Concave-Convex PDMP-based sampling

Matthew Sutton
Centre for Data Science, Queensland University of Technology
and
Paul Fearnhead
Department of Mathematics and Statistics, Lancaster University*

May 12, 2023

Abstract

Recently non-reversible samplers based on simulating piecewise deterministic Markov processes (PDMPs) have shown potential for efficient sampling in Bayesian inference problems. However, there remains a lack of guidance on how to best implement these algorithms. If implemented poorly, the computational costs of simulating event times can out-weigh the statistical efficiency of the non-reversible dynamics. Drawing on the adaptive rejection literature, we propose the concave-convex adaptive thinning approach for simulating a piecewise deterministic Markov process, which we call CC-PDMP. This approach provides a general guide for constructing bounds that may be used to facilitate PDMP-based sampling. A key advantage of this method is its additive structure - adding concave-convex decompositions yields a concave-convex decomposition. This makes the construction of bounds modular, as given a concave-convex decomposition for a class of likelihoods and a family of priors, they can be combined to construct bounds for the posterior. We show that constructing our bounds is simple and leads to computationally efficient thinning. Our approach is well suited to local PDMP simulation where conditional independence of the target can be exploited for potentially huge computational gains. We provide an R package and compare with existing approaches to simulating events in the PDMP literature.

1 Introduction

Monte Carlo methods based on continuous-time Markov processes have shown potential for efficient sampling in Bayesian inference challenges (Goldman and Singh, 2021; Fearnhead *et al.*, 2018). These new sampling algorithms are based on simulating piecewise-deterministic Markov processes (PDMPs). Such processes evolve deterministically between random event times with possibly random dynamics at event times (see Davis, 1993, for an introduction to PDMPs). Monte Carlo methods based on simulating a PDMP with a desired invariant distribution were first proposed in the computational physics literature where they were motivated as examples of non-reversible Markov processes (Peters and de With, 2012; Michel *et al.*, 2014). These ideas transitioned to the statistics community as an alternative to traditional Markov chain Monte Carlo (MCMC) samplers. Popular algorithms in this development include the Bouncy Particle Sampler (Bouchard-Côté *et al.*, 2018) and the Zig-Zag sampler (Bierkens *et al.*, 2019) amongst others (Vanetti *et al.*, 2017; Wu and Robert, 2020; Michel *et al.*, 2020; Bierkens *et al.*, 2020).

There is substantial theoretical and empirical evidence to support the claim that non-reversible MCMC samplers offer more efficient sampling as they reduce the chance of the sampler moving back to areas of the state space that have recently been visited (Diaconis *et al.*, 2000; Bierkens and Roberts, 2017; Bouchard-Côté *et al.*, 2018). Samplers based on PDMPs simulate from a

*This research was supported by EPSRC grant EP/R018561 and EP/R034710/1

density $\pi(\boldsymbol{\theta})$ by introducing a velocity component $\boldsymbol{v}$ which is used to evolve the position of the state, $\boldsymbol{\theta}$, with deterministic dynamics. At stochastic event times the velocity component is updated using a Markov kernel after which the process continues with the new velocity. The main challenge to implementing a PDMP-based sampler is the simulation of the random event times. This involves simulating the next event time in a time-inhomogeneous Poisson process with rate function that depends on the current position of the sampler.

Simulation of event times in a Poisson process is often facilitated through a method known as thinning (Lewis and Shedler, 1979). The aim of this paper is to provide a general framework to aid practitioners in implementing PDMP-based samplers. Specifically we introduce the concave-convex thinning approach to facilitate simulating a PDMP-based sampler, which we call CC-PDMP. This method can be applied whenever the rate function of the PDMP can be decomposed into the sum of concave and convex functions, and can be used to facilitate thinning from polynomial rate functions, allowing for broad applicability of the method. We discuss the efficiency of the method compared to alternative approaches in the literature for exact sampling of event times.

Related to our work is the concave-convex adaptive rejection sampler (CC-ARS) of Görür and Teh (2011) which uses the same upper-bounding approach to construct bounds on the log density within a rejection sampler that draws independent samples from univariate density functions. In contrast, by applying this technique within PDMP sampling, the bounds are constructed on the gradient of the log density (i.e. the rate) and used within the non-reversible Markov process framework to generate samples from multivariate densities. Moreover, the CC-PDMP framework allows for both subsampling and local updating schemes to facilitate these methods for high dimensional sampling. A short comparison between concave-convex adaptive rejection sampling and CC-PDMP for sampling from a univariate density is given in Appendix C.

Some authors have proposed using automatic but approximate methods for simulating a PDMP-based sampler. Among these approaches, Cotter *et al.* (2020) propose numerical integration and root-finding algorithms to facilitate simulation. Others have considered using approximate local bounds which can be simulated exactly (Goldman and Singh, 2021; Goan and Fookes, 2021; Pakman *et al.*, 2017). Both of these approaches sacrifice exact sampling of the posterior and involve a trade-off between the computational cost and the approximation of the sampling distribution.

The paper is organised as follows. In Section 2, we introduce the technical details of simulating from a PDMP-based sampler and the details of some popular samplers. Section 3 reviews the literature on thinning for PDMP-based samplers. We introduce the concave-convex PDMP approach for adaptive thinning in Section 4 and 5. Empirical evaluation of CC-PDMP, and comparison to existing methods, is given in Section 6. We conclude with discussion of limitations and extensions that are possible for the method. Code for implementing our method and replicating our results is available at `https://github.com/matt-sutton/ccpdmp`.

2 Sampling using a PDMP

2.1 Piecewise deterministic Markov processes

Before we explore PDMP-based samplers, we first review PDMPs. A PDMP is a stochastic process, and we will denote its state at time t by $\boldsymbol{z}_t$. There are three key components that define the dynamics of a PDMP:

1. A set of deterministic dynamics defined as $\boldsymbol{z}_{t+s} = \psi(\boldsymbol{z}_t, s)$.
2. Event times that are driven by an inhomogeneous Poisson process with rate $\lambda(\boldsymbol{z}_t)$.
3. A Markov transition kernel $q(\boldsymbol{z}|\boldsymbol{z}_{t-})$ where $\boldsymbol{z}_{t-} := \lim_{s \uparrow t} \boldsymbol{z}_s$.

A PDMP with these specifications will evolve according to its deterministic dynamics between event times and at event times will update according to the Markov transition kernel $q(\boldsymbol{z}|\boldsymbol{z}_{t-})$.

The event times must be simulated from a Poisson process with rate $\lambda(\boldsymbol{z}_t)$, which depends on the state of the process at time t. The result of simulating this process is a PDMP skeleton $\{(t_k, \boldsymbol{z}_{t_k})\}_{k=1}^n$ where for $k = 1, \ldots, n$, t_k denotes the kth event time and we have stored the state of the process immediately after this event time, $\boldsymbol{z}_{t_k}$. Given this skeleton it is possible to fill in the values of the state between the events using the deterministic dynamics.

2.2 PDMP-based samplers

Assume we wish to construct a PDMP process to sample from a target distribution of interest, $\pi(\boldsymbol{\theta})$. Current PDMP-based samplers are defined on an augmented space $\boldsymbol{z} \in \mathcal{E}$, where $\mathcal{E} = \mathcal{X} \times \mathcal{V}$, that can be viewed as having a position, $\boldsymbol{\theta}_t$, and a velocity, $\boldsymbol{v}_t$. As described below, the dynamics of the PDMP can be chosen so that the PDMP's invariant distribution has the form $\nu(\boldsymbol{z}) = \pi(\boldsymbol{\theta})p(\boldsymbol{v}|\boldsymbol{\theta})$, for some conditional distribution of the velocities, $p(\boldsymbol{v}|\boldsymbol{\theta})$. As a result, the $\boldsymbol{\theta}$-marginal of the invariant distribution is the target distribution we wish to sample from. We will consider target distributions of the form

$$\pi(\boldsymbol{\theta}) \propto \exp(-U(\boldsymbol{\theta}))$$

where $\boldsymbol{\theta} \in \mathbb{R}^p$ and $U(\boldsymbol{\theta})$ is the associated potential. An important feature of PDMP samplers is that they need to know the target distribution only up to a constant of proportionality.

If we simulate the PDMP, and the process converges to stationarity, then we can use the simulated skeleton of the PDMP to estimate expectations with respect to the target distribution in two ways. Assume we are interested in a Monte Carlo estimate of $\int g(\boldsymbol{z})\nu(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$. The first estimator averages over the continuous time path of the PDMP

$$\frac{1}{t_n - t_1} \sum_{k=1}^{n-1} \int_{t_k}^{t_{k+1}} g(\psi(\boldsymbol{z}_{t_k}, s))ds.$$

This requires that the integral of $g(\psi(\boldsymbol{z}_{t_k}, s))$ with respect to s may be computed. The second, simpler, approach is to "discretise" the PDMP path. This involves taking an integer $M > 0$, defining $s = (t_n - t_1)/M$, calculating the value of the state at discrete-time points, $\boldsymbol{z}_{t_1+s}, \ldots, \boldsymbol{z}_{t_1+Ms}$, and then using the standard Monte Carlo estimator

$$\frac{1}{M} \sum_{j=1}^{M} g(\boldsymbol{z}_{t_1+js}).$$

The points $\boldsymbol{z}_{t_1+js}$ may be found by evolving the process along its deterministic dynamics for the appropriate event time interval.

2.3 Bouncy particle sampler

The Bouncy Particle Sampler (Bouchard-Côté *et al.*, 2018) takes the velocity space $\mathcal{V}$ to be either $\mathbb{R}^p$ or the unit sphere $\mathcal{S}^{p-1}$ and targets an invariant distribution $\nu(\boldsymbol{z}) = \pi(\boldsymbol{\theta})p(\boldsymbol{v})$ where $p(\boldsymbol{v})$ is either the standard p-dimensional Normal distribution, or a uniform distribution on $\mathcal{S}^{p-1}$. This sampler evolves a PDMP with linear deterministic dynamics $\psi((\boldsymbol{\theta}_t, \boldsymbol{v}_t), s) = (\boldsymbol{\theta}_t + s\boldsymbol{v}_t, \boldsymbol{v}_t)$ between events. The canonical event rate is given by

$$\lambda^c(\boldsymbol{z}_t) = \max\left\{0, \langle \boldsymbol{v}_t, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_t)\rangle\right\}.$$

At an event time t generated according to this rate the velocity is updated as $\boldsymbol{v}_t = R_{\boldsymbol{\theta}_{t-}}(\boldsymbol{v}_{t-})$ where

$$R_{\boldsymbol{\theta}}(\boldsymbol{v}) = \boldsymbol{v} - 2\frac{\langle \boldsymbol{v}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\rangle}{\|\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\|^2} \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}).$$

This update can be interpreted as reflecting the velocity off the hyperplane tangential to the gradient of $U(\boldsymbol{\theta})$. Additionally, at random times t an additional event occurs according to a homogeneous Poisson process with rate $\lambda^{\text{ref}} > 0$ in which the velocity is refreshed, drawing $\boldsymbol{v}_t \sim p(\boldsymbol{v})$. Refreshment ensures that the Bouncy Particle Sampler samples the invariant distribution and does not get "stuck" on contours of the potential (Bouchard-Côté *et al.*, 2018). Formally the Bouncy Particle Sampler has event rate $\lambda(\boldsymbol{z}_t) = \lambda^c(\boldsymbol{z}_t) + \lambda^{\text{ref}}$ with Markov transition kernel

$$q(\boldsymbol{z}|\boldsymbol{z}_{t-}) = \frac{\lambda^c(\boldsymbol{z}_{t-})}{\lambda(\boldsymbol{z}_{t-})}\delta_{\boldsymbol{\theta}_{t-}}(\boldsymbol{\theta})\delta_{R_{\boldsymbol{\theta}_{t-}}(\boldsymbol{v}_{t-})}(\boldsymbol{v}) + \frac{\lambda^{\text{ref}}}{\lambda(\boldsymbol{z}_{t-})}\delta_{\boldsymbol{\theta}_{t-}}(\boldsymbol{\theta})p(\boldsymbol{v}),$$

where $\delta_a(A)$ denotes the Dirac delta function with point mass at a. The Bouncy Particle Sampler is an example of a *global PDMP* as all components of the velocity $\boldsymbol{v}$ are changed according to the Markov transition kernel.

2.4 Zig-Zag sampler

The Zig-Zag sampler (Bierkens *et al.*, 2019, 2021a) takes the space of velocities, $\mathcal{V}$, to be $\{-1, 1\}^p$ with $p(\boldsymbol{v}) = 2^{-p}$ uniform on this space. The Zig-Zag sampler also uses linear dynamics $\psi((\boldsymbol{\theta}_t, \boldsymbol{v}_t), s) = (\boldsymbol{\theta}_t + s\boldsymbol{v}_t, \boldsymbol{v}_t)$ between events, but unlike the BPS only a single element of the velocity vector is updated at event times. The overall Zig-Zag process can be viewed as a composition of p *local event rates* where each event rate has a corresponding Markov transition kernel which operates on a single component of the velocity. Specifically, the i-th component of the velocity is updated with local event rate

$$\lambda_i(\boldsymbol{z}_t) = \max\{0, v_i\partial_{\theta_i}U(\boldsymbol{\theta}_t)\}$$

for $i = 1, ..., p$, where ∂_{θ_i} denotes the partial derivative of $U(\boldsymbol{\theta}_t)$ with respect to θ_i. Simulating an event time in this local framework consists of simulating an event time for each local event rate and then taking the minimum time as the event time t for the overall process. At an event time t triggered by event rate λ_i the velocity is updated as $\boldsymbol{v}_t = F_i(\boldsymbol{v}_{t-})$ where

$$F_i(\boldsymbol{v}) = (v_1, \ldots, v_{i-1}, -v_i, v_{i+1}, \ldots, v_p)^T.$$

Once the velocity is updated for component i local event times must be re-simulated only for rates $\lambda_j(\boldsymbol{z}_t)$ which are functions of θ_i. This fact can induce massive computational savings when there is conditional independence between the components of $\boldsymbol{\theta}$. To see this, consider the extreme case where the target is fully independent across dimensions. In this case each event rate, $\lambda_i(\boldsymbol{z}_t)$, will only depend on the i-th components of $\boldsymbol{\theta}_t$ and $\boldsymbol{v}_t$. Thus at each event, it is only the time of the next event for the component whose velocity changes that needs to be recalculated. The computational cost of simulation in this case will scale as $O(1)$ per event as opposed to $O(d)$ for the global Bouncy Particle Sampler.

2.5 Global and local PDMP methods

As described above, we can divide PDMP samplers into two main classes: *global methods* and *local methods*. These PDMPs differ in how they operate on the velocity vector when an event is triggered. Global methods are PDMP-based methods where the transition kernel acts on the entire velocity vector. By contrast, local methods are defined so that the transition kernel only affects a subset of the velocity. In this section we will give a general algorithm for constructing a local PDMP-based sampler (though see also Bouchard-Côté *et al.*, 2018). Let $\boldsymbol{v}^{[S]}$ denote the sub-vector of $\boldsymbol{v}$ with elements indexed by $S \subseteq \{1, \ldots, p\}$ and $\boldsymbol{v}^{[-S]}$ denote the sub-vector of $\boldsymbol{v}$ without the elements indexed in S. For some set S define the local kernel as one that satisfies

$$q^{[S]}_{\boldsymbol{\theta}}(\boldsymbol{v}|\boldsymbol{v}_{t-}) = \delta_{\boldsymbol{v}^{[-S]}_{t-}}\left(\boldsymbol{v}^{[-S]}\right) q^{[S]}_{\boldsymbol{\theta}}(\boldsymbol{v}|\boldsymbol{v}_{t-}),$$

that is, $q_{\boldsymbol{\theta}}^{[S]}(\boldsymbol{v}|\boldsymbol{v}_{t-})$ only updates the sub-vector $\boldsymbol{v}^{[S]}$. Suppose we have a partition of the components of $\theta_1, ..., \theta_p$ given by $\mathcal{S} = \{S_1, S_2, \ldots, S_F\}$. Subset S_f of $\boldsymbol{v}$ is updated with local kernel $q_{\boldsymbol{\theta}}^{[S_f]}(\boldsymbol{v}|\boldsymbol{v}_{t-})$ and associated event rate

$$\lambda_f(\boldsymbol{z}_t) = \max\left\{0, \langle \boldsymbol{v}^{[S_f]}, \nabla_{\boldsymbol{\theta}^{[S_f]}} U(\boldsymbol{\theta}_t)\rangle\right\},$$

for $f = 1, \ldots, F$. Simulating a local PDMP with this structure involves simulating an event time for each of the F rates and applying the local transition kernel corresponding to the smallest simulated event time. For the partition $\mathcal{S}$ define $\mathcal{N} = \{N_1, \ldots, N_F\}$ with

$$N_f = \{m \in \{1, \ldots, F\} \mid \boldsymbol{\theta}^{[S_m]} \not\perp \boldsymbol{\theta}^{[S_f]}\}$$

for $f = 1, \ldots, F$ where $\boldsymbol{\theta}^{[S_m]} \not\perp \boldsymbol{\theta}^{[S_f]}$ denotes that the rate of events associated with S_m depends on some element of $\boldsymbol{\theta}^{[S_f]}$. When an event is triggered for a local rate f the velocity $\boldsymbol{v}^{[S_f]}$ is updated and new event times are simulated for rates which depend on the value of one or more θ_i for $i \in S_f$. This simulation process is summarised in Algorithm 1.

In this framework the Zig-Zag sampler is a local PDMP with partition $\mathcal{S} = \{\{1\}, \{2\}, \ldots, \{p\}\}$ and local kernels $q_{\boldsymbol{\theta}}^{[S_f]}(\boldsymbol{v}|\boldsymbol{v}_{t-}) = \delta_{F_f(\boldsymbol{v}_{t-})}(\boldsymbol{v})$. The local Bouncy Particle Sampler uses kernel $q_{\boldsymbol{\theta}}^{[S]}(\boldsymbol{v}|\boldsymbol{v}_{t-}) = \delta_{\boldsymbol{v}_*}(\boldsymbol{v})$ where $\boldsymbol{v}_*^{[-S]} = \boldsymbol{v}_{t-}^{[-S]}$ and $\boldsymbol{v}_*^{[S]} = R_{\boldsymbol{\theta}}^{[S]}(\boldsymbol{v}_{t-}^{[S]})$ with

$$R_{\boldsymbol{\theta}}^{[S]}(\boldsymbol{v}^{[S]}) = \boldsymbol{v}^{[S]} - 2\frac{\langle \boldsymbol{v}^{[S]}, \nabla_{\boldsymbol{\theta}^{[S]}} U(\boldsymbol{\theta})\rangle}{\|\nabla_{\boldsymbol{\theta}^{[S]}} U(\boldsymbol{\theta})\|^2} \nabla_{\boldsymbol{\theta}^{[S]}} U(\boldsymbol{\theta}).$$

If there is a single factor $\mathcal{S} = \{S\}$ containing all indices $S = \{1, \ldots, p\}$ we recover the global Bouncy Particle Sampler.

Algorithm 1: Simulating a PDMP with local structure
Inputs: $t_1 = 0$, $\boldsymbol{\theta}_{t_1}, \boldsymbol{v}_{t_1}$, factorisation $(\mathcal{S}, \mathcal{N})$ with $\mathcal{S} = \{S_1, S_2, \ldots, S_F\}$ and $\mathcal{N} = \{N_1, N_2, \ldots, N_F\}$.
Sample τ_f for $f = 1, ..., F$ where

$$\mathbb{P}(\tau_f \geq t) = \exp\left(-\int_0^t \lambda_f(\boldsymbol{\theta}_0 + \boldsymbol{v}_0 s, \boldsymbol{v}_0)\mathrm{d}s\right)$$

For $k = 1, \ldots, n$

(a) Let $f^* = \mathrm{argmin}_f(\tau_f)$ and update $t_{k+1} = t_k + \tau_{f^*}$

(b) Update states: $\boldsymbol{\theta}_{t_{k+1}} = \boldsymbol{\theta}_{t_k} + \tau_{f^*}\boldsymbol{v}_{t_k}$

(c) Update velocity: $\boldsymbol{v}_{t_{k+1}} = q_{\boldsymbol{\theta}_{t_{k+1}}}^{[S_{f^*}]}(\boldsymbol{v}_{t_k}|\boldsymbol{v})$

(d) For $f \in N_{f^*}$ update times, i.e. resample τ_f where

$$\mathbb{P}(\tau_f \geq t) = \exp\left(-\int_0^t \lambda_f(\boldsymbol{\theta}_{t_{k+1}} + \boldsymbol{v}_{t_{k+1}} s, \boldsymbol{v}_{t_{k+1}})\mathrm{d}s\right)$$

Outputs: PDMP skeleton $\{(t_k, \boldsymbol{\theta}_{t_k}, \boldsymbol{v}_{t_k})\}_{k=1}^n$

3 Algorithms for event-time simulation

Simulating the first event time, τ, from a Poisson process with event rate $\lambda(t)$ is equivalent to simulating $u \sim \text{Uniform}[0,1]$ and solving

$$\tau = \text{argmin}_t \left\{ -\log(u) = \int_0^t \lambda(\boldsymbol{z}_s) ds \right\}.$$

If the event rate is sufficiently simple this can be done exactly. For example if the rate is constant or linear, then there are analytical solutions for τ in terms of u. If the rate function is convex a solution may be found using numerical methods (Bouchard-Côté *et al.*, 2018). For more complex functions, the rate can be simulated by a process known as thinning. This process makes use of Theorem 1.

Theorem 1 (Lewis and Shedler (1979))**.** *Let $\lambda(t)$ and $\bar{\lambda}(t)$ be continuous functions where $\lambda(t) \leq \bar{\lambda}(t)$ for all $0 \leq t_{\max}$. Let $\tau_1, \tau_2, ..., \tau_n$ be event times of the Poisson process with rate $\bar{\lambda}(t)$ over the interval $[0, t_{\max})$. For $i = 1, ..., n$ retain the event times τ_i with probability $\lambda(\tau_i)/\bar{\lambda}(\tau_i)$ to obtain a set of event times from a non-homogeneous Poisson process with rate λ over the interval $(0, t_{\max}]$.*

The efficiency of simulating via thinning depends on how tightly the rate $\bar{\lambda}$ upper-bounds λ and how costly it is to simulate from $\bar{\lambda}$. Another key tool for enabling the simulation of event times is known as superposition and the general process is outlined in Theorem 2.

Theorem 2 (Kingman (1992))**.** *Suppose that $\Lambda_1, ..., \Lambda_n$ are a set of independent Poisson processes with rates $\lambda_1(t), ..., \lambda_n(t)$ with first arrival times $\tau^{[1]}, ..., \tau^{[n]}$. The Poisson process with rate $\lambda(t) = \sum_{i=1}^n \lambda_i(t)$ may be simulated by returning the first arrival time as $\tau = \min_{i=1,...,n} \tau^{[i]}$.*

Both superposition and thinning provide useful tools in the simulation of event times from a non-homogeneous Poisson process. However, constructing the upper-bounds required for thinning is largely an ad hoc and time consuming process for the practitioner. A common desire for a practitioner is to reuse simulation knowledge. For example if an event rate can be written as the sum of several sub-event rates, that can be exactly simulated, ideally this knowledge should be used to simulate from the sum. We refer to this class of event rates as *additive* rates. Specifically, we will refer to a Poisson process with a rate

$$\lambda(\boldsymbol{z}_t) = \max\{0, f_1(t) + f_2(t)\}$$

as process with an additive rate. The superposition method from Theorem 2 facilitates additive event-time simulation by thinning using an upper-bounding event rate

$$\bar{\lambda}(t) = \max\{0, f_1(t)\} + \max\{0, f_2(t)\},$$

which satisfies $\lambda(\boldsymbol{z}_t) \leq \bar{\lambda}(t)$. We can simulate the first event from a process with this rate as $\tau = \min(\tau_1, \tau_2)$, where τ_1 is simulated with rate $\bar{\lambda}_1(t) = \max\{0, f_1(t)\}$ and τ_2 with rate $\bar{\lambda}_2(t) = \max\{0, f_2(t)\}$. This simulated time will be accepted with probability

$$\frac{\max\{0, f_1(\tau) + f_2(\tau)\}}{\max\{0, f_1(\tau)\} + \max\{0, f_2(\tau)\}},$$

which will be one when both $f_1(\tau) > 0$ and $f_2(\tau) > 0$. This procedure is useful since it allows the re-use of thinning procedures for f_1 and f_2. Consequently a practitioner can build up to simulating from a complex rate by combining smaller simpler rates. Thinning via superposition is often recommended in the literature; notable examples include Bouchard-Côté *et al.* (2018) who illustrate this approach when simulating from an exponential family and Wu and Robert (2020) who discuss the approach generally. The approach was also recommended broadly by Sen *et al.* (2020) where it was suggested for practical implementation of the Zig-Zag where f_1 and f_2 correspond to the terms from the likelihood and prior respectively. Once simulation for a choice of prior or likelihood are individually established this knowledge can be reused in future modelling.

4 Concave-convex adaptive thinning

Our general proposal for simulating events in a PDMP is based on concave-convex adaptive thinning. As the dynamics between events are deterministic, conditional on the current state of the PDMP, we can re-write the rate of the next event as a function of time. With slight abuse of notation we will denote this rate as $\lambda(t)$ with the argument of the function making it clear whether we are viewing it as a function of the state, or as here, as a function of time. Suppose $\lambda(t) = \max\{0, f(t)\}$ where $f(t)$ can be decomposed as:

$$f(t) = f_{\cup}(t) + f_{\cap}(t) \tag{1}$$

on a finite interval $t \in [0, \tau_{\max})$ where $f_{\cup}(t)$ is a convex function and $f_{\cap}(t)$ is a concave function in time. We will later discuss general conditions where such a decomposition is possible. The problem of upper-bounding $f(t)$ is recast to finding upper-bounding piecewise linear functions $\ell_u(t)$ and $\ell_n(t)$ such that $f_{\cup}(t) \leq \ell_u(t)$ and $f_{\cap}(t) \leq \ell_n(t)$. We may then apply the bound $f(t) \leq \ell(t)$ where $\ell(t) = \ell_n(t) + \ell_u(t)$. Since $\ell(t)$ is the sum of piecewise linear functions it will also be a piecewise linear function and direct simulation from a Poisson process with rate $\ell(t)$ is possible (see Appendix A).

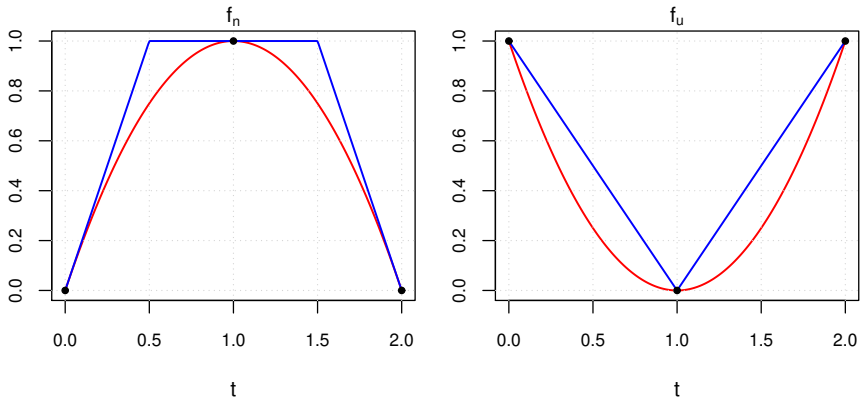


Figure 1: Upper bounds for a rate function based on concave (left) and convex (right) information on three abscissae ($t_i = 0, 1, 2$). Concave bounds are formed using linear segments from the gradient of $f_{\cap}(t_i)$. Convex bounds are constructed using linear segments connecting evaluations of $f_{\cup}(t_i)$.

Concave-convex adaptive thinning proceeds by constructing a piecewise linear function $\ell(t)$ over a set of m abscissae $t_1 = 0 < t_2 < \cdots < t_m = \tau_{\max}$ at which the evaluation of $f_{\cup}(t_i), f_{\cap}(t_i)$ and derivative $f'_{\cap}(t_i)$ are known. An event time τ is simulated from the Poisson process with rate $\bar{\lambda}(t) = \max\{0, \ell(t)\}$ and is accepted with probability

$$\frac{\max\{0, f_{\cup}(\tau) + f_{\cap}(\tau)\}}{\bar{\lambda}(\tau)}.$$

If the event is rejected, we can reuse the information from the evaluation of $f_{\cup}(\tau)$ and $f_{\cap}(\tau)$ with an additional evaluation of $f'_n(\tau)$ to refine the simulation on the abscissae $t_1 = \tau$ to $\tau_{\max}$ where all existing evaluations for $\tau < t_k \leq \tau_{\max}$ may be reused. If an event does not occur on the range $t \in [0, \tau_{\max})$ the PDMP process is evolved by $\tau_{\max}$ and the thinning process is repeated. We give the general construction of the piecewise linear function $\ell(t)$ for $t \in [t_1, t_2)$ and note this construction may be applied iteratively over the abscissae. This construction is also depicted visually in Figure 1. A convex function $f_{\cup}(t)$, is by definition a function that can be upper-bounded by the line segment connecting any two function evaluations. So, for any

$t \in [t_1, t_2)$ we have the upper-bound $f_\cup(t) \leq \ell_u(t)$ where

$$\ell_u(t) = f_\cup(t_1) + \frac{f_\cup(t_2) - f_\cup(t_1)}{t_2 - t_1}(t - t_1).$$

For a concave function $f_\cap(t)$ with derivative $f'_\cap(t)$ the line segment corresponding to the tangent of $f_\cap(t)$ will upper-bound the function. Thus, for $t \in [t_1, t_2)$, $f_\cap(t)$ will be upper-bounded by

$$\ell_n(t) = \min\left\{f_\cap(t_1) + f'_\cap(t_1)(t - t_1), f_\cap(t_2) + f'_\cap(t_2)(t_2 - t)\right\}.$$

This minimum will switch at the point of intersection between the lines $f_\cap(t_1) + f'_\cap(t_1)(t - t_1)$ and $f_\cap(t_2) + f'_\cap(t_2)(t_2 - t)$. It is simple to find this intersection point (Görür and Teh, 2011),

$$t^* = \frac{f_\cap(t_2) - f'_\cap(t_2)t_2 - f_\cap(t_1) + f'_\cap(t_1)t_1}{f'_\cap(t_1) - f'_\cap(t_2)}$$

which will be a point on the interval $[t_1, t_2]$ provided the derivatives $f'_\cap(t_1)$ and $f'_\cap(t_2)$ are not equal. If $f'_\cap(t_1) = f'_\cap(t_2)$, the linear segment will not change over the interval and we take $t^* = t_2$. So we take

$$\ell_n(t) = \begin{cases} f_\cap(t_1) + f'_\cap(t_1)t & t \in [t_1, t^*) \\ f_\cap(t_2) + f'_\cap(t_2)(t_2 - t) & t \in [t^*, t_2) \end{cases},$$

and combining these bounds we have $\ell(t) = \ell_u(t) + \ell_n(t)$ which is a piecewise linear function upper-bounding $f(t) \leq \ell(t)$ for $t \in [t_1, t_2)$.

4.1 Concave-convex decompositions

The proposition below gives simple conditions for the class of non-homogeneous Poisson processes that admit thinning using the concave-convex approach.

Proposition 1. *If a Poisson process with rate function $\lambda(t)$ can be written as $\lambda(t) = \max\{0, f(t)\}$ where $f(t)$ is continuous then the process admits thinning using concave-convex adaptive thinning.*

Proof. Consider $f(t)$ on the interval $[0, \tau_{\max}]$ where $\tau_{\max}$ is some arbitrary value $\tau_{\max} > 0$. On this closed compact interval $f(t)$ is continuous and by the Stone-Weierstrass theorem (Stone, 1948), for any $\epsilon > 0$ there exists a polynomial $g(t)$ on the interval with $\|f - g\| < \epsilon$. The function g is a polynomial so it admits the concave-convex decomposition

$$g(t) = \sum_{\{i:a_i>0\}} a_i t^i + \sum_{\{i:a_i<0\}} a_i t^i$$

with convex function $g_u(t) = \sum_{\{i:a_i>0\}} a_i t^i$ and concave function $g_n(t) = \sum_{\{i:a_i<0\}} a_i t^i$. The Process with rate $\lambda(t)$ admits thinning on the interval $[0, \tau_{\max})$ using the Poisson process with rate $\hat{\lambda}(t) = \max\{0, g(t) + \epsilon\}$ which has a convex-concave decomposition. If an event does not occur on the interval $[0, \tau_{\max})$ then the process is evolved to $\tau_{\max}$ and the thinning process repeated. □

Remark 1. While continuity of $f(t)$ is required for the Stone-Weirestrass theorem, the concave-convex process may also be applied more generally. If $f(t)$ is continuous for all $0 < t < t^*$ and $\lim_{t\to(t^*)-} f(t) = \infty$ then concave-convex thinning can be applied. Consider $f(t)$ using the abscissae $t_0 = 0, t_1 = \frac{t^*}{2}, \text{tmax} = t^*$. Proposition 1 ensures a concave-convex decomposition on the interval $[0, t^*/2)$. On $[t^*/2, t^*)$ we upper bound the rate by infinity. So if a time is not simulated on $[0, t^*/2)$, we will propose an event at t_1, and this event will be rejected. The concave-convex bound will then be updated based on $f(t^*/2)$, and used to simulate an event on $[t^*/2, t^*)$. The abscissae on this increment will be $t^*/2$, $3t^*/4$, and t^*, and the process will be repeated. Importantly, as $\lim_{t\to(t^*)-} f(t) = \infty$ there will be an actual event before t^* with probability one, thus repeating this process will yield an (accepted) event time.

The proof of Proposition 1 relies on finding a polynomial that can upper-bound the process over a finite interval. If a bound can be given for higher order derivatives of $f(t)$ then there are two main approaches for finding an upper-bounding polynomial. Using Taylor's expansion of $f(t)$ about zero, we get

$$f(t) = f(0) + f'(0)t + \frac{f''(0)}{2!}t^2 + \cdots + \frac{f^{(k)}(0)}{k!}t^k + \int_0^t f^{(k+1)}(s)\frac{t^k}{k!}ds.$$

If there is a known constant M such that $|f^{(k+1)}(t)| \leq M$ we can employ thinning based on the polynomial bound $g(t) = f(0) + f'(0)t + \frac{f''(0)}{2!}t^2 + \cdots + \frac{f^{(k)}(0)}{k!}t^k + \frac{M}{(k+1)!}t^{(k+1)}$. Alternatively polynomial upper-bounds can be constructed based on interpolation where the error is controlled. For example using Lagrange polynomial interpolation on $[0, \tau_{\max}]$ with $k+1$ evaluations of the function has error bounded by $\tau_{\max}^{k+1}\frac{M}{(k+1)!}$. Adding this bound as a constant yields an upper-bounding polynomial without requiring evaluation of the derivatives of $f(t)$ required for the Taylor series expansion.

4.2 Concave-convex adaptive thinning and PDMP-based samplers

A key advantage to CC-PDMP is that it can easily and efficiently simulate from rates with an additive structure. That is, if $f_1(t)$ and $f_2(t)$ both have concave-convex decompositions then $f(t) = f_1(t) + f_2(t)$ will also have a concave-convex decomposition. The rates associated to a PDMP-based sampler with $\pi(\boldsymbol{\theta}) \propto \exp(-U(\boldsymbol{\theta}))$ as its target will have an additive structure. In particular, a local PDMP sampler as described in Section 2.4 will have rates

$$\lambda_f(\boldsymbol{z}_t) = \max\left\{0, \sum_{j \in S_f} f_j(t)\right\}$$

where $f_j(t) = v_j\partial_{\theta_j}U(\boldsymbol{\theta}+t\boldsymbol{v})$. This means that the concave-convex decompositions for the partial derivatives $f_j(t)$ can be trivially combined to simulate from new local or global implementations of the PDMP. There is a subtle trade-off between computational and statistical efficiency offered by global and local PDMP-based samplers which we explore in Section 6.

Additive rates are also present when using a PDMP to simulate from a Bayesian posterior distribution. For a model with parameter $\boldsymbol{\theta} \in \mathbb{R}^p$, the target has the form

$$\pi(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta})p(y_{1:n}|\boldsymbol{\theta})$$

where $p(y_{1:n}|\boldsymbol{\theta})$ is the likelihood of the observed data $y_{1:n} = (y_1, ..., y_n)$, and $p(\boldsymbol{\theta})$ is the prior for the parameters of the model. The function associated to the jth partial derivative of $U(\boldsymbol{\theta})$ can be written as $f_j(t) = f_j^p(t) + f_j^\ell(t)$ where $f_j^p(t) = v_j\partial_{\theta_j}\log p(\boldsymbol{\theta}+t\boldsymbol{v})$ depends on the log prior and $f_j^\ell(t) = v_j\partial_{\theta_j}\log p(y_{1:n}|\boldsymbol{\theta}+t\boldsymbol{v})$ depends on the log likelihood. This allows for the rate function to be split into more manageable pieces and once a decomposition has been found for a choice of prior or likelihood, it can be reused to facilitate simulating from that prior or likelihood in future models. Specific examples are given in Section 6.

5 Tuning parameters

5.1 Choice of abscissae

A key user choice in CC-PDMP sampling is that of the position and number of abscissae on the interval $[0, \tau_{\max})$. Suppose that CC-PDMP simulation is implemented with abscissae $t_0 = 0 < t_1 < \cdots < t_n = \tau_{\max}$ if an event does not occur on the interval $[0, \tau_{\max})$ the PDMP will be evolved by $\tau_{\max}$ and the thinning process will repeat where the function evaluations at $\tau_{\max}$ can be reused at $t_0 = 0$. Thus one simulation with n abscissae would be computationally equivalent

to n evolutions of the CC-PDMP approach using abscissae with only two points $t_0 = 0$ and $t_1 = \tau_{\max}$. This encourages choosing a minimal number of abscissae and more carefully choosing the length of the interval. There are two main issues that can occur when tuning the parameter $\tau_{\max}$. If $\tau_{\max}$ is too small, then the proposal frequently lie outside the interval $[0, \tau_{\max})$ and simulating an event will requiring many iterations of bounding the event rate. Whilst if $\tau_{\max}$ is too large then the bound on the rate may be be poor, leading to many events that are rejected. In this article, we refer to an iteration of the CC-PDMP that does not generate an event time (i.e. a rejected proposal event or not generating on the interval) as a *shadow event*. The total number of iterations will be the sum of the number of events and shadow events. *Efficiency* is measured as the proportion of iterations that are events. Adapting the value of $\tau_{\max}$ will not change the sampling dynamics. Consequently, unlike in adaptive MCMC, this parameter may be adapted throughout the course of simulating the PDMP. Ideally this parameter should be a little larger than the average event time so that events are proposed on the interval. A simple automatic approach for choosing this parameter is to set $\tau_{\max}$ equal to the q-th percentile of previous simulated event times, and update $\tau_{\max}$ every 100 iterations of the sampler. We found that setting $q = 80$th percentile worked well for automatically selecting $\tau_{\max}$. We investigate the dependency on this tuning parameter by implementing the Zig-Zag sampler on the 2-dimensional Banana distribution, with

$$U(\boldsymbol{\theta}) = (\theta_1 - 1)^2 + \kappa(\theta_2 - \theta_1^2)^2,$$

where κ controls how much mass concentrates around the region $\theta_2 \approx \theta_1^2$. Since the potential is polynomial in both parameters it is trivial to see that the Zig-Zag rate functions will also be polynomial. Further implementation details may be found in the supplementary material and associated GitHub.

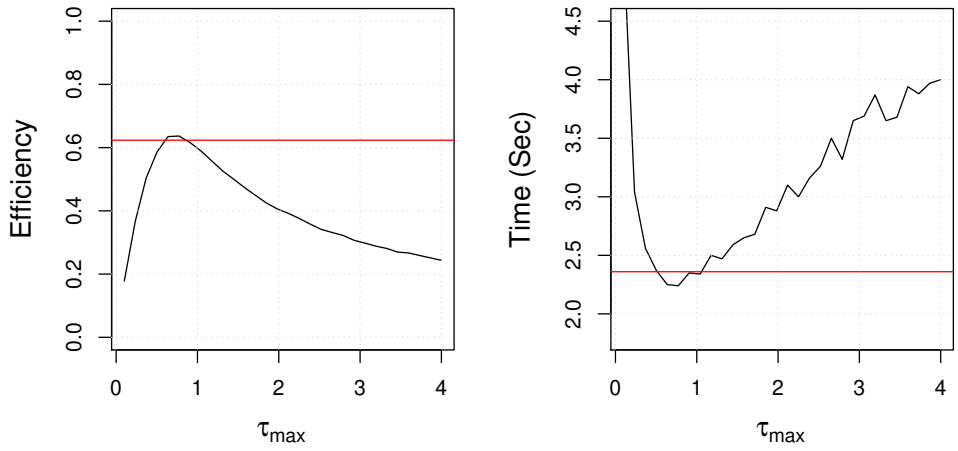


Figure 2: Efficiency, defined as the proportion of iterations (events + shadow events) that are events, of Zig-Zag using the CC-PDMP thinning with varying choices for $\tau_{\max}$ on the Banana target. Plot on the right shows the actual computation time for running the sampler with different choices of $\tau_{\max}$. The red line shows the performance of an automatic choice for this parameter.

Figure 2 shows the efficiency of the Zig-Zag sampler on the Banana distribution with $\kappa = 1$ when changing the interval length $\tau_{\max}$ from 0.1 to 4 with two abscissae. The Zig-Zag algorithm was simulated for 10000 events at each value of $\tau_{\max}$. The effect of selecting $\tau_{\max}$ too small or too large is clearly seen and the red line shows the performance using the adaptive approach. The main computational cost is due to an artefact of the otherwise fast C++ code having to call an R function each time the concave-convex decomposition is evaluated. For simple event rates such as polynomials this computation can be implemented in C++ as well. Appendix D

illustrates the effect of $\tau_{\max}$ on computation time when the convex-concave decomposition is evaluated in C++.

5.2 Choice of decomposition

The choice of concave-convex decomposition is not unique (Görür and Teh, 2011). Arbitrary convex functions can be added to $f_\cup(t)$ and subtracted from $f_\cap(t)$ to give a new valid decomposition. A concave-convex decomposition is minimal if there is no non-affine convex function that can be added to $f_\cap(t)$ and subtracted from $f_\cup(t)$ while preserving the convexity of the decomposition (Hartman, 1959). For example, consider the function

$$f(t) = -t^3 + 3t^2 - 3t + 3,$$

where $f''(t) = -6t + 6$. It is simple to see that the function is convex for $t < 1$ and concave for $t > 1$. A minimal decomposition is given by the piecewise functions

$$f^1_\cup(t) = \begin{cases} -t^3 + 3t^2 - 3t + 3 & t \leq 1 \\ 0 & t > 1 \end{cases} \text{ and } f^1_\cap(t) = \begin{cases} 0 & t \leq 1 \\ -t^3 + 3t^2 - 3t + 3 & t > 1 \end{cases}.$$

Görür and Teh (2011) give a general construction for a minimal concave-convex decomposition. However, this construction relies on finding all points of inflection, which are points where the function changes convexity. Alternatively, Proposition 1 gives a simple decomposition $f^2_\cup(t) = 3t^2 + 3$ and $f^2_\cap(t) = -t^3 - 3t$. While the decomposition from Proposition 1 is not minimal, it does not require finding all points of inflection. These two decompositions are shown in Figure 3 using abscissae at 0 and 1. The optimal decomposition gives a tighter bound, here reducing the bounding rate by approximately 0.33 on average across the interval. However, using this bound comes with additional computation cost of finding inflection points. This has the potential to reduce the overall efficiency of the method. Our experience is that the efficiency gains for using the optimal polynomial are generally not sufficient for it to be beneficial. See Appendix D for further discussion and simulations.

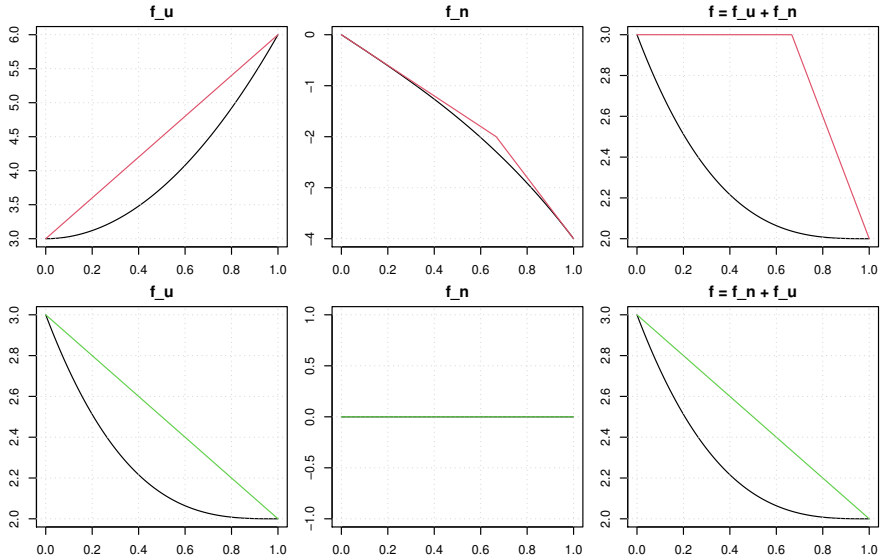


Figure 3: The upper-bounds resulting from two different concave-convex decompositions of the function $f(t) = -t^3 + 3t^2 - 3t + 3$, based on abscissae at 0 and 1. The top row corresponds to the decomposition from Proposition 1 and the bottom row corresponds to an optimal decomposition which recognises $f(t)$ as a convex function on $[0, 1)$. The columns show the functions $f_\cup$, $f_\cap$ and f as well as their piece-wise linear upper-bounds.

6 Experiments

We now present empirical evaluation of CC-PDMP and comparison with other approaches to simulate PDMPs. Our experiments were implemented using the R package CCPDMP available at `https://github.com/matt-sutton/ccpdmp`.

The package enables you to simulate a PDMP provided one specifies the concave and convex decomposition of the rate function. If the rate function is polynomial (or bounded by a polynomial) the practitioner may parse this instead and the concave-convex decomposition will be handled internally. The package contains some basic example use cases and code to reproduce the experiments. Code snippets are given in the additional material.

6.1 Application in generalised linear models

Generalised linear models (GLMs) provide a rich class of models that are frequently used in Bayesian inference. Let $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^n$ be the observed data, where y_i is an observed response and $\boldsymbol{x}_i \in \mathbb{R}^p$ is a vector of associated covariates for $i = 1, \ldots, n$. The expected value of y_i is modelled by $g^{-1}(\boldsymbol{x}_i^T\boldsymbol{\theta})$ where $g^{-1} : \mathbb{R} \to \mathbb{R}$ is the inverse link function. The potential of the likelihood has the form

$$U^{(\ell)}(\boldsymbol{\theta}) = \sum_{i=1}^n \phi(\boldsymbol{x}_i^T\boldsymbol{\theta}, y_i)$$

where $\phi : \mathbb{R}^2 \to \mathbb{R}$ is the GLM mapping function $\phi(a, y) = \log p(y|g^{-1}(a))$ returning the log likelihood for observation y given a. The partial derivative with respect to θ_k has the form

$$\partial_{\theta_k} U^{(\ell)}(\boldsymbol{\theta}) = \sum_{i=1}^n \phi'(\boldsymbol{x}_i^T\boldsymbol{\theta}, y_i)x_{ik}$$

where $\phi'(a, y) = \partial_a\phi(a, y)$ and higher order derivatives are defined similarly. Over the time interval $t \in [0, \tau]$ let $f_k(t) = v_k\partial_{\theta_k}U^{(\ell)}(\boldsymbol{\theta} + t\boldsymbol{v})$ which is

$$f_k(t) = v_k \sum_{i=1}^n \phi'(a_i(t), y_i)x_{ik}$$

where $a_i(t) = \boldsymbol{x}_i^T(\boldsymbol{\theta} + t\boldsymbol{v})$. We can use this to define local rates as $\lambda_k(t) = \max\{0, f_k(t)\}$ for $k = 1, ..., p$. For GLMs, repeated application of the chain rule yields:

$$f_k^{(j)}(t) = v_k \sum_{i=1}^n \phi^{(j+1)}(a_i(t), y_i)\left(\frac{\partial a_i}{\partial t}\right)^j x_{ik}$$

where $\frac{\partial a_i}{\partial t} = \boldsymbol{x}_i^T\boldsymbol{v}$ and $f_k^{(j)}$ denotes the jth derivative of $f_k(t)$. When there are bounds on $\phi^{(j)}$ we can use the upper-bounding Taylor polynomial. In Appendix B we provide bounds for several modelling choices. Here we consider logistic regression, which has the mapping function $\phi(a, y) = \log(1 + \exp(a)) - ya$ where $y \in \{0, 1\}$. We look at the efficiency of CC-PMDP thinning for a five dimensional logistic regression problem with $n = 200$ observations. The covariates were generated from a multivariate normal, $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{0}, V^{-1})$ for $i = 1, .., 200$, with mean zero, precision matrix

$$V = \begin{pmatrix} 1 & \rho & 0 & 0 & 0 \\ \rho & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and data generated using $\theta = (-1.25, 0.5, -0.4, -0.4, -0.4)^T$. We take a Gaussian prior $\theta_j \sim \mathcal{N}(0, 1)$ for $j = 1, \ldots, 5$. As the correlation is increased the thinning becomes more challenging.

We investigate the performance of CC-PDMP for increasing polynomial bounds with increasing correlation ρ.

Polynomial	Correlation						
Order	0.00	0.25	0.50	0.65	0.75	0.85	0.95
1	0.64 (4.13)	0.58 (4.41)	0.49 (4.87)	0.41 (5.57)	0.34 (6.47)	0.26 (8.06)	0.13 (14.80)
2	0.77 (4.13)	0.76 (4.12)	0.75 (4.16)	0.72 (4.26)	0.68 (4.42)	0.62 (4.75)	0.39 (6.56)
3	0.80 (5.56)	0.80 (5.56)	0.79 (5.58)	0.78 (5.67)	0.75 (5.76)	0.70 (6.03)	0.47 (7.91)

Table 1: Efficiency of thinning in Zig-Zag sampling of logistic regression for increasing order of Taylor series polynomial thinning bound. Efficiency is measured as the average proportion of iterations that are events over 20 repetitions of the sampler. Average runtime, in seconds, of sampler is shown in ()s.

To date only linear bounds (polynomial order 1) for logistic regression have been used for thinning (Bierkens *et al.*, 2018; Bouchard-Côté *et al.*, 2018). These are based on the bound $|\phi''(a,y)| \leq 1/4$. Table 1 shows the efficiency for thinning using polynomials of order 1-3 (using bounds $|\phi''(a,y)| \leq 1/4$, $|\phi'''(a,y)| \leq 1/(6\sqrt{3})$ and $|\phi^{(4)}(a,y)| \leq 1/8$). The linear 1st order bound matches the bound used in Bierkens *et al.* (2018) for logistic regression using the Zig-Zag sampler. Table 1 shows that higher order polynomial thinning facilitated through the CC-PDMP allows more efficient thinning. However, these bounds can incur higher computational cost as they require evaluation of the higher order derivatives. For this example an order 2 polynomial appears to have the trade-off in computation and thinning efficiency giving the best performance. For the linear bounds we used a large fixed $\tau_{\max}$ to ensure the linear rate proposal is simulated on the interval $[0, \tau_{\max})$. For other polynomial orders we used the adaptive procedure described in the tuning section to select $\tau_{\max}$.

6.2 Comparison to thinning via superposition

In this example we demonstrate the advantages of CC-PDMP when the event rate is additive. In particular, we compare thinning using the CC-PDMP approach with thinning using superposition as outlined in Section 3. Let $\{y_i\}_{i=1}^n$ be the observed data with the following model

$$\begin{aligned} y_i \mid \theta_i &\sim \text{Poisson}(\exp(\theta_i)), \\ \theta_i &\sim \mathcal{N}(0,1) \end{aligned}$$

independently for $i = 1..., n$. The derivative of the potential is $\partial_{\theta_k} U(\boldsymbol{\theta}) = \theta_k - y_k + \exp(\theta_k)$ and

$$f_k(t) = v_k \partial_{\theta_k} U(\boldsymbol{\theta} + t\boldsymbol{v}) = v_k(\theta_k + v_k t) - y_k v_k + v_k \exp(\theta_k + v_k t).$$

This has the concave-convex decomposition $f_\cup(t) = f_k(t)$ when $v_k > 0$ and $f_\cup(t) = v_k(\theta_k + v_k t) - y_k v_k$, $f_\cap(t) = v_k \exp(\theta_k + v_k t)$ when $v_k < 0$. The global Bouncy Particle Sampler rate can be defined as

$$\lambda(t) = \max\left\{0, \sum_{k=1}^n v_k \partial_{\theta_k} U(\boldsymbol{\theta} + t\boldsymbol{v})\right\} = \max\left\{0, \sum_{k=1}^n f_k(t)\right\},$$

which has concave-convex decomposition defined by the decompositions of the individual f_k functions. In contrast, thinning via superposition involves simulating an event time for the linear and exponential terms of the rate individually. The proposed event time is the minimum of all simulated times. This approach to simulating event times for exponential likelihood and Poisson-Gaussian Markov random fields has been previously used in implementing PDMPs (Bouchard-Côté *et al.*, 2018; Vanetti *et al.*, 2017).

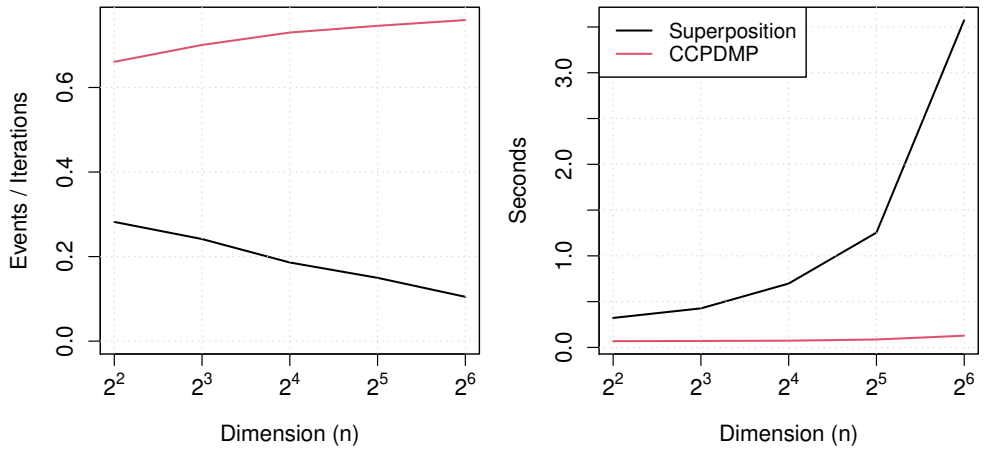


Figure 4: Efficiency of thinning for the global BPS using thinning via super-position and CC-PDMP thinning with increasing dimension. Efficiency of the thinning (left) and total computation time (right) is averaged over 20 repeated runs. The Bouncy Particle Sampler samplers were simulated for 1000 event times on each run.

Figure 4 compares the thinning and overall computational efficiency for the Bouncy Particle Sampler applied to the Poisson likelihood problem. In the CC-PDMP approach we used the adaptive update for $\tau_{\max}$ as described in the tuning section. As the dimension increases we see the proportion of iterations that are events using superposition drops quickly, consequently the overall computation time for this approach scales poorly. The poor performance of the superposition approach with increasing dimension is the result of the large number of exponential terms, $v_k \exp(\theta_k + v_k t)$, in the event rate. The thinning acceptance rate for this proposal is

$$\frac{\max\left\{0, \sum_{k=1}^{n} v_k(\theta_k + v_k t - y_k + \exp(\theta_k + v_k t))\right\}}{\max\left\{0, \sum_{k=1}^{n}(v_k(\theta_k + v_k t) - y_k v_k)\right\} + \sum_{k=1}^{n} \max\left\{0, v_k \exp(\theta_k + v_k t)\right\}}$$

When $v_k < 0$ these exponential terms contribute zero to the denominator of the superposition approach. In comparison the CC-PDMP approach uses a linear upper-bound on these exponential terms which can be negative and reduce the denominator term, leading to more efficient thinning proposals. This is seen in Figure 4 where the thinning efficiency remains roughly constant with increasing dimension.

6.3 Local Methods

Local PDMP methods take advantage of the conditional independence between parameters. Consider the following extension to the previous Poisson example

$$\begin{aligned} y_i \mid \theta_i &\sim \text{Poisson}(\exp(\theta_i)) \qquad \text{independently for } i = 1, \ldots, n \\ \theta_1 &\sim \mathcal{N}(0, 1/(1-\rho^2)) \\ \theta_i \mid \theta_{i-1} &\sim \mathcal{N}(\rho\theta_{i-1}, 1), \qquad \text{for } i = 2, \ldots, n \end{aligned}$$

where we fix $\rho = 0.5$. The prior on $\boldsymbol{\theta}$ corresponds to an AR(1) process. The partial derivative of the potential is

$$\partial_{\theta_k} U(\boldsymbol{\theta}) = \begin{cases} (1+\rho^2)\theta_k - \rho\theta_{k+1} - y_k + \exp(\theta_k) & k = 1 \\ (1+\rho^2)\theta_k - \rho\theta_{k-1} - \rho\theta_{k-1} - y_k + \exp(\theta_k) & 1 < k < n \\ \theta_k - \rho\theta_{k-1} - y_k + \exp(\theta_k) & k = n \end{cases}$$

which has the same linear and exponential form as the rate in Section 6.2 so we can use an analogous concave-convex decomposition for this rate. In this section we consider local PDMP implementations. The CC-PDMP implementation facilitates simple construction of thinning bounds for local PDMP factorisations. For the partition $\mathcal{S} = \{S_1, \ldots, S_F\}$ the local rate for factor f is

$$\lambda_f(\boldsymbol{z}_t) = \max\left\{0, \sum_{k \in S_f} v_k \partial_{\theta_k} U(\boldsymbol{\theta}_t)\right\} = \max\left\{0, \sum_{k \in S_f} f_k(t)\right\},$$

for $f = 1, \ldots, F$. In this section we consider a number of local decompositions of the form $\mathcal{S}^{(j)} = \{S_1, \ldots, S_F\}$ where $S_1 = \{1, 2, ..., j\}$, $S_2 = \{j+1, j+2, .., 2j\}$, $\ldots, S_F = \{n-j, \ldots, n-1, n\}$. We assume n is divisible by F so each S_f contains j elements. For these decompositions the conditional independence between parameters gives the following neighbours $\mathcal{N} = \{N_1, \ldots, N_F\}$ where

$$N_f = \begin{cases} \{f, f+1\} & f = 1 \\ \{f-1, f, f+1\} & 1 < f < F \\ \{f-1, f\} & f = F \end{cases}.$$

This local decomposition offers computational advantages since updating a single rate requires recalculation of at most three rates regardless of the dimension of the problem. However, this computational efficiency comes at a loss of statistical efficiency. In particular, a global PDMP will be able to move further in stochastic time than local methods before requiring a new event simulation as the global rate function will lower bound the local rate. The overall efficiency of the PDMP method should therefore be considered in terms of both its computational and statistical efficiency.

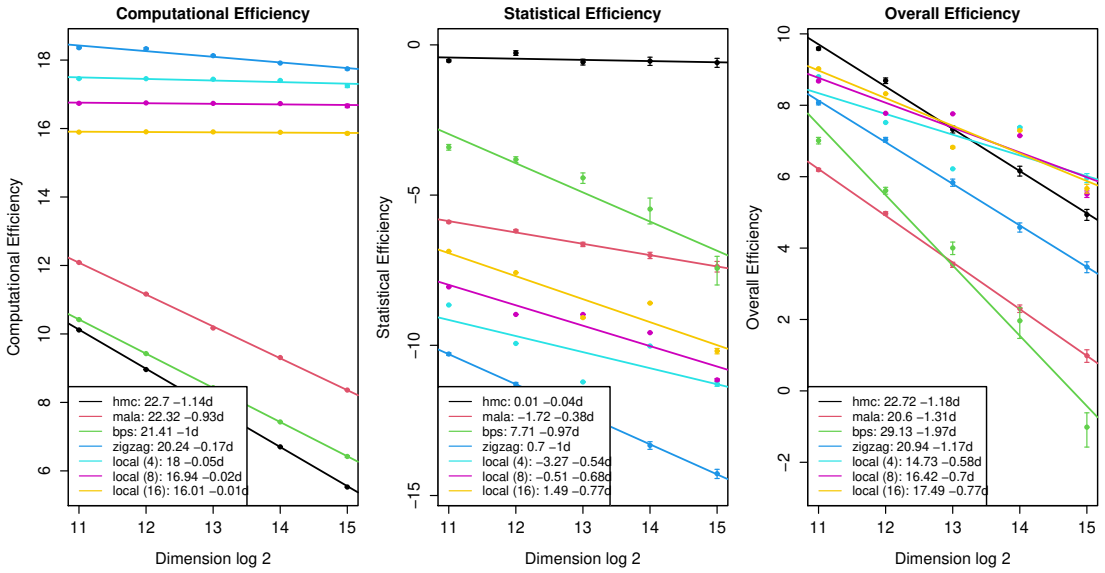


Figure 5: Log-log breakdown of computational, statistical and overall empirical efficiency scaling with dimension. The ESS values are calculated with respect to the first coordinate θ_1 using the coda R package on a discretised trajectory of the PDMPs. Plotted are the average rates and error bars of all methods calculated from 50 repeated runs of the methods. The legend shows the slope and intercept fitted for each method giving empirical evidence for scaling rates.

In Figure 5 we measure overall efficiency as the effective sample size (ESS) per second, the computational efficiency as the number of events or MCMC samples per second and the statistical efficiency as the ESS per MCMC sample or PDMP event.

Figure 5 shows the scaling performance of local methods in comparison with alternative state-of-the-art MCMC methods. We compare local PDMP methods with well-tuned implementations of Metropolis adjusted Langevin algorithm (MALA) and Hamiltonian Monte Carlo (HMC). MALA is tuned to obtain acceptance rate approximately equal to 0.5 and HMC tuned to scale with acceptance probability approximately equal to 0.6. These are close to the optimal values from the MCMC optimal scaling literature (Roberts and Rosenthal, 1998; Beskos *et al.*, 2013). For MALA this involves scaling the variance in the proposal and for HMC this involves adjusting the number of leapfrog steps per iteration. These implementations are in line with theoretical scaling results when the parameters of the distribution are all independent.

As expected, the computational efficiency for the local methods remains (approximately) constant with increasing dimension. Based on the fitted models shown alongside the legends, it appears that computational efficiency for both MALA and Bouncy Particle Sampler scales approximately as $O(d^{-1})$. The computational efficiency for HMC scales at a rate worse than $O(d^{-1})$ due to the increase in the number of leapfrog iterations. The statistical efficiency for the local methods improves with larger local factors and is highest for the Global Bouncy Particle Sampler. The statistical efficiency for MALA drops at a rate roughly equal to $O(d^{-1/3})$. Overall efficiency can be seen as the sum of the log computational and log statistical efficiencies. The Zig-Zag and local Bouncy Particle Sampler methods attain the best overall efficiency scaling rates around $O(d)$ or better. It is clear that there is a trade-off between the statistical efficiency of larger local factors and the increased computational cost. This decomposition can be thought of as an additional choice for PDMP implementation that can easily be tuned using the CC-PDMP approach for thinning. Despite a poorer scaling with dimension, HMC remains competitive with local methods to a very high dimension.

7 Conclusion

PDMP-based samplers have shown advantages over traditional MCMC samplers, but their use has been limited by the perceived difficulty in simulating the PDMP dynamics. We have introduced CC-PDMP as a general approach that can simulate the PDMP provided we can specify a concave-convex decomposition of the event rate. This method has broad applicability, enables simple implementation of local PDMP methods, and empirically outperforms alternative simulation approaches. Additional generalisations of the CC-PDMP approach are also possible by making use of other ideas for adaptive rejection sampling. For example, to bound the concave term, $f_\cap(t)$, of the rate function we require that the derivative $f'_\cap(t)$ is known on the interval $t \in [0, \tau_{\max})$. However, we could instead use a looser bound that does not require this derivative information based on the adaptive rejection bounds proposed in Gilks (1992). If the rate function is particularly computationally intensive, a lower bound based on the abscissae where $f_\cup, f_\cap$ and f'_n and $f'_\cup$ are evaluated may be used to perform early rejection in the thinning. Additional generalisations may also be found in the difference of convex functions programming literature (Le Thi and Pham Dinh, 2018). A general approach to facilitate thinning using a concave-convex decomposition was described in Section 4. This approach advocates finding an upper-bounding polynomial approximation for the rate function. This upper-bounding polynomial may be thinned using the concave-convex adaptive thinning as described in Proposition 1. It is natural to consider an approximate version of our algorithm where this polynomial is estimated via interpolation. Without controlling for the potential error in the polynomial the overall algorithm would be biased. Recently Corbella *et al.* (2022) have proposed an automatic Zig-Zag implementation which uses optimisation methods to obtain a valid linear thinning bound. Future research may consider how to automate the concave-convex thinning procedure for higher order polynomial thinning based on this new work. This may enable more efficient automatic PDMP-based samplers.

Finally while our CC-PDMP approach has been illustrated only on PDMPs with linear dynamics the approach could be used more generally on PDMPs with nonlinear dynamics, such as the Boomerang sampler of Bierkens *et al.* (2020). The only requirement is that the rate

function can be bounded by a function with a concave-convex decomposition.

References

Beskos, A., Pillai, N., Roberts, G., maria Sanz-serna, J. and Stuart, A. (2013). Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli* **19**(5A), 1501–1534.

Bierkens, J. and Roberts, G. (2017). A piecewise deterministic scaling limit of lifted Metropolis–Hastings in the Curie–Weiss model. *The Annals of Applied Probability* **27**(2), 846–882.

Bierkens, J., Bouchard-Côté, A., Doucet, A., Duncan, A. B., Fearnhead, P., Lienart, T., Roberts, G. and Vollmer, S. J. (2018). Piecewise deterministic Markov processes for scalable Monte Carlo on restricted domains. *Statistics & Probability Letters* **136**, 148–154.

Bierkens, J., Fearnhead, P. and Roberts, G. (2019). The zig-zag process and super-efficient sampling for Bayesian analysis of big data. *The Annals of Statistics* **47**(3), 1288–1320.

Bierkens, J., Grazzi, S., Kamatani, K. and Roberts, G. (2020). The boomerang sampler. In: *International Conference on Machine Learning*, PMLR, 908–918.

Bierkens, J., Grazzi, S., van der Meulen, F. and Schauer, M. (2021a). A piecewise deterministic Monte Carlo method for diffusion bridges. *Statistics and Computing* **31**(3).

Bierkens, J., Grazzi, S., van der Meulen, F. and Schauer, M. (2021b). Sticky PDMP samplers for sparse and local inference problems. ArXiv.2103.08478.

Bouchard-Côté, A., Vollmer, S. J. and Doucet, A. (2018). The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method. *Journal of the American Statistical Association* **113**(522), 855–867.

Chevallier, A., Power, S., Wang, A. Q. and Fearnhead, P. (2021). PDMP Monte Carlo methods for piecewise-smooth densities. ArXiv.2111.05859.

Chevallier, A., Fearnhead, P. and Sutton, M. (2022). Reversible jump PDMP samplers for variable selection. *Journal of the American Statistical Association* .

Corbella, A., Spencer, S. E. F. and Roberts, G. O. (2022). Automatic zig-zag sampling in practice. *Statistics and computing* **32**(6), 107.

Cotter, S., House, T. and Pagani, F. (2020). The NuZZ: Numerical zigzag sampling for general models. ArXiv.2003.03636.

Davis, M. (1993). *Markov Models and Optimization*. Chapman Hall.

Diaconis, P., Holmes, S. and Neal, R. M. (2000). Analysis of a nonreversible Markov chain sampler. *Annals of Applied Probability* **10**, 726–752.

Fearnhead, P., Bierkens, J., Pollock, M. and Roberts, G. O. (2018). Piecewise deterministic Markov processes for continuous-time Monte Carlo. *Statistical Science* **33**(3), 386–412.

Gilks, W. R. (1992). Derivative-free adaptive rejection sampling for Gibbs sampling. *Bayesian Statistics* **4**(4), 641–649.

Goan, E. and Fookes, C. (2021). Stochastic bouncy particle sampler for Bayesian neural networks. In: *Stochastic Bouncy Particle Sampler for Bayesian Neural Networks*.

Goldman, J. V. and Singh, S. S. (2021). Spatiotemporal blocking of the bouncy particle sampler for efficient inference in state-space models. *Statistics and Computing* **31**(5), 68.

Görür, D. and Teh, Y. W. (2011). Concave-convex adaptive rejection sampling. *Journal of Computational and Graphical Statistics* **20**(3), 670–691.

Hartman, P. (1959). On functions representable as a difference of convex functions. *Pacific Journal of Mathematics* **9**, 707–713.

Kingman, J. F. C. (1992). *Poisson Processes*. Clarendon Press.

Le Thi, H. A. and Pham Dinh, T. (2018). DC programming and DCA: thirty years of developments. *Mathematical Programming. A Publication of the Mathematical Programming Society* **169**(1), 5–68.

Lewis, P. A. W. and Shedler, G. S. (1979). Simulation of nonhomogeneous Poisson processes by thinning. *Naval Research Logistics Quarterly* **26**(3), 403–413.

Michel, M., Kapfer, S. C. and Krauth, W. (2014). Generalized event-chain Monte Carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps. *The Journal of Chemical Physics* **140**(5), 054116.

Michel, M., Durmus, A. and Sénécal, S. (2020). Forward event-chain Monte Carlo: Fast sampling by randomness control in irreversible Markov chains. *Journal of Computational and Graphical Statistics* **29**, 689–702.

Pakman, A., Gilboa, D., Carlson, D. and Paninski, L. (2017). Stochastic bouncy particle sampler. In: *Proceedings of the 34th International Conference on Machine Learning*, volume 70, PMLR, 2741–2750.

Peters, E. A. and de With, G. (2012). Rejection-free Monte Carlo sampling for general potentials. *Physical Review E* **85**(2), 026703.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **60**(1), 255–268.

Sen, D., Sachs, M., Lu, J. and Dunson, D. (2020). Efficient posterior sampling for high-dimensional imbalanced logistic regression. *Biometrika* **107**(4), 1005–1012.

Stone, M. H. (1948). The generalized Weierstrass approximation theorem. *Mathematics Magazine* **21**(5), 237–254.

Vanetti, P., Bouchard-Côté, A., Deligiannidis, G. and Doucet, A. (2017). Piecewise-Deterministic Markov Chain Monte Carlo. *arXiv:1707.05296* .

Wu, C. and Robert, C. P. (2020). Coordinate sampler: a non-reversible Gibbs-like MCMC sampler. *Statistics and Computing* **30**(3), 721–730.

A Supplementary Material

A.1 Simulating from a piecewise linear rate

Suppose the rate function $\lambda(t) = \max\{0, \ell(t)\}$ is piecewise linear on times $t_0, t_1, \ldots, t_n$ with

$$\ell(t) = \begin{cases} a_0 + b_0 t & 0 < t < t_1 \\ a_1 + b_1 t & t_1 < t < t_2 \\ \ldots & \ldots \\ a_{n-1} + b_{n-1} t & t_{n-1} < t < t_n \end{cases} \tag{2}$$

The event time can be simulated using the following process:

Simulate $u \sim$Uniform[0,1]
Set $E = -\log(u)$
For $k = 0, ..., n-1$:
1. If $\int_{t_k}^{t_{k+1}} \max\{0, a_k + b_k s\} ds > E$, then return τ solving:

$$\tau = \text{argmin}_t \left\{ E = \int_{t_k}^{t} \max(0, a_k + b_k s) ds \right\}.$$

2. If $\int_{t_k}^{t_{k+1}} \max\{0, a_k + b_k s\} ds \leq E$, then update E as

$$E = E - \int_{t_k}^{t_{k+1}} \max\{0, a_k + b_k s\} ds$$

Return

B Alternative GLM likelihood and prior specifications

The likelihood potential for a GLM posterior has the form

$$U^{(\ell)}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \phi(\boldsymbol{x}_i^T \boldsymbol{\theta}, y_i)$$

where $\phi : \mathbb{R}^2 \to \mathbb{R}$ is the GLM mapping function $\phi(a, y) = \log(y|g^{-1}(a))$. As described in Section 6.1 polynomial rates can be constructed if the following functions can be evaluated and bounded

$$f_k^{(j)}(t) = v_k \sum_{i=1}^{n} \phi^{(j+1)}(a_i(t), y_i) \left(\frac{\partial a_i}{\partial t} \right)^j x_{ik}.$$

Functions and derivatives can be found using mathematical software such as Mathematica.

B.1 Robust regression

For robust regression of a response $y \in \mathbb{R}$ we could model the error with wide tails using the **Cauchy likelihood:**

$$\phi(a, y) = -\log\left(1 + \frac{(a-y)^2}{b}\right)$$

with scale $b > 0$. The derivatives are $\phi'(a, y) = -2\frac{a-y}{b+(a-y)^2}$, $\phi''(a, y) = 2\frac{(a-y)^2 - b}{(b+(a-y)^2)^2}$ and $\phi'''(a, y) = -4\frac{(a-y)((a-y)^3 - 3b)}{(b+(a-y)^2)^3}$ with upper-bounds $|\phi'(a, y)| \leq \frac{1}{\sqrt{b}}$, $|\phi''(a, y)| \leq \frac{2}{b}$. For $b \geq 1$

$|\phi'''(a,y)| \leq 3$ (though better bounds are available). For $b \leq 1$, the gradient can be bounded but closed form solutions for this bound are not available.

Mixture of normals:

$$\phi(a,y) = -\log\left(0.5\mathcal{N}(y-a;0,1) + 0.5\mathcal{N}(y-a;0,10^2)\right),$$

upper-bounds on the derivatives for this function are $|\phi'(a,y)| \leq 2$, $|\phi''(a,y)| \leq 0.91$ and $|\phi^{(3)}(a,y)| \leq 1.8$.

B.2 Poisson regression

For $y \sim$Poisson(exp(a)) the mapping function is,

$$\phi(a,y) = -\log\left(\exp(ya - \exp(a))\right).$$

The derivatives for the mapping function are $\phi'(a,y) = \exp(a) - y$ and $\phi^{(j)}(a,y) = \exp(a)$ for $j > 1$. Upper-bounds on the interval $\tau \in [0, \tau_{\max}]$ are $\phi^{(j)}(a,y) \leq \exp(a^*)$ where $a(t) = \boldsymbol{x}^T(\boldsymbol{\theta} + t\boldsymbol{v})$ and $a^* = \max\{a(0), a(\tau_{\max})\}$.

B.3 Gaussian Spike and Slab Prior

Following Chevallier *et al.* (2022) and Bierkens *et al.* (2021b), the spike and slab prior can be simulated using PDMP dynamics. The Gaussian Spike and Slab prior has the form $\theta_j \sim w\mathcal{N}(0,\sigma) + (1-w)\delta_0$ for $j = 1, ..., p$ where w is the prior probability of inclusion and δ is the Dirac spike at zero.

For RJPDMPs there are three types of events that occur; a reversible jump (RJ) move if a variable hits the axis (θ_j =0), events according to the likelihood and prior $\mathcal{N}(0,\sigma)$ for θ_j non-zero, or the RJ move to reintroduce a variable. The rate to reintroduce a variable is calculated in Chevallier *et al.* (2022) where it is found to be

$$\frac{p_{rj}}{\sqrt{2\pi\sigma^2}} \frac{w}{(1-w)}$$

where p_{rj} is a tuning parameter for the probability of jumping to the reduced model when hitting the axis in Chevallier *et al.* (2022). The time to hit an axis for parameter θ_j will be $-\theta_j/v_j$ when $\theta_j v_j > 0$ and ∞ otherwise. Finally simulating non RJ events is equivalent to simulating from the Gaussian slab and thus the part of the event rate contributed by the prior on parameter θ_j is $v_j(\theta_j + v_j t)/\sigma^2$ for $t \in [0, \tau_{\max}]$. This is linear in time so will be exactly simulated by the CC-PDMP approach.

B.4 Cauchy Prior

A Cauchy prior on θ with scale $b \geq 1$ and mean m has the form $p(\theta) \propto \exp(-U(\theta))$ where

$$U(\theta) = \log\left(1 + \frac{(\theta - m)^2}{b}\right)$$

the term contributed to the rate depends on

$$v\partial_\theta U(\theta) = v\frac{2(m-\theta)}{b + (m-\theta)^2}$$

where v is the velocity corresponding to θ. Following the bounds from the Cauchy likelihood (B.1) we have

$$v\partial_\theta U(\theta + tv) \leq f(0) + f'(0)t + \frac{M}{2!}t^2$$

where $f(0) = v\frac{2(m-\theta)}{b+(m-\theta)^2}$, $f'(0) = 2v\frac{(\theta-m)^2-b}{(b+(\theta-m)^2)^2}$ and $M = 3$.

B.5 PDMP-based samplers on restricted domains

PDMP-based samplers can be implemented on a restricted domain following Bierkens *et al.* (2018) or when the density is only piece-wise smooth following Chevallier *et al.* (2021). We consider some PDMP samplers on the restricted domain $\theta > 0$. Simulation involves tracking the time to hitting the boundary $\tau_b = \inf\{t > 0 : \theta + tv = 0\}$. If an event time occurs before the boundary is hit the sampler evolves with the usual rate. If the boundary is hit before the switching event, the sampler is reflected off of the boundary by sampling a new velocity from a probability measure concentrated on directions that are normal to the boundary. We refer the reader to Bierkens *et al.* (2018) for details.

B.6 Generalised inverse Gaussian prior

The generalised inverse Gaussian prior has the form $p(\theta) \propto \exp(-U(\theta))$ where

$$U(\theta) = \theta + \theta^{-1} + 2\log(\theta)$$

and $\theta > 0$. The term contributed to the rate depends on

$$v\partial_\theta U(\theta) = v - v\theta^{-2} + 2\frac{v}{\theta}.$$

When $v > 0$ we have decomposition $f_\cup(t) = v(1+\frac{2}{\theta+vt})$, $f_\cap(t) = -\frac{v}{(\theta+tv)^2}$ and $f'_\cap(t) = v^2\frac{2}{(\theta+vt)^3}$. When $v < 0$ we have decomposition $f_\cup(t) = v(1+\frac{1}{(\theta+vt)^2})$, $f_\cap(t) = \frac{v}{(\theta+tv)}$ and $f'_\cap(t) = -v^2\frac{2}{(\theta+vt)^2}$.

B.7 Gamma prior

The Gamma prior has the form $p(\theta) \propto \exp(-U(\theta))$ where

$$U(\theta) = \beta\theta - (\alpha-1)\log(\theta)$$

and $\theta > 0$ for hyper-parameters $\alpha, \beta > 0$. The term contributed to the rate depends on

$$v\partial_\theta U(\theta) = v\beta - v\frac{(\alpha-1)}{\theta}.$$

When $v(\alpha-1) > 0$ we have decomposition $f_\cup(t) = v(\beta + \frac{(\alpha-1)}{\theta+vt})$, $f_\cap(t) = 0$.
When $v(\alpha-1) < 0$ we have decomposition $f_\cup(t) = v\beta$, $f_\cap(t) = v\frac{(\alpha-1)}{\theta+vt}$ and $f'_\cap(t) = -v^2\frac{(\alpha-1)}{(\theta+vt)^2}$.

C Comparison of CC-PDMP and CC-ARS

We present a comparison between CC-PDMP and sampling directly with CC-ARS where computational cost is measured by the number of proposals used in the adaptive thinning (CC-PDMP) or adaptive sampling (CC-ARS). Both sampling methods are implemented on a generalised inverse Gaussian distribution (GIG). The unnormalised GIG density function is

$$\pi(x) \propto \exp\left(x + x^{-1} + 2\log(x)\right),$$

which is log concave for $2\log(x)$ and log convex for $x + x^{-1}$, see section B.6 for details on the CC-PDMP. We favour the performance of CC-ARS in the 1-dimensional setting since the method yields independent samples while CC-PDMP has correlated samples. Event times in the PDMP-based method give a continuous trajectory where samples can be taken at a consistent times along the trajectory. We observe the performance of the samplers for 60, 300 and 500 proposals – where a proposal is either a thinning proposal for CC-PDMP or a rejection proposal for CC-ARS.

The left column of Figure 6 shows the estimated KDE from the samples based on CC-ARS (black), the KDE based on the PDMP sampler (red) and the true density (blue). The right column shows the samples (black points) and associated PDMP trajectory (red) for the samplers. Rows 1 and 2 of Figure 6 show the methods have similar performance in exploring the density. Jumping from row 2 to 3 we can see that the PDMP sampler only needs to go into the tail once to give a good approximation to the tail of the density. In higher dimensional targets CC-ARS must be used within Gibbs sampling where the efficiency of the sampler will be reduced.

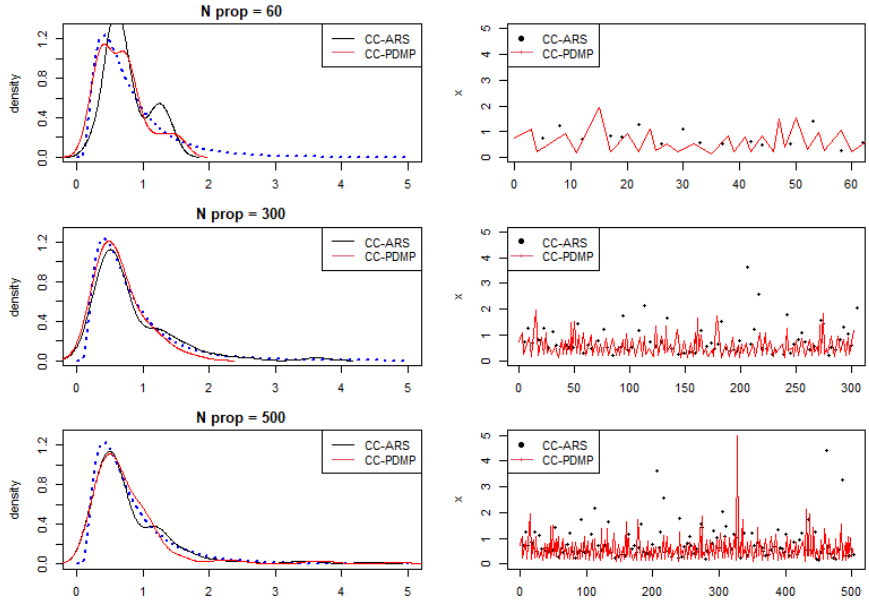


Figure 6: Comparison of typical sampling dynamics for PDMP and ARS sampling using concave-convex upper-bounding in a 1-dimensional GIG distribution. Each row shows the results for an increasing number of proposals using in the sampling procedure. Continuous lines are shown for the PDMP dynamics and points for the ARS method.

D Extra implementation details, sensitivity to $\tau_{\max}$ and optimal decomposition

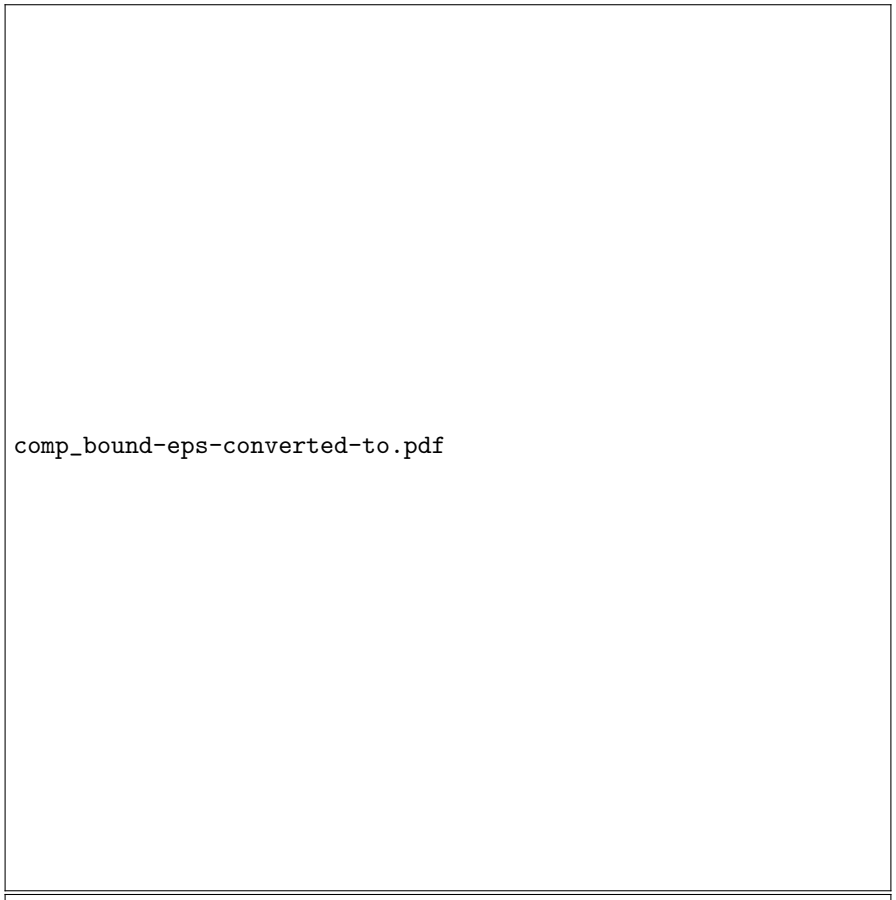
In the R package, the specification of the rates is enabled through direct R coding and the concave-convex thinning procedure is implemented efficiently in C++ classes that are exposed in the R session. This allows for less technical programming via R from the user at the cost of some computational speed. For the case where the thinning bound is a polynomial we provide an efficient C++ implementation that pushes more of the computation inside of the C++ class. The underlying simulation process uses the concave-convex procedure but requires fewer calls to the R environment to evaluate the decomposition when proposing events.

Returning to the banana distribution example from Section 5.1 we explore the implementation speedup using the internal C++ classes to simulate from the polynomial rate. The total computation time is shown in Figure 7. The implementation is more robust to large values of $\tau_{\max}$ and slightly faster than the optimal performance from Section 5.1.

Figure 7: CC-PDMP thinning using C++ to evaluate the decomposition for a polynomial rate. Total computation time for running the Zig-Zag sampler with different choices of $\tau_{\max}$ on the Banana target.

The C++ polynomial class also allows simulation via optimal decomposition or using the default concave-convex polynomial decomposition. The performance of the methods is compared

in Figure 8 based on thinning from a 5th order polynomial. While the bound for the optimal polynomial is far better the computational performance of the two methods is reasonably similar. For this example, the optimal decomposition reliably takes around 25 microseconds to simulate an event while the default method either takes under 25 microseconds or over 35 microseconds. This multi-modal timing may be due to the multi-modal rate function. The optimal rate was also used for thinning in the Zig-Zag sampler targeting the Banana distribution and found to offer very similar performance to the default decomposition (roughly 2 seconds for large $\tau_{\max}$).



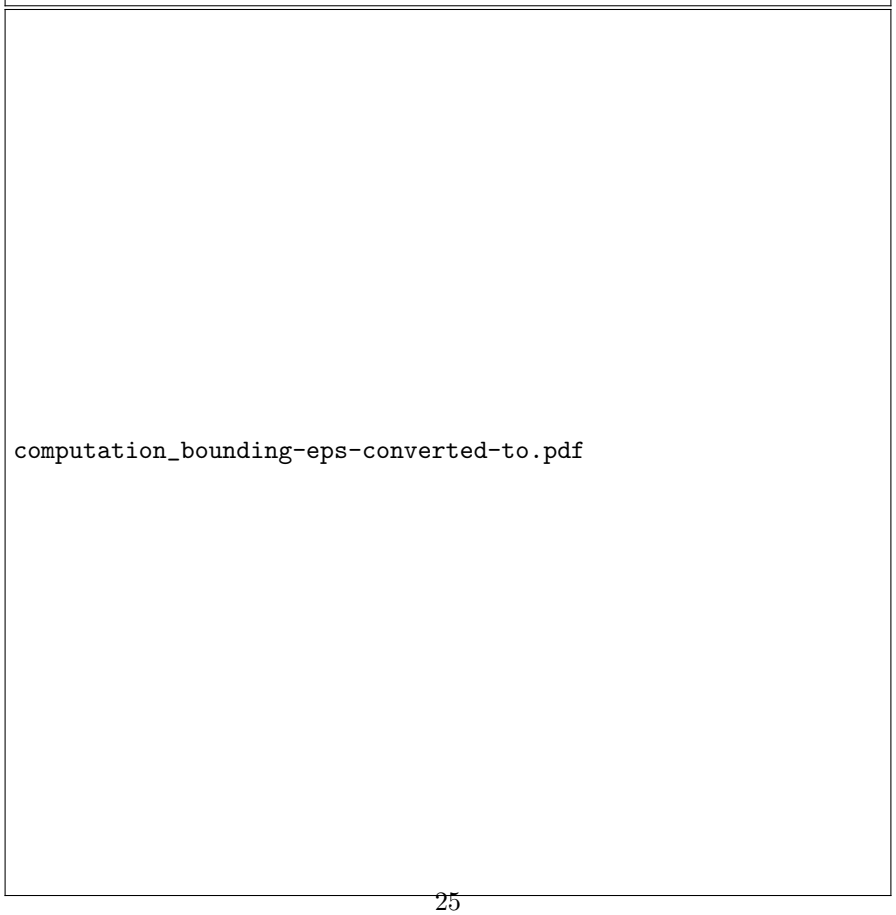


Figure 8: Comparison of simulating an event using the optimal and default polynomial decomposition - typical bounds are illustrated from top row and average computational performance is shown on the bottom row. The top row shows the true event rate, which is the positive part of a 5th order polynomial function (black), together with the concave-convex piecewise linear