}(dz)$ all admit densities with respect to $\mathrm{Leb}_{\mathsf{Z}}(dz)$, the Lebesgue measure on this space, and we denote the density of $\mu(dz)$ by $\mu(z)$. Using change of variables formulas for a diffeomorphism on a Euclidean space of Corollary A.1, we get the standard result

$$\mu^{\psi}(dz) = \mu(\psi^{-1}(z))|\det \mathrm{J}_{\psi^{-1}}(z)|\mathrm{Leb}_{\mathsf{Z}}(dz)$$
$$\mu^{\psi^{-1}}(dz) = \mu(\psi(z))|\det \mathrm{J}_{\psi}(z)|\mathrm{Leb}_{\mathsf{Z}}(dz),$$

where J_{ϕ} denotes the Jacobian matrix of a function ϕ, as usual. Notice that not only the pushforward distributions μ^{ψ} and $\mu^{\psi^{-1}}$ are probability measures whenever μ is one, but their densities share the same normalizing constants, meaning $\mathscr{Z}_{\mu} = \mathscr{Z}_{\mu^{\psi}} = \mathscr{Z}_{\mu^{\psi^{-1}}}$. In this work, we shall focus on the particular case when ψ is a volume-preserving transformation and therefore

$$|\det \mathrm{J}_{\psi}(z)| = \frac{1}{|\det \mathrm{J}_{\psi^{-1}}(z)|} = 1 \qquad \forall z \in \mathsf{Z},$$

so that we can safely write $\mu(\psi^{-1}(z))$ and $\mu(\psi(z))$ for the densities of the two measures above (notice that a good mnemonic trick is that the sign of the power of ψ in the density is always the opposite of the one in the measure). We highlight that our work continues to be valid even when ψ is not volume-preserving, with the caveat that one needs to compute the Jacobians.

3.2.2 Pushforward importance sampling

In this section, we build an intuition for the algorithms in the remaining of the chapter. Consider again $(\mathsf{Z}, \mathcal{Z})$ a measurable space, μ and η be two probability distributions on it, and $\psi : \mathsf{Z} \to \mathsf{Z}$ be an invertible and volume-preserving transformation. Similar to Section 1.7, we shall assume $\mu \ll \eta^{\psi}$ and that both distributions admit a density with respect to a common dominating measure on $(\mathsf{Z}, \mathcal{Z})$. Suppose interest is in estimating $\mu(h)$ for some test function $h : \mathsf{Z} \to \mathbb{R}$. The following remarkably simple property provides us with a new way of tackling this common task,

$$\begin{aligned}
\mu(h) &= \int h(z) \frac{\mu(dz)}{\eta^{\psi}(dz)} \eta^{\psi}(dz) \\
&= \int h(z) \frac{\mu(z)}{\eta(\psi^{-1}(z))} \eta^{\psi}(dz) \\
&\approx \frac{1}{N} \sum_{i=1}^{N} h(\psi(Z^{(i)})) \frac{\mu(\psi(Z^{(i)}))}{\eta(Z^{(i)})} \qquad\qquad Z^{(i)} \sim \eta.
\end{aligned}$$

where we have assumed we can evaluate densities with their normalizing constant for ease of exposition. Given a suitable volume-preserving and invertible ψ, ψ^k also satisfies these properties for $k \in \mathbb{Z}$ so it becomes natural to choose an integer $T \in \mathbb{Z}_+$ and using $\psi^0, \psi^1, \ldots, \psi^T$ exactly as above

$$\begin{aligned}
\mu(h) &= \int h(z) \frac{1}{T+1} \sum_{k=0}^{T} \frac{\mu(dz)}{\eta(\psi^{-k}(z))} \eta^{\psi^k}(dz) \\
&= \frac{1}{T+1} \sum_{k=0}^{T} \int h(z) \frac{\mu(z)}{\eta(\psi^{-k}(z))} \eta^{\psi^k}(dz)
\end{aligned}$$

Again, this requires $\mu \ll \eta^{\psi^k}$ for every $k \in [\![0, T]\!]$ and all distributions admitting densities with respect to a common reference measure. A naive way to estimate $\mu(h)$ would be to obtain $T+1$ independent sets of N samples from each $\eta^{\psi^k}(dz)$. For each k, this would require sampling N times from η and feeding these samples through ψ^k. The final estimator would be

$$\mu(h) \approx \frac{1}{N(T+1)} \sum_{k=0}^{T} \sum_{i=1}^{N} h(Z^{(i),k}) \frac{\mu(\psi^k(Z^{(i),k}))}{\eta(Z^{(i),k})} \qquad Z^{(i),k} \sim \eta \text{ for } i \in [\![N]\!], k \in [\![0, T]\!].$$

Alternatively, and this is the approach we consider here, one could sample a single set of N particles from η and use it to generate the sets of samples for *all* pushforward distributions

$$\mu(h) \approx \frac{1}{N(T+1)} \sum_{k=0}^{T} \sum_{i=1}^{N} h(\psi^k(Z^{(i)})) \frac{\mu(\psi^k(Z^{(i)}))}{\eta(Z^{(i)})} \qquad Z^{(i)} \sim \eta \text{ for } i \in [\![N]\!].$$

This is of course cheaper by a factor of $T+1$, and in addition our hope is that sharing the initial randomness may provide us with a variance reduction mechanism. This second approach can also be interpreted as constructing $T+1$ empirical distributions, using the same initial samples

$$\hat{\mu}^{N,k}(dz) = \frac{1}{N} \sum_{i=1}^{N} \frac{\mu(\psi^k(Z^{(i)}))}{\eta(Z^{(i)})} \delta_{\psi^k(Z^{(i)})}(dz) \qquad k \in [\![0, T]\!],$$

and use them as components of a (uniform) mixture

$$\hat{\hat{\mu}}^{N,T}(dz) = \frac{1}{T+1}\sum_{k=0}^{T}\hat{\mu}^{N,k}(dz) = \frac{1}{N(T+1)}\sum_{k=0}^{T}\sum_{i=1}^{N}\frac{\mu(\psi^k(Z^{(i)}))}{\eta(Z^{(i)})}\delta_{\psi^k(Z^{(i)})}(dz).$$

3.3 Integrator Snippets

3.3.1 Folded Integrator Snippets - SMC samplers for a mixture

Consider an SMC sampler that rather than targeting $\{\mu_n\}$, targets the sequence of distributions

$$\bar{\mu}_n(dz) = \frac{1}{T+1}\sum_{k\in[\![0,T]\!]}{\mu_n}^{\psi_n^{-k}}(dz) \qquad n \in [\![0,P]\!], \tag{3.1}$$

using the forward kernels

$$\bar{M}_n(z_{n-1}, dz_n) = \sum_{k\in[\![0,T]\!]} W_{n-1,k}(z_{n-1}) R_n(\psi_{n-1}^k(z_{n-1}), dz_n) \tag{3.2}$$

with

$$\begin{aligned} \mu_{n-1}R_n &= \mu_{n-1} \\ W_{n,k}(z_n) &= \frac{{\mu_n}^{\psi_n^{-k}}(dz_n)}{\sum_{\ell\in[\![0,T]\!]}{\mu_n}^{\psi_n^{-\ell}}(dz_n)}, \end{aligned} \tag{3.3}$$

and the near-optimal backward kernels (see section 1.8.6.2)

$$\bar{L}_{n-1}(z_n, dz_{n-1}) = \frac{\bar{\mu}_{n-1}(dz_{n-1})\bar{M}_n(z_{n-1}, dz_n)}{\bar{\mu}_{n-1}\bar{M}_n(dz_n)}. \tag{3.4}$$

This leads to the following (unnormalized) incremental weights

$$\bar{w}_n(z_n) = \frac{\bar{\mu}_n(dz_n)}{\mu_{n-1}(dz_n)}, \tag{3.5}$$

as shown in Theorem D.1, which requires of course $\bar{\mu}_n \ll \mu_{n-1}$. We focus here on the following choice for the R_n kernel (which we call the refreshment kernel)

$$R_n(z, dz') = \delta_x \otimes \varpi_{n-1}(d(x', v')), \tag{3.6}$$

which means that we keep the position fixed and refresh the velocity. Given a particle $Z^{(i)}$, the kernel described in Equation (3.2) corresponds to constructing deterministic trajectories

$$\mathbf{Z}_{n-1}^{(i)} = (Z_{n-1}^{(i)}, \psi_{n-1}(Z_{n-1}^{(i)}), \ldots, \psi_{n-1}^T(Z_{n-1}^{(i)}))$$

computing the normalized weights $\left\{W_{n-1,\ell}(Z_{n-1}^{(i)}) : i \in [\![N]\!], \ell \in [\![0,T]\!]\right\}$ using Equation (3.3), sampling a trajectory index

$$k_n^{(i)} \sim \mathrm{Cat}\left(W_{n-1,0}(Z_{n-1}^{(i)}), W_{n-1,1}(Z_{n-1}^{(i)}), \ldots, W_{n-1,T}(Z_{n-1}^{(i)})\right), \tag{3.7}$$

Algorithm 6: Folded Integrator Snippets

INPUT : Number of particles $N \in \mathbb{Z}_+$, number of iterations $P \in \mathbb{Z}_+$, integration time $T \in \mathbb{Z}_+$, targets $\mu_n = \pi_n \otimes \varpi_n$ for $n \in [\![0, P]\!]$, integrators $\{\psi_n : n \in [\![P]\!]\}$ with $\psi_0 = \mathrm{Id}$.

Sample initial particles $\check{Z}_0^{(i)} \sim \bar{\mu}_0$.
for $n = 1, \ldots, P$ **do**
  for $i = 1, \ldots, N$ **do**
    Sample from forward kernel $\tilde{Z}_n^{(i)} \sim \bar{M}_n(\check{Z}_{n-1}^{(i)}, \cdot)$ as follows:
      Compute trajectory $\check{\mathbf{Z}}_{n-1}^{(i)} = (\check{Z}_{n-1}^{(i)}, \check{Z}_{n-1,1}^{(i)}, \ldots, \check{Z}_{n-1,T}^{(i)})$ with $\check{Z}_{n-1,k}^{(i)} = \psi_{n-1}^k(\check{Z}_{n-1}^{(i)})$.
      Compute weights $W_{n-1,k}(\check{Z}_{n-1}^{(i)})$ using equation (3.3)

$$W_{n-1,k}(\check{Z}_{n-1}^{(i)}) = \frac{\mu_{n-1} \circ \psi_{n-1}^k(\check{Z}_{n-1}^{(i)})}{\sum_{\ell=0}^{T} \mu_{n-1} \circ \psi_{n-1}^\ell(\check{Z}_{n-1}^{(i)})}.$$

      Sample a trajectory index

$$k_{n-1}^{(i)} \sim \mathrm{Cat}\left(W_{n-1,0}(\check{Z}_{n-1}^{(i)}), W_{n-1,1}(\check{Z}_{n-1}^{(i)}), \ldots, W_{n-1,T}(\check{Z}_{n-1}^{(i)})\right)$$

      Set new particle to $\tilde{Z}_n^{(i)} = \left(\check{X}_{n-1,k_{n-1}^{(i)}}^{(i)}, \tilde{V}_{n-1}^{(i)}\right)$ where $\tilde{V}_{n-1}^{(i)} \sim \varpi_{n-1}$.
  end
  Sample $A_n^{(i)}$ such that $\mathbb{P}(A_n^{(i)} = j) = \bar{W}_n(\tilde{Z}_n^{(j)})$ for $i, j \in [\![N]\!]$, where

$$\bar{W}_n(\tilde{Z}_n^{(j)}) \propto \frac{\bar{\mu}_n(\tilde{Z}_n^{(j)})}{\mu_{n-1}(\tilde{Z}_n^{(j)})}. \tag{3.8}$$

  and set $\check{Z}_n^{(i)} = \tilde{Z}_n^{A_n^{(i)}}$ for $i \in [\![N]\!]$.
end
Return particles $\left\{\left(\check{Z}_n^{(i)}, 1\right) : i \in [\![N]\!]\right\}$.

and refreshing the velocity of the corresponding particle. We call this specific SMC sampler the *folded Integrator Snippets* algorithm.

The incremental weights computation leads to the following expression (notice that below we do not cancel out $\mu_n(\tilde{Z}_n^{(i)})$ just yet, we will do that later to devise an equivalent algorithm)

$$\begin{aligned} \bar{w}_n(\tilde{Z}_n^{(i)}) &= \frac{\mu_n(\tilde{Z}_n^{(i)})}{\mu_{n-1}(\tilde{Z}_n^{(i)})} \left[\frac{1}{T+1} \sum_{k=0}^{T} \frac{\mu_n \circ \psi_n^k(\tilde{Z}_n^{(i)})}{\mu_n(\tilde{Z}_n^{(i)})} \right] \\ &= \frac{\mu_n(\tilde{Z}_n^{(i)})}{\mu_{n-1}(\tilde{Z}_n^{(i)})} \left[\frac{1}{T+1} \sum_{k=0}^{T} \frac{\mu_n \circ \psi_n^k \circ \psi_{n-1}^{k_{n-1}^{(i)}}(\check{Z}_{n-1}^{(i)})}{\mu_n(\tilde{Z}_n^{(i)})} \right]. \end{aligned} \tag{3.9}$$

This is problematic because even if we assume $\psi_n = \psi$ for all $n \in [\![1, P]\!]$, it can require computing additional trajectory points that will not be used, except for the weights themselves. Indeed, when there exist particle indices that have not been resampled, i.e. $\{A_n^{(i)} : i \in [\![N]\!]\} \subset [\![N]\!]$, then for any $k > T - k_{n-1}^{(i)}$ the trajectory points $\psi^{k+k_{n-1}^{(i)}}(\tilde{Z}_n^{(i)})$ will only be used for the weight computation and cannot be recycled at the next iteration (in case we do not fully refresh the velocity, see the extensions subsection 3.9).

3.3.2 Unfolded Integrator Snippets - A more efficient implementation

The innovation of this work is that it is possible to write an equivalent but much more efficient algorithm that is probabilistically equivalent to this SMC sampler, which we call *unfolded Integrator Snippets* algorithm and is shown in Algorithm 7.

Algorithm 7: Unfolded Integrator Snippets

INPUT : Number of particles $N \in \mathbb{Z}_+$, number of iterations $P \in \mathbb{Z}_+$, integration time $T \in \mathbb{Z}_+$, targets $\mu_n = \pi_n \otimes \varpi_n$ for $n \in [\![0, P]\!]$, integrators $\{\psi_n : n \in [\![P]\!]\}$.

Sample initial particles $Z_0^{(i)} \sim \mu_0$.
for $n = 1, \ldots, P$ **do**
  for $i = 1, \ldots, N$ **do**
    Compute trajectory $\mathbf{Z}_{n-1}^{(i)} = (Z_{n-1}^{(i)}, Z_{n-1,1}^{(i)}, \ldots, Z_{n-1,T}^{(i)})$ where $Z_{n-1,k}^{(i)} = \psi_n^k(Z_{n-1}^{(i)})$.
  end
  Resample N particles $\{\bar{Z}_n^{(i)} = (\bar{X}_n^{(i)}, \bar{V}_n^{(i)}) : i \in [\![N]\!]\}$ from these $N \times (T+1)$ using weights

$$\breve{W}_{n,k}^{(i)} = \frac{\breve{w}_{n,k}^{(i)}}{\sum_{j=1}^{N} \sum_{\ell=0}^{T} \breve{w}_{n,\ell}^{(j)}} \qquad \text{where} \qquad \breve{w}_{n,k}^{(i)} := \frac{\mu_n(Z_{n-1,k}^{(i)})}{\mu_{n-1}(Z_{n-1}^{(i)})}$$

  and set $Z_n^{(i)} = (\bar{X}_n^{(i)}, V_n^{(i)})$ where $V_n^{(i)} \sim \varpi_n$ for $i \in [\![N]\!]$.
end
Return particles $\left\{\left(Z_n^{(i)}, 1\right) : i \in [\![N]\!]\right\}$.

The equivalence between the folded and unfolded algorithm is proved in Appendix D.1.1 and in Figure 3.1 we provide a diagram to intuitively demonstrate this. They is to realise that the distribution of the random variables $\tilde{Z}_n^{(1:N)}$ given $\tilde{Z}_{n-1}^{(1:N)}$ in Algorithm 6 is the same as the distribution of the random variables $Z_n^{(1:N)}$ given $Z_{n-1}^{(1:N)}$.

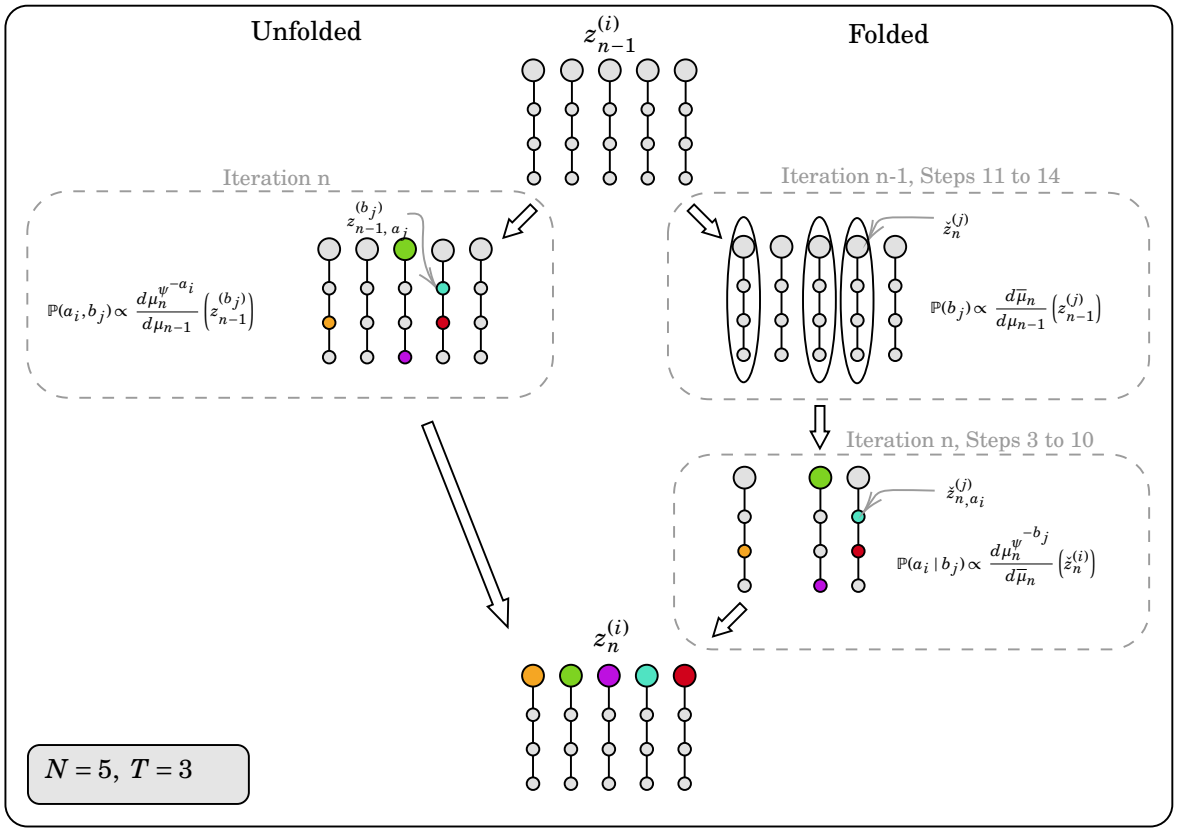

Figure 3.1: Illustration of the equivalence between folded and unfolded. At the top we consider $N = 5$ seed particles (bigger circles) and their respective snippets with $T + 3$. Left: the unfolded algorithm resamples N particles from the $N(T + 1)$ in one go. Right: the folded algorithm, when considered from the second half of the previous iteration (steps 11 to 14) to the first half of the next iteration (steps 3 to 10) first selects the snippets, and then picks particles along those snippets.

We notice that resampling N particles from $N(T + 1)$ can be done in three ways

- sampling the trajectory indices, and then the seed particle indices
- sampling the seed particle indices, and then the trajectory indices
- sampling both at once.

These three methods are equivalent as should be evident by simple rules of conditional probability, Appendix D.1.4 provides a simple proof of this. It will often be helpful in our numerical

experiments to compare an integrator snippet with a slightly different version of itself, i.e. where we use the integrator $\tilde{\psi}_n := \psi_n^T$ for $T \in \mathbb{Z}_+$ (of course, the two are the same algorithm for $T = 1$).

Definition 3.2 (Endpoint Integrator Snippet)**.** Given an integrator snippet that at iteration $n \in [\![0, P]\!]$ targets μ_n using integrator ψ_n, we call an integrator snippet using $\tilde{\psi}_n = \psi_n^T$ its endpoint version.

3.3.2.1 The distribution of particles in the unfolded Integrator snippet

As showed in the previous section, Algorithm 6 and 7 initialize particles $\left\{\check{Z}_0^{(i)} : i \in [\![N]\!]\right\}$ and $\left\{Z_0^{(i)} : i \in [\![N]\!]\right\}$, respectively, from the same starting distribution μ_0, and the distribution of "unfolded" particles $\left\{Z_{n-1}^{(i)} : i \in [\![N]\!]\right\}$ from one iteration to the next is the same as the distribution of the "folded" particles sampled from the forward kernel at each iteration, i.e. $\left\{\tilde{Z}_n^{(i)} : i \in [\![N]\!]\right\}$. When $N \to \infty$, we know the distribution of the proposal particles in an SMC sampler targeting $\bar{\mu}_n$ with forward kernel $\bar{M}_n$ is $\bar{\mu}_{n-1}\bar{M}_n = \mu_{n-1}$, which of course means that asymptotically the unfolded particles are also distributed according to μ_{n-1}. The particles of interest in the two algorithms above are therefore distributed as

$$\begin{aligned} \check{Z}_n &\sim \bar{\mu}_n && n \in [\![0, P]\!] \\ \tilde{Z}_n &\sim \mu_{n-1} && n \in [\![P]\!] \\ Z_n &\sim \mu_n && n \in [\![0, P]\!]. \end{aligned}$$

Importantly, by running the unfolded algorithm one can still compute the SMC incremental weights of the folded algorithm, since these are simply the mean unfolded weights along a trajectory, that is

$$\frac{1}{T+1}\sum_{k=0}^{T} \check{w}_{n,k}^{(i)} = \frac{1}{T+1}\sum_{k=0}^{T} \frac{\mu_n \circ \psi_n(Z_{n-1}^{(i)})}{\mu_{n-1}(Z_{n-1}^{(i)})} = \frac{\bar{\mu}_n(Z_{n-1}^{(i)})}{\mu_{n-1}(Z_{n-1}^{(i)})} = \bar{w}_n(Z_{n-1}^{(i)}).$$

Notice that in the folded Integrator Snippet the incremental weights are only computed for the new particles $\tilde{Z}_n^{(j)}$. This means, that all the remaining particles within the trajectory do not actually get an incremental weight.

3.3.3 Estimating Expectations

3.3.3.1 Distributions involved

Define the following joint distribution over $([\![0, T]\!] \times \mathsf{Z}, \mathscr{P}([\![0, T]\!]) \otimes \mathscr{Z})$

$$\bar{\mu}_{n,T}(\mathsf{A} \times \mathsf{B}) = \frac{1}{T+1}\sum_{k \in \mathsf{A}} \delta_k(\mathsf{A}) {\mu_n}^{\psi_n^{-k}}(\mathsf{B}) \qquad \forall\, \mathsf{A} \in \mathscr{P}([\![0, T]\!]), \forall\, \mathsf{B} \in \mathscr{Z},$$

where δ_k is a Dirac measure, which admits the marginals on $(\mathsf{Z}, \mathscr{Z})$ and $([\![0,T]\!], \mathscr{P}([\![0,T]\!]))$ respectively

$$\bar{\mu}_n(dz) = \frac{1}{T+1} \sum_{k \in [\![0,T]\!]} {\mu_n}^{\psi_n^{-k}}(dz)$$
$$\bar{\mu}_n(\mathsf{A}) = \frac{|\mathsf{A}|}{T+1}.$$

Under modest conditions, there exists a regular conditional distribution $\bar{\mu}_n(\cdot \mid z)$ associated with $\bar{\mu}_{n,T}$ and $\bar{\mu}_n(dz)$ which has the form

$$\bar{\mu}_n(\mathsf{A} \mid z) = \sum_{k \in \mathsf{A}} \delta_k(\mathsf{A}) \frac{{\mu_n}^{\psi_n^{-k}}(dz)}{\sum_{\ell \in [\![0,T]\!]} {\mu_n}^{\psi_n^{-\ell}}(dz)} \qquad \forall\, \mathsf{A} \in \mathscr{P}([\![0,T]\!]),\ \forall\, z \in \mathsf{Z}.$$

In the context of Integrator Snippets, we shall only be interested in sets $\{k\} \times dz$ for any $k \in [\![0,T]\!]$, in which case the expressions of the distributions above simplify

$$\bar{\mu}_{n,T}(k, dz) = \frac{1}{T+1} \mu_n^{\psi_n^{-k}}(dz)$$
$$\bar{\mu}_n(dz) = \frac{1}{T+1} \sum_{k=0}^{T} \mu_n^{\psi_n^{-k}}(dz)$$
$$\bar{\mu}_n(k \mid z) = \frac{\bar{\mu}_{n,T}(k, dz)}{\bar{\mu}_n(dz)} = \frac{\mu_n^{\psi_n^{-k}}(dz)}{\sum_{\ell \in [\![0,T]\!]} \mu_n^{\psi_n^{-\ell}}(dz)}.$$

The key property of this kernel is that expectations of a measurable test function F with respect to the joint can be decomposed into nested expectations with respect to the marginal and the kernel

$$\bar{\mu}_{n,T}(F) = \int \left[\sum_{k=0}^{T} F(k,z) \bar{\mu}_n(k \mid z) \right] \bar{\mu}_n(dz). \tag{3.10}$$

3.3.3.2 Folded Estimator

The expectation $\mu_n(h)$ can be rewritten by leveraging the mixture structure

$$\begin{aligned}
\mu_n(h) &= \frac{1}{T+1} \sum_{k=0}^{T} \int h(z) \mu_n(dz) && \text{indep of } k \\
&= \frac{1}{T+1} \sum_{k=0}^{T} \int h \circ \psi_n^k(z) {\mu_n}^{\psi_n^{-k}}(dz) && \text{change of variables} \\
&= \sum_{k=0}^{T} \int h \circ \psi_n^k(z) \bar{\mu}_{n,T}(k, dz) && \text{def } \bar{\mu}_{n,T} \\
&= \int \sum_{k=0}^{T} h \circ \psi_n^k(z) \bar{\mu}_{n,T}(k, dz) && (3.11)
\end{aligned}$$

$$= \int \left[\sum_{k=0}^{T} h \circ \psi_n^k(z) \bar{\mu}_n(k \mid z) \right] \bar{\mu}_n(dz) \qquad \text{property (3.10),} \tag{3.12}$$

and having obtained samples $\check{Z}_n^{(i)} \sim \bar{\mu}_n$ for $i \in [\![N]\!]$ expectations can be estimated as follows

$$\mu_n(h) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{k=0}^{T} h \circ \psi_n^k(\check{Z}_n^{(i)}) \frac{\mu_n \circ \psi_n^k(\check{Z}_n^{(i)})}{\sum_{\ell \in [\![0,T]\!]} \mu_n \circ \psi_n^\ell(\check{Z}_n^{(i)})}. \tag{3.13}$$

This estimator is impractical because it requires using Algorithm 6 which we have seen wastes computational resources.

3.3.3.3 Unfolded Estimators

An alternative estimator can be found by assuming $\bar{\mu}_{n,T}(k, dz) \ll \mu_{n-1}$

$$\begin{aligned}
\mu_n(h) &= \int \left[\sum_{k=0}^{T} h \circ \psi_n^k(z) \bar{\mu}_n(k \mid z) \right] \frac{\bar{\mu}_n(dz)}{\mu_{n-1}(dz)} \mu_{n-1}(dz) \\
&= \int \left[\sum_{k=0}^{T} h \circ \psi_n^k(z) \frac{\bar{\mu}_{n,T}(k, dz)}{\bar{\mu}_n(dz)} \right] \frac{\bar{\mu}_n(dz)}{\mu_{n-1}(dz)} \mu_{n-1}(dz) \\
&= \int \left[\sum_{k=0}^{T} h \circ \psi_n^k(z) \frac{\bar{\mu}_{n,T}(k, dz)}{\mu_{n-1}(dz)} \right] \mu_{n-1}(dz) \\
&= \frac{1}{T+1} \int \left[\sum_{k=0}^{T} h \circ \psi_n^k(z) \frac{\mu_n \circ \psi_n^k(z)}{\mu_{n-1}(z)} \right] \mu_{n-1}(dz) \\
&\approx \frac{1}{N(T+1)} \sum_{i=1}^{N} \sum_{k=0}^{T} h \circ \psi_n^k(Z_{n-1}^{(i)}) \frac{\mu_n \circ \psi_n^k(Z_{n-1}^{(i)})}{\mu_{n-1}(Z_{n-1}^{(i)})} \qquad Z_{n-1}^{(i)} \sim \mu_{n-1}.
\end{aligned}$$

Notice the resemblance between this estimator and the estimator that one would obtain by simply replacing $\bar{\mu}_n$ with the mixture of empirical distributions (3.1) in the integral above, with μ_{n-1} as proposal distribution. This estimator is correct only if we can evaluate normalized densities μ_n and μ_{n-1}. In most circumstances, one can only evaluate densities up to a normalizing constant

$$\begin{aligned}
\mu_n(h) &= \int \left[\sum_{k=0}^{T} h \circ \psi_n^k(z) \frac{\bar{\mu}_{n,T}(k, dz)}{\mu_{n-1}(dz)} \right] \mu_{n-1}(dz) \\
&= \frac{1}{T+1} \sum_{k=0}^{T} \frac{\mathcal{Z}_{\mu_{n-1}}}{\mathcal{Z}_{\mu_n^{\psi^{-k}}}} \int h \circ \psi_n^k(z) \frac{\tilde{\mu}_n \circ \psi_n^k(z)}{\tilde{\mu}_{n-1}(z)} \tilde{\mu}_{n-1}(dz).
\end{aligned}$$

There are two approaches to constructing a self-normalized integrator snippet estimator. The first approach consists in estimating each ratio separately for each $k \in [\![0, T]\!]$. We do this using the standard self-normalized importance sampling trick shown in Section 1.7 for target $\mu_n^{\psi^{-k}}$ with proposal μ_{n-1}

$$\frac{\mathcal{Z}_{\mu_n^{\psi^{-k}}}}{\mathcal{Z}_{\mu_{n-1}}} \approx \frac{1}{N} \sum_{i=1}^{N} \breve{w}_{n,k}(Z_{n-1}^{(i)}) \qquad \text{where} \qquad \breve{w}_k(z) = \frac{\tilde{\mu}_n \circ \psi_n^k(z)}{\tilde{\mu}_{n-1}(z)}. \tag{3.14}$$

Plugging this into the estimator we obtain

$$\hat{h}_A = \frac{1}{T+1}\sum_{k=0}^{T}\frac{\sum_{i=1}^{N} h\circ\psi_n^k(Z_{n-1}^{(i)})\check{w}_{n,k}(Z_{n-1}^{(i)})}{\sum_{j=1}^{N}\check{w}_{n,k}(Z_{n-1}^{(j)})}. \tag{3.15}$$

In practice, it can happen that for a specific $k \in [\![0,T]\!]$, the weights are all zero i.e. $\check{w}_{n,k}^{(i)} = 0$ for $i \in [\![N]\!]$. We therefore suggest first computing all the sums of the form $\sum_{j=1}^{N}\check{w}_{n,k}^{(j)}$ and then run the outer sum only for those $k \in [\![0,T]\!]$ such that this sum is positive. The second approach uses the fact that all pushforwards share the same normalizing constant, as noted in Subsection 3.2.2. This means we can take the normalizing constant out and estimate it using all samples

$$\frac{\mathcal{Z}_{\mu_n}}{\mathcal{Z}_{\mu_{n-1}}} \approx \frac{1}{N(T+1)}\sum_{k=0}^{T}\sum_{i=1}^{N}\check{w}_{n,k}(Z_{n-1}^{(i)}), \tag{3.16}$$

leading to a different estimator

$$\hat{h}_B = \frac{\sum_{k=0}^{T}\sum_{i=1}^{N} h\circ\psi_n^k(Z_{n-1}^{(i)})\check{w}_{n,k}(Z_{n-1}^{(j)})}{\sum_{k=0}^{T}\sum_{i=1}^{N}\check{w}_{n,k}(Z_{n-1}^{(i)})}. \tag{3.17}$$

Estimating the ratio of normalizing constants is important to estimate expectations and, as we shall see in subsection 3.4.2, to define a metric of distance similar in spirit to ESJD. In general, we will refer to each of the $T+1$ estimators (3.14) (and consequently (3.15)) as estimators A, and to (3.16) (and consequently (3.17)) as estimator B. Notice that estimator B of the ratio of normalizing constants is simply the average of all the $T+1$ estimators A. These two classes of estimators are compared in subsection 3.5.1.

3.3.4 Rao-Blackwellization

In the previous subsection we described one way to estimate expectations using the output of the folded algorithm. In particular, Equation (3.12) shows that expectations of a test function h with respect to μ_n can be written in TOWER form, as two nested expectations where the outer-most one is with respect to the mixture $\bar{\mu}_n$. In this subsection we show that the variance of the estimator defined as the inner expectation, i.e.

$$\sum_{k=0}^{T} h\circ\psi_n^k(\check{Z}_n^{(i)})\frac{\mu_n\circ\psi_n^k(\check{Z}_n^{(i)})}{\sum_{\ell\in[\![0,T]\!]}\mu_n\circ\psi_n^\ell(\check{Z}_n^{(i)})}$$

can be lower than that of the naive estimator of h found using μ_n. For this section, define $\bar{h}_n(k,z_n) = h\circ\psi_n^k(z_n)$ and recall that in section 3.3.3 we found

$$\mathbb{E}_{\mu_n}[h(Z_n)] = \mathbb{E}_{\bar{\mu}_{n,T}}[\bar{h}(K,Z)] = \mathbb{E}_{\bar{\mu}_n}[\mathbb{E}_{\bar{\mu}_n}[\bar{h}(K,Z)\mid Z]]. \tag{3.18}$$

We now find two expressions for the variance of the test function under the joint target $\bar{\mu}_{n,T}$. The first is found using the variance decomposition formula

$$\mathbb{V}_{\bar{\mu}_{n,T}}[\bar{h}(K,Z)] = \mathbb{V}_{\bar{\mu}_n}[\mathbb{E}_{\bar{\mu}_n}[\bar{h}(K,Z) \mid Z]] + \mathbb{E}_{\bar{\mu}_n}[\mathbb{V}_{\bar{\mu}_n}[\bar{h}(K,Z) \mid Z]],$$

and the second using the standard property $\mathbb{V}[Y] = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2$.

$$\begin{aligned}\mathbb{V}_{\bar{\mu}_{n,T}}[\bar{h}(K,Z)] &= \mathbb{E}_{\bar{\mu}_{n,T}}[\bar{h}(K,Z)^2] - \mathbb{E}_{\bar{\mu}_{n,T}}[\bar{h}(K,Z)]^2 \\ &= \mathbb{E}_{\mu_n}[h(Z)^2] - \mathbb{E}_{\mu_n}[h(Z)]^2 \\ &= \mathbb{V}_{\mu_n}[h(Z)].\end{aligned}$$

This tells us that one can decompose the original variance as

$$\mathbb{V}_{\mu_n}[h(Z)] = \mathbb{V}_{\bar{\mu}_n}[\mathbb{E}_{\bar{\mu}_n}[\bar{h}(K,Z) \mid Z]] + \mathbb{E}_{\bar{\mu}_n}[\mathbb{V}_{\bar{\mu}_n}[\bar{h}(K,Z) \mid Z]]$$

We call $\mathbb{E}_{\bar{\mu}_n}[\bar{h}(K,Z) \mid Z]$ the Rao-Blackwellized estimator of $\bar{h}$ (and hence h) since it has the same mean, and the variance reduction term is provided by $\mathbb{E}_{\bar{\mu}_n}[\mathbb{V}_{\bar{\mu}_n}[\bar{h}(K,Z) \mid Z]]$, since $\mathbb{V}_{\bar{\mu}_n}[\bar{h}(K,Z) \mid Z] \geq 0$. The Rao-Blackwellized estimator is

$$\mathbb{E}_{\bar{\mu}_n}[\bar{h}(K,Z) \mid Z] = \sum_{k=0}^{T} \bar{h}(k,z)\bar{\mu}_n(k \mid z) = \sum_{k=0}^{T} h(\psi_n^k(z))\bar{\mu}_n(k \mid z),$$

which is, once again, the inner term in the folded estimator (3.13).

3.4 Tracking performance of Integrator Snippets

The performance of SMC samplers is usually assessed in terms of ESS, acceptance probability of its forward MH kernel and, sometimes, Root Mean Squared Error (RMSE) of the estimates of expectations from the true values. While the folded integrator snippet in Algorithm 6 is technically an SMC sampler, in practice we recommend using the unfolded algorithm instead, to avoid extra computations, as illustrated in equation (3.9). The unfolded algorithm, however, is not an SMC sampler and as such, it is not clear how one should assess its performance. The aim of this section is to introduce and derive metrics that we use in our experiments.

3.4.1 Acceptance Metrics

3.4.1.1 Acceptance probability in folded integrator snippets

For the folded integrator snippet in Algorithm 6, we sample $\tilde{Z}_n^{(i)} \sim \bar{M}_n(\check{Z}_{n-1}^{(i)}, \cdot)$ by sampling a trajectory index $k \in [\![0, T]\!]$ and then refreshing the velocity. For this reason, by analogy with MH, for each particle $\check{Z}_{n-1}^{(i)}$ once could define rejection to be to sampling $k = 0$ and $\tilde{V}_{n-1}^{(i)} = \check{V}_{n-1}^{(i)}$, but for practical purposes, we are not interested in the velocity component of each particle, and thus

define a rejection when $k = 0$, and a consequently define an acceptance when $k \geq 1$. The expected acceptance probability could then be estimated as

$$\hat{a}_{n,\text{folded}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left(k_{n-1}^{(i)} \geq 1\right).$$

3.4.1.2 Estimating acceptance in unfolded integrator snippets

As mentioned in subsection 3.3.2 and proved in Appendix D.1.4, one way to resample N particles out of the $N(T+1)$ in Algorithm 7 is to first sample N particle indices

$$t_n^{(j)} \sim \text{Cat}\left(\sum_{k=0}^{T} \breve{W}_{n,0}^{(0)}, \ldots, \sum_{k=0}^{T} \breve{W}_{n,k}^{(N)}\right) \qquad j = 1, \ldots, N,$$

and then sample N trajectory indices from each of these trajectories

$$k_n^{(j)} \sim \text{Cat}\left(\frac{\breve{W}_{n,0}^{(t_n^{(j)})}}{\sum_{k=0}^{T} \breve{W}_{n,k}^{(t_n^{(j)})}}, \ldots, \frac{\breve{W}_{n,T}^{(t_n^{(j)})}}{\sum_{k=0}^{T} \breve{W}_{n,k}^{(t_n^{(j)})}}\right) \qquad j = 1, \ldots, N.$$

Notice that the weights used to sample the particle indices correspond exactly to the incremental weights in the folded algorithm (Algorithm 6), and the weights used to sample the trajectory indices correspond exactly to the weights within the forward kernel $\bar{M}_{n+1}$, i.e. the $W_{n,k}$'s, which is why the two implementations are probabilistically equivalent. This means that by using this two-stage resampling step in Algorithm 7 we are able to recover a Monte Carlo estimate of the acceptance probability of the folded algorithm

$$\hat{a}_{n,\text{unfolded}} = \frac{1}{N}\sum_{j=1}^{N} \mathbb{I}\left(k_n^{(j)} \geq 1\right).$$

However, it is possible to estimate the folded acceptance probability when running the unfolded algorithm, its interpretation as an acceptance probability is lost. Therefore, to avoid confusion, we define this metric as the *proportion of particles moved*, shorted pm_n,

$$\text{pm}_n = \frac{1}{N}\sum_{j=1}^{N} \mathbb{I}\left(k_n^{(j)} \geq 1\right). \tag{3.19}$$

This metric estimates the true acceptance probability of the forward kernel, but it does not contain enough information to paint the whole picture. Suppose that half of our trajectory indices are one and half are zero (for simplicity, let N be even)

$$k_n^{(j)} = \begin{cases} 0 & j = 1, \ldots, \frac{N}{2} \\ 1 & j = \frac{N}{2}+1, \ldots, N \end{cases},$$

then by definition $\text{pm}_n = 0.5$, which may seem like a high acceptance probability in a standard SMC context, but if $T \in \mathbb{Z}_+$ is large then it means we are not leveraging one of the benefits of

integrator snippets, which is how they manage to use the whole trajectory to explore the state space. Most of the computation is being wasted. On its own then, the proportion of particles moved does not tell us how far along the trajectory we resample on average. A better metric is provided by *median index proportion*

$$\text{mip}_n = \frac{\text{quantile}_{0.5}\left(k_n^{(1)}, \ldots, k_n^{(N)}\right)}{T} \in [0,1]. \tag{3.20}$$

This metric, has the complementary problem of pm_n. Suppose that out of the $Z_{n-1}^{(i)}$ particles at the start of the n^{th} iteration of Algorithm 7 only $Z_{n-1}^{(1)}$ will construct a trajectory with positive weights. Then when sampling the particle indices $t_n^{(j)}$ we will only sample $t_n^{(j)} = 1$ and the $k_n^{(j)}$'s will never reflect the particle diversity needed to actually sample effectively from μ, since both pm_n and mip_n can be high even if all resampled particles come from the trajectory constructed starting from $Z_{n-1}^{(1)}$. We define *particle diversity* as the proportion of unique "trajectories" from which we resample

$$\text{pd}_n = \frac{\left|\left\{t_n^{(1)}, \ldots, t_n^{(N)}\right\}\right| - 1}{N-1} \in [0,1],$$

notice that we have defined it so that is lies in $[0,1]$. It is important to monitor both of these metrics separately to assess how far along the trajectory we are moving on average, and how many particles are contributing to our exploration of the space, respectively. It is possible to obtain a holistic metric of performance, the *median path diversity*

$$\text{mpd}_n := \sqrt{\text{mip}_n \times \text{pd}_n} \in [0,1]. \tag{3.21}$$

3.4.2 Distance Metrics

3.4.2.1 ESJD in unfolded integrator snippets

We show in Appendix D.1.6 that it is possible to define two equivalent metrics of expected travelled distance such that when they are maximised, the correlation between samples is minimised

$$\text{ESJD}_{A,n} = \frac{1}{(T+1)^2} \sum_{k=0}^{T} \sum_{\ell=k+1}^{T} \sum_{i=1}^{N} \left[h(\psi_n^{\ell}(Z_{n-1}^{(i)})) \frac{\breve{w}_{n,\ell}(Z_{n-1}^{(i)})}{\sum_{j=1}^{N} \breve{w}_{n,\ell}(Z_{n-1}^{(j)})} - h(\psi_n^{k}(Z_{n-1}^{(i)})) \frac{\breve{w}_{n,k}(Z_{n-1}^{(i)})}{\sum_{j=1}^{N} \breve{w}_{n,k}(Z_{n-1}^{(j)})} \right]^2.$$

$$\text{ESJD}_{B,n} \approx \frac{\sum_{k=0}^{T} \sum_{\ell=k+1}^{T} \sum_{i=1}^{N} \left(h \circ \psi_n^{\ell}(Z_{n-1}^{(i)}) \breve{w}_{n,\ell}(Z_{n-1}^{(i)}) - h \circ \psi_n^{k}(Z_{n-1}^{(i)}) \breve{w}_{n,k}(Z_{n-1}^{(i)}) \right)^2}{\left(\sum_{k=0}^{T} \sum_{i=1}^{N} \breve{w}_{n,k}(Z_{n-1}^{(i)}) \right)^2}.$$

The difference between these two estimators is the same between the different estimators found in subsection 3.3.3.2. The first one approximates the normalizing constant of each pushforward individually, while the second one compute an average across all $N(T+1)$ particles.

3.4.2.2 Histograms of resampled particles

One of the great advantages of using the entire trajectory is that after the resampling step (step 6 in Algorithm 7) one has at disposal information about how far, on average, particles have travelled along the trajectory. Suppose that

$$\{(t^{(1)}, k^{(1)}), \ldots, (t^{(N)}, k^{(N)})\}$$

are the N pairs of particle-trajectory obtained during resampling, then one can construct a histogram of the $k^{(j)}$'s. This histogram can be used to assess the performance of the algorithm and to devise adaptation strategies for the parameter T, as described in Section 3.8.

3.4.3 Measuring ESS

Consider the folded integrator snippet in Algorithm 6. Given the normalized incremental weights at iteration $n \in [\![P]\!]$, one can compute the ESS as in Section 1.7.3

$$\mathrm{ESS} \approx \frac{1}{\sum_{i=1}^{N} \bar{W}_n (\tilde{Z}_n^{(i)})^2}.$$

This ESS, however, is in relation to the target of Algorithm 6, which is the mixture $\bar{\mu}_n$ and not our actual target μ. Following a similar process to derive the ESS in importance sampling one can find expressions to estimate the ESS for μ, but these depend on the test function h and involve the expression of the Monte Carlo variance for an IID sample of the same size. These expressions are discussed in Appendix D.1.3 but we don't use them in our experiments.

3.5 Numerical Experiments for Integrator Snippets

3.5.1 Pushforward Importance Sampling

3.5.1.1 Task overview

We wish to understand the behavior and effectiveness of using integrator snippets, as well as the metrics we introduced in the previous section. Consider a bivariate target distribution $\pi(dx) = \mathcal{N}(dx \mid \varkappa, \Sigma)$ with covariance matrix

$$\Sigma = \begin{pmatrix} 1 & 0.3 \\ 0.3 & 0.1 \end{pmatrix},$$

and an auxiliary distribution $\varpi(dv) = \mathcal{N}(dv \mid 0_2, \mathrm{I}_2)$, so that our extended target (in the sense of the beginning of 3) is $\mu = \pi \otimes \varpi$. Our initial distribution is a standard bivariate normal $\pi_0(dx) = \mathcal{N}(dx \mid 0_2, \mathrm{I}_2)$, and we set $\mu_0 = \pi_0 \otimes \varpi$. We will consider $\varkappa = (-2, 2)$ to test the behavior of the algorithms for $P = 1$ (this is the number of iterations in Algorithm 7) when the proposal

and the target are separated by low-density regions. Our baseline algorithm is an SMC sampler whose forward kernel (defined below) is a product of $T \in \mathbb{Z}_+$ RWM kernels targeting π with the same step size $\delta > 0$, meaning that to mutate a particle, we perform T RWM steps, we call this the SMC-RWM algorithm. More precisely, we consider an SMC sampler with a reversible triplet, where the forward kernel is specified by

$$
\begin{aligned}
M(x_0, dx_T) &= \int_{\mathbb{R}^{2(T-1)}} \prod_{k=1}^{T} P(x_{k-1}, dx_k) \\
P(x_{k-1}, dx_k) &= a(x_{k-1}, x_k) \mathcal{N}(dx_k \mid x_{k-1}, \mathrm{I}_2) + \left[\int (1 - a(x_{k-1}, x'_k)) \mathcal{N}(dx'_k \mid x_{k-1}, \mathrm{I}_2) \right] \delta_{x_{k-1}}(dx_k) \\
a(x_{k-1}, x_k) &= \min \left\{ 1, \frac{\pi(x_k)}{\pi(x_{k-1})} \right\},
\end{aligned}
$$

so that the new particle $X_T^{(i)}$ is obtained from $X_0^{(i)}$ by performing $T \in \mathbb{Z}_+$ steps of RWM. This is compared against two competitors. The first competitor is an integrator snippet that uses the Deterministic Walk (DW) integrator, which constructs a RWM-like trajectory using T separate perturbations and uses this whole trajectory of length $T+1$ within the integrator snippet framework described above. Full details can be found in Appendix D.1.5.1, but here we give some intuition. Our space of interest is $(\mathsf{Z}, \mathscr{Z}) = (\mathsf{X} \times \mathsf{V}^T, \mathscr{X} \otimes \mathscr{V}^{\otimes T})$ and our target distribution is now $\mu(dx, dv_{1:T}) = \pi(dx) \bigotimes_{k=1}^{T} \varpi(dv_k)$, meaning that our auxiliary space now consists of T distinct velocities. Loosely speaking (some modifications is required to make sure the integrator satisfies the properties described in the rest of the text, see Appendix D.1.5.1), the integrator constructs the following trajectory (in the X component) when applied to a particle $Z_0^{(i)} = (X_0^{(i)}, V_0^{(i)}, \ldots, V_T^{(i)})$

$$
X_k^{(i)} := X_{k-1}^{(i)} + \delta V_k^{(i)} \qquad k \in [\![T]\!].
$$

The power of our framework, is that it can easily be applied to generate random trajectories using appropriate auxiliary variables and integrators, although for more complex transformations, the user is recommended to use the more general framework that will be introduced in Section 3.6. The second competitor is its endpoint version (see Definition 3.2), so that the trajectory consists only of the first and last point in the trajectory above, $(X_0^{(i)}, X_0^{(i)} + \delta \sum_{k=1}^{T} V_k^{(i)})$. We run all three algorithms for only one iteration (i.e. $P = 1$), to study the behavior from one step to the next. The two integrator snippets correspond to Pushforward Importance Sampling (PISA) introduced in subsection 3.2.2, although we will perform the final resampling step (step 6 in Algorithm 7) in order to compute metrics such as pm, mpd and mip. All random seeds are fixed, meaning that all algorithms start from the same initial particles, sampled from μ_0, and aim to approximate μ, using the same perturbations for each particle. For a given particle, this means that if during SMC-RWM it accepts the proposal in each of the T MH steps then the trajectory will be exactly that generated by the integrator snippet for that particle. In this scenario, the advantage of the integrator snippet (not the endpoint version) is to be able to use the entire trajectory and therefore be much more robust in the choice of T and δ. At first, it might seem that

there is no advantage in using the endpoint integrator snippet instead of the SMC-RWM but we highlight two important differences. The first one, is that in the endpoint integrator snippet the trajectory is always constructed in full, without the MH step hindering the distanced travelled by the particle, quality that might be helpful in traversing low-density regions. Furthermore, in an SMC-RWM the mutated particles are accepted or rejected by comparing them against the initial particle, but there is no cross-particle comparison, which only happens at the resampling step, by which time good particles could still have been rejected. Instead, in integrator snippets we resample N particles from all $N(T+1)$ trajectory points, which means that when selecting particles we are using a global criterion of goodness. For instance, suppose that a certain particle is in a high-density region and so is most of the trajectory generated from it. In an SMC-RWM we would only keep one point from this trajectory regardless of how good the remaining points were, even if each particle in the path had a much higher density than all the other particles. In contrast, an Integrator Snippet would assign high weight to all the particles in that path and hence will be likely to resample multiple points from it.

3.5.1.2 Reproducibility Details

For the first experiment, we use $N = 10000$ particles samples from μ_0 and study the performance of the three algorithms for $T = 10, 20, 30, 40, 50$ and nine logarithmically spaced values for δ between 0.01 and 100.0. All results are averaged across 100 independent runs. The aim is to show how integrator snippets behave when the target has a narrow ridge of high-density region separated from the more diffuse high-density region of the proposal by a small low-density region.

3.5.1.3 Assessing Effective Sample Size

As mentioned in subsection 3.4.3, the ESS measured based on the weights $\bar{W}_n$ is relative to $\bar{\mu}$ and not μ. Figure 3.2 shows this metric for PISA/ PISA-Endpoint and the standard ESS relative to μ for the SMC sampler. Although these two metrics are not directly comparable, we can observe two behaviors. The SMC sampler's ESS has a clear optimal region (around $\delta = 10^{-1}$) and then quickly drops for larger step sizes. In contrast, the ESS for both PISA/PISA-Endpoint is relatively constant for different values of T and δ, and for PISA there seems to be a slight optimal region around $\delta = 1.0$.

Although the ESS for μ is not a byproduct of the integrator snippet algorithm, for each test function $h : \mathbb{R}^2 \to \mathbb{R}$ it is possible to estimate it empirically. For each of the 100 runs, we compute the estimators $\hat{h}_A$ and $\hat{h}_B$ and calculate their sample variances, which then we plug into the ESS definition in Equation 1.13. The quality of this estimate of the ESS depends on the test function and on the number of independent runs, so for a fair comparison, we estimate the ESS for the SMC sampler in the same manner, even though the closed form (1.14) is available (and shown in the previous figure). The first row of Figure 3.3 shows the approximate ESS, computed from $\hat{h}_A$, averaged across the two components of the mean of μ (first moments), the second row shows

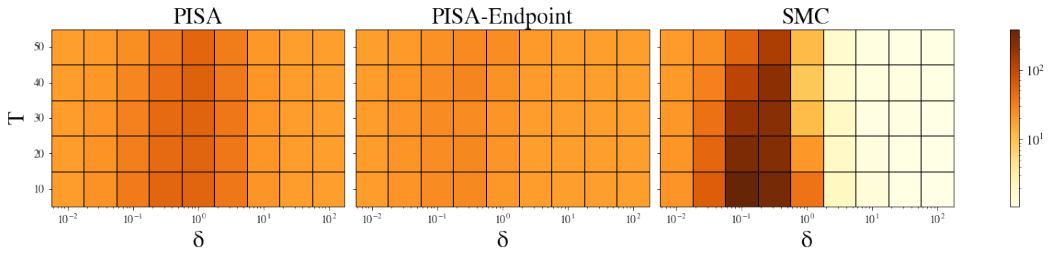

Figure 3.2: The first two plots show the ESS for $\bar{\mu}$, whereas the final plot shows the ESS on μ. We can see how the integrator snippet algorithms are more robust to the choices of T and δ. To help understand the logarithmic scale, the average value at $\delta = 1.0$ for PISA is 54, and the average value at $\delta = 0.3$ for SMC is 227.

the averaged ESS of its second moments, and the final row shows the mixed second moment of the two components. We notice that the robustness observed in the ESS for $\bar{\mu}$ is also present in the ESS for μ for all test functions involved here (more complex test functions were tested and the same behavior was observed), since it manages to maintain higher ESS for larger step sizes. This robustness is observed also in our endpoint version, however we notice how the endpoint version does not seem to have an optimal region of ESS around $\delta = 1.0$, unlike PISA. In the latter, we can see how the optimal region is between $\delta = 1$ and $\delta = 10$ whereas the one for the SMC sampler seems to be between $\delta = 0.1$ and $\delta = 1$. To aid with the logarithmic scale, we report the δ leading to the highest ESS value averaged across all combinations of T. For instance, for PISA, the highest values were achieved at (top to bottom) $\delta = 1.0, 1.0, 1.0, 3.16$ and were respectively ESS $= 135, 754, 142$. PISA-Endpoint achieved its best values at $\delta = 0.316, 0.1, 0.1$ which were ESS $= 23, 137, 34$. SMC achieved its highest ESS values $44, 287, 99$ at $\delta = 0.1$.

We now compare the ESS estimate obtained from estimators $\hat{h}_B$ in Figure 3.4. The first observation is that robustness to δ and T is even more evident for $\hat{h}_B$, for both PISA and PISA-Endpoint. The ESS does not decrease substantially for the last two values of δ as in the previous picture. Surprisingly, robustness is observed even for the PISA-Endpoint algorithm. The downside of this estimator, compared to $\hat{h}_A$, is that now there is no optimal ESS region between $\delta = 1.0$ and $\delta = 10.0$ but the value is roughly constant for all choices of δ and T considered. To summarize, estimators of type B are robust to choices of step size and number of integration steps but, in this specific example, do not achieve as high an ESS as the SMC sampler. On the other hand, estimates of type A are slightly less robust but manage to achieve higher or equal values or ESS for all test functions for PISA, and its endpoint version for its correlation function. To give an idea of the values, the average value across the whole grid for the three rows of PISA are ess $= 15, 86, 21$, whereas the maximum average values over T for the SMC sampler are ess $= 44, 287, 99$ achieved at $\delta = 0.1$.

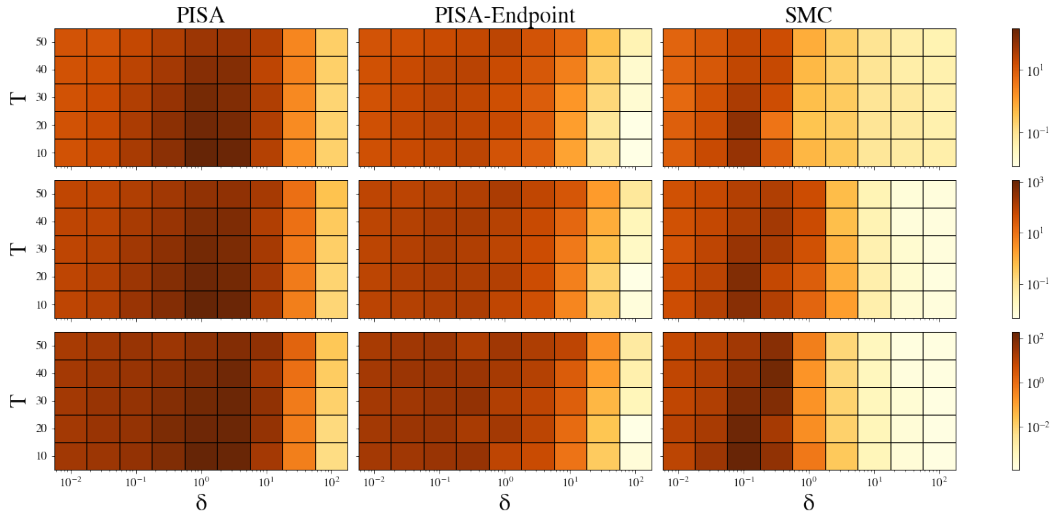

Figure 3.3: ESS for μ estimated empirically over 100 runs from estimators $\hat{h}_A$. First row: average over target mean. Second row: average over target second moments. Third row: ESS for mixed second moment of the two components.

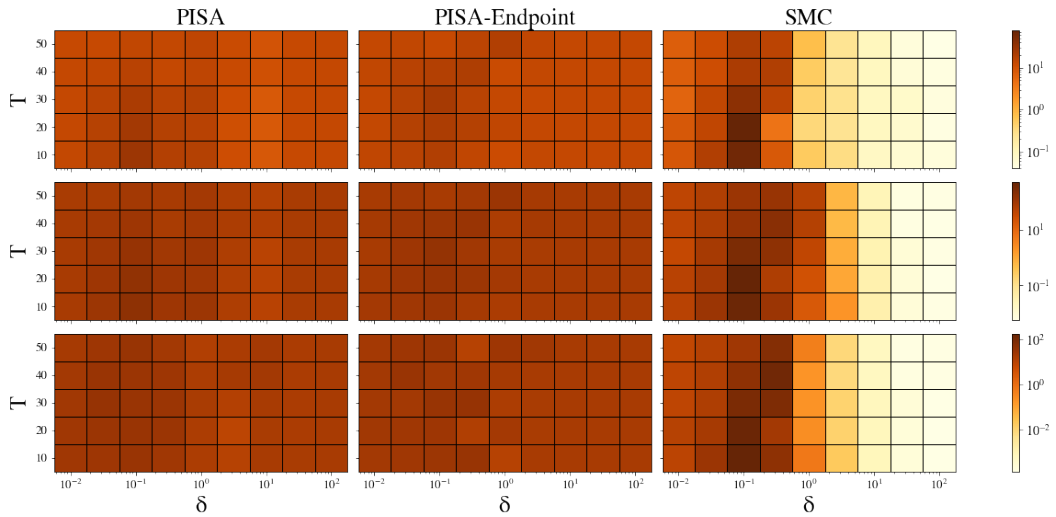

Figure 3.4: ESS for μ estimated empirically over 100 runs from estimators $\hat{h}_B$. First row: average over target mean. Second row: average over target second moments. Third row: ESS for mixed second moment of the two components.

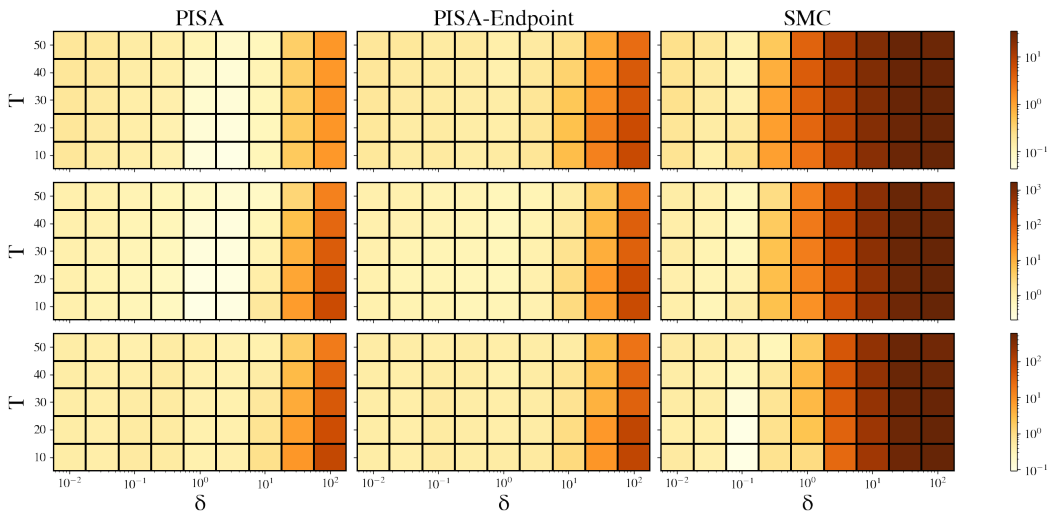

Figure 3.5: RMSE for estimators $\hat{h}_A$. First row: target mean, averaged over coordinates. Second row: target second moments, averaged over coordinates. Third row: target mixed second moment.

3.5.1.4 Assessing Root Mean Squared Error

We now study the RMSE of the estimates for the three algorithms, and just as we did in the previous section, we compute it for both $\hat{h}_A$ and $\hat{h}_B$. Figure 3.5 shows the RMSE of the posterior mean, posterior second moments and mixed second moment computed for estimates of type A. PISA reaches the smallest RMSE for the target mean and target second moments, and is also the most robust to the step size, maintaining values around 10^{-1} across all except the final two δs. Interestingly, the SMC sampler's RMSE for the mixed second moment is lower than that of the other two test functions, whereas the opposite is true for PISA/PISA-Endpoint. Figure 3.6 is a similar plot but shows the RMSE for estimates of type B. Just as it was evident for the ESS, we notice how these estimates are more robust and their RMSE is roughly constant for any choice of step size or number of integration steps.

3.5.1.5 Assessing Expected Squared Jump Distance

As mentioned in subsection 3.4.2, the two different estimates of the normalizing constant also lead to two different metrics of travelled distance. Figure 3.7 shows ESJD_A and ESJD_B for the target mean (we average across both coordinates, but notice that overall patterns are identical for each coordinate, except simply shifted by one order of magnitude of course). We make similar observations as before. The ESJD obtained with method A is more robust to the choice of δ than the SMC sampler's one, and achieves the highest overall value. The one obtained with method B is more robust to choices of T and δ, yet still achieves about the same values of ESJD as the SMC sampler.

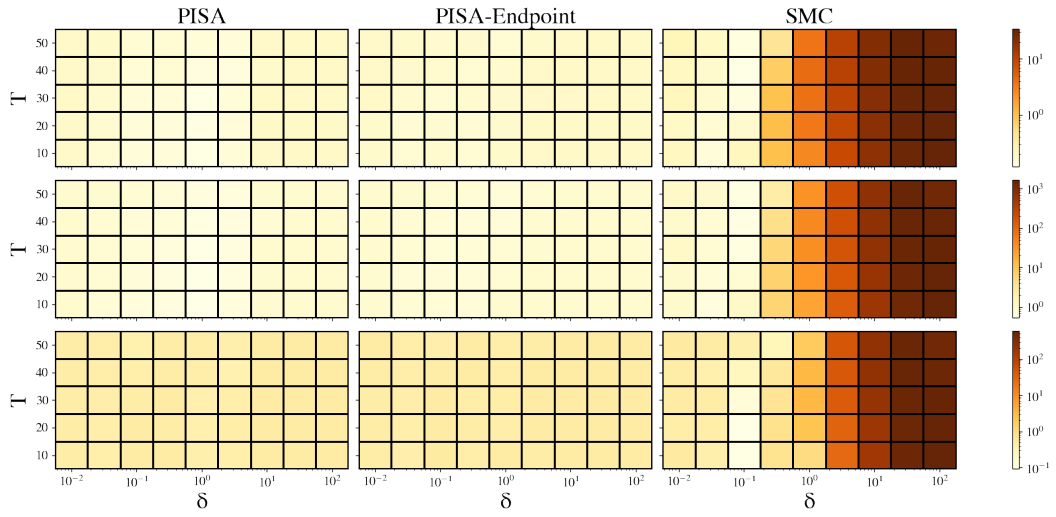

Figure 3.6: RMSE for estimators $\hat{h}_B$. First row: target mean, averaged over coordinates. Second row: target second moments, averaged over coordinates. Third row: target mixed second moments.

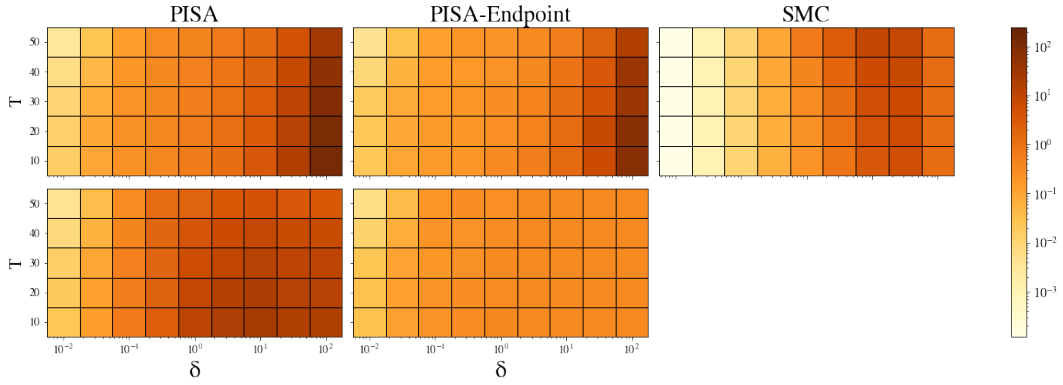

Figure 3.7: ESJD_A and ESJD_B for the target mean, averaged across components.

3.5.1.6 Assessing measures of acceptance and variance

All the plots and metrics discussed so far were computed before resampling (step 6 in Algorithm 7): they simply require the unfolded weights. In order to compute measures of acceptance such as the proportion of particles moved (pm), the median index proportion (mip), and the median path diversity (mpd), we need to perform the final resampling step for PISA and PISA-Endpoint, since we require the resampled pairs $(t^{(j)}, k^{(j)})$ described in Section 3.4. Performing the resampling step, would effectively conclude the first step of an integrator snippet and so PISA is a misnomer, since in practice these metrics are relative to Pushforward Importance Resampling. Nevertheless, we keep the acronyms the same to avoid confusion with previous results. Figure 3.9 shows pm against the acceptance probability of the SMC sampler (estimated as shown in Equation 1.30). These two metrics are comparable on the basis that they both provide a measure of acceptance

for an SMC sampler (recall that we defined pm to be the estimate of acceptance for the folded algorithm, which is an SMC sampler), however this is where the compatibility between these two metrics end. We stress that they are different metrics and we do expect them to not only achieve different values, but also to convey different information. The pm metric is really informing us about how often we actually utilise the trajectory, but of course (this is the whole point of the algorithm), we expect our algorithm to use points in the trajectory more often than the standard SMC sampler would accept a MH move. Nevertheless, we display them side-by-side as they both represent measures of acceptance. We highlight that our algorithm maintains a roughly constant acceptance probability above 90% for the first half of the set of step sizes we analysed in this experiment, and then suddenly drops to 70% at $\delta = 3.16$, then 26%, 0.04% and 0.002% for the largest step size. Of course, there is an intrinsic limit posed by the choice of the integrator, which in this case is a DW, which can be better or worse suited for a certain task (in this case, there is no reason to believe that a "random" walk exploration would be good for this problem), yet we observe that PISA is able to maintain high "acceptance" for step sizes one order of magnitude larger than SMC. In general, it seems reasonable to expect an integrator snippet to improve on metrics of distance and acceptance on a naive SMC sampler, by about the order of magnitude of the number of integration steps. Figure 3.10 shows the mpd and mip for PISA. We notice that the region of high mpd approximately captures the optimal region observed with previous metrics. The mip plot mimics the proportion of particles moved in that it maintains relatively constant until $\delta \approx 1.0$, by which points it drops steadily. We have tested using these three metrics within the integrator snippets framework to adapt integrator step sizes in the spirit of [5], and have found that all three allow for efficient adaptation. Although, as argued in subsection 3.4.1, mpd is the most complete metric, measuring both how far we travel on average along the path, and how many different paths we resample from, we have found the median index proportion to be a more intuitive (and yet with a similar performance) metric. For this reason, in the rest of the manuscript we will use this metric to adapt step sizes of integrators, and we aim to target $\text{mip}_n \approx 0.3$ since we have found this to work well in our applications and is also the average value achieved at $\delta = 1.0$ in this experiment. This value of mip produces a histogram of the resampled trajectory indices that has a consistent shape across our experiments, which we show for this toy problem in Figure 3.8. While at first one may think it would be beneficial to aim for a uniform probability of sampling along the trajectory, we can see that this strategy is not optimal based on the various metrics shown so far. This makes intuitive sense: a flat histogram suggests that the step size or the number of integration steps (or both) are too small and could be increased, and we notice that for $\delta = 0.1$ we have a histogram whose "tail" decreases steadily towards the end of the trajectory, but not so much so that the final points are very unlikely to be chosen. We have found that adapting the step size (or T) to stay roughly constant around $\text{mip}_n \approx 0.3$ across iterations leads to histograms of these type, which provide an intuitive diagnostic and adaptive tool. Details on how to adapt the step size are given in subsection 3.7.1.3, and is just an application of [5].

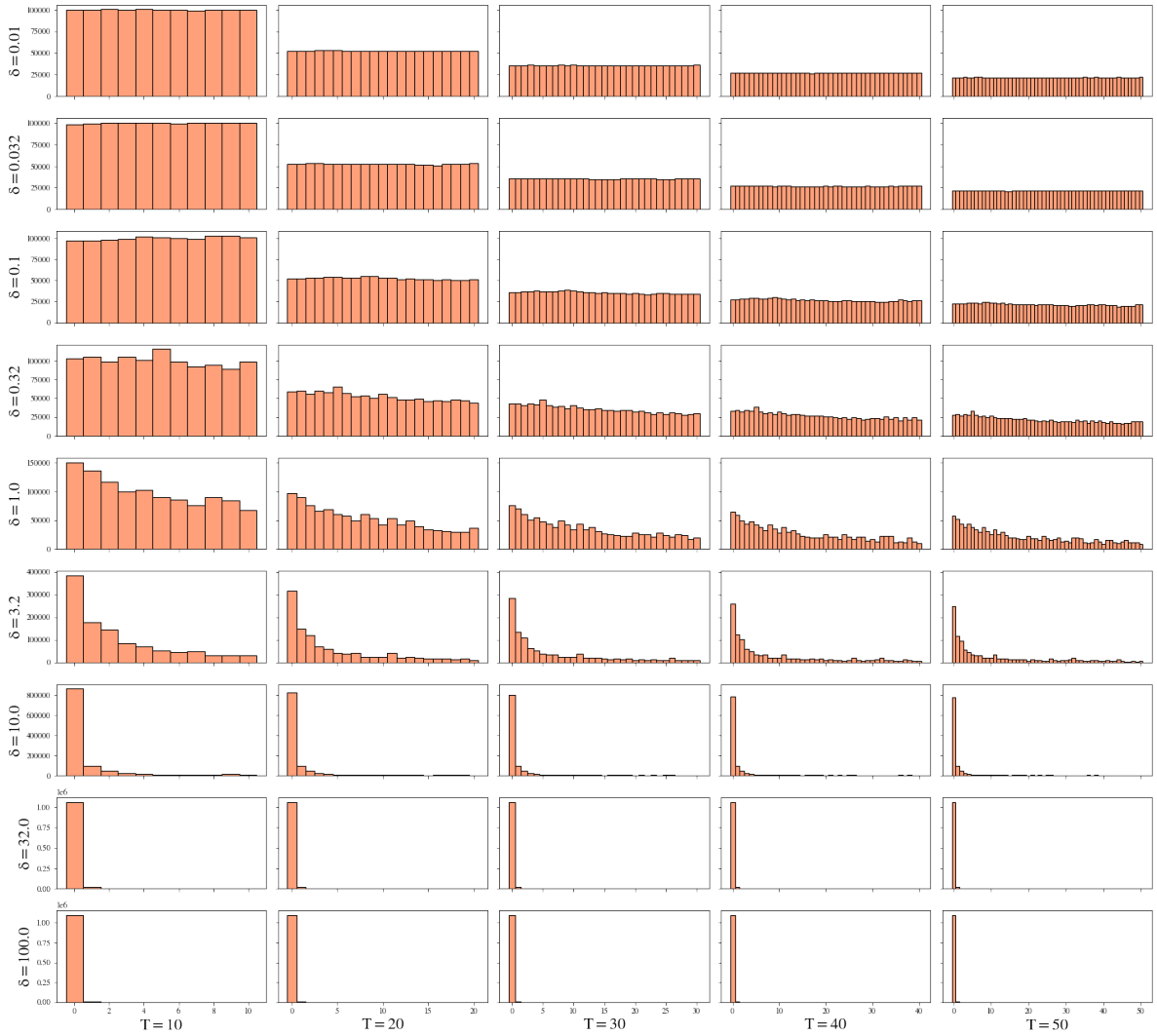

Figure 3.8: Histogram of the resampled indices as described in subsection 3.4.2.2, averaged over 100 runs of 10000 particles each. Notice that since the target is continuous, we expect the histograms to look uniform when the step size is small and decay when it starts increasing, since by definition of continuity, it won't change too fast in a small enough neighbourhood.

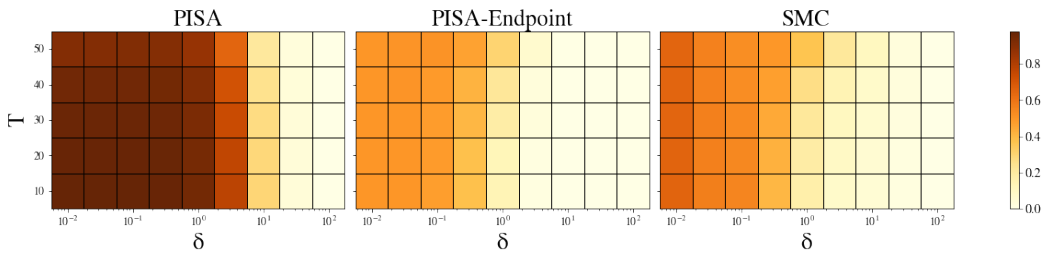

Figure 3.9: Proportion of particles moved (first two plots) and estimated acceptance probability for PISA, PISA-Endpoint, and SMC. While pm for PISA is not directly comparable with the estimated acceptance probability for the SMC sampler, the pm for PISA-Endpoint is.

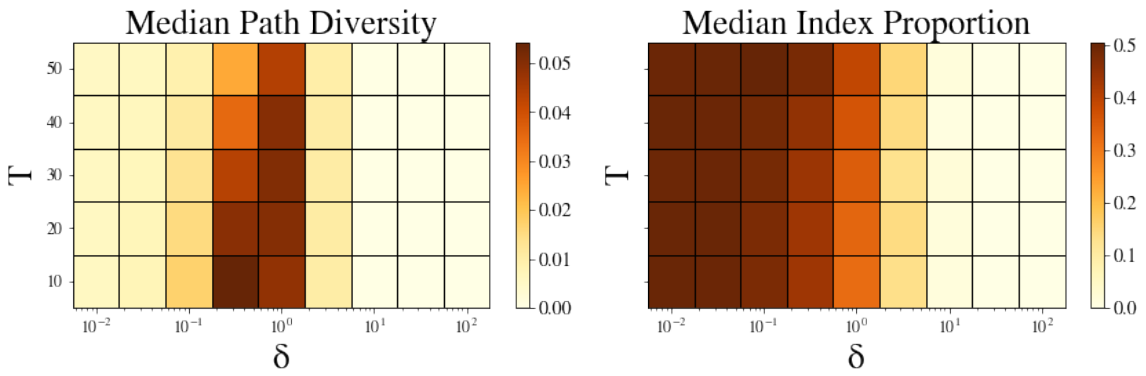

Figure 3.10: Left: Median Path Diversity for PISA. Right: Median Index Proportion for PISA.

Next, we report the Percentage of Variance Reduction (PVR) (as described in 3.3.4) obtained by sampling from $\bar{\mu}$ using the integrator snippet mechanism, for our usual three test functions (mean, variance and correlation), as shown in Figure 3.11. Firstly, we highlight how regions of higher PVR align with our usual region around $\delta = 1.0$, signaling that indeed the variance is reduced the most when the algorithm has the highest ESS and the lowest RMSE. In addition, we observe that for both PISA and PISA-Endpoint the percentage of variance reduction increases for more non-linear and variable test functions.

Finally, we show the histograms of all algorithms. Figures 3.12, 3.13, 3.14 show the histograms for the first coordinate and Figures 3.15, 3.16 and 3.17 for the second coordinate, for PISA, PISA-Endpoint and SMC respectively. The red line is the true target. For either coordinate, it is clear that the SMC sampler's performance is much more dependent on the choices of T and δ than the integrator snippet counterparts, since for any step size larger than $\delta = 1.0$, the histograms completely fail to capture the bulk of the target.

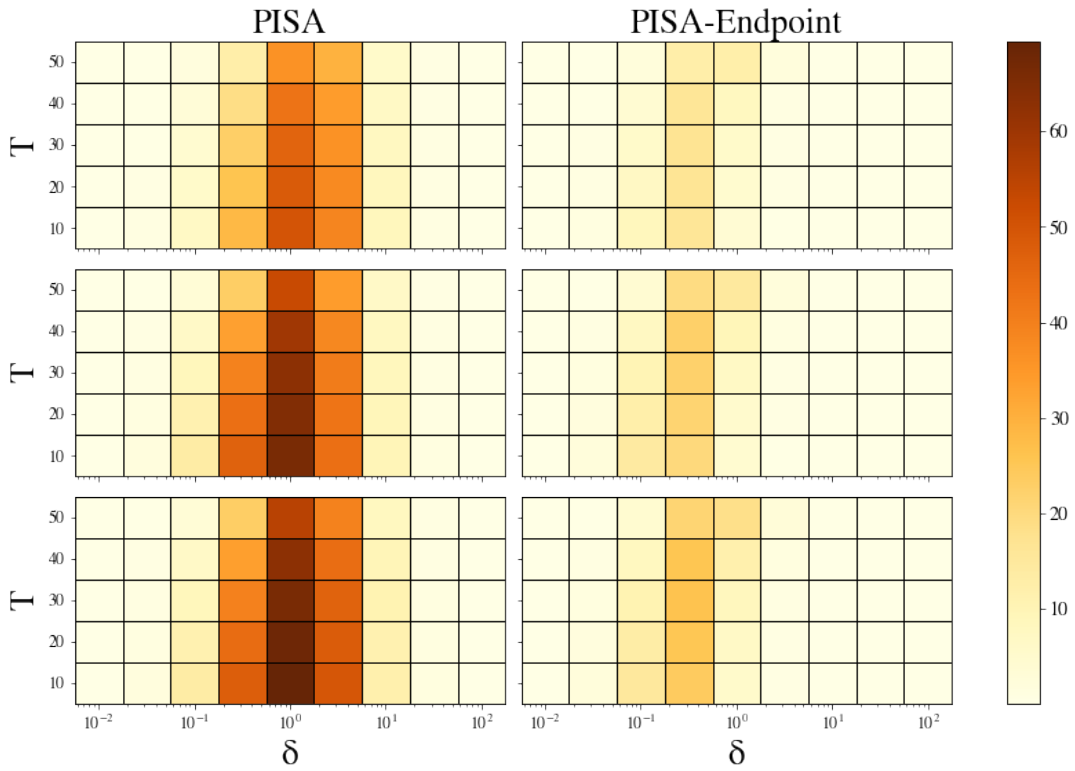

Figure 3.11: Percentage of variance reduction for three test functions: target mean, target second moments (both averaged across coordinates), and mixed second moment.

3.5.1.7 Assessing normalizing constant estimators

In section 3.3.3 we have seen how there are two estimators for integrator snippets, based on how one decides to approximate the normalizing constant. The first estimator, $\hat{h}_A$, is found by estimating the normalizing constants separately for each pushforward, similar to how one would estimate it in standard importance sampling. This leads to $T+1$ separate importance sampling estimators, and each of them is used to normalize the weights for the corresponding distribution. The second estimator, $\hat{h}_B$, is found by leveraging the fact that all pushforwards share the same normalizing constant and we approximate it by the average of the $T+1$ normalizing constants. Therefore, we have a single estimator (of the normalizing constant) of type B.

In order to understand how $\hat{h}_A$ and $\hat{h}_B$ behave, we shall begin with studying the behavior of the normalizing constant estimators. We denote by $\hat{Z}_{A,k}$ the k^{th} estimator of the normalizing constant of type A

$$\hat{Z}_{A,k} := \frac{1}{N}\sum_{i=1}^{N} \breve{w}_k(Z^{(i)}) \qquad k \in [\![0, T]\!].$$

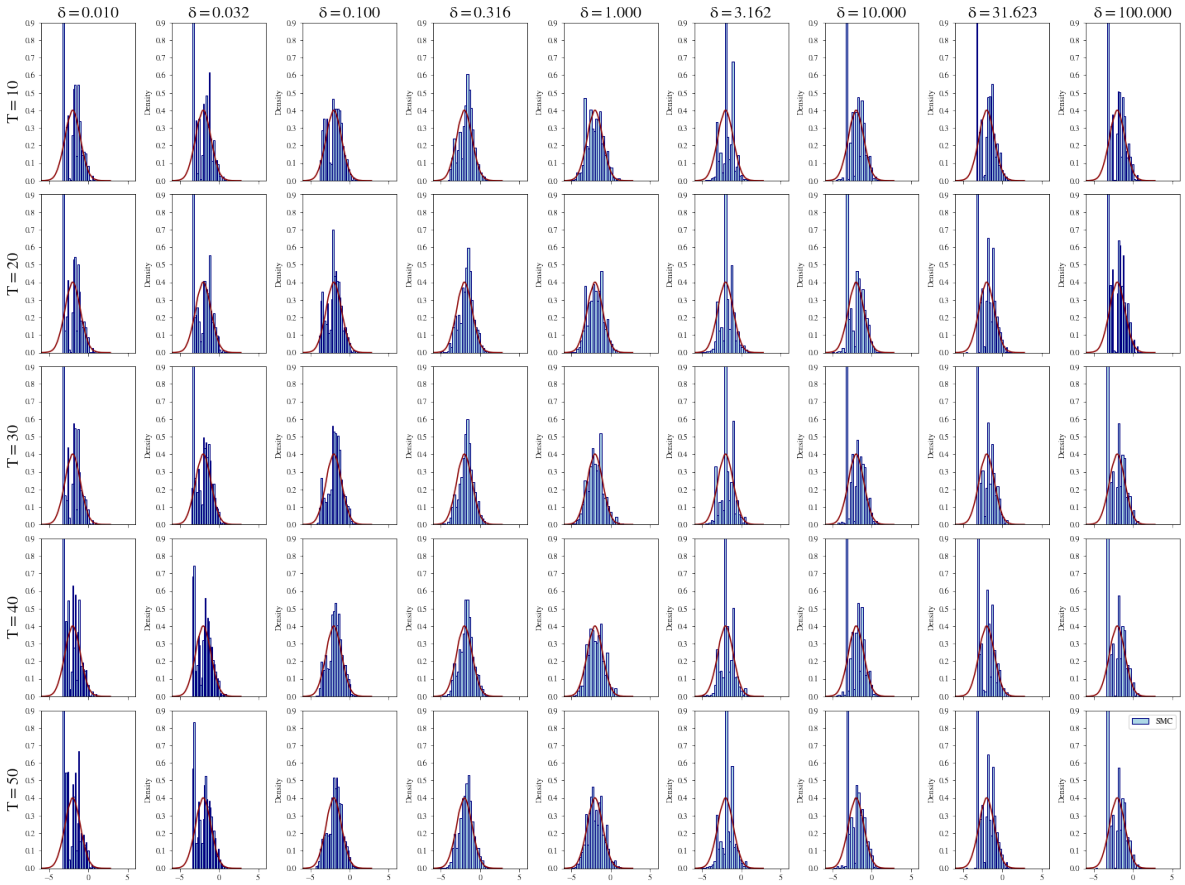

Figure 3.12: Histograms for first coordinate for PISA

Of course, the sample average of these $T+1$ estimators is the estimator of type B, denoted

$$\hat{Z}_B := \frac{1}{T+1} \sum_{k=0}^{T} \hat{Z}_{A,k}.$$

Therefore, the (unbiased) sample variance of the estimators of type A can be written as

$$\hat{\mathbb{V}}_{A,T} := \frac{1}{T} \sum_{k=0}^{T} \left(\hat{Z}_{A,k} - \hat{Z}_B\right)^2,$$

and it represents, by definition, the variability of the $T+1$ estimators of type A around the estimator of type B. This is displayed in Figure 3.18 for PISA and PISA-Endpoint. Interestingly, there doesn't seem to be an obvious dependence of this variability on the parameter T, whereas it does have a clear and strong dependence on the parameter δ. In particular, for PISA there is large variation among the estimators around $\delta = 1.0$ which then decreases for larger and smaller step sizes, whereas for PISA-Endpoint the variance does not decrease for the larger values of δ.

In this particular experiment, when δ is too large, almost the entirety of the trajectory is going to be far out into the tails of both the proposal and the target distribution (particularly

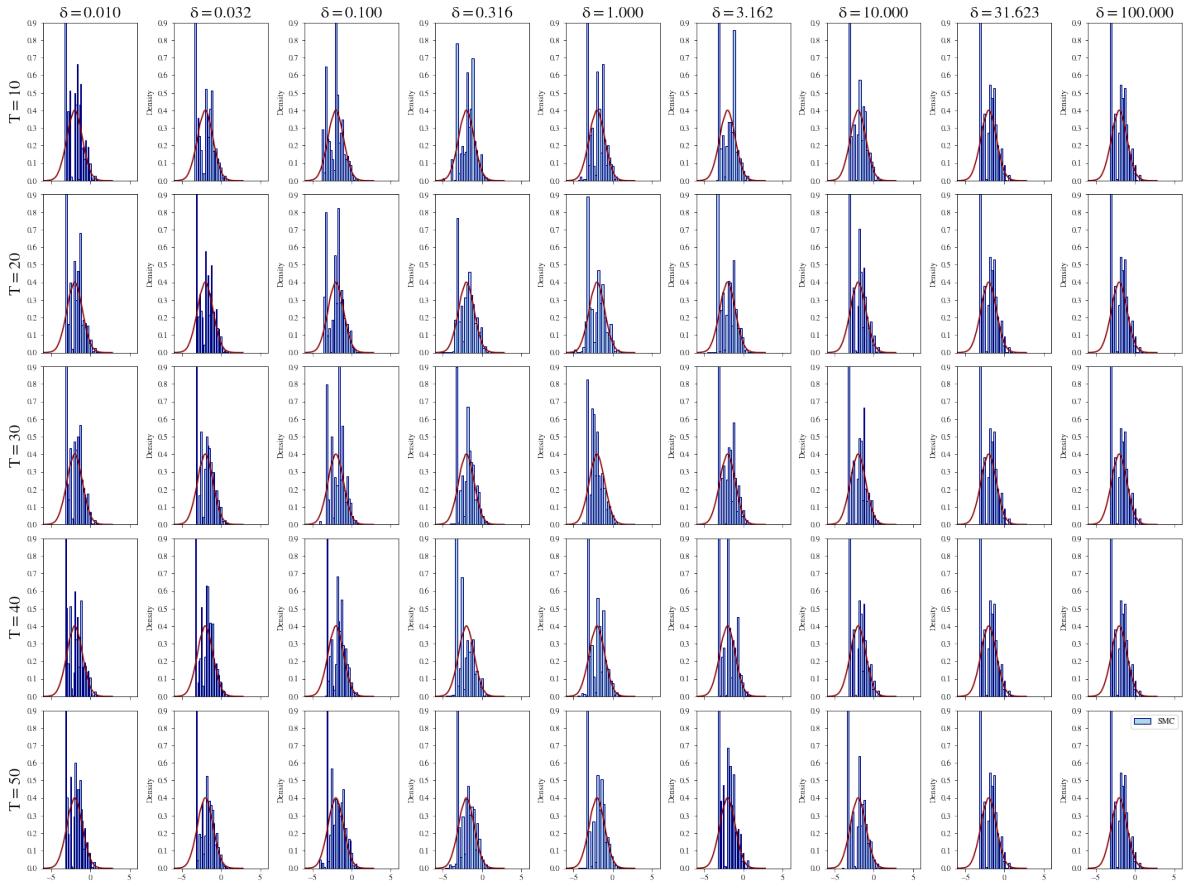

Figure 3.13: Histograms for first coordinate for PISA-Endpoint.

so because the integrator is using no information about the geometry). This means that only the very first few steps of few particles will effectively have large weights, so that most of the $T+1$ estimators will practically be zero or very close to zero, and this will have an impact on the magnitude of the variance. Thus the decrease of variance on the left-most plot of Figure 3.18 is not signalling any improvement in the efficacy of the algorithm.

Figure 3.19 shows the sample variance of the two estimators $\hat{h}_A$ and $\hat{h}_B$ for different values of T and δ for four test functions (first and second moments for each coordinate). We also compare the variance of the estimators found with the endpoint version of the algorithm. The figure indicates that for this specific example, $\hat{h}_A$ is less robust to the choice of δ but manages to achieve lower variance for a window of optimal step sizes. In contrast $\hat{h}_B$ is more stable but does not achieve as small variance. This makes intuitive sense: by averaging the normalizing constant across the trajectories the estimate is more robust when the path is too long, but at the same time it trades-off the smaller variance.

Finally, Figure 3.20 shows the bias of the estimators. The figure presents a separate colorbar for each plot due to a large variety in the magnitude of bias incurred by the different algorithms.

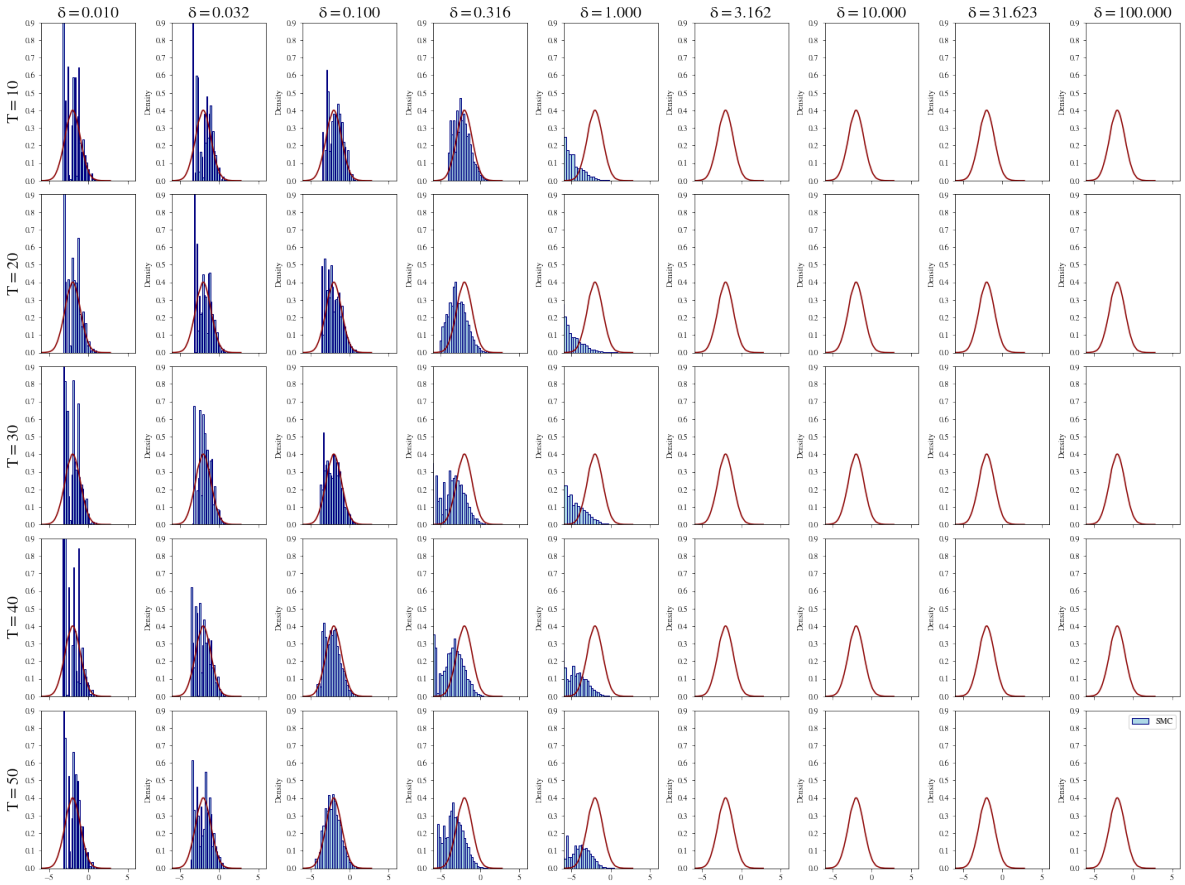

Figure 3.14: Histograms for first coordinate for SMC

The rows correspond to first and second moments for both components, respectively. In the first row we can observe that $\hat{h}_B$ presents a clear optimal region between $\delta = 1$ and $\delta = 10$, and from the colorbar we notice that the maximum value is about half that achieved by $\hat{h}_A$. The contrast is even more striking if we compare $\hat{h}_A$ Endpoint with $\hat{h}_B$ Endpoint. This behaviour repeats in the second row and becomes more evident for non-linear functions as we can see in the remaining two rows. Particularly, across the entire range of δs and Ts, $\hat{h}_B$ manages to keep a modest bias, whereas $\hat{h}_A$ can lead to values up to four orders of magnitude larger for the largest δs. Once more, this behavior mirrors what has been observed in the experiments so far: $\hat{h}_B$ provides much more robust estimates, which we attribute the fact that it is averaged over all estimates $\hat{h}_A$.

3.6 Markov Snippets

3.6.1 Constructing trajectories by iterating a single Markov kernel

Integrator Snippets use a deterministic transformation to generate trajectories, which are then interpreted as particles in an SMC sampler. Although this framework is already very

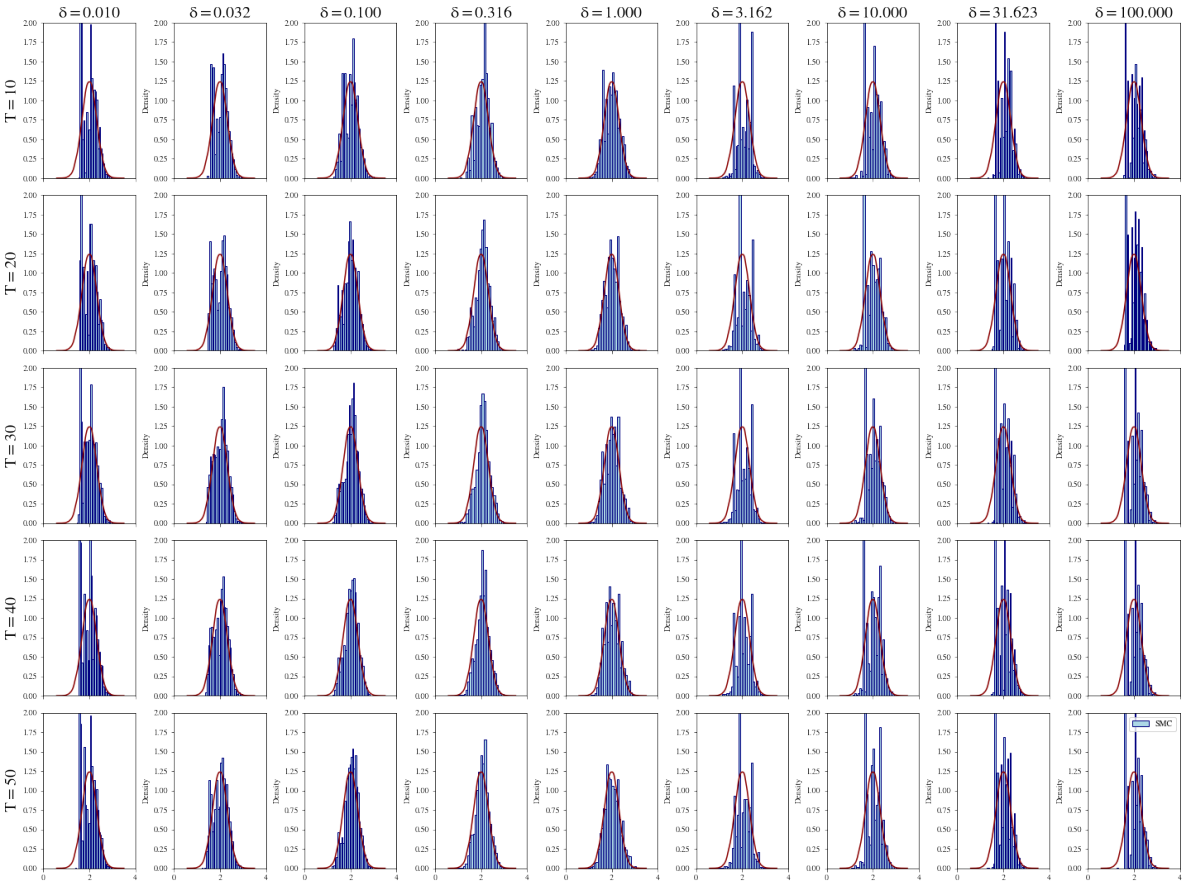

Figure 3.15: Histograms for second coordinate for PISA.

flexible, it turns out it is possible to naturally extend it to generate the trajectory stochastically, using Markov kernels, in a way that is reminiscent of Waste-Free SMC [37]. The simplest way to generate a trajectory stochastically is by iterating T times a single Markov kernel $M : \mathsf{Z} \times \mathcal{Z} \to [0,1]$ that represents the one-step update, for each particle. Given $z_0 \in \mathsf{Z}$, the trajectory is then generated according to the joint kernel

$$M^{\otimes T}(z_0, dz_{1:T}) = \prod_{i=1}^{T} M(z_{i-1}, dz_i).$$

In this work, we do not make explicit use of the Markov Snippet framework, however we highlight that PISA with the DW integrator that was tested in the previous section is technically a Markov Snippet that has been written as an Integrator Snippet by suitable choice of auxiliary variables and integrators. For simple scenarios, it might be advantageous to write down a cumbersome integrator, as we have done in the previous section, that "hides away" the stochasticity (after all, while the Deterministic Walk integrator is a deterministic update, it is fundamentally constructing a classical random walk trajectory), but for more complex scenarios, the Markov Snippet framework may be inevitable. One way to think about this, is by linking it to Metropolis-

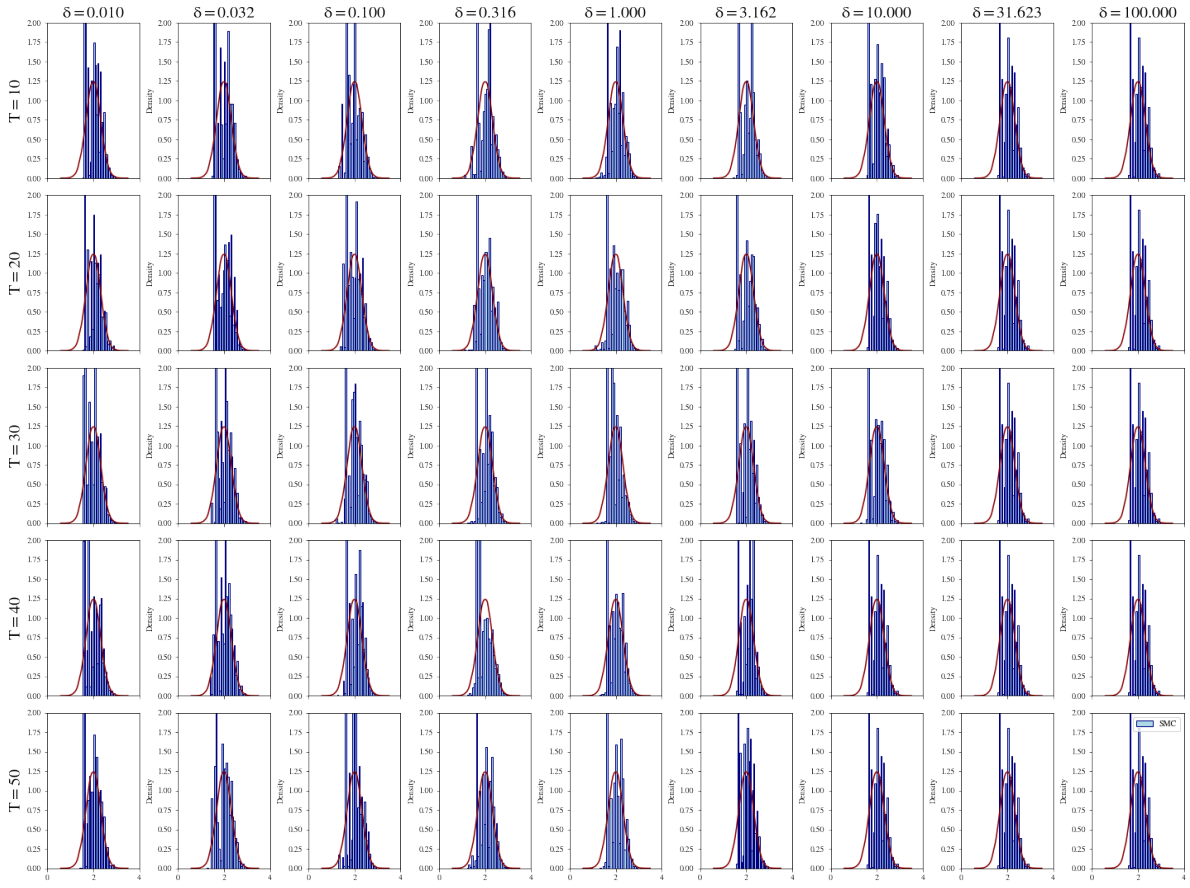

Figure 3.16: Histograms for second coordinate for PISA-Endpoint

Hastings using involutions, as introduced in subsection 1.6.2.3. Most MH updates are often most naturally conceived with stochasticity in mind (i.e. one chooses the Markov kernel Q), however the involutive framework is, instead, often the most natural way to show correctness. This seems to translate easily to Integrator Snippets, were for certain applications a Markov update may be more intuitive and natural, but re-framing it (cumbersomely) as an integrator snippet in a suitable extended space may be the fastest way to demonstrate correctness. Indeed, this will be the approach taken in subsection 3.6.3.

The next theorem, builds on Theorem D.1 where now the trajectories are constructed stochastically and the proof is in Appendix D.2.1. Theorem D.3 in the supplementary material shows how one can use Markov Snippets to estimate expectations with respect to μ analogously to the previous section. We highlight that in the following theorem, the weights $w_{\mu,k}$ and $w_{\eta,k}$ are written as functions of a vector $\mathbf{z} \in \mathsf{Z}^{T+1}$ solely to make the various expressions appearing in the statement and proof of the Theorem below easier to read, but in reality they depend only on the elements z_0 and z_k.

Theorem 3.1 (A near-optimal SMC sampler for mixtures)**.** *Let $\mu \ll \eta$ be two probability distribu-*

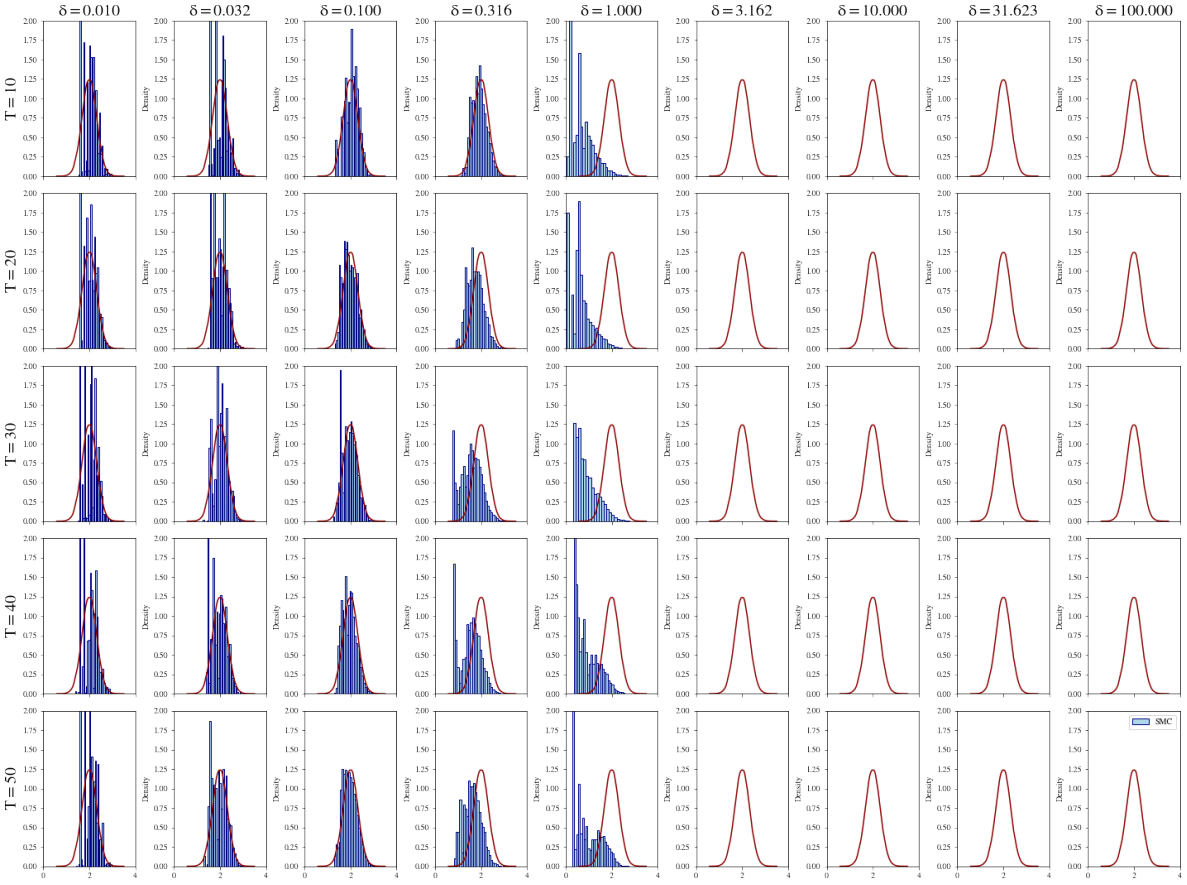

Figure 3.17: Histograms for second coordinate, for SMC.

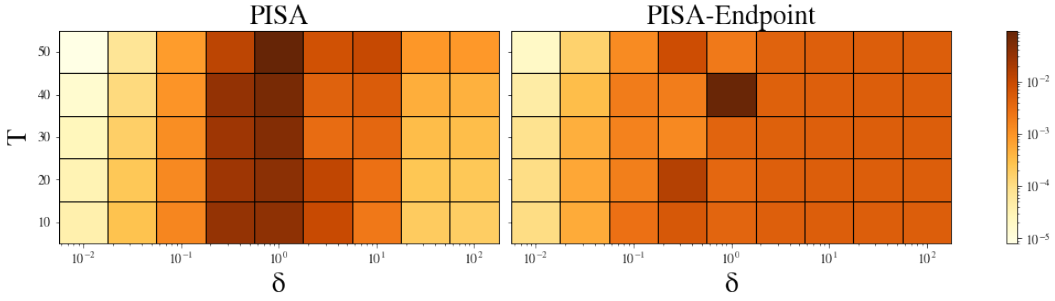

Figure 3.18: Variance of normalizing constant estimators of type A around the estimator of type B.

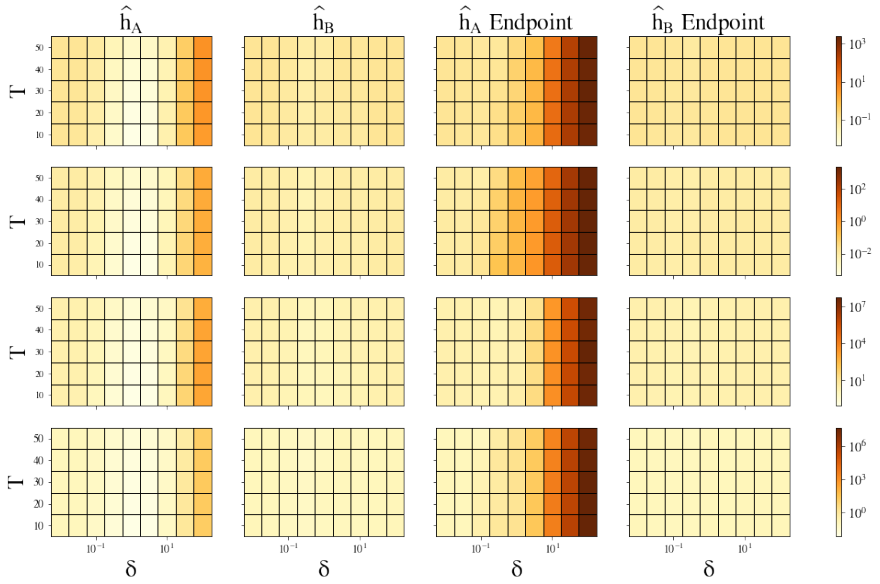

Figure 3.19: Comparison between the variance of $\hat{h}_A$ and $\hat{h}_B$. Results are averaged over 500 independent runs.

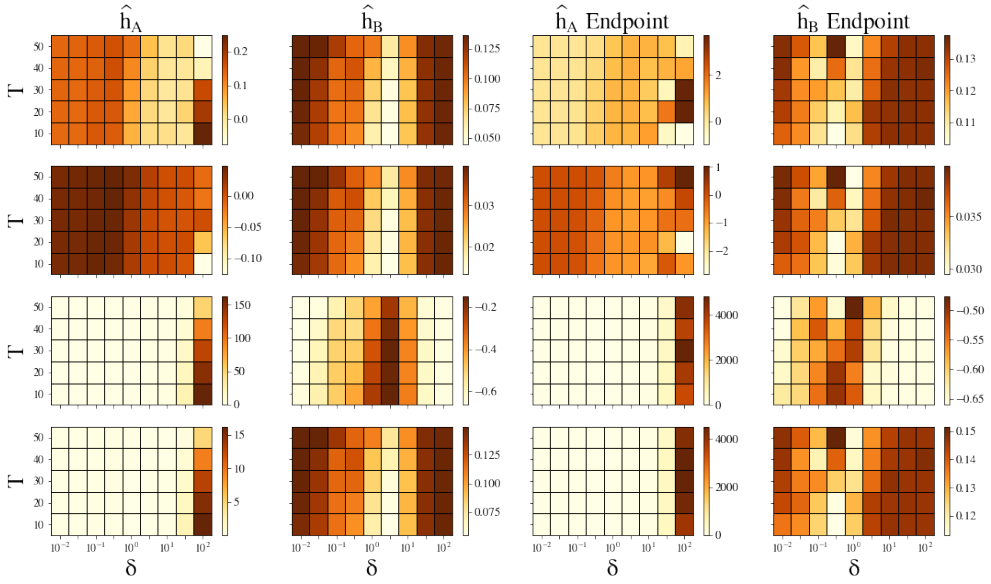

Figure 3.20: Comparison between the bias of $\hat{h}_A$ and $\hat{h}_B$ over 500 independent runs.

tions on $(\mathsf{Z},\mathscr{Z})$*, and* $M,L,R:\mathsf{Z}\times\mathscr{Z}\to[0,1]$ *be Markov kernels such that* $\mu\oplus L^k\ll\mu\otimes M^k$ *for any* $k\in\mathbb{N}$ *so that the Radon-Nikodym derivatives on* $(\mathsf{Z}^{T+1},\mathscr{Z}^{\otimes(T+1)})$

$$w_{\mu,k}(\mathbf{z})=\frac{d(\mu\oplus L^k)}{d(\mu\otimes M^k)}(\mathbf{z})\qquad\text{and}\qquad w_{\eta,k}(\mathbf{z})=\frac{d(\eta\oplus L^k)}{d(\eta\otimes M^k)}(\mathbf{z})$$

are well-defined, with $\mathbf{z}=(z_0,z_1,\ldots,z_T)$ *and* $T\in\mathbb{Z}_+$*. Define the marginal distributions*

$$\bar{\mu}(d\mathbf{z})=\left[\frac{1}{T+1}\sum_{k=0}^{T}w_{\mu,k}(\mathbf{z})\right]\mu\otimes M^{\otimes T}(d\mathbf{z})$$
$$\bar{\eta}(d\mathbf{z})=\left[\frac{1}{T+1}\sum_{k=0}^{T}w_{\eta,k}(\mathbf{z})\right]\eta\otimes M^{\otimes T}(d\mathbf{z})$$

on $(\mathsf{Z}^{T+1},\mathscr{Z}^{\otimes(T+1)})$ *as well as the Markov kernel* $\bar{M}_\eta:\mathsf{Z}^{T+1}\times\mathscr{Z}^{\otimes(T+1)}\to[0,1]$

$$\bar{M}_\eta(\mathbf{z},d\mathbf{z}')=\sum_{k=0}^{T}\frac{w_{\eta,k}(\mathbf{z})}{\sum_{\ell=0}^{T}w_{\eta,k}(\mathbf{z})}R(z_k,dz'_0)M^{\otimes T}(z'_0,dz'_{1:T}).$$

Then the near-optimal (in the sense of subsection 1.8.6.2) backward kernel

$$\bar{L}_\eta(\mathbf{z}',d\mathbf{z})=\frac{\bar{\eta}(d\mathbf{z})\bar{M}_\eta(\mathbf{z},d\mathbf{z}')}{\bar{\eta}\bar{M}_\eta(d\mathbf{z}')}$$

exists and when $\eta R=\eta$ *the incremental weights (of Equation* (1.27)*) are given by*

$$\bar{w}(\mathbf{z}')=\frac{\mu(dz'_0)}{\eta(dz'_0)}\left[\frac{1}{T+1}\sum_{k=0}^{T}\frac{\mu(dz'_k)L^k(z'_k,dz'_0)}{\mu(dz'_0)M^k(z'_0,dz'_k)}\right],$$

whenever $\bar{\mu}\otimes\bar{L}_\eta\ll\bar{\eta}\otimes\bar{M}_\eta$ *(Notice* $\bar{w}(\mathbf{z}')$ *effectively depends only on* $\mathbf{z}'$ *even though it is a function of both* $\mathbf{z}$ *and* $\mathbf{z}'$*. This is due to the choice of backward kernel, as explained in subsection 1.8.6.2).*

Finally, Lemma D.1 in Appendix D.2.2 provides an expression for the weights $w_{\mu,k}$ that can be computed under mild conditions (as long as we choose the correct backward kernel)

$$w_{\mu,k}(z_0,z_k)=\frac{\mu(z_k)}{\mu(z_0)},$$

where μ is the density of μ with respect to a suitable reference measure. This allows an easy computation of the incremental weights

$$\bar{w}(\mathbf{z}')=\frac{1}{T+1}\sum_{k=0}^{T}\frac{\mu(z'_k)}{\eta(z'_0)},$$

and to define an equivalent "unfolded" version of the Markov Snippet algorithm.

3.6.2 Integrator Snippets as a special case of Markov Snippets

Intuitively, it seems reasonable that integrator snippets are just a special case of this framework with

$$M(z, dz') = \delta_{\psi(z)}(dz')$$

for a suitable integrator ψ. It turns out that suitable here means that it is invertible (with unit absolute determinant Jacobian) and such that $\upsilon^{\psi} = \upsilon$ where υ is the reference measure, typically the Lebesgue or a product of Lebesgue and counting measures. This is justified in Appendix D.2.3.

3.6.3 Integrator Snippets with a mixture of integrators

In certain applications, it might be helpful if not all trajectories are constructed using the same integrator, and instead have a collection of them, each tailored to a specific task, and at each step and for each particle, we first choose which type of trajectory we wish to construct, and then construct it with the respective integrator. For simplicity, we shall focus on the scenario where the mixture consists only on two integrators, although our results easily generalize to any finite number of mixture components. The idea is simple: for each particle we sample $\iota \sim \text{Bernoulli}(p)$ where $p \in [0,1]$ and then use ψ_ι as our integrator. It turns out, that for each particle, the unfolded weights are exactly of the same form as $\breve{w}$ in Algorithm 7. To show correctness, we will construct an SMC sampler just like in Section 3.3 but on an extended space that includes the auxiliary variable ι, and then leverage again Lemma D.1. Consider an SMC sampler on the extended space

$$(\Gamma, \mathscr{G}) := (\mathsf{Z} \times \{0,1\}, \mathscr{Z} \otimes \mathscr{P}(\{0,1\}))$$

where $\gamma = (z, \iota) \in \Gamma$, targeting the sequence of distributions

$$\mathring{\bar{\mu}}(d\gamma_n) = \frac{1}{T+1} \sum_{k=0}^{T} \mathring{\mu}_n^{\mathring{\psi}_n^{-k}}(d\gamma_n),$$

with $\mathring{\psi}_n(\gamma) = (\psi_{n,\iota}(z_n), \iota_n)$ and $\{\psi_{n,\iota} : n \in [\![P]\!], \iota \in \{0,1\}\}$ is a sequence of integrators. We define the refreshment kernel as $\mathring{R}(\gamma, d\gamma') = R(z, dz')\rho(\iota')$ where $\rho(\iota)$ is a Bernoulli distribution (we shall assume the parameter of this distribution is constant across iterations, but again, this can be easily generalised). If we define everything as in Section 3.3, the validity of the algorithm (we will find an expression for the weights after showing correctness) depends on the assumptions of Lemma D.1, which requires $\mathring{\mu}_{n-1}\mathring{R}_n = \mathring{\mu}_{n-1}$ and $\mathring{\bar{\mu}}_n \ll \mathring{\mu}_{n-1}$. The first condition is straightforward assuming $\mu_{n-1}R_n = \mu_{n-1}$

$$\mathring{\mu}_{n-1}\mathring{R}_n(d\gamma) = \left[\int \mu_{n-1}(d\tilde{z}) R_n(\tilde{z}, dz)\right] \left[\sum_{\iota'=0,1} \rho(\iota')\right] \rho(\iota) = \mathring{\mu}_{n-1}(d\gamma).$$

For the second condition, assuming $\bar{\mu}_{n,\iota} \ll \mu_{n-1}$ for $\iota = 0, 1$ where $\bar{\mu}_{n,\iota}$ is the mixture with integrator $\psi_{n,\iota}$, we only need to show that for any measurable rectangle $\mathsf{B} \times \mathsf{C}$ with $\mathsf{B} \in \mathscr{Z}$ and $\mathsf{C} \in \mathscr{P}(\{0,1\})$,

$\mathring{\mu}_{n-1}(\mathsf{B}\times\mathsf{C})=0$ implies $\mathring{\bar{\mu}}_n(\mathsf{B}\times\mathsf{C})=0$. This follows immediately from

$$\mathring{\bar{\mu}}_n(\mathsf{B}\times\mathsf{C})=\rho(C)\left[\frac{1}{T+1}\sum_{k=0}^{T}\mu_n^{\psi_{n,\iota}^{-k}}(\mathsf{B})\right]=\rho(\mathsf{C})\bar{\mu}_{n,\iota}(\mathsf{B}).$$

In this extended space, the SMC incremental weights (i.e. the folded weights) become

$$\mathring{w}_n(\gamma_n)=\frac{1}{T+1}\sum_{k=0}^{T}\frac{\mathring{\mu}_n^{\mathring{\psi}_n^{-k}}(d\gamma_n)}{\mathring{\mu}_{n-1}(d\gamma_n)}=\frac{1}{T+1}\sum_{k=0}^{T}\frac{\mu_n^{\psi_{n,\iota n}^{-k}}(dz_n)}{\mu_{n-1}(dz_n)},$$

and therefore the unfolded weights would be the promised

$$\check{w}_{n,k,\iota}(z)=\frac{\mu_n(\psi_{n,\iota}^k(z))}{\mu_{n-1}(z)}\qquad \iota=0,1.$$

In Appendix D.2.4 we consider the more general case of a mixture of Markov kernels.

3.6.4 A Markov Snippet for Filamentary Distributions

3.6.4.1 Task description

Consider spaces $(\mathsf{X},\mathscr{X})=(\mathbb{R}^n,\mathscr{B}(\mathbb{R}^n))$ and $(\mathsf{V},\mathscr{V})=(\mathbb{R}^n,\mathscr{B}(\mathbb{R}^n))$ with associated Lebesgue measures $\mathrm{Leb}_{\mathsf{X}}$ and $\mathrm{Leb}_{\mathsf{V}}$, and let $(\mathsf{Z},\mathscr{Z})=(\mathsf{X}\times\mathsf{V},\mathscr{X}\otimes\mathscr{V})$ be their product space, with product Lebesgue measure $\mathrm{Leb}_{\mathsf{Z}}$. Suppose $f:\mathbb{R}^n\to\mathbb{R}^m$, with $n>m$ is Lipschitz continuous and similar to Chapter 2 define, for any $\epsilon>0$, the following probability distribution on $(\mathsf{X},\mathscr{X})$

$$\eta_\epsilon(dx)=\frac{v(x)k_\epsilon(f(x))}{\mathscr{Z}_{\eta_\epsilon}}\mathrm{Leb}_{\mathsf{X}}(dx),$$

where k_ϵ is an approximation to the identity, $v(dx)$ is a base probability distribution on $(\mathsf{X},\mathscr{X})$ with density with respect to $\mathrm{Leb}_{\mathsf{X}}$ given by $v(x)$. In the expression above, $\mathscr{Z}_{\eta_\epsilon}$ is the normalizing constant of the density of η_ϵ (with respect to $\mathrm{Leb}_{\mathsf{X}}$). Our aim is to sample from η_ϵ for $\epsilon>0$ as small as possible. This is, for small $\epsilon>0$, a filamentary distribution concentrated around the manifold

$$\mathscr{M}:=\{x\in\mathsf{X}:f(x)=0\}.$$

In this section we design a Markov snippet that is well-suited to this task. In the notation of this Chapter, our extended target distribution on $(\mathsf{Z},\mathscr{Z})$ is

$$\mu_\epsilon(dz)=\eta_\epsilon\otimes\varpi(d(x,v)),$$

where ϖ is a Gaussian measure on $(\mathsf{V},\mathscr{V})$. We will sample from μ_ϵ sequentially for a sequence of tolerances found adaptively, and for brevity we will write $\mu_n:=\mu_{\epsilon_n}$ for $n\in[\![P]\!]$, with $P\in\mathbb{Z}_+$, and similarly for η_{ϵ_n} and k_{ϵ_n}. By construction, the filamentary distribution η_ϵ is such that at any point with positive density, the variability in the directions tangential to $\mathscr{M}$ is much higher than the variability in the directions normal to it. Nonetheless, for ergodicity, it is important to explore

the perpendicular directions just as well as the tangential ones, even if they have a much smaller scaling. As we have seen in Chapter 2, general purpose samplers are unable to deal with this difference in scaling effectively. In this section, we devise a Markov Snippet whose kernel is a mixture of two integrators: one that is well-adapted at travelling far in tangential directions, and one that is able to explore the normal directions. For the first integrator we choose THUG (from Section 2.3.2) and for the second, we develop a new integrator in the next subsection.

3.6.4.2 Normal Hug: An integrator to explore normal directions

The NHUG integrator is very similar to the THUG integrator, except that we negate the velocity after reflecting it. As a result, while THUG flips the component of the velocity normal to $\mathscr{M}$, NHUG flips the component tangential to it and thus performs a step in the perpendicular directions of $\mathscr{M}$. More precisely, suppose that we are currently at $(x_0, v_0) \in \mathsf{Z}$, then one step of THUG and NHUG lead to the following position updates respectively

$$x_1^{\text{THUG}} = x_0 + \delta \mathrm{T}_{\frac{1}{2}} v_0$$
$$x_1^{\text{NHUG}} = x_0 + \delta \mathrm{N}_{\frac{1}{2}} v_0,$$

where $\mathrm{T}_{1/2}$ and $\mathrm{N}_{\frac{1}{2}}$ are the projection matrices for the tangent space and normal space at $x_{1/2} = x_0 + (\delta/2)v_0$. This representation makes it explicit that THUG performs a step in the tangential direction (at the midpoint) and NHUG performs one in the normal direction.

The NHUG integrator iterated for T steps is described in Algorithm 8. Figure 3.21 display sample trajectories constructed using the THUG and NHUG integrators. This particular trajectory moves closer to the manifold, however in general one cannot guarantee that and the update will simply be approximately on the normal space.

Algorithm 8: NHUG Integrator

for $b = 1, \ldots, T$ **do**
  Move: $x \longleftarrow x + (\delta/2)v$
  Bounce: $v \longleftarrow -v + 2\text{LinearProjection}(\mathrm{J}_x, v)$
  Move: $x \longleftarrow x + (\delta/2)v$
end

We will refer to Algorithm 8 as the NHUG integrator, and when used with a MH step we shall refer to the corresponding algorithm as NHUG kernel or simply NHUG-MH. We recall that the LinearProjection function was defined in Chapter 2.

3.6.4.3 Sequential Monte Carlo with a mixture of THUG and NHUG

It is natural to construct an SMC sampler using these two kernels and we describe it here as it will provide with our baseline competitor against the Markov Snippet. The algorithm outlined

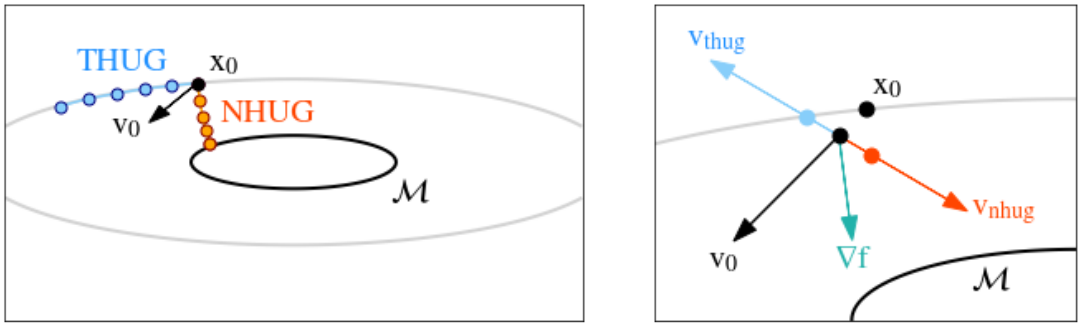

Figure 3.21: Left: THUG and NHUG trajectories generated from the same initial point $z_0 = (x_0, v_0)$. Right: Illustration of how NHUG produces points along the direction of the gradient at the midpoint. The light-blue point corresponds to $x_0 + \delta \mathrm{T}_{1/2} v_0$ and the red point corresponds to $x_0 + \delta \mathrm{N}_{1/2} v_0$, the midpoint is represented in black, but is unlabelled to avoid clutter.

below works on the space $(\mathsf{X}, \mathscr{X})$ directly and therefore all kernels and targets will be on this space: velocity variables are marginalised out. Consider an SMC sampler with forward kernel (we drop the index n for simplicity) defined as follows

$$M(x, dx') = \sum_{\iota=0,1} p_{\text{thug}}^{\iota} (1 - p_{\text{thug}})^{1-\iota} \sum_{\iota'=0,1} \delta_{\iota\iota'} M_{\iota}^{T}(x, dx'), \tag{3.22}$$

where $p_{\text{thug}} \in (0, 1]$ is the probability of choosing the THUG update, $\delta_{\iota\iota'}$ is the Kronecker delta. The kernels $M_{\iota}^{T} : \mathsf{X} \times \mathscr{X} \to [0, 1]$ are T-step THUG or NHUG Metropolis-Hastings kernels, meaning that they are of the following form

$$M_{\iota}^{T}(x_0, dx_T) = \int_{\mathsf{X}^{T-1}} \prod_{k=1}^{T} M_{\iota}(x_{k-1}, dx_k),$$

with $M_0(x_{k-1}, dx_k)$ and $M_1(x_{k-1}, dx_k)$ being the one-step THUG and NHUG MH kernels, i.e. each of these kernels does one bounce and then performs the MH accept-reject step. Additionally, we define the backward kernels to be the corresponding reversal kernels

$$L(x', dx) = \sum_{\iota'=0,1} p_{\text{thug}}^{\iota'} (1 - p_{\text{thug}})^{1-\iota'} \sum_{\iota=0,1} \delta_{\iota'\iota} L_{\iota'}^{T}(x', dx),$$

where for symmetry we choose L_0 and L_1 to be the reversals of M_0 and M_1. Behind this seemingly complicated mathematical description, is an intuitive algorithm - for each particle we sample an auxiliary variable $\iota \sim \text{Bernoulli}(p_{\text{thug}})$ and use this to decide whether to perform T MH steps of THUG or NHUG. The forward kernel in Equation (3.22) is the marginal kernel on the space $(\mathsf{X}, \mathscr{X})$. We have chosen this formulation for a simple reason: at an implementation level, one does not need to work in the joint space (z, ι) since $\delta_{\iota\iota'}$ makes sure that every particle performs T steps using the same kernel, either THUG or NHUG but never both. The resulting SMC sampler's correctness can be readily established. Since at each iteration $n \in [\![P]\!]$, $M_{\iota,n}$ are MH kernels

leaving $\eta_n(dx)$ invariant, we know from subsection 1.8.6.3 that $(\eta_n, M_{\iota,n}, L_{\iota,n-1})$ are reversible triplets for $\iota = 0,1$. Notice that we have said nothing about the cross triplets $(\eta_n, M_{0,n}, L_{1,n-1})$ and $(\eta_n, M_{1,n}, L_{0,n-1})$, since this is not necessary. First, notice that iterating the kernels $M_{\iota,n}$ and $L_{\iota,n-1}$ we produce reversible triplets

$$\eta_n(dx_{n-1})M_{\iota,n}^T(x_{n-1}, dx_n) = \eta_n(dx_n)L_{\iota,n-1}^T(x_n, dx_{n-1})$$

due to the linearity of integration, and by repeatedly applying the definition of reversible triplet. Then, it's a simple matter of algebra

$$\begin{aligned}
\eta_n(dx_{n-1})M_n(x_{n-1}, dx_n) &= \sum_{\iota=0,1} p_{\text{thug}}^{\iota}(1-p_{\text{thug}})^{1-\iota} \sum_{\iota'=0,1} \delta_{\iota\iota'}\eta_n(dx_{n-1})M_{\iota,n}^T(x, dx') \\
&= \sum_{\iota=0,1} p_{\text{thug}}^{\iota}(1-p_{\text{thug}})^{1-\iota} \sum_{\iota'=0,1} \delta_{\iota\iota'}\eta_n(dx_n)L_{\iota',n-1}^T(x', dx) \\
&= \eta_n(dx_n) \sum_{\iota'=0,1} p_{\text{thug}}^{\iota'}(1-p_{\text{thug}})^{1-\iota'} \sum_{\iota=0,1} \delta_{\iota'\iota}L_{\iota',n-1}^T(x', dx) \\
&= \eta_n(dx_n)L_{n-1}(x_n, dx_{n-1}),
\end{aligned}$$

showing that (η_n, M_n, L_{n-1}) is a reversible triplet. As a consequence the SMC incremental weights become just as in Equation (1.28)

$$w_n(x_{n-1}, x_n) = \frac{\eta_n(dx_n)}{\eta_{n-1}(dx_{n-1}} \propto \frac{v(x_n)k_n(f(x_n))}{v(x_{n-1})k_{n-1}(f(x_{n-1}))}.$$

3.6.4.4 GHUMS - A Markov Snippet with a mixture of THUG and NHUG

The SMC sampler described in the previous subsection will provide us with a baseline sampler for filamentary distributions. In this subsection, we detail an integrator snippet using a mixture of integrators (as described in subsection 3.6.3) with a similar flavour. The SMC sampler updates particles either using B steps of THUG or B steps of NHUG. It is natural to construct an Integrator Snippet which for each particle, similar to the SMC sampler, samples $\iota \sim \text{Bernoulli}(p_{\text{thug}})$ and uses this to decide whether to construct a trajectory with THUG or NHUG. Our proposed algorithm, Gibbs Hug Markov Snippet (GHUMS), is a special case of the class of Integrator Snippets described in subsection 3.6.3. More precisely, let $p_{\text{thug}} \in (0,1]$ be the probability of choosing the THUG integrator over the NHUG one, then we set $\psi_{1,n}$ and $\psi_{0,n}$ to their integrators respectively. Notice the simplicity of this framework: when evaluating the weights, we do not need to know which integrator generated the particle and all weights can naturally be combined together, since the weights are simply

$$\breve{w}_{n,k}^{(i)} = \frac{\mu_n(Z_{n-1,k}^{(i)})}{\mu_{n-1}(Z_{n-1}^{(i)})}.$$

The sample $\iota^{(i)}$ is only used to determine which integrator generates points along the trajectory, but it is not needed for weights computations or resampling, which means that one can effectively

work in the marginal space $(\mathsf{Z}, \mathscr{Z})$. Our proposed sampler for filamentary distributions is detailed in Algorithm 9.

Algorithm 9: Gibbs Hug Markov Snippet

INPUT : Number of particles $N \in \mathbb{Z}_+$, number of iterations $P \in \mathbb{Z}_+$, integration time $T \in \mathbb{Z}_+$, targets $\mu_n = \pi_n \otimes \varpi_n$ for $n \in [\![0, P]\!]$, integrators $\{\psi_{\iota,n} : n \in [\![P]\!], \iota \in \{0,1\}\}$, $p_{\text{thug}} \in (0,1)$.

Sample initial particles $Z_0^{(i)} \sim \mu_0$.
for $n = 1, \ldots, P$ **do**
  for $i = 1, \ldots, N$ **do**
    Sample $\iota^{(i)} \sim \text{Bernoulli}(p_{\text{thug}})$.
    Compute trajectory $\mathbf{Z}_{n-1}^{(i)} = (Z_{n-1}^{(i)}, Z_{n-1,1}^{(i)}, \ldots, Z_{n-1,T}^{(i)})$ where $Z_{n-1,k}^{(i)} = \psi_{\iota,n}^k(Z_{n-1}^{(i)})$.
  end
  Resample N particles $\{\bar{Z}_n^{(i)} : i \in [\![N]\!]\}$ from these $N \times (T+1)$ using weights

$$\breve{W}_{n,k}^{(i)} = \frac{\breve{w}_{n,k}^{(i)}}{\sum_{j=1}^{N} \sum_{\ell=0}^{T} \breve{w}_{n,\ell}^{(j)}} \qquad \text{where} \qquad \breve{w}_{n,k}^{(i)} := \frac{\mu_n(Z_{n-1,k}^{(i)})}{\mu_{n-1}(Z_{n-1}^{(i)})}$$

  and set $Z_n^{(i)} = (\bar{X}_n^{(i)}, V_n^{(i)})$ where $V_n^{(i)} \sim \varpi_n$ for $i \in [\![N]\!]$.
end
Return particles $\left\{\left(Z_n^{(i)}, 1\right) : i \in [\![N]\!]\right\}$.

3.6.4.5 A gradient-based alternative to MALA

We briefly digress from the rest of the thesis to present a simple idea that naturally stems from the NHUG mechanism. By construction, the NHUG integrator is unitary in the velocity component just as THUG is, meaning that the norm of the velocity remains constant along the trajectory. In other words, if R is the orthogonal matrix representing the NHUG bounce, then

$$\langle \mathrm{R}v, \mathrm{R}w \rangle = \langle v, w \rangle \qquad \forall\, v, w \in \mathsf{V} \implies \|\mathrm{R}v\| = \|v\| \qquad \forall\, v \in \mathsf{V}.$$

This suggests using the NHUG update within a MH algorithm in a manner akin to Metropolis-Adjusted Langevin Algorithm (MALA) or the HOP kernel [94]. When used in this context, we point out that this makes NHUG remarkably special in that it is the first update that we are aware of that produces a new point in a gradient direction (albeit the gradient at the midpoint) without apparent trade-off in the acceptance ratio in the form of the discrepancy between the norms of the gradient of the log-density at the current and proposed point. More precisely, consider a MALA algorithm with step size $\delta > 0$ targeting a distribution π (we use this symbol but it bears no relationship with the previous subsections). If x is the current state of the Markov

Chain, the MALA proposal is sampled from

$$x' \sim \mathcal{N}\left(dx' \middle| x + \frac{\delta}{2}\nabla \log \pi(x), \delta \mathrm{I}_d\right),$$

then the MH acceptance ratio becomes

$$a_{\mathrm{MALA}}(x', x) = \min\left\{1, \frac{\pi(x')}{\pi(x)} \cdot \frac{\mathcal{N}(x \mid x' + \frac{\delta}{2}\nabla \log \pi(x'))}{\mathcal{N}(x' \mid x + \frac{\delta}{2}\nabla \log \pi(x))}\right\}.$$

The ratio of normal densities corrects for the fact that the update is not symmetric, and the gradient at x and at x' will most likely be different. The correction term reduces to

$$\mathscr{C}_{\mathrm{MALA}}(x', x) = \exp\left(-\frac{1}{2\delta}\left[\frac{\delta^2}{4}\left\{\|\nabla \log \pi(x)\|^2 - \|\nabla \log \pi(x')\|^2\right\} + \delta(x - x')^\top \left\{\nabla \log \pi(x) + \nabla \log \pi(x')\right\}\right]\right),$$

and clearly depends on the discrepancy between the norm of the gradients of the log-density at x and x'. The larger this discrepancy, the smaller the acceptance probability. Similarly, the HOP kernel [94] samples the proposal as follows

$$x' \sim \mathcal{N}\left(dx' \middle| x, \frac{1}{\|\nabla \log \pi(x)\|^2}\left\{\mu^2 \mathrm{I}_d + (\lambda^2 - \mu^2)\frac{\nabla \log \pi(x) \nabla \log \pi(x)^\top}{\|\nabla \log \pi(x)\|^2}\right\}\right),$$

where $\lambda/\|\nabla \log \pi(x)\|$ is the scaling along the gradient direction and $\mu/\|\nabla \log \pi(x)\|$ is the scaling in the other directions, and requires $\mu > 0$. The correction term in this case

$$\begin{aligned}\mathscr{C}_{\mathrm{HOP}} = \frac{\|\nabla \log \pi(x')\|^d}{\|\nabla \log \pi(x)\|^d} \exp\Big\{&-\frac{1}{2\mu^2}\|x' - x\|^2\left(\|\nabla \log \pi(x')\|^2 - \|\nabla \log \pi(x)\|^2\right) \\ &- \frac{\mu^2 - \lambda^2}{2\lambda^2\mu^2}\left[(x' - x)^\top\left(\nabla \log \pi(x') - \nabla \log \pi(x)\right)\right]^2\Big\},\end{aligned}$$

and we notice that this also depends on $\|\nabla \log \pi(x')\|^2 - \|\nabla \log \pi(x)\|^2$. In contrast, one could use the NHUG mechanism to propose a new point

$$x' = x + \delta \frac{\nabla \log \pi(x + (\delta/2)v) \nabla \log \pi(x + (\delta/2)v)^\top v}{\|\nabla \log \pi(x + (\delta/2)v)\|^2} \qquad \text{where} \quad v \sim \mathcal{N}(0_d, \mathrm{I}_d)$$

and there would be no correction, i.e. $\mathscr{C}_{\mathrm{NHUG}} = 1$, so that the acceptance ratio would simply be the ratio $\pi(x')/\pi(x)$. Of course, the larger δ, the more different $\nabla \log \pi(x + (\delta/2)v)$ will be from $\nabla \log \pi(x)$. Both MALA and the HOP kernel utilise the gradient at the current point of the Markov chain, which means they have to correct for the mismatch with the gradient at the final point. NHUG uses a symmetric update in the direction of the gradient at the midpoint, which means that there is no correction term in the MH ratio, but it is not clear if this could bring advantages in practice or not. We do not explore this algorithm further in this thesis and leave it to future work.

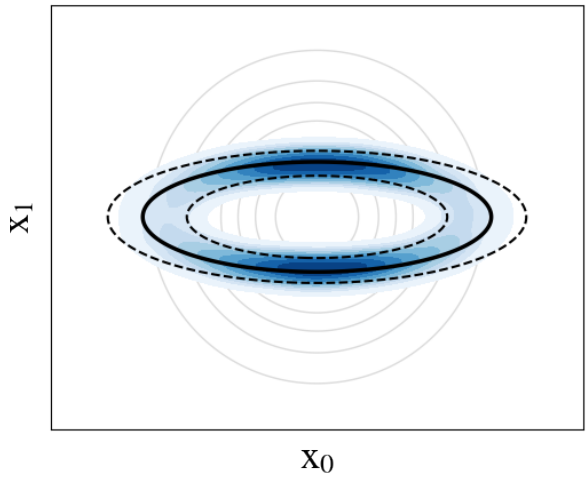

Figure 3.22: Kernel Density Estimate of 10000 samples from the target for $\epsilon = 1$ obtained with GHUMS. Light gray lines represent the contours of v, the black solid line is the manifold, and the dashed lines represent the bounds enforced by the uniform kernel

3.7 Numerical Experiments on GHUMS

3.7.1 2-dimensional Ellipse

3.7.1.1 Reproducibility Details

Once more, to test the ideas of this chapter, we start with a simple 2-dimensional example. On $(\mathbb{R}^2, \mathscr{B}(\mathbb{R}^2))$ let v be an isotropic Gaussian distribution $v(dx) = \mathcal{N}(dx \mid 0_2, \mathrm{I}_2)$ that we wish to concentrate around an ellipse using a uniform kernel. The ellipse we consider is a contour of an anisotropic bivariate normal centered at the origin and with covariance matrix $\Sigma = \mathrm{diag}(1, 0.1)$. Given a density value c of this anisotropic Gaussian, our constraint function is (with $x = (x_0, x_1)$)

$$f(x) = \log \mathcal{N}(x \mid 0_2, \Sigma) - \log c = \log \sqrt{10} - \log(2\pi) - \frac{x_0^2}{2} - 10\frac{x_2^2}{2} - \log c$$

and here we choose $c = -2.95$. For any $n \in [\![P]\!]$, the target density we wish to sample from is

$$\mu_n(x) \propto \mathbb{I}(|f(x)| \leq \epsilon_n)\mathcal{N}(x \mid 0_2, \mathrm{I}_2) \qquad \text{where } \epsilon_n > 0.$$

Figure 3.22 shows a kernel density estimate of the target for $\epsilon = 1$. It has two high-density regions at the north and south poles, whereas almost no mass on the sides since v is isotropic.

3.7.1.2 Automatic Tolerance Selection Strategy

For SMC samplers, it is often beneficial to have an adaptive tolerance schedule based on the Effective Sample Size (ESS), as suggested in [39]. In the context of filamentary distributions, a simple strategy would be to set the next tolerance ϵ_n to a given quantile of the distribution of

the distances of the particles from the manifold. However, as noticed by [13] and [48], adapting the tolerance to have a certain ESS may not be a good criterion when the acceptance rate of the kernels is particularly low (see cited papers and references therein for more details). They address this issue by instead aiming for a certain number $0 < U \leq N$ of unique particles after resampling. Importantly, using either scheme adds a small bias that becomes smaller as the number of particles increases [28]. When using a uniform kernel, our tolerance adaptation strategy can be summarised as follows: given particles $X_n^{(i)}$ we set

$$\epsilon_{n+1} := \min\left[\epsilon_n, \text{quantile}_q\left(\left\{\|f(X_n^{(i)})\| : i \in [\![N]\!]\right\}\right)\right], \tag{3.23}$$

where $q \in (0,1)$ is a quantile value and determines how fast we aim to decrease ϵ. The smaller the value, the faster the decrease in tolerance. At the start of an iteration, we will therefore have $U = qN$ unique particles, which will typically lead to an ESS less than or equal to qN. This strategy works well both for a standard SMC sampler on $(\mathsf{X}, \mathscr{X})$ as well as for an integrator snippet on $(\mathsf{Z}, \mathscr{Z})$. When using Algorithm 7, we found that at setting ϵ_n as in Equation (3.23) at the start of iteration n, considering only the X component of the N seed particles, worked well. We notice that it is also possible to make the tolerance schedule doubly adaptive by adapting the quantile value q_n at each iteration based on a metric such as the acceptance probability of the forward kernel (for SMC samplers) or some other metric of acceptance, in the spirit of [5], and as described in the next subsection. This can be helpful to decrease ϵ faster when the problem is "easier" and then more conservatively when ϵ is small. However, we do not apply this to our experiments. We note that the adaptation strategy in Equation (3.23) is not guaranteed to work well when the kernel is not compactly-supported, for instance when it is a Gaussian kernel. In that case, we suggest enforcing a strictly decreasing tolerance schedule in those scenario, it is enough to choose a proportion $\rho_\epsilon \in (0,1)$ and set

$$\epsilon_{n+1} := \min\left[\rho_\epsilon \epsilon_n, \text{quantile}_q\left(\left\{\|f(X_n^{(i)})\| : i \in [\![N]\!]\right\}\right)\right].$$

3.7.1.3 A framework for adapting step-sizes based on adaptive MCMC

In our experiments we will sometimes update a step size δ using the estimate of a performance measure $\hat{a} \in [0,1]$ (we remain vague to allow this framework to remain general). Typically, for SMC samplers, $\hat{a}$ will be an estimate of the acceptance probability of the forward kernel, estimated using Equation (1.30). In integrator snippets, the metric of performance will typically be mip or mpd (in the experiments that we report in this thesis, we have stuck to mip). The aim is to update δ so that we would achieve $\hat{a} \approx a^*$, for $a^* \in [0,1]$ some target performance. For instance, when using a SMC-RWM $a^* = 0.23$ whereas for a SMC sampler with a mixture of THUG and NHUG, one could set $a^* = 0.5$. In integrator snippets, we set $\text{mip}^* = 0.3$, as explained in subsection 3.5.1.6. Given a learning rate $\gamma \in (0,1]$, the next step size $\delta' > 0$ is then set to

$$\delta' := \text{clip}\left(\exp(\log(\delta) + \gamma(\hat{a} - a^*)), \delta_{\min}, \delta_{\max}\right) \qquad \text{where} \quad 0 < \delta_{\min} < \delta_{\max},$$

where for any $b < c$, $\text{clip}(a, b, c)$ is a function that returns a whenever $b \leq a \leq c$, and returns whichever among b and c is closest to a otherwise. When updating the NHUG kernel, we set $\delta_{\min} = 10^{-30}$ and $\delta_{\max} = 100$ since we are not interested in restricting how small the NHUG step size can get.

3.7.1.4 Algorithmic Details

We run GHUMS, its endpoint version, and an SMC sampler with a mixture of THUG and NHUG with an adaptive tolerance schedule, which at each step aims to keep around 80% of the target ESS, as described above. For each algorithm, we keep the step size for THUG fixed but allow the NHUG step-size to be adaptive, since the filamentary distribution will naturally become tighter as the algorithms progress. For the GHUMS algorithms, the NHUG step size is adapted so as to achieve $\text{mip}_n \approx 0.3$, whereas for the SMC sampler we aim to achieve a target acceptance probability for the NHUG kernel of around 0.5, which is in line with what a sensible acceptance probability for THUG was found in Chapter 2. The addition of the NHUG integrator is helpful in the initial stages of the algorithms to help "locate" the manifold (having Figure 3.21 in mind might help with intuition). In later stages of the algorithms, when ϵ is sufficiently small, we still use NHUG updates since it seems reasonable to assume that they may help both SMC and GHUMS to cover the target well, since we do not know whether using exclusively THUG updates (when ϵ is small enough) would allow the sampler to explore the whole support of the target. We do not show that a mixture of these two integrators (or kernels) is able to reach any region of positive mass, and leave that for future work. We remark that when $\epsilon < \delta^2$, then it not too unreasonable to assume that one would not need the NHUG updates, due to Theorem 2.2, however we still keep the NHUG update in our algorithms and leave any discussion of ergodicity to future work. All algorithms are terminated either when they reach a terminal tolerance of $\epsilon_{\min} = 10^{-16}$, when they execute over 5000 iterations or when the acceptance probability (or the proportion of particles moved) for the THUG update falls below 0.01. The terminal tolerance is chosen to be of the same order of magnitude than floating-point arithmetic error. If the algorithms compared here used RWM kernels or the DW integrator, then one could choose a much smaller acceptance probability (as well as target probability/mip, in the sense of 3.7.1.3), since it has been shown that on densities concentrated around a (linear) manifold of codimension 1 it is preferable to maintain a low acceptance probability of taking large jumps [15].

3.7.1.5 Minimum tolerance

Figure 3.23 shows the minimum epsilon achieved by GHUMS, GHUMS-endpoint and SMC with a mixture of THUG and NHUG for a grid of values of T and δ, averaged over 10 runs, with 5000 particles each. Firstly, we notice that all algorithms, unsurprisingly, are able to achieve very small tolerances even with large step sizes, which is a testament to the effectiveness of the THUG update. GHUMS-endpoint achieves consistently one or two orders of magnitude smaller epsilon

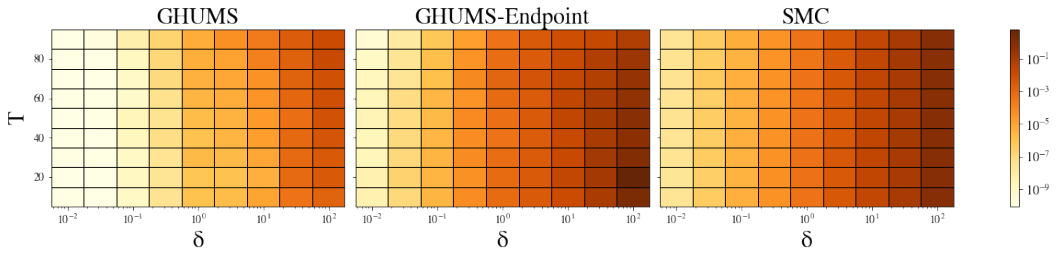

Figure 3.23: Average terminal tolerance for GHUMS, GHUMS-endpoint, and the SMC sampler.

than the SMC sampler, and GHUMS achieves consistently two to four orders of magnitude smaller tolerances. The effectiveness of the Markov snippet in being able to choose the best point along and across each trajectory allows it to reach smaller epsilons.

3.7.1.6 Performance on the same filamentary distribution

To compare the algorithms sensibly, we need to test their performance on the same final target distribution. To do this, we need to find a suitably small ϵ. We run 10 instances of the SMC sampler until termination and select the maximum across all the terminal tolerances, which in these experiments is $\epsilon_{\text{min}} = 5.00 \times 10^{-6}$. We then run all algorithms with $T = 50$, $\delta = 0.1$ and with an adaptive ϵ schedule but with a terminal tolerance of ϵ_{min}, meaning the the samples and metrics at the last iteration will all be with respect to the same target distribution.

	GHUMS	GHUMS-Endpoint	SMC
ESJD$_A$/s	2.4×10^{-4} ($\pm 2.9 \times 10^{-5}$)	$\mathbf{5.7 \times 10^{-4}}$ ($\pm 1.5 \times 10^{-4}$)	1.1×10^{-6}($\pm 1.0 \times 10^{-7}$)
ESJD-THUG$_A$/s	3.0×10^{-4} ($\pm 3.7 \times 10^{-5}$)	$\mathbf{7.5 \times 10^{-4}}$ ($\pm 2.2 \times 10^{-4}$)	1.4×10^{-6}($\pm 9.1 \times 10^{-8}$)
ESJD-NHUG$_A$/s	1.3×10^{-3} ($\pm 3.4 \times 10^{-4}$)	$\mathbf{2.5 \times 10^{-3}}$ ($\pm 8.8 \times 10^{-4}$)	1.3×10^{-14}($\pm 4.9 \times 10^{-15}$)
Var$_{0,A}$	$\mathbf{1.3 \times 10^{-4}}$	1.7×10^{-2}	1.3×10^{-2}
Var$_{1,A}$	4.9×10^{-3}	5.5×10^{-3}	$\mathbf{3.3 \times 10^{-3}}$
RMSE$_A$	$\mathbf{1.2 \times 10^{-5}}$ ($\pm 2.8 \times 10^{-3}$)	1.4×10^{-4} ($\pm 9.3 \times 10^{-3}$)	2.4×10^{-4}($\pm 8.8 \times 10^{-3}$)
ESJD$_B$/s	$\mathbf{1.6 \times 10^{-4}}$ ($\pm 1.2 \times 10^{-5}$)	2.8×10^{-5} ($\pm 4.1 \times 10^{-6}$)	1.1×10^{-6}($\pm 1.0 \times 10^{-7}$)
ESJD-THUG$_B$/s	$\mathbf{2.0 \times 10^{-4}}$ ($\pm 1.5 \times 10^{-5}$)	3.5×10^{-5} ($\pm 5.1 \times 10^{-6}$)	1.4×10^{-6}($\pm 9.1 \times 10^{-8}$)
ESJD-NHUG$_B$/s	$\mathbf{8.4 \times 10^{-4}}$ ($\pm 1.2 \times 10^{-4}$)	1.4×10^{-4} ($\pm 2.1 \times 10^{-5}$)	1.3×10^{-14}($\pm 4.9 \times 10^{-15}$)
Var$_{0,B}$	$\mathbf{6.9 \times 10^{-5}}$	2.6×10^{-2}	1.3×10^{-2}
Var$_{1,B}$	4.2×10^{-3}	2.6×10^{-2}	$\mathbf{3.3 \times 10^{-3}}$
RMSE$_B$	$\mathbf{3.2 \times 10^{-6}}$ ($\pm 2.6 \times 10^{-3}$)	2.2×10^{-4} ($\pm 1.4 \times 10^{-2}$)	2.4×10^{-4}($\pm 8.8 \times 10^{-3}$)
AP/PM	$\mathbf{0.6}$ (± 0.01)	0.01 ($\pm 2.3 \times 10^{-3}$)	9.9×10^{-3} ($\pm 9.6 \times 10^{-5}$)

Table 3.1: Performance metrics for the three algorithms for the target with ϵ_{min} averaged over 20 runs. Standard deviation reported in brackets. AP is the estimate of the SMC sampler's acceptance probability for the THUG kernel, calculated using Equation (1.30) for particles with $\iota = 1$. PM is the proportion of particles moved for particles using the THUG integrator.

Table 3.1 displays several metrics for the target with tolerance ϵ_{min}, averaged across 20

runs. The table is divided into three horizontal sections to help readability. The first section reports metrics obtained using estimators of type A, the second using estimators of type B, and the final section displays measures of acceptance. ESJD$_A$/s and ESJD$_B$/s display the ESJD at the final iteration divided by total running time (in seconds). ESJD-THUG$_A$/s, ESJD-NHUG$_A$/s, ESJD-THUG$_B$/s, ESJD-THUG$_B$/s display the same metric but computed only for the THUG and NHUG updates using estimators of type A and B respectively. Var$_{0,A}$, Var$_{1,A}$ show the variance of the estimators for the two components of the posterior mean, computed using estimators of type A, and Var$_{0,B}$, Var$_{1,B}$ show the equivalent variances using estimators of type B. RMSE$_A$ and RMSE$_B$ show the average RMSE for the two components of the posterior mean using the respective estimators A and B. Finally, The last section of the table shows the acceptance probability for the THUG kernel (for the SMC sampler) and the proportion of particles moved that used the THUG integrator. In parenthesis, we show the standard deviation of each of these estimates. Bold values represent the best value along the corresponding row, where best is dependent on the metric at hand (for instance, we want large ESJD and low RMSE).

Firstly, we notice that, as expected, GHUMS is able to retain a much higher measure of acceptance. Secondly, we note that in terms of ESJD, GHUMS achieves the best value in all metric when we compute them using estimators of type B, whereas its endpoint version achieves the best values with estimators of type A. We attribute this behavior to what was observed when studying these two types of estimators. The SMC sampler beats the other two algorithms only in the variance for the estimates of the second component of the posterior mean. This difference between the components can be attributed to the fact that due to the prior being an isotropic Gaussian and the constraint function being an ellipse with major axis parallel to the x-axis, most of the variability is on the first component (recall that Figure 3.22 shows the target for $\epsilon = 1$ whereas here we are considering $\epsilon = 5 \times 10^{-6}$, there will be virtually no mass on the eastern and western poles: all the mass will be concentrated on two very thin strips near the north and south pole). For small enough tolerances, we can expect the variability on the second component to depend, in large part, on the size of the minor axis and therefore not on the performance of the algorithm. The impressive difference on the NHUG update is that at times the trajectory, which is constructed without any MH update, will intersect the ellipse, whereas the MH update prevents this from happening, since stepping out of the tolerance region will result in a sure rejection.

3.7.1.7 Influence of N and T on ESJD for a fixed budget $\Omega = NT$

Next, we study the behavior of the algorithms for a fixed budget $\Omega = NT = 120000$ for a range of integration steps $T = [2,5,10,20,30,40,50,75,100,150,250,375,500]$. For each pair (T,N), with $\Omega = NT$, we first run an SMC sampler with a mixture of THUG and NHUG kernels until termination (same rules as subsection 3.7.1.4 apply) and thus obtain a final epsilon ε_{smc} corresponding to the last target distribution that the algorithm was able to sample from. Then, we

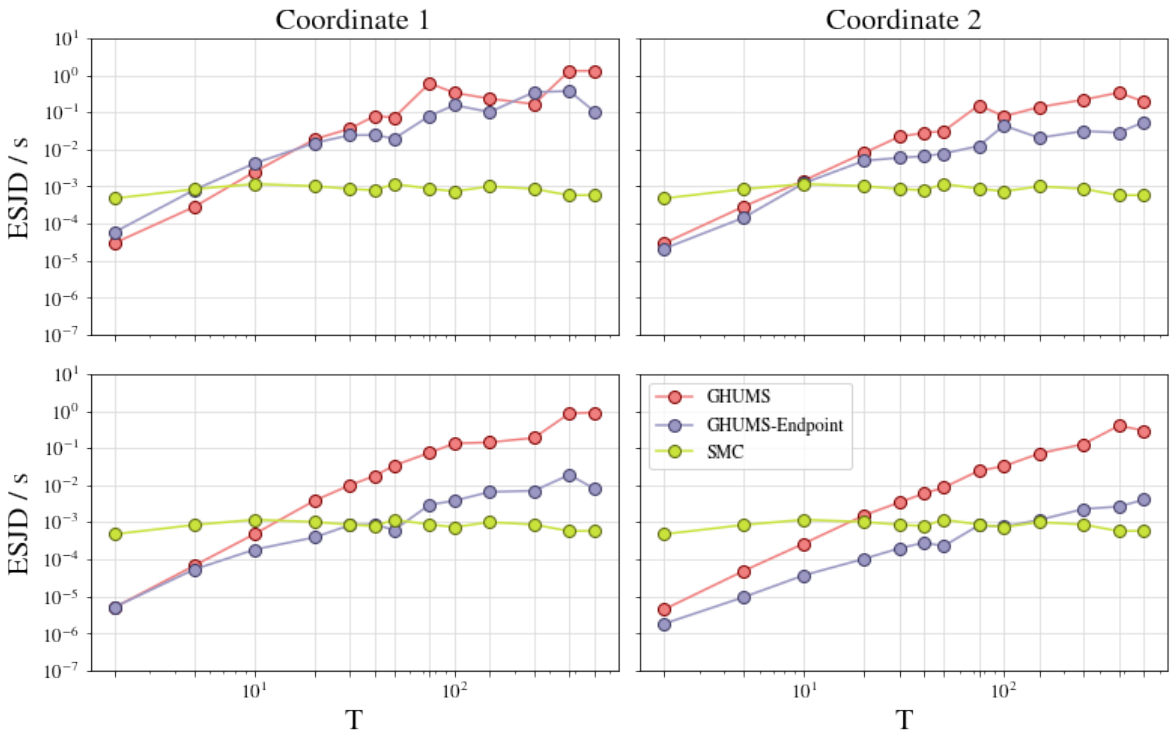

Figure 3.24: ESJD divide by total runtime for the three algorithms, for a fixed budget $\Omega = BN$. For GHUMS the ESJD is computed for test functions corresponding to the two coordinates of the posterior mean.

run GHUMS and GHUMS-Endpoint twice each. The first time, we run them until they reach ϵ_{smc}, the second time we run them until termination. The aim of this is twofold: the first set of runs allow us to compare SMC and GHUMS/GHUMS-Endpoint on the same final target distribution, whereas the second set of runs allows us to see what is the tightest filamentary distribution they can sample from. The top row of Figure 3.24 shows ESJD_A for ϵ_{smc} divided by total running time (in seconds) for the two identity test functions corresponding to the posterior mean, as the number of integration steps T (and therefore N) varies. The second row displays ESJD_B/s for the same two test functions. On the first coordinate (the first column of plots) both ESJD metrics show that for $T > 10$ GHUMS already starts being more efficient than the SMC sampler, and for large T we obtain up to 3 orders of magnitude ESJD. Interestingly, this improvement is reflected also for the endpoint algorithm. On the second component, we see similar results with an improvement up to two orders of magnitude but now the threshold is shifted to $T > 20$.

Figure 3.25 consists of three plots. The left-most plot shows the final tolerances (for GHUMS and GHUMS-Endpoint we consider the final tolerance from the second run) as a function of total runtime (both ordered by time). Firstly, we notice that GHUMS consistently achieves smaller tolerances than the SMC sampler, however for $T = 2$ its runtime is four times larger than for most other values of T. The SMC sampler scales well with different values of T and N in terms of

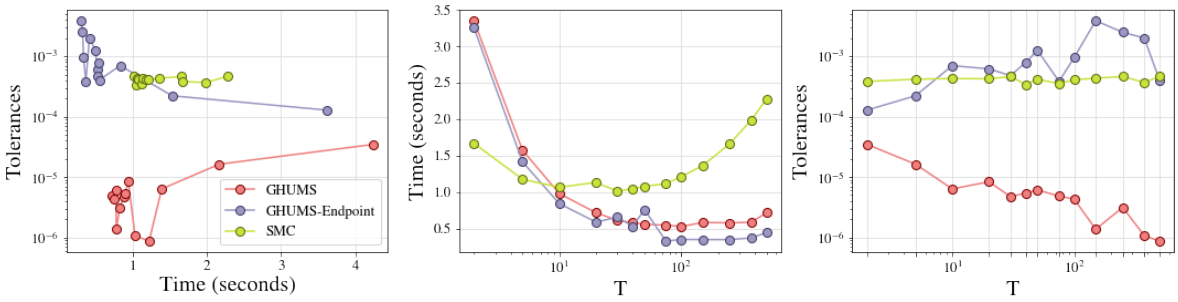

Figure 3.25: Various metrics for a fixed budget $\Omega = TN$.

total runtime but it is unable to achieve as small tolerances. The middle plot in Figure 3.25 offers a clearer picture regarding our earlier comment: for small T GHUMS and GHUMS-Endpoint are less efficient than the SMC sampler, but their total runtime scales much better with increasing T, since we can see the SMC sampler's total runtime is quickly increasing with T, but GHUMS's total runtime remains relatively stable. Finally, the right-most plot shows final tolerances as a function of T and we can clearly see that GHUMS's final tolerance steadily decreases for larger T. This suggests, that when targeting filamentary distributions, it may be beneficial assigning more budget to a larger value of T rather than a larger value of N. There is, of course, a trade-off: a small value of N means our estimates will be more biased. All experiments in this thesis were vectorised but not parallelised, but in practice one could adapt the code to take full advantage of modern computing hardware. The resampling steps remains a bottleneck, although parallel resampling schemes have been suggested [102] and could help integrator snippets become even more competitive.

Interestingly, for small T, both GHUMS and GHUMS-Endpoint are less efficient than the SMC sampler. This is a behavior that persists even when we are not restricted to a fixed budget Ω, as we show in Figure 3.26 for ESJD_A/s. For $T > 5$ the ESJD of GHUMS for all test functions is at least the same as that of the SMC sampler, and for $T \geq 20$ is around one order of magnitude larger. A lower ESJD for small T is exacerbated, in the fixed budget scenario, by the resampling step, which requires sampling N particles out of $N(T+1)$ for integrator snippets, and provides a bottleneck for the parallelisation of the algorithm. All algorithms utilise multinomial resampling, which is known to be sub-optimal in terms of variance of the resulting estimates, although the results changed only marginally when we used systematic or more advanced resampling methods (discussed in Subsection 3.10.2.4). Parallel resampling algorithms are available and are discussed in the same subsection.

3.7.1.8 T-vs-P trade-off for a fixed NTP budget

In the previous subsection, we have kept fixed a budget for NT but in practice the total runtime of the algorithm depends on all three parameters N, T, and P. In particular, while mutation

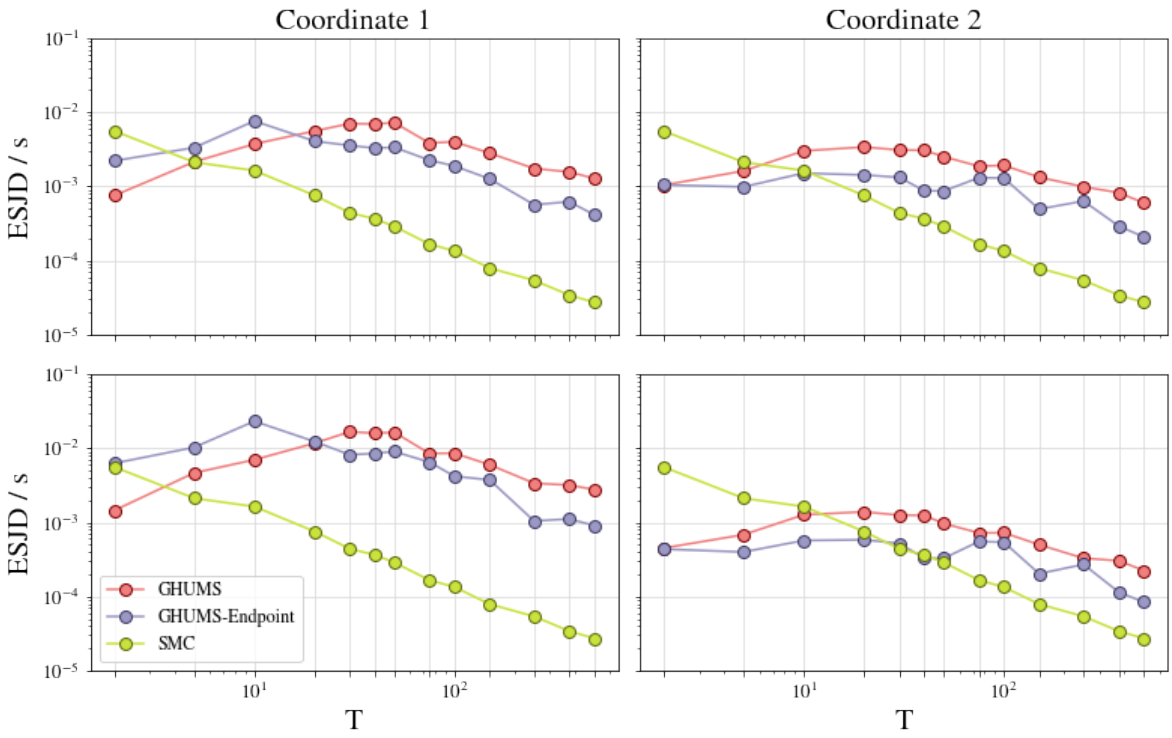

Figure 3.26: ESJD_A over total runtime (in seconds) for a fixed number of particles $N = 10000$ but increasing T (same values as for the fixed budget). The ESJD for GHUMS/GHUMS-Endpoint is for the posterior means (first row) and second moments (second row).

and weighting of the particles can be computed independently, the parameters T and P are linked to two inherently sequential mechanisms of the algorithms: the construction of the trajectories and the number of intermediate distributions. We fix a budget $NTP = 15\,000\,000 =: \Omega$, an initial tolerance $\epsilon_0 = 100$ and a final one of $\epsilon_P = 1e-5$, so that the endpoints of the sequence of distributions are identical and results are comparable. As before, we let $p_{\text{thug}} = 0.8$ and compute the ESJD for the posterior mean of each coordinate on the final distribution μ_p, divided by total runtime. We run the algorithms for three different values of $N = [100, 1000, 5000]$ and, for each of these values, we use $T = [2, 10, 25, 50, 100, 200, 500]$ and the respective values for P i.e. Ω/T. This is shown in Figure 3.27. We notice that, just as in the previous subsection, the difference in performance is not the same for the two coordinates. Additionally, for small N, we have find an important improvement in overall efficiency by using GHUMS rather than the SMC sampler. This gap reduces as both N and T increase, which makes sense: for a fixed budget NTP, whenever N and T are large, P must be small and therefore the transition between densities will likely make both algorithms suffer. Overall, we find results compatible with the previous subsection: increasing T seem to be a reliable way to improve efficiency.

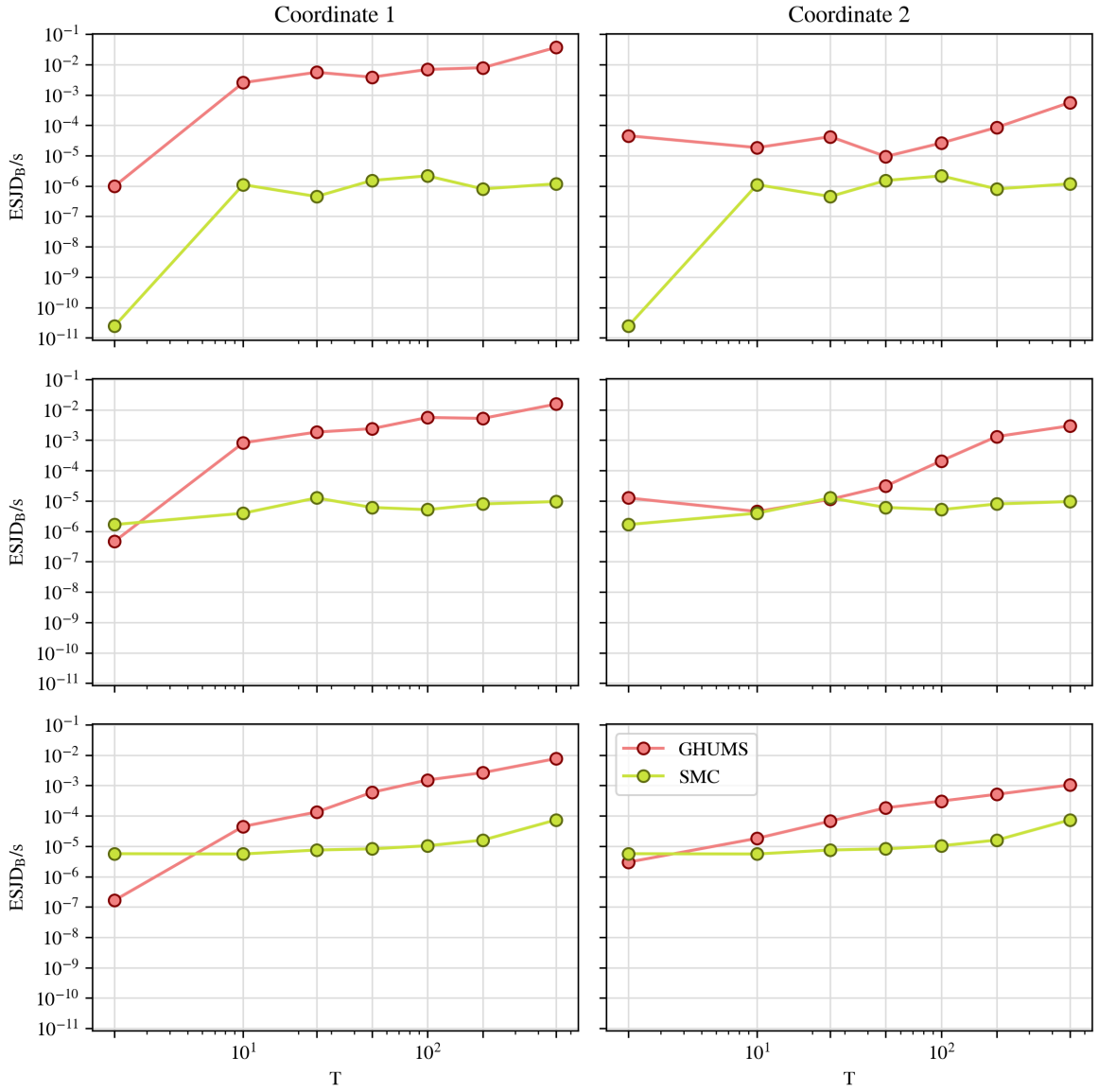

Figure 3.27: ESJD_B for the first (left column) and second (right column) coordinate on the final target distribution, divided by total runtime. The rows from top to bottom correspond to $N = 100, 1000, 5000$ respectively.

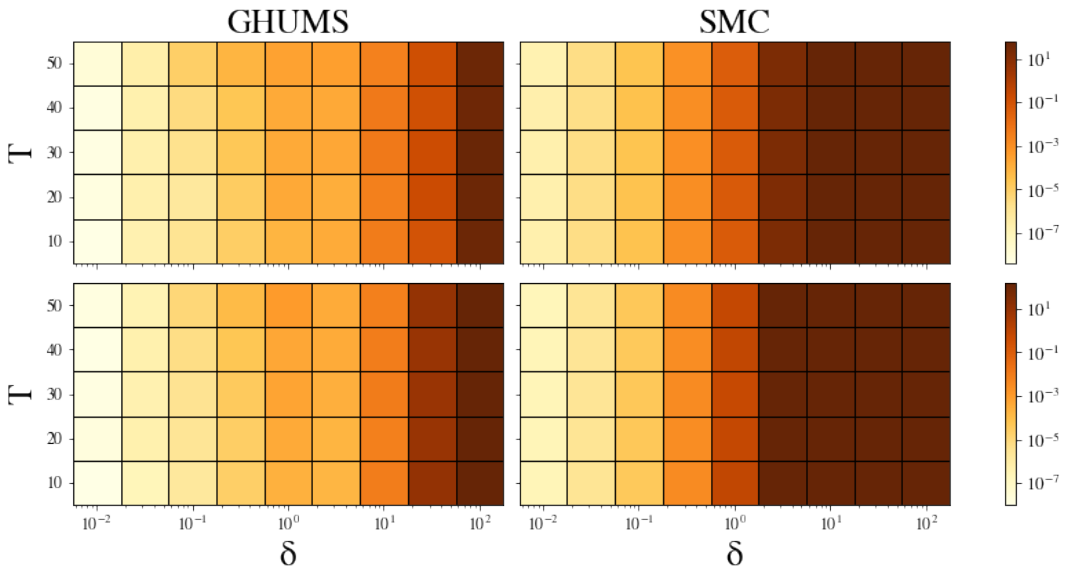

Figure 3.28: Minimum epsilon achieved by GHUMS and SMC with 5000 particles for $T \in [10, 20, 30, 40, 50]$ and δ equally log-spaced between 0.01 and 100.0. The top row shows the results for a 20-dimensional ellipsoid problem, whereas the bottom corresponds to a 50-dimensional problem.

3.7.2 20-dimensional Ellipsoid

To study how the improvements brought by Markov Snippets behave in larger dimensions, we now consider a 20-dimensional ellipsoid generated as the level set of a zero-centered multivariate normal with a diagonal covariance matrix whose entries are, in alternation, 1.0 and 0.1. We choose the $\exp(-12)$-level set, and the reason for this choice is explained in Appendix D.3.1.

3.7.2.1 Minimum Epsilons

The top row of Figure 3.28 shows the minimum epsilon achieved by GHUMS and the SMC sampler with 5000 particles for a grid of T and δ values. GHUMS is more robust to the choice of the step size than the SMC sampler and manages to sample from approximately the same filamentary distribution with a step size more than an order of magnitude larger. The bottom row shows the same results for a 50-dimensional ellipsoid.

3.7.2.2 Performance on the same filamentary distribution

We choose to compare GHUMS with the SMC sampler on a common filamentary distribution with $\epsilon = 3 \times 10^{-7}$. The step size for the THUG update is $\delta = 0.01$ and the one for the NHUG update is adapted as usual. We set $T = 50$ integration steps and average all results over 20 independent

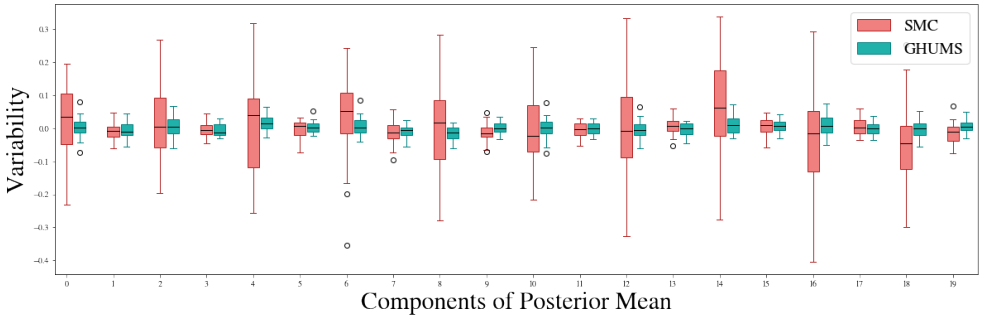

Figure 3.29: Spread of estimates of posterior mean components for SMC and GHUMS on 20-dimensional ellipsoid example.

runs with $N = 5000$ samples. Table 3.2 showcases equivalent metrics to those in Table 3.1 for our 20-dimensional scenario, all ESJD metrics are averaged across all 20 components.

	GHUMS	SMC
ESJD_A/s	$\mathbf{1.2 \times 10^{-3}}$ $(\pm 3.5 \times 10^{-4})$	4.7×10^{-7} $(\pm 3.3 \times 10^{-8})$
$\text{ESJD-THUG}_A/s$	$\mathbf{1.5 \times 10^{-3}}$ $(\pm 4.5 \times 10^{-4})$	6.4×10^{-7} $(\pm 3.2 \times 10^{-8})$
$\text{ESJD-NHUG}_A/s$	$\mathbf{3.8 \times 10^{-3}}$ $(\pm 1.2 \times 10^{-3})$	2.0×10^{-18} $(\pm 2.1 \times 10^{-19})$
ESJD_B/s	$\mathbf{1.1 \times 10^{-4}}$ $(\pm 9.7 \times 10^{-6})$	4.7×10^{-7} $(\pm 3.3 \times 10^{-8})$
$\text{ESJD-THUG}_B/s$	$\mathbf{1.3 \times 10^{-4}}$ $(\pm 1.2 \times 10^{-5})$	6.4×10^{-7} $(\pm 3.2 \times 10^{-8})$
$\text{ESJD-NHUG}_B/s$	$\mathbf{5.3 \times 10^{-4}}$ $(\pm 6.1 \times 10^{-5})$	2.0×10^{-18} $(\pm 2.1 \times 10^{-19})$
AP/PM	**0.29** (± 0.01)	0.01 $(\pm 6.4 \times 10^{-4})$

Table 3.2: Equivalent to Table 3.1 on 20-dimensional ellipsoid example, showcasing various ESJD and acceptance metrics.

Figure 3.29 is a boxplot displaying the variability of the estimates of the posterior mean, over the 20 independent runs. As noticed in the previous example, GHUMS has an advantage on all the components with larger scale and about the same performance on those with smaller scale.

3.7.3 G-and-K Example

Similarly to Chapter 2, we now test GHUMS on the G-and-K problem with $m = 50$ latent variables. To compare the algorithms, we first obtain an initial set of $N = 2500$ particles by sampling (with thinning of 100) from η_ϵ with $\epsilon = 3.0$ using a simple RWM sampler with a step size chosen to accept around 23% of samples. From this set of particles, we first run an SMC sampler with a mixture of THUG and NHUG until termination, which happens when we hit an acceptance probability for the THUG kernel below 0.01. We let the step size for the NHUG kernel to adapt to arbitrarily small values, while maintaining an acceptance probability of 50%, as described in subsection 3.7.1.3. The step size for THUG is initially set to $\delta = 0.3$ and adapted similarly, except we do not let it decrease to arbitrarily small values, but rather impose a minimum value

$\delta_{\min} = 0.01$. The adaptation strategy for THUG is only needed to speed up the initial exploration of the algorithms, which otherwise could take a long time to reach small values of ϵ. Once the algorithm terminates, we store the final epsilon achieved $\epsilon_{\min}$. We then run GHUMS in a similar way except adaptation of the step sizes seeks to maintain a median index proportion of 0.3, as explained in subsection 3.5.1.6. We terminate the algorithm once it reaches $\epsilon_{\min}$, so that results are comparable for the final target distribution. Figures 3.30 and 3.31 show ESJD (of type A and B respectively for GHUMS) over cumulative running time in seconds as a function of the tolerance. Each of these figures has three columns, each corresponding to a different value of T, and four columns corresponding to the test functions for the first four components of the posterior mean, corresponding to the parameter of interest $\theta = (a, b, g, k)$. In either figure, we can see that for $T = 10$ GHUMS is more efficient than the SMC sampler and for the final target we get at least four orders of magnitude improvement for the first three coordinates, and more than one order of magnitude for the final coordinate (generally speaking, we have found inference for the parameter k to be the one where our algorithms have struggled the most). As T becomes larger, GHUMS still has a considerable improvement over the SMC sampler, but the gap narrows by one order of magnitude for the first three coordinates, which is in contradiction to earlier results, that showed a starker difference between the algorithms for larger T. Upon examination, we found that this was mostly due to the algorithm requiring more iterations to reach the final target, when T was larger, since the ESJD for $T = 10$ and $T = 20$ was in the same order of magnitude at the final target (but it was one order of magnitude smaller for $T = 40$). The reason why GHUMS is requiring more iterations to reach the final target in this particular example is unclear. One possible way to interpret this is that the GHUMS algorithm is actually exploring the normal directions to the manifold more thoroughly thanks to the NHUG trajectories being constructed without a MH step, which also allows the algorithm to jump in and out of the tolerance region, thus achieving large values of ESJD along these trajectories. Figure 3.32 provides some clues about this behavior. It shows the ESJD for the NHUG update (for GHUMS we display the Type B ESJD, but type A has similar values and trends) and we can clearly see a stark difference, which we attribute to the lack of a MH step. Finally, in Figure 3.33 we show the acceptance probability and the proportion of particles moved for the THUG update for both algorithms. In accordance to previous experiments, the GHUMS algorithm is able to select particles different from the seed ones (i.e. avoid rejection) more often than the SMC sampler for smaller tolerances. In particular, we can see that for the final tolerance the SMC sampler's acceptance probability is close to zero, whereas the proportion of particles moved is still very high, for all three settings of T.

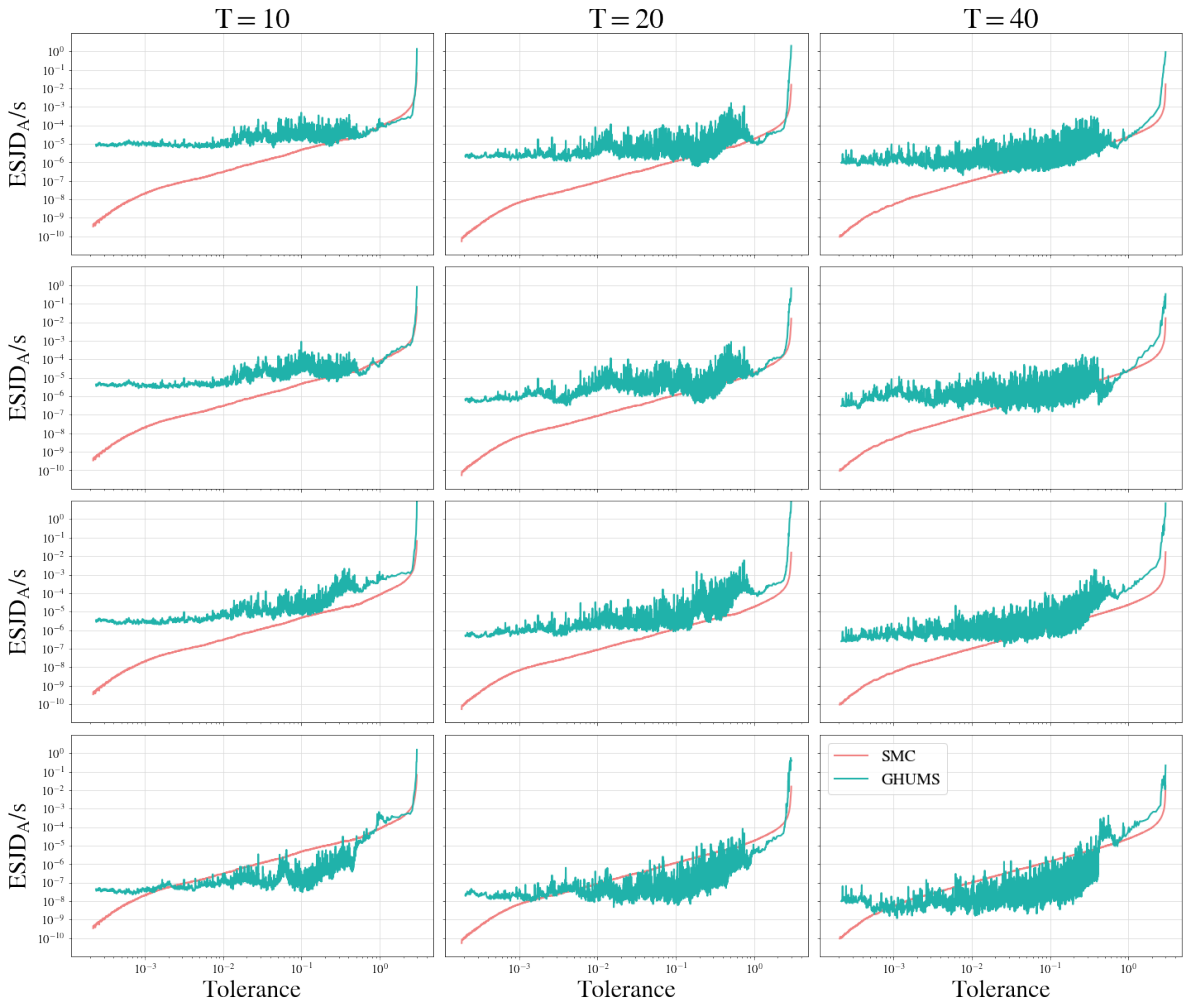

Figure 3.30: ESJD over cumulative running time (in seconds) on the G-and-K problem for GHUMS and SMC. For GHUMS we report the ESJD of type A. Rows correspond to the four parameters $\theta = (a, b, g, k)$, columns correspond to $T = 10, 20, 40$.

3.8 Adaptive Tuning of Integrator Snippets

3.8.1 Aim and Motivation

For a thorough review of the relevant literature regarding optimal tuning of sampling parameters, we refer the reader to our later Section 3.10. The aim of this section is not to be a thorough investigation of adaptation techniques in Integrator Snippets, but rather to convince the reader that our framework naturally and, elegantly, lends itself to adaptation of its hyperparameters. Once more, we shall focus on the specific task of approximate manifold sampling, but we remark that the observations made here are likely to apply to more general sampling scenarios.

Furthermore, although different integrators will behave differently and thus potentially require bespoke strategies, we believe that the content of this section is going to provide a

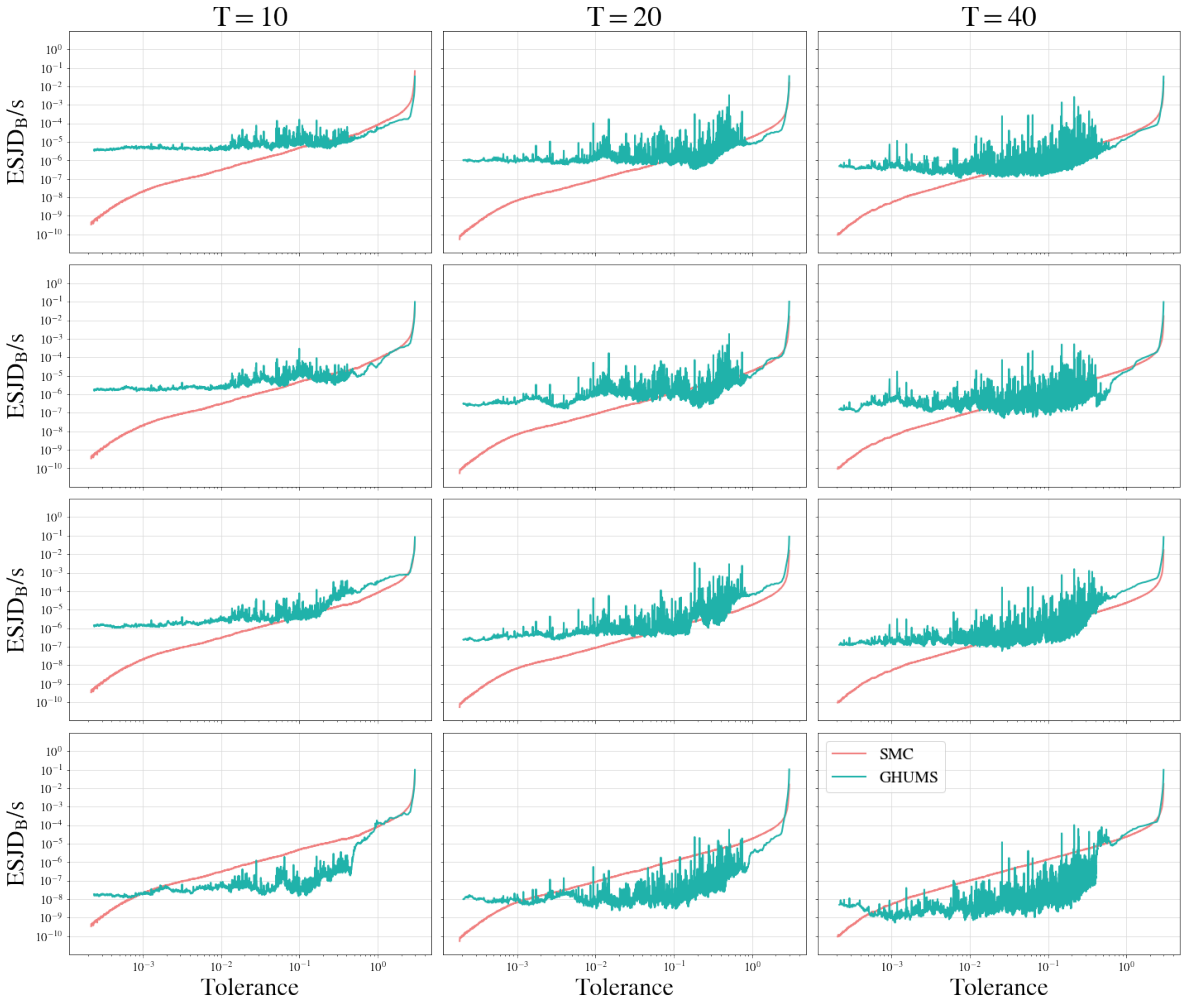

Figure 3.31: ESJD over cumulative running time (in seconds) on the G-and-K problem for GHUMS and SMC. For GHUMS report the ESJD of type B. Rows correspond to the four parameters $\theta = (a, b, g, k)$, columns correspond to $T = 10, 20, 40$.

good foundation for adapting integrator snippets in other tasks, and especially so when using integrators of Hamiltonian dynamics.

3.8.2 Adaptively tuning the step size

In Hamiltonian Monte Carlo samplers, determining the step size δ is a much easier problem than the number of integration steps T. This is because one can easily resort to stochastic approximation/dual averaging techniques to tune δ to achieve a certain acceptance probability. Scaling limit arguments have provided practitioners with rules of thumbs, such as tuning the step sizes to achieve roughly a 65.1% acceptance rate. Rigorously derived criteria motivating adaptation strategies for the stepsize in Integrator Snippets are at present lacking, and while we

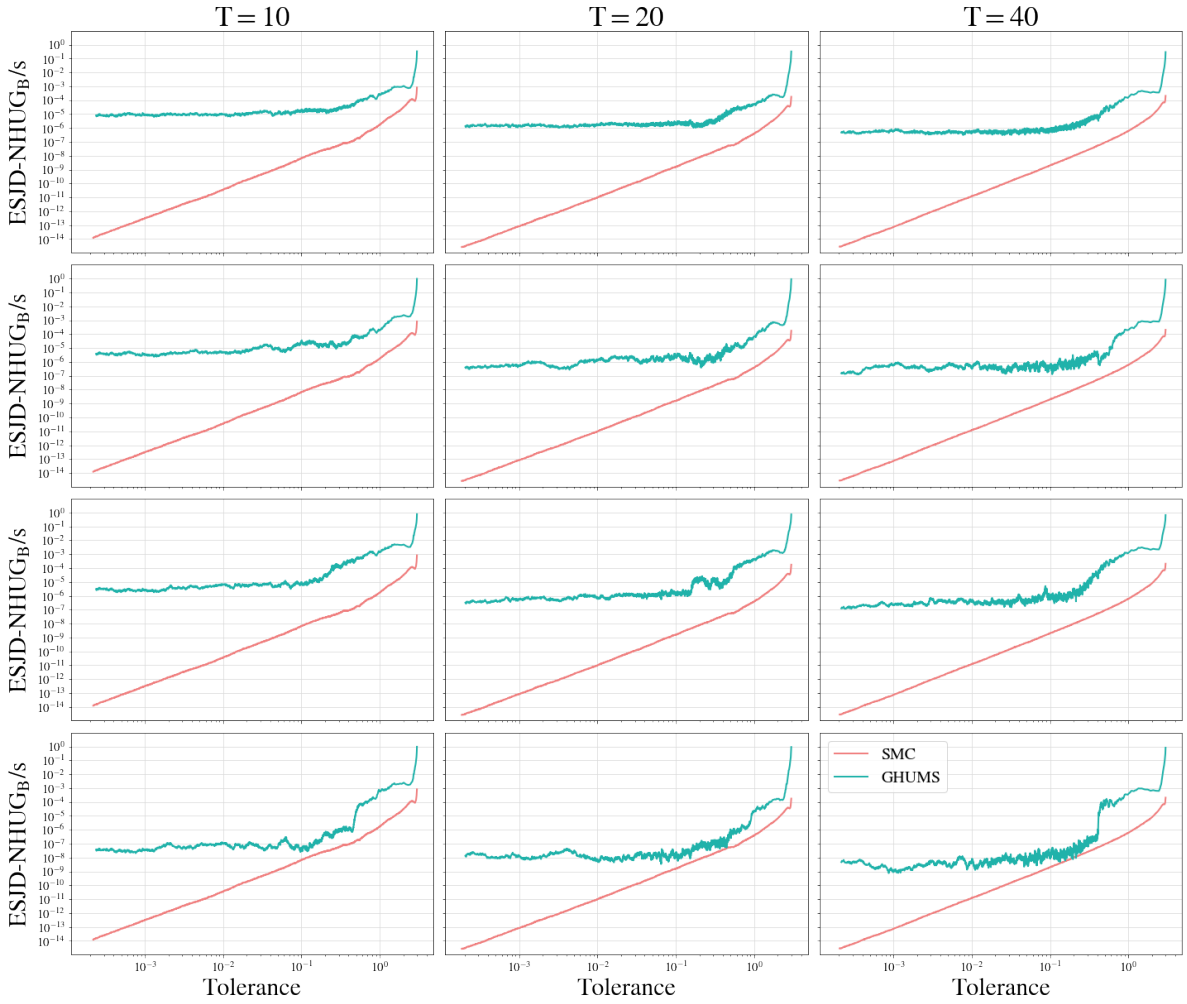

Figure 3.32: ESJD for the NHUG update.

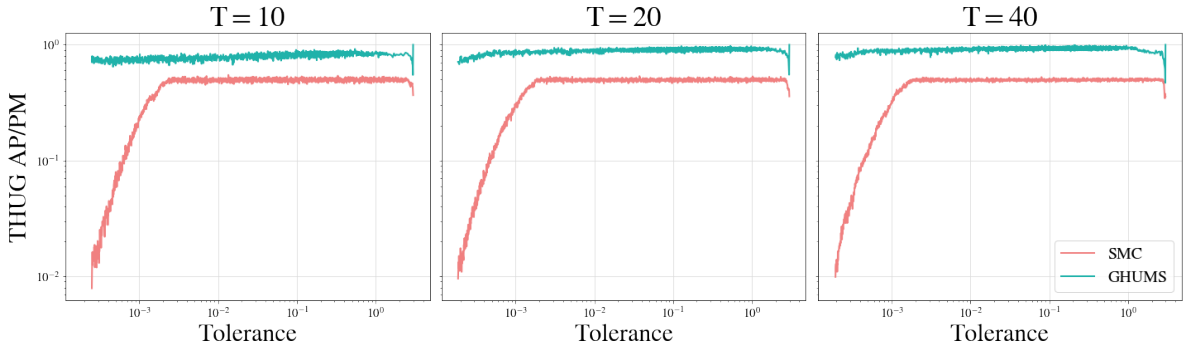

Figure 3.33: Acceptance probability of the THUG kernel of the SMC sampler, and proportion of particles moved of the THUG integrator of the GHUMS algorithm.

believe that it would be an important avenue of research, we leave it to future work.

Nonetheless, we have found that tuning the step size to achieve a certain median index proportion to work well with a number of integrators, see Subsection 3.7.1.3 for more details. We believe that this adaptation strategy is likely to work well in general, since mip and mpd (Equations (3.20) and (3.21) respectively) contain information regarding the distribution the resampled trajectory indices, and it seems reasonable to aim for a decaying distribution as the one show in Figure 3.8.

3.8.3 Adaptively tuning the number of integration steps

Tuning T is still an open problem in HMC samplers. Most of the methods reviewed in the literature review of Section 3.10 can be divided into two categories:

- Select T by minimizing a ESJD-inspired criterion using gradient-based optimization.
- Learn or construct a discrete distribution over the number of integration steps.

For the rest of this section, we will work on the 2D ellipsoid example of subsection 3.7.1 and we will focus on an integrator snippet using only the THUG integrator: we will not be using the NHUG integrator. This is to make the main ideas clearer and aid exposition, but we will show in the last subsection how one can perform adaptive tuning of T when using a mixture of integrators, such as when using GHUMS. Since we will only be using the THUG integrator, we will initialise particles using GHUMS to a small enough $\epsilon > 0$ such that the error in the THUG integrator will be enough to let it change contours well.

3.8.3.1 How not to adapt T: adaptation based on ESJD

When using a ESJD-based criterion, several works have found that taking the computational cost (typically in terms of number of integration steps or gradient evaluations) into account was beneficial [32, 132, 156]. Inspired by these works, a simple criterion would be to compute each of the T terms inside our ESJD_B metric (the same can be done with ESJD_A; we observed the same behavior as the one reported in this Subsection) and normalise them by the number of integration steps or its square root. Doing this for test functions $h_p(z) = x_p$ with $p \in [\![d]\!]$ and summing all these terms, leads to the following metric (we drop the index n for clarity)

$$\text{ESJD-NORM}(\ell) = \frac{1}{\ell}\sum_{k=0}^{\ell-1} \frac{\sum_{i=1}^{N} \left\| X_\ell^{(i)} \breve{w}_\ell(Z^{(i)}) - X_k^{(i)} \breve{w}_k(Z^{(i)}) \right\|^2}{\left(\sum_{\kappa=0}^{T} \sum_{i=1}^{N} \breve{w}_\kappa(Z^{(i)}) \right)^2} \qquad \ell \in [\![1, T]\!]. \tag{3.24}$$

The ℓ^{th} term corresponds to the expected squared jump distance for the ℓ^{th} point in the trajectory, normalised by the number of integration steps. Here we have used a linear test function, which

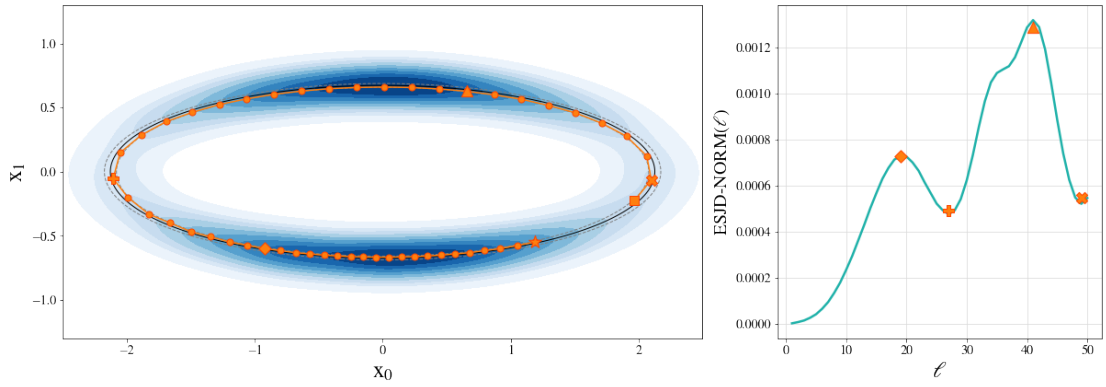

Figure 3.34: Left: A THUG trajectory (orange, size of the markers is proportional to their weight) initialised at the star marker and ended at the square marker. The diamond and plus sign represent a high and a low density region respectively. The Kernel Density Estimate plot was generated using 10000 GHUMS samples. Black solid line is the ellipsoid, and the dashed lines represent the active region under k_ϵ. Right: We run an integrator snippets using only the THUG integrator and estimate the metric in Equation (3.24), which is displayed in green for all values $\ell \in [\![1, T]\!]$. The orange diamond and plus sign correspond to the ones in the left plot.

minimizes lag-one autocorrelation, but more complex functions can be chosen, and we refer the reader to [70] and [132] for a discussion on the choice of test functions $h_p(z) = x_p$. For a single particle $Z^{(i)} = (X^{(i)}, V^{(i)})$, we have found that this metric often correctly highlights high-density regions that are further away from the initial seed particle, as well as far away low-density regions, for a single particle. Figure 3.34 shows the trajectory generated with a THUG integrator in orange. The trajectory starts at the star symbol and ends at the square symbol. The target density is shown in blue using a Kernel Density Estimate (the construction of this target distribution is described in Subsection 3.7.1). The plus sign denotes a far away point of low-density and the diamond is a relatively far away point on a high density region (A remark is in order: the high-density regions are technically far away along the contours, but relatively nearby in terms of Euclidean norm, and the north/south poles are closer than the east/west poles. However, this does not have an impact on our observations and remarks, as shown in Appendix D.3.2, where we place high-density regions at the east and west poles using a mixture of Gaussians). The manifold, which is a simple ellipsoid, is shown in a black solid line and the region between the dashed lines is the active region under a uniform kernel (we have chosen ϵ large for visualisation purposes, but the same results apply for smaller ϵ). This metric works well for a single particle, especially when the THUG trajectory does not double back on itself due to a high curvature region: when the curvature is high, the bouncing mechanism will sometimes produce points that backtrack before moving forward again after a few integration steps, and in these situations the signal from plots like the left one in Figure 3.34 will be harder to discern.

Nevertheless, we have found that when we sum this metric across all particles, we do not

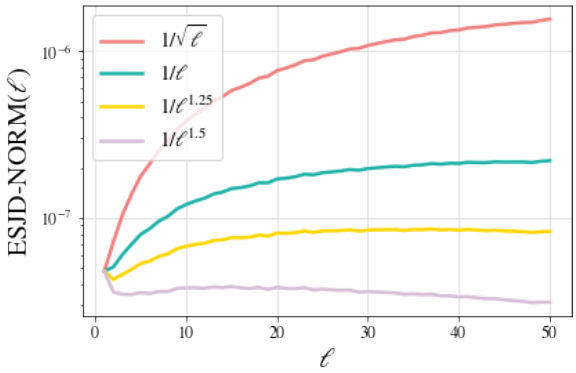

Figure 3.35: Metric in Equation (3.24) normalised by number of integration steps with different factors. The one displayed in the equation corresponds to $1/\ell$.

obtain a single well-defined peak, which instead was found in [132] and [32]. Figure 3.35 shows the metric in Equation (3.24) with different scaling factors $\ell^{-0.5}, \ell^{-1}, \ell^{-1.25}, \ell^{-1.5}$, for 10000 particles with $T = 50$. We can see that the metric is still increasing for scaling factors $1/\ell^p$ with $p > 1$, and only starts decreasing for $1/\ell^{1.5}$, even if in practice the cost is linear in ℓ.

There are various reasons for this, but perhaps one of the most immediate ones is that although the THUG integrator shares many properties of the Leapfrog integrator, this is not an integrator of Hamiltonian dynamics and hence behaves differently. One behavior that we have observed is that in high-curvature regions, the integrator sometimes backtracks, before moving forward again. For the current toy example, this means that even though most of the particles are concentrated in the north and south poles (even more so when ϵ is smaller), this backtracking behavior results in different particles having different optimal numbers of integration steps. As a result, none of the ESJD-based metrics that we have tried, based on the current literature, present a single well-defined peak. We will discuss this further at the end of this section.

[132] found a well-defined peak by focusing on computing the ESJD in the direction of highest variability. In the context of sampling from filamentary distributions, we know that the highest variability is going to be in the tangential directions, and hence one would be tempted to use the *local* criterion (we stress that the metric below uses a different high-variability direction for each particle, which is different from [132], however in this toy example, we have found that the results do not change when using a global direction)

$$\text{TANG-ESJD}(\ell) = \frac{1}{\ell}\sum_{k=0}^{\ell-1} \frac{\sum_{i=1}^{N}\left(X_\ell^{(i),\parallel}\breve{w}_\ell(X_\ell^{(i)}) - X_k^{(i),\parallel}\breve{w}_k(X_k^{(i)})\right)^2}{\left(\sum_{\kappa=0}^{T}\sum_{i=1}^{N}\breve{w}_\kappa(Z_0^{(i)})\right)^2} \qquad \ell \in [\![1, T]\!],$$

where $X_k^{(i),\parallel} = \mathrm{T}_{X_k^{(i)}} X_k^{(i)}$ is the projection of $X_k^{(i)}$ onto the tangent space at that point. By construction, the direction of highest variability will be different at each point (although in this very

simple toy example, most of these directions will be relatively well-aligned with the x-axis of course, but in general one cannot expected this to be the case). Once again, we have found that this metric behaves in a similar way as the the one introduced earlier: for a single particle, it often correctly captures a good value for $\ell \in [\![0, T]\!]$ but when summed over all particles, we get a non-decreasing function of ℓ, indicating that, in this form, TANG-ESJD might not be a good criterion to select an optimal T, if it exists.

A different approach is to define our mixture distribution to take into account the computational cost by construction. Recall that we have defined $\bar{\mu}$ to be a mixture with *uniform* weights. However, given $T+1$ mixture weights $\omega_k > 0$ that sum up to one, it is possible to define

$$\bar{\mu}(dz) = \sum_{k=0}^{T} \omega_k \mu^{\psi^{-k}}(dz).$$

One can still use $\bar{\mu}$ to estimate expectations with respect to μ, since if we define $\bar{h}(k,z) = h(\psi^k(z))$ we get

$$\bar{\mu}(\bar{h}) = \sum_{k=0}^{T} \omega_k \int h(\psi^k(z)) \mu^{\psi^{-k}}(dz) = \left[\sum_{k=0}^{T} \omega_k \right] \mu(h) = \mu(h),$$

and one can easily find that the these expectations can be estimated similarly to before

$$\mu(h) \approx \frac{\sum_{i=1}^{N} \sum_{k=0}^{T} h(\psi^k(Z^{(i)})) \omega_k \breve{w}_k(Z^{(i)})}{\sum_{i=1}^{N} \sum_{k=0}^{T} \omega_k \breve{w}_k(Z^{(i)})}.$$

There is a lot of freedom in the choice of these weights. One could choose an arithmetic or geometric progression, however, since the cost is linear, it seems reasonable to set

$$\omega_k \propto \frac{1}{k+1} \qquad k \in [\![0, T]\!].$$

That is, we are biasing the algorithm towards the beginning of the trajectory. This is because our goal, in this section, is to save computational expenses, however if good mixing was the priority, it is likely that $\omega_k \propto (T-k+1)^{-1}$ would be a better choice). Just as in our previous exposition (with $\omega_k = (T+1)^{-1}$) the normalization factor of the mixture weights disappears from both the expectations and ESJD expressions (of type B) and thus

$$\text{ESJD-NORM}_{\omega}(\ell) = \frac{1}{\ell} \sum_{k=0}^{\ell-1} \frac{\sum_{i=1}^{N} \left\| X_{\ell}^{(i)} \frac{\breve{w}_{\ell}(Z^{(i)})}{\ell+1} - X_k^{(i)} \frac{\breve{w}_k(Z^{(i)})}{k+1} \right\|^2}{\left(\sum_{\kappa=0}^{T} \sum_{i=1}^{N} \frac{\breve{w}_{\kappa}(Z^{(i)})}{\kappa+1} \right)^2}. \tag{3.25}$$

Once more, this metric is non-decreasing with ℓ as shown in Figure 3.36. In a sequential setting, one could select $T_n := \ell$ such that this metric tails off, for instance in this case $\ell \approx 25$ would

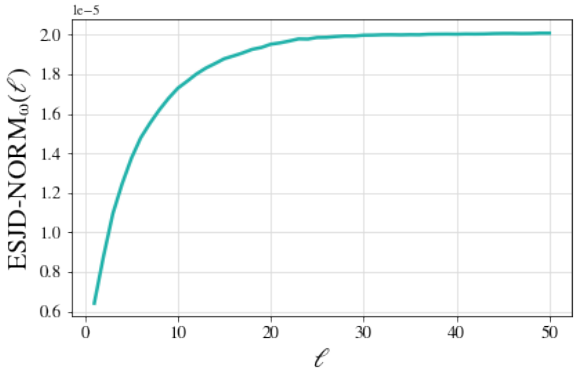

Figure 3.36: ESJD-NORM$_\omega$ in Equation (3.25) when using $\omega_k \propto (k+1)^{-1}$.

suffice, however we do not pursue this here, as we have found this criterion to be unreliable on preliminary experiments due to its high variability when T is small, although more work should be done.

Instead, the aim of this subsection was to illustrate our belief that it is not necessary to use ESJD in integrator snippets to select the number of integration steps. One immediate reason for this is that, as we have shown extensively in the experiments of the previous sections, integrator snippets are incredibly robust with respect to the choice of the parameter T. In the context of HMC samplers, it is precisely because HMC is very sensitive to the choice of its hyperparameters that it makes sense to devise better and better strategies to identify a good value for the number of Leapfrog steps, in order to save computational cost. We will return to this point in the next subsection, but now we provide one final reason as to why we believe ESJD-based metrics are sub-optimal criterions to adapt T in integrator snippets.

Perhaps the most important factor for the poor performance of the ESJD-based metrics that we have experimented with, is the global comparison performed *across* all trajectories at the resampling stage. Does a measure of "distance" still make sense as a criterion to select T when particles can be resampled across all trajectories? We do not believe so. A measure of distance in MCMC makes sense because sampling happens along chains and there is typically no resampling mechanism, unlike integrator snippets. Nevertheless, we shall not despair, as it turns out that a discrete distribution over the number of integration steps is available as a by-product of integrator snippets, and this can provide an important tool to optimise performance by computational cost.

3.8.3.2 How to adapt T: out-of-the-box discrete distribution over T

[150] and [32] learned discrete distributions over the number of integration steps and then sampled from these distributions at each iteration, to determine the current value of T. One of the appealing features of integrator snippets is that, at the end of each iteration, after the

resampling step in Algorithm 7, we have at our disposal N trajectory indices

$$\mathscr{K}_n = \left\{ k_n^{(i)} \in [\![0, T_{n-1}]\!] : i \in [\![N]\!] \right\},$$

where T_{n-1} is the number of integration steps used at iteration n. If N is large enough, then it is reasonable to assume that the discrete distribution with coefficients

$$\mho_k := \frac{\left| \left\{ i \in [\![N]\!] : k_n^{(i)} = k \right\} \right|}{N} \qquad k \in [\![0, T_{n-1}]\!],$$

provides sensible information about how far, on average, the particles are "travelling" along the trajectories. This distribution does not contain any information about computational cost, and in theory one could penalise these coefficients in many ways, such as

$$\tilde{\mho}_k \propto \mho_k \left(1 - \frac{k}{T+1} \right) \qquad k \in [\![0, T]\!].$$

However, we can include the computational cost *directly* into the algorithm, by choosing $\omega_k \propto (k+1)^{-1}$ for $k \in [\![0, T_{n-1}]\!]$, as shown earlier. A very simple approach to adapting the parameter T is to set it as follows

$$T_n = \min \left\{ \ell : \sum_{k=0}^{\ell} \mho_k \geq \beta N \right\} \qquad \beta \in (0, 1]. \tag{3.26}$$

When $\beta = 1$ we are simply setting T_n to the largest number trajectory index that has been resampled, meaning that curtail the those extra steps in the tail of the trajectory that contribute almost nothing to the ESJD (they could still have positive, but very small weights) but waste computational cost. One way to think about this simple strategy is that we are approximately reducing our trajectory to its β quantile.

We set up the following experiment to test these ideas. Using GHUMS we obtain a set of 5000 particles using $\delta = 0.2$ and $T_{\max} = 200$, without any adaptivity, until we reach $\epsilon_0 = 0.01$. This is done to obtain a set of well-distributed particles for η_{ϵ_0}. We then run four algorithms, all using only the THUG integrator, until $\epsilon_{\text{final}} = 10^{-6}$ starting with the same parameters $T_{\max}$ and δ. The first algorithm, which acts as a baseline, does not perform any adaptivity of the number of integration steps across iterations, and uses the standard choice $\omega_k = (T_n + 1)^{-1}$, where $T_n = T_{\max}$ for all iterations. The second algorithm uses the same uniform mixture weights but we adapt T_n using Equation (3.26) with $\beta = 1.0$. The third and fourth algorithm use the same adaptation strategy, but have mixture $\omega_k \propto (k+1)^{-1}$ and $\omega_k \propto (T_n - k + 1)^{-1}$ respectively.

At each iteration, we compute the Total ESJD-NORM (TEN) rescaled by the total number of integration steps (notice that we first multiply each term of the form in Equation (3.24) by ℓ, sum them up, and divide by T_n)

$$\text{TEN}_{n,\omega} = \frac{1}{T_n} \sum_{\ell=1}^{T_n} \sum_{k=0}^{\ell-1} \frac{\sum_{i=1}^{N} \left\| X_{n-1,\ell}^{(i)} \omega_\ell \breve{w}_{n-1,\ell}(Z_{n-1}^{(i)}) - X_{n-1,k}^{(i)} \omega_k \breve{w}_{n-1,k}(Z_{n-1}^{(i)}) \right\|^2}{\left(\sum_{\kappa=0}^{T_n} \sum_{i=1}^{N} \omega_\kappa \breve{w}_{n-1,\kappa}(Z_{n-1}^{(i)}) \right)^2}.$$

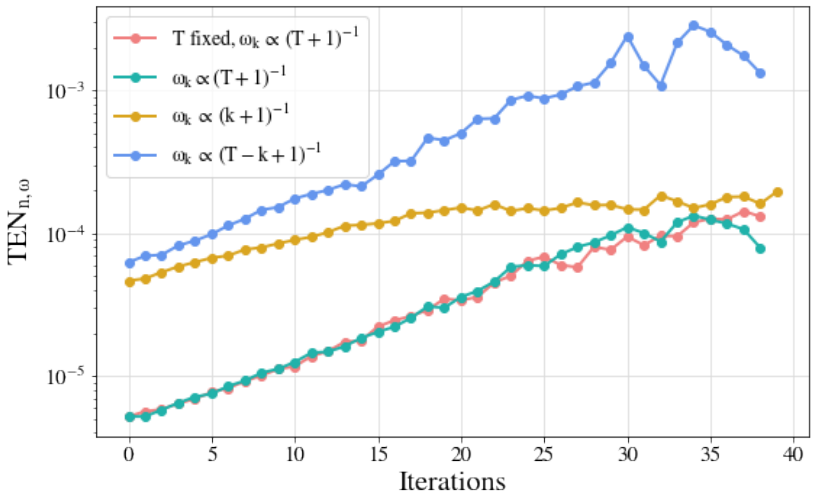

Figure 3.37: $\mathrm{TEN}_{n,\omega}$ for the four integrator snippets considered as a function of the iteration number.

This metric is displayed in Figure 3.37. Firstly, we notice that using $\beta = 1$ with $\omega_k \propto (T+1)^{-1}$ provides no improvement over our classic algorithm, however when we bias the algorithm towards smaller numbers of integration steps, we gain a considerable improvement at the beginning of the algorithm, which then steadily decreases as we progress. Surprisingly, using $\omega_k \propto (T-k+1)^{-1}$ provides consistently one order of magnitude better performance than our vanilla version! While this may sound at first counter-intuitive, it is in line with our findings so far about integrator snippets: their performance, even when normalised by computational cost, not only is very robust to the choice of T, but does not degrade as T increases (within reasonable values tested in this thesis, at least). This is a fundamental difference between integrator snippets and HMC samplers. The choice $\omega_k \propto (T_n - k + 1)^{-1}$ is biasing the algorithm towards resampling particles at the end of their trajectory, which is bringing an improvement that surpasses the additional computational cost of a long trajectory. The very simple adaptation rule in Equation (3.26) simply makes sure that we curtail this computational cost as much as possible, and we believe expect this to work well in practice.

3.8.3.3 Adaptive tuning with a mixture of integrators

When designing the GHUMS algorithm, we have shown that it is possible to use a mixture of two integrators. Of course this is generalizable to a mixture of $I \in \mathbb{Z}_+$ integrators, each with probability $p_\iota \in [0,1]$. It turns out, that each of these integrators can actually have a different number of integrator steps $T_\iota \in \mathbb{Z}_+$. Indeed, one can define the mixture

$$\bar{\mu}(dz) = \sum_{\iota=0}^{I-1} p_\iota \sum_{k=0}^{T_\iota} \omega_{\iota,k} \mu^{\psi_\iota^{-k}}(dz) \qquad \text{as long as} \qquad \sum_{k=0}^{T_\iota} \omega_{\iota,k} = 1 \quad \forall \iota \in [\![0, I-1]\!],$$

and it is easy to show that $\bar{\mu}(\bar{h}) = \mu(h)$ with $\bar{h}(i,k,z) = h(\psi_\iota^k(z))$. The weights are just as in the previous subsection. This means, for instance, that when using GHUMS we can use a single integration step for NHUG, and T integration steps in for THUG. In the context of adaptivity of the number of integration steps, this is helpful because it allows us to only adapt the number of integration steps for the THUG integrator, while keeping the number of integration steps for NHUG unchanged (of course though, all the metrics discussed so far should be computed only with the particles obtained using the THUG integrator). In general, if one has a mixture of integrators with different numbers of integration steps, then this will hinder parallelism since particles with shorter trajectories will take less time than particles with longer ones. However, since it is known in advance which is going to be used to construct the trajectory for each particle, one can batch together particles with the same integrator and proceed in lock-step.

3.8.3.4 An adaptive strategy inspired by the attention mechanism

It is natural to imagine an adaptive algorithm where we progressively learn the mixture ω_k in a similar style as [32]. One way to do so is to define our mixture as

$$\bar{\mu}(dz) \sum_{k=0}^{T} \omega_k \mu^{\psi^{-k}}(dz) \quad \text{where} \quad \omega_k \propto \tilde{\omega}_k c_k \quad \text{where} \quad c_k = \text{softmax}(U_k),$$

The idea is that the non-negative coefficients $\tilde{\omega}_k$ provides a-priori belief about the importance of the various mixture components, whereas c_k is learned and provides a-posteriori information. One could learn the U_k's by minimising the loss

$$\mathscr{L}(U_0,\ldots,U_T) = -\frac{\sum_{i=1}^{N}\sum_{k=0}^{T}\sum_{\ell=k+1}^{T}\left\| X_\ell^{(i)}\tilde{\omega}_\ell c_\ell \breve{w}_\ell(X^{(i)}) - X_k^{(i)}\tilde{\omega}_k c_k \breve{w}_k(X^{(i)})\right\|^2}{\left(\sum_{k=0}^{T}\sum_{i=1}^{N}\tilde{\omega}_k c_k \breve{w}_k(X^{(i)})\right)^2}$$

with respect to the parameters $U_0,\ldots,U_T$ using a gradient-based optimiser, such as ADAM. It is unclear whether it would be advantageous to use the weights c_k directly into the mixture or to have them as coefficients in the loss, as was done in Self-Tuning HMC [32]. It also seems plausible that the softmax function may be sub-optimal in the context of Integrator Snippets since it tends to concentrate the probability mass of the discrete distribution on a specific number of integrator steps, which may be a good strategy in HMC, but not necessarily in our context.

3.8.3.5 An adaptive strategy based on the U-turn criterion

Another simple idea, particularly relevant for the Leapfrog integrator, is to utilise a strategy based on the No-U-Turn criterion [71] as in Empirical HMC (EHMC) [150]. This seems natural because in Integrator Snippets, contrary to standard HMC, we always have available the full trajectories and the weights of each point on these trajectories, hence one can compute the *longest*

batches at a reasonable extra cost

$$L_{n-1}^{(i)} = \min\left\{\ell \in [\![0, T_n]\!] : \left(X_{n-1,\ell}^{(i)} - X_{n-1}^{(i)}\right)^\top V_{n-1,k} < 0\right\},$$

and use these to construct an empirical distribution

$$\hat{P}_L = \frac{1}{N}\sum_{i=1}^{N} \delta_{L_{n-1}^{(i)}}.$$

A potential advantage of this over the original paper, is that we would not need an initial training phase an could instead learn this distribution from one iteration to the next for the entire duration of the algorithm.

3.9 Extensions

Here we discuss extensions of this work that we leave to future work.

1. **Non-volume-preserving integrators**: The most obvious extension is for ψ to not be a volume-preserving transformation, which was assumed in Subsection 3.2.1. In this case, $\mu(h)$ in Subsection 3.2.2 can be written as follows

   $$\mu(h) = \frac{1}{T+1}\sum_{k=0}^{T}\int h(\psi^k(z))\frac{\mu(\psi^k(z))}{\eta(z)}|\det \mathrm{J}_{\psi^k}(z)|\eta(dz),$$

   and the rest of the theory holds, except that now the Jacobian determinants appear.

2. **Mixture of pushforward by arbitrary invertible transformations**: Throughout the thesis we have worked with $\bar{\mu}$ defined as the mixture of pushforwards $\mu^{\psi^{-k}}$, so that we used a single invertible transformation (or two, when working with a mixture of integrators) to define all pushforwards by simply iterating it. However, this is not necessary, and given a collection of (possibly unrelated) invertible transformations $\{\psi_k : k \in [\![0, T]\!]\}$ one could define the mixture

   $$\bar{\mu}(dz) = \frac{1}{T+1}\sum_{k=0}^{T}\mu^{\psi_k^{-1}}(dz).$$

   Then, writing $\bar{h}(k, z) := h(\psi_k(z))$ the remarkable property that we leveraged for PISA still holds, meaning that $\bar{\mu}(\bar{h}) = \mu(h)$ and one can write

   $$\mu(h) = \frac{1}{T+1}\sum_{k=0}^{T}\int h(\psi_k(z))\frac{\mu(\psi_k(z))}{\eta(z)}|\det \mathrm{J}_{\psi_k}(z)|\eta(dz).$$

3. **Multiple Integrators per particle**: A natural application of the extension above, that lends itself well to parallel machines, is to use multiple integrators per particle. Suppose,

for simplicity, that for each particle, we wish to obtain two different trajectories. This is a straightforward by setting

$$\psi_k = \begin{cases} \bar{\psi}^k & k \in [\![0, T/2-1]\!] \\ \tilde{\psi}^{k-\frac{T}{2}} & k \in [\![T/2, T]\!]. \end{cases}$$

4. **Normalizing Flows**: One of the strengths of integrator snippets is that if one has knowledge about the geometry of the problem, this can be leveraged by choosing the appropriate integrator. One could push this idea even further and learn ψ as the algorithm progresses. For example, by leveraging both of the extensions above, one could choose $\{\psi_k : k \in [\![0, T]\!]\}$ to be a family of parametrized diffeomorphisms defined by invertible neural networks whose Jacobian determinant can be easily computed, such as Normalizing Flows [79, 109]. There is a vast literature on augmenting sampling algorithms using generative models, and we simply highlight how our framework could benefit naturally from such ideas [89, 131] One could grab the intermediary states of the neural network to be $\psi_k(z)$, and this would lead to an algorithm similar in spirit to Distilled Importance Sampling [113].

5. **Non-Lebesgue Dominating Measure**: In all of our applications the dominating measure was a Lebesgue or the product of a Lebesgue and counting measure. However, one could easily generalise this to the scenario where μ and η are dominated by a non-Lebesgue measure υ, in which case we would have

$$\mu(h) = \frac{1}{T+1} \sum_{k=0}^{T} \int h(\psi_k(z)) \frac{\frac{d\mu}{d\upsilon}(\psi_k(z))}{\frac{d\eta}{d\upsilon}(z)} \frac{d\upsilon^{\psi_k^{-1}}}{d\upsilon}(z) \eta(dz).$$

6. **State-dependent non-uniform mixture weights**: The latter idea can be pushed even further by having the mixture weights to be a function of $z \in \mathsf{Z}$, i.e. $\omega_k(z)$. Then if one defines $\check{h}(k,z) = h(\psi_k(z))/\omega_k(z)$ (which is always well-defined whenever the weights are strictly positive), we have once more $\bar{\mu}(\check{h}) = \mu(h)$.

7. **Partial Velocity Refreshment**: When the targets decompose as $\mu = \pi \otimes \varpi$, where ϖ is the distribution of an auxiliary velocity variable, one can perform partial refreshment [73]. More generally, one can define a conditional update on the auxiliary variables

$$R(z, dz') = \delta_x(dx')\varpi(dv' \mid v),$$

as long as $\varpi(dv' \mid v)$ is a valid regular conditional distribution corresponding to a joint on $(\mathsf{V}^2, \mathcal{V}^{\otimes 2})$ with marginal $\varpi(dv)$, and then it is easy to show that on measurable rectangles $\mathsf{B} \times \mathsf{C}$ with $\mathsf{B} \in \mathscr{X}$ and $\mathsf{C} \in \mathcal{V}$ we have $\mu R(\mathsf{B} \times \mathsf{C}) = \mu(\mathsf{B} \times \mathsf{C})$. One could devise strategies to combine this with a non-reversible update of the underlying uniform random variables used to perform the resampling step, in the spirit of [105].

3.10 Related Work

In this section, we discuss work related to Markov Snippets, and in particular we focus on two main branches of the literature that are relevant. The first one, much more modest in size, contains similar work using and recycling snippets generated with Markov kernels or integrators. The second branch of the literature contains work regarding the robustness of HMC-like samplers with respect to their tuning parameters. These works are relevant to us for two reasons: firstly, the construction of Hamiltonian trajectories by obtained iterating a reversible integrator shares two main challenges with integrator snippets, i.e. the selection of the step size and the number of integration steps. Secondly, one of the most natural and appealing applications of Integrator Snippets consists in using an integrator of Hamiltonian dynamics, as was done in the precursor work [154].

3.10.1 Literature on Recycling and Windows of States

[103, 104] introduced the idea that using windows of states in Hamiltonian trajectories would help with the acceptance probability. This is, to our knowledge, the closest work to Integrator Snippets particularly when ψ is the Leapfrog integrator of Hamilton's equations. Windowed-HMC (see Figure 3.38) fundamentally maps from μ to $\bar{\mu}$, performs a MH targeting $\bar{\mu}$, and maps the result back to μ. The author reports considerable gains on a Gaussian distribution toy example with different scales, with the improvement increasing with dimensionality. Notice that an important difference in our work is that we show how it is not necessary to map back and forth between the two spaces $(\mathsf{Z},\mathcal{Z})$ and $(\mathsf{Z}^{T+1},\mathcal{Z}^{\otimes T+1})$. Hence, the idea of recycling states along a trajectory is not new, but we generalise to new, unexplored directions.

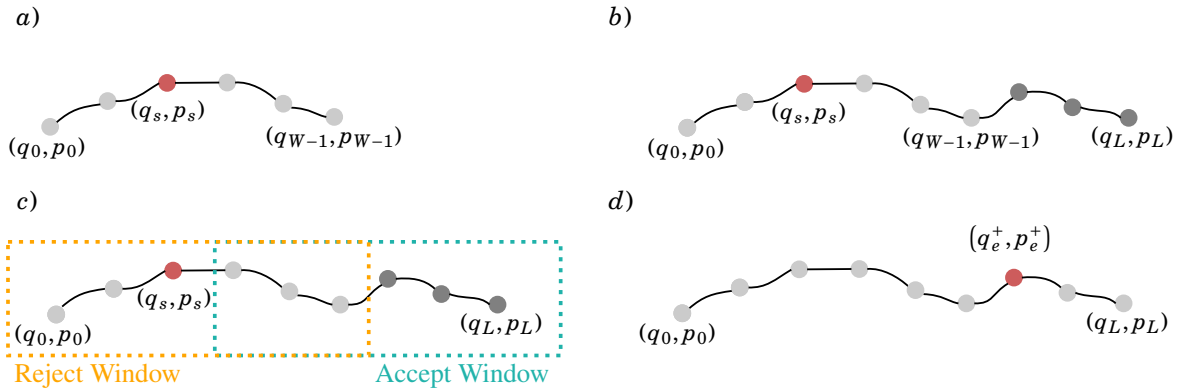

Figure 3.38: Illustration of the Windowed Hamiltonian Monte Carlo method.

Waste-Free SMC [37] is another work that is related to Markov Snippets. The central idea of the paper is that recycling the intermediary states obtained in an SMC sampler using an MCMC forward kernel. Importantly, they show strong empirical results that recycling intermediary states especially when the forward kernel is mixing poorly, such as in rare-estimation problems.

As shown in our experiments, this is also a behavior we have observed when sampling from filamentary distributions using Integrator Snippets - the ability to use the intermediary states confers robustness to the sampler.

Non-Equilibrium Orbits (NEO) [138] constructs a finite mixture of pushforwards by a diffeomorphism (in their case, they pushforward both forward and backward), of a reference distribution. They then use this mixture to construct an importance sampling estimator of the normalising constant of the target. Perhaps the biggest difference in our work is that we *target* a mixture of pushforward distributions, rather than use the mixture as our proposal. More precisely, they sample $Z \sim \mu$ and $k \sim \mathcal{U}([\![0, T]\!])$ and then generate $\psi^k(z) \sim \bar{\mu}$, whereas we do not assume we can sample from μ. NEO could therefore be used as a post-processing step to our algorithm.

3.10.2 Literature on Robustness in HMC-like samplers

HMC has two tuning parameters: the number of integration steps and the integration step size, and we shall refer to their product as the trajectory length or total integration time. Tuning these parameters is essential for the good performance of the algorithm, and a lot of research has focused on their adaptivity or in deriving HMC variants that are more robust to these parameters. The step size determines the discretization error and hence, contributes to the acceptance rate of the algorithm. For this reason, it is common to adapt the step size using Adaptive MCMC techniques based on Stochastic Approximation (described in [5]) or primal-dual averaging [106]. This latter is the approach, for instance, of the popular No-U-Turn Sampler (NUTS) algorithm [71], discussed later.

3.10.2.1 Robustness through persistence

One of the earliest attempts at making HMC more robust was described in [73] and consists in partially refreshing the momentum. The idea is to further reduce Random-Walk behavior by making the algorithm more persistent. However, contrary to its aim, this variation often results in poor performance since it requires considerably smaller step sizes, because it aims to avoid rejections, which would cause the algorithm to backtrack. HMC algorithms partially-refreshing their momenta are known as Generalised HMC (GHMC). Recently, [105] noticed that, by means of slice sampling, it is possible to non-reversibly update the uniform random variable used in the MH step and that, while this does not generally change the overall acceptance probability, it often clusters acceptances and rejections together, which renders GHMC more competitive with HMC. One downside of this approach, is that parameters are generally harder to tune, since the interpretation of the acceptance probability is not as straightforward. [72] offers parallelism-friendly heuristics for tuning GHMC. Specifically, they tune the step size by estimating the largest eigenvalue of the Hessian of the score.

3.10.2.2 Robustness through jittering

[95] suggested randomising the step size to alleviate issues with ergodicity and reduce autocorrelations. The step-size was typically sampled from $\mathcal{U}(\delta_{\min}, \delta_{\max})$, see discussion in [103]. Randomised HMC (RHMC) [21] proposed instead to randomise the total integration times $\tau = \delta T$, which are taken to be IID exponential random variables. Although in their paper they assume the step size is small enough that all proposals get accepted, this method is widely used in practice.

3.10.2.3 Adapting hyperparameters using measures of ESJD

Adapting the step size in HMC-type algorithms is often easier than adapting the number of integration steps and, as mentioned previously, one typically targets an acceptance probability of 65.1% [14] using stochastic approximation. Tuning T, the number of integration steps, is trickier. A relatively large number of integration steps is necessary to suppress random walk behavior, but if the dynamic is integrated for too long, the algorithm may make a U-turn and thus we may waste important computational effort. For unimodal targets, it is easy to build an intuition for this.

Adaptive HMC (AHMC) [156] adapts both the step size and the number of integration steps so as to maximise the ESJD [57] normalised by $\sqrt{T}$, to take into account computing time. They perform this optimization using Bayesian Optimization.

No-U-Turn Sampler (NUTS) [71] was developed to provide a tuning-free alternative, and has become a workhorse of Bayesian inference with popular implementations in Stan [136] and TensorFlow [1]. NUTS keeps doubling its integration window forwards or backwards until the integrator starts turning backwards, meaning that the trajectory points are getting closer to the starting point. The idea is to avoid U-turns and thus travel further. The U-turn criterion was motivated by the goal of maximising the ESJD [70]. [17] generalised this algorithm to Riemann Manifolds. One disadvantage of NUTS is that it does not bode well in parallel architecture due to its complex control flow and random number of integration steps. More precisely, since the algorithm dynamically sets the integration time by detecting U-turns, the total runtime per iteration is inherently unpredictable. This leads to synchronization overhead: faster threads (i.e. threads that have detected a U-turn with a smaller number of integration steps) have to wait for slower threads and this wastes computation. [116] devised clever strategies to use batching in the context of control-intensive programs such as NUTS, which are now implemented in TensorFlow Probability [42, 137]. However, this and other SIMD-friendly implementations of NUTS require significantly more runtime and overhead and come at a loss compared to well-tuned HMC methods with a static number of integration steps [70]. The latter work, suggests to select the step size using dual-averaging and sets the integration time by maximising the ChEES criterion, which is a form of generalised ESJD, dependent on a test function h

$$\mathrm{ESJD}(h) = \mathbb{E}\left[\left\|h(X') - h(X)\right\|^2\right].$$

In the display above, X is the current state of the Markov Chain and X' is the proposal. The ChEES criterion then corresponds to $f(X) = \|X - \mathbb{E}[X]\|^2/2$ [132]. This metric is maximised using ADAM [77] and the noise of the jittered integration times is controlled using quasi-random Halton sequences [108]. Importantly, they notice that jittering has an important effect since it allows to "average out" autocorrelations, which is a behavior that we also observe in Integrator Snippets. Maximising this criterion, is equivalent to minimising the autocorrelation of the second moments of the chain, and favours high-variance directions. This is in contrast with [57] maximising the ESJD of a MH sampler, which produces good results for the particular test functions corresponding to the posterior mean, but not as good for non-linear test functions such as second moments [70]. The authors of ChEES remark that maximising the ESJD for the posterior means leads to unnecessarily large trajectories since this metric is sensitive to anti-correlated samples. One of the shortcomings of ChEES is that, contrary to AHMC, it does not incorporate information about the computing time in its metric, and this can fool the Stochastic Approximation updates to overshoot the value of T.

Stochastic Gradient Ascent HMC (SGA-HMC) utilises a more robust criterion, known as SNAPER, which consists in a (centered) ESJD criterion along a the direction of greatest variance v, normalized by integration time δT

$$\text{SNAPER} = \frac{1}{4\delta T}\mathbb{E}\left[\left\|(X - \mathbb{E}[X])^\top v - (X' - \mathbb{E}[X])^\top v\right\|^2\right],$$

where the direction v is obtained by Principal Component Analysis. The step size is adapted using ADAM to guide the harmonic mean of the acceptance rates towards a target acceptance probability. The total integration time $\tau = \delta T$ is also updated using ADAM but using the objective function above, and at each iterations its values is jittered uniformly. One of the advantages of this criterion over ChEES is that it tends to have one well-defined peak, and hence is well-suited to optimization. Using this criterion assumes that good exploration along the direction of highest variance will automatically lead to good exploration in lower variance directions. However, if in these directions the distribution is particularly complicated, thus requiring longer integration times than obtained by ADAM, the method may perform sub-optimally.

More recently, Self-Tuning HMC [32] constructs an objective function closely-related to the ESJD but that takes into account the computational cost of building the trajectory, and uses an attention-like [144] mechanism to emphasise suitable integration times. During a training phase, the coefficients of an empirical distribution over the number of integration steps are learned via gradient-based optimization. More specifically, they learn the step size δ and the parameters $c_1, \ldots, c_T$ of a discrete probability distribution over the number of integration steps (where $T \in \mathbb{Z}_+$ is user-defined, and represents the maximum number of integration steps considered) using ADAM to minimise the following loss function

$$L(\delta, c_{1:T}) = -\frac{1}{dE}\sum_{k=1}^{T}\frac{c_k}{k}a_{\text{MH}}(X_0, X_k')\left\|X_k' - X_0\right\|^b \qquad b \geq 1,$$

where $d \in \mathbb{Z}_+$ is the dimensionality of the state space, $E \in \mathbb{Z}_+$ is the number of epochs used during training. Here $X_0 \in \mathbb{R}^d$ is the current state of the Markov Chain and X'_k is obtained by integrating Hamilton's equations for k steps with jittered step size $\delta' \sim \mathcal{N}(\delta, s\delta)$ using scale $s > 0$. The term $a_{\text{MH}}(X_0, X'_k)$ is the MH acceptance probability. The values c_k represent how likely k is to be a good number of integration steps and are defined as $c_k = \text{softmax}(C_k)$, with C_k initialised from $\mathcal{U}(0,1)$. They observe that jittering the step size has a significant impact in smoothing out the loss landscape. Additionally, they observed that $b = 4$ was superior to the standard choice of $b = 2$ and provided a better proxy for the autocorrelation. After the training phase, one samples k from the discrete distribution with weights c_k to determine the number of integration steps.

This method is loosely related to Empirical HMC (EHMC) [150], where they also construct an empirical distribution over the number of integration steps in a training phase, which is then sampled from at runtime. However, in this earlier work, the distribution is a *uniform* mixture of Dirac deltas located at the $K \in \mathbb{Z}_+$ *longest batches* found during the training phase. Each of the longest batches is the smallest number of integration steps such that NUTS' U-turn condition is satisfied

$$\text{longest-batch}\,(\kappa) = \min\left\{k \in [\![L_0]\!] : (X_k^{(i)} - X_0^{(i)})^\top V_k < 0\right\} \qquad \kappa \in [\![K]\!].$$

3.10.2.4 Resampling in SMC samplers

Multinomial, stratified and systematic resampling are three of the most popular resampling schemes used in SMC samplers and particles filters. A naive implementation of multinomial resampling is $\mathcal{O}(NM)$, where N is the number of weights and M the number of sampled indices, and NumPy's [67] `random.choice` is $\mathcal{O}(N + M \log N)$, however it is possible to achieve $\mathcal{O}(N + M)$ by ordering the random variates, which is still a valid resampling algorithm as long as the downstream estimators are order-invariant, which is typically the case in SMC samplers [31]. When implemented correctly, systematic and stratified resampling schemes are $\mathcal{O}(N + M)$ [20]. Stratified resampling estimates have lower variance than multinomial resampling, and although systematic resampling in practice often performs better than multinomial resampling and is faster than stratified resampling, it does not have as much theoretical support [31]. Recently, [58] introduced Srinivasan Sampling Process (SSP) and proved it to be consistent (a property that systematic resampling does not posses), meaning that the associated empirical distribution converges weakly to the true underlying probability distribution. It also shares the $\mathcal{O}(N + M)$ computational complexity. The resampling step is considered the bottleneck of SMC samplers as they typically require the cumulative sum of the weights, known as *prexif sum* [102], and the same applies, of course, to integrator snippets. While there has been work that exploited parallel versions of prefix sums [61], many Single Instruction, Multiple Data (SIMD) architectures, such as those employed by modern GPUs, are significantly faster when using single-digit precision and, when N is large enough, prefix sums can lead to numerical instabilities [102]. In order to find a paralellisable alternative that does not suffer from numerical instabilities, the latter work

suggested running an independence MH sampler or a rejection sampler on each thread. These are fully-paralellisable meaning that they can run on separate thread and do not require the weights to be normalised (although the rejection sampler requires an upper bound, as usual). Future work could explore using such resampling schemes in the context of integrator snippets run on parallel architectures.

3.11 Future Directions and Conclusion

3.11.1 Approximate Manifold Sampling

Theorem 2.4 says that the change in norm of the velocity in the α-THUG sampler, with $\alpha > 0$ depends linearly on the step size. Although we have shown experimentally that the impact of this on the acceptance ratio is typically overshadowed by the change in potential, it would still be an important avenue of research to understand its impact more thoroughly. More generally, additional work is required to understand the theoretical underpinnings of samplers for Approximate Manifold Sampling and currently we are collaborating with Professor Jesus Sanz-Serna towards developing a better understanding of the THUG integrator. Further experiments are needed in order to compare α-THUG with Metropolis-unadjusted samplers based on the discretization of SDEs on manifolds, and in order for these comparisons to be robust, new metrics of performance, specifically tailored for the task of approximate manifold sampling, should be developed.

3.11.2 Markov and Integrator Snippets

Markov and Integrator Snippets are still in their infancy yet in this work we have covered many of the foundational aspects such as sufficiency and necessary conditions, as well as various metrics to assess their performance, such as the ESJD in Subsection 3.4.2. An important avenue of research would be to develop valid ESS metrics for Markov and Integrator snippets, and this is currently the subject of joint work with Professor Christophe Andrieu. In this thesis, we have only provided a glimpse of Integrator Snippets' potential for adaptation, and it was mostly based on heuristics, we expect the development of adaptation strategies grounded in theory to become a popular topic of research among practitioners and theorists of the sequential Bayesian learning community, due to the high degree of flexibility of this new class of inference algorithms. All of our examples in Chapter 3 were artificial and their purpose was to study several of the appealing qualities and advantages of Integrators Snippets over standard SMC samplers, particularly their robustness to the choice of their hyperparameters. We expect the gains on real-world problems to be comparable to those obtained on these toy examples, and the application of Integrator Snippets to leverage a-priori knowledge of an inference problem by careful selection of the deterministic (or stochastic) updates is subject of imminent work. Finally, since in practice Integrator Snippets' only requirement is the definition of an appropriate family of invertible deterministic transformations, we expect parallel and highly-optimised, off-the-shelf

implementations to be extremely accessible and provide practitioners with a flexible, robust, and powerful new tool to tackle sequential Bayesian problems in machine learning.

APPENDIX A

APPENDIX FOR INTRODUCTORY CHAPTER

A.1 Measure Theory

Theorem A.1 (Change of Variables)**.** *Let* $(\mathsf{Z}, \mathcal{Z}, \mu)$ *be a measure space,* $(\mathsf{Y}, \mathcal{Y})$ *be a measurable space, and* $\psi : \mathsf{Z} \to \mathsf{Y}$ *be a* $\mathcal{Z}$*-measurable function. A* $\mathcal{Y}$*-measurable function* $h : \mathsf{Y} \to \mathbb{R}$ *is integral with respect to* μ^{ψ} *if and only if* $h \circ \psi$ *is integrable with respect to* μ*, in which case*

$$\int_{\mathsf{Y}} h \; d\mu^{\psi} = \int_{\mathsf{Z}} h \circ \psi \; d\mu. \tag{A.1}$$

A famous special case of this theorem is found if ψ is a diffeomorphism with Jacobian J_{ψ} and it acts between open subsets of an Euclidean space.

Corollary A.1 (Change of Variables by a Diffeomorphism)**.** *Let* U *and* V *be open subsets of* $\mathbb{R}^d$*,* $\psi : U \to V$ *be a diffeomorphism with Jacobian matrix* J_{ψ}*, and* λ *denote the* d*-dimensional Lebesgue measure on* $(\mathbb{R}^d, \mathscr{B}(\mathbb{R}^d))$*. Then for any measurable function* $h : V \to \mathbb{R}$*,* h *is* λ*-integrable if and only if* $h \circ \psi$ *is* $|\det \mathrm{J}_{\psi}|\lambda$*-integrable, in which case*

$$\int_V h(x)\lambda(dx) = \int_U h \circ \psi(x) |\det \mathrm{J}_{\psi}(x)| \lambda(dx). \tag{A.2}$$

If instead one considers $\psi^{-1} : V \to U$ *and measurable function* $h \circ \psi : U \to \mathbb{R}$ *then* $h \circ \psi$ *is* λ*-integrable if and only if* h *is* $|\det \mathrm{J}_{\psi^{-1}}|\lambda$*-integrable, in which case*

$$\int_U h \circ \psi(x)\lambda(dx) = \int_V h(x) |\det \mathrm{J}_{\psi^{-1}}(x)| \lambda(dx). \tag{A.3}$$

One of the interesting takeaways from the corollary above is that one finds expressions for the pushforwards of the Lebesgue measure

$$\lambda^{\psi^{-1}} = |\det \mathrm{J}_{\psi}| \lambda$$
$$\lambda^{\psi} = |\det \mathrm{J}_{\psi^{-1}}| \lambda.$$

Next, we reproduce a simplified version of the disintegration theorem that is sufficiently rigorous for our purposes, while keeping the narration simple. For more precise Theorems we refer the reader to [29]. We shall borrow the simplified version of the Disintegration Theorem in [35] (Theorem 2.18 in the reference) which applies to probability measures in nice spaces (see page 11 for a definition), since we will mostly be concerned with $(\mathbb{R}^d, \mathscr{B}(\mathbb{R}^d))$ and these are nice spaces for any $d \in \mathbb{Z}_+$.

Theorem A.2 (Simplified Disintegration Theorem)**.** *Let μ be a probability measure on $(\mathsf{Z}, \mathscr{Z}) = (\mathsf{X} \times \mathsf{Y}, \mathscr{X} \otimes \mathscr{Y})$ and let $(\mathsf{X}, \mathscr{X})$ be a nice space. Then there exists a probability measure π on $(\mathsf{X}, \mathscr{X})$ and a transition probability kernel $M : \mathsf{X} \times \mathscr{Y} \to [0,1]$ such that*

$$\mu(\mathsf{A} \times \mathsf{B}) = \int_{\mathsf{A}} M(x, \mathsf{B}) \pi(dx) = \int_{\mathsf{A} \times \mathsf{B}} M(x, dy) \pi(dx) \qquad \forall\, \mathsf{A} \in \mathscr{X},\, \forall \mathsf{B} \in \mathscr{Y}.$$

Notice that when also $(\mathsf{Y}, \mathscr{Y})$ is a nice measurable space, the Theorem above implies that there exist another probability measure υ on $(\mathsf{Y}, \mathscr{Y})$ and a transition probability kernel $L : \mathsf{Y} \times \mathscr{X} \to [0,1]$ such that

$$\mu(\mathsf{A} \times \mathsf{B}) = \int_{\mathsf{A} \times \mathsf{B}} L(y, dx) \upsilon(dy) \qquad \forall\, \mathsf{A} \in \mathscr{X},\, \forall \mathsf{B} \in \mathscr{Y}.$$

By the (uniqueness part) of the Radon-Nikodym theorem, we know that L and M are Radon-Nikodym derivatives

$$L(y, dx) = \frac{\mu(dx, dy)}{\upsilon(dy)}$$
$$M(x, dy) = \frac{\mu(dx, dy)}{\pi(dx)}$$

and combining these two results we find

$$L(y, dx) = \frac{\pi(dx) M(x, dy)}{\upsilon(dy)}. \tag{A.4}$$

We make a subtle but important remark. One may think the definition above is ill-defined since the Radon-Nikodym theorem requires both the dominated and dominating measure to be on the same space, but here $\mu = \pi \otimes M$ is on on the joint space $(\mathsf{Z}, \mathscr{Z})$ whereas υ is on the marginal space $(\mathsf{Y}, \mathscr{Y})$. This conundrum is readily solved by resorting to more rigorous version of the disintegration theorem, but in summary one can define the projection functions $P_{\mathsf{Y}} : \mathsf{X} \times \mathsf{Y} \to \mathsf{Y}$ and $P_{\mathsf{X}} : \mathsf{X} \times \mathsf{Y} \to \mathsf{X}$ such that $P_{\mathsf{Y}}(x, y) = y$ and $P_{\mathsf{X}}(x, y) = x$ have define

$$\pi = \mu^{P_{\mathsf{X}}}$$
$$\upsilon = \mu^{P_{\mathsf{Y}}},$$

and therefore rewrite Equation (A.4) as

$$L(y, dx) = \frac{\mu^{P_{\mathsf{X}}}(dx) M(x, dy)}{\mu^{P_{\mathsf{Y}}}(dy)}.$$

A.2 Derivation of the Effective Sample Size

To derive the formula for the ESS we just need to derive an expression for the variance of the Importance Sampling estimator

$$\hat{h} = \frac{\frac{1}{N}\sum_{i=1}^{N} h(X^{(i)})w(X^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} w(X^{(i)})} \qquad X^{(i)} \sim \eta.$$

For ease of exposition, we define the following random variables for any $i \in [\![N]\!]$

$$W_i = \frac{\mu(X^{(i)})}{\eta(X^{(i)})}$$
$$Z_i = h(X^{(i)})\frac{\mu(X^{(i)})}{\eta(X^{(i)})}$$

and then we notice that the estimator (when viewed as a random variable) can be written as the ratio of samples means $\bar{Z}$ and $\bar{W}$ (which are themselves random variables)

$$\hat{H}_i = \frac{\frac{1}{N}\sum_{i=1}^{N} Z_i}{\frac{1}{N}\sum_{i=1}^{N} W_i} := \frac{\bar{Z}}{\bar{W}}$$

Notice that the sample means are unbiased

$$\mathbb{E}[\bar{Z}] = \mathbb{E}[Z] \tag{A.5}$$

$$\mathbb{E}[\bar{W}] = \mathbb{E}[W], \tag{A.6}$$

and since X_i's are independent samples of η, both $\{Z_1,\dots,Z_N\}$ and $\{W_1,\dots,W_N\}$ are sets of independent samples, and therefore

$$\mathbb{V}[\bar{Z}] = \frac{1}{N^2}\sum_{i=1}^{N}\mathbb{V}[Z_i] = \frac{1}{N}\mathbb{V}[Z]$$
$$\mathbb{V}[\bar{W}] = \frac{1}{N^2}\sum_{i=1}^{N}\mathbb{V}[W_i] = \frac{1}{N}\mathbb{V}[W].$$

Since the estimator $\hat{H}$ is a ratio of two random variables, we give a name to this function

$$\hat{H} = g(\bar{Z},\bar{W}) \qquad \text{where} \quad g(\bar{Z},\bar{W}) = \frac{\bar{Z}}{\bar{W}},$$

and we Taylor expand the estimator around the mean (we write $\theta = (\mathbb{E}[\bar{Z}],\mathbb{E}[\bar{W}])$ for simplicity)

$$\hat{H} \approx g(\mathbb{E}[\bar{Z}],\mathbb{E}[\bar{W}]) + (\bar{Z}-\mathbb{E}[\bar{Z}])\frac{\partial g(\theta)}{\partial \bar{Z}} + (\bar{W}-\mathbb{E}[\bar{W}])\frac{\partial g(\theta)}{\partial \bar{W}}$$

Taking the variance of this Taylor expansion we get

$$\mathbb{V}[\hat{H}] \approx \left(\frac{\partial g(\theta)}{\partial \bar{Z}}\right)^2 \mathbb{V}[\bar{Z}] + \left(\frac{\partial g(\theta)}{\partial \bar{W}}\right)^2 \mathbb{V}[\bar{W}] + 2\frac{\partial g(\theta)}{\partial \bar{Z}}\frac{\partial g(\theta)}{\partial \bar{W}}\mathbb{E}[(\bar{Z}-\mathbb{E}[\bar{Z}])(\bar{W}-\mathbb{E}[\bar{W}])].$$

The partial derivatives evaluated at θ are (note $\mathbb{E}[W]=1$)

$$\frac{\partial g(\theta)}{\partial \bar{Z}} = 1 \qquad\qquad \frac{\partial g(\theta)}{\partial \bar{W}} = -\mathbb{E}[Z],$$

therefore the expression for the variance then becomes (writing $\mu(h)=\mathbb{E}[Z]$)

$$\mathbb{V}[\hat{h}] = \frac{1}{N}\mathbb{V}[Z] + \frac{\mu(h)^2}{N}\mathbb{V}[W] - 2\mu(h)\mathrm{Cov}(\bar{W},\bar{Z}).$$

Now, for the covariance notice we can write it as

$$\begin{aligned}
\mathrm{Cov}(\bar{W},\bar{Z}) &= \mathbb{E}[\bar{W}\bar{Z}] - \mathbb{E}[W]\mathbb{E}[Z] \\
&= \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} W_i\right)\left(\frac{1}{N}\sum_{i=1}^{N} Z_i\right)\right] - \mathbb{E}[W]\mathbb{E}[Z] \\
&= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{E}[W_i Z_j] - \mathbb{E}[W]\mathbb{E}[Z] \\
&= \frac{N-1}{N}\mathbb{E}[W]\mathbb{E}[Z] + \frac{1}{N}\mathbb{E}[WZ] - \mathbb{E}[W]\mathbb{E}[Z] \\
&= \frac{1}{N}\mathbb{E}[WZ] - \frac{1}{N}\mathbb{E}[W]\mathbb{E}[Z] \\
&= \frac{1}{N}\mathrm{Cov}(W,Z),
\end{aligned}$$

which leads to the following

$$\mathbb{V}[\hat{H}] \approx \frac{1}{N}\left[\mu(h)^2\mathbb{V}[W] + \mathbb{V}[Z] - 2\mu(h)\mathrm{Cov}(W,Z)\right].$$

We wish to write the expression for $\mathbb{V}[\hat{H}]$ as a product of the variance of the ordinary Monte Carlo estimator and a function of $\mathbb{V}[W]$, so that $\mathbb{V}[\hat{H}_{\mathrm{MC}}]$ cancels out in the formula for the ESS. To achieve this, we write $\mathbb{V}[Z]$ and $\mathrm{Cov}(W,Z)$ in terms of $\mathrm{Cov}_\mu[W,H]$ with the aim to cancel out the dependence of the formula on the test function. Firstly, notice

$$\mathrm{Cov}(W,Z) = \mathbb{E}[WZ] - \mathbb{E}[W]\mathbb{E}[Z] = \mathbb{E}_\mu[HW] - \mu(h) = \mathrm{Cov}_\mu[W,H] + \mu(h)\mathbb{E}_\mu[W] - \mu(h), \tag{A.7}$$

and then expand $\mathbb{V}[Z]$ in a similar way,

$$\mathbb{V}[Z] = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2 = \mathbb{E}_\mu[H^2W] - \mu(h)^2,$$

where we have written $H=h(Y)$, where $Y\sim\mu$, as a random variable. We use the Delta method on $\mathbb{E}_\mu[H^2W]$ in order to obtain a $\mathrm{Cov}_\mu(W,H)$ term that would cancel out the one in Equation (A.7)

$$\mathbb{E}_\mu[H^2W] \approx \mu(h)^2\mathbb{E}_\mu[W] + \mathbb{V}_\mu[H]\mathbb{E}_\mu[W] + 2\mu(h)\mathrm{Cov}_\mu(H,W).$$

The variance of the importance sampling estimator now becomes

$$\begin{aligned}
\mathbb{V}[\hat{H}] &\approx \frac{1}{N}\left[\mu(h)^2\mathbb{V}[W] + \mathbb{V}[Z] - 2\mu(h)\mathrm{Cov}(W,Z)\right] \\
&\approx \frac{1}{N}\left[\mu(h)^2\mathbb{V}[W] + \mathbb{E}_\mu[H^2W] - \mu(h)^2 - 2\mu(h)\left(\mathrm{Cov}_\mu[W,H] + \mu(h)\mathbb{E}[W] - \mu(h)\right)\right] \\
&\approx \frac{1}{N}\left[\mu(h)^2\mathbb{V}[W] + \mu(h)^2\mathbb{E}_\mu[W] + \mathbb{V}_\mu[H]\mathbb{E}_\mu[W] - \mu(h)^2 - 2\mu(h)^2\mathbb{E}_\mu[W] + 2\mu(h)^2\right] \\
&\approx \frac{1}{N}\left[\mu(h)^2\mathbb{V}[W] - \mu(h)^2\mathbb{E}_\mu[W] + \mathbb{V}_\mu[H]\mathbb{E}_\mu[W] + \mu(h)^2\right]
\end{aligned}$$

Finally, we use the fact that

$$\mathbb{E}_\mu[W] = \mathbb{E}[W^2] = \mathbb{V}[W] + 1,$$

which gives

$$\begin{aligned}
\mathbb{V}[\hat{H}] &\approx \frac{1}{N}\left[\mu(h)^2\mathbb{V}[W] - \mu(h)^2 - \mu(h)^2\mathbb{V}[W] + \mathbb{V}_\mu[H] + \mathbb{V}_\mu[H]\mathbb{V}[W] + \mu(h)^2\right] \\
&\approx \frac{1}{N}\mathbb{V}_\mu[H](1 + \mathbb{V}[W]) \\
&\approx \mathbb{V}[\hat{H}_{\mathrm{MC}}](1 + \mathbb{V}[W]).
\end{aligned}$$

The ESS is then

$$\mathrm{ESS} \approx \frac{N}{1 + \mathbb{V}[W]}. \tag{A.8}$$

We can approximate the variance of the weights using importance sampling

$$\begin{aligned}
\mathbb{V}[W] &= \mathbb{E}[W^2] - 1 \\
&= \left(\frac{\mathcal{Z}_\eta}{\mathcal{Z}_\mu}\right)^2 \int \left(\frac{\tilde{\mu}(x)}{\tilde{\eta}(x)}\right)^2 \eta(dx) - 1 \\
&\approx \frac{\frac{1}{N}\sum_{i=1}^N w(X^{(i)})^2}{\frac{1}{N^2}\left(\sum_{i=1}^N w(X^{(i)})\right)^2} - 1.
\end{aligned}$$

Plugging this into the expression for the ESS we get the usual formula

$$\mathrm{ESS} \approx \frac{\left(\sum_{i=1}^N w(X^{(i)})\right)^2}{\sum_{i=1}^N w(X^{(i)})^2}$$

A.2.1 Remainder expression

Notice that the third-order partial derivatives are all zero except $\partial_w\partial_h^2$ which means that the remainder consists only of a single third-order term

$$R(W,H) = \mathbb{E}_\mu\left[(W - \mathbb{E}_\mu[W])(H - \mu(h))^2\right]$$

A.3 Derivation of ESJD

An alternative form of the ESJD can be found by plugging in the definition of P

$$\begin{aligned}
\mathbb{E}_{\pi\otimes P}[\|Y-X\|^2] &= \int \|x-y\|^2 \pi(dx)\otimes P(x,dy) \\
&= \int \left[\int \|x-y\|^2 \pi(dx)\right] \left(a(x,y)Q(x,dy) + \left[\int (1-a(x,y))Q(x,dy)\right]\delta_x(dy)\right) \\
&= \int \|x-y\|^2 a(x,y)\pi(dx)Q(x,dy) + \int \|x-y\|^2 \left[\int (1-a(x,y))Q(x,dy)\right]\pi(dx)\delta_x(dy) \\
&= \int \|x-y\|^2 a(x,y)\pi(dx)Q(x,dy) \\
&= \mathbb{E}_{\pi\otimes Q}[\|X-Y\|^2 a(X,Y)].
\end{aligned}$$

To approximate this expectation, we run a MH algorithm and store the proposals as $X^*_{n-1} \sim Q(X_{n-1},\cdot)$ for $n = 1,\dots,N$. Then one can construct an empirical approximation (notice that, of course (!), the samples are dependent) and estimate this expectation as

$$\text{ESJD} \approx \frac{1}{N}\sum_{i=1}^{N} \|X^*_{n-1} - X_{n-1}\|^2 a_{\text{MH}}(X_{n-1}, X^*_{n-1}).$$

A.4 Sketch of proof of Metropolis-Hastings with involutions

This only serves as a sketch of the main ideas, a more rigorous proof is available in the original paper [4]. Let $(\mathsf{X},\mathscr{X})$ be a measurable space, $\mu \ll \lambda$ be two sigma-finite measures on it, and $\phi : \mathsf{X} \to \mathsf{X}$ be an involution. Assume that $\mu^\phi \ll \mu$ and $\lambda^\phi \ll \lambda$, then one hand we have

$$\begin{aligned}
\mu^\phi(h) &= \mu(h\circ\phi) && \text{change of variables} \\
&= \lambda\left((h\circ\phi)\cdot\frac{d\mu}{d\lambda}\right) && \text{Radon-Nikodym theorem} \\
&= \lambda^\phi\left(h\cdot\frac{d\mu}{d\lambda}\circ\phi^{-1}\right) && \text{change of variables} \\
&= \lambda\left(h\cdot\left(\frac{d\mu}{d\lambda}\circ\phi^{-1}\right)\cdot\frac{d\lambda^\phi}{d\lambda}\right) && \text{Radon-Nikodym theorem.}
\end{aligned}$$

and on the other hand

$$\mu^\phi(h) = \lambda\left(h\cdot\left(\frac{d\mu^\phi}{d\mu}\right)\cdot\frac{d\mu}{d\lambda}\right).$$

Loosely speaking, when $d\mu/d\lambda(x) > 0$, we can write

$$\frac{d\mu^\phi}{d\mu}(x) = \frac{\frac{d\mu}{d\lambda}(\phi^{-1}(x))}{\frac{d\mu}{d\lambda}(x)}\frac{d\lambda^\phi}{d\lambda}(x) \qquad \lambda-\text{almost everywhere.}$$

When λ is the Lebesgue measure on $(\mathsf{X},\mathscr{X})$ we write $\mu(x)$ for the density of μ with respect to it. Using the fact that $\phi = \phi^{-1}$ we get

$$\frac{d\mu^\phi}{d\mu}(x) = \frac{\mu\circ\phi(x)}{\mu(x)}|\det \mathrm{J}_{\phi^{-1}}(x)|.$$

A.5 Incremental Weights Derivations for SMC

A.5.1 Derivation for the Optimal Backward Kernel

Suppose no resampling is performed, and consider the following backward kernel

$$L_{n-1}^{\text{opt}}(x_n, dx_{n-1}) = \frac{\eta_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\eta_n(dx_n)} \tag{A.9}$$

where $\eta_n(dx_n)$ is defined as in equation (1.22). The IS weights become

$$\begin{aligned} w_n(x_{1:n}) &= \frac{\mu_n(dx_n)\prod_{k=1}^{n-1}\frac{\eta_k(dx_k)M_{k+1}(x_k, dx_{k+1})}{\eta_{k+1}(dx_{k+1})}}{\nu(dx_1)\prod_{k=2}^{n}M_k(x_{k-1}, x_k)} \\ &= \frac{\mu_n(dx_n)\left[\prod_{k=1}^{n-1}\eta_k(dx_k)\right]\left[\prod_{k=1}^{n-1}M_{k+1}(x_k, dx_{k+1})\right]}{\nu(dx_1)\left[\prod_{k=2}^{n}M_k(x_{k-1}, x_k)\right]\left[\prod_{k=1}^{n-1}\eta_{k+1}(dx_{k+1})\right]} \\ &= \frac{\mu_n(dx_n)\nu(dx_1)}{\nu(dx_1)\eta_n(dx_n)} \\ &= \frac{\mu_n(dx_n)}{\eta_n(dx_n)}, \end{aligned}$$

which means that this backward kernel reverts extended IS between $\mu_n(dx_{1:n})$ and $\eta_n(dx_{1:n})$ into standard IS between $\mu_n(dx_n)$ and $\eta_n(dx_n)$. To see why this kernel is optimal, first use the variance decomposition formula

$$\mathbb{V}[w_n(X_{1:n})] = \mathbb{E}[\mathbb{V}[w_n(X_{1:n}) \mid X_n]] + \mathbb{V}[\mathbb{E}[w_n(X_{1:n}) \mid X_n]]. \tag{A.10}$$

Then find an expression for the conditional expectation in the second term of Equation (A.10)

$$\begin{aligned} \mathbb{E}[w_n(X_{1:n}) \mid X_n] &= \int_{x_{1:n-1}\in\mathsf{X}^{n-1}} w_n(x_{1:n})\eta_{1:n-1|n}(dx_{1:n-1} \mid x_n) \\ &= \int_{x_{1:n-1}\in\mathsf{X}^{n-1}} \frac{\mu_n(dx_{1:n})}{\eta_n(dx_{1:n})}\frac{\eta_n(dx_{1:n})}{\eta_n(dx_n)} \\ &= \frac{\mu_n(dx_n)}{\eta_n(dx_n)}. \end{aligned}$$

The second term then becomes

$$\begin{aligned} \mathbb{V}[\mathbb{E}[w_n(X_{1:n}) \mid X_n]] &= \mathbb{E}\left[(\mathbb{E}[w_n(X_{1:n}) \mid X_n] - \mathbb{E}[w_n(X_{1:n})])^2\right] \qquad \text{Def of variance, TOWER} \\ &= \mathbb{E}\left[\left(\frac{\mu_n(X_n)}{\eta_n(X_n)} - \mu_n(X_n)\right)^2\right], \end{aligned}$$

which does not depend on the backward kernels and therefore can be disregarded. An optimal kernel, in terms of minimizing the variance, would then satisfy

$$L_{n-1}^{*} = \min_{L_{n-1}\in\mathscr{L}} \mathbb{E}[\mathbb{V}[w_n(X_{1:n}) \mid X_n]]$$

for a suitable space of Markov probability kernels $\mathscr{L}$. Notice, however, that choosing L_{n-1}^{opt} gives

$$\mathbb{V}[w_n(X_{1:n}) \mid X_n] = \mathbb{V}\left[\frac{d\mu_n}{d\eta_n}(X_n) \,\middle|\, X_n \right] = 0,$$

in other words, $L_{n-1}^{\text{opt}} = L_{n-1}^{*}$.

A.5.2 Derivation for the Near-Optimal Backward kernel

Consider again the situation where no resampling is performed and notice that by construction (1.16) can be written recursively and together with (1.23) leads to

$$\begin{aligned}
\eta_n(dx_n) &= \int_{x_{1:n-1}\in\mathsf{X}^{n-1}} \eta_{n-1}(dx_{n-1})\eta_{1:n-2|n-1}(x_{n-1}, dx_{n-2:1})M_n(x_{n-1}, dx_n) \\
&= \int_{x_{n-1}\in\mathsf{X}} \eta_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n) \int_{x_{1:n-2}\in\mathsf{X}^{n-2}} \eta_{1:n-2|n-1}(x_{n-1}, dx_{n-2:1}) \\
&= \eta_{n-1}M_n(dx_n),
\end{aligned}$$

which offers an alternative way of writing the optimal backward kernel

$$L_{n-1}^{\text{opt}}(x_n, dx_{n-1}) = \frac{\eta_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\eta_{n-1}M_n(dx_n)}.$$

The near-optimal backward kernel is found by simply replacing η_{n-1} with μ_{n-1}

$$L_{n-1}^{\text{near-opt}}(x_n, dx_{n-1}) = \frac{\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\mu_{n-1}M_n(dx_n)}. \tag{A.11}$$

This choice of backward kernel, leads to the following incremental weights

$$\omega_n(x_{n-1}, x_n) = \frac{d\mu_n}{d\mu_{n-1}M_n}(x_n), \tag{A.12}$$

notice that in this case the incremental weights only depend on x_n and not x_{n-1}.

A.5.3 Derivation when forward kernel leaves target invariant

To find the expression for the incremental weights given by this choice of kernel, we integrate out x_{n-1} from the Feynman-Kac formula (1.20)

$$\mu_n(dx_n) = \int_{x_{n-1}\in\mathsf{X}} \omega_n(x_{n-1}, x_n)\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n). \tag{A.13}$$

Another expression can be found by a simple change-of-measure argument and using the condition $\mu_{n-1}M_n = \mu_{n-1}$

$$\begin{aligned}
\mu_n(dx_n) &= \frac{\mu_n(dx_n)}{\mu_{n-1}(dx_n)}\mu_{n-1}(dx_n) \\
&= \frac{\mu_n(dx_n)}{\mu_{n-1}(dx_n)}\mu_{n-1}M_n(dx_n) \\
&= \frac{\mu_n(dx_n)}{\mu_{n-1}(dx_n)} \int_{x'_{n-1}\in\mathsf{X}} \mu_{n-1}(dx'_{n-1})M_n(x'_{n-1}, dx_n) \\
&= \int_{x'_{n-1}\in\mathsf{X}} \frac{\mu_n(dx_n)}{\mu_{n-1}(dx_n)}\mu_{n-1}(dx'_{n-1})M_n(x'_{n-1}, dx_n).
\end{aligned}$$

Comparing the last expression above with (A.13) we immediately find

$$\omega(x_{n-1}, x_n) = \frac{\mu_n(x_n)}{\mu_{n-1}(x_n)}, \tag{A.14}$$

and from the integrand we also discover

$$L_{n-1}(x_n, dx_{n-1}) = \frac{\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\mu_{n-1}(dx_n)}.$$

A.5.4 Derivation for constant incremental weights

We want $L_{n-1}^{\text{near-opt}}$ to be such that

$$\frac{\mu_n(dx_n)L_{n-1}^{\text{near-opt}}(x_n, dx_{n-1})}{\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)} = \text{const.}$$

The derivative is constant only when the measures are equivalent. Assuming $M_n(x_{n-1}, \cdot) \ll \mu_n$, we therefore require for any $\mathsf{A}, \mathsf{B} \in \mathscr{X}$

$$\begin{aligned}\int_{\mathsf{A}} \mu_n(dx_n)L_{n-1}^{\text{near-opt}}(x_n, \mathsf{B}) &= \int_{\mathsf{B}} \mu_{n-1}(dx_{n-1})M_n(x_{n-1}, \mathsf{A}) \\ &= \int_{\mathsf{B}} \mu_{n-1}(dx_{n-1}) \int_{\mathsf{A}} \frac{dM_n(x_{n-1}, \cdot)}{d\mu_n}(x_n)\mu_n(dx_n) \\ &= \int_{\mathsf{A}} \mu_n(dx_n) \int_{\mathsf{B}} \mu_{n-1}(dx_{n-1}) \frac{dM_n(x_{n-1}, \cdot)}{d\mu_n}(x_n),\end{aligned}$$

which means

$$L_{n-1}^{\text{near-opt}}(x_n, dx_{n-1}) = \mu_{n-1}(dx_{n-1}) \frac{dM_n(x_{n-1}, \cdot)}{d\mu_n}(x_n) \qquad \mu_n - \text{almost surely.} \tag{A.15}$$

This is often written in the nicer, yet slightly more confusing, form

$$L_{n-1}^{\text{near-opt}}(x_n, dx_{n-1}) = \frac{\mu_{n-1}(dx_{n-1})M_n(x_{n-1}, dx_n)}{\mu_n(dx_n)}.$$

With this choice, the incremental weights are constant. This choice does not seem to be used in the literature.

APPENDIX B

APPENDIX FOR EXACT MANIFOLD SAMPLING

B.1 Ordinary Differential Equations and Integrators

A system of ODEs in two variables is *partitioned* if, for functions F and G, it can be written as

$$\dot{x} = \mathrm{F}(x, v)$$
$$\dot{v} = \mathrm{G}(x, v).$$

The generalised Stormer-Verlet is a common integrator for this kind of systems and it comes in two forms: a position and velocity version (they are here given in their Leapfrog formulation)

$$x_{n+\frac{1}{2}} = x_n + \frac{\delta}{2}\mathrm{F}(x_{n+\frac{1}{2}}, v_n) \qquad v_{n+\frac{1}{2}} = v_n + \frac{\delta}{2}\mathrm{G}(x_n, v_{n+\frac{1}{2}})$$
$$v_{n+1} = v_n + \frac{\delta}{2}\left[\mathrm{G}(x_{n+\frac{1}{2}}, v_n) + \mathrm{G}(x_{n+\frac{1}{2}}, v_{n+1})\right] \qquad x_{n+1} = x_n + \frac{\delta}{2}\left[\mathrm{F}(x_n, v_{n+\frac{1}{2}}) + \mathrm{F}(x_{n+1}, v_{n+\frac{1}{2}})\right]$$
$$x_{n+1} = x_{n+\frac{1}{2}} + \frac{\delta}{2}\mathrm{F}(x_{n+\frac{1}{2}}, v_{n+1}), \qquad v_{n+1} = v_{n+\frac{1}{2}} + \frac{\delta}{2}\mathrm{G}(x_{n+1}, v_{n+\frac{1}{2}}).$$

The integrators above are second-order, symmetric, symplectic but, for a general partitioned system, implicit. An example is the Hamiltonian system corresponding to a Hamiltonian with potential $V(x) = -\log\pi(x)$ and a quadratic kinetic energy with position-dependent mass matrix

$$H(x, v) = V(x) + \frac{1}{2}v^\top M^{-1}(x)v + \frac{1}{2}\log|M(x)|.$$

[59] integrated the corresponding Hamiltonian system using the velocity version above (corresponding to Equations 16-18 in their paper), and used a fixed-point iterator to solve the first velocity and position updates, which are defined implicitly. When the system of ODEs is *separable*

$$\dot{x} = \mathrm{F}(v)$$
$$\dot{v} = \mathrm{G}(x),$$

then the generalised Stormer-Verlet reduces to the standard Leapfrog integrator. An example is the system corresponding to a Hamiltonian with global mass matrix

$$H(x,v) = V(x) + \frac{1}{2} v^\top M^{-1} v.$$

B.2 Constrained Hamiltonian systems and their simulation

Consider a Hamiltonian subject to the holonomic constraint (i.e. depending only on x) $f(x) = 0$, for a sufficiently smooth function $f : \mathbb{R}^n \to \mathbb{R}^m$. As detailed in [84], one way to obtain the corresponding Hamiltonian system is to modify the potential to include a penalty term that grows larger as we part from the manifold $\mathscr{M} = f^{-1}(0)$ (here on we shall assume $M = \mathrm{I}$ since a rescaling of the momentum variable is always possible)

$$H_c(x,v) = V(x) + \frac{1}{2} v^\top v + \frac{1}{2\epsilon} f(x)^\top f(x),$$

with corresponding system (introduce the variable $\lambda = \epsilon^{-1} f(x)$, thus the third equation)

$$\begin{aligned} \dot{x} &= \nabla_p H_c(x,v) = v \\ \dot{v} &= -\nabla_q H_c(x,v) = -\nabla_x V(x) - \mathrm{J}(x)^\top \lambda \\ \lambda &= \frac{1}{\epsilon} f(x). \end{aligned}$$

Sending the tolerance $\epsilon \to 0$ we obtain a constrained Hamiltonian system

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -\nabla_x V(x) - \mathrm{J}(x)^\top \lambda \\ 0 &= f(x) \end{aligned} \tag{B.1}$$

This is an example of a semi-explicit, index-three DAEs. Its index (minus one) refers to the number of times we need to differentiate the algebraic constraint $f(x) = 0$ in order to transform (B.1) into an ODE. Differentiating the constraint twice we obtain (notice the conflict of notation: $\mathrm{H}[\cdot](\cdot,\cdot)$ denotes the Hessian tensor[1], whereas $H(\cdot,\cdot)$ denotes a Hamiltonian function)

$$\begin{aligned} \mathrm{J}(x) v &= 0 \\ \mathrm{H}[x](v,v) + \mathrm{J}(x)\dot{v} &= 0, \end{aligned}$$

and plugging the expression for $\dot{v}$ we can solve for λ, obtaining the *standard underlying ODE*

$$\ddot{x} = -\nabla_x V(x) - \mathrm{J}(x)^\top (\mathrm{J}(x)\mathrm{J}(x)^\top)^{-1} \mathrm{H}[x](v,v) + \mathrm{N}_x \nabla_x V(x). \tag{B.2}$$

[1]The Hessian of $f : \mathbb{R}^n \to \mathbb{R}^m$ at any point $x \in \mathbb{R}^n$ is denoted $\mathrm{H}[x]$ and it is an order-three tensor, which can be thought of as a function that takes two vectors $v_1, v_2 \in \mathbb{R}^n$ and returns a third vector $\mathrm{H}[x](v_1,v_2) \in \mathbb{R}^m$. When $m = 1$ then $\mathrm{H}[x]$ is simply the Hessian matrix $H(x)$, and $\mathrm{H}[x](v_1,v_2)$ simplifies to a quadratic form $v_1^\top \mathrm{H}(x) v_2$.

This system is not Hamiltonian in general, and definitely not separable. Multiple approaches have been suggested in the literature to simulate constrained Hamiltonian dynamics. Indirect approaches, detailed in Section 2.5, integrate the stiff ODE (B.2). Direct approaches integrate the constrained system (B.1). For instance, the RATTLE scheme (see Algorithm 10 and [8] and references therein) corresponds to using the velocity Stormer-Verlet scheme together with additional projection steps to keep $f(x) = 0$ and $\mathrm{J}(x)v = 0$ satisfied.

Algorithm 10: RATTLE (Constrained Leapfrog - velocity version)

INPUT : Initial position x_0 and momentum v_0, step size $\delta > 0$, convergence tolerance $\rho > 0$, number of optimization iterations N_{iter}.

1 Unconstrained momentum update: $\tilde{v}_{\frac{1}{2}} \longleftarrow v_0 - \frac{\delta}{2}\nabla_x V(x_0)$

2 Project momentum: $\overline{v}_{\frac{1}{2}} \longleftarrow \mathrm{T}_0\tilde{v}_{\frac{1}{2}}$

3 Unconstrained position update: $\tilde{x}_1 \longleftarrow x_0 + \delta\overline{v}_{\frac{1}{2}}$

4 Project position: $x_1, \text{flag} \longleftarrow \text{ProjectPosition}(\tilde{x}_1, x_0, \rho, N_{\text{iter}})$

5 $v_{\frac{1}{2}} \longleftarrow \frac{1}{\delta}(x_1 - x_0)$

6 Unconstrained momentum update: $\tilde{v}_1 \longleftarrow v_{\frac{1}{2}} - \frac{\delta}{2}\nabla_x V(x_1)$

7 Project momentum: $v_1 \longleftarrow \mathrm{T}_1\tilde{v}_1$

OUTPUT : Final position x_1 and momentum v_1, and optimization flag.

B.3 Constrained Hamiltonian Monte Carlo

C-HMC generalises C-RWM described in Section 2.2.3. C-HMC integrates the Hamiltonian dynamics (B.1), performs a reversibility check for detailed balance, and then uses a Metropolis-Hastings step to guarantee sampling from the correct target distribution. Notice that in HMC/C-HMC, the unconstrained/constrained Hamiltonian is used twice: its corresponding Hamiltonian system is integrated to propose a new candidate, and its used in the MH step for acceptance. However, [25] observed that one doesn't necessarily need to use the same Hamiltonian for the proposal and the acceptance step and describe a family of C-HMC methods using two Hamiltonians

- A *guidance* Hamiltonian $H_g(x,v)$ whose corresponding system is integrated to propose a new candidate.
- An *acceptance* Hamiltonian $H_a(x,v)$ used in the acceptance ratio.

We shall refer to this family of C-HMC algorithms as Generalized Constrained HMC (GC-HMC). As an illustrative example, suppose the distribution of interest is a conditional probability measure η on $\mathscr{M}$ with density (2.2). One could set the acceptance and guidance Hamiltonians to

$$H_g(x,v) = -\log\pi(x) + \frac{1}{2}v^\top v + \mathrm{J}_x^\top\lambda$$
$$H_a(x,v) = -\log\eta(x) + \frac{1}{2}v^\top v + \mathrm{J}_x^\top\lambda.$$

Using a potential based on $\pi(x)$ rather than $\eta(x)$ in the guidance Hamiltonian avoids computing the (very expensive) gradient of the manifold potential in the Leapfrog steps [8]

$$\nabla_x(-\log\eta(x)) = -\nabla_x \log\pi(x) + \frac{1}{2}\nabla_x \log|\mathrm{J}_x\mathrm{J}_x^\top|.$$

B.4 Relationship of THUG with C-HMC

THUG can be thought of a particular instance of GC-HMC where we use the filamentary distribution η_ϵ in the acceptance Hamiltonian

$$H_a(x,v) = -\log\eta_\epsilon(x) + \frac{1}{2}v^\top v,$$

and propose samples by discretizing the underlying ODE for a constant potential $V(x) = \text{const}$

$$\ddot{x} = -\mathrm{J}_x^\top(\mathrm{J}_x\mathrm{J}_x^\top)^{-1}\mathrm{H}(v,v)[x].$$

using the THUG bounce as a numerical integrator using approximate second-order information (see Appendix C.3).

B.5 Co-Area formula and conditional probability measures

Geometric Measure Theory results in this section are inspired from [50, 81]. Before stating a version of the Co-Area formula, we need to defined the generalized Jacobian (also known as n-dimensional Jacobian).

Definition B.1 (Generalized Jacobian)**.** Let $f:\mathbb{R}^n \to \mathbb{R}^m$ be differentiable at $x \in \mathbb{R}^n$ with Jacobian J_x. Then, the generalized Jacobian of f at x is

$$\mathscr{J}_m f(x) = \begin{cases} \left|\det \mathrm{J}_x\mathrm{J}_x^\top\right|^{1/2} & \text{if } n \geq m \\ \left|\det \mathrm{J}_x^\top\mathrm{J}_x\right|^{1/2} & \text{if } n \leq m \\ |\det \mathrm{J}_x| & \text{if } n = m \end{cases}$$

Next, we see a special case of the Co-Area formula for Lipschitz functions. The theorem below is equivalent to Theorem 5.3.9 in [81] and has appeared in [40].

Theorem B.1 (Co-Area formula for Lipschitz functions)**.** *Let $f:\mathbb{R}^n \to \mathbb{R}^m$ with $n > m$ be Lipschitz and $g:\mathbb{R}^n \to \mathbb{R}$ be $\mathscr{B}(\mathbb{R}^n)$-measurable. Then*

$$\int_{\mathbb{R}^n} g(x)\mathscr{J}_m f(x)dx = \int_{\mathbb{R}^m}\int_{f^{-1}(\{y\})} g(x)\mathscr{H}^{n-m}(dx)dy$$

An important special case is when f is Lipschitz and has full row-rank Jacobian almost everywhere.

Theorem B.2 (Co-Area formula with full row rank Jacobian)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ with $n > m$ be Lipschitz with Jacobian J_x full row-rank almost everywhere, and $g : \mathbb{R}^n \to \mathbb{R}$ be $\mathscr{B}(\mathbb{R}^n)$-measurable. Then $\mathscr{J}_m f(x) > 0$ almost everywhere and*

$$\int_{\mathbb{R}^n} g(x)dx = \int_{\mathbb{R}^m} \int_{f^{-1}(\{y\})} g(x)(\mathscr{J}_m f)^{-1}(x)\mathscr{H}^{n-m}(dx)dy$$

A consequence of the Co-Area formula is that we can find an expression for the regular conditional distribution of a random variable given that its constrained image under a suitable constraint function. Similar statements related to statistical physics can be found [34, 155].

Theorem B.3 (Density on a Fiber)**.** *Let $(\Omega, \mathscr{F}, \mathbb{P})$ be a probability space, $U : \Omega \to \mathbb{R}^n$ be a random variable with distribution $P_U = \mathbb{P} \circ U^{-1}$, and density p_U with respect to du, the Lebesgue measure on $(\mathbb{R}^n, \mathscr{B}(\mathbb{R}^n))$. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a Lipschitz function with Jacobian matrix $\mathrm{J}_f(u)$ full row-rank du-almost everywhere, and let $C = f \circ U$ have distribution $P_C = P_U \circ f^{-1}$. Then*

- *P_C admits a density with respect to dc, the Lebesgue measure on $(\mathbb{R}^m, \mathscr{B}(\mathbb{R}^m))$, of the form*

  $$p_C(c) = \int_{f^{-1}(\{c\})} p_U(u)\mathscr{J}_m f^{-1}(u)\mathscr{H}^{n-m}(du),$$

  where $\mathscr{H}^{n-m}(du)$ is the $(n-m)$-dimensional Hausdorff measure on $f^{-1}(\{c\})$, and $\mathscr{J}_m f^{-1}$ is the inverse of the generalised Jacobian of Definition B.1.

- *A version of the regular conditional distribution $\mathbb{E}[\mathbb{1}_{U \in \mathsf{A}} \mid \sigma(f(U))]$ is given by*

  $$\mathbb{P}(U \in \mathsf{A} \mid f(U) = c) = \frac{1}{p_C(c)} \int_{\mathsf{A} \cap f^{-1}(\{c\})} p_U(u)\mathscr{J}_m f^{-1}(u)\mathscr{H}^{n-m}(du),$$

  defined only for those $c \in \mathbb{R}^m$ such that $p_C(c) > 0$. The probability measure defined as $\mathrm{K}_c(\mathsf{A}) = \mathbb{P}(U \in \mathsf{A} \mid f(U) = c)$ admits a density with respect to $d\mathscr{H}^{n-m}$ given by

  $$\frac{d\mathrm{K}_c}{d\mathscr{H}^{n-m}}(u) = \frac{p_U(u)\mathscr{J}_m f^{-1}(u)}{p_C(c)}.$$

Proof. For the first part, it is enough to show that there exists a $\mathscr{B}(\mathbb{R}^m)$-measurable function $p_C(c) : \mathbb{R}^m \to [0, +\infty)$ such that for any $\mathscr{B}(\mathbb{R}^m)$-measurable set A

$$\mathbb{P}(C \in \mathsf{A}) = \int_{\mathsf{A}} p_C(c)dc.$$

With this in mind, we notice that using the Co-Area formula one can write the LHS as

$$\begin{aligned}
\mathbb{P}(C \in \mathsf{A}) &= \int_{\mathbb{R}^m} \mathbb{1}_{\mathsf{A}}(c) P_C(dc) \\
&= \int_{\mathbb{R}^n} \mathbb{1}_{\mathsf{A}} \circ f(u) P_U(du) && \text{Change of Variables} \\
&= \int_{\mathbb{R}^n} \mathbb{1}_{\mathsf{A}} \circ f(u) p_U(u) du && P_U \text{ admits a density} \\
&= \int_{\mathbb{R}^m} \int_{f^{-1}(\{c\})} \mathbb{1}_{\mathsf{A}} \circ f(u) p_U(u) \mathscr{J}_m f^{-1}(u) \mathscr{H}^{n-m}(du) dc && \text{Theorem B.1} \\
&= \int_{\mathsf{A}} \left[\int_{f^{-1}(\{c\})} p_U(u) \mathscr{J}_m f^{-1}(u) \mathscr{H}^{n-m}(du) \right] dc && f(u) \text{ constant on } f^{-1}(\{c\}),
\end{aligned}$$

which suggests taking

$$p_C(c) = \int_{f^{-1}(\{c\})} p_U(u) \mathscr{J}_m f^{-1}(u) \mathscr{H}^{n-m}(du).$$

Next, we prove the second statement of the theorem. Denote the support of the density above by

$$\mathsf{S} = \left\{ c \in \mathbb{R}^m : p_c(c) > 0 \right\},$$

and notice that one can write the joint over (U, C) as follows

$$\mathbb{P}(U \in \mathsf{A}, C \in \mathsf{B}) = \mathbb{P}(U \in \mathsf{A}, C \in \mathsf{B} \cap \mathsf{S}) + \mathbb{P}(U \in \mathsf{A}, C \in \mathsf{B} \cap \mathsf{S}^C) \tag{B.3}$$

$$= \mathbb{P}(U \in \mathsf{A}, C \in \mathsf{B} \cap \mathsf{S}) \tag{B.4}$$

The second term disappears because on S^C the density is zero, meaning $P_C(\mathsf{S}^C) = 0$, and hence

$$\mathbb{P}(U \in \mathsf{A}, C \in \mathsf{B} \cap \mathsf{S}^C) = \mathbb{P}(U \in \mathsf{A} \mid C \in \mathsf{B} \cap \mathsf{S}^C) \mathbb{P}(C \in \mathsf{S}^C) = 0.$$

The same joint distribution can be written in an alternative form by further algebraic manipulation

$$\mathbb{P}(U \in \mathsf{A}, C \in \mathsf{B}) = \mathbb{P}(\{U \in \mathsf{A}\} \cap \{U \in f^{-1}(\mathsf{B} \cap \mathsf{S})\}) = \mathbb{P}(U \in \{\mathsf{A} \cap f^{-1}(\mathsf{B} \cap \mathsf{S})\}). \tag{B.5}$$

We can use the Co-Area formula to show the existence of a version of the conditional distribution mentioned in the statement of the theorem.

$$\begin{aligned}
\mathbb{P}(U \in \{\mathsf{A} \cap f^{-1}(\mathsf{B} \cap \mathsf{S})\}) &= \int_{\mathsf{A} \cap f^{-1}(\mathsf{B} \cap \mathsf{S})} P_U(du) \\
&= \int_{\mathbb{R}^n} \mathbb{1}_{\mathsf{B} \cap \mathsf{S}}(f(u)) \mathbb{1}_{\mathsf{A}}(u) p_U(u) du \\
&= \int_{\mathbb{R}^m} \int_{f^{-1}(\{c\})} \mathbb{1}_{\mathsf{B} \cap \mathsf{S}}(f(u)) \mathbb{1}_{\mathsf{A}}(u) p_U(u) (\mathscr{J}_m f^{-1})(u) \mathscr{H}^{n-m}(du) dc \\
&= \int_{\mathsf{B} \cap \mathsf{S}} \int_{\mathsf{A} \cap f^{-1}(\{y\})} p_U(u) (\mathscr{J}_m f^{-1})(u) \mathscr{H}^{n-m}(du) dc \\
&= \int_{\mathsf{B} \cap \mathsf{S}} \mathbb{P}(U \in \mathsf{A} \mid f(U) = c) p_C(c) dc, && \text{(B.6)}
\end{aligned}$$

where we have defined the function $c \mapsto \mathbb{P}(U \in \mathsf{A} \mid f(U) = c)$ as

$$\mathbb{P}(U \in \mathsf{A} \mid f(U) = c) = \frac{1}{p_C(c)} \int_{\mathsf{A} \cap f^{-1}(\{c\})} p_U(u) \mathscr{J}_m f^{-1}(u) \mathscr{H}^{n-m}(du),$$

which is well-defined on $\mathsf{B} \cap \mathsf{S}$. Equations (B.6), (B.5), and (B.4) show that this function is indeed a version of the regular conditional distribution $\mathbb{E}[\mathbb{1}_{U \in \mathsf{A}} \mid \sigma(f(U))]$. Finally, the last statement of the theorem follows from the display above by bringing the denominator inside the integral, obtaining the desired density. ■

B.6 A clarification on manifold sampling

Typically, the level set $f^{-1}(y)$ will not be a manifold, in the classical mathematical sense, unless f is smooth. In that case, Sard's theorem says that almost every $y \in \mathbb{R}^m$ is a regular value, and from the preimage theorem it follows that for almost every $y \in \mathbb{R}^m$, $f^{-1}(y)$ is a manifold.

In the literature, authors have alternated between assuming f being smooth [87, 153] or Lipschitz [40, 63], yet have always referred to the task of manifold sampling. When f is Lipschitz, manifold sampling is a misnomer and one should technically talk about level-set sampling. In practice, this does not matter because whenever f is Lipschitz, and the Jacobian is full row-rank almost everywhere, then the Co-Area formula, Theorem B.3, and Theorem C.5 hold, and hence

1. J_x and $|\mathrm{J}_x \mathrm{J}_x^\top|^{-1/2}$ exist and can be computed almost everywhere,
2. $\eta(x) \propto \pi(x)|\mathrm{J}_x \mathrm{J}_x^\top|^{-1/2}$ is defined on $f^{-1}(y)$ almost everywhere, and has the correct form (although one should refer to it as a level-set distribution),
3. $\mathbb{E}_{\eta_\epsilon}[\psi] \to \mathbb{E}_\eta[\psi]$ in the sense of Theorems C.3 and C.4,

which is all that matters from a computational and practical point of view.

The next theorem shows that with mild assumptions on the test and constraint functions, expectations with respect to a filamentary distribution tend to expectations with respect to manifold distributions, as $\epsilon \to 0^+$. Proofs of related results can be found in [34], [155], and references therein.

APPENDIX C

APPENDIX FOR APPROXIMATE MANIFOLD SAMPLING

C.1 Convergence of Filamentary Distributions

C.1.1 Notation and terminology

Almost everywhere statements are always with respect to the Lebesgue measure, which is typically obvious from the context. Within an integral with respect to a Lebesgue measure, the Lebesgue measure will be denoted by d followed by the variable of integration, e.g. dx, dy, or dz. To denote the Lebesgue measure of a set B, we shall write Leb(B). For a radius $r > 0$ we denote by $\mathbb{B}(p,r)$ and $\mathbb{B}^C(p,r)$ the open ball of radius r centered at p and its complement respectively

$$\mathbb{B}(p,r) = \left\{q \in \mathbb{R}^m : \|q-p\| < r\right\}$$
$$\mathbb{B}^C(p,r) = \left\{q \in \mathbb{R}^m : \|q-p\| \geq r\right\},$$

where the dimensionality of the ball should always be clear from context. For a ball centered at the origin, we will just write $\mathbb{B}_r$ and $\mathbb{B}_r^C$ for short. Similarly, given a function $f : \mathbb{R}^n \to \mathbb{R}^m$ we define

$$\mathbb{B}f(p,r) = \{s \in \mathbb{R}^n : \|f(s)-p\| < r\}$$
$$\mathbb{B}^C f(p,r) = \{s \in \mathbb{R}^n : \|f(s)-p\| \geq r\},$$

where again we write $\mathbb{B}_r f$ and $\mathbb{B}_r^C f$ for short when the balls are centered at the origin. Additionally, we also denote by $\overline{\mathbb{B}}(p,r)$ and $\overline{\mathbb{B}f}(p,r)$ the respective closures. Before stating and proving Theorem C.3, we need some ingredients from real analysis and measure theory. In particular, we need to understand which functions k_ϵ are allowed. In the main text we called them smoothing kernel, following the literature in ABC [128], however a more general and precise notion is of an approximation to the identity, where by identity we mean the Dirac delta Schwartz distribution.

Definition C.1 (Approximation to the Identity)**.** A sequence of integrable functions $\{k_\epsilon : \mathbb{R}^m \to \mathbb{R}\}_{\epsilon>0}$ is called an approximation to the identity if there exists a constant A such that the following three conditions are satisfied

$$\int_{\mathbb{R}^m} k_\epsilon(x)dx = 1 \qquad \forall\, \epsilon > 0 \tag{C.1}$$

$$|k_\epsilon(x)| \le \frac{A}{\epsilon^m} \qquad \forall\, \epsilon > 0, \forall\, x \in \mathbb{R}^m \tag{C.2}$$

$$|k_\epsilon(x)| \le \frac{A\epsilon}{|x|^{m+1}} \qquad \forall\, \epsilon > 0, \forall\, x \in \mathbb{R}^m \setminus \{0\}. \tag{C.3}$$

A common way to construct approximations to the identity is to take any compactly-supported, non-negative, bounded function $k : \mathbb{R}^m \to \mathbb{R}_+$ and for any $\epsilon > 0$ define

$$k_\epsilon(x) = \frac{1}{\epsilon^m} k\left(\frac{x}{\epsilon}\right).$$

The condition regarding a compact support can be dropped if the function decays to zero quickly enough in the tails. An example is that of the Gaussian kernel.

Example C.1.1.1 (Gaussian Approximation to the Identity)**.** *Consider the following function*

$$k_\epsilon(y) = \frac{1}{(2\pi)^{m/2}\epsilon^m} \exp\left(-\frac{\|y\|^2}{2\epsilon^2}\right) = \mathcal{N}(y \mid 0, \epsilon^2 \mathrm{I})$$

It is straightforward that the first two properties of an approximation to the identity are satisfied since k_ϵ is a probability density function, and since $\exp(-\|y\|^2/(2\epsilon^2)) \le 1$. To show that the third property holds, we simply rewrite it in the following form

$$k_\epsilon(y) = \frac{\epsilon}{(2\pi)^{m/2}\|y\|^{m+1}} \cdot h\left(\frac{\|y\|}{\epsilon}\right),$$

where for any $r \ge 0$ we have defined

$$h(r) = r^{m+1} \exp\left(-\frac{r^2}{2}\right).$$

Clearly $h \to 0$ whenever $r \to \infty$, and so one can always find $R > 0$ large enough so that $|h(r)| < 1$, for any $r > R$, implying that $h(r)$ is bounded on $(R, \infty]$. Since h is continuous and $[0, R]$ is closed, by the extreme value theorem h is bounded, say by $\overline{h} > 0$. Specifically, one could choose the upper bound

$$M := \max\left\{1, \overline{h}\right\},$$

to conclude

$$|k_\epsilon(y)| \le \frac{A\epsilon}{\|y\|^{m+1}} \qquad \textit{where} \qquad A = \frac{M}{(2\pi)^{m/2}}.$$

First, we recall the notion of a locally-integrable function.

Definition C.2 (Locally-integrable function)**.** A function f on $\mathbb{R}^n$ is said to be locally-integrable if for any radius $R > 0$, the function $y \mapsto \mathbb{1}_{\mathbb{B}_R}(y) f(y)$ is integrable on $\mathbb{R}^n$.

We give a standard definition of the Lebesgue set of a function [134].

Definition C.3 (Lebesgue Set)**.** Let f be locally-integrable on $\mathbb{R}^n$. The Lebesgue set of f is

$$\text{LebSet}(f) := \left\{ x \in \mathbb{R}^n : f(x) < \infty \text{ and } \lim_{\substack{\text{Leb}(B)\to 0 \\ x\in B}} \frac{1}{\text{Leb}(B)} \int_B |f(y) - f(x)| dy = 0 \right\}.$$

Two well-known facts about the Lebesgue set of a function are collected in the following theorem, which is a standard result [134].

Theorem C.1. *If f is locally-integrable on $\mathbb{R}^n$ then almost every point belongs to the Lebesgue set of f, and if $x \in LebSet(f)$ then*

$$\lim_{\substack{Leb(B)\to 0 \\ x\in B}} \frac{1}{Leb(B)} \int_B f(y) dy = f(x).$$

We note that a sufficient condition for a point in the domain of f to be in the Lebesgue set of f is that f is continuous at that point. It follows that for a continuous function, the Lebesgue set coincides with the domain. The theorem below (see Theorem 2.1, Chapter 3 in [134]) tells us that the functions $\{k_\epsilon\}_{\epsilon>0}$ provide indeed an approximation to the identity element, under mild conditions.

Theorem C.2 (Property of Approximations to the Identity)**.** *Let $\{k_\epsilon : \mathbb{R}^m \to \mathbb{R}\}_{\epsilon>0}$ be an approximation to the identity and F be an integrable function on $\mathbb{R}^m$. Then $(F * k_\epsilon)(x) \to F(x)$ as $\epsilon \to 0^+$ for every x in the Lebesgue set of F, and this holds for almost every x.*

We are now ready to show that expectations with respect to a manifold distributions can be approximated with expectations with respect to a filamentary distribution.

Theorem C.3 (Convergence of Filamentary Distributions)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be Lipschitz with $n > m$, such that $\mathrm{J}_f(x)$ is full row-rank almost everywhere. Let $\psi : \mathbb{R}^n \to \mathbb{R}$ be π-integrable, and $\{k_\epsilon\}_{\epsilon>0}$ an approximation to the identity. Then for almost every $y^* \in \mathbb{R}^m$*

$$\lim_{\epsilon\to 0^+} \int_{\mathbb{R}^n} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx = \int_{f^{-1}(\{y^*\})} \psi(x)\pi(x)(\mathscr{J}_m f^{-1})(x)\mathscr{H}^{n-m}(dx).$$

Proof. Since $\psi(x)$ is π-integrable, the integral

$$\int_{\mathbb{R}^n} \psi(x)\pi(x)dx$$

exists, and since f is Lipschitz, it can be rewritten using the Co-Area formula

$$\int_{\mathbb{R}^n} \psi(x)\pi(x)dx = \int_{\mathbb{R}^m} \left[\int_{f^{-1}(\{y\})} \psi(x)\pi(x)(\mathscr{J}_m f^{-1})(x)\mathscr{H}^{n-m}(dx) \right] dy,$$

where we have used the fact that $\mathrm{J}_f(x)$ full row-rank almost everywhere implies $\mathscr{J}_m f(x) > 0$ almost everywhere. We denote the function corresponding to the inner integral by

$$W(y) := \int_{f^{-1}(\{y\})} \psi(x)\pi(x)(\mathscr{J}_m f^{-1})(x)\mathscr{H}^{n-m}(dx),$$

and in particular the Co-Area formula implies that it is dy-integrable. On the other hand, $\{k_\epsilon\}_{\epsilon>0}$ is an approximation to the identity and hence, for each $\epsilon > 0$, it is bounded, and so for any y^*

$$\int_{\mathbb{R}^n} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx$$

also exists. Again, using the Co-Area formula

$$\begin{aligned}\int_{\mathbb{R}^n} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx &= \int_{\mathbb{R}^m}\int_{f^{-1}(\{y\})} \psi(x)\pi(x)k_\epsilon(y^* - f(x))(\mathscr{J}_m f^{-1})(x)\mathscr{H}^{n-m}(dx)dy \\ &= \int_{\mathbb{R}^m} k_\epsilon(y^* - y)W(y)dy.\end{aligned}$$

Finally, recall that $W(y)$ was shown to be dy-integrable so that Theorem C.2 can be used, and for y^* in the Lebesgue set of W

$$\lim_{\epsilon\to 0^+}\int_{\mathbb{R}^n} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx = \int_{f^{-1}(\{y^*\})} \psi(x)\pi(x)(\mathscr{J}_m f^{-1})(x)\mathscr{H}^{n-m}(dx).$$

■

In the theorem above, the assumption that ψ is π-integrable leads to $W(y)$ being integrable on the whole of $\mathbb{R}^m$. However, this seems a particularly strong, and unnecessary, assumption. More realistic we expect to be able to prove a similar result in the case where $W(y)$ is integrable only in a region around the fixed observed value y^*. This is because we expect the kernel k_ϵ to allow us to neglect values that are far away from y^*. With this in mind, we shall state a new version of the theorem, where we assume that one can find a function $V \geq 1$ that can help bound the tails of the kernel fast enough so that we can only consider regular values in a neighborhood of y^*.

Below is an example of how the conditions of Theorem C.3 can be relaxed.

Theorem C.4 (Convergence of Filamentary Distributions II)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be Lipschitz, with $n > m$, such that $\mathrm{J}_f(x)$ is full row-rank almost everywhere. Let $V : \mathbb{R}^m \to \mathbb{R}$ be a measurable function such that*

$$V(y) \geq 1 \qquad \forall y \in \mathbb{R}^m \tag{C.4}$$

$$\forall B \subset \mathbb{R}^m \text{ compact, } \exists K > 0 \text{ such that } V(y) < K \qquad \forall y \in B \tag{C.5}$$

Let $y^* \in \mathbb{R}^m$ be fixed, and assume that for any $R > 0$, y^* is a point in the Lebesgue set of the function

$$y \longmapsto \mathbb{1}_{\mathbb{B}_R}(y)\frac{|W(y^* - y) - W(y^*)|}{V(y)},$$

where

$$W(y) := \int_{f^{-1}(y)} \psi(x)\pi(x)(\mathscr{J}_m f^{-1})(x)\mathscr{H}^{n-m}(dx).$$

Let $\psi : \mathbb{R}^n \to \mathbb{R}$ be a function such that the following integral exists and is finite

$$\int_{\mathbb{R}^n} \frac{\psi(x)\pi(x)}{V(y^* - f(x))}dx, \tag{C.6}$$

where $\pi(x)$ is a probability density function on $\mathbb{R}^n$. For any $\epsilon > 0$, let $k_\epsilon : \mathbb{R}^m \to \mathbb{R}$ be an approximation to the identity with the additional assumption that there exists a constant A such that

$$|k_\epsilon(y)| \leq \frac{A\epsilon}{\|y\|^{m+1} V(y)} \qquad \forall\, y \in \mathbb{R}^m \setminus \{0\},\ \forall\, \epsilon > 0. \tag{C.7}$$

Then

$$\lim_{\epsilon \to 0^+} \int_{\mathbb{R}^n} \psi(x)\pi(x) k_\epsilon(y^* - f(x)) dx = W(y^*). \tag{C.8}$$

Proof. First, we show that $W(y)/V(y^* - y)$ is dy-integrable. Since (C.6) is finite, using the Co-Area formula of Theorem B.1 we find

$$\int_{\mathbb{R}^n} \frac{\psi(x)\pi(x)}{V(y^* - f(x))} dx = \int_{\mathbb{R}^m} \frac{W(y)}{V(y^* - y)} dy = \int_{\mathbb{R}^m} \frac{W(y^* - u)}{V(u)} du < \infty. \tag{C.9}$$

Next, we show that $\psi(x)\pi(x)k_\epsilon(y^* - f(x))$ is dx-integrable.

$$\begin{aligned}
\int_{\mathbb{R}^n} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx &= \int_{\overline{\mathbb{B}_\epsilon f}} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx + \int_{\overline{\mathbb{B}_\epsilon f}^C} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx \\
&\leq \frac{A}{\epsilon^m} \int_{\overline{\mathbb{B}_\epsilon f}} \frac{V(y^* - f(x))}{V(y^* - f(x))} \psi(x)\pi(x)dx + \int_{\overline{\mathbb{B}_\epsilon f}^C} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx \\
&\leq \frac{AK}{\epsilon^m} \int_{\overline{\mathbb{B}_\epsilon f}} \frac{\psi(x)\pi(x)}{V(y^* - f(x))} dx + A\epsilon \int_{\overline{\mathbb{B}_\epsilon f}^C} \frac{\psi(x)\pi(x)}{\|y^* - f(x)\|^{m+1} V(y^* - f(x))} dx \\
&\leq \frac{AK}{\epsilon^m} \int_{\overline{\mathbb{B}_\epsilon f}} \frac{\psi(x)\pi(x)}{V(y^* - f(x))} dx + \frac{A}{\epsilon^m} \int_{\overline{\mathbb{B}_\epsilon f}^C} \frac{\psi(x)\pi(x)}{V(y^* - f(x))} dx \\
&< \infty.
\end{aligned}$$

In the first line, we simply split the integral into an integral over

$$\overline{\mathbb{B}_\epsilon f} = \{x \in \mathbb{R}^n : y^* - f(x) \in \mathbb{B}_\epsilon\},$$

and an integral over the complement $\overline{\mathbb{B}_\epsilon f}^C$. In the second line, we use property (C.2) for the first term and leave the second unchanged. In this third line, for the first term, we use property (C.5) since for any $x \in \overline{\mathbb{B}_\epsilon f}$, $y^* - f(x)$ is in the compact set $\overline{\mathbb{B}}_\epsilon$, whereas for the second term we use property (C.3). In the fourth line, we upper bound $\|y^* - f(x)\|^{-(m+1)}$ by $\epsilon^{-(m+1)}$. Finally, the last line follows from assumption (C.6).

Since the LHS on the display above is finite, we can use the Co-Area formula to conclude

$$\int_{\mathbb{R}^n} \psi(x)\pi(x)k_\epsilon(y^* - f(x))dx = \int_{\mathbb{R}^m} W(y)k_\epsilon(y^* - y)dy = \int_{\mathbb{R}^m} W(y^* - y)k_\epsilon(y)dy < \infty. \tag{C.10}$$

To prove the theorem, one needs to show that

$$\lim_{\epsilon \to 0^+} |(W * k_\epsilon)(y^*) - W(y^*)| = 0,$$

where $W * k_\epsilon$ denotes the convolution operation. We start with the argument of the limit above and aim to bound it appropriately.

$$
\begin{aligned}
\text{(C.11)} \qquad \text{LHS} &= \left| \int_{\mathbb{R}^m} W(y^* - y) k_\epsilon(y) dy - \int_{\mathbb{R}^m} W(y^*) k_\epsilon(y) dy \right| \\
&\leq \int_{\mathbb{R}^m} |W(y^* - y) - W(y^*)| |k_\epsilon(y)| dy \\
&= \int_{\mathbb{B}_\epsilon} |W(y^* - y) - W(y^*)| |k_\epsilon(y)| dy + \int_{\mathbb{B}_\epsilon^C} |W(y^* - y) - W(y^*)| |k_\epsilon(y)| dy \\
&\leq \frac{AK}{\epsilon^m} \int_{\mathbb{B}_\epsilon} \frac{|W(y^* - y) - W(y)|}{V(y)} dy + \sum_{k=0}^{\infty} \int_{2^k\epsilon \leq \|y\| < 2^{k+1}\epsilon} |W(y^* - y) - W(y^*)| |k_\epsilon(y)| dy \\
&= AK\mathscr{A}(\epsilon) + A \sum_{k=0}^{\infty} \epsilon \int_{2^k\epsilon \leq \|y\| < 2^{k+1}\epsilon} \frac{|W(y^* - y) - W(y^*)|}{\|y\|^{m+1} V(y)} dy \\
&\leq AK\mathscr{A}(\epsilon) + A \sum_{k=0}^{\infty} \frac{\epsilon}{(2^k\epsilon)^{m+1}} \int_{2^k\epsilon \leq \|y\| < 2^{k+1}\epsilon} \frac{|W(y^* - y) - W(y^*)|}{V(y)} dy \\
&= AK\mathscr{A}(\epsilon) + A2^m \sum_{k=0}^{\infty} \frac{1}{2^k(2^{k+1}\epsilon)^m} \int_{2^k\epsilon \leq \|y\| < 2^{k+1}\epsilon} \frac{|W(y^* - y) - W(y^*)|}{V(y)} dy \\
\text{(C.12)} \qquad &\leq AK\mathscr{A}(\epsilon) + A2^m \sum_{k=0}^{\infty} \frac{\mathscr{A}(2^{k+1}\epsilon)}{2^k}.
\end{aligned}
$$

In the first line we have used the definition of convolution and property (C.1). The second line follows from the triangle inequality for integrals. In the third line we have split the integral inside and outside an open ball of radius $\epsilon > 0$. In the fourth line, for the first term we have used property (C.2), multiplied and divided by $V(y)$ and used property (C.5). For the second term, we have written the integral outside the open ball as an infinite sum over annuli. In the fifth line, we have defined for any $r > 0$

$$\mathscr{A}(r) = \frac{1}{r^m} \int_{\mathbb{B}_r} \frac{|W(y^* - y) - W(y^*)|}{V(y)} dy,$$

and used property (C.7) for the second term. In the sixth line we have bounded $\|y\|^{-(m+1}$ by $\epsilon^{-(m+1)}$ outside the ball. The seventh line is just a rewriting to obtain $(2^{k+1}\epsilon)^m$ in the denominator, and in the final line we upper bound the integrals inside the series by integrals over open balls since the integrands are non-negative. If we can show that the RHS tends to zero as $\epsilon \to 0^+$, then we are done.

First, let us notice that for any $R > 0$ the function

$$y \longmapsto \mathbb{1}_{\mathbb{B}_R}(y) \frac{|W(y^* - y) - W(y^*)|}{V(y)}$$

is integrable on $\mathbb{R}^m$. In other words, the function $|W(y^* - y) - W(y^*)|/V(y)$ is locally-integrable. Indeed

$$
\begin{aligned}
\int_{\mathbb{R}^m} \mathbb{1}_{\mathbb{B}_R}(y) \frac{|W(y^* - y) - W(y^*)|}{V(y)} dy &\leq \int_{\mathbb{B}_R} \frac{|W(y^* - y))|}{V(y)} dy + |W(y^*)| \int_{\mathbb{B}_R} \frac{1}{V(y)} dy \\
\text{(C.13)} \qquad &< \infty,
\end{aligned}
$$

using the triangle inequality for integrals, properties (C.9), and (C.4).

Before showing that (C.12) converges to 0, we need to prove that $\mathscr{A}(r)$ is a continuous function of $r > 0$, that $\mathscr{A}(r) \to 0$ as $r \to 0$, and that $\mathscr{A}(r)$ is bounded. Notice that the first two are local properties, hence we will show that they hold within any open ball of radius $R > 0$. To show that $\mathscr{A}$ is continuous at a fixed $0 < s < R$, we need to show that for any $\alpha > 0$, there exists $\beta > 0$ such that

$$|s - r| < \beta \qquad \Longrightarrow \qquad |\mathscr{A}(s) - \mathscr{A}(r)| < \alpha.$$

First, let us expand the difference $|\mathscr{A}(s) - \mathscr{A}(r)|$

$$\begin{aligned}
|\mathscr{A}(s) - \mathscr{A}(r)| &= \left| \mathscr{A}(s) - \frac{r^m}{s^m}\mathscr{A}(r) + \frac{r^m}{s^m}\mathscr{A}(r) - \mathscr{A}(r) \right| \\
&\leq \left| \mathscr{A}(s) - \frac{r^m}{s^m}\mathscr{A}(r) \right| + \left| \frac{r^m}{s^m}\mathscr{A}(r) - \mathscr{A}(r) \right| \\
&= \left| \mathscr{A}(s) - \frac{r^m}{s^m}\mathscr{A}(r) \right| + \frac{|r^m - s^m|}{s^m} |\mathscr{A}(r)| \\
&= \left| \mathscr{A}(s) - \frac{r^m}{s^m}\mathscr{A}(r) \right| + \frac{\left|(r-s)(r^{m-1} + r^{m-2}s + \cdots + s^{m-1})\right|}{s^m} |\mathscr{A}(r)| \\
&\leq \left| \mathscr{A}(s) - \frac{r^m}{s^m}\mathscr{A}(r) \right| + \frac{|r-s|\, mR^{m-1}}{s^m} |\mathscr{A}(r)| \qquad \text{(C.14)}
\end{aligned}$$

The rest of the proof is concerned with showing that both terms go to zero as $|r - s|$ goes to zero. For the second term, it's enough to show that $|\mathscr{A}(r)|$ is bounded above. When $s < r < R$, we get

$$\begin{aligned}
|\mathscr{A}(r)| &= \frac{1}{r^m} \left| \int_{\mathbb{B}_r} \mathbb{1}_{\mathbb{B}_R}(y) \frac{|W(y^* - y) - W(y^*)|}{V(y)} dy \right| \\
&< \frac{1}{s^m} \int_{\mathbb{B}_r} \mathbb{1}_{\mathbb{B}_R}(y) \frac{|W(y^* - y) - W(y^*)|}{V(y)} dy \\
&< \infty,
\end{aligned}$$

where the penultimate inequality follows from $1/r < 1/s$, and the last line from the fact that the integrand is integrable on $\mathbb{R}^n$ and hence finite. For the scenario $r \leq s$, we assume that r is bounded away from zero, for instance $r > s/2$ suffices

$$|\mathscr{A}(r)| < \frac{2^m}{s^m} \int_{\mathbb{B}_r} \mathbb{1}_{\mathbb{B}_R}(y) \frac{|W(y^* - y) - W(y^*)|}{V(y)} dy < \infty.$$

Consider now the first term in (C.14). When $r \leq s$ then

$$\mathscr{A}(s) - \frac{r^m}{s^m}\mathscr{A}(r) = \frac{1}{s^m} \int_{\mathbb{B}_s \setminus \mathbb{B}_r} \mathbb{1}_{\mathbb{B}_R}(y) \frac{|W(y^* - y) - W(y^*)|}{V(y)} dy \geq 0,$$

and since the integrand is integrable on $\mathbb{R}^m$ it follows by absolute continuity (see Proposition 1.12 in [134]) that for any $\alpha_0 > 0$, there exists $\beta_0 > 0$ such that

$$\text{Leb}(\mathbb{B}_s \setminus \mathbb{B}_r) < \beta_0 \qquad \Longrightarrow \qquad \left| \mathscr{A}(s) - \frac{r^m}{s^m}\mathscr{A}(r) \right| < \alpha_0.$$

Since, if we denote by v_m the volume of an m-dimensional unit ball, then

$$\text{Leb}(\mathbb{B}_s \setminus \mathbb{B}_r) = v_m(s-r)(s^{m-1} + s^{m-2}r + \cdots + r^{m-1}) < v_m(s-r)R^{m-1},$$

and s is fixed, it follows that as $r \uparrow s$ the first term in (C.14) goes to zero. When $s < r < R$ then

$$\begin{aligned} 0 \le \mathscr{A}(r) - \frac{s^m}{r^m}\mathscr{A}(s) &= \frac{1}{r^m}\int_{\mathbb{B}_r \setminus \mathbb{B}_s} \mathbb{1}_{\mathbb{B}_R}(y)\frac{|W(y^*-y)-W(y^*)|}{V(y)}dy \\ &< \frac{1}{s^m}\int_{\mathbb{B}_r \setminus \mathbb{B}_s} \mathbb{1}_{\mathbb{B}_R}(y)\frac{|W(y^*-y)-W(y^*)|}{V(y)}dy. \end{aligned}$$

Once more, by absolute continuity, we can make the integral above arbitrarily small by making $r-s$ small enough. By writing

$$\left|\mathscr{A}(s) - \frac{r^m}{s^m}\mathscr{A}(r)\right| = \frac{r^m}{s^m}\left|\mathscr{A}(r) - \frac{s^m}{r^m}\mathscr{A}(s)\right| \le \left|\mathscr{A}(r) - \frac{s^m}{r^m}\mathscr{A}(s)\right| \qquad \text{since } s < r,$$

we can make the first term in (C.14) arbitrarily small even when $s < r < R$. It follows that $\mathscr{A}$ is a continuous function for any $0 < s < R$.

Next, we want to show that $\mathscr{A} \to 0$ as $r \to 0$, which means that for any $\alpha > 0$, there exists $\beta > 0$ such that

$$r < \beta \qquad \Longrightarrow \qquad |\mathscr{A}(r)| < \alpha.$$

To show this, we will make use of a consequence of the Lebesgue Differentiation Theorem (see Subsection 1.2 in Chapter 3 of [134]).

Theorem C.5 (Consequence of the Lebesgue Differentiation Theorem)**.** *Let f be locally-integrable on $\mathbb{R}^m$. If x is in the Lebesgue set of f then $f(x)$ is finite, and*

$$\lim_{\substack{Leb(B)\to 0 \\ x\in B}} \frac{1}{Leb(B)}\int_B f(y)dy = f(x). \tag{C.15}$$

Let $r > 0$ be arbitrary. Since $\lim_{r\to 0}\mathscr{A}(r) = 0$ is a local property we can once again study what happens inside any ball with $R > r$ and consider the change of variables $z = y^* - y$

$$\begin{aligned} |\mathscr{A}(r)| &= \left|\frac{1}{r^m}\int_{\mathbb{B}_r} \mathbb{1}_{\mathbb{B}_R}(y)\frac{|W(y^*-y)-W(y^*)|}{V_*(y)}dy\right| \\ &= \left|\frac{1}{r^m}\int_{\mathbb{B}(y^*,r)} \mathbb{1}_{\mathbb{B}(y^*,R)}(y^*-z)\frac{|W(z)-W(y^*)|}{V_*(y^*-z)}dz\right| \end{aligned}$$

The fact that $\mathscr{A}(r) \to 0$ as $r \to 0$ simply follows from Theorem C.5. Next, we want to show that $\mathscr{A}(r)$ is bounded.

$$\begin{aligned} \mathscr{A}(r) &= \frac{1}{r^m}\int_{\mathbb{B}_r} \frac{|W(y^*-y)-W(y^*)|}{V_*(y)}dy \\ &\le \frac{1}{r^m}\int_{\mathbb{B}_r} \frac{|W(y^*-y))|}{V_*(y)}dy + |W(y^*)|\int_{\mathbb{B}_r}\frac{1}{V_*(y)}dy \\ &\le \frac{1}{r^m}\int_{\mathbb{B}_r} \frac{|W(y^*-y))|}{V_*(y)}dy + |W(y^*)|v_m \\ &< \infty \end{aligned}$$

In the second line we have used the triangle inequality, and in the third line property (C.4). The final line uses property (C.9).

Finally, we use all these ingredients to show that (C.12) converges to 0 as $\epsilon \to 0^+$. Choose any $\alpha > 0$, then it's clear from $\lim_{r\to 0} \mathscr{A}(r) = 0$ that there exists $\beta_1 > 0$ such that $AK\mathscr{A}(\epsilon) < \alpha/3$ whenever $\epsilon < \beta_1$. Next, choose $N(\alpha)$ large enough so that

$$\left| \sum_{k \geq N(\alpha)} \frac{\mathscr{A}(2^{k+1}\epsilon)}{2^k} \right| \leq \overline{\mathscr{A}} \sum_{k \geq N(\alpha)} \frac{1}{2^k} < \alpha/3,$$

where $\overline{\mathscr{A}} > 0$ is an upper bound for the function $\mathscr{A}$. It remains to be shown that the sum of the first $N(\alpha)$ terms can be made arbitrarily small. This follows from $\mathscr{A}(r) \to 0$ as $r \to 0$, since

$$\lim_{\epsilon\to 0} \sum_{k=0}^{N(\alpha)-1} \frac{\mathscr{A}(2^{k+1}\epsilon)}{2^k} = \sum_{k=0}^{N(\alpha)} \lim_{\epsilon\to 0} \frac{\mathscr{A}(2^{k+1}\epsilon)}{2^k} = 0,$$

and in particular one can always find $\beta_2 > 0$ such that whenever $\epsilon < \beta_2$

$$\left| \sum_{k=0}^{N(\alpha)-1} \frac{\mathscr{A}(2^{k+1}\epsilon)}{2^k} \right| < \alpha/3.$$

Therefore setting $\beta = \min(\beta_1, \beta_2)$ completes the proof since for arbitrary $\alpha > 0$, one can find $\beta > 0$ with

$$|(W * k_\epsilon)(y^*) - W(y^*)| < \alpha \qquad \text{whenever } \epsilon < \beta.$$

■

Example C.1.1.2 (Example C.1.1.1 continued)**.** *Let $y^* \in \mathbb{R}^m$ be fixed, and consider once more the Gaussian kernel*

$$k_\epsilon(y) = \frac{1}{(2\pi)^{m/2}\epsilon^m} \exp\left(-\frac{\|y\|^2}{2\epsilon^2}\right).$$

We would like to show that the tail property in Theorem C.4 is satisfied by the Gaussian kernel for a particular family of functions V. Specifically, we wish to show that there exists a constant A and a maximum tolerance $\epsilon_{max} > 0$, such that

$$|k_\epsilon(y)| \leq \frac{A\epsilon}{\|y\|^{m+1}V(y)} \qquad \forall\, y \neq 0, \forall\, 0 < \epsilon < \epsilon_{max}, \tag{C.16}$$

where

$$V(y) = \exp\left(\frac{\|y\|^2}{c^2}\right) \qquad c \neq 0.$$

Notice we can write the kernel as follows

$$k_\epsilon(y) = \frac{\epsilon}{(2\pi)^{m/2}\|y\|^{m+1}V(y)} \cdot h_c(\|y\|, \epsilon),$$

where

$$h_c(\|y\|, \epsilon) := \frac{\|y\|^{m+1}}{\epsilon^{m+1}} \exp\left(-\frac{\|y\|^2}{2\epsilon^2} + \frac{\|y\|^2}{c^2}\right).$$

Simply define $r = \|y\|/\epsilon$*, then* h_c *becomes*

$$h_c(r,\epsilon) = r^{m+1}\exp\left(-\frac{r^2}{2} + \frac{\epsilon^2 r^2}{c^2}\right),$$

and if we choose $\epsilon_{max} < c/\sqrt{2}$ *then* h_c *is bounded above as a function of* $r > 0$ *and* $\epsilon \in [0, \epsilon_{max})$

$$\sup_{r>0} h_c(\|y\|, c) < \infty \qquad \forall\, \epsilon < 0 < \epsilon_{max}.$$

Given any $c \neq 0$*, one can set* $\epsilon_{max} < c/\sqrt{2}$ *so that property* (C.16) *is satisfied. In practice, however, one would typically run the THUG sampler for a fixed* $\epsilon_0 > 0$*, in which case one should set* $c \geq \sqrt{2}\epsilon_0$*, and this would in turn restrict the class of functions* ψ *for which Theorem C.4 holds.*

C.2 Projections, Reflections and Squeezing

The following theorem collects together several standard notion of linear algebra.

Theorem C.6 (Properties of Projection and Reflection Matrices)**.** *Let* A *be a full row-rank* $m \times n$ *matrix with* $n > m$*. Then the following properties are true*

1. $\mathrm{AA}^\top$ *is invertible.*
2. $\mathrm{P} = \mathrm{A}^\top(\mathrm{AA}^\top)^{-1}\mathrm{A}$ *is a projection matrix with rank* m*.*
3. $\mathrm{R} = \mathrm{I} - 2\mathrm{P}$ *is an orthogonal matrix with determinant* $(-1)^m$*.*

Proof. It's a standard result that $\mathrm{rank}(\mathrm{AA}^\top) = \mathrm{rank}(\mathrm{A})$ which immediately tells us $\mathrm{AA}^\top$ is full-rank, and hence invertible. Notice the converse is not true. Simply take

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

then A is not full row-rank, but $\mathrm{AA}^\top$ is invertible. Continuing our proof, since $\mathrm{AA}^\top$ is invertible, P is well-defined and clearly

$$\mathrm{PP} = \mathrm{A}^\top(\mathrm{AA}^\top)^{-1}\mathrm{AA}^\top(\mathrm{AA}^\top)^{-1}\mathrm{A} = \mathrm{P},$$

so it is a projection matrix. It is a standard result in Ordinary Least Squares that

$$\mathrm{rank}(\mathrm{A}^\top(\mathrm{AA}^\top)^{-1}\mathrm{A}) = \mathrm{rank}(\mathrm{A})$$

which immediately gives $\mathrm{rank}(\mathrm{P}) = m$. It is also a standard result that a projection matrix P has eigenvalues 1, with multiplicity rank(P), and 0, with multiplicity $n - \mathrm{rank}(\mathrm{P})$. Therefore the eigenvalues of R are -1 with multiplicity m and 1 with multiplicity $n - m$. Since the determinant is the product of the eigenvalues one has $\det(\mathrm{R}) = (-1)^m$. ■

The following theorem determines the proposal distribution for the velocity variable in THUG.

Theorem C.7 (Squeezed Normal)**.** *Let* A *be a full row-rank* $m \times n$ *matrix and for* $\alpha \in \mathbb{R}$ *define the matrix*

$$\mathrm{S}_\alpha = \mathrm{I}_n - \alpha \mathrm{A}^\top (\mathrm{A}\mathrm{A}^\top)^{-1} \mathrm{A}.$$

Then S_α *is invertible if and only if* $\alpha \neq 1$*. In addition, when* $\mathrm{A} = \mathrm{J}_x$ *where* J_x *is the Jacobian evaluated at* x *of any smooth function* $f : \mathbb{R}^n \to \mathbb{R}^m$*, the normal distribution* $\mathcal{N}(0, \mathrm{S}_\alpha \mathrm{S}_\alpha^\top)$ *has variance* 1 *along any basis vector of the tangent space, and variance* $(1-\alpha)^2$ *along any basis vector of the normal space.*

Proof. S_α is invertible if and only if its determinant is non zero. Its determinant is given by the product of its eigenvalues. Denote $\mathrm{P} = \mathrm{A}^\top (\mathrm{A}\mathrm{A}^\top)^{-1} \mathrm{A}$ then if $v \in \mathbb{R}^n \setminus \{0\}$ is an eigenvector of P with eigenvalue λ

$$\mathrm{P}v = \lambda v,$$

then clearly v is also an eigenvector of S_α with eigenvalue $\tilde{\lambda} = 1 - \alpha\lambda$. As seen in the previous theorem, it is well-known that the rank of P is m and that its eigenvalues are 1 and 0 with multiplicity m and $n - m$ respectively. Thus

$$\det(\mathrm{S}_\alpha) = (1-\alpha)^m,$$

which vanishes only for $\alpha = 1$. Let $\hat{t}_x$ be any basis vector of the tangent space $\mathcal{T}_x$ and suppose $\mathrm{A} = \mathrm{J}_x$. Then

$$\hat{t}_x^\top \mathrm{S}_\alpha \mathrm{S}_\alpha^\top \hat{t}_x = \hat{t}_x^\top \mathrm{S}_\alpha \hat{t}_x = \hat{t}_x^\top \hat{t}_x = 1.$$

Similarly, let $\hat{n}_x$ be any basis vector of the normal space $\mathcal{N}_x$. Then

$$\hat{n}_x^\top \mathrm{S}_\alpha \mathrm{S}_\alpha^\top \hat{n}_x = (1-\alpha)\hat{n}_x^\top \mathrm{S}_\alpha \hat{n}_x = (1-\alpha)^2 \hat{n}_x^\top \hat{n}_x = (1-\alpha)^2.$$

■

C.3 Derivation of the Bounce Mechanism

Theorem C.8 (Dynamic of a Particle with Constant Speed and Centripetal Acceleration on a Level Set of a Smooth Function)**.** *A particle* x_t *moving with constant speed and centripetal acceleration on the level set* $f^{-1}(y)$ *of a smooth function* $f : \mathbb{R}^n \to \mathbb{R}^m$ *with full row-rank Jacobian everywhere, follows the ODE below*

$$\ddot{x}_t = -J_t^\top (J_t J_t^\top)^{-1} D_f(\dot{x}_t, \dot{x}_t)[x_t],$$

where J_t *is the Jacobian matrix of* f *at* x_t*, and* $D_f[\cdot](\cdot,\cdot)$ *is a bilinear form representing the second-order derivative tensor of* f*.*

Proof. The Fundamental Theorem of Linear Algebra decomposes the ambient space as

$$\mathbb{R}^n = \text{RowSpace}(J_t) \oplus \text{NullSpace}(J_t),$$

which means $\ddot{x}_t = J_t^\top \lambda + \ddot{x}_t^{\text{null}}$ for some $\lambda \in \mathbb{R}^m$ and where $\ddot{x}_t^{\text{null}}$ lies on $\mathscr{T}_{x_t}\mathscr{M}$. Differentiating the condition $f(x) = y$ twice with respect to time yields

$$D_f(\dot{x}_t, \dot{x}_t)[x_t] + J_f(x_t)\ddot{x}_t = 0.$$

Substituting the decomposition for the acceleration and since J_t has full row-rank, we can solve for λ

$$\lambda = -(J_t J_t^\top)^{-1} D_f(\dot{x}_t, \dot{x}_t)[x_t],$$

where $D_f[\cdot](\cdot,\cdot)$ is a tensor representing the second-order derivative of f, i.e. a bilinear form. This straightforwardly leads to the ODE

$$\ddot{x}_t = -J_t^\top (J_t J_t^\top)^{-1} D_f(\dot{x}_t, \dot{x}_t)[x_t] + \ddot{x}_t^{\text{null}}.$$

Assuming only a centripetal acceleration with $\ddot{x}_t^{\text{null}} = 0$ we recover the ODE

$$\ddot{x}_t = -J_t^\top (J_t J_t^\top)^{-1} D_f(\dot{x}_t, \dot{x}_t)[x_t].$$

■

Solving for λ has transformed the ODE system into a non-separable dynamic because λ is both a function of x and v. One could go ahead and integrate the system

$$\begin{aligned}
\dot{x}_t &= v_t \\
\dot{v}_t &= -J_t^\top (J_t J_t^\top)^{-1} D_f(v_t, v_t)[x_t]
\end{aligned}$$

using a generalized Stormer-Verlet method but the velocity update would be implicit due to the presence of v_{t+1} on both sides of the update. Instead, we replace v_{t+1} on the RHS with v_t. While in general this is not advisable, it turns out to be a good choice in our setting. The resulting discretization for a step size $\delta > 0$ is

$$\begin{aligned}
x_{t+1/2} &= x_t + (\delta/2) v_t \\
v_{t+1} &= v_t - \delta J_f(x_{t+1/2})^\top (J_f(x_{t+1/2}) J_f(x_{t+1/2})^\top)^{-1} D_f(v_t, v_t)[x_{t+1/2}] \\
x_{t+1} &= x_{t+1/2} + (\delta/2) v_{t+1}.
\end{aligned}$$

At this point, we would like to avoid computing the rank-three tensor representing second-order derivative information. With this in mind, consider $x_t = x_{t+1/2} - (\delta/2)v_t$ and Taylor expand J_f and multiply on the right by v_t to obtain

$$J_f(x_t)v_t = J_f(x_{t+1/2})v_t + \frac{\delta}{2} D_f(v_t, v_t)[x_{t+1/2}] + \mathscr{O}(\delta^2).$$

Since v_t is in the tangent space the LHS disappears and we get

$$-\delta D_f(v_t, v_t)[x_{t+1/2}] = 2J_f(x_{t+1/2})v_t + \mathcal{O}(\delta^2). \tag{C.17}$$

Plugging this into the discretization we obtain the proposal mechanism of the THUG algorithm

$$\begin{aligned} x_{t+1/2} &= x_t + (\delta/2)v_t \\ v_{t+1} &= v_t - 2J_f(x_{t+1/2})^\top (J_f(x_{t+1/2})J_f(x_{t+1/2})^\top)^{-1} J_f(x_{t+1/2})v_t \\ x_{t+1} &= x_{t+1/2} + (\delta/2)v_{t+1}. \end{aligned} \tag{C.18}$$

Essentially the bounce mechanism is approximating second order curvature information. Generalized Stormer-Verlet integrators for partitioned systems have many appealing properties such as being second-order, symmetric and symplectic, but are typically implicit. Replacing v_{t+1} with v_t meant leaving this class of integrators and hence losing all these properties. However, the second-order bounce approximation (C.17) recovered two properties:

- The integrator is now *symmetric* thanks to the symmetric matrix

$$P_{1/2} = J_{1/2}^\top (J_{1/2}J_{1/2}^\top)^{-1} J_{1/2}.$$

- The integrator is now volume-preserving since it has Jacobian determinant 1

The advantage of this integrator is that it is now *explicit*. Nonetheless, at this point the reader should wonder what is the order of this method. It turns out that if one assumes that $D_f(\cdot,\cdot)[\cdot]$ is both bounded and Lipschitz continuous then it is possible to replicate the proof given in [94] to bound the change in f between the initial point x_0 and the final point x_B (see Theorem 2.2).

C.4 Bases for Normal and Tangent Spaces

This section summarises standard Linear Algebra results that can be used t find bases for the tangent and the normal space via the QR decomposition. Let $\mathrm{J} \in \mathbb{R}^{m\times n}$ be a full-row rank Jacobian matrix and let $\mathrm{J}^\top = \mathrm{QR}$ denote the QR decomposition of its transpose, where $\mathrm{Q} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $\mathrm{R} \in \mathbb{R}^{n\times m}$ is an upper-triangular rectangular matrix with the last $n-m$ rows being zero, since $\mathrm{rank(J)} < n$. The Fundamental Theorem of Linear Algebra implies

$$\mathbb{R}^n = \mathrm{RowSpace(J)} \oplus \mathrm{NullSpace(J)}$$

with dimensions m and $n-m$ respectively. Denote the columns of the Q matrix as $\mathrm{Q} = [q_1, \ldots, q_n]$, and define the following two sets of such vectors

$$\begin{aligned} \mathscr{B}_N &= \{q_{m+1}, \ldots, q_n\} \\ \mathscr{B}_R &= \{q_1, \ldots, q_m\}. \end{aligned}$$

One can easily show that $\mathscr{B}_N$ is a basis for NullSpace(J). Indeed all one needs to show is that they belong to the null space, since they are already orthonormal and $n-m$ vectors. For each $j \in \{m+1,\dots,n\}$ one has

$$\mathrm{J}q_j = \mathrm{R}^\top \mathrm{Q}^\top q_j = (\mathrm{R})_{:,j} = 0$$

where $(\mathrm{R})_{:,j}$ denotes the j^{th} row of R. It is also possible to show that $\mathscr{B}_R$ is a basis for the row space of J. One only needs to show that for each q_j with $j \in \{1,\dots,m\}$ it is possible to find $\lambda_j \in \mathbb{R}^m$ such that

$$\mathrm{J}^\top \lambda_j = q_j.$$

Multiplying by Q on both sides gives

$$\mathrm{R}\lambda_j = e_j.$$

However it is possible to write R as

$$\mathrm{R} = \begin{bmatrix} \mathrm{R}_1 \\ 0_{(n-m)\times m} \end{bmatrix}$$

where R_1 is an $m \times m$ upper triangular matrix, and therefore

$$\begin{bmatrix} \mathrm{R}_1 \lambda_j \\ 0_{(n-m)\times 1} \end{bmatrix} = \begin{bmatrix} \tilde{e}_j \\ 0_{(n-m)\times 1} \end{bmatrix}$$

where $\tilde{e}_j$ is now a standard basis vector in $\mathbb{R}^m$. In order for this to be solvable we require R_1 to be invertible, i.e. all of its diagonal entries are non-zero. Suppose that $(\mathrm{R}_1)_{kk} = 0$ for some $k \in \{1,\dots,m\}$. Then the first k columns would be linearly dependent and hence $\mathrm{rank}(\mathrm{R}_1) < m$. However, if we construct $\mathrm{Q} = [\mathrm{Q}_{1:m}; \mathrm{Q}_{m+1:n}]$ with

$$\mathrm{Q}_{1:m} = \begin{bmatrix} q_1 & \cdots q_m \end{bmatrix} \qquad \mathrm{Q}_{m+1:n} = \begin{bmatrix} q_{m+1} & \cdots & q_n \end{bmatrix}$$

then clearly

$$m = \mathrm{rank}(\mathrm{J}) = \mathrm{rank}(\mathrm{QR}) = \mathrm{rank}(\mathrm{Q}_{1:m}\mathrm{R}_1) = \mathrm{rank}(\mathrm{R}_1)$$

since $\mathrm{Q}_{1:m}$ is full rank. By contradiction this means that all the diagonal elements of R_1 are non-zero and the solution is

$$\lambda_j = \mathrm{R}_1^{-1} \tilde{e}_j.$$

C.5 Sampling on the tangent space

To sample from a standard Gaussian on the tangent space $\mathscr{T}_x$, [8] sampled in the ambient space and then projected this onto $\mathscr{T}_x$, i.e.

$$\tilde{v} \sim \mathscr{N}(0, \mathrm{I}_n)$$
$$v = \mathrm{T}_x \tilde{v}.$$

[153] used a different approach and first sampled coefficients and then multiplied them by the basis of the tangent space which, by the previous section in the Appendix, is $\mathrm{Q}_{m+1:n}$, i.e.

$$\tilde{w} \sim \mathcal{N}(0, \mathrm{I}_{n-m})$$
$$w = \mathrm{Q}_{m+1:n}\tilde{w}.$$

One can recover the latter approach from the former by simply determining the coefficients of the basis. This is easily done

$$\tilde{w} = \left[\mathrm{Q}^{-1}v\right]_{m+1:n}.$$

Clearly one has $\|w\| = \|\tilde{w}\| = \|v\|$, which explains the different acceptance ratios in [153] and [8].

C.6 Time-reversibility and volume-preservation of bounce

To prove time-reversibility simply let $(x', v') = (x + (\delta/2)v + (\delta/2)\mathrm{R}v, \mathrm{R}v) = \phi \circ \mathbb{B}_{\mathrm{R},\delta}(x, v)$. Then

$$(x'', v'') = \phi \circ \mathbb{B}_{\mathrm{R}}(x', v') = x + (\delta/2)v + (\delta/2)\mathrm{R}v - (\delta/2)\mathrm{R}v - (\delta/2)v, \mathrm{R}\mathrm{R}v) = (x, v),$$

which proves that $\phi \circ \mathbb{B}_{\mathrm{R},\delta}$ is an involution. To prove volume-preservation, notice that a bounce is the composition of three operations

$$\mathbb{B}_{\mathrm{R}}(x, v) = \psi_{\mathrm{move}} \circ \psi_{\mathrm{reflect}} \circ \psi_{\mathrm{move}}(x, v)$$

where

$$\psi_{\mathrm{move}}(x, v) = (x + (\delta/2)v, v)$$
$$\psi_{\mathrm{reflect}}(x, v) = (x, \mathrm{R}(x)v).$$

Here we consider the general case in which the orthogonal matrix R is allowed to depend on x. The Jacobian determinant of the "move" operation is clearly one

$$|\mathrm{J}_{\psi_{\mathrm{move}}}(x, v)| = \left|\begin{pmatrix} \mathrm{I} & 0 \\ \frac{\delta}{2}\mathrm{I} & \mathrm{I} \end{pmatrix}\right| = 1.$$

There are multiple ways to see that ψ_{reflect} is volume-preserving. Let A be any open set in $\mathbb{R}^{2n}$ with volume Vol(A) and define the image of A under $\mathrm{R}(x)$ as

$$\mathrm{R}(x)\mathrm{A} := \{\mathrm{R}(x)a : a \in \mathrm{A}\}.$$

The volume of this new set is then the same of A

$$\mathrm{Vol}(\mathrm{R}(x)\mathrm{A}) = \int_{\mathrm{R}(x)\mathrm{A}} dz = \int_{\mathrm{A}} |\det \mathrm{R}(x)|\, dz = \int_{\mathrm{A}} dz = \mathrm{Vol}(\mathrm{A}),$$

using the change of variables formula and since $\mathrm{R}(x)$ is an orthogonal matrix and thus has determinant ± 1. As a result, the reflection operation is volume-preserving. Alternatively, one may show

$$\left|\mathrm{J}_{\psi_{\mathrm{reflect}}}(x, v)\right| = \left|\begin{pmatrix} \mathrm{I} & \left(\frac{d}{dx}\mathrm{R}(x)\right)v \\ 0 & \mathrm{R}(x) \end{pmatrix}\right| = |\mathrm{R}(x)| = 1,$$

although this requires the additional assumption that $\mathrm{R}(x)$ is differentiable with respect to x.

C.7 Bounce Precision

C.7.1 One step distance

In this subsection, we aim find an expression for $f(x_{b+1}) - f(x_b)$ in terms of the step size δ. This is simply an adaptation of the proof given in [94]. First, write

$$x_b = x'_b - \frac{\delta}{2} v_b$$
$$x_{b+1} = x'_b + \frac{\delta}{2} v_{b+1},$$

and Taylor expand f around x'_b for both expressions

$$f(x_b) = f(x'_b) - \frac{\delta}{2} \mathrm{J}(x'_b) v_b + \frac{\delta^2}{8} \mathrm{H}(v_b, v_b)[x_b^+]$$
$$f(x_{b+1}) = f(x'_b) + \frac{\delta}{2} \mathrm{J}(x'_b) v_{b+1} + \frac{\delta^2}{8} \mathrm{H}(v_{b+1}, v_{b+1})[x_{b+1}^-]$$

where J and H are the Jacobian matrix and Hessian tensor respectively, x_b^+ lies on the line between x_b and x'_b, and x_{b+1}^- on the line between x'_b and x_{b+1}. Subtracting the first expansion from the second gives

$$f(x_{b+1}) - f(x_b) = \frac{\delta^2}{8} \left[\mathrm{H}(v_{b+1}, v_{b+1})[x_{b+1}^-] - \mathrm{H}(v_b, v_b)[x_b^+] \right],$$

since $\mathrm{J}(x'_b)(v_{b+1} + v_b) = 0$.

C.7.2 Full integration

In this subsection we show that integrating for a time $T = B\delta$ leads to an $\mathcal{O}(\delta^2)$ error given some sufficiency conditions. First, let us define the induced norm for the order-three tensor H as

$$\|\mathrm{H}[x]\|_I = \sup_{v_1, v_2 \neq 0} \frac{|\mathrm{H}(v_1, v_2)[x]|}{\|v_1\| \|v_2\|}.$$

We shall assume that the Hessian tensor is bounded, that is there exists $\beta \in (0, \infty)$ such that

$$\sup_x \|\mathrm{H}[x]\|_I \leq \beta.$$

In addition, we assume the Hessian tensor is Lipschitz continuous, that is there exists $\gamma > 0$ such that

$$\|\mathrm{H}[x] - \mathrm{H}[y]\|_I \leq \gamma \|x - y\|.$$

Given these two assumptions, we can again follow the proof given in [94].

$$\begin{aligned} f(x_B) - f(x_0) &= \sum_{b=1}^{B} f(x_b) - f(x_{b-1}) \\ &= \frac{\delta^2}{8} \sum_{b=1}^{B} \left[\mathrm{H}(v_b, v_b)[x_b^-] - \mathrm{H}(v_{b-1}, v_{b-1})[x_{b-1}^+] \right] \\ &= \frac{\delta^2}{8} \left(\mathrm{H}(v_B, v_B)[x_B^-] - \mathrm{H}(v_0, v_0)[x_0^+] + \sum_{b=1}^{B-1} \left(\mathrm{H}[x_b^-] - \mathrm{H}[x_b^+] \right)(v_b, v_b) \right). \end{aligned}$$

The points x_b^- and x_b^+ lie on the line segments joining the two consecutive bounce points x'_{b-1} and x'_b. In addition, since between bounces we reuse the same velocity $x'_b = x'_{b-1} + \delta v_b$, and so

$$\begin{aligned}
\|(\mathrm{H}[x_b^-] - \mathrm{H}[x_b^+])(v_b, v_b)\| &\leq \|v_b\|^2 \|\mathrm{H}[x_b^-] - \mathrm{H}[x_b^+]\|_I && \text{Induced norm} \\
&\leq \gamma \|v_b\|^2 \|x_b^+ - x_b^-\| && \text{Lipschitz Continuity} \\
&\leq \gamma\delta \|v_b\|^3.
\end{aligned}$$

Similarly, we also have

$$\begin{aligned}
\|\mathrm{H}(v_B, v_B)[x_B^-] - \mathrm{H}(v_0, v_0)[x_0^+]\| &\leq \|\mathrm{H}(v_B, v_B)[x_B^-]\| + \|\mathrm{H}(v_0, v_0)[x_0^+]\| && \text{Triangle Inequality} \\
&\leq \|v_B\|^2 \|\mathrm{H}[x_B^-]\|_I + \|v_0\|^2 \|\mathrm{H}[x_0^+]\|_I && \text{Induced Norm} \\
&\leq \beta(\|v_B\|^2 + \|v_0\|^2) && \text{Bounded Hessian} \\
&= 2\beta \|v_0\|^2 && \text{Reflections are unitary.}
\end{aligned}$$

Overall, this means we can bound the difference

$$\|f(x_B) - f(x_0)\| \leq \frac{\delta^2 \|v_0\|^2}{8} \left(2\beta + \gamma \|T v_0\|\right)$$

where $T = B\delta$.

C.7.3 Bounce Precision when squeezing

Clearly for THUG we will have the same bound but for $w_0 = (\mathrm{I} - \alpha \mathrm{N}_0) v_0$

$$\|f(x_B^{\mathrm{THUG}}) - f(x_0)\| \leq \frac{\delta^2 \|w_0\|^2}{8} (2\beta + \gamma \|T w_0\|).$$

First of all, notice that since $\mathcal{N}_x$ and $\mathcal{T}_x$ are orthogonal compliments one can always write

$$v_0 = \mathrm{N}_0 v_0 + \mathrm{T}_0 v_0,$$

and its norm is then

$$\|v_0\|^2 = v_0^\top \mathrm{N}_0 v_0 + v_0^\top \mathrm{T}_0 v_0 =: \|v_0^\perp\|^2 + \|v_0^\parallel\|^2,$$

where we have defined the components of v_0 on the normal space $v_0^\perp$ and on the tangent space $v_0^\parallel$. The norm of the squeezed velocity is then, as expected

$$\begin{aligned}
\|w_0\|^2 &= v_0^\top (\mathrm{I} - \alpha \mathrm{N}_0)^\top (\mathrm{I} - \alpha \mathrm{N}_0) v_0 \\
&= v_0^\top (\mathrm{I} - \alpha(2 - \alpha) \mathrm{N}_0) v_0 \\
&= \|v_0\|^2 - \alpha(2 - \alpha) \|v_0^\perp\|^2,
\end{aligned}$$

which can also be written as $(1-\alpha)^2\|v_0^\perp\|^2 + \|v_0^\parallel\|^2$. The bound is improved whenever $\alpha > 0$ and $\|v_0^\perp\| > 0$

$$\begin{aligned}
\|f(x_B^{\text{THUG}}) - f(x_0)\| &\leq \frac{\delta^2\|w_0\|^2}{8}\left(2\beta + \gamma\|Tw_0\|\right) \\
&< \frac{\delta^2\|w_0\|^2}{8}\left(2\beta + \gamma\|Tv_0\|\right) && \|w_0\| < \|v_0\| \\
&\leq \frac{\delta^2}{8}\left(\|v_0\|^2 - \alpha(2-\alpha)\|v_0^\perp\|^2\right)(2\beta + \gamma\|Tv_0\|) \\
&= \frac{\delta^2\|v_0\|^2}{8}(2\beta + \gamma\|Tv_0\|) - \frac{\alpha(2-\alpha)\delta^2\|v_0^\perp\|^2}{8}(2\beta + \gamma\|Tv_0\|) \\
&= \mathscr{B}_{\text{HUG}} - \frac{\alpha(2-\alpha)\delta^2\|v_0^\perp\|^2}{8}(2\beta + \gamma\|Tv_0\|) \\
&< \mathscr{B}_{\text{HUG}}.
\end{aligned}$$

C.8 Change in Kinetic Energy for THUG

C.8.1 Preliminaries

For any $x \in \mathbb{R}^n$, we shall denote by J_x the $m \times n$ Jacobian matrix of f at x. The pseudo-inverse of J_x is an $n \times m$ matrix given by $\mathrm{J}_x^+ = \mathrm{J}_x^\top(\mathrm{J}_x\mathrm{J}_x^\top)^{-1}$ and it is well-defined whenever J_x is full row-rank, which we assume to be the case almost everywhere. As per the main text, we shall denote by $\mathrm{N}_x = \mathrm{J}_x^+\mathrm{J}_x$ the projection matrix for the normal space at x. Corollary 4.2 in [60] is reproduced below and tells us the form of the Fréchet derivative of N_x. We note that Taylor's theorem with both Lagrangian and integral remainders are available for Fréchet derivatives, see for example section 5.6 in [27].

Theorem C.9 (Fréchet Derivative of Normal Projection Matrix)**.** *Let* J_x *be full row-rank everywhere on* $\mathbb{R}^n$*, and let* $\mathrm{N}_x = \mathrm{J}_x^+\mathrm{J}_x$*. Then the Fréchet derivative of* $\mathrm{N}(x)$ *is*

$$D\mathrm{N}_x = \mathrm{J}_x^+(D\mathrm{J}_x)(\mathrm{I} - \mathrm{N}_x) + (\mathrm{J}_x^+(D\mathrm{J}_x)(\mathrm{I} - \mathrm{N}_x))^\top. \tag{C.19}$$

Recall that by definition $\mathrm{T}_x = \mathrm{I} - \mathrm{N}_x$, and that we denote the Hessian tensor by $\mathrm{H}_x = D\mathrm{J}_x$. Both $D\mathrm{N}_x$ and H_x are third-order tensors and they can be thought of as linear maps taking $v \in \mathbb{R}^n$ as input, and output a matrix denoted $D\mathrm{N}_x v$ or $\mathrm{H}_x v$ respectively. This allows us to express Equation (C.19) in a more useful and familiar form

$$D\mathrm{N}_x v = \mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x + (\mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x)^\top \qquad \forall v \in \mathbb{R}^n.$$

The corollary below lists some consequence of the theorem.

Corollary C.1 (Properties of $D\mathrm{N}_x$)**.** *Let $D\mathrm{N}_x$ be as in Theorem C.9, then for any $v \in \mathbb{R}^n$ the following properties hold.*

$$\begin{aligned} \mathrm{N}_x(D\mathrm{N}_x v)\mathrm{N}_x &= 0 \\ \mathrm{N}_x(D\mathrm{N}_x v) + (D\mathrm{N}_x v)\mathrm{N}_x &= D\mathrm{N}_x \\ w^\top (D\mathrm{N}_x v) w &= 2w^\top \mathrm{J}_x^+ (\mathrm{H}_x v)\mathrm{T}_x w \qquad \forall w \in \mathbb{R}^n. \end{aligned}$$

Proof. Firstly, notice that $\mathrm{N}_x\mathrm{T}_x = 0$ and

$$\mathrm{N}_x\mathrm{J}_x^+ = \mathrm{J}_x^\top(\mathrm{J}_x\mathrm{J}_x^\top)^{-1}\mathrm{J}_x\mathrm{J}_x^\top(\mathrm{J}_x\mathrm{J}_x^\top)^{-1} = \mathrm{J}_x^\top(\mathrm{J}_x\mathrm{J}_x^\top)^{-1} = \mathrm{J}_x^+.$$

Then for any $v \in \mathbb{R}^n$

$$\begin{aligned} \mathrm{N}_x(D\mathrm{N}_x v) &= \mathrm{N}_x\mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x + \mathrm{N}_x(\mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x)^\top \\ &= \mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x \end{aligned}$$

and similarly

$$\begin{aligned} (D\mathrm{N}_x v)\mathrm{N}_x &= \mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x\mathrm{N}_x + (\mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x)^\top \mathrm{N}_x \\ &= (\mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x)^\top. \end{aligned}$$

It follows that, for any $v \in \mathbb{R}^n$

$$\begin{aligned} \mathrm{N}_x(D\mathrm{N}_x v)\mathrm{N}_x &= \mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x\mathrm{N}_x = 0 \\ \mathrm{N}_x(D\mathrm{N}_x v) + (D\mathrm{N}_x v)\mathrm{N}_x &= \mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x + (\mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x)^\top = D\mathrm{N}_x. \end{aligned}$$

Finally, consider any $w \in \mathbb{R}^n$. Then, since for any matrix A we have $w^\top \mathrm{A} w = w^\top \mathrm{A}^\top w$, we get

$$w^\top (D\mathrm{N}_x v) w = w^\top \mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x w + w^\top (\mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x)^\top w = 2w^\top \mathrm{J}_x^+(\mathrm{H}_x v)\mathrm{T}_x w$$

■

C.8.2 Change in Kinetic Energy

We first prove intermediary results that will be useful. To simplify notation, we shall write $\mathrm{N}_b := \mathrm{N}_{x_b}$, and for $b \in \{1, \ldots, B\}$ we define

$$\Gamma_b = \mathrm{N}_b - 2\mathrm{N}_b\mathrm{N}_{b-1/2} - 2\mathrm{N}_{b-1/2}\mathrm{N}_b + 4\mathrm{N}_{b-1/2}\mathrm{N}_b\mathrm{N}_{b-1/2}.$$

We now show how to approximate Γ_b in terms of N_{b-1} up to order δ^2, by Taylor expanding each of the terms in Γ_b. To Taylor expand we use the following identities

$$x_{b-1/2} = x_{b-1} + \frac{\delta}{2} w_{b-1} \tag{C.20}$$

$$x_b = x_{b-1} + \delta \mathrm{T}_{b-1/2} w_{b-1}. \tag{C.21}$$

First, Taylor expand N_x using identity (C.20)

$$\mathrm{N}_{b-1/2} = \mathrm{N}_{b-1} + \frac{\delta}{2} D\mathrm{N}_{b-1} w_{b-1} + \mathcal{O}(\delta^2), \tag{C.22}$$

and using identity (C.21)

$$\mathrm{N}_b = \mathrm{N}_{b-1} + \delta D\mathrm{N}_{b-1}\mathrm{T}_{b-1/2} w_{b-1} + \mathcal{O}(\delta^2). \tag{C.23}$$

Multiplying (C.22) and (C.23) together and using the fact that N_x is an idempotent matrix for any $x \in \mathbb{R}^n$ we obtain

$$\mathrm{N}_b\mathrm{N}_{b-1/2} = \mathrm{N}_{b-1} + \frac{\delta}{2}\mathrm{N}_{b-1} D\mathrm{N}_{b-1} w_{b-1} + \delta(D\mathrm{N}_{b-1}\mathrm{T}_{b-1/2} w_{b-1})\mathrm{N}_{b-1} + \mathcal{O}(\delta^2) \tag{C.24}$$

$$\mathrm{N}_{b-1/2}\mathrm{N}_b = \mathrm{N}_{b-1} + \delta\mathrm{N}_{b-1}(D\mathrm{N}_{b-1}\mathrm{T}_{b-1/2} w_{b-1}) + \frac{\delta}{2} D\mathrm{N}_{b-1} w_{b-1}\mathrm{N}_{b-1} + \mathcal{O}(\delta^2). \tag{C.25}$$

Next, we multiply (C.22) and (C.24) to get

$$\mathrm{N}_{b-1/2}\mathrm{N}_b\mathrm{N}_{b-1/2} = \mathrm{N}_{b-1} + \frac{\delta}{2} D\mathrm{N}_{b-1} w_{b-1} + \mathcal{O}(\delta^2).$$

Therefore the term Γ_b can be approximated as follows

$$\begin{aligned}
\Gamma_b =& \mathrm{N}_{b-1} + \delta D\mathrm{N}_{b-1}\mathrm{T}_{b-1/2} w_{b-1} \\
& - 2\mathrm{N}_{b-1} - \delta\mathrm{N}_{b-1} D\mathrm{N}_{b-1} w_{b-1} - 2\delta(D\mathrm{N}_{b-1}\mathrm{T}_{b-1/2} w_{b-1})\mathrm{N}_{b-1} \\
& - 2\mathrm{N}_{b-1} - 2\delta\mathrm{N}_{b-1}(D\mathrm{N}_{b-1}\mathrm{T}_{b-1/2} w_{b-1}) - \delta(D\mathrm{N}_{b-1} w_{b-1})\mathrm{N}_{b-1} \\
& + 4\mathrm{N}_{b-1} + 2\delta D\mathrm{N}_{b-1} w_{b-1} + \mathcal{O}(\delta^2) \\
=& \mathrm{N}_{b-1} + \delta D\mathrm{N}_{b-1} w_{b-1} - \delta D\mathrm{N}_{b-1}\mathrm{T}_{b-1/2} w_{b-1} + \mathcal{O}(\delta^2) \\
=& \mathrm{N}_{b-1} + \delta D\mathrm{N}_{b-1}\mathrm{N}_{b-1/2} w_{b-1} + \mathcal{O}(\delta^2),
\end{aligned}$$

where we have used the properties in the Corollary and in the last line we have invoked the linearity of the third-order tensor with respect to its inputs and so

$$\delta D\mathrm{N}_{b-1}[(\mathrm{I} - \mathrm{T}_{b-1/2}) w_{b-1}] = \delta D\mathrm{N}_{b-1}\mathrm{N}_{b-1/2} w_{b-1}.$$

Finally, notice that one can recursively use this approximation to obtain the following

$$\begin{aligned}
w_B^\top \mathrm{N}_B w_B &= w_{B-1}^\top \Gamma_B w_{B-1} \\
&= w_{B-1}^\top \mathrm{N}_{B-1} w_{B-1} + \delta w_{B-1}^\top D\mathrm{N}_{B-1}(\mathrm{N}_{B-1/2} w_{B-1}) w_{B-1} + \mathcal{O}(\delta^2) \\
&= w_0^\top \mathrm{N}_0 w_0 + \delta \sum_{b=0}^{B-1} w_b^\top (D\mathrm{N}_b \mathrm{N}_{b+1/2} w_b) w_b + \mathcal{O}(B\delta^2) \\
&= w_0^\top \mathrm{N}_0 w_0 + \mathcal{O}(\delta).
\end{aligned} \tag{C.26}$$

In the last line we have used the fact that $\mathcal{O}(B\delta^2) = \mathcal{O}(\delta)$. Next, we aim to expand the difference between the squared norms of the velocities at the beginning and end of the trajectory $\Delta_v =$

$\|v_B\|^2 - \|v_0\|^2$. Before we do that, recall that the final velocity v_B in the acceptance ratio was obtained by unsqueezing the velocity w_B at the end of the trajectory

$$v_B = \left(\mathrm{I} + \frac{\alpha}{1-\alpha}\mathrm{N}_B\right) w_B, \tag{C.27}$$

where we use the symbol v to denote an "unsqueezed" velocity (therefore either v_0, before squeezing, or v_B, after unsqueezing), and the symbol w for a "squeezed" velocity in the middle of the algorithm.

$$\begin{aligned}
\Delta_v &= \|v_B\|^2 - \|v_0\|^2 \\
&= \left\| \left(\mathrm{I} + \frac{\alpha}{1-\alpha}\mathrm{N}_B\right) w_B \right\|^2 - \|v_0\|^2 \\
&= w_B^\top \left(\mathrm{I} + \frac{\alpha}{1-\alpha}\mathrm{N}_B\right)\left(\mathrm{I} + \frac{\alpha}{1-\alpha}\mathrm{N}_B\right) w_B - \|v_0\|^2 \\
&= w_B^\top \left(\mathrm{I} + \frac{2\alpha}{1-\alpha}\mathrm{N}_B + \frac{\alpha^2}{(1-\alpha)^2}\mathrm{N}_B\right) w_B - \|v_0\|^2 \\
&= w_B^\top \left(\mathrm{I} + \frac{\alpha(2-\alpha)}{(1-\alpha)^2}\mathrm{N}_B\right) w_B - \|v_0\|^2 \\
&= \|w_0\|^2 + \frac{\alpha(2-\alpha)}{(1-\alpha)^2} w_B^\top \mathrm{N}_B w_B - \|v_0\|^2 \\
&= -\alpha(2-\alpha)\|v_0^\perp\|^2 + \frac{\alpha(2-\alpha)}{(1-\alpha)^2}\left(w_0^\top \mathrm{N}_0 w_0 + \mathcal{O}(\delta)\right) \\
&= -\alpha(2-\alpha)\|v_0^\perp\|^2 + \frac{\alpha(2-\alpha)}{(1-\alpha)^2}\left((1-\alpha)^2\|v_0^\perp\|^2 + \mathcal{O}(\delta)\right) \\
&= \mathcal{O}\left(\delta\frac{\alpha(2-\alpha)}{(1-\alpha)^2}\right).
\end{aligned}$$

In the second line, we have plugged in (C.27), in the fourth line we have expanded the product and used the fact that N_x is idempotent. In the sixth line we have expanded the quadratic form and used the fact that $\|w_0\|^2 = \|w_b\|^2$ for all $b \in \{0,\dots,B\}$ since the bounce is a unitary transformation. In the seventh line we have used the definition of w_0, i.e.

$$w_0 = (\mathrm{I} - \alpha\mathrm{N}_0)v_0,$$

to write down an expression for its norm squared, and we have plugged in (C.26). In the eight line, we have used the fact that N_0 is idempotent and we have used the definition of w_0. Finally, in the last line the first two terms cancel each other out and we are left with the desired result.

C.8.3 Additional Results for the 2D Bayesian Inverse Problem

In Figure 2.11 we have shown the computational cost of THUG, HMC and C-RWM for a fixed integration time of $\tau = 2.0$ achieved with $B = 20$ and $\delta = 0.1$. Here we provide additional plots for different integration times. Figure C.1 shows the minimum bulk ESS for different pairs of B and δ. More precisely, columns correspond to $B = 10, 20, 30$ and rows correspond to $\delta = 0.05, 0.1, 0.5$.

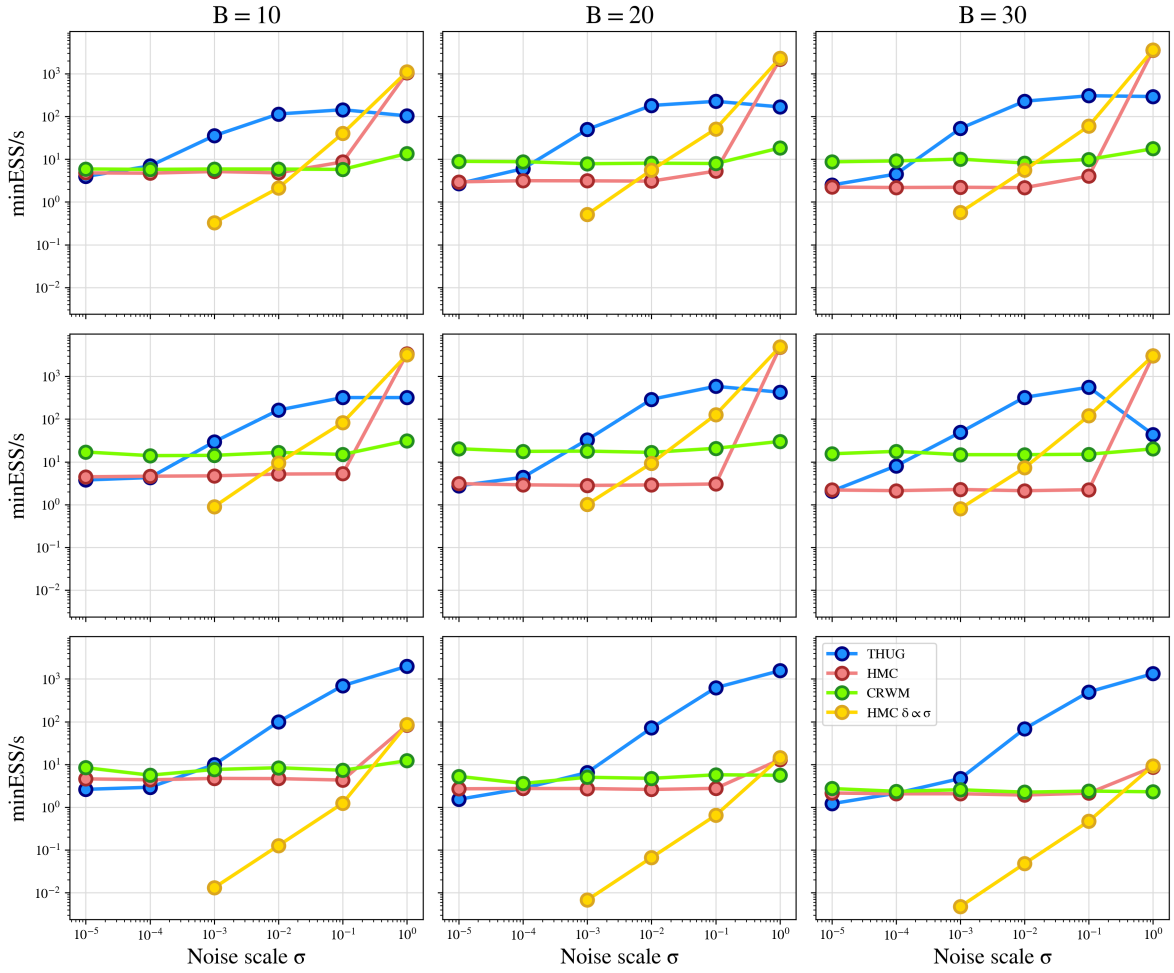

Figure C.1: Minimum bulk ESS for different values of τ.

APPENDIX D

MARKOV SNIPPETS

D.1 Supplementary Material for Integrator Snippets

D.1.1 Equivalence of Folded and Unfolded Integrator Snippets

Consider Algorithm 6, and in particular consider iteration $n = r$ for some $r \in [\![1, P]\!]$. At the beginning of this iteration, the algorithm will start with N particles

$$\left\{\check{Z}_{r-1}^{(t)} : t \in [\![N]\!]\right\},$$

it will use the integrator ψ_{r-1} to construct the trajectories and use weights

$$\left\{W_{r-1,k}(\check{Z}_{r-1}^{(t)}) : k \in [\![0, T]\!], t \in [\![N]\!]\right\}$$

to sample one point from each trajectory and refresh its velocity, thus obtaining particles

$$\left\{\tilde{Z}_r^{(m)} = \left(\tilde{X}_r^{(m)}, \tilde{V}_r^{(m)}\right) : m \in [\![N]\!]\right\}.$$

We wish to find out what is the probability of obtaining the j^{th} particle in the next batch of particles of this form, at the next iteration, i.e. $\tilde{X}_{r+1}^{(j)}$ (we focus on the position component only). This requires two steps. First, given $\tilde{Z}_r^{(m)}$ for $m \in [\![N]\!]$, we need to resample these particles using equation (3.8), and in particular the probability that at the j^{th} resampling draw, we sample particle $\tilde{Z}_r^{(i)}$ is

$$\mathbb{P}(\check{Z}_r^{(j)} = \tilde{Z}_r^{(i)} \mid \tilde{Z}_r^{(m)}, m \in [\![N]\!]) = \frac{1}{Z_{\bar{w}_r}} \cdot \frac{\bar{\mu}_r(\tilde{Z}_r^{(i)})}{\mu_{r-1}(\tilde{Z}_r^{(i)})} \quad \text{where} \quad Z_{\bar{w}_r} := \frac{1}{T+1} \sum_{m=1}^{N} \sum_{\kappa=0}^{T} \frac{\mu_r \circ \psi_r^{\kappa}(\tilde{Z}_r^{(m)})}{\mu_{r-1}(\tilde{Z}_r^{(m)})}.$$

After the particles have been resampled, the iteration index increases to $n = r+1$ and Algorithm 6 starts again from line 3. At this point, for each of the resampled particles, we construct trajectories

using the integrator ψ_r. In particular, for our chosen particle $\check{Z}_r^{(j)}$, the probability of choosing the k^{th} point in its trajectory, i.e. $\check{Z}_{r,k}^{(j)} = \psi_r^k(\check{Z}_r^{(j)})$, given a new velocity $\tilde{V}_{r+1}^{(j)}$, is by definition

$$\mathbb{P}\left(\tilde{X}_{r+1}^{(j)} = \check{X}_{r,k}^{(j)} \,\middle|\, \check{Z}_r^{(j)}, \tilde{V}_{r+1}^{(j)}\right) = \frac{\mu_r \circ \psi_r^k(\check{Z}_r^{(j)})}{\sum_{\ell=0}^{T} \mu_r \circ \psi_r^\ell(\check{Z}_r^{(j)})}.$$

All together, we obtain

$$\begin{aligned}
\mathbb{P}\left(\tilde{X}_{r+1}^{(j)} = \check{X}_{r,k}^{(j)}, \check{Z}_r^{(j)} = \tilde{Z}_r^{(i)} \,\middle|\, \tilde{V}_{r+1}^{(j)}, \tilde{Z}_r^{(m)}, m \in [\![N]\!]\right) &= \frac{1}{Z_{\bar{w}_r}} \cdot \frac{\mu_r \circ \psi_r^k(\check{Z}_r^{(j)})}{\sum_{\ell=0}^{T} \mu_r \circ \psi_r^\ell(\check{Z}_r^{(j)})} \times \frac{\bar{\mu}_r(\tilde{Z}_r^{(i)})}{\mu_{r-1}(\tilde{Z}_r^{(i)})} \\
&= \frac{1}{(T+1)Z_{\bar{w}_r}} \cdot \frac{\mu_r \circ \psi_r^k(\check{Z}_r^{(j)})}{\sum_{\ell=0}^{T} \mu_r \circ \psi_r^\ell(\check{Z}_r^{(j)})} \times \frac{\sum_{\ell=0}^{T} \mu_r \circ \psi_r(\tilde{Z}_r^{(i)})}{\mu_{r-1}(\tilde{Z}_r^{(i)})} \\
&= \frac{1}{\sum_{m=1}^{N} \sum_{\kappa=0}^{T} \frac{\mu_r \circ \psi_r^\kappa(\check{Z}_r^{(m)})}{\mu_{r-1}(\check{Z}_r^{(m)})}} \cdot \frac{\mu_r \circ \psi_r^k(\check{Z}_r^{(j)})}{\mu_{r-1}(\check{Z}_r^{(j)})}.
\end{aligned} \tag{D.1}$$

We now instead consider Algorithm 7 at iteration $n = s$ with particles

$$\left\{Z_{s-1}^{(m)} : m \in [\![N]\!]\right\},$$

which get extended to $N(T+1)$ particles using the integrator ψ_s

$$\left\{Z_{s-1,\kappa}^{(m)} : m \in [\![N]\!], \kappa \in [\![0,T]\!]\right\}.$$

and then get whittled down to a new set of N particles

$$\left\{\bar{Z}_s^{(m)} : m \in [\![N]\!]\right\},$$

where the probability that $\bar{X}_s^{(i)} = X_{s-1,k}^{(j)}$ is given by

$$\begin{aligned}
\mathbb{P}\left(\bar{X}_s^{(i)} = X_{s-1,k}^{(j)} \,\middle|\, Z_{s-1}^{(m)}, m \in [\![N]\!]\right) &= \frac{1}{\sum_{m=1}^{N} \sum_{\kappa=0}^{T} \frac{\mu_s(Z_{s-1,\kappa}^{(m)})}{\mu_{s-1}(Z_{s-1}^{(m)})}} \cdot \frac{\mu_s(Z_{s-1,k}^{(j)})}{\mu_{s-1}(Z_{s-1}^{(j)})} \\
&= \frac{1}{\sum_{m=1}^{N} \sum_{\kappa=0}^{T} \frac{\mu_s \circ \psi_s^\kappa(Z_{s-1}^{(m)})}{\mu_{s-1}(Z_{s-1}^{(m)})}} \cdot \frac{\mu_s \circ \psi_s^k(Z_{s-1}^{(j)})}{\mu_{s-1}(Z_{s-1}^{(j)})}
\end{aligned} \tag{D.2}$$

Equivalence then results from comparing equations (D.1) and (D.2), setting $r = s$, and by choosing the initial particles $Z_{s-1}^{(m)}$ and $\tilde{Z}_r^{(m)}$ to have the same distribution. Importantly, since $\psi_0 = \mathrm{Id}$, notice that for $r = s = 0$ then $\tilde{Z}_1 \sim \bar{\mu}_0 \bar{M}_1 = \mu_0$, and of course $Z_0 \sim \mu_0$, and this concludes the proof.

D.1.1.1 Proof of Equivalence between Folded and Unfolded algorithm

We will provide a proof of the probabilistic equivalence between Algorithm 6 (folded) and Algorithm 7, as mentioned in the main text. We will use bold font to refer to vectors of random variables, meaning that $\tilde{\mathbf{Z}}_n$ is the vector of N random variables obtained at the end of step 8 in Algorithm 6 at iteration n, whereas $\mathbf{Z}_n$ will denote the vector of N random variables obtained at the end of iteration n of Algorithm 7. We will write $\tilde{z}_n^{(i)}$ and $z_n^{(i)}$ for the realizations of their i^{th} random elements respectively.

The equivalence between the two algorithms stems from the fact that the distribution of $\tilde{\mathbf{Z}}_n$ given $\tilde{\mathbf{Z}}_{n-1}$ in the folded algorithm is the same as the distribution of $\mathbf{Z}_n$ given $\mathbf{Z}_{n-1}$ in the unfolded one, together with the fact that the two algorithms initialize the particles from the same distribution and since $\psi_0 = \mathrm{Id}$ and $\bar{\mu}_0 \bar{M}_1 = \mu_0$.

More precisely, to obtain $\tilde{\mathbf{Z}}_n$ from $\tilde{\mathbf{Z}}_{n-1}$ in the folded algorithm we perform the following operations

1. Sample $\mathbf{i} = (i_1, \dots, i_N)$ independently with probabilities $\bar{W}_n(\tilde{z}_{n-1}^{(i_j)})$.
2. Sample $\mathbf{k} = (k_1, \dots, k_N)$ given $\mathbf{i}$ independently (conditionally) with probabilities $W_{n,k}(\tilde{z}_{n-1}^{(i_j)})$
3. Refresh velocities $\tilde{\mathbf{V}}_n \sim \varpi_n^{\otimes N}$.

For this reason, we write their conditional distribution in terms of the marginalization of these auxiliary variables

$$\mathbb{P}(\tilde{\mathbf{Z}}_n \in \mathsf{A} \mid \tilde{\mathbf{Z}}_{n-1}) = \sum_{\mathbf{k},\mathbf{i}} \mathbb{P}(\tilde{\mathbf{Z}} \in \mathsf{A} \mid \tilde{\mathbf{Z}}_{n-1} = \tilde{\mathbf{z}}_{n-1}, \mathbf{K} = \mathbf{k}, \mathbf{I} = \mathbf{i}) \mathbb{P}(\mathbf{K} = \mathbf{k}, \mathbf{I} = \mathbf{i}) \qquad \forall\, \mathsf{A} \in \mathscr{Z}^{\otimes N}, \tag{D.3}$$

where $\mathbf{K}$ and $\mathbf{I}$ are the random vectors with realizations $\mathbf{k}$ and $\mathbf{i}$ respectively, and we don't write the subscript n for ease of notation. From this decomposition, the proof follows immediately, since the unfolded algorithm corresponds to sampling from the joint directly, whereas the folded algorithm corresponds to sampling the particle indices first, and then the trajectory indices conditionally. Mathematically, the folded algorithm corresponds to simply splitting this joint as

$$\begin{aligned}
\mathbb{P}(\mathbf{K} = \mathbf{k}, \mathbf{I} = \mathbf{i}) &= \mathbb{P}(\mathbf{K} = \mathbf{k} \mid \mathbf{I} = \mathbf{i}) \mathbb{P}(\mathbf{I} = \mathbf{i}) \\
\mathbb{P}(\mathbf{K} = \mathbf{k} \mid \mathbf{I} = \mathbf{i}) &\propto \prod_{j=1}^{N} W_{n,k_j}(\tilde{z}_{n-1}^{(i_j)}) \\
\mathbb{P}(\mathbf{I} = \mathbf{i}) &\propto \prod_{j=1}^{N} \bar{W}_n(\tilde{z}_{n-1}^{(i_j)})
\end{aligned}$$

and noticing that the product of these expressions gives us the unfolded weights

$$\mathbb{P}(\mathbf{K} = \mathbf{k}, \mathbf{I} = \mathbf{i}) = \prod_{j=1}^{N} \bar{W}_n(\tilde{z}_{n-1}^{(i_j)}) W_{n,k_j}(\tilde{z}_{n-1}^{(i_j)}) = \prod_{j=1}^{N} \frac{\bar{\mu}_n(\tilde{z}_{n-1}^{(i_j)})}{\mu_{n-1}(\tilde{z}_{n-1}^{(i_j)})} \frac{\mu_n \circ \psi_n^{k_j}(\tilde{z}_{n-1}^{(i_j)})}{(T+1)\bar{\mu}_n(\tilde{z}_{n-1}^{(i_j)})} \propto \prod_{j=1}^{N} \check{W}_{n,k_j}(\tilde{z}_{n-1}^{(i_j)}).$$

Therefore, sampling at once from the joint $\mathbb{P}(\mathbf{K}=\mathbf{k},\mathbf{I}=\mathbf{i})$ using the unfolded weights (which corresponds to using the unfolded algorithm) is equivalent to sampling from the marginal on $\mathbf{i}$ and then $\mathbf{k} \mid \mathbf{i}$ using the folded weights. The other term in Equation (D.3) is simply

$$\mathbb{P}(\tilde{\mathbf{Z}} \in \mathsf{A} \mid \tilde{\mathbf{Z}}_{n-1} = \tilde{\mathbf{z}}_{n-1}, \mathbf{K}=\mathbf{k}, \mathbf{I}=\mathbf{i}) = \int_{\mathsf{A}} \left[\prod_{j=1}^{N} \delta_{\psi_n^{k_j}(\tilde{x}_{n-1}^{(i_j)})}(d\tilde{x}_n^{(i_j)}) \right] \varpi_n^{\otimes N}(d\tilde{\mathbf{v}}_n),$$

i.e. we set the new particle's positions to the points along the snippets that we have selected and refresh the velocities, and we recognise this as being exactly the same mechanism used in the unfolded algorithm.

D.1.2 Existence and sufficient conditions for Integrator Snippets

The aim of this section is to present the conditions required for the existence of the key elements of this chapter. In particular, we show the existence of the backward kernel (3.4) and of the incremental weights (3.5) for the folded algorithm. Notice that the weights (3.3) used in the definition of the forward kernel always exist since

$$\mu_n^{\psi_n^{-k}} \ll \sum_{\ell \in [\![0,T]\!]} \mu_n^{\psi_n^{-\ell}} \qquad \forall\, k \in [\![0,T]\!],\, \forall n \in [\![0,P]\!],$$

and as a consequence the forward kernel is always a valid Markov kernel.

Theorem D.1 (Existence of backward kernel and incremental weights)**.** *Suppose that $\mu_{n-1} \gg \bar{\mu}_n$ and $\mu_{n-1}R_n = \mu_{n-1}$, and for any $n \in [\![P]\!]$,*

1. *$\bar{\mu}_{n-1}\bar{M}_n = \mu_{n-1}$ for all $n \in [\![P]\!]$.*
2. *$\bar{\mu}_{n-1}$-almost surely, $\bar{M}(z,\cdot) \ll \mu_{n-1}$.*
3. *$\bar{L}_{n-1}$ in (3.4) exists.*
4. *If $\bar{\mu}_n \ll \mu_{n-1}$ then $\bar{w}_n$ in (3.5) exists.*

Proof. First, we show that $\mu_{n-1}R_n = \mu_{n-1}$ implies $\bar{\mu}_{n-1}\bar{M}_n = \mu_{n-1}$ for any $n \in [\![P]\!]$

$$\begin{aligned}
\bar{\mu}_{n-1}\bar{M}_n(dz') &= \int \left[\frac{1}{T+1} \sum_{\ell=0}^{T} \mu_{n-1}^{\psi_{n-1}^{-\ell}}(dz) \right] \sum_{k=0}^{T} \frac{\mu_{n-1}^{\psi_{n-1}^{-k}}(dz)}{\sum_{\ell \in [\![0,T]\!]} \mu_{n-1}^{\psi_{n-1}^{-\ell}}(dz)} R_n(\psi_{n-1}^k(z), dz') \\
&= \frac{1}{T+1} \sum_{k=0}^{T} \int \mu_{n-1}^{\psi_{n-1}^{-k}}(dz) R_n(\psi_{n-1}^k(z), dz') \\
&= \frac{1}{T+1} \sum_{k=0}^{T} \int \mu_{n-1}(dz) R_n(z, dz') \\
&= \mu_{n-1},
\end{aligned}$$

where in the second line we have performed a change of variables. For the second statement, let $\mathsf{A} \in \mathcal{Z}$ be any measurable set. We wish to show that when $\mu_{n-1}(\mathsf{A}) = 0$ then $\bar{\mu}_{n-1}$-almost surely $\bar{M}_n(z, \mathsf{A}) = 0$. The first statement tells us that for any $\mathsf{A} \in \mathcal{Z}$ such that $\mu_{n-1}(\mathsf{A}) = 0$ then $\bar{\mu}_{n-1}\bar{M}_n(\mathsf{A}) = 0$, or written explicitly

$$\int_{\mathsf{Z}} \bar{M}_n(z, \mathsf{A}) \bar{\mu}_{n-1}(dz) = 0.$$

Since $\bar{M}_n(\cdot, \mathsf{A})$ is a non-negative measurable function, it is a standard result (see, for instance Proposition 4.13 part c in [35]) that this implies $\bar{M}_n(\cdot, \mathsf{A}) = 0$ $\bar{\mu}_{n-1}$-almost everywhere. We now turn to showing the existence of the backward kernel $\bar{L}_{n-1}$. First of all, notice that it is defined as the Radon-Nikodym derivative of the joint and the marginal

$$\bar{L}_{n-1}(z', dz) = \frac{\bar{\mu}_{n-1}(dz)\bar{M}_n(z, \cdot)}{\bar{\mu}_{n-1}\bar{M}_n(\cdot)}(z').$$

As such, it is a function in the first argument and a probability measure in the second. One might think that the dominated measure is the random measure corresponding to $\bar{M}_n$, but this is not true, and one has to consider the full joint distribution. One way to think about this, is as a generalised Bayes theorem and we refer the reader to our discussion in Appendix A.1, specifically Theorem A.2 and the comments following it. To show that this Radon-Nikodym exists and is well-defined, we need to show that $\bar{\mu}_{n-1}(dz)\bar{M}_n(z, dz') \ll \bar{\mu}_{n-1}\bar{M}_n(dz')$. This is immediate from Theorem A.2 as long as $(\mathsf{Z}, \mathcal{Z})$ is a nice measurable space, however we provide a simple argument here. Let $\mathsf{A} \in \mathcal{Z}$ be such that $\bar{\mu}_{n-1}\bar{M}_n(\mathsf{A}) = 0$. Then for any $\mathsf{B} \in \mathcal{Z}$ we have

$$\bar{\mu}_{n-1} \otimes \bar{M}_n(\mathsf{B} \times \mathsf{A}) = \int_{\mathsf{B}} \int_{\mathsf{A}} \bar{\mu}_{n-1}(dz)\bar{M}_n(z, dz') = \int_{\mathsf{B}} \bar{\mu}_{n-1}\bar{M}_n(\mathsf{A}) = 0,$$

which ends the proof. Finally, the third statement follows from $\bar{\mu}_n \ll \mu_{n-1}$ and standard SMC arguments, specifically the two-step Feynman-Kac formulation and the choice of the near-optimal backward kernel (1.26). ■

While this lemma deals exclusively with the folded algorithm, for the unfolded one the same assumptions are required since the resampling weights in (3.8) require $\mu_n^{\psi_n^{-k}} \ll \mu_{n-1}$ for every $k \in [\![0, T]\!]$, which is equivalent to $\bar{\mu}_n \ll \mu_{n-1}$. To prove the first direction of the implication, suppose $\mu_n^{\psi_n^{-k}} \ll \mu_{n-1}$ is true for every $k \in [\![0, T]\!]$, and let $\mathsf{A} \in \mathcal{Z}$ be such that $\mu_{n-1}(\mathsf{A}) = 0$ and therefore $\mu_n^{\psi_n^{-k}}(\mathsf{A}) = 0$. We wish to show that this implies that $\bar{\mu}_n(\mathsf{A}) = 0$, but this follows directly from its definition

$$\forall k \in [\![0, T]\!] \quad \mu_n^{\psi_n^{-k}}(\mathsf{A}) = 0 \implies \frac{1}{T+1} \sum_{k=0}^{T} \mu_n^{\psi_n^{-k}}(\mathsf{A}) = 0.$$

Similarly, suppose that $\bar{\mu}_n \ll \mu_{n-1}$ and that $\mathsf{A} \in \mathcal{Z}$ is such that μ_{n-1}. We wish to show that this implies $\mu_n^{\psi_n^{-k}} \ll \mu_{n-1}$ for every $k \in [\![0, T]\!]$. Once more, this follows from the fact that each $\mu_n^{\psi_n^{-k}}$ is a probability measure and hence non-negative, so if the average below is zero

$$\frac{1}{T+1} \sum_{k=0}^{T} \mu_n^{\psi_n^{-k}}(\mathsf{A}) = 0,$$

it implies $\mu_n^{\psi_n^{-k}}(\mathsf{A}) = 0$ for every $k \in [\![0, T]\!]$.

D.1.3 Effective Sample Size for a Mixture

D.1.3.1 Exact formula of ESS for a mixture

The integrator snippet estimator is

$$\hat{h} = \frac{1}{T+1}\sum_{k=0}^{T}\frac{\sum_{i=1}^{N} h(\psi^k(X_i))w_k(X_i)}{\sum_{i=1}^{N} w_k(X_i)} \qquad w_k(X_i) = \frac{\mu\circ\psi^k(X_i)}{\eta(X_i)} \qquad X_i \sim \eta.$$

Notice the weights above are not normalized. We now define the random variables

$$\begin{aligned} W_k &= \frac{\mu(Y)}{\eta(\psi^{-k}(Y))} && Y \sim \eta^{\psi^k} \\ Z_k &= h(Y)\frac{\mu(Y)}{\eta(\psi^{-k}(Y))} && Y \sim \eta^{\psi^k}. \end{aligned}$$

Given a sample $Y_i \sim \eta^{\psi^k}$ with $i = 1,\ldots,N$ we write $Z_{k,i}$ and $W_{k,i}$ for the random variables resulting from feeding Y_i as above. Their sample averages are

$$\begin{aligned} \bar{Z}_k &= \frac{1}{N}\sum_{i=1}^{N} Z_{k,i} \\ \bar{W}_k &= \frac{1}{N}\sum_{i=1}^{N} W_{k,i}. \end{aligned}$$

The random variable corresponding to the estimator above is then

$$\hat{H} = \frac{1}{T+1}\sum_{k=0}^{T}\frac{\bar{Z}_k}{\bar{W}_k}$$

To make things simpler, we also identify the ratios as random variables. We define $\hat{H}_k$ as the random variable corresponding to a standard importance sampling estimate We now define

$$\hat{H}_k = \frac{\bar{Z}_k}{\bar{W}_k} = \frac{\sum_{i=1}^{N} h(\psi^k(X_i))\frac{\mu(\psi^k(X_i))}{\eta(X_i)}}{\sum_{i=1}^{N}\frac{\mu(\psi^k(X_i))}{\eta(X_i)}}.$$

Therefore we can write the integrator snippet estimator as an average of these estimators

$$\hat{H} = \frac{1}{T+1}\sum_{k=0}^{T}\hat{H}_k.$$

Using the formula for the variance of a linear combination of random variables we get

$$\mathbb{V}[\hat{H}] = \frac{1}{(T+1)^2}\left[\sum_{k=0}^{T}\mathbb{V}[\hat{H}_k] + 2\sum_{k=0}^{T}\sum_{\ell>k}\mathrm{Cov}(\hat{H}_k,\hat{H}_\ell)\right]$$

From [80] we know that

$$\mathbb{V}[\hat{H}_k] \approx \mathbb{V}_\mu[\hat{H}_{\text{MC}}](\mathbb{V}[W_k]+1),$$

where $\mathbb{V}_\mu[\hat{H}_{\text{MC}}]$ denotes the Ordinary Monte Carlo variance for a set of N samples

$$\mathbb{V}_\mu[\hat{H}_{\text{MC}}] = \mathbb{V}_\mu\left[\frac{1}{N}\sum_{i=1}^{N} h(X_i)\right] = \frac{\mathbb{V}_\mu[H]}{N} \qquad H = h(X) \qquad X \sim \mu.$$

Dividing $\mathbb{V}[\hat{H}]$ by the Ordinary Monte Carlo variance we obtain

$$\frac{\mathbb{V}[\hat{H}]}{\mathbb{V}_\mu[\hat{H}_{\text{MC}}]} = \frac{1}{(T+1)^2}\left[\sum_{k=0}^{T}(\mathbb{V}[W_k]+1) + 2\sum_{k=0}^{T}\sum_{\ell>k}\frac{\text{Cov}(\hat{H}_k,\hat{H}_\ell)}{\mathbb{V}_\mu[\hat{H}_{\text{MC}}]}\right]$$

Again, following [80] we can estimate $\mathbb{V}[W_k]$ as follows

$$\begin{aligned}
\mathbb{V}[W_k] &= \mathbb{E}[W_k^2] - \mathbb{E}[W_k]^2 \\
&= \int\left(\frac{\mu(y)}{\eta^{\psi^k}(y)}\right)^2 \eta^{\psi^k}(dy) - \left(\int\frac{\mu(y)}{\eta^{\psi^k}(y)}\eta^{\psi^k}(dy)\right)^2 \\
&= \left(\frac{\mathcal{Z}_{\eta^{\psi^k}}}{\mathcal{Z}_\mu}\right)^2 \int\left(\frac{\widetilde{\mu}(y)}{\widetilde{\eta^{\psi^k}}(y)}\right)^2 \eta^{\psi^k}(dy) - 1 \\
&\approx \frac{\frac{1}{N}\sum_{i=1}^{N} w_k(X_i)^2}{\left(\frac{1}{N}\sum_{i=1}^{N} w_k(X_i)\right)^2} - 1 \qquad\qquad X_i \sim \eta \\
&\approx N\frac{\sum_{i=1}^{N} w_k(X_i)^2}{\left(\sum_{i=1}^{N} w_k(X_i)\right)^2} - 1.
\end{aligned}$$

Therefore the ratio of variances becomes

$$\frac{\mathbb{V}[\hat{H}]}{\mathbb{V}_\mu[\hat{H}_{\text{MC}}]} = \frac{1}{(T+1)^2}\left[N\sum_{k=0}^{T}\frac{\sum_{i=1}^{N} w_k(X_i)^2}{\left(\sum_{i=1}^{N} w_k(X_i)\right)^2} + 2\sum_{k=0}^{T}\sum_{\ell>k}\frac{\text{Cov}(\hat{H}_k,\hat{H}_\ell)}{\mathbb{V}_\mu[\hat{H}_{\text{MC}}]}\right]$$

D.1.3.2 Approximating the Covariance terms with the Delta method

The covariance can be written as

$$\text{Cov}(\hat{H}_k,\hat{H}_\ell) = \mathbb{E}\left[\left(\frac{\bar{Z}_k}{\bar{W}_k} - \mathbb{E}\left[\frac{\bar{Z}_k}{\bar{W}_k}\right]\right)\left(\frac{\bar{Z}_\ell}{\bar{W}_\ell} - \mathbb{E}\left[\frac{\bar{Z}_\ell}{\bar{W}_\ell}\right]\right)\right].$$

Define the function corresponding the the argument of the expectation

$$g(\bar{W}_k,\bar{W}_\ell,\bar{Z}_k,\bar{Z}_k) = \frac{\bar{Z}_k\bar{Z}_\ell}{\bar{W}_k\bar{W}_\ell} - \frac{\bar{Z}_k}{\bar{W}_k}\mathbb{E}\left[\frac{\bar{Z}_\ell}{\bar{W}_\ell}\right] - \frac{\bar{Z}_\ell}{\bar{W}_\ell}\mathbb{E}\left[\frac{\bar{Z}_k}{\bar{W}_k}\right] + \mathbb{E}\left[\frac{\bar{Z}_k}{\bar{W}_k}\right]\mathbb{E}\left[\frac{\bar{Z}_\ell}{\bar{W}_\ell}\right].$$

Partial derivatives are

$$\frac{\partial g}{\partial \bar{W}_k} = -\frac{\bar{Z}_k \bar{Z}_\ell}{\bar{W}_k^2 \bar{W}_\ell} + \frac{\bar{Z}_k}{\bar{W}_k^2} \mathbb{E}\left[\frac{\bar{Z}_\ell}{\bar{W}_\ell}\right]$$
$$\frac{\partial g}{\partial \bar{W}_\ell} = -\frac{\bar{Z}_k \bar{Z}_\ell}{\bar{W}_k \bar{W}_\ell^2} + \frac{\bar{Z}_\ell}{\bar{W}_\ell^2} \mathbb{E}\left[\frac{\bar{Z}_k}{\bar{W}_k}\right]$$
$$\frac{\partial g}{\partial \bar{Z}_k} = \frac{\bar{Z}_\ell}{\bar{W}_k \bar{W}_\ell} - \frac{1}{\bar{W}_k} \mathbb{E}\left[\frac{\bar{Z}_\ell}{\bar{W}_\ell}\right]$$
$$\frac{\partial g}{\partial \bar{Z}_\ell} = \frac{\bar{Z}_k}{\bar{W}_k \bar{W}_\ell} - \frac{1}{\bar{W}_\ell} \mathbb{E}\left[\frac{\bar{Z}_k}{\bar{W}_k}\right]$$

The expectation of each of the four variables is

$$\mathbb{E}[\bar{W}_k] = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} \frac{\mu(Y_i)}{\eta^{\psi^k}(Y_i)}\right] = \mathbb{E}\left[\frac{\mu(Y)}{\eta^{\psi^k}(Y)}\right] = 1 \qquad \text{indep}$$
$$\mathbb{E}[\bar{W}_\ell] = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} \frac{\mu(Y_i)}{\eta^{\psi^\ell}(Y_i)}\right] = \mathbb{E}\left[\frac{\mu(Y)}{\eta^{\psi^\ell}(Y)}\right] = 1 \qquad \text{indep}$$
$$\mathbb{E}[\bar{Z}_k] = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} h(Y_i)\frac{\mu(Y_i)}{\eta^{\psi^k}(Y_i)}\right] = \mathbb{E}\left[h(Y)\frac{\mu(Y)}{\eta^{\psi^k}(Y)}\right] = \mathbb{E}[Z_k] = \mu(h)$$
$$\mathbb{E}[\bar{Z}_\ell] = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} h(Y_i)\frac{\mu(Y_i)}{\eta^{\psi^\ell}(Y_i)}\right] = \mathbb{E}\left[h(Y)\frac{\mu(Y)}{\eta^{\psi^\ell}(Y)}\right] = \mathbb{E}[Z_\ell] = \mu(h)$$

The variance of the sample mean weights is

$$\mathbb{V}[\bar{W}_k] = \mathbb{V}\left[\frac{1}{N}\sum_{i=1}^{N} W_{k,i}\right] = \frac{\mathbb{V}[W_k]}{N} \qquad \text{indep.}$$

The covariance between $\bar{Z}_k$ and $\bar{W}_k$ is

$$\begin{aligned}
\mathrm{Cov}(\bar{Z}_k, \bar{W}_k) &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{E}\left[h(\psi^k(Y_i))\frac{\mu(\psi^k(Y_i))}{\eta(Y_i)}\frac{\mu(\psi^\ell(Y_j))}{\eta(Y_j)}\right] - \mathbb{E}[Z_k]\mathbb{E}[W_k] \\
&= \frac{N-1}{N}\mathbb{E}[Z_k]\mathbb{E}[W_k] + \frac{1}{N}\mathbb{E}[Z_k W_k] - \mathbb{E}[Z_k]\mathbb{E}[W_k] \\
&= \frac{1}{N}\mathbb{E}[Z_k W_k] - \frac{1}{N}\mu(h)
\end{aligned}$$

The expected value of the ratios become

$$\mathbb{E}\left[\frac{\bar{Z}_k}{\bar{W}_k}\right] = \mu(h) + \frac{\mu(h)}{N}\mathbb{E}[W_k^2] - \frac{\mathbb{E}[Z_k W_k]}{N}$$
$$\mathbb{E}\left[\frac{\bar{Z}_\ell}{\bar{W}_\ell}\right] = \mu(h) + \frac{\mu(h)}{N}\mathbb{E}[W_\ell^2] - \frac{\mathbb{E}[Z_\ell W_\ell]}{N}$$

The second-order partial derivatives evaluated at θ then become

$$\frac{\partial^2 g(\theta)}{\partial \bar{W}_k^2} = -2\frac{\mu(h)^2}{N}\mathbb{E}[W_\ell^2] + 2\frac{\mu(h)}{N}\mathbb{E}[Z_\ell W_\ell]$$

$$
\begin{aligned}
\frac{\partial^2 g(\theta)}{\partial \bar{W}_\ell^2} &= -2\frac{\mu(h)^2}{N}\mathbb{E}[W_k^2] + 2\frac{\mu(h)}{N}\mathbb{E}[Z_k W_k] \\
\frac{\partial^2 g(\theta)}{\partial \bar{Z}_k^2} &= 0 \\
\frac{\partial^2 g(\theta)}{\partial \bar{Z}_\ell^2} &= 0 \\
\frac{\partial^2 g(\theta)}{\partial \bar{W}_k \partial \bar{W}_\ell} &= \mu(h)^2 \\
\frac{\partial^2 g(\theta)}{\partial \bar{W}_k \partial \bar{Z}_k} &= \frac{\mu(h)}{N}\mathbb{E}[W_\ell^2] - \frac{\mathbb{E}[Z_\ell W_\ell]}{N} \\
\frac{\partial^2 g(\theta)}{\partial \bar{W}_k \partial \bar{Z}_\ell} &= -\mu(h) \\
\frac{\partial^2 g(\theta)}{\partial \bar{W}_\ell \bar{Z}_k} &= -\mu(h) \\
\frac{\partial^2 g(\theta)}{\partial \bar{W}_\ell \partial \bar{Z}_\ell} &= \frac{\mu(h)}{N}\mathbb{E}[W_k^2] - \frac{\mathbb{E}[Z_k W_k]}{N} \\
\frac{\partial^2 g(\theta)}{\partial \bar{Z}_k \partial \bar{Z}_\ell} &= 1.
\end{aligned}
$$

The function evaluated at θ becomes

$$
\begin{aligned}
g(\theta) = &-\frac{\mu(h)^2}{N}\mathbb{E}[W_\ell^2] + \frac{\mu(h)}{N}\mathbb{E}[Z_\ell W_\ell] - \mu(h)^2 - \frac{\mu(h)^2}{N}\mathbb{E}[W_k^2] + \frac{\mu(h)}{N}\mathbb{E}[Z_k W_k] \\
&+ \mu(h)^2 + \frac{\mu(h)^2}{N}\mathbb{E}[W_\ell^2] - \frac{\mu(h)}{N}\mathbb{E}[Z_\ell W_\ell] + \frac{\mu(h)^2}{N}\mathbb{E}[W_k^2] + \frac{\mu(h)^2}{N^2}\mathbb{E}[W_k^2]\mathbb{E}[W_\ell^2] - \frac{\mu(h)}{N^2}\mathbb{E}[W_k^2]\mathbb{E}[Z_\ell W_\ell] \\
&- \frac{\mu(h)}{N}\mathbb{E}[Z_k W_k] - \frac{\mu(h)}{N^2}\mathbb{E}[Z_k W_k]\mathbb{E}[W_\ell^2] + \frac{\mathbb{E}[Z_k W_k]\mathbb{E}[Z_\ell W_\ell]}{N^2} \\
= &\frac{\mu(h)^2}{N^2}\mathbb{E}[W_k^2]\mathbb{E}[W_\ell^2] - \frac{\mu(h)}{N^2}\mathbb{E}[W_k^2]\mathbb{E}[Z_\ell W_\ell] - \frac{\mu(h)}{N^2}\mathbb{E}[Z_k W_k]\mathbb{E}[W_\ell^2] + \frac{\mathbb{E}[Z_k W_k]\mathbb{E}[Z_\ell W_\ell]}{N^2} \\
= &\frac{\left(\mu(h)\mathbb{E}[W_\ell^2] - \mathbb{E}[Z_\ell W_\ell]\right)\left(\mu(h)\mathbb{E}[W_k^2] - \mathbb{E}[Z_k W_k]\right)}{N^2}.
\end{aligned}
$$

The Taylor expansion then becomes

$$
\begin{aligned}
\mathrm{Cov}(\hat{H}_k, \hat{H}_\ell) \approx &\frac{\mu(h)^2}{N^2}\mathbb{E}[W_k^2]\mathbb{E}[W_\ell^2] - \frac{\mu(h)}{N^2}\mathbb{E}[W_k^2]\mathbb{E}[Z_\ell W_\ell] - \frac{\mu(h)}{N^2}\mathbb{E}[Z_k W_k]\mathbb{E}[W_\ell^2] + \frac{\mathbb{E}[Z_k W_k]\mathbb{E}[Z_\ell W_\ell]}{N^2} \\
&- \frac{\mu(h)^2}{N^2}\mathbb{E}[W_k^2]\mathbb{E}[W_\ell^2] + \frac{\mu(h)}{N^2}\mathbb{E}[W_k^2]\mathbb{E}[Z_\ell W_\ell] + \frac{\mu(h)^2}{N^2}\mathbb{E}[W_\ell^2] - \frac{\mu(h)}{N^2}\mathbb{E}[Z_\ell W_\ell] \\
&- \frac{\mu(h)^2}{N^2}\mathbb{E}[W_\ell^2]\mathbb{E}[W_k^2] + \frac{\mu(h)}{N^2}\mathbb{E}[W_\ell^2]\mathbb{E}[Z_k W_k] + \frac{\mu(h)^2}{N^2}\mathbb{E}[W_k^2] - \frac{\mu(h)}{N^2}\mathbb{E}[Z_k W_k] \\
&+ \frac{\mu(h)^2}{N}\mathbb{E}[W_k W_\ell] - \frac{\mu(h)^2}{N} + \frac{\mu(h)}{N^2}\mathbb{E}[Z_k W_k]\mathbb{E}[W_\ell^2] - \frac{\mathbb{E}[Z_k W_k]\mathbb{E}[Z_\ell W_\ell]}{N^2} - \frac{\mu(h)^2}{N^2}\mathbb{E}[W_\ell^2] \\
&+ \frac{\mu(h)}{N^2}\mathbb{E}[Z_\ell W_\ell] - \frac{\mu(h)}{N}\mathbb{E}[Z_\ell W_k] + \frac{\mu(h)^2}{N} - \frac{\mu(h)}{N}\mathbb{E}[Z_k W_\ell] + \frac{\mu(h)^2}{N} + \frac{\mu(h)}{N^2}\mathbb{E}[Z_\ell W_\ell]\mathbb{E}[W_k^2] \\
&- \frac{\mathbb{E}[Z_\ell W_\ell]\mathbb{E}[Z_k W_k]}{N^2} - \frac{\mu(h)^2}{N^2}\mathbb{E}[W_k^2] + \frac{\mu(h)}{N^2}\mathbb{E}[Z_k W_k] + \frac{\mathbb{E}[Z_k Z_\ell]}{N} - \frac{\mu(h)^2}{N}.
\end{aligned}
$$

Simplifying this we get

$$\begin{aligned}\mathrm{Cov}(\hat{H}_k,\hat{H}_\ell) \approx &-\frac{\mu(h)^2}{N^2}\mathbb{E}[W_\ell^2]\mathbb{E}[W_k^2]+\frac{\mu(h)^2}{N}\mathbb{E}[W_kW_\ell]+\frac{\mu(h)}{N^2}\mathbb{E}[Z_kW_k]\mathbb{E}[W_\ell^2]-\frac{\mu(h)}{N}\mathbb{E}[Z_\ell W_k]\\ &-\frac{\mu(h)}{N}\mathbb{E}[Z_kW_\ell]+\frac{\mu(h)}{N^2}\mathbb{E}[Z_\ell W_\ell]\mathbb{E}[W_k^2]-\frac{\mathbb{E}[Z_\ell W_\ell]\mathbb{E}[Z_kW_k]}{N^2}+\frac{\mathbb{E}[Z_kZ_\ell]}{N}\end{aligned}$$

Notice that the $1/N$ terms can be collected together to form

$$\frac{\mathbb{E}\left[\left(Z_k-\mu(h)W_k\right)\left(Z_\ell-\mu(h)W_\ell\right)\right]}{N}$$

and the $1/N/^2$ terms can be collected together to form

$$-\frac{\left(\mathbb{E}[Z_kW_k]-\mu(h)\mathbb{E}[W_k^2]\right)\left(\mathbb{E}[Z_\ell W_\ell]-\mu(h)\mathbb{E}[W_\ell^2]\right)}{N^2},$$

therefore we can write the covariance terms as

$$\mathrm{Cov}(\hat{H}_k,\hat{H}_\ell) \approx -\frac{\left(\mathbb{E}[Z_kW_k]-\mu(h)\mathbb{E}[W_k^2]\right)\left(\mathbb{E}[Z_\ell W_\ell]-\mu(h)\mathbb{E}[W_\ell^2]\right)}{N^2}+\frac{\mathbb{E}\left[\left(Z_k-\mu(h)W_k\right)\left(Z_\ell-\mu(h)W_\ell\right)\right]}{N}.$$

D.1.4 Equivalent Resampling methods for Integrator Snippets

In this section we show that, as intuition would suggest, the order in which we resample the trajectory or particle indices in Algorithm 7 does not matter, in terms of correctness. In other words, we can either perform a stratification by sampling the particle indices first and then the trajectory indices afterwards (or vice versa) or we can simultaneously sample pairs of indices. We consider 3 methods for sampling.

1. Sample trajectory indices, then sample particle indices.
2. Sample particle indices, then sample trajectory indices.
3. Flatten array and sample a single index, then transform into a pair of trajectory and particle indices.

D.1.4.1 Strategy One: Trajectory, then Particle

We have the following weights available:

$$\breve{W}_{n,k}^{(i)} = \frac{\breve{w}_{n,k}^{(i)}}{\sum_{j=1}^{N}\sum_{\ell=0}^{T}\breve{w}_{n,\ell}^{(j)}} \qquad \text{where} \qquad \breve{w}_{n,k}^{(i)} := \frac{\mu_n(Z_{n-1,k}^{(i)})}{\mu_{n-1}(Z_{n-1}^{(i)})}$$

I want to find the probability that I will choose particle

$$Z_{n-1,\kappa}^{(t)}$$

for a specific $\kappa \in [\![0, T]\!]$ and a specific $t \in [\![N]\!]$ by first sampling the trajectory index, and then sampling the particle index. The probability of choosing a particle with $k = \kappa$ is

$$\mathbb{P}(k = \kappa) = \sum_{i=1}^{N} \breve{W}_{n,\kappa}^{(i)}.$$

Now, given that we have sampled $k = \kappa$, now we want to sample $i = t$. Notice we are doing this conditionally. In theory, this should be

$$\mathbb{P}(i = t \mid k = \kappa) = \frac{\breve{W}_{n,\kappa}^{(t)}}{\sum_{i=1}^{N} \breve{W}_{n,\kappa}^{(i)}}$$

Overall, the probability that we choose the pair (i, k) is

$$\begin{aligned}
\mathbb{P}(i = t, k = \kappa) &= \mathbb{P}(i = t \mid k = \kappa)\mathbb{P}(k = \kappa) \\
&= \frac{\breve{W}_{n,\kappa}^{(t)}}{\sum_{i=1}^{N} \breve{W}_{n,\kappa}^{(i)}} \times \sum_{i=1}^{N} \breve{W}_{n,\kappa}^{(i)} \\
&= \breve{W}_{n,\kappa}^{(t)}.
\end{aligned}$$

which makes sense.

D.1.4.2 Strategy Two: Particle, then Trajectory

The probability that we choose a certain particle is

$$\mathbb{P}(i = t) = \sum_{k=0}^{T} \breve{W}_{n,k}^{(t)}$$

and the probability that we choose a certain index κ within this trajectory is

$$\mathbb{P}(k = \kappa \mid i = t) = \frac{\breve{W}_{n,\kappa}^{(t)}}{\sum_{k=0}^{T} \breve{W}_{n,k}^{(t)}}$$

Therefore overall the probability that we choose both is

$$\mathbb{P}(i = t, k = \kappa) = \breve{W}_{n,\kappa}^{(t)},$$

as expected.

D.1.4.3 Strategy Three: Both at the same time

In this case, we flatten the particles. To do this, we simply create a new variable

$$\xi_{n-1}^{(m)} = Z_{n-1,m \mod T}^{(\lfloor m/T \rfloor + 1)}$$

and then we just sample these particles with the flattened weights

$$\Lambda_n^{(m)} = \check{W}_{n,m \mod T}^{(\lfloor m/T \rfloor + 1)}.$$

Automatically, the probability of choosing any given particle is exactly the weight corresponding to that particle.

D.1.5 Implementation details for common integrators

D.1.5.1 Deterministic Walk Integrator

We now describe this integrator to show the flexibility of this framework. This integrator requires B auxiliary velocity variables (or perturbations) and an index $i \in [\![0, B-1]\!]$ which determines which of these perturbations is to be used. Therefore, we introduce an auxiliary measurable space $(\mathbb{R}^{2B} \times [\![0, B-1]\!], \mathscr{B}(\mathbb{R}^{2B}) \times \mathscr{P}([\![0, B-1]\!]))$ and the DW integrator is the function

$$\psi_\delta : \mathbb{R}^{2(B+1)} \times [\![0, B-1]\!] \to \mathbb{R}^{2(B+1)} \times [\![0, B-1]\!],$$

which performs the following operation

$$\psi_\delta(x, v_0, \ldots, v_{B-1}, i) = (x + \delta v_i, v_0, \ldots, v_{B-1}, i + 1 \bmod B).$$

Naturally, the composition ψ^B constructs a path using all the perturbations

$$x \longmapsto x + \delta \sum_{i=0}^{B-1} v_i.$$

Notice that the function ψ_δ is invertible, with inverse

$$\psi_\delta^{-1}(x - \delta v_{i-1}, v_0, \ldots, v_{B-1}, i - 1 \bmod B).$$

Invertibility of ψ_δ is all that is needed for ψ_δ to be used as a valid integrator in an integrator snippet. However, we shall show that this integrator satisfies the stronger condition that there exists a function σ on $\mathbb{R}^{2(B+1)} \times [\![0, B-1]\!]$ such that $\sigma^2 = \mathrm{Id}$, $\psi^{-1} = \sigma \circ \psi \circ \sigma$. In this case, we define

$$\sigma(x, v_0, \ldots, v_{B-1}, i) = (x, -v_{B-1}, \ldots, -v_0, B - i - 1 \bmod B).$$

That this operation is an involution is straightforward for the first $B+1$ variables and one only needs to show that

$$B - (B - i - 1)) - 1 \bmod B = i \bmod \mathrm{B}.$$

Now we need to show that $\sigma \circ \psi_\delta \circ \sigma = \psi_\delta^{-1}$

$$\begin{aligned}
\sigma \circ \psi_\delta \circ \sigma(x, v_0, \ldots, v_B, i) &= \sigma \circ \psi_\delta(x, -v_{B-1}, \ldots, -v_0, B - i - 1 \bmod B) \\
&= \sigma(x - \delta v_i, -v_{B-1}, \ldots, -v_0, B - i \bmod B) \\
&= (x - \delta v_i, v_0, \ldots, v_{B-1}, i - 1 \bmod B) \\
&= \psi_\delta^{-1}(x, v_0, \ldots, v_{B-1}, \iota).
\end{aligned}$$

Finally, one needs to show that σ leaves the dominating measure invariant. In this case, the dominating measure is a product of a Lebesgue measure and a counting measure. Negating and changing the orders of the velocities does not affect volumes, so we only need to show that $\tilde{\sigma}(i) = B - i - 1 \bmod B$ leaves the counting measure invariant. In other words, we wish to show that

$$|\tilde{\sigma}^{-1}(\mathsf{A})| = |\mathsf{A}| \qquad \forall\, \mathsf{A} \in \mathscr{P}([\![0, B-1]\!]).$$

Since $\mathsf{A} \in \mathscr{P}([\![0,B-1]\!])$, we know that each of its elements is in $[\![0,B-1]\!]$ and therefore the function $\tilde{\sigma}$ is monotonically decreasing, hence one to one, which means that the number of elements in $\tilde{\sigma}^{-1}(\mathsf{A})$ is the same as the number of elements in A.

D.1.6 Derivation of ESJD for Integrator Snippets

From subsection 3.2.2 we know that the PISA estimator for $\mu(h)$ is

$$H = \frac{1}{N(T+1)} \sum_{i=1}^{N} \sum_{k=0}^{T} g_k(Z^{(i)}) \qquad Z^{(i)} \sim \eta. \tag{D.4}$$

where we have defined

$$g_k(Z) := h(\psi^k(Z)) \frac{\mu(\psi^k(Z))}{\eta(Z)} \qquad k \in [\![0, T]\!].$$

We are interested in deriving metric of expected travelled distance such that when it is maximised, the variance of this estimator is minimized. For simplicity, we will assume $\mu(g) = 0$ since its value will not matter for variance purposes. The variance of (D.4), assuming independence of the $Z^{(i)}$'s is

$$\mathbb{V}[H] = \frac{1}{N(T+1)^2} \left\{ \sum_{k=0}^{T} \mathbb{E}[g_k^2(Z)] + 2 \sum_{k=0}^{T} \sum_{\ell=k+1}^{T} \mathbb{E}[g_\ell(Z) g_k(Z)] \right\}. \tag{D.5}$$

Inside the curly brackets, the first term is a sum of variances and the second term is a sum of the covariance terms stemming from the fact that for fixed Z, $g_\ell(Z)$ and $g_k(Z)$ are generally not independent. We wish to write this covariance term in a sum-of-squared-distances form

$$\sum_{k=0}^{T} \sum_{\ell=0}^{T} \mathbb{E}\left[(g_\ell(Z) - g_k(Z))^2\right] = 2T \sum_{k=0}^{T} \mathbb{E}[g_k^2(Z)] - 4 \sum_{k=0}^{T} \sum_{\ell=k+1}^{T} \mathbb{E}[g_\ell(Z) g_k(Z)].$$

Substituting the second term on the RHS into (D.5) we get

$$\mathbb{V}[H] = \frac{1}{N(T+1)^2} \left\{ (T+1) \sum_{k=0}^{T} \mathbb{E}[g_k^2(Z)] - \sum_{k=0}^{T} \sum_{\ell=k+1}^{T} \mathbb{E}[(g_\ell(Z) - g_k(Z))^2] \right\}.$$

Notice that the larger the second term, the smaller the overall variance of H, hence we take

$$\mathrm{ESJD} = \frac{1}{N(T+1)^2} \sum_{k=0}^{T} \sum_{\ell=k+1}^{T} \mathbb{E}[(g_\ell(Z) - g_k(Z))^2].$$

We approximate the inner expectation as follows

$$\mathbb{E}[(g_\ell(Z)-g_k(Z))^2] = \int \left[h\circ\psi^\ell(z)\frac{\mu\circ\psi^\ell(z)}{\eta(z)} - h\circ\psi^k(z)\frac{\mu\circ\psi^k(z)}{\eta(z)}\right]^2 \eta(dz)$$
$$\approx N\sum_{i=1}^{N}\left[h(\psi^\ell(Z^{(i)}))\frac{\breve{w}_\ell(Z^{(i)})}{\sum_{j=1}^{N}\breve{w}_\ell(Z^{(j)})} - h(\psi^k(Z^{(i)}))\frac{\breve{w}_k(Z^{(i)})}{\sum_{j=1}^{N}\breve{w}_k(Z^{(j)})}\right]^2.$$

At iteration n this leads to the following estimator for the ESJD

$$\mathrm{ESJD}_n = \frac{1}{(T+1)^2}\sum_{k=0}^{T}\sum_{\ell=k+1}^{T}\sum_{i=1}^{N}\left[h(\psi_n^\ell(Z_{n-1}^{(i)}))\frac{\breve{w}_{n,\ell}(Z_{n-1}^{(i)})}{\sum_{j=1}^{N}\breve{w}_{n,\ell}(Z_{n-1}^{(j)})} - h(\psi_n^k(Z_{n-1}^{(i)}))\frac{\breve{w}_{n,k}(Z_{n-1}^{(i)})}{\sum_{j=1}^{N}\breve{w}_{n,k}(Z_{n-1}^{(j)})}\right]^2.$$

An alternative estimator can be found by recalling that the normalizing constants of $\mu_n^{\psi^{-k}}$ are identical for any $k\in\mathbb{Z}$, since ψ is volume preserving

$$\mathrm{ESJD}_n \approx \frac{\sum_{k=0}^{T}\sum_{\ell=k+1}^{T}\sum_{i=1}^{N}\left(h\circ\psi_n^\ell(Z_{n-1}^{(i)})\breve{w}_{n,\ell}(Z_{n-1}^{(i)}) - h\circ\psi_n^k(Z_{n-1}^{(i)})\breve{w}_{n,k}(Z_{n-1}^{(i)})\right)^2}{\left(\sum_{k=0}^{T}\sum_{i=1}^{N}\breve{w}_{n,k}(Z_{n-1}^{(i)})\right)^2}.$$

D.2 Supplementary Materials for Markov Snippets

D.2.1 A near-optimal SMC sampler for certain mixture distributions

In subsection 1.8.6.2 we have seen the general formula for the incremental weights of an SMC sampler using the near-optimal backward kernel. The next theorem gives an explicit formula for those incremental weights when the target and the proposal distributions are the joint distributions on the space of (Markov) snippets such that their marginals are mixtures, akin to those introduced in section 3.3.

Theorem D.2 (A near-optimal SMC sampler for mixtures)**.** *Let $\mu \ll \eta$ be two probability distributions on $(\mathsf{Z},\mathcal{Z})$, and $M,L,R : \mathsf{Z}\times\mathcal{Z}\to[0,1]$ be Markov kernels such that $\mu\oplus L^k \ll \mu\otimes M^k$ for any $k\in\mathbb{N}$ so that the Radon-Nikodym derivatives on $(\mathsf{Z}^{T+1},\mathcal{Z}^{\otimes(T+1)})$*

$$w_{\mu,k}(\mathbf{z}) = \frac{d\mu\oplus L^k}{d\mu\otimes M^k}(\mathbf{z}) \qquad \textit{and} \qquad w_{\eta,k}(\mathbf{z}) = \frac{d\eta\oplus L^k}{d\eta\otimes M^k}(\mathbf{z})$$

are well-defined, with $\mathbf{z} = (z_0, z_1, \dots, z_T)$ *and* $T \in \mathbb{Z}_+$. *Define the marginal distributions*

$$\bar{\mu}(d\mathbf{z}) = \left[\frac{1}{T+1} \sum_{k=0}^{T} w_{\mu,k}(\mathbf{z}) \right] \mu \otimes M^{\otimes T}(d\mathbf{z})$$
$$\bar{\eta}(d\mathbf{z}) = \left[\frac{1}{T+1} \sum_{k=0}^{T} w_{\eta,k}(\mathbf{z}) \right] \eta \otimes M^{\otimes T}(d\mathbf{z})$$

on $(\mathsf{Z}^{T+1}, \mathscr{Z}^{\otimes(T+1)})$ *as well as the Markov kernel* $\bar{M}_\eta : \mathsf{Z}^{T+1} \times \mathscr{Z}^{\otimes(T+1)} \to [0,1]$

$$\bar{M}_\eta(\mathbf{z}, d\mathbf{z}') = \sum_{k=0}^{T} \frac{w_{\eta,k}(\mathbf{z})}{\sum_{\ell=0}^{T} w_{\eta,k}(\mathbf{z})} R(z_k, dz'_0) M^{\otimes T}(z'_0, dz'_{1:T}).$$

Then the near-optimal (in the sense of subsection 1.8.6.2) backward kernel

$$\bar{L}_\eta(\mathbf{z}', d\mathbf{z}) = \frac{\bar{\eta}(d\mathbf{z}) \bar{M}_\eta(\mathbf{z}, \cdot)}{\bar{\eta}\bar{M}_\eta(\cdot)}(\mathbf{z}')$$

exists and when $\eta R = \eta$ *the incremental weights (of Equation* (1.27)*) are given by*

$$\bar{w}(\mathbf{z}') = \frac{\mu(dz'_0)}{\eta(dz'_0)} \left[\frac{1}{T+1} \sum_{k=0}^{T} \frac{\mu(dz'_k) L^k(z'_k, dz'_0)}{\mu(dz'_0) M^k(z'_0, dz'_k)} \right],$$

whenever $\bar{\mu} \otimes \bar{L}_\eta \ll \bar{\eta} \otimes \bar{M}_\eta$.

Proof. To prove the existence of the backward kernel, the same arguments in the proof of Theorem D.1 remain valid, since it is once more covered by the general theory of Theorem A.2. Assuming $\bar{\mu} \otimes \bar{L}_\eta \ll \bar{\eta} \otimes \bar{M}_\eta$, the weights are then given by

$$\begin{aligned} \bar{w}(\mathbf{z}') &= \frac{d\bar{\mu} \oplus \bar{L}_\eta}{\bar{\eta} \otimes \bar{M}_\eta}(\mathbf{z}, \mathbf{z}') \\ &= \frac{\bar{\mu}(d\mathbf{z})}{\bar{\eta}\bar{M}_\eta(d\mathbf{z})} \qquad (1.27). \end{aligned}$$

To proceed, we need to simplify the denominator.

$$\begin{aligned} \bar{\eta}\bar{M}_\eta(d\mathbf{z}) &= \int_{\mathsf{Z}^{T+1}} \left[\frac{1}{T+1} \sum_{k=0}^{T} w_{\eta,k}(\mathbf{z}) \right] \eta \otimes M^{\otimes T}(d\mathbf{z}) \sum_{k=0}^{T} \frac{w_{\eta,k}(\mathbf{z})}{\sum_{\ell=0}^{T} w_{\eta,k}(\mathbf{z})} R(z_k, dz'_0) M^{\otimes T}(z'_0, dz'_{1:T}) \\ &= \frac{1}{T+1} \sum_{k=0}^{T} \int_{\mathsf{Z}^{T+1}} \eta(dz_0) M^{\otimes T}(z_0, dz_{1:T}) \frac{\eta(dz_k) L^k(z_k, dz_0)}{\eta(dz_0) M^k(z_0, dz_k)}(\mathbf{z}) R(z_k, dz'_0) M^{\otimes T}(z'_0, dz'_{1:T}) \\ &= \left[\frac{1}{T+1} \sum_{k=0}^{T} \int_{\mathsf{Z}^2} \eta(dz_0) L^k(z_k, dz_0) R(z_k, dz'_0) \right] M^{\otimes T}(z'_0, dz'_{1:T}) \\ &= \left[\frac{1}{T+1} \sum_{k=0}^{T} \int_{\mathsf{Z}} \eta(dz_0) R(z_k, dz'_0) \right] M^{\otimes T}(z'_0, dz'_{1:T}) \\ &= \eta R(dz'_0) M^{\otimes T}(z'_0, dz'_{1:T}) \\ &= \eta(dz'_0) M^{\otimes T}(z'_0, dz'_{1:T}), \end{aligned}$$

where in the last line we have used $\eta R = \eta$. The incremental weights then simplify to

$$\begin{aligned}
\bar{w}(\mathbf{z}') &= \frac{\bar{\mu}(d\mathbf{z}')}{\eta \otimes M^{\otimes T}(d\mathbf{z}')} && \eta R = \eta \\
&= \frac{\frac{1}{T+1}\sum_{k=0}^{T} w_{\mu,k}(\mathbf{z}')\mu \otimes M^{\otimes T}(d\mathbf{z}')}{\eta \otimes M^{\otimes T}(d\mathbf{z}')} && \bar{\mu} \text{ def.} \\
&= \frac{\mu(dz_0')}{\eta(dz_0')}\left[\frac{1}{T+1}\sum_{k=0}^{T} w_{\mu,k}(\mathbf{z}')\right].
\end{aligned}$$

■

It turns out that in the context of this SMC sampler for mixtures, one can estimate expectations with respect to μ by using samples from $\bar{\mu}$ and this is exactly what the next theorem shows.

Theorem D.3 (Estimating μ-expectations with a near-optimal SMC sampler for mixtures)**.** *Let μ be a probability distribution on $(\mathsf{Z}, \mathscr{Z})$, $M, L : \mathsf{Z} \times \mathscr{Z} \to [0,1]$ be Markov kernels such that for any $k \in [\![0,T]\!]$, $T \in \mathbb{Z}_+$, the Radon-Nikodym derivative*

$$w_{\mu,k}(\mathbf{z}) = \frac{d\mu \oplus L^k}{d\mu \otimes M^k}(\mathbf{z})$$

is well-defined, with the convention $w_{\mu,0}(\mathbf{z}) = 1$. Define the joint distribution which for cylinder sets $\{k\} \times d\mathbf{z}$ has the form

$$\bar{\mu}_T(k, d\mathbf{z}) = \frac{1}{T+1} w_{\mu,k}(\mathbf{z})\mu \otimes M^{\otimes T}(d\mathbf{z})$$

on $([\![0,T]\!] \times \mathsf{Z}^{T+1}, \mathscr{P}([\![0,T]\!]) \otimes \mathscr{Z}^{\otimes(T+1)})$, with marginal $\bar{\mu}(d\mathbf{z})$ as in Theorem 3.1. Then for any test function $h : \mathsf{Z}^{T+1} \to \mathbb{R}$ such that $\mu(|h|) < \infty$ for which we define $\bar{h}(k, \mathbf{z}) := h(z_k)$, we have

$$\bar{\mu}_T(\bar{h}) = \mu(h),$$

and these expectations can be written as

$$\int_{\mathsf{Z}^{T+1}} \sum_{k=0}^{T} \bar{h}(k, \mathbf{z}) \frac{w_{\mu,k}(\mathbf{z})}{\sum_{\ell=0}^{T} w_{\mu,\ell}(\mathbf{z})} \bar{\mu}(d\mathbf{z}) = \mu(h).$$

Proof. The first statement follows from the Radon-Nikodym theorem

$$\begin{aligned}
\bar{\mu}_T(\bar{h}) &= \sum_{k=0}^{T} \int_{\mathsf{Z}^{T+1}} \hat{h}(k, d\mathbf{z})\bar{\mu}_T(k, d\mathbf{z}) \\
&= \frac{1}{T+1}\sum_{k=0}^{T} \int_{\mathsf{Z}^{T+1}} h(z_k) \frac{\mu(z_k)L^k(z_k, dz_0)}{\mu(dz_0)M^k(z_0, dz_k)} \mu(z_0)M^{\otimes T}(z_0, dz_{1:T}) && \text{defs of } \bar{h}, \bar{\mu}_T, w_{\mu,k} \\
&= \frac{1}{T+1}\sum_{k=0}^{T} \int_{\mathsf{Z}^2} h(z_k) \frac{\mu(z_k)L^k(z_k, dz_0)}{\mu(dz_0)M^k(z_0, dz_k)} \mu(z_0)M^k(z_0, dz_k) && \text{def of } M^k
\end{aligned}$$

$$= \frac{1}{T+1}\sum_{k=0}^{T}\int_{\mathsf{Z}^2} h(z_k)\mu(z_k)L^k(z_k,dz_0) \qquad \text{RN theorem}$$

$$= \mu(h) \qquad L^k \text{ Markov kernel}$$

For the second statement, first notice that for cylinder sets $\{k\}\times d\mathbf{z}$ the natural regular conditional distribution related to the marginal $\bar{\mu}(d\mathbf{z})$ and the joint $\bar{\mu}_T(k,d\mathbf{z})$ is given by

$$\bar{\mu}(k \mid \mathbf{z}) = \frac{\bar{\mu}_T(k,d\mathbf{z})}{\bar{\mu}(d\mathbf{z})} = \frac{w_{\mu,k}(\mathbf{z})}{\sum_{\ell=0}^{T} w_{\mu,\ell}(\mathbf{z})}.$$

Then, using the first line of the proof of the first statement and Fubini's theorem

$$\begin{aligned}
\bar{\mu}_T(\bar{h}) &= \sum_{k=0}^{T}\int_{\mathsf{Z}^{T+1}} \bar{h}(k,\mathbf{z})\bar{\mu}_T(k,d\mathbf{z}) \\
&= \int_{\mathsf{Z}^{T+1}} \sum_{k=0}^{T} \bar{h}(k,\mathbf{z})\bar{\mu}(k \mid \mathbf{z})\bar{\mu}(d\mathbf{z}) \\
&= \int_{\mathsf{Z}^{T+1}} \sum_{k=0}^{T} \bar{h}(k,\mathbf{z})\frac{w_{\mu,k}(\mathbf{z})}{\sum_{\ell=0}^{T} w_{\mu,\ell}(\mathbf{z})}\bar{\mu}(d\mathbf{z}) \\
&= \mu(h).
\end{aligned}$$

■

To either compute the incremental weights $\bar{w}(\mathbf{z}')$ or estimating expectations, we need to be able to compute $w_{\mu,k}(\mathbf{z})$. The next subsection shows how to do this under mild conditions.

D.2.2 Computing weights in an SMC sampler for mixtures

In this subsection we derive a computable expression for the weights $w_{\mu,k}(\mathbf{z})$ appearing in the two preceding theorems. The first theorem of this section brings together various results from [4] which will be useful throughout. It defines a reversible triplet, states that it implies invariance and how it can be formulated in terms of measurable functions.

Theorem D.4 (Reversible Triplets and Invariance)**.** *Let* $(\mathsf{Z},\mathcal{Z})$ *be a measurable space,* v *be a probability measure on it, and* $M,L:\mathsf{Z}\times\mathcal{Z}\to[0,1]$ *be Markov probability kernels. We say that* (v,M,L) *is a reversible triplet if*

$$v(dz)M(z,dz') = v(dz')L(z',dz),$$

or, equivalently, if for any two bounded, measurable functions $f,g:\mathsf{Z}\to\mathbb{R}$

$$\int_{\mathsf{Z}^2} f(z)g(z')v(dz)M(z,dz') = \int_{\mathsf{Z}^2} f(z)g(z')v(dz')L(z',dz)$$

If (v,M,L) *is a reversible triplet, then* M *leaves* v *invariant.*

Proof. First we prove equivalence. For the backward direction, simply take $f = \mathbb{1}_A$ and $g = \mathbb{1}_B$ for any measurable sets $A, B \in \mathcal{Z}$. For the forward direction, we write

$$\int_{Z^2} \mathbb{1}_A(z)\mathbb{1}_B(z')\nu(dz)M(z,dz') = \int_{Z^2} \mathbb{1}_A(z)\mathbb{1}_B(z')\nu(dz')L(z',dz) \qquad A, B \in \mathcal{Z},$$

and then use the standard machinery [35]. To show invariance, first write the definition of reversible triplet in its integral form

$$\int_A \nu(dz)M(z,B) = \int_B \nu(dz')L(z',A)$$

and then simply take $A = Z$ and use the fact that $L(z', Z) = 1$ for any $z' \in Z$. ■

Reversible triplets are very useful in their own rights, but it turns out to be of fundamental importance in Markov Snippets. Given a reversible triplet (ν, M, L) where ν is some reference measure, one can find a simple expression for the Radon-Nikodym derivative of $\mu \otimes L$ with respect to $\mu \otimes M$, under very mild conditions.

Lemma D.1 (Radon-Nikodym derivatives and reversible triplets)**.** *Let $(Z, \mathcal{Z})$ be a measurable space, $\nu \gg \mu$ be two measures on it, and (ν, M, L) be a reversible triplet. Then for any $z, z' \in Z$ with $\mu(z) > 0$ and $k \in \mathbb{N}$ we have*

$$\frac{d\mu \otimes L^k}{d\mu \otimes M^k}(z,z') = \frac{\frac{d\mu}{d\nu}(z')}{\frac{d\mu}{d\nu}(z)}.$$

Proof. For any $z, z' \in Z$ with $\frac{d\mu}{d\nu}(z) > 0$

$$\begin{aligned}
\mu(dz')L(z',dz) &= \frac{\mu(dz')}{\nu(dz')}\nu(dz')L(z',dz) && \mu \ll \nu \\
&= \frac{\mu(dz')}{\nu(dz')}\nu(dz)M(z,dz') && \text{reversible triplet} \\
&= \frac{\frac{d\mu}{d\nu}(z')}{\frac{d\mu}{d\nu}(z)}\frac{\mu(dz)}{\nu(dz)}\nu(dz)M(z,dz') && \frac{d\mu}{d\nu}(z) > 0 \\
&= \frac{\frac{d\mu}{d\nu}(z')}{\frac{d\mu}{d\nu}(z)}\mu(dz)M(z,dz') && \text{Radon-Nikodym theorem.}
\end{aligned}$$

Once more, by the Radon-Nikodym theorem, we have

$$\frac{d\mu \otimes L}{d\mu \otimes M} = \frac{\frac{d\mu}{d\nu}(z')}{\frac{d\mu}{d\nu}(z)}.$$

Now let $k \in \mathbb{N}$. Notice that if M (or respectively L) is a Markov kernel, then so is

$$M^k(z_0, dz_k) = \int \prod_{i=1}^{k} M(z_{i-1}, dz_i),$$

where the integration is over all the intermediate states and thus we refer to M^k (and respectively L^k) as the k-step marginal kernel. Importantly, we see that if (υ, M, L) is a reversible triplet, so is (υ, M^k, L^k) since

$$\begin{aligned}
\upsilon(dz_k)L^k(z_k, dz_0) &= \upsilon(dz_k)\int \prod_{i=1}^{k} L(z_i, dz_{i-1}) \\
&= \int \upsilon(dz_k)L(z_k, dz_{k-1})\prod_{i=1}^{k-1} L(z_i, dz_{i-1}) \\
&= \int M(z_{k-1}, dz_k)\upsilon(dz_{k-1})\prod_{i=1}^{k-1} L(z_i, dz_{i-1}) \\
&\vdots \\
&= \int \upsilon(dz_0)\prod_{i=1}^{k} M(z_{i-1}, dz_i) \\
&= \upsilon(dz_0)M^k(z_0, dz_k).
\end{aligned}$$

As a consequence, the proof derived earlier for (υ, M, L) also works for (υ, M^k, L^k) and we have

$$\frac{d\mu \otimes L^k}{d\mu \oplus M^k}(z_0, z_k) = \frac{\frac{d\mu}{d\upsilon}(z_k)}{\frac{d\mu}{d\upsilon}(z_0)}. \tag{D.6}$$

In typical scenarios, υ will be the Lebesgue measure and μ our target measure. ■

Therefore the weights appearing in Theorems 3.1 and D.3 can typically be computed as

$$w_{\mu,k}(\mathbf{z}) = \frac{\mu(z_k)}{\mu(z_0)}$$

where $\mu(\cdot)$ is the density of $\mu(dz)$ with respect to the Lebesgue measure.

D.2.3 Integrator Snippets as a special case of Markov Snippets

In this subsection we show how integrator snippets are simply a special case of Markov snippets where the Markov kernel M is deterministic and has the form

$$M(z, dz') = \delta_{\psi(z)}(dz'), \tag{D.7}$$

for a suitable function ψ.

Theorem D.5 (Deterministic Reversible Triplets)**.** *Let υ be a finite measure on $(\mathsf{Z}, \mathcal{Z})$ and $\psi : \mathsf{Z} \to \mathsf{Z}$ be an invertible mapping such that $\upsilon^\psi = \upsilon$. Then $(\upsilon, \delta_{\psi(z)}(dz'), \delta_{\psi^{-1}(z')}(dz))$ is a reversible triplet.*

Proof. From Theorem D.4 we only need to show that for any bounded measurable functions $f, g : \mathsf{Z} \to \mathbb{R}$

$$\int_{\mathsf{Z}^2} f(z)g(z')\upsilon(dz)\delta_{\psi(z)}(dz') = \int_{\mathsf{Z}^2} f(z)g(z')\upsilon(dz')\delta_{\psi(z')}(dz).$$

Starting from the LHS we have

$$\begin{aligned}
\int_{Z^2} f(z)g(z')v(dz)\delta_{\psi(z)}(dz') &= \int_Z f(z)g(\psi(z))v(dz) && \text{Dirac Delta definition} \\
&= \int_Z f(\psi^{-1}(\psi(z))g(\psi(z))v(dz) \\
&= \int_Z f(\psi^{-1}(z'))g(z')v^\psi(dz') && \text{(A.1)} \\
&= \int_{Z^2} f(z)g(z')v^\psi(dz')\delta_{\psi^{-1}(z')}(dz) && \text{Dirac Delta definition.}
\end{aligned}$$

■

In the previous result we have assumed that $\nu^\psi = \nu$. Typically ν will be the Lebesgue measure but our measure of interest will instead be another measure $\mu \ll \nu$ that might not satisfy $\mu^\psi = \mu$. The following theorem covers that case, and tells us the form of $w_{\mu,k}$ when using (D.7).

Theorem D.6 (Weights $w_{\mu,k}$ for deterministic kernels)**.** *Let v be a finite measure on $(Z, \mathcal{Z})$, and $\psi : Z \to Z$ be an invertible transformation such that $v^\psi = v$. Let $\mu \ll v$ be a probability measure on $(Z, \mathcal{Z})$ and $M, L : Z \times \mathcal{Z} \to [0,1]$ be Markov kernels. For any $k \in \mathbb{N}$, if $\frac{d\mu}{dv}(z) > 0$ v-almost everywhere, then*

$$\frac{\mu(dz')\delta_{\psi^{-k}(z')}(dz)}{\mu(dz)\delta_{\psi^k(z)}(dz')} = \frac{\dfrac{d\mu}{dv}(\psi^k(z))}{\dfrac{d\mu}{dv}(z)}.$$

Proof. Assume that $\frac{d\mu}{dv}(z) > 0$ v-almost everywhere on Z. Then

$$\begin{aligned}
\int_{Z^2} f(z)g(z')\mu(dz')\delta_{\psi^{-1}(z')}(dz) &= \int_{Z^2} f(z)g(z')\frac{d\mu}{dv}(z')v(dz')\delta_{\psi^{-1}(z')}(dz) && \text{RN theorem} \\
&= \int_Z f(\psi^{-1}(z'))g(z')\frac{d\mu}{dv}(z')v(dz') && \text{Dirac delta definition} \\
&= \int_Z f(\psi^{-1}(z'))g(z')\frac{d\mu}{dv}(z')v^\psi(dz') && v^\psi = v \\
&= \int_Z f(z)g\circ\psi(z)\frac{d\mu}{dv}\circ\psi(z)v(dz) && \text{(A.1)} \\
&= \int_{Z^2} f(z)g(z')\frac{d\mu}{dv}(z')v(dz)\delta_{\psi(z)}(dz') && \text{Dirac delta definition} \\
&= \int_{Z^2} f(z)g(z')\frac{\dfrac{d\mu}{dv}(z')}{\dfrac{d\mu}{dv}(z)}\mu(dz)\delta_{\psi(z)}(dz') && \frac{d\mu}{dv}(z) > 0 \ v\text{-a.e.}
\end{aligned}$$

and since this works for all bounded, measurable functions $f, g : Z \to \mathbb{R}$ we conclude

$$\frac{\mu(dz')\delta_{\psi^{-1}(z')}(dz)}{\mu(dz)\delta_{\psi(z)}(dz')} = \frac{\dfrac{d\mu}{dv}(\psi(z))}{\dfrac{d\mu}{dv}(z)}.$$

Finally, we notice that the proof above can be repeated by taking ψ^k as our invertible transformation, for any $k \in \mathbb{N}$ since $v^{\psi} = v$ implies $v^{\psi^k} = v$. ■

In most cases of interest, v will be the Lebesgue measure, in which case we can write the weights as

$$w_{\mu,k}(\mathbf{z}) = \frac{\mu \circ \psi^k(z)}{\mu(z)},$$

and automatically the incremental weights of Theorem 3.1 become

$$\bar{w}(\mathbf{z}') = \frac{1}{T+1} \sum_{k=0}^{T} \frac{\mu \circ \psi^k(z'_0)}{\eta(z'_0)},$$

which explains the weights introduced in Section 3.3, where z'_0 should be replaced by $\tilde{Z}_n^{(i)}$. For completeness, one should also show that $M^k(z, dz') = \delta_{\psi^k(z)}(dz')$, and we provide this below.

Lemma D.2 (Iterated Deterministic Kernels)**.** *Let* $(\mathsf{Z}, \mathscr{Z})$ *be a measurable space,* $\psi : \mathsf{Z} \to \mathsf{Z}$ *be an invertible function, and* $M(z, dz') = \delta_{\psi(z)}(dz')$ *and* $L(z', dz) = \delta_{\psi^{-1}(z')}(dz)$ *be deterministic Markov probability kernels. Then*

$$M^k(z_0, dz_k) := \delta_{\psi^k(z)}(dz')$$
$$L^k(z_k, dz_0) := \delta_{\psi^{-k}(z')}(dz).$$

Proof. By definition the k-iterations joint kernels $M^k, L_k : \mathsf{Z} \times \mathscr{Z}^{\otimes k} \to [0,1]$ are

$$M^{\otimes k}(z_0, dz_{1:k}) = \prod_{i=1}^{k} \delta_{\psi(z_{i-1})}(dz_i) = \delta_{(\psi(z_0),\ldots,\psi(z_{k-1}))}(dz_{1:k}) = \delta_{(\psi(z_0),\ldots,\psi^k(z_{k-1}))}(dz_{1:k})$$
$$L^{\otimes k}(z_k, dz_{k-1:1}) = \prod_{i=1}^{k} \delta_{\psi^{-1}(z_i)}(dz_{i-1}) = \delta_{(\psi^{-1}(z_k),\ldots,\psi^{-1}(z_0))}(dz_{k-1:1}) = \delta_{(\psi^{-1}(z_k),\ldots,\psi^{-k}(z_k))}(dz_{k-1:1}).$$

and the k-iteration marginal kernels are

$$M^k(z_0, dz_k) = \int_{z_{1:k-1} \in \mathsf{Z}^{k-1}} \delta_{(\psi(z_0),\ldots,\psi^k(z_{k-1}))}(dz_{1:k}) = \delta_{\psi^k(z_0)}(dz_k)$$
$$L^k(z_k, dz_0) = \int_{z_{k-1:1} \in \mathsf{Z}^{k-1}} \delta_{(\psi^{-1}(z_k),\ldots,\psi^{-k}(z_k))}(dz_{k-1:1}) = \delta_{\psi^{-k}(z_k)}(dz_0).$$

■

D.2.4 A Markov Snippet with a mixture of Markov kernels

In this section of the Appendix, we generalise the treatment in subsection 3.6.3 to a mixture of two Markov kernels, although the same ideas can be recycled to make this work for any finite mixture. For a mixture of two components, we shall work on the space $(\Gamma, \mathscr{G})$ defined in subsection

3.6.3 and use the mathring accent to denote quantities on this space. The target and proposal will be $\mathring{\mu}(d\gamma) = \mu(dz)\rho(\iota)$ and $\mathring{\eta}(d\gamma) = \eta(dz)\rho(\iota)$. We define the one-step kernels $\mathring{M}, \mathring{L} : \Gamma \times \mathscr{G} \to [0,1]$

$$\mathring{M}(\gamma, d\gamma') = \delta_{\iota\iota'} M_\iota(z, dz')$$
$$\mathring{L}(\gamma', d\gamma) = \delta_{\iota'\iota} L_{\iota'}(z', dz)$$

and it is straightforward to find out the expression for the product kernels

$$\mathring{M}^{\otimes T}(\gamma_0, d\gamma_{1:T}) = \left\{ \prod_{\ell=1}^{T} \delta_{\iota_{\ell-1}\iota_\ell} \right\} M_{\iota_0}^{\otimes T}(z_0, dz_{1:T})$$
$$\mathring{L}^{\otimes T}(\gamma_T, d\gamma_{T-1:0}) = \left\{ \prod_{\ell=1}^{T} \delta_{\iota_\ell\iota_{\ell-1}} \right\} L_{\iota_T}^{\otimes T}(z_T, dz_{T-1:0}),$$

and marginal k-step kernels

$$\mathring{M}^k(\gamma_0, d\gamma_k) = \left\{ \sum_{\iota_{1:k-1}, \iota_{k+1:T}} \prod_{\ell=1}^{T} \delta_{\iota_\ell\iota_{\ell-1}} \right\} M_{\iota_0}^k(z_0, dz_k)$$
$$\mathring{L}^k(\gamma_k, d\gamma_0) = \left\{ \sum_{\iota_{1:k-1}} \prod_{\ell=1}^{T} \delta_{\iota_\ell\iota_{\ell-1}} \right\} L_{\iota_k}^k(z_k, dz_0).$$

We define the refreshment kernel $\mathring{R}(\gamma, d\gamma') = R(z, dz')\rho(\iota')$ and we wish to use Theorem 3.1 with these quantities (and by defining all the Radon-Nikodym in an equivalent manner) on $(\Gamma, \mathscr{G})$ to show correctness. There are three things that needs to be shown

1. $\mu \oplus L_\iota^k \ll \mu \otimes M_\iota^k$ for $\iota = 0,1$ implies $\mathring{\mu} \oplus \mathring{L}^k \ll \mathring{\mu} \otimes \mathring{M}^k$ for any $k \in \mathbb{N}$.
2. $\eta R = \eta$ implies $\mathring{\eta}\mathring{R} = \mathring{\eta}$.
3. $\mathring{\overset{\circ}{\eta}}\mathring{\overset{\circ}{M}}_{\mathring{\eta}} = \mathring{\eta}$

and once this is done, we can find an expression for the folded weights. To prove the first statement, it is enough to focus on measurable rectangles $\mathsf{A} = \mathsf{B} \times \mathsf{C} \times \mathsf{D} \times \mathsf{E}$ where $\mathsf{B}, \mathsf{D} \in \mathscr{Z}$ and $\mathsf{C}, \mathsf{E} \in \mathscr{P}(\{0,1\})$, it follows from

$$\mathring{\mu} \otimes \mathring{M}^k(\mathsf{A}) = \int_{\mathsf{B}\times\mathsf{D}} \sum_{\iota\in\mathsf{C}, \iota'\in\mathsf{E}} \mu(dz)\rho(\iota)\delta_{\iota\iota'} M_\iota^k(z, dz') = \sum_{\iota'\in\mathsf{E}} \rho(\iota')\mu \otimes M_{\iota'}^k(\mathsf{B}\times\mathsf{D})$$
$$\mathring{\mu} \oplus \mathring{L}^k(\mathsf{A}) = \int_{\mathsf{B}\times\mathsf{D}} \sum_{\iota\in\mathsf{C}, \iota'\in\mathsf{E}} \mu(dz')\rho(\iota')\delta_{\iota'\iota} L_{\iota'}^k(z', dz) = \sum_{\iota\in\mathsf{C}} \rho(\iota)\mu \oplus L_\iota^k(\mathsf{B}\times\mathsf{D}).$$

The second statement follows immediately from the definition of $\mathring{R}$ and the assumption that $\eta R = \eta$. Once more, we consider a measurable rectangle $\mathsf{A} \in \mathscr{G}$ of the form $\mathsf{A} = \mathsf{B} \times \mathsf{C}$ with $\mathsf{B} \in \mathscr{Z}$ and $\mathsf{C} \in \mathscr{P}(\{0,1\})$

$$\mathring{\eta}\mathring{R}(\mathsf{A}) = \int_{\mathsf{Z}} \sum_{\iota=0,1} \eta(dz)\rho(\iota)R(z, \mathsf{B})\rho(\mathsf{C}) = \rho(\mathsf{C})\eta(\mathsf{B}) = \mathring{\eta}(\mathsf{A}).$$

For the final proof, we first find an expression for the weights $\mathring{w}_{\mathring{\eta},k}$

$$\mathring{w}_{\mathring{\eta},k}(\gamma_0,\gamma_k) = \delta_{\iota_0\iota_k}\frac{\eta(dz_k)L^k_{\iota_0}(z_k,dz_0)}{\eta(dz_0)M^k_{\iota_0}(z_0,dz_k)}.$$

and then proceed as in the proof of Theorem 3.1

$$\mathring{\tilde{\eta}}\mathring{\tilde{M}}_{\mathring{\eta}}(d\vec{\gamma}') = \left[\frac{1}{T+1}\sum_{k=0}^{T}\sum_{\vec{\iota}}\rho(\iota_0)\left\{\prod_{\ell=1}^{T}\delta_{\iota_{\ell-1}\iota_\ell}\right\}\delta_{\iota_0\iota_k}\right]\mathring{\eta}\otimes\mathring{M}^{\otimes T}(d\vec{\gamma}') = \mathring{\eta}\otimes\mathring{M}^{\otimes T}(d\vec{\gamma}'),$$

where above the summation over $\vec{\iota}$ means that we are summing over all $\iota_0,\ldots,\iota_T$. With this result, the SMC incremental weights on this extended space become

$$\mathring{\tilde{w}}(\vec{\gamma}') = \frac{\mu(dz'_0)}{\eta(dz'_0)}\left[\frac{1}{T+1}\sum_{k=0}^{T}\delta_{\iota'_0\iota'_k}\frac{\mu(dz'_k)L^k_{\iota'_0}(z'_k,dz'_0)}{\mu(dz'_0)M^k_{\iota'_0}(z'_0,dz'_k)}\right].$$

Notice that in the special case when the Markov kernels are deterministic updates as those considered in subsection 3.6.3, the unfolded weights reduce to the ones found in the main text.

D.2.5 Correctness of SMC sampler with mixture of THUG and NHUG

To prove correctness of the marginal SMC sampler described in the main text, we need to show the correctness of the joint one, i.e. the one working both on the x and ι variables. Let $(\Gamma,\mathscr{G}) = (\mathsf{X}\times\{0,1\},\mathscr{X}\otimes\mathscr{P}(\{0,1\}))$ be the product space with reference dominating measure $\lambda_\Gamma = \mathrm{Leb}_\mathsf{X}\otimes\#$, where # is the counting measure on $(\{0,1\},\mathscr{P}(\{0,1\}))$. Define the extended target on this space as

$$\eta_n(d\gamma) = \eta_n(dx)p^\iota(1-p)^{1-\iota},$$

where we define the variable $\gamma = (x,\iota)$, $p\in(0,1]$, and η_n is defined as in the main text. Let $M_{\iota,n}, L_{\iota,n-1}:\mathsf{X}\times\mathscr{X}\to[0,1]$ be Markov kernels for $\iota = 0,1$ such that $(\eta_n, M_{\iota,n}, L_{\iota,n-1})$ are reversible triplets for $\iota = 0,1$. Notice that we are only defining two reversible triplets, one for each value of ι but we are saying nothing about the cross-triplets $(\mu_n, M_{0,n}, L_{1,n-1})$ and $(\mu_n, M_{1,n}, L_{0,n-1})$. We now define the extended kernels

$$M_n(\gamma_{n-1},d\gamma_n) = p^{\iota_{n-1}}(1-p)^{1-\iota_{n-1}}\sum_{\iota_n=0,1}\delta_{\iota_{n-1}\iota_n}M_{\iota_{n-1},n}(x_{n-1},dx_n)$$
$$L_{n-1}(\gamma_n,d\gamma_{n-1}) = p^{\iota_n}(1-p)^{1-\iota_n}\sum_{\iota_{n-1}=0,1}\delta_{\iota_n\iota_{n-1}}L_{\iota_n,n-1}(x_n,dx_{n-1}),$$

and we wish to show that $(\eta_n, M_{\iota,n}, L_{\iota,n-1})$ being reversible triplets implies that (η_n, M_n, L_{n-1}) is also a reversible triplet (notice the conflict of notation, in the first set of triplets η_n is the marginal $\eta_n(dx)$, whereas in the latter triplet it is the extended distribution $\eta_n(d\gamma)$). Proving this is straightforward

$$\eta_n(d\gamma_{n-1})M_n(\gamma_{n-1},d\gamma_n) = p^{\iota_{n-1}}(1-p)^{1-\iota_{n-1}}\sum_{\iota_n=0,1}\delta_{\iota_{n-1},\iota_n}M_{\iota_{n-1},n}(x_{n-1},dx_n)$$

D.3 Experimental Details

D.3.1 Selection of ellipsoid in many dimensions

For a d-dimensional multivariate normal distribution with diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ it is well-known that most of its mass will lie around an shell away from the mean. In order to select a non-degenerate contour, we notice that

$$\mathcal{N}(\Delta \mid 0_d, \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)) \approx \sqrt{\sum_{i=1}^{d} \sigma_i^2}$$

where $\Delta = (\sigma_1, \ldots, \sigma_d^2)$, and in our case since

$$\sigma_i = \begin{cases} 1 & i \text{ odd} \\ 0.1 & \text{even} \end{cases} \qquad i = 1, \ldots, d,$$

we have that approximately $\mathcal{N}(\Delta \mid 0_d, \Sigma) \approx \exp(-12)$ when $d = 20$, which is why we choose that as our contour value. In general, for $d > 2$ with Σ of the form above, we suggest choosing the contour value to be

$$\mathrm{c}_d = \left\lceil \frac{d}{4}\log(10) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{d} \sigma_i^2 \right\rceil.$$

D.3.2 Additional results for adaptivity in Integrator Snippets

We consider a toy example similar to the 2D-ellipsoid in the main text, but where the underlying distribution $v(dx)$ is now a (uniform) mixture of two Gaussians

$$v(dx) = \frac{1}{2}\sum_{c=0,1} \frac{1}{2\pi\sqrt{|\det \Sigma_c|}} \exp\left(-\frac{1}{2}(x-\mu_c)^\top \Sigma_c^{-1}(x-\mu_c)\right) dx$$

with parameters

$$\Sigma_0 = \Sigma_1 = \mathrm{diag}(0.8, 0.8)$$
$$\mu_0 = (-2, 0)^\top \qquad \mu_1 = (2, 0)^\top$$

Figure D.1 is Equivalent to Figure 3.37. We can see that the metric is indeed able to capture far away high-density regions and far away (or nearby) low-density regions correctly. However, as Figure D.2 shows, the signal is lost when summed over all particles. Interestingly, in this example the metric is more robust than in the previous one, probably due to the fact that now high density regions are far away from each other in Euclidean distance.

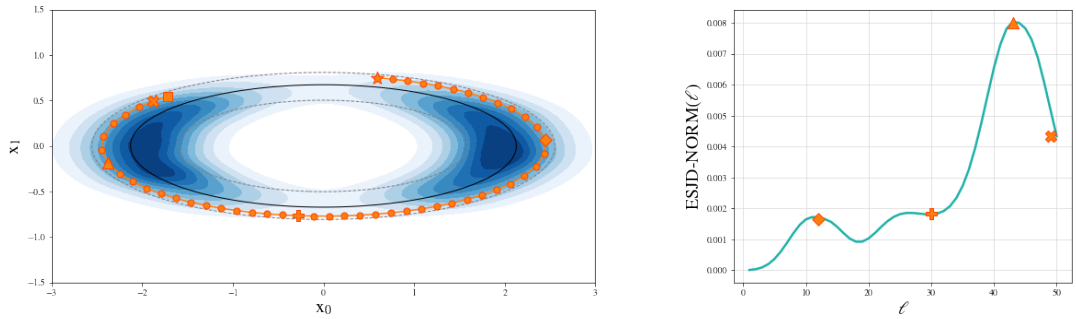

Figure D.1: ESJD-NORM for a single particle/trajectory.

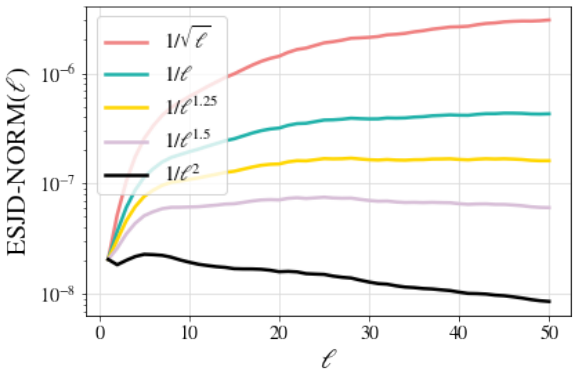

Figure D.2: ESJD-NORM with different coefficients taking into account the computational cost of constructing the path more or less severely.

APPENDIX E

G-AND-K REPARAMETRIZATION

E.1 Additional Details for G and K Experiment

The prior distribution used in ABC inference problems for the G and K distribution is traditionally

$$\theta \sim \mathcal{U}([0,10]^4).$$

The downside of using this prior distribution is that the samplers will need to work on the constrained space $[0,10]^4$. We prefer to work in unconstrained spaces and therefore reparametrize the problem so that the prior distribution is a four-dimensional standard normal distribution

$$\vartheta \sim \mathcal{N}(0, \mathrm{I}_4).$$

If one can find a mapping $\mathrm{G} : \vartheta \mapsto \theta$ then by the change of variables formula the new prior distribution would be

$$p_\vartheta(\vartheta) = p_\theta(\mathrm{G}(\vartheta))|\mathrm{J}_\mathrm{G}(\vartheta)|,$$

and the posterior distribution of ϑ given y would be

$$p_\vartheta(\vartheta \mid y) \propto p_\theta(\mathrm{G}(\vartheta))|\mathrm{J}_\mathrm{G}(\vartheta)|p(y \mid \mathrm{G}(\vartheta)).$$

To identify the reparametrized manifold in the (ϑ, z)-space, it is necessary to identify the new constraint function. Before reparametrization the constraint function was (see Equation (2.8))

$$f(\theta, z) = \left[\theta_1 + \theta_2\left(1 + 0.8\tanh\left(\frac{\theta_3 z}{2}\right)\right)(1+z^2)^{\theta_4} z\right] - y$$

where $\theta = (\theta_1, \theta_2, \theta_3, \theta_4) = (a, b, g, k)$, y is the observed data, z are latent variables and all operations are broadcasted from scalar to vector when necessary. After reparametrization, the constraint function is given by

$$f^\vartheta = f \circ \overline{\mathrm{G}}$$

where $\overline{G} : (\vartheta, z) \mapsto (\mathrm{G}(\vartheta), z)$, and the resulting manifold is therefore

$$\mathscr{M}^{\vartheta} = \left\{(\vartheta, z) : f^{\vartheta}(\vartheta, z) = 0\right\},$$

and the Jacobian function identifying the tangent planes is given by

$$\mathrm{J}_{f^{\vartheta}} = \mathrm{J}_{f}(\overline{\mathrm{G}}(\vartheta, z))\mathrm{J}_{\overline{\mathrm{G}}}(\vartheta, z).$$

If Φ is the CDF of a univariate standard normal distribution, then $\mathrm{G}(\vartheta) = 10(\Phi(\vartheta_1), \ldots, \Phi(\vartheta_4))$ and $\mathrm{J}_{\mathrm{G}}(\vartheta, z) = \mathrm{Diag}(10(\varphi(\vartheta_1), \ldots, \varphi(\vartheta_4)))$, where φ is the pdf of a univariate standard Gaussian distribution. The Jacobian of $\overline{\mathrm{G}}$ is thus

$$\mathrm{J}_{\overline{\mathrm{G}}}(\vartheta, z) = \begin{pmatrix} \mathrm{J}_{\mathrm{G}}(\vartheta, z) & 0_{4 \times m} \\ 0_{m \times 4} & \mathrm{I}_m \end{pmatrix}.$$

BIBLIOGRAPHY

[1] M. ABADI, A. AGARWAL, P. BARHAM, E. BREVDO, Z. CHEN, C. CITRO, G. S. CORRADO, A. DAVIS, J. DEAN, M. DEVIN, S. GHEMAWAT, I. GOODFELLOW, A. HARP, G. IRVING, M. ISARD, Y. JIA, R. JOZEFOWICZ, L. KAISER, M. KUDLUR, J. LEVENBERG, D. MANÉ, R. MONGA, S. MOORE, D. MURRAY, C. OLAH, M. SCHUSTER, J. SHLENS, B. STEINER, I. SUTSKEVER, K. TALWAR, P. TUCKER, V. VANHOUCKE, V. VASUDEVAN, F. VIÉGAS, O. VINYALS, P. WARDEN, M. WATTENBERG, M. WICKE, Y. YU, AND X. ZHENG, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.
Software available from tensorflow.org.

[2] H. C. ANDERSEN, *Rattle: A "Velocity" Version of the Shake Algorithm for Molecular Dynamics Calculations*, Journal of Computational Physics, 52 (1983), pp. 24–34.

[3] C. ANDRIEU, A. DOUCET, AND A. LEE, *Some discussions of d. fearnhead and d. prangle's read paper "constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation"*, 2012.

[4] C. ANDRIEU, A. LEE, AND S. LIVINGSTONE, *A general perspective on the metropolis-hastings kernel*, 2020.

[5] C. ANDRIEU AND J. THOMS, *A tutorial on adaptive mcmc*, Statistics and Computing, 18 (2008), pp. 343–373.

[6] U. ASCHER AND H. CHIN, *Stabilization of daes and invariant manifolds*, Numerische Mathematik, 67 (1994), pp. 131–149.

[7] U. M. ASCHER AND L. R. PETZOLD, *Stability of computational methods for constrained dynamics systems*, SIAM Journal on Scientific Computing, 14 (1993), pp. 95–120.

[8] K. X. AU, M. M. GRAHAM, AND A. H. THIERY, *Manifold lifting: scaling mcmc to the vanishing noise regime*, 2021.

[9] A. A. BARKER, *Monte carlo calculations of the radial distribution functions for proton-electron plasma*, Australian J. Phys., 18 (1965).

[10] E. BARTH, K. KUCZERA, B. J. LEIMKUHLER, AND R. D. SKEEL, *Algorithms for constrained molecular dynamics*, Journal of Computational Chemistry, 16 (1995).

[11] M. A. BEAUMONT, J.-M. CORNUET, J.-M. MARIN, AND C. P. ROBERT, *Adaptive approximate bayesian computation*, Biometrika, 96 (2009), pp. 983–990.

[12] M. A. BEAUMONT, W. ZHANG, AND D. J. BALDING, *Approximate bayesian computation in population genetics*, Genetics, 162 (2002), pp. 2025–2035.

[13] E. BERNTON, P. E. JACOB, M. GERBER, AND C. P. ROBERT, *On parameter estimation with the wasserstein distance*, 2019.

[14] A. BESKOS, N. S. PILLAI, G. O. ROBERTS, J. M. SANZ-SERNA, AND A. M. STUART, *Optimal tuning of the hybrid monte-carlo algorithm*, 2010.

[15] A. BESKOS, G. ROBERTS, A. THIERY, AND N. PILLAI, *Asymptotic analysis of the random walk Metropolis algorithm on ridged densities*, The Annals of Applied Probability, 28 (2018), pp. 2966 – 3001.

[16] M. BETANCOURT, *A conceptual introduction to hamiltonian monte carlo*, 2018.

[17] M. J. BETANCOURT, *Generalizing the no-u-turn sampler to riemannian manifolds*, 2013.

[18] P. BILLINGSLEY, *Probability and Measure*, John Wiley and Sons, second ed., 1986.

[19] D. M. BLEI, A. KUCUKELBIR, AND J. D. MCAULIFFE, *Variational inference: A review for statisticians*, Journal of the American Statistical Association, 112 (2017), p. 859–877.

[20] M. BOLIC, P. DJURIC, AND S. HONG, *Resampling algorithms for particle filters: A computational complexity perspective*, EURASIP Journal on Advances in Signal Processing, 2004 (2004).

[21] N. BOU-RABEE AND J. M. SANZ-SERNA, *Randomized hamiltonian monte carlo*, The Annals of Applied Probability, 27 (2017).

[22] A. BOUCHARD-CÔTÉ, S. J. VOLLMER, AND A. DOUCET, *The bouncy particle sampler: A non-reversible rejection-free markov chain monte carlo method*, 2017.

[23] J. BREHMER, G. LOUPPE, J. PAVEZ, AND K. CRANMER, *Mining gold from implicit models to improve likelihood-free inference*, Proceedings of the National Academy of Sciences, 117 (2020), p. 5242–5249.

[24] S. BROOKS, A. GELMAN, G. JONES, AND X.-L. MENG, *Handbook of Markov Chain Monte Carlo*, CRC press, 2011.

[25] M. BRUBAKER, M. SALZMANN, AND R. URTASUN, *A family of mcmc methods on implicitly defined manifolds*, in Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, N. D. Lawrence and M. Girolami, eds., vol. 22 of Proceedings

of Machine Learning Research, La Palma, Canary Islands, 21–23 Apr 2012, PMLR, pp. 161–172.

[26] S. L. CAMPBELL AND B. LEIMKUHLER, *Differentiation of constraints in differential-algebraic equations*, Mechanics of Structures and Machines, 19 (1991), pp. 19–39.

[27] H. CARTAN, *Differential Calculus*, International studies in mathematics, Hermann, 1983.

[28] F. CÉROU, P. DEL MORAL, T. FURON, AND A. GUYADER, *Sequential monte carlo for rare event estimation*, Statistics and Computing, 22 (2012), pp. 795–808.

[29] J. T. CHANG AND D. POLLARD, *Conditioning as disintegration*, Statistica Neerlandica, 51 (1997), pp. 287–317.

[30] T. CHEN, E. B. FOX, AND C. GUESTRIN, *Stochastic gradient hamiltonian monte carlo*, 2014.

[31] N. CHOPIN AND O. PAPASPILIOPOULOS, *An Introduction to Sequential Monte Carlo*, Springer Series in Statistics, Springer International Publishing, 2020.

[32] H. CHRISTIANSEN, F. ERRICA, AND F. ALESIANI, *Self-tuning Hamiltonian Monte Carlo for accelerated sampling*, The Journal of Chemical Physics, 159 (2023), p. 234109.

[33] G. CICCOTTI, R. KAPRAL, AND E. VANDEN-EIJNDEN, *Blue moon sampling, vectorial reaction coordinates, and unbiased constrained dynamics*, ChemPhysChem, 6 (2005), pp. 1809–1814.

[34] G. CICCOTTI, T. LELIÈVRE, AND E. VANDEN-EIJNDEN, *Projection of diffusions on submanifolds: Application to mean force computation*, Communications on Pure and Applied Mathematics, 61 (2008), pp. 371–408.

[35] E. ÇINLAR, *Probability and Stochastics*, Graduate Texts in Mathematics, Springer New York, 2011.

[36] P. E. CROUCH AND R. GROSSMAN, *Numerical integration of ordinary differential equations on manifolds*, Journal of Nonlinear Science, 3 (1993), pp. 1–33.

[37] H.-D. DAU AND N. CHOPIN, *Waste-free sequential monte carlo*, 2020.

[38] P. DEL MORAL, A. DOUCET, AND A. JASRA, *Sequential monte carlo samplers*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68 (2006), pp. 411–436.

[39] ——, *An adaptive sequential monte carlo method for approximate bayesian computation*, Statistics and Computing, 22 (2012), pp. 1009–1020.

[40] P. DIACONIS, S. HOLMES, AND M. SHAHSHAHANI, *Sampling from a manifold*, 2012.

[41] P. J. DIGGLE AND R. J. GRATTON, *Monte carlo methods of inference for implicit statistical models*, Journal of the Royal Statistical Society. Series B (Methodological), 46 (1984), pp. 193–227.

[42] J. V. DILLON, I. LANGMORE, D. TRAN, E. BREVDO, S. VASUDEVAN, D. MOORE, B. PATTON, A. ALEMI, M. HOFFMAN, AND R. A. SAUROUS, *Tensorflow distributions*, 2017.

[43] N. DING, Y. FANG, R. BABBUSH, C. CHEN, R. D. SKEEL, AND H. NEVEN, *Bayesian sampling using stochastic gradient thermostats*, in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, Cambridge, MA, USA, 2014, MIT Press, p. 3203–3211.

[44] S. DOGG, *I wanna thank me*.
`https://youtu.be/NfF3bThOW0Q?si=2qy4m-FqDr8X3E6v`, 2018.
Hollywood Star speech.

[45] A. DOUCET, N. DE FREITAS, AND N. GORDON, *Sequential Monte Carlo Methods in Practice.*, Springer, New York, 2001.

[46] A. DOUCET, N. DE FREITAS, AND N. J. GORDON, *An introduction to sequential monte carlo methods*, in Sequential Monte Carlo Methods in Practice, 2001.

[47] S. DUANE, A. KENNEDY, B. J. PENDLETON, AND D. ROWETH, *Hybrid monte carlo*, Physics Letters B, 195 (1987), pp. 216–222.

[48] R. G. EVERITT AND P. A. ROWIŃSKA, *Delayed acceptance abc-smc*, 2020.

[49] I. FATKULLIN, G. KOVACIC, AND E. VANDEN-EIJNDEN, *Reduced dynamics of stochastically perturbed gradient flows*, Communications in Mathematical Sciences, 8 (2010), pp. 439 – 461.

[50] H. FEDERER, *Geometric Measure Theory*, Classics in Mathematics, Springer Berlin Heidelberg, 2014.

[51] J.-J. FORNERON AND S. NG, *A likelihood-free reverse sampler of the posterior distribution*, 2015.

[52] ——, *The abc of simulation estimation with auxiliary statistics*, 2017.

[53] Y. X. FU AND W. H. LI, *Estimating the age of the common ancestor of a sample of DNA sequences.*, Molecular Biology and Evolution, 14 (1997), pp. 195–199.

[54] C. FÜHRER AND B. J. LEIMKUHLER, *Numerical solution of differential-algebraic equations for constrained mechanical motion*, Numerische Mathematik, 59 (1991), pp. 55–69.

[55] T. FUNAKI AND H. NAGAI, *Degenerative convergence of diffusion process toward a submanifold by strong drift*, Stochastics and Stochastic Reports, 44 (1993), pp. 1–25.

[56] A. GELMAN, W. R. GILKS, AND G. O. ROBERTS, *Weak convergence and optimal scaling of random walk Metropolis algorithms*, The Annals of Applied Probability, 7 (1997), pp. 110 – 120.

[57] A. GELMAN AND C. PASARICA, *Adaptively scaling the metropolis algorithm using expected squared jumped distance*, Statistica Sinica, 20 (2010).

[58] M. GERBER, N. CHOPIN, AND N. WHITELEY, *Negative association, ordering and convergence of resampling methods*, 2020.

[59] M. GIROLAMI AND B. CALDERHEAD, *Riemann manifold langevin and hamiltonian monte carlo methods*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73 (2011), pp. 123–214.

[60] G. H. GOLUB AND V. PEREYRA, *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate*, SIAM Journal on Numerical Analysis, 10 (1973), pp. 413–432.

[61] P. GONG, Y. O. BASCIFTCI, AND F. OZGUNER, *A parallel resampling algorithm for particle filtering on shared-memory architectures*, 2012.

[62] M. M. GRAHAM, *Auxiliary variable markov chain monte carlo methods*, 2018.

[63] M. M. GRAHAM AND A. J. STORKEY, *Asymptotically exact inference in differentiable generative models*, 2017.

[64] M. M. GRAHAM, A. H. THIERY, AND A. BESKOS, *Manifold markov chain monte carlo methods for bayesian inference in diffusion models*, 2019.

[65] E. HAIRER, *Symmetric projection methods for differential equations on manifolds*, BIT, 40 (2000), pp. 726–734.

[66] E. HAIRER, C. LUBICH, AND G. WANNER, *Geometric numerical integration*, vol. 31 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, second ed., 2006.
Structure-preserving algorithms for ordinary differential equations.

[67] C. R. HARRIS, K. J. MILLMAN, S. J. VAN DER WALT, R. GOMMERS, P. VIRTANEN, D. COURNAPEAU, E. WIESER, J. TAYLOR, S. BERG, N. J. SMITH, R. KERN, M. PICUS, S. HOYER, M. H. VAN KERKWIJK, M. BRETT, A. HALDANE, J. F. DEL RÍO, M. WIEBE, P. PETERSON, P. GÉRARD-MARCHANT, K. SHEPPARD, T. REDDY, W. WECKESSER, H. ABBASI,

C. GOHLKE, AND T. E. OLIPHANT, *Array programming with numpy*, Nature, 585 (2020), p. 357–362.

[68] W. K. HASTINGS, *Monte carlo sampling methods using markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109.

[69] M. HIRD, S. LIVINGSTONE, AND G. ZANELLA, *A fresh take on 'barker dynamics' for mcmc*, 2021.

[70] M. HOFFMAN, A. RADUL, AND P. SOUNTSOV, *An adaptive-mcmc scheme for setting trajectory lengths in hamiltonian monte carlo*, in Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, A. Banerjee and K. Fukumizu, eds., vol. 130 of Proceedings of Machine Learning Research, PMLR, 13–15 Apr 2021, pp. 3907–3915.

[71] M. D. HOFFMAN AND A. GELMAN, *The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo*, Journal of Machine Learning Research, 15 (2014), pp. 1593–1623.

[72] M. D. HOFFMAN AND P. SOUNTSOV, *Tuning-free generalized hamiltonian monte carlo*, in Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, G. Camps-Valls, F. J. R. Ruiz, and I. Valera, eds., vol. 151 of Proceedings of Machine Learning Research, PMLR, 28–30 Mar 2022, pp. 7799–7813.

[73] A. M. HOROWITZ, *A generalized guided monte carlo algorithm*, Physics Letters B, 268 (1991), pp. 247–252.

[74] B. IKONOMOV AND M. U. GUTMANN, *Robust optimisation monte carlo*, 2020.

[75] L. JAY, *Symplectic partitioned runge–kutta methods for constrained hamiltonian systems*, SIAM Journal on Numerical Analysis, 33 (1996), pp. 368–387.

[76] G. S. KATZENBERGER, *Solutions of a Stochastic Differential Equation Forced Onto a Manifold by a Large Drift*, The Annals of Probability, 19 (1991), pp. 1587 – 1628.

[77] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, 2017.

[78] D. P. KINGMA AND M. WELLING, *Auto-encoding variational bayes*, 2014.

[79] I. KOBYZEV, S. J. PRINCE, AND M. A. BRUBAKER, *Normalizing flows: An introduction and review of current methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 43 (2021), p. 3964–3979.

[80] A. KONG, *A note on importance sampling using renormalized weights*, 1992.

[81] S. KRANTZ AND H. PARKS, *Geometric Integration Theory*, Cornerstones, Birkhäuser Boston, 2008.

[82] B. LEIMKUHLER AND S. REICH, *Simulating Hamiltonian Dynamics*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2005.

[83] B. J. LEIMKUHLER AND R. D. SKEEL, *Symplectic numerical integrators in constrained hamiltonian systems*, Journal of Computational Physics, 112 (1994), pp. 117–125.

[84] ——, *Symplectic numerical integrators in constrained hamiltonian systems*, Journal of Computational Physics, 112 (1994), pp. 117–125.

[85] T. LELIEVRE, M. ROUSSET, AND G. STOLTZ, *Langevin dynamics with constraints and computation of free energy differences*, 2011.

[86] T. LELIÈVRE, M. ROUSSET, AND G. STOLTZ, *Free Energy Computations*, IMPERIAL COLLEGE PRESS, 2010.

[87] T. LELIÈVRE, M. ROUSSET, AND G. STOLTZ, *Hybrid monte carlo methods for sampling probability measures on submanifolds*, 2019.

[88] T. LELIÈVRE, G. STOLTZ, AND W. ZHANG, *Multiple projection mcmc algorithms on submanifolds*, 2020.

[89] D. LEVY, M. D. HOFFMAN, AND J. SOHL-DICKSTEIN, *Generalizing hamiltonian monte carlo with neural networks*, 2018.

[90] J. LIU, *Metropolized independent sampling with comparisons to rejection sampling and importance sampling*, Statistics and Computing, 6 (2000).

[91] J. S. LIU, *Monte Carlo Strategies in Scientific Computing*, Springer Publishing Company, Incorporated, 2008.

[92] S. LIVINGSTONE AND G. ZANELLA, *The barker proposal: combining robustness and efficiency in gradient-based mcmc*, 2020.

[93] S. LIVINGSTONE AND G. ZANELLA, *The Barker Proposal: Combining Robustness and Efficiency in Gradient-Based MCMC*, Journal of the Royal Statistical Society Series B: Statistical Methodology, 84 (2022), pp. 496–523.

[94] M. LUDKIN AND C. SHERLOCK, *Hug and hop: a discrete-time, non-reversible markov chain monte-carlo algorithm*, 2019.

[95] P. B. MACKENZE, *An improved hybrid monte carlo method*, Physics Letters B, 226 (1989), pp. 369–371.

[96] F. MAIRE AND P. VANDEKERKHOVE, *On markov chain monte carlo for sparse and filamentary distributions*, 2018.

[97] P. MARJORAM, J. MOLITOR, V. PLAGNOL, AND S. TAVARÉ, *Markov chain monte carlo without likelihoods*, Proceedings of the National Academy of Sciences, 100 (2003), pp. 15324–15328.

[98] E. MEEDS, R. LEENDERS, AND M. WELLING, *Hamiltonian abc*, 2015.

[99] E. MEEDS AND M. WELLING, *Optimization monte carlo: Efficient and embarrassingly parallel likelihood-free inference*, 2015.

[100] N. METROPOLIS, A. W. ROSENBLUTH, M. N. ROSENBLUTH, A. H. TELLER, AND E. TELLER, *Equation of state calculations by fast computing machines*, The Journal of Chemical Physics, 21 (1953), pp. 1087–1092.

[101] I. MURRAY AND M. M. GRAHAM, *Pseudo-marginal slice sampling*, 2016.

[102] L. M. MURRAY, A. LEE, AND P. E. JACOB, *Parallel resampling in the particle filter*, 2015.

[103] P. NEAL, *Efficient likelihood-free bayesian computation for household epidemics*, Statistics and Computing, 22 (2012), pp. 1239–1256.

[104] R. M. NEAL, *An improved acceptance procedure for the hybrid monte carlo algorithm*, Journal of Computational Physics, 111 (1994), pp. 194–203.

[105] ——, *Non-reversibly updating a uniform [0,1] value for metropolis accept/reject decisions*, 2020.

[106] Y. NESTEROV, *Primal-dual subgradient methods for convex problems*, Mathematical Programming, 120 (2009), pp. 221–259.

[107] J. ORBAN AND J. P. RYCKAERT, *Methods in molecular dynamics, report on cecam workshop*, 1974.

[108] A. B. OWEN, *A randomized halton algorithm in r*, 2017.

[109] G. PAPAMAKARIOS, E. NALISNICK, D. J. REZENDE, S. MOHAMED, AND B. LAKSHMINARAYANAN, *Normalizing flows for probabilistic modeling and inference*, 2021.

[110] P. H. PESKUN, *Optimum monte-carlo sampling using markov chains*, Biometrika, 60 (1973), pp. 607–612.

[111] G. W. PETERS, Y. FAN, AND S. A. SISSON, *On sequential monte carlo, partial rejection control and approximate bayesian computation*, 2009.

[112] D. PRANGLE, *gk: An r package for the g-and-k and generalised g-and-h distributions*, 2017.

[113] D. PRANGLE, *Distilling importance sampling*, 2020.

[114] D. PRANGLE, R. G. EVERITT, AND T. KYPRAIOS, *A rare event approach to high-dimensional approximate bayesian computation*, Statistics and Computing, 28 (2018), pp. 819–834.

[115] J. K. PRITCHARD, M. T. SEIELSTAD, A. PEREZ-LEZAUN, AND M. W. FELDMAN, *Population growth of human Y chromosomes: a study of Y chromosome microsatellites.*, Molecular Biology and Evolution, 16 (1999), pp. 1791–1798.

[116] A. RADUL, B. PATTON, D. MACLAURIN, M. D. HOFFMAN, AND R. A. SAUROUS, *Automatically batching control-intensive programs for modern accelerators*, 2020.

[117] G. D. RAYNER AND H. L. MACGILLIVRAY, *Numerical maximum likelihood estimation for the g-and-k and generalized g-and-h distributions*, Statistics and Computing, 12 (2002), pp. 57–75.

[118] S. REICH, *Sympletic integration of constrained hamiltonian systems by runge-kutta methods*, Technical Report 93-13, (1993).

[119] J. A. RICE, *Mathematical Statistics and Data Analysis.*, Belmont, CA: Duxbury Press., third ed., 2006.

[120] C. ROBERT AND G. CASELLA, *Monte Carlo statistical methods*, Springer Verlag, 2004.

[121] D. B. RUBIN, *Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician*, The Annals of Statistics, 12 (1984), pp. 1151 – 1172.

[122] D. RUDOLF AND B. SPRUNGK, *Robust random walk-like metropolis-hastings algorithms for concentrating posteriors*, 2022.

[123] J.-P. RYCKAERT, G. CICCOTTI, AND H. J. BERENDSEN, *Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes*, Journal of Computational Physics, 23 (1977), pp. 327–341.

[124] J. SANZ-SERNA AND M. CALVO, *Numerical Hamiltonian Problems*, Dover Books on Mathematics, Dover Publications, 2018.

[125] U. SHARMA AND W. ZHANG, *Nonreversible sampling schemes on submanifolds*, SIAM Journal on Numerical Analysis, 59 (2021), pp. 2989–3031.

[126] C. SHERLOCK AND A. H. THIERY, *A discrete bouncy particle sampler*, 2021.

[127] S. SISSON, Y. FAN, AND M. BEAUMONT, *Handbook of Approximate Bayesian Computation*, Chapman & Hall/CRC Handbooks of Modern Statistical Methods, CRC Press, 2018.

[128] S. A. SISSON, Y. FAN, AND M. A. BEAUMONT, *Overview of approximate bayesian computation*, 2018.

[129] S. A. SISSON, Y. FAN, AND M. M. TANAKA, *Sequential monte carlo without likelihoods*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 1760–1765.

[130] ——, *Correction for sisson et al., sequential monte carlo without likelihoods*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 16889–16889.

[131] J. SONG, S. ZHAO, AND S. ERMON, *A-nice-mc: Adversarial training for mcmc*, 2018.

[132] P. SOUNTSOV AND M. D. HOFFMAN, *Focusing on difficult directions for learning hmc trajectory lengths*, 2022.

[133] J. SPALL, *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation*, IEEE Transactions on Automatic Control, 37 (1992), pp. 332–341.

[134] E. STEIN AND R. SHAKARCHI, *Real Analysis: Measure Theory, Integration, and Hilbert Spaces*, Princeton University Press, 2009.

[135] S. TAVARÉ, D. J. BALDING, R. C. GRIFFITHS, AND P. DONNELLY, *Inferring coalescence times from dna sequence data*, Genetics, 145 (1997), pp. 505–518.

[136] S. D. TEAM, *Stan modeling language users guide and reference manual*, 2023.

[137] T. T. TEAM, *Tensorflow probability*, 2018-2019.

[138] A. THIN, Y. JANATI, S. L. CORFF, C. OLLION, A. DOUCET, A. DURMUS, E. MOULINES, AND C. ROBERT, *Neo: Non equilibrium sampling on the orbit of a deterministic transform*, 2021.

[139] L. TIERNEY, *Markov Chains for Exploring Posterior Distributions*, The Annals of Statistics, 22 (1994), pp. 1701–1728.

[140] L. TIERNEY, *A note on Metropolis-Hastings kernels for general state spaces*, The Annals of Applied Probability, 8 (1998), pp. 1 – 9.

[141] W. VAN GUNSTEREN AND H. BERENDSEN, *Algorithms for macromolecular dynamics and constraint dynamics*, Molecular Physics, 34 (1977), pp. 1311–1327.

[142] G. VAN ROSSUM AND F. L. DRAKE JR, *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[143] E. VANDEN-EIJNDEN AND G. CICCOTTI, *Second-order integrators for langevin equations with holonomic constraints*, Chemical Physics Letters, 429 (2006), pp. 310–316.

[144] A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. KAISER, AND I. POLOSUKHIN, *Attention is all you need*, 2023.

[145] L. VERLET, *Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules*, Phys. Rev., 159 (1967), pp. 98–103.

[146] J. VOGRINC, S. LIVINGSTONE, AND G. ZANELLA, *Optimal design of the Barker proposal and other locally balanced Metropolis–Hastings algorithms*, Biometrika, 110 (2022), pp. 579–595.

[147] G. WEISS AND A. VON HAESELER, *Inference of Population History Using a Likelihood Approach*, Genetics, 149 (1998), pp. 1539–1546.

[148] M. WELLING AND Y. W. TEH, *Bayesian learning via stochastic gradient langevin dynamics*, in Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, Madison, WI, USA, 2011, Omnipress, p. 681–688.

[149] S. N. WOOD, *Statistical inference for noisy nonlinear ecological dynamic systems*, Nature, 466 (2010), pp. 1102–1104.

[150] C. WU, J. STOEHR, AND C. P. ROBERT, *Faster hamiltonian monte carlo by learning leapfrog scale*, 2019.

[151] J. YANG, G. O. ROBERTS, AND J. S. ROSENTHAL, *Optimal scaling of random-walk metropolis algorithms on general target distributions*, Stochastic Processes and their Applications, 130 (2020), pp. 6094–6132.

[152] J. YANG, K. ŁATUSZYŃSKI, AND G. O. ROBERTS, *Stereographic markov chain monte carlo*, 2022.

[153] E. ZAPPA, M. HOLMES-CERFON, AND J. GOODMAN, *Monte carlo on manifolds: Sampling densities and integrating functions*, Communications on Pure and Applied Mathematics, 71 (2018), pp. 2609–2647.

[154] C. ZHANG, *On the improvements and innovations of monte carlo methods*. `https://research-information.bris.ac.uk/ws/portalfiles/portal/347663871/Final_Copy_2022_10_13_Zhang_C_PhD.pdf`, 2022.

[155] W. ZHANG, *Ergodic sdes on submanifolds and related numerical sampling schemes*, 2017.

[156] ZIYU WANG, S. MOHAMED, AND N. DE FREITAS, *Adaptive hamiltonian and riemann manifold monte carlo samplers*, 2013.

Acronyms

ABC Approximate Bayesian Computation. xv, 23–25, 43–45, 50, 51, 143, 191

AHMC Adaptive HMC. 120, 121

BIP Bayesian Inverse Problems. 23, 39, 49, 50

C-HMC Constrained HMC. 30, 40, 49, 50, 137

C-RWM Constrained RWM. xv, xvi, 27, 28, 30, 32, 39–41, 44–47, 137, 163

DAEs Differential Algebraic Equations. 48, 136

DRY Don't Repeat Yourself. 2

DW Deterministic Walk. 68, 74, 81, 95, 176

EHMC Empirical HMC. 115, 122

ESJD Expected Squared Jump Distance. xiii, xvi, xviii, xix, 6, 7, 21, 22, 36, 37, 63, 72, 97–100, 103–106, 108, 110–113, 120, 121, 123, 130, 178

ESS Effective Sample Size. xvi, xvii, xix, 12, 40, 41, 44, 46, 47, 50, 64, 67, 69–72, 76, 93–95, 123, 127–129, 163, 164

GC-HMC Generalized Constrained HMC. 137, 138

GHMC Generalised HMC. 119

GHUMS Gibbs Hug Markov Snippet. xviii, xix, 53, 90, 91, 93, 95–100, 102–109, 113–115

HMC Hamiltonian Monte Carlo. xvi, 8, 27, 30, 39–41, 50, 53, 108, 112, 114, 115, 118–120, 137, 163

IID independent and identically-distributed. 4, 7, 12, 67, 120

IR Importance Resampling. 17

IS Importance Sampling. 10–15, 18–20, 131

LHS Left Hand Side. 155, 184

LV Lotka-Volterra. xvi, 45, 47

MALA Metropolis-Adjusted Langevin Algorithm. 91, 92

MCMC Markov Chain Monte Carlo. 4, 5, 13, 50, 112, 118, 119

MD Molecular Dynamics. 23, 27, 48, 49

MH Metropolis-Hastings. 5–10, 21, 22, 27, 28, 32, 33, 48, 51, 64, 68, 69, 74, 82, 88, 89, 91, 92, 97, 104, 118, 119, 121–123, 130, 137

NHUG Normal HUG. xviii, 88–92, 94, 95, 97, 102–104, 107, 108, 115

NUTS No-U-Turn Sampler. 119, 120, 122

ODEs Ordinary Differential Equations. 3, 8, 9, 135

PISA Pushforward Importance Sampling. xvi, xvii, 53, 68–70, 72–74, 76, 78, 79, 81, 82, 116, 177

PVR Percentage of Variance Reduction. 76

RHMC Randomised HMC. 120

RHS Right Hand Side. 154, 177

RM-HMC Riemann-Manifold HMC. xvi, 39, 49

RMSE Root Mean Squared Error. xvii, 64, 72, 73, 76, 97

RWM Random Walk Metropolis. xv, xvi, 7, 8, 21, 25–27, 30, 41–46, 49, 50, 68, 95, 103

SDEs Stochastic Differential Equations. 45, 48, 49

SGA-HMC Stochastic Gradient Ascent HMC. 121

SIMD Single Instruction, Multiple Data. 120, 122

SIR Sequential Importance Resampling. 17

SMC Sequential Monte Carlo. xiii, xvi–xviii, 1, 13, 15–19, 21, 22, 40–43, 50, 51, 53, 56, 58, 60, 64, 65, 68–70, 72–74, 76, 80, 81, 83, 86–90, 93–99, 102–107, 118, 122, 123, 169, 178, 180, 187

SMC-RWM Sequential Monte Carlo with Random Walk Metropolis kernel. 68, 69, 94

TEN Total ESJD-NORM. 113

THUG Tangential HUG. xiii, xv, xvi, xviii, xix, 1, 25, 26, 30–35, 37, 39–49, 88–91, 94–97, 102–104, 107–110, 113, 115, 123, 138, 152, 153, 159, 163