Piecewise-Deterministic MCMC

P. Vanetti, A. Bouchard-Côté, G. Deligiannidis & A. Doucet

Potsdam - 07/12/2018

Set up

- Consider a probability distribution on $\mathbb{R}^d$ of density

$$\pi(x) = \frac{\exp(-U(x))}{Z},$$

where the potential U can be evaluated pointwise but $Z = \int_{\mathbb{R}^d} \exp(-U(x))dx$ cannot.

Set up

- Consider a probability distribution on $\mathbb{R}^d$ of density

$$\pi(x) = \frac{\exp(-U(x))}{Z},$$

  where the potential U can be evaluated pointwise but $Z = \int_{\mathbb{R}^d} \exp(-U(x))dx$ cannot.

- We are interested in computing expectations w.r.t. π.

Set up

- Consider a probability distribution on $\mathbb{R}^d$ of density

$$\pi(x) = \frac{\exp(-U(x))}{Z},$$

where the potential U can be evaluated pointwise but $Z = \int_{\mathbb{R}^d} \exp(-U(x))dx$ cannot.

- We are interested in computing expectations w.r.t. π.

- MCMC are the tools of choice in statistics/physics/chemistry/CS.

Examples

- Bayesian inference for high-dimensional graphical models

$$\pi(x) \propto \exp\left\{ -\sum_{i \sim j} \psi(x_i, x_j) - \sum_i \varphi(x_i, y_i) \right\}$$

Examples

- Bayesian inference for high-dimensional graphical models

$$\pi(x) \propto \exp\left\{-\sum_{i\sim j}\psi(x_i, x_j) - \sum_i \varphi(x_i, y_i)\right\}$$

- Bayesian inference for big data

$$\pi(x) \propto p(x)\prod_{i=1}^{n} p(y_i|x)$$

Examples

- Bayesian inference for high-dimensional graphical models

$$\pi(x) \propto \exp\left\{ -\sum_{i\sim j} \psi(x_i, x_j) - \sum_i \varphi(x_i, y_i) \right\}$$

- Bayesian inference for big data

$$\pi(x) \propto p(x) \prod_{i=1}^{n} p(y_i | x)$$

- Bayesian inference for intractable likelihood

$$\pi(x) \propto \exp\left(-\int U_\omega(x)\, \mu(d\omega) \right)$$

Piecewise-deterministic MCMC

- Non-reversible MCMC schemes based on piecewise-deterministic Markov processes have emerged recently.

Piecewise-deterministic MCMC

- Non-reversible MCMC schemes based on piecewise-deterministic Markov processes have emerged recently.
- First appeared in physics (Peters & De With, 2012; Krauth et al., 2009, 2015, 2016) and control (Hespanha & Mesquita, 2012).

Piecewise-deterministic MCMC

- Non-reversible MCMC schemes based on piecewise-deterministic Markov processes have emerged recently.
- First appeared in physics (Peters & De With, 2012; Krauth et al., 2009, 2015, 2016) and control (Hespanha & Mesquita, 2012).
- Provide state-of-the-art performance for a wide range of large scale physical models.

Piecewise-deterministic MCMC

- Non-reversible MCMC schemes based on piecewise-deterministic Markov processes have emerged recently.
- First appeared in physics (Peters & De With, 2012; Krauth et al., 2009, 2015, 2016) and control (Hespanha & Mesquita, 2012).
- Provide state-of-the-art performance for a wide range of large scale physical models.
- Exhibit some appealing features:

Piecewise-deterministic MCMC

- Non-reversible MCMC schemes based on piecewise-deterministic Markov processes have emerged recently.
- First appeared in physics (Peters & De With, 2012; Krauth et al., 2009, 2015, 2016) and control (Hespanha & Mesquita, 2012).
- Provide state-of-the-art performance for a wide range of large scale physical models.
- Exhibit some appealing features:
  - Local updates in graphical models without blocking (Peters & De With 2012, Bouchard-Côté et al. 2015).

Piecewise-deterministic MCMC

- Non-reversible MCMC schemes based on piecewise-deterministic Markov processes have emerged recently.
- First appeared in physics (Peters & De With, 2012; Krauth et al., 2009, 2015, 2016) and control (Hespanha & Mesquita, 2012).
- Provide state-of-the-art performance for a wide range of large scale physical models.
- Exhibit some appealing features:
  - Local updates in graphical models without blocking (Peters & De With 2012, Bouchard-Côté et al. 2015).
  - Exactness under subsampling (Bierkens et al. 2016, Bouchard-Côté et al. 2016; Kapfer & Krauth, 2016).

Piecewise-deterministic MCMC

- Non-reversible MCMC schemes based on piecewise-deterministic Markov processes have emerged recently.
- First appeared in physics (Peters & De With, 2012; Krauth et al., 2009, 2015, 2016) and control (Hespanha & Mesquita, 2012).
- Provide state-of-the-art performance for a wide range of large scale physical models.
- Exhibit some appealing features:
  - Local updates in graphical models without blocking (Peters & De With 2012, Bouchard-Côté et al. 2015).
  - Exactness under subsampling (Bierkens et al. 2016, Bouchard-Côté et al. 2016; Kapfer & Krauth, 2016).
  - Ability to deal with intractable potential $U(x) = \int U_\omega(x)\,\mu(d\omega)$ (Pakman et al. 2016).

Set up

- All MCMC schemes presented here target an extended distribution on $\mathcal{Z} = \mathbb{R}^d \times \mathbb{R}^d$

$$\rho(z) = \pi(x)\psi(v) = \exp(-H(z))$$

where $z = (x, v)$ is the extended state and $\psi(v)$ is the multivariate standard normal.

Set up

- All MCMC schemes presented here target an extended distribution on $\mathcal{Z} = \mathbb{R}^d \times \mathbb{R}^d$

$$\rho(z) = \pi(x)\psi(v) = \exp(-H(z))$$

  where $z = (x, v)$ is the extended state and $\psi(v)$ is the multivariate standard normal.

- Think of v as velocity or momentum variables allowing us to define a deterministic dynamics on $\mathbb{R}^d$.

Set up

- All MCMC schemes presented here target an extended distribution on $\mathcal{Z} = \mathbb{R}^d \times \mathbb{R}^d$

$$\rho(z) = \pi(x)\psi(v) = \exp(-H(z))$$

where $z = (x, v)$ is the extended state and $\psi(v)$ is the multivariate standard normal.

- Think of v as velocity or momentum variables allowing us to define a deterministic dynamics on $\mathbb{R}^d$.

- Sampling from ρ provides samples from π.

Continuous-time PDMP

- *Deterministic dynamics*: An ordinary differential of drift ϕ

$$\frac{dz_t}{dt} = \phi(z_t),$$

inducing a flow

$$z_t = \Phi_t(z_0).$$

Continuous-time PDMP

- *Deterministic dynamics*: An ordinary differential of drift ϕ

$$\frac{dz_t}{dt} = \phi\left(z_t\right),$$

  inducing a flow

$$z_t = \Phi_t\left(z_0\right).$$

- *Event rate* $\lambda : \mathcal{Z} \to \mathbb{R}^+$, with $\lambda\left(z_t\right)\epsilon + o\left(\epsilon\right)$ being the probability of having an event in the time interval $[t, t+\epsilon]$.

Continuous-time PDMP

- *Deterministic dynamics*: An ordinary differential of drift ϕ
$$\frac{dz_t}{dt} = \phi\left(z_t\right),$$
inducing a flow
$$z_t = \Phi_t\left(z_0\right).$$
- *Event rate* $\lambda : \mathcal{Z} \to \mathbb{R}^+$, with $\lambda\left(z_t\right)\epsilon + o\left(\epsilon\right)$ being the probability of having an event in the time interval $[t, t+\epsilon]$.
- *Markov kernel* Q where the state at event time t is given by $z_t \sim Q\left(z_{t^-}, \cdot\right)$, z_{t^-} being the state of the process just before the event.

Simulation of a PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.

Simulation of a PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.

- For $k = 1, 2, \ldots$ do

Simulation of a PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.
- For $k = 1, 2, \ldots$ do
  - Sample inter-event time τ_k, where τ_k is a non-negative random variable such that

$$\mathbb{P}\left(\tau_k \geq t\right) = \exp\left[-\int_{r=0}^{t} \lambda\left\{\Phi_r(z_{t_{k-1}})\right\} dr\right].$$

Simulation of a PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.

- For $k = 1, 2, \ldots$ do
  - Sample inter-event time τ_k, where τ_k is a non-negative random variable such that

$$\mathbb{P}(\tau_k \geq t) = \exp\left[-\int_{r=0}^{t} \lambda\left\{\Phi_r(z_{t_{k-1}})\right\} dr\right].$$

  - For $r \in (0, \tau_k)$, set

$$z_{t_{k-1}+r} \leftarrow \Phi_r(z_{t_{k-1}}).$$

Simulation of a PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.
- For $k = 1, 2, \ldots$ do
  - Sample inter-event time τ_k, where τ_k is a non-negative random variable such that
    $$\mathbb{P}(\tau_k \geq t) = \exp\left[-\int_{r=0}^{t} \lambda\left\{\Phi_r(z_{t_{k-1}})\right\} dr\right].$$
  - For $r \in (0, \tau_k)$, set
    $$z_{t_{k-1}+r} \leftarrow \Phi_r(z_{t_{k-1}}).$$
  - Set $t_k \leftarrow t_{k-1} + \tau_k$ and sample
    $$z_{t_k} \sim Q(z_{t_k^-}, \cdot).$$

Simulation of a PDMP

- Requires being able to compute exactly the flow $z_t = \Phi_t(z_0)$.

Simulation of a PDMP

- Requires being able to compute exactly the flow $z_t = \Phi_t(z_0)$.
  - All existing algorithms use $\phi(z) = (v, 0_d)$ so that $\Phi_t(z_0) = (x_0 + v_0 t, v_0)$, except Hamiltonian BPS (Vanetti et al., 2017).

Simulation of a PDMP

- Requires being able to compute exactly the flow $z_t = \Phi_t(z_0)$.
  - All existing algorithms use $\phi(z) = (v, 0_d)$ so that $\Phi_t(z_0) = (x_0 + v_0 t, v_0)$, except Hamiltonian BPS (Vanetti et al., 2017).
- Requires being able to simulate the event times.

Simulation of a PDMP

- Requires being able to compute exactly the flow $z_t = \Phi_t(z_0)$.
  - All existing algorithms use $\phi(z) = (v, 0_d)$ so that $\Phi_t(z_0) = (x_0 + v_0 t, v_0)$, except Hamiltonian BPS (Vanetti et al., 2017).
- Requires being able to simulate the event times.
  - Inversion, thinning, superposition (Devroye, 1986).

Simulation of a PDMP

- Requires being able to compute exactly the flow $z_t = \Phi_t(z_0)$.
  - All existing algorithms use $\phi(z) = (v, 0_d)$ so that $\Phi_t(z_0) = (x_0 + v_0 t, v_0)$, except Hamiltonian BPS (Vanetti et al., 2017).
- Requires being able to simulate the event times.
  - Inversion, thinning, superposition (Devroye, 1986).
- Requires being able to simulate from Q.

PDMP for Sampling

- The generator of a PDMP is given by

$$
\begin{aligned}
\mathcal{L}f(z) &= \lim_{\epsilon \to 0} \frac{\mathbb{E}\left[f(z_{t+\epsilon}) \,|z_t = z\right] - f(z)}{\epsilon} \\
&= \langle \phi(z), \nabla f(z) \rangle + \lambda(z) \int \left[f(z') - f(z)\right] Q(z, dz').
\end{aligned}
$$

PDMP for Sampling

- The generator of a PDMP is given by

$$\begin{aligned}\mathcal{L}f(z) &= \lim_{\epsilon\to 0} \frac{\mathbb{E}\left[f\left(z_{t+\epsilon}\right)|z_t = z\right] - f(z)}{\epsilon} \\ &= \langle \phi(z), \nabla f(z)\rangle + \lambda(z)\int \left[f(z') - f(z)\right] Q(z, dz').\end{aligned}$$

- For ρ-invariance, we need

$$\int \rho(dz)\,\mathcal{L}f(z) = 0.$$

PDMP for Sampling

- The generator of a PDMP is given by

$$\begin{aligned}\mathcal{L}f(z) &= \lim_{\epsilon\to 0}\frac{\mathbb{E}\left[f\left(z_{t+\epsilon}\right)|z_t = z\right] - f(z)}{\epsilon}\\ &= \langle\phi(z), \nabla f(z)\rangle + \lambda(z)\int\left[f\left(z'\right) - f(z)\right]Q\left(z, dz'\right).\end{aligned}$$

- For ρ-invariance, we need

$$\int \rho(dz)\,\mathcal{L}f(z) = 0.$$

- Sufficient conditions

PDMP for Sampling

- The generator of a PDMP is given by

$$\begin{aligned}\mathcal{L}f(z) &= \lim_{\epsilon\to 0}\frac{\mathbb{E}\left[f\left(z_{t+\epsilon}\right)|z_t = z\right] - f(z)}{\epsilon}\\ &= \langle\phi(z), \nabla f(z)\rangle + \lambda(z)\int\left[f(z') - f(z)\right]Q(z, dz').\end{aligned}$$

- For ρ-invariance, we need

$$\int \rho(dz)\,\mathcal{L}f(z) = 0.$$

- Sufficient conditions
  - **C1** - The event rate λ satisfies

$$\lambda(\mathcal{S}(z)) - \lambda(z) = \sum_{i=1}^{n}\partial_i\phi_i(z) - \langle\nabla H(z), \phi(z)\rangle.$$

PDMP for Sampling

- The generator of a PDMP is given by

$$\begin{aligned}\mathcal{L}f(z) &= \lim_{\epsilon\to 0}\frac{\mathbb{E}\left[f\left(z_{t+\epsilon}\right)|z_t = z\right] - f(z)}{\epsilon} \\ &= \langle \phi(z), \nabla f(z)\rangle + \lambda(z)\int \left[f(z') - f(z)\right] Q\left(z, dz'\right).\end{aligned}$$

- For ρ-invariance, we need

$$\int \rho(dz)\,\mathcal{L}f(z) = 0.$$

- Sufficient conditions
  - **C1** - The event rate λ satisfies

$$\lambda(\mathcal{S}(z)) - \lambda(z) = \sum_{i=1}^{n}\partial_i\phi_i(z) - \langle \nabla H(z), \phi(z)\rangle.$$

  - **C2** - For the flip operator $\mathcal{S}(z) = (x, -v)$, Q satisfies

$$\int \rho(dz)\,\lambda(z)\,Q\left(z, dz'\right) = \rho\left(\mathcal{S}\left(dz'\right)\right)\lambda\left(\mathcal{S}\left(z'\right)\right).$$

Bouncy Particle Sampler (Peters & De With, 2012)

- $\phi(z) = (v, 0_d)$: linear dynamics.

Bouncy Particle Sampler (Peters & De With, 2012)

- $\phi(z) = (v, 0_d)$: linear dynamics.
- $\lambda(z) = \lambda_{\text{ref}} + \langle \nabla U(x), v \rangle_+$ where $[a]_+ := \max(0, a)$.

Bouncy Particle Sampler (Peters & De With, 2012)

- $\phi(z) = (v, 0_d)$: linear dynamics.
- $\lambda(z) = \lambda_{\text{ref}} + \langle \nabla U(x), v \rangle_+$ where $[a]_+ := \max(0, a)$.
- The kernel Q satisfies

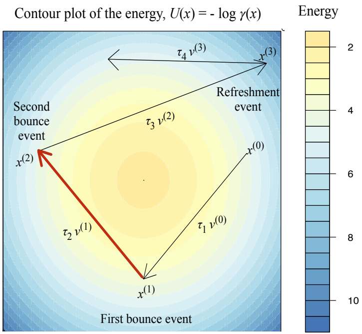
$$Q(z, dz') = \frac{\lambda_{\text{ref}}}{\lambda(z)} \delta_x(dx') \psi(dv') + \frac{\langle \nabla U(x), v \rangle_+}{\lambda(z)} \delta_x(dx') \delta_{R_{\nabla U(x)} v}(dv'),$$

where

$$R_{\nabla U}(x) v := v - 2 \frac{\langle \nabla U(x), v \rangle}{|\nabla U(x)|^2} \nabla U(x)$$

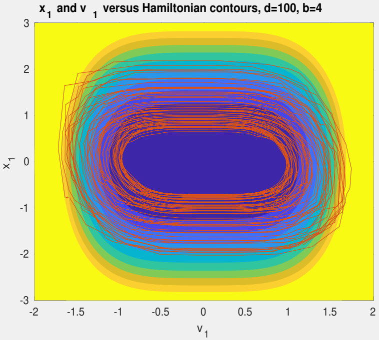
corresponds to a reflection on the hyperplane tangential to ∇U.

Bouncy Particle Sampler Path



A trajectory of BPS on a 2d isotropic normal

Bouncy Particle Sampler Path in High Dimensions



In high-dimensions, BPS converges towards randomized HMC (Deligiannidis et al., 2018).

Local Bouncy Particle Sampler: Graphical Models

- Graphical models: $U(x) = \sum_{i=1}^{n} U_i(x)$ where $U_i(x) = U_i(x_{S_i})$ depends only subset x_{S_i} of components of $x = (x_1, ..., x_d)$.

Local Bouncy Particle Sampler: Graphical Models

- Graphical models: $U(x) = \sum_{i=1}^{n} U_i(x)$ where $U_i(x) = U_i(x_{S_i})$ depends only subset x_{S_i} of components of $x = (x_1, ..., x_d)$.
- Gibbs sampling samples iterately from $\pi(x_{S_i} | x_{-S_i})$: blocking.

Local Bouncy Particle Sampler: Graphical Models

- Graphical models: $U(x) = \sum_{i=1}^{n} U_i(x)$ where $U_i(x) = U_i(x_{S_i})$ depends only subset x_{S_i} of components of $x = (x_1, ..., x_d)$.
- Gibbs sampling samples iterately from $\pi(x_{S_i} | x_{-S_i})$: blocking.
- Local BPS uses

$$\lambda(z) = \sum_{i=1}^{n} \lambda_i(z) = \sum_{i=1}^{n} \langle \nabla U_i(x), v \rangle_+ ,$$

$$Q(z, dz') = \sum_{i=1}^{n} \frac{\lambda_i(z)}{\lambda(z)} \delta_x(dx') \delta_{R_{\nabla U_i(x)} v}(dv').$$

Local Bouncy Particle Sampler: Graphical Models

- Graphical models: $U(x) = \sum_{i=1}^{n} U_i(x)$ where $U_i(x) = U_i(x_{S_i})$ depends only subset x_{S_i} of components of $x = (x_1, ..., x_d)$.
- Gibbs sampling samples iterately from $\pi\left(x_{S_i} | x_{-S_i}\right)$: blocking.
- Local BPS uses

$$\lambda(z) = \sum_{i=1}^{n} \lambda_i(z) = \sum_{i=1}^{n} \langle \nabla U_i(x), v \rangle_+ ,$$

$$Q\left(z, dz'\right) = \sum_{i=1}^{n} \frac{\lambda_i(z)}{\lambda\left(z\right)} \delta_x(dx') \delta_{R_{\nabla U_i}(x) v}(dv').$$

- **Superposition implementation**: sample arrival times of PP for factor i of intensity $\lambda_i(z)$ and apply $v' \leftarrow R_{\nabla U_i}(x) v$.

Local Bouncy Particle Sampler: Graphical Models

- Graphical models: $U(x) = \sum_{i=1}^{n} U_i(x)$ where $U_i(x) = U_i(x_{S_i})$ depends only subset x_{S_i} of components of $x = (x_1, ..., x_d)$.
- Gibbs sampling samples iterately from $\pi\left(x_{S_i} | x_{-S_i}\right)$: blocking.
- Local BPS uses

$$\lambda(z) = \sum_{i=1}^{n} \lambda_i(z) = \sum_{i=1}^{n} \langle \nabla U_i(x), v \rangle_+ ,$$

$$Q\left(z, dz'\right) = \sum_{i=1}^{n} \frac{\lambda_i(z)}{\lambda(z)} \delta_x(dx') \delta_{R_{\nabla U_i}(x)v}(dv').$$

- **Superposition implementation**: sample arrival times of PP for factor i of intensity $\lambda_i(z)$ and apply $v' \leftarrow R_{\nabla U_i}(x)v$.
- Reflection $R_{\nabla U_i}$ only requires updating components x_{S_i} and recomputing arrival times for factors $\{j : x_{S_i} \cap x_{S_j} \neq \emptyset\}$.

Local Bouncy Particle Sampler: Graphical Models

- **Thinning implementation**:

Local Bouncy Particle Sampler: Graphical Models

- **Thinning implementation**:
  - Sample first arrival time of a PP of intensity $\bar{\lambda} = \sum_{i=1}^{n} \bar{\lambda}_i$ with $\bar{\lambda}_i \geq \lambda_i(z)$.

Local Bouncy Particle Sampler: Graphical Models

- **Thinning implementation**:
  - Sample first arrival time of a PP of intensity $\bar{\lambda} = \sum_{i=1}^{n} \bar{\lambda}_i$ with $\bar{\lambda}_i \geq \lambda_i(z)$.
  - Sample $I \in [n]$ with proba $\bar{\lambda}_I / \bar{\lambda}$.

Local Bouncy Particle Sampler: Graphical Models

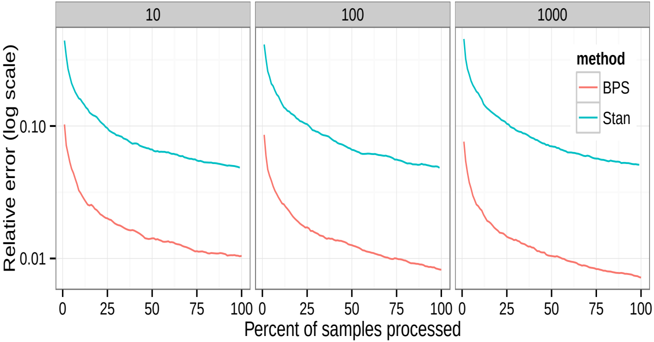
- **Thinning implementation**:
  - Sample first arrival time of a PP of intensity $\bar{\lambda} = \sum_{i=1}^{n} \bar{\lambda}_i$ with $\bar{\lambda}_i \geq \lambda_i(z)$.
  - Sample $I \in [n]$ with proba $\bar{\lambda}_I / \bar{\lambda}$.
  - With proba $\lambda_I(z)/\bar{\lambda}_I$

Local Bouncy Particle Sampler: Graphical Models

- **Thinning implementation**:
  - Sample first arrival time of a PP of intensity $\bar{\lambda} = \sum_{i=1}^{n} \bar{\lambda}_i$ with $\bar{\lambda}_i \geq \lambda_i(z)$.
  - Sample $I \in [n]$ with proba $\bar{\lambda}_I / \bar{\lambda}$.
  - With proba $\lambda_I(z)/\bar{\lambda}_I$
    - Reflection $R_{\nabla U_I}$ only requires updating components x_{S_I} and recomputing arrival times for factors $\{j : x_{S_I} \cap x_{S_j} \neq \emptyset\}$.

- **Thinning implementation**:
  - Sample first arrival time of a PP of intensity $\bar{\lambda} = \sum_{i=1}^{n} \bar{\lambda}_i$ with $\bar{\lambda}_i \geq \lambda_i(z)$.
  - Sample $I \in [n]$ with proba $\bar{\lambda}_I / \bar{\lambda}$.
  - With proba $\lambda_I(z)/\bar{\lambda}_I$
    - Reflection $R_{\nabla U_I}$ only requires updating components x_{S_I} and recomputing arrival times for factors $\{j : x_{S_I} \cap x_{S_j} \neq \emptyset\}$.
- Efficient implementation via alias method (Bouchard-Côté et al. 2016; Kapfer & Krauth, 2016).

Experimental Results on Poisson-Gaussian Random Fields



Relative error for BPS vs HMC for $d = 10$ (left), $d = 100$ (middle) and $d = 1000$ (right) at fixed computational budget

Continuous-time PDMP

- Continuous-time PDMP are useful but have severe limitations.

Continuous-time PDMP

- Continuous-time PDMP are useful but have severe limitations.

- Technical machinery non-standard to most MCMC practitioners.

Continuous-time PDMP

- Continuous-time PDMP are useful but have severe limitations.
- Technical machinery non-standard to most MCMC practitioners.
- Is it possible to derive more flexible discrete-time schemes?

Continuous-time PDMP

- Continuous-time PDMP are useful but have severe limitations.
- Technical machinery non-standard to most MCMC practitioners.
- Is it possible to derive more flexible discrete-time schemes?
- Is it possible to obtain discrete-time schemes enjoying similar features as continuous-time schemes?

Discrete-time PDMP

- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$ and define $\Phi^0(z) = z$ and $\Phi^{r+1}(z) = \Phi^r \circ \Phi(z)$ for $r \in \mathbb{N}$.

Discrete-time PDMP

- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$ and define $\Phi^0(z) = z$ and $\Phi^{r+1}(z) = \Phi^r \circ \Phi(z)$ for $r \in \mathbb{N}$.

- *Acceptance probability* $\alpha : \mathcal{Z} \to [0, 1]$ with $1 - \alpha(z)$ being the proba of having an event at the next time step when the current state is z.

Discrete-time PDMP

- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$ and define $\Phi^0(z) = z$ and $\Phi^{r+1}(z) = \Phi^r \circ \Phi(z)$ for $r \in \mathbb{N}$.
- *Acceptance probability* $\alpha : \mathcal{Z} \to [0,1]$ with $1 - \alpha(z)$ being the proba of having an event at the next time step when the current state is z.
- *Markov kernel* Q used to sample state at event time t is given is given by $z_t \sim Q(z_{t-1}, \cdot)$.

Discrete-time PDMP

- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$ and define $\Phi^0(z) = z$ and $\Phi^{r+1}(z) = \Phi^r \circ \Phi(z)$ for $r \in \mathbb{N}$.

- *Acceptance probability* $\alpha : \mathcal{Z} \to [0, 1]$ with $1 - \alpha(z)$ being the proba of having an event at the next time step when the current state is z.

- *Markov kernel* Q used to sample state at event time t is given is given by $z_t \sim Q(z_{t-1}, \cdot)$.

- (Φ, α, Q) defines a Markov transition kernel

$$K(z, dz') = \alpha(z)\, \delta_{\Phi(z)}(dz') + (1 - \alpha(z))\, Q(z, dz').$$

Simulation of a discrete-time PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.

Simulation of a discrete-time PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.
- For $k = 1, 2, \ldots$ do

Simulation of a discrete-time PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.
- For $k = 1, 2, \ldots$ do
  - Sample inter-event time τ_k, where τ_k is a non-negative integer-valued random variable such that

$$\mathbb{P}\left(\tau_k = j\right) = \left\{1 - \alpha\left(\Phi^j\left(z_{t_{k-1}}\right)\right)\right\} \prod_{i=0}^{j-1} \alpha\left(\Phi^i\left(z_{t_{k-1}}\right)\right).$$

Simulation of a discrete-time PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.
- For $k = 1, 2, \ldots$ do
  - Sample inter-event time τ_k, where τ_k is a non-negative integer-valued random variable such that

$$\mathbb{P}\left(\tau_k = j\right) = \left\{1 - \alpha\left(\Phi^j\left(z_{t_{k-1}}\right)\right)\right\} \prod_{i=0}^{j-1} \alpha\left(\Phi^i\left(z_{t_{k-1}}\right)\right).$$

  - If $\tau_k \geq 1$ then for $r \in \{1, ..., \tau_k\}$, set

$$z_{t_{k-1}+r} \leftarrow \Phi^r(z_{t_{k-1}}).$$

Simulation of a discrete-time PDMP

- Initialize z_0 arbitrarily on $\mathcal{Z}$ and set $t_0 \leftarrow 0$.
- For $k = 1, 2, \ldots$ do
  - Sample inter-event time τ_k, where τ_k is a non-negative integer-valued random variable such that

$$\mathbb{P}(\tau_k = j) = \left\{1 - \alpha\left(\Phi^j\left(z_{t_{k-1}}\right)\right)\right\} \prod_{i=0}^{j-1} \alpha\left(\Phi^i\left(z_{t_{k-1}}\right)\right).$$

  - If $\tau_k \geq 1$ then for $r \in \{1, ..., \tau_k\}$, set

$$z_{t_{k-1}+r} \leftarrow \Phi^r(z_{t_{k-1}}).$$

  - Set $t_k \leftarrow t_{k-1} + \tau_k + 1$ and sample

$$z_{t_k} \sim Q(z_{t_k-1}, \cdot).$$

Sufficient conditions for invariance

- We want
$$\int \rho(dz)\,K(z,dz') = \rho(dz')\,.$$

Sufficient conditions for invariance

- We want
$$\int \rho(dz)\, K(z, dz') = \rho(dz')\,.$$
- **C1** - The acceptance probability α satisfies
$$\{-\log \alpha(\mathcal{S} \circ \Phi(z))\} - \{-\log \alpha(z)\} = \log |\nabla \Phi(z)| - \{H(\Phi(z)) - H(z)\}\,.$$

Sufficient conditions for invariance

- We want
$$\int \rho\left(dz\right) K\left(z, dz'\right) = \rho\left(dz'\right).$$

- **C1** - The acceptance probability α satisfies
$$\left\{-\log\alpha\left(\mathcal{S}\circ\Phi\left(z\right)\right)\right\} - \left\{-\log\alpha\left(z\right)\right\} = \log\left|\nabla\Phi\left(z\right)\right| - \left\{H\left(\Phi\left(z\right)\right) - H\left(z\right)\right\}.$$

- **C2** -The kernel Q satisfies
$$\int \rho\left(dz\right)\left(1-\alpha\left(z\right)\right) Q\left(z, dz'\right) = \rho\left(\mathcal{S}\left(dz'\right)\right)\left(1-\alpha\left(\mathcal{S}\left(z'\right)\right)\right).$$

Generalized MH

- Consider the target distribution $\nu(dz) \propto \rho(dz)(1-\alpha(z))$ and proposal $M(z, dz')$.

Generalized MH

- Consider the target distribution $\nu(dz) \propto \rho(dz)(1-\alpha(z))$ and proposal $M(z, dz')$.
- Let Q be defined as

$$Q(z, dz') = \beta(z, z') M(z, dz') + \left\{1 - \int \beta(z, w) M(z, dw)\right\} \delta_{\mathcal{S}(z)}(dz')$$

where

$$\beta(z, z') = \min\left(1, \frac{\nu(\mathcal{S}(dz')) M(\mathcal{S}(z'), \mathcal{S}(dz))}{\nu(dz) M(z, dz')}\right)$$

Generalized MH

- Consider the target distribution $\nu(dz) \propto \rho(dz)(1-\alpha(z))$ and proposal $M(z, dz')$.
- Let Q be defined as

$$Q(z, dz') = \beta(z, z') M(z, dz') + \left\{1 - \int \beta(z, w) M(z, dw)\right\} \delta_{\mathcal{S}(z)}(dz')$$

where

$$\beta(z, z') = \min\left(1, \frac{\nu(\mathcal{S}(dz')) M(\mathcal{S}(z'), \mathcal{S}(dz))}{\nu(dz) M(z, dz')}\right)$$

- This satisfes the *skewed detailed balance*

$$\nu(dz) Q(z, dz') = \nu(\mathcal{S}(dz')) Q(\mathcal{S}(z'), \mathcal{S}(dz))$$

thus condition **C2.**

Generalized MH

- Consider the target distribution $\nu(dz) \propto \rho(dz)(1-\alpha(z))$ and proposal $M(z, dz')$.
- Let Q be defined as

$$Q(z, dz') = \beta(z, z') M(z, dz') + \left\{1 - \int \beta(z, w) M(z, dw)\right\} \delta_{\mathcal{S}(z)}(dz')$$

where

$$\beta(z, z') = \min\left(1, \frac{\nu(\mathcal{S}(dz')) M(\mathcal{S}(z'), \mathcal{S}(dz))}{\nu(dz) M(z, dz')}\right)$$

- This satisfes the *skewed detailed balance*

$$\nu(dz) Q(z, dz') = \nu(\mathcal{S}(dz')) Q(\mathcal{S}(z'), \mathcal{S}(dz))$$

thus condition **C2.**
- For $M(z, dz') = \delta_{\Psi(z)}(dz')$, this kernel is well-defined if Ψ admits an inverse $\Psi^{-1} = \mathcal{S} \circ \Psi \circ \mathcal{S}$ and $\beta(z, z') = \beta(z) = \min\left(1, \frac{\nu(\mathcal{S} \circ \Psi(dz))}{\nu(dz)}\right)$.

Existing PD-MCMC algorithms and Extensions

- *Guided random walk* (Gustfason 1998): $\Phi(z) = (x + \epsilon v, v)$,
  $\alpha(z) = \min\{1, \rho(\Phi(z)) / \rho(z)\} = \min\{1, \pi(x + v\epsilon) / \pi(x)\}$,
  $Q(z, dz') = \delta_{\mathcal{S}(z)}(dz')$.

Existing PD-MCMC algorithms and Extensions

- *Guided random walk* (Gustfason 1998): $\Phi(z) = (x + \epsilon v, v)$, $\alpha(z) = \min\{1, \rho(\Phi(z))/\rho(z)\} = \min\{1, \pi(x + v\epsilon)/\pi(x)\}$, $Q(z, dz') = \delta_{\mathcal{S}(z)}(dz')$.
- *Hamiltonian Monte Carlo* (Duane et al. 1987): $\Phi(z)$ leapfrog integrator, $\alpha(z) = \min\{1, \rho(\Phi(z))/\rho(z)\}$, $Q(z, dz') = \delta_{\mathcal{S}(z)}(dz')$.

Existing PD-MCMC algorithms and Extensions

- *Guided random walk* (Gustfason 1998): $\Phi(z) = (x + \epsilon v, v)$, $\alpha(z) = \min\{1, \rho(\Phi(z))/\rho(z)\} = \min\{1, \pi(x + v\epsilon)/\pi(x)\}$, $Q(z, dz') = \delta_{\mathcal{S}(z)}(dz')$.
- *Hamiltonian Monte Carlo* (Duane et al. 1987): $\Phi(z)$ leapfrog integrator, $\alpha(z) = \min\{1, \rho(\Phi(z))/\rho(z)\}$, $Q(z, dz') = \delta_{\mathcal{S}(z)}(dz')$.
- *Reflective Slice Sampling* (Neal 2003): $\Phi(z) = (x + \epsilon v, v)$ and

$$Q(z, dz') = \beta(z)\,\delta_{(x, R_{\nabla U}(x)v)}(dz') + \{1 - \beta(z)\}\,\delta_{\mathcal{S}(z)}(dz')$$

where

$$\beta(z) = \min\left\{1, \frac{[\pi(x) - \pi(x - \epsilon R_{\nabla U}(x)v))]_+}{[\pi(x) - \pi(x + \epsilon v)]}\right\}.$$

As $\epsilon \to 0$, we have $\beta(z) \to 1$ and the algorithm converges to BPS.

Existing PD-MCMC algorithms and Extensions

- *Guided random walk* (Gustfason 1998): $\Phi(z) = (x + \epsilon v, v)$, $\alpha(z) = \min\{1, \rho(\Phi(z))/\rho(z)\} = \min\{1, \pi(x + v\epsilon)/\pi(x)\}$, $Q(z, dz') = \delta_{\mathcal{S}(z)}(dz')$.
- *Hamiltonian Monte Carlo* (Duane et al. 1987): $\Phi(z)$ leapfrog integrator, $\alpha(z) = \min\{1, \rho(\Phi(z))/\rho(z)\}$, $Q(z, dz') = \delta_{\mathcal{S}(z)}(dz')$.
- *Reflective Slice Sampling* (Neal 2003): $\Phi(z) = (x + \epsilon v, v)$ and

$$Q(z, dz') = \beta(z)\,\delta_{(x, R_{\nabla U}(x)v)}(dz') + \{1 - \beta(z)\}\,\delta_{\mathcal{S}(z)}(dz')$$

where

$$\beta(z) = \min\left\{1, \frac{[\pi(x) - \pi(x - \epsilon R_{\nabla U}(x)v))]_+}{[\pi(x) - \pi(x + \epsilon v)]}\right\}.$$

As $\epsilon \to 0$, we have $\beta(z) \to 1$ and the algorithm converges to BPS.

- Randomized bounces & gradient-free algorithms can be derived (Sherlock & Thiery, 2017, Vanetti et al., 2017).

Implementation

- Almost all implementations of discrete-time schemes consist of sampling a Bernoulli of parameter $1 - \alpha(z)$ when in state z.

Implementation

- Almost all implementations of discrete-time schemes consist of sampling a Bernoulli of parameter $1 - \alpha(z)$ when in state z.

- Exact sampling of event time for convex U is feasible.

Implementation

- Almost all implementations of discrete-time schemes consist of sampling a Bernoulli of parameter $1 - \alpha(z)$ when in state z.
- Exact sampling of event time for convex U is feasible.
- Equivalent of thinning and superpositions for Poisson processes can be used:

Implementation

- Almost all implementations of discrete-time schemes consist of sampling a Bernoulli of parameter $1-\alpha(z)$ when in state z.
- Exact sampling of event time for convex U is feasible.
- Equivalent of thinning and superpositions for Poisson processes can be used:
  - "Thinning": If $\exists\ \bar{\alpha} : \mathcal{Z} \to (0,1]$ s.t. $\alpha\left(\Phi^k(z)\right) \geq \bar{\alpha}(z,k) \geq \bar{\alpha}(z)$ then sample a candidate event time from $\bar{\mathbb{P}}(\tau = j) = \{1-\bar{\alpha}(z)\}\,\bar{\alpha}^{j-1}$ and accept/reject.

Implementation

- Almost all implementations of discrete-time schemes consist of sampling a Bernoulli of parameter $1 - \alpha(z)$ when in state z.

- Exact sampling of event time for convex U is feasible.

- Equivalent of thinning and superpositions for Poisson processes can be used:
  - "Thinning": If $\exists\ \bar{\alpha} : \mathcal{Z} \to (0,1]$ s.t. $\alpha\left(\Phi^k(z)\right) \geq \bar{\alpha}(z,k) \geq \bar{\alpha}(z)$ then sample a candidate event time from $\bar{\mathbb{P}}(\tau = j) = \{1 - \bar{\alpha}(z)\}\, \bar{\alpha}^{j-1}$ and accept/reject.

  - "Superposition then thinning": If $\alpha(z) = \min\{1, \rho(\Phi(z))/\rho(z)\}$ for $\rho(z) = \prod_{i=1}^n \rho_i(z)$ then $\bar{\alpha}(z,k) = \prod_{i=1}^n \min\left\{1, \rho_i\left(\Phi^{k+1}(z)\right)/\rho_i\left(\Phi^k(z)\right)\right\}$ is a lower bound.

Discrete-time local PDMP

- **Motivation**: exploit $H(z) = \sum_{i=1}^{n} H_i(z)$, where $H_i(z)$ might only depend on a subset of components of z.

Discrete-time local PDMP

- **Motivation**: exploit $H(z) = \sum_{i=1}^{n} H_i(z)$, where $H_i(z)$ might only depend on a subset of components of z.
- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$.

Discrete-time local PDMP

- **Motivation**: exploit $H(z) = \sum_{i=1}^{n} H_i(z)$, where $H_i(z)$ might only depend on a subset of components of z.
- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$.
- *Acceptance probability* $\alpha : \mathcal{Z} \to [0, 1]$ where

$$\alpha(z) = \prod_{i=1}^{n} \alpha_i(z)$$

  with $\alpha_i : \mathcal{Z} \to [0, 1]$ and define $B_i \overset{ind}{\sim} \mathrm{Ber}(1 - \alpha_i(z))$. For $B := (B_1, \ldots B_n)$ and $|B| := \sum_{i=1}^{n} B_i$, $\mathbb{P}(|B| \geq 1) = 1 - \alpha(z)$.

Discrete-time local PDMP

- **Motivation**: exploit $H(z) = \sum_{i=1}^{n} H_i(z)$, where $H_i(z)$ might only depend on a subset of components of z.
- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$.
- *Acceptance probability* $\alpha : \mathcal{Z} \to [0,1]$ where

$$\alpha(z) = \prod_{i=1}^{n} \alpha_i(z)$$

  with $\alpha_i : \mathcal{Z} \to [0,1]$ and define $B_i \overset{ind}{\sim} \mathrm{Ber}(1 - \alpha_i(z))$. For $B := (B_1, \dots B_n)$ and $|B| := \sum_{i=1}^{n} B_i$, $\mathbb{P}(|B| \geq 1) = 1 - \alpha(z)$.
- *Markov kernel*: conditional upon B, where $|B| \geq 1$, sample $Z' \sim Q_B(Z, \cdot)$ so

$$Q(z, dz') = \sum_{b \in \mathcal{B}} \mathbb{P}(B = b | z, |B| \geq 1) Q_b(z, dz'),$$

  where

$$\mathbb{P}(B = b | z, |B| \geq 1) = \frac{\prod_{i=1}^{n} \mathrm{Ber}(b_i; 1 - \alpha_i(z))}{1 - \alpha(z)} \mathbb{I}\left(\sum_{i=1}^{n} b_i \geq 1\right).$$

Sampling a discrete-time local PDMP

- ***Sampling using Bernoulli trials***

Sampling a discrete-time local PDMP

- ***Sampling using Bernoulli trials***
  - For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\,\{1 - \alpha_i(z)\}$.

Sampling a discrete-time local PDMP

- ***Sampling using Bernoulli trials***
  - For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\,\{1 - \alpha_i(z)\}$.
  - If $|B| = 0$ then set $z' \leftarrow \Phi(z)$.

Sampling a discrete-time local PDMP

- ***Sampling using Bernoulli trials***
  - For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{1 - \alpha_i(z)\right\}$.
  - If $|B| = 0$ then set $z' \leftarrow \Phi(z)$.
  - If $|B| \geq 1$ then sample $z' \sim Q_B\left(z, \cdot\right)$.

Sampling a discrete-time local PDMP

- ***Sampling using Bernoulli trials***
  - For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{1 - \alpha_i(z)\right\}$.
  - If $|B| = 0$ then set $z' \leftarrow \Phi(z)$.
  - If $|B| \geq 1$ then sample $z' \sim Q_B\left(z, \cdot\right)$.
- ***Sampling using Bernoulli processes***

Sampling a discrete-time local PDMP

- ***Sampling using Bernoulli trials***
  - For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{1 - \alpha_i(z)\right\}$.
  - If $|B| = 0$ then set $z' \leftarrow \Phi(z)$.
  - If $|B| \geq 1$ then sample $z' \sim Q_B\left(z, \cdot\right)$.
- ***Sampling using Bernoulli processes***
  - For $i \in [n]$, sample τ_i s.t. $\mathbb{P}(\tau_i = j) = \left\{1 - \alpha_i\left(\Phi^j\left(z_{t_{k-1}}\right)\right)\right\} \prod_{i=0}^{j-1} \alpha_i\left(\Phi^i\left(z_{t_{k-1}}\right)\right)$.

Sampling a discrete-time local PDMP

- ***Sampling using Bernoulli trials***
  - For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\{1 - \alpha_i(z)\}$.
  - If $|B| = 0$ then set $z' \leftarrow \Phi(z)$.
  - If $|B| \geq 1$ then sample $z' \sim Q_B(z, \cdot)$.
- ***Sampling using Bernoulli processes***
  - For $i \in [n]$, sample τ_i s.t. $\mathbb{P}(\tau_i = j) = \left\{1 - \alpha_i\left(\Phi^j\left(z_{t_{k-1}}\right)\right)\right\}\prod_{i=0}^{j-1}\alpha_i\left(\Phi^i\left(z_{t_{k-1}}\right)\right)$.
  - If $\tau := \min \tau_i \geq 1$ then set $z_{t_{k-1}+r} \leftarrow \Phi^r(z_{t_{k-1}})$ for $r \in \{1, ..., \tau\}$.

Sampling a discrete-time local PDMP

- ***Sampling using Bernoulli trials***
  - For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{1 - \alpha_i(z)\right\}$.
  - If $|B| = 0$ then set $z' \leftarrow \Phi(z)$.
  - If $|B| \geq 1$ then sample $z' \sim Q_B\left(z, \cdot\right)$.
- ***Sampling using Bernoulli processes***
  - For $i \in [n]$, sample τ_i s.t. $\mathbb{P}(\tau_i = j) = \left\{1 - \alpha_i\left(\Phi^j\left(z_{t_{k-1}}\right)\right)\right\} \prod_{i=0}^{j-1} \alpha_i\left(\Phi^i\left(z_{t_{k-1}}\right)\right)$.
  - If $\tau := \min \tau_i \geq 1$ then set $z_{t_{k-1}+r} \leftarrow \Phi^r(z_{t_{k-1}})$ for $r \in \{1, ..., \tau\}$.
  - Set $t_k \leftarrow t_{k-1} + \tau + 1$ and sample $z_{t_k} \sim Q_B(z_{t_k - 1}, \cdot)$.

Sufficient conditions for invariance

- **C1** - For mappings such that $|\nabla\Phi| = 1$, the acceptance probabilities α_i satisfy for all $i \in [n]$

$$\{-\log \alpha_i\left(\mathcal{S}\circ\Phi\left(z\right)\right)\} - \{-\log \alpha_i\left(z\right)\} = -\left\{H_i\left(\Phi\left(z\right)\right) - H_i\left(z\right)\right\}.$$

Sufficient conditions for invariance

- **C1** - For mappings such that $|\nabla\Phi| = 1$, the acceptance probabilities α_i satisfy for all $i \in [n]$

$$\{-\log \alpha_i\left(\mathcal{S}\circ\Phi\left(z\right)\right)\} - \{-\log \alpha_i\left(z\right)\} = -\left\{H_i\left(\Phi\left(z\right)\right) - H_i\left(z\right)\right\}.$$

- **C2** - For all $b \in \mathcal{B}$, the transition kernel Q_b satisfies

$$\begin{aligned} & \int \rho\left(dz\right)\left(1-\alpha\left(z\right)\right)\mathbb{P}(B=b|z,|B|\geq 1)Q_b\left(z,dz'\right) \\ = \; & \rho(\mathcal{S}\left(dz'\right))\left(1-\alpha\left(\mathcal{S}\left(z'\right)\right)\right)\mathbb{P}(B=b|\mathcal{S}\left(z'\right),|B|\geq 1). \end{aligned}$$

Sufficient conditions for invariance

- **C1** - For mappings such that $|\nabla\Phi| = 1$, the acceptance probabilities α_i satisfy for all $i \in [n]$

$$\{-\log\alpha_i(\mathcal{S}\circ\Phi(z))\} - \{-\log\alpha_i(z)\} = -\{H_i(\Phi(z)) - H_i(z)\}.$$

- **C2** - For all $b \in \mathcal{B}$, the transition kernel Q_b satisfies

$$\begin{aligned} & \int \rho(dz)(1-\alpha(z))\mathbb{P}(B=b|z,|B|\geq 1)Q_b(z,dz') \\ = \quad & \rho(\mathcal{S}(dz'))(1-\alpha(\mathcal{S}(z')))\mathbb{P}(B=b|\mathcal{S}(z'),|B|\geq 1). \end{aligned}$$

- Condition **C2** is satisfied if Q_b satisfies a skewed detailed balance w.r.t. $\rho(dz)(1-\alpha(z))\mathbb{Q}_{|B|\geq 1}(b|z)$.

Discrete-time local BPS

- Discrete-time version of local BPS using $\Phi(z) = (x + \epsilon v, v)$ (Peters & De With, 2012; Bouchard-Côté et al., 2015).

Discrete-time local BPS

- Discrete-time version of local BPS using $\Phi(z) = (x + \epsilon v, v)$ (Peters & De With, 2012; Bouchard-Côté et al., 2015).
- Consider $U(x) = \sum_{i=1}^{n} U_i(x)$ and no refreshment for simplicity.

Discrete-time local BPS

- Discrete-time version of local BPS using $\Phi(z) = (x + \epsilon v, v)$ (Peters & De With, 2012; Bouchard-Côté et al., 2015).
- Consider $U(x) = \sum_{i=1}^{n} U_i(x)$ and no refreshment for simplicity.
- Set $H_i(z) = U_i(x)$ for $i \in [n]$ and $H_{n+1}(z) = \frac{1}{2} v^t v$.

Discrete-time local BPS

- Discrete-time version of local BPS using $\Phi(z) = (x + \epsilon v, v)$ (Peters & De With, 2012; Bouchard-Côté et al., 2015).
- Consider $U(x) = \sum_{i=1}^{n} U_i(x)$ and no refreshment for simplicity.
- Set $H_i(z) = U_i(x)$ for $i \in [n]$ and $H_{n+1}(z) = \frac{1}{2} v^t v$.
- Set $\alpha_i(z) = \min\{1, \pi_i(x + v\epsilon)/\pi_i(x)\}$ for $\pi_i(x) \propto \exp(-U_i(x))$.

Discrete-time local BPS

- Discrete-time version of local BPS using $\Phi(z) = (x + \epsilon v, v)$ (Peters & De With, 2012; Bouchard-Côté et al., 2015).
- Consider $U(x) = \sum_{i=1}^{n} U_i(x)$ and no refreshment for simplicity.
- Set $H_i(z) = U_i(x)$ for $i \in [n]$ and $H_{n+1}(z) = \frac{1}{2} v^t v$.
- Set $\alpha_i(z) = \min\{1, \pi_i(x + v\epsilon)/\pi_i(x)\}$ for $\pi_i(x) \propto \exp(-U_i(x))$.
- For all $b \in \mathcal{B}$, Q_b uses deterministic reflection w.r.t.

$$\nabla \overline{U}(x) := \sum_{i:b_i=1} \nabla U_i(x).$$

Discrete-time local BPS

- Discrete-time version of local BPS using $\Phi(z) = (x + \epsilon v, v)$ (Peters & De With, 2012; Bouchard-Côté et al., 2015).
- Consider $U(x) = \sum_{i=1}^{n} U_i(x)$ and no refreshment for simplicity.
- Set $H_i(z) = U_i(x)$ for $i \in [n]$ and $H_{n+1}(z) = \frac{1}{2} v^t v$.
- Set $\alpha_i(z) = \min\{1, \pi_i(x + v\epsilon)/\pi_i(x)\}$ for $\pi_i(x) \propto \exp(-U_i(x))$.
- For all $b \in \mathcal{B}$, Q_b uses deterministic reflection w.r.t.

$$\nabla \overline{U}(x) := \sum_{i:b_i=1} \nabla U_i(x).$$

- Same idea provides a discrete-time version of multidimensional Zig-Zag (Bierkens et al. 2016).

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{\left[\pi_i\left(x\right) - \pi_i\left(x + \epsilon v\right)\right]_+ / \pi_i\left(x\right)\right\}$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{\left[\pi_i\left(x\right) - \pi_i\left(x + \epsilon v\right)\right]_+ / \pi_i\left(x\right)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + \epsilon v, v)$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{ \left[\pi_i\left(x\right) - \pi_i\left(x + \epsilon v\right)\right]_+ / \pi_i\left(x\right) \right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + \epsilon v, v)$.
- If $|B| \geq 0$, then

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{\left[\pi_i\left(x\right) - \pi_i\left(x + \epsilon v\right)\right]_+ / \pi_i\left(x\right)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + \epsilon v, v)$.
- If $|B| \geq 0$, then
  - Compute
$$\nabla \overline{U}\left(x\right) := \sum_{i:B_i = \mathbf{1}} \nabla U_i\left(x\right)$$
    and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x)v$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{[\pi_i(x) - \pi_i(x + \epsilon v)]_+ / \pi_i(x)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + \epsilon v, v)$.
- If $|B| \geq 0$, then
  - Compute
  $$\nabla \overline{U}(x) := \sum_{i:B_i=1} \nabla U_i(x)$$
  and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x) v$.
  - With proba
  $$\prod_{i:B_i=0} \min\left\{1, \frac{\min(\pi_i(x), \pi_i(x - \epsilon v^*))}{\min(\pi_i(x), \pi_i(x + \epsilon v))}\right\} \prod_{i:B_i=1} \min\left\{1, \frac{[\pi_i(x) - \pi_i(x - \epsilon v^*)]_+}{[\pi_i(x) - \pi_i(x + \epsilon v)]}\right\},$$
  output $z' = (x, v^*)$, otherwise output $z' = (x, v^*)$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{\left[\pi_i\left(x\right) - \pi_i\left(x + v\epsilon\right)\right]_+ / \pi_i\left(x\right)\right\}$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{\left[\pi_i\left(x\right) - \pi_i\left(x + v\epsilon\right)\right]_+ / \pi_i\left(x\right)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + v\epsilon, v)$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{\left[\pi_i\left(x\right) - \pi_i\left(x + v\epsilon\right)\right]_+ / \pi_i\left(x\right)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + v\epsilon, v)$.
- If $|B| \geq 0$, then

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{[\pi_i(x) - \pi_i(x + v\epsilon)]_+ / \pi_i(x)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + v\epsilon, v)$.
- If $|B| \geq 0$, then
  - Compute
$$\nabla \overline{U}(x) := \sum_{i:B_i=\mathbf{1}} \nabla U_i(x)$$
    and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x)v$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{[\pi_i(x) - \pi_i(x + v\epsilon)]_+ / \pi_i(x)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + v\epsilon, v)$.
- If $|B| \geq 0$, then
  - Compute
  $$\nabla \overline{U}(x) := \sum_{i:B_i=1} \nabla U_i(x)$$
  and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x)v$.
  - For $i \in V$, sample $B_i' \sim \mathrm{Ber}\left(1 - \min\left\{1, \frac{[\pi_i(x) - \pi_i(x - v^*\epsilon)]_+}{[\pi_i(x) - \pi_i(x + v\epsilon)]}\right\}\right)$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{\left[\pi_i\left(x\right) - \pi_i\left(x + v\epsilon\right)\right]_+ / \pi_i\left(x\right)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + v\epsilon, v)$.
- If $|B| \geq 0$, then
  - Compute
    $$\nabla \overline{U}\left(x\right) := \sum_{i:B_i=\mathbf{1}} \nabla U_i\left(x\right)$$
    and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x)v$.
  - For $i \in V$, sample $B_i' \sim \mathrm{Ber}\left(1 - \min\left\{1, \frac{[\pi_i(x) - \pi_i(x - v^*\epsilon)]_+}{[\pi_i(x) - \pi_i(x + v\epsilon)]}\right\}\right)$.
  - For $i \in [m] \setminus V$, sample
    $B_i' \sim \mathrm{Ber}\left(1 - \min\left\{1, \frac{\min\left(\pi_i(x), \pi_i(x - v^*\epsilon)\right)}{\min(\pi_i(x), \pi_i(x + v\epsilon))}\right\}\right)$.

Discrete-time local BPS

- For $i \in [n]$, sample $B_i \sim \mathrm{Ber}\left\{\left[\pi_i\left(x\right) - \pi_i\left(x + v\epsilon\right)\right]_+ / \pi_i\left(x\right)\right\}$.
- If $|B| = 0$ then set $z' \leftarrow (x + v\epsilon, v)$.
- If $|B| \geq 0$, then
  - Compute
  $$\nabla \overline{U}\left(x\right) := \sum_{i:B_i=1} \nabla U_i\left(x\right)$$
  and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x)v$.
  - For $i \in V$, sample $B_i' \sim \mathrm{Ber}\left(1 - \min\left\{1, \frac{[\pi_i(x)-\pi_i(x-v^*\epsilon)]_+}{[\pi_i(x)-\pi_i(x+v\epsilon)]}\right\}\right)$.
  - For $i \in [m] \setminus V$, sample
  $B_i' \sim \mathrm{Ber}\left(1 - \min\left\{1, \frac{\min\left(\pi_i(x),\pi_i(x-v^*\epsilon)\right)}{\min(\pi_i(x),\pi_i(x+v\epsilon))}\right\}\right)$.
  - If $B_i' = 1$ for any $i \in [n]$ then output $z' = (x, -v)$, otherwise output $z' = (x, v^*)$.

Interlude: Fast simulation of Bernoulli random variables

- Let $X_i \sim \mathrm{Ber}(p_i)$ for all $i \in I$ with $p_i \le \bar{p} < 1$ for $i \in [n]$.

Interlude: Fast simulation of Bernoulli random variables

- Let $X_i \sim \mathrm{Ber}(p_i)$ for all $i \in I$ with $p_i \leq \bar{p} < 1$ for $i \in [n]$.
- Direct simulation of $\{X_i; i \in [n]\}$ has expected cost $O(|I|)$ but we can do this in $O(1 + |I|\bar{p})$ once the bounds are computed.

Interlude: Fast simulation of Bernoulli random variables

- Let $X_i \sim \mathrm{Ber}(p_i)$ for all $i \in I$ with $p_i \leq \bar{p} < 1$ for $i \in [n]$.

- Direct simulation of $\{X_i; i \in [n]\}$ has expected cost $O(|I|)$ but we can do this in $O(1 + |I|\bar{p})$ once the bounds are computed.

- **Algorithm**

Interlude: Fast simulation of Bernoulli random variables

- Let $X_i \sim \mathrm{Ber}(p_i)$ for all $i \in I$ with $p_i \leq \bar{p} < 1$ for $i \in [n]$.
- Direct simulation of $\{X_i; i \in [n]\}$ has expected cost $O(|I|)$ but we can do this in $O(1 + |I|\bar{p})$ once the bounds are computed.
- **Algorithm**
    - Sample $N \sim \mathrm{Bin}\left(|I|, \bar{p}\right)$.

Interlude: Fast simulation of Bernoulli random variables

- Let $X_i \sim \mathrm{Ber}(p_i)$ for all $i \in I$ with $p_i \leq \bar{p} < 1$ for $i \in [n]$.

- Direct simulation of $\{X_i; i \in [n]\}$ has expected cost $O(|I|)$ but we can do this in $O(1 + |I|\bar{p})$ once the bounds are computed.

- **Algorithm**
  - Sample $N \sim \mathrm{Bin}\left(|I|, \bar{p}\right)$.

  - Sample N indices $i_1, ..., i_N$ in I uniformly at random without replacement and let $\mathcal{A} := (i_1, ..., i_N)$.

Interlude: Fast simulation of Bernoulli random variables

- Let $X_i \sim \mathrm{Ber}(p_i)$ for all $i \in I$ with $p_i \leq \bar{p} < 1$ for $i \in [n]$.
- Direct simulation of $\{X_i; i \in [n]\}$ has expected cost $O(|I|)$ but we can do this in $O(1 + |I|\bar{p})$ once the bounds are computed.
- **Algorithm**
  - Sample $N \sim \mathrm{Bin}\left(|I|, \bar{p}\right)$.
  - Sample N indices $i_1, ..., i_N$ in I uniformly at random without replacement and let $\mathcal{A} := (i_1, ..., i_N)$.
  - For $i \in \mathcal{A}$, sample $X_i \sim \mathrm{Ber}\left(p_i/\bar{p}\right)$.

Interlude: Fast simulation of Bernoulli random variables

- Let $X_i \sim \mathrm{Ber}(p_i)$ for all $i \in I$ with $p_i \leq \bar{p} < 1$ for $i \in [n]$.
- Direct simulation of $\{X_i; i \in [n]\}$ has expected cost $O(|I|)$ but we can do this in $O(1 + |I|\bar{p})$ once the bounds are computed.
- **Algorithm**
  - Sample $N \sim \mathrm{Bin}\,(|I|, \bar{p})$.
  - Sample N indices $i_1, ..., i_N$ in I uniformly at random without replacement and let $\mathcal{A} := (i_1, ..., i_N)$.
  - For $i \in \mathcal{A}$, sample $X_i \sim \mathrm{Ber}\,(p_i/\bar{p})$.
  - Optional: For $i \in I \setminus \mathcal{A}$, set $X_i \leftarrow 0$.

Subsampling for Metropolis-like chains

- Assume you want to sample from

$$\pi(x) \propto \prod_{i=1}^{n} \pi_i(x).$$

Subsampling for Metropolis-like chains

- Assume you want to sample from

$$\pi(x) \propto \prod_{i=1}^{n} \pi_i(x).$$

- **Algorithm**

Subsampling for Metropolis-like chains

- Assume you want to sample from

$$\pi(x) \propto \prod_{i=1}^{n} \pi_i(x).$$

- **Algorithm**
  - Sample $x^* \sim q(x, \cdot)$ from a symmetric proposal.

Subsampling for Metropolis-like chains

- Assume you want to sample from

$$\pi(x) \propto \prod_{i=1}^{n} \pi_i(x).$$

- **Algorithm**
  - Sample $x^* \sim q(x, \cdot)$ from a symmetric proposal.
  - With proba

$$\alpha(x, x^*) = \prod_{i=1}^{n} \alpha_i(x, x^*), \text{ where } \alpha_i(x, x^*) := \min\left\{1, \frac{\pi_i(x^*)}{\pi_i(x)}\right\},$$

set $x' = x^*$.

Subsampling for Metropolis-like chains

- Assume you want to sample from

$$\pi\left(x\right)\propto\prod_{i=1}^{n}\pi_{i}\left(x\right).$$

- **Algorithm**
  - Sample $x^{*}\sim q\left(x,\cdot\right)$ from a symmetric proposal.
  - With proba

$$\alpha\left(x,x^{*}\right)=\prod_{i=1}^{n}\alpha_{i}\left(x,x^{*}\right),\ \text{where}\ \alpha_{i}\left(x,x^{*}\right):=\min\left\{1,\frac{\pi_{i}\left(x^{*}\right)}{\pi_{i}(x)}\right\},$$

    set $x'=x^{*}$.
  - Otherwise set $x'=x$.

Subsampling for Metropolis-like chains

- Assume you want to sample from

$$\pi(x) \propto \prod_{i=1}^{n} \pi_i(x).$$

- **Algorithm**
  - Sample $x^* \sim q(x, \cdot)$ from a symmetric proposal.
  - With proba

$$\alpha(x, x^*) = \prod_{i=1}^{n} \alpha_i(x, x^*), \text{ where } \alpha_i(x, x^*) := \min\left\{1, \frac{\pi_i(x^*)}{\pi_i(x)}\right\},$$

    set $x' = x^*$.
  - Otherwise set $x' = x$.
- As long as one can upper bound cheaply $\mathbb{P}(B_i = 1) = 1 - \alpha_i(x, x^*)$, subsampling tricks are applicable; e.g. logistic regression.

Subsampling for Metropolis-like chains

- Assume you want to sample from

$$\pi(x) \propto \prod_{i=1}^{n} \pi_i(x).$$

- **Algorithm**
  - Sample $x^* \sim q(x, \cdot)$ from a symmetric proposal.
  - With proba

$$\alpha(x, x^*) = \prod_{i=1}^{n} \alpha_i(x, x^*), \text{ where } \alpha_i(x, x^*) := \min\left\{1, \frac{\pi_i(x^*)}{\pi_i(x)}\right\},$$

    set $x' = x^*$.
  - Otherwise set $x' = x$.
- As long as one can upper bound cheaply $\mathbb{P}(B_i = 1) = 1 - \alpha_i(x, x^*)$, subsampling tricks are applicable; e.g. logistic regression.
- For logistic regression, sufficient conditions for geometric ergodicity presented in (Cornish et al., 2018).

Discrete-time local BPS via Priority Queue

- Initialization: For $i \in [n]$, sample inter-event times τ_i with distribution

$$\max\left(0, 1 - \frac{\pi_i(x + v(\tau_i + 1)\epsilon)}{\pi_i(x + v\tau_i\epsilon)}\right) \prod_{k=0}^{\tau_i - 1} \min\left(1, \frac{\pi_i(x + v(k+1)\epsilon)}{\pi_i(x + vk\epsilon)}\right).$$

Discrete-time local BPS via Priority Queue

- Initialization: For $i \in [n]$, sample inter-event times τ_i with distribution

$$\max\left(0, 1 - \frac{\pi_i(x + v(\tau_i + 1)\epsilon)}{\pi_i(x + v\tau_i\epsilon)}\right) \prod_{k=0}^{\tau_i - 1} \min\left(1, \frac{\pi_i(x + v(k+1)\epsilon)}{\pi_i(x + vk\epsilon)}\right).$$

- Iterate.

Discrete-time local BPS via Priority Queue

- Initialization: For $i \in [n]$, sample inter-event times τ_i with distribution

$$\max\left(0, 1 - \frac{\pi_i(x + v(\tau_i + 1)\epsilon)}{\pi_i(x + v\tau_i\epsilon)}\right) \prod_{k=0}^{\tau_i - 1} \min\left(1, \frac{\pi_i(x + v(k+1)\epsilon)}{\pi_i(x + vk\epsilon)}\right).$$

- Iterate.
  - If $\min \tau_i \geq 1$, then output $z' = (x + \epsilon v, v)$. Update $\tau_i \leftarrow \tau_i - 1$.

Discrete-time local BPS via Priority Queue

- Initialization: For $i \in [n]$, sample inter-event times τ_i with distribution

$$\max\left(0, 1 - \frac{\pi_i(x + v(\tau_i + 1)\epsilon)}{\pi_i(x + v\tau_i\epsilon)}\right) \prod_{k=0}^{\tau_i - 1} \min\left(1, \frac{\pi_i(x + v(k+1)\epsilon)}{\pi_i(x + vk\epsilon)}\right).$$

- Iterate.
  - If $\min \tau_i \geq 1$, then output $z' = (x + \epsilon v, v)$. Update $\tau_i \leftarrow \tau_i - 1$.
  - Otherwise

Discrete-time local BPS via Priority Queue

- Initialization: For $i \in [n]$, sample inter-event times τ_i with distribution

$$\max\left(0, 1 - \frac{\pi_i(x + v(\tau_i + 1)\epsilon)}{\pi_i(x + v\tau_i\epsilon)}\right) \prod_{k=0}^{\tau_i - 1} \min\left(1, \frac{\pi_i(x + v(k+1)\epsilon)}{\pi_i(x + vk\epsilon)}\right).$$

- Iterate.
  - If $\min \tau_i \geq 1$, then output $z' = (x + \epsilon v, v)$. Update $\tau_i \leftarrow \tau_i - 1$.
  - Otherwise
    - Compute

$$\nabla \overline{U}(x) := \sum_{i:\tau_i = 0} \nabla U_i\left(x_{S_i}\right)$$

and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x) v$.

Discrete-time local BPS via Priority Queue

- Initialization: For $i \in [n]$, sample inter-event times τ_i with distribution

$$\max\left(0, 1 - \frac{\pi_i(x + v(\tau_i + 1)\epsilon)}{\pi_i(x + v\tau_i\epsilon)}\right) \prod_{k=0}^{\tau_i - 1} \min\left(1, \frac{\pi_i(x + v(k+1)\epsilon)}{\pi_i(x + vk\epsilon)}\right).$$

- Iterate.
  - If $\min \tau_i \geq 1$, then output $z' = (x + \epsilon v, v)$. Update $\tau_i \leftarrow \tau_i - 1$.
  - Otherwise
    - Compute
$$\nabla \overline{U}(x) := \sum_{i:\tau_i = 0} \nabla U_i\left(x_{S_i}\right)$$
and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x) v$.
    - With proba

$$\prod_{i:\tau_i > 0} \min\left\{1, \frac{\min\left(\pi_i(x), \pi_i(x - v^*\epsilon)\right)}{\min\left(\pi_i(x), \pi_i(x + v\epsilon)\right)}\right\} \prod_{i:\tau_i = 0} \min\left\{1, \frac{[\pi_i(x) - \pi_i(x - v^*\epsilon)]_+}{[\pi_i(x) - \pi_i(x + v\epsilon)]}\right\},$$

output $z' = (x, v^*)$. Sample again τ_i for all i where $v_j^* \neq v_j$ for some $j \in S_i$.

Discrete-time local BPS via Priority Queue

- Initialization: For $i \in [n]$, sample inter-event times τ_i with distribution

$$\max\left(0, 1 - \frac{\pi_i(x + v(\tau_i + 1)\epsilon)}{\pi_i(x + v\tau_i\epsilon)}\right) \prod_{k=0}^{\tau_i - 1} \min\left(1, \frac{\pi_i(x + v(k+1)\epsilon)}{\pi_i(x + vk\epsilon)}\right).$$

- Iterate.
  - If $\min \tau_i \geq 1$, then output $z' = (x + \epsilon v, v)$. Update $\tau_i \leftarrow \tau_i - 1$.
  - Otherwise
    - Compute
    $$\nabla \overline{U}(x) := \sum_{i:\tau_i = 0} \nabla U_i\left(x_{S_i}\right)$$
    and set $z^* = (x, v^*)$ where $v^* = R_{\nabla \overline{U}}(x)v$.
    - With proba

$$\prod_{i:\tau_i > 0} \min\left\{1, \frac{\min\left(\pi_i(x), \pi_i(x - v^*\epsilon)\right)}{\min\left(\pi_i(x), \pi_i(x + v\epsilon)\right)}\right\} \prod_{i:\tau_i = 0} \min\left\{1, \frac{[\pi_i(x) - \pi_i(x - v^*\epsilon)]_+}{[\pi_i(x) - \pi_i(x + v\epsilon)]}\right\},$$

    output $z' = (x, v^*)$. Sample again τ_i for all i where $v_j^* \neq v_j$ for some $j \in S_i$.
    - Otherwise output $z' = (x, -v)$. Sample τ_i for all i.

Discrete-time PDMP for intractable targets

- **Motivation**: intractable potential $H(z) = \int_\Omega H_\omega(z)\ \mu(d\omega)$.

Discrete-time PDMP for intractable targets

- **Motivation**: intractable potential $H(z) = \int_{\Omega} H_{\omega}(z)\, \mu(d\omega)$.
- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$.

Discrete-time PDMP for intractable targets

- **Motivation**: intractable potential $H(z) = \int_\Omega H_\omega(z)\ \mu(d\omega)$.
- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$.
- *Acceptance probability* $\alpha : \mathcal{Z} \to [0, 1]$ where

$$\alpha(z) = \exp\left\{ \int_\Omega \log \alpha_\omega(z)\, \mu(d\omega) \right\}$$

with $\alpha_\omega : \mathcal{Z} \to (0, 1]$. Sample a Poisson process P on Ω of rate $\Lambda(d\omega) = -\log \alpha_\omega(z)\ \mu(d\omega)$ of law denoted $\mathbb{P}(dP|z)$: $\alpha(z)$ is the void probability of P.

Discrete-time PDMP for intractable targets

- **Motivation**: intractable potential $H(z) = \int_\Omega H_\omega(z)\,\mu(d\omega)$.
- *Deterministic dynamics*: a diffeomorphism $\Phi : \mathcal{Z} \to \mathcal{Z}$.
- *Acceptance probability* $\alpha : \mathcal{Z} \to [0,1]$ where

$$\alpha(z) = \exp\left\{\int_\Omega \log \alpha_\omega(z)\,\mu(d\omega)\right\}$$

  with $\alpha_\omega : \mathcal{Z} \to (0,1]$. Sample a Poisson process P on Ω of rate $\Lambda(d\omega) = -\log \alpha_\omega(z)\,\mu(d\omega)$ of law denoted $\mathbb{P}(dP|z)$: $\alpha(z)$ is the void probability of P.
- *Markov kernel:* Conditional on P, with $|P| \geq 1$, sample $Z' \sim Q_P(Z,\cdot)$ so that

$$Q(z, dz') = \int_{\mathcal{P}} \mathbb{P}(dP|z, |P| \geq 1)\, Q_P(z, dz')$$

  where $\mathbb{P}(dP|z, |P| \geq 1)$ is the law of P conditioned upon $|P| \geq 1$

$$\mathbb{P}(dP|z, |P| \geq 1) = \frac{\mathbb{I}(|P| \geq 1)}{1-\alpha(z)}\mathbb{P}(dP|z).$$

.

Sufficient conditions for invariance

- Simple generalizations of the conditions **C1-C2** given before: allows to derive non-reversible algorithms for intractable targets.

Sufficient conditions for invariance

- Simple generalizations of the conditions **C1-C2** given before: allows to derive non-reversible algorithms for intractable targets.
- Main idea provides simple Metropolis-like algorithms in this scenario to sample from $\pi(x) \propto \exp(-\int_\Omega H_\omega(z)\ \mu(d\omega))$.

Sufficient conditions for invariance

- Simple generalizations of the conditions **C1-C2** given before: allows to derive non-reversible algorithms for intractable targets.
- Main idea provides simple Metropolis-like algorithms in this scenario to sample from $\pi(x) \propto \exp(-\int_{\Omega} H_{\omega}(z)\, \mu(d\omega))$.
- **Algorithm**

Sufficient conditions for invariance

- Simple generalizations of the conditions **C1-C2** given before: allows to derive non-reversible algorithms for intractable targets.

- Main idea provides simple Metropolis-like algorithms in this scenario to sample from $\pi(x) \propto \exp(-\int_\Omega H_\omega(z)\, \mu(d\omega))$.

- **Algorithm**
  - Sample $z^* \sim q(\cdot | z)$ from a symmetric proposal.

Sufficient conditions for invariance

- Simple generalizations of the conditions **C1-C2** given before: allows to derive non-reversible algorithms for intractable targets.
- Main idea provides simple Metropolis-like algorithms in this scenario to sample from $\pi(x) \propto \exp(-\int_{\Omega} H_{\omega}(z)\,\mu(d\omega))$.
- **Algorithm**
  - Sample $z^* \sim q(\cdot|\, z)$ from a symmetric proposal.
  - Sample a Poisson process P on Ω of rate $[H_{\omega}(z^*) - H_{\omega}(z)]_+\,\mu(d\omega)$.

Sufficient conditions for invariance

- Simple generalizations of the conditions **C1-C2** given before: allows to derive non-reversible algorithms for intractable targets.

- Main idea provides simple Metropolis-like algorithms in this scenario to sample from $\pi(x) \propto \exp(-\int_{\Omega} H_{\omega}(z)\ \mu(d\omega))$.

- **Algorithm**
  - Sample $z^* \sim q(\cdot|\,z)$ from a symmetric proposal.
  - Sample a Poisson process P on Ω of rate $[H_{\omega}(z^*) - H_{\omega}(z)]_+\ \mu(d\omega)$.
  - If $P = \emptyset$, then output $z' = z^*$.

Sufficient conditions for invariance

- Simple generalizations of the conditions **C1-C2** given before: allows to derive non-reversible algorithms for intractable targets.
- Main idea provides simple Metropolis-like algorithms in this scenario to sample from $\pi(x) \propto \exp(-\int_\Omega H_\omega(z)\ \mu(d\omega))$.
- **Algorithm**
  - Sample $z^* \sim q(\cdot|\, z)$ from a symmetric proposal.
  - Sample a Poisson process P on Ω of rate $[H_\omega(z^*) - H_\omega(z)]_+\ \mu(d\omega)$.
  - If $P = \emptyset$, then output $z' = z^*$.
  - Otherwise output $z' = z$.

Discussion

- Discrete-time non-reversible piecewise deterministic MCMC schemes can be derived easily

Discussion

- Discrete-time non-reversible piecewise deterministic MCMC schemes can be derived easily
  - Complex dynamics can be used,

Discussion

- Discrete-time non-reversible piecewise deterministic MCMC schemes can be derived easily
  - Complex dynamics can be used,
  - Can be used in gradient-free scenarios,

Discussion

- Discrete-time non-reversible piecewise deterministic MCMC schemes can be derived easily
  - Complex dynamics can be used,
  - Can be used in gradient-free scenarios,
  - Can be extended to discrete spaces.

Discussion

- Discrete-time non-reversible piecewise deterministic MCMC schemes can be derived easily
  - Complex dynamics can be used,
  - Can be used in gradient-free scenarios,
  - Can be extended to discrete spaces.
- All the methodology developed for continuous-time algorithms has a discrete time equivalent.

Discussion

- Discrete-time non-reversible piecewise deterministic MCMC schemes can be derived easily
  - Complex dynamics can be used,
  - Can be used in gradient-free scenarios,
  - Can be extended to discrete spaces.
- All the methodology developed for continuous-time algorithms has a discrete time equivalent.
- Subsampling ideas/intractable target ideas/Local updating can be used for discrete-time/reversible schemes.

Discussion

- Discrete-time non-reversible piecewise deterministic MCMC schemes can be derived easily
  - Complex dynamics can be used,
  - Can be used in gradient-free scenarios,
  - Can be extended to discrete spaces.
- All the methodology developed for continuous-time algorithms has a discrete time equivalent.
- Subsampling ideas/intractable target ideas/Local updating can be used for discrete-time/reversible schemes.
- Quantitative convergence results?

Discussion

- Discrete-time non-reversible piecewise deterministic MCMC schemes can be derived easily
  - Complex dynamics can be used,
  - Can be used in gradient-free scenarios,
  - Can be extended to discrete spaces.
- All the methodology developed for continuous-time algorithms has a discrete time equivalent.
- Subsampling ideas/intractable target ideas/Local updating can be used for discrete-time/reversible schemes.
- Quantitative convergence results?
- Complexity?