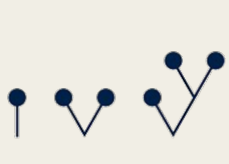
Markov Chain Cubature for Bayesian Inference

James Foster
University of Bath

joint with Thomas Coxon (Loughborough), Daniel Burrows (Bath), Terry Lyons (Oxford), Harald Oberhauser (Oxford) & Tom L (GCHQ)

UNIVERSITY OF BATH



DataSig

A rough path between mathematics and data science

The Alan Turing Institute

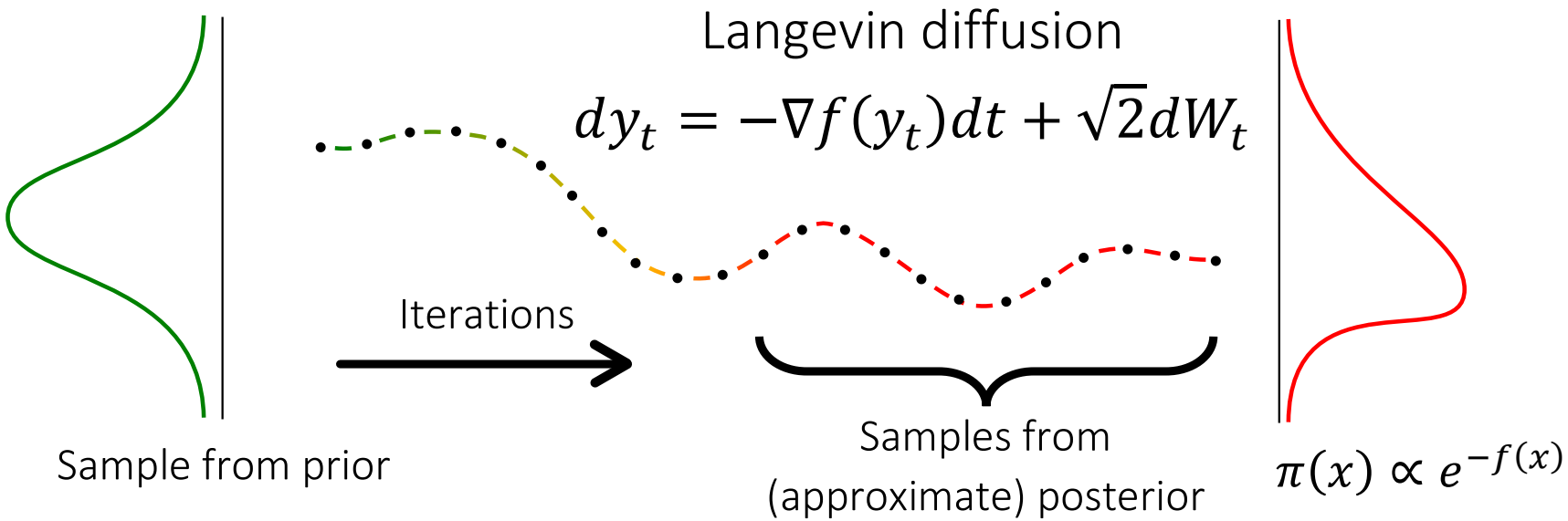
Engineering and Physical Sciences Research Council

Outline

1. Markov Chain Monte Carlo
2. One-step cubature formula
3. Distribution compression
4. Error analysis of cubature
5. Numerical experiments
6. Conclusion and ongoing work
7. References

Markov Chain Monte Carlo

A fundamental challenge in Bayesian inference is computing integrals with respect to posterior distributions and Markov Chain Monte Carlo (MCMC) is widely regarded as the "go-to" approach for these problems.



Under mild conditions on $f : \mathbb{R}^d \to \mathbb{R}$, the Langevin SDE admits a unique strong solution that is ergodic with stationary measure $\pi(x) \propto e^{-f(x)}$ [1].

Unadjusted Langevin Algorithm (ULA): $Y_{n+1} := Y_n - \nabla f(Y_n)h + \sqrt{2}W_n$

Convergence of Langevin Monte Carlo

For sufficiently small h, ULA "ends up close" to the target distribution [2]. However, ULA is a Markov chain and might not explore the space quickly.

Broadly speaking, this leads to two possible approaches

- Run N independent ULA chains and sample points at a fixed time T.
- Run a ULA chain over a long time horizon and sample at each step.

The latter strategy is much preferred by practitioners, but relies on the Markov chain to have a **fast mixing time**.

Convergence of Langevin Monte Carlo

For sufficiently small h, ULA "ends up close" to the target distribution [2]. However, ULA is a Markov chain and might not explore the space quickly.

Broadly speaking, this leads to two possible approaches

- Run N independent ULA chains and sample points at a fixed time T.
- Run a ULA chain over a long time horizon and sample at each step.

The latter strategy is much preferred by practitioners, but relies on the Markov chain to have a **fast mixing time**.

Motivating questions

Can Langevin dynamics be simulated as a cloud of (dependent) points?

Accuracy? Computational cost? Exploration of parameter space?

Outline

1. Markov Chain Monte Carlo
2. One-step cubature formula
3. Distribution compression
4. Error analysis of cubature
5. Numerical experiments
6. Conclusion and ongoing work
7. References

One-step cubature formula

Recall the Langevin diffusion with invariant measure $\pi \propto e^{-f}$ is given by

$$dy_t = -\nabla f(y_t)\,dt + \sqrt{2}\,dW_t, \tag{1}$$

By Taylor expanding (1), we can see that y has the following moments:

$$\mathbb{E}\big[(y_t - y_s)\,|\,y_s\big] = -\nabla f(y_s)h + O(h^2), \tag{2}$$

$$\mathbb{E}\big[(y_t - y_s)^{\otimes 2}\,|\,y_s\big] = 2hI_d + O(h^2), \tag{3}$$

for $s, t \geq 0$, where $h = t - s > 0$.

Construction of a first order one-step cubature rule started at y_s

We want to construct a cloud of points and weights $\{(x_i, w_i)\}_{1\leq i\leq N}$ with

$$\mu := \sum_{i=1}^{N} w_i x_i = -\nabla f(y_s)h, \qquad \Sigma := \sum_{i=1}^{N} w_i (x_i - \mu)(x_i - \mu)^\top = 2hI_d.$$

One-step cubature formula via Hadamard matrices

The Hadamard (or Walsh) matrices are defined inductively as

$$H_1 := (1)\,, \quad H_{2^{k+1}} := \begin{pmatrix} H_{2^k} & H_{2^k} \\ H_{2^k} & -H_{2^k} \end{pmatrix}.$$

Let $n := 2^{\lceil \log d \rceil}$ and define n vectors $\{e_i\}_{1 \le i \le n}$ in $\mathbb{R}^d$ as columns of H_n

$$H_n = \begin{pmatrix} e_1 & e_2 & \cdots & e_{n-1} & e_n \\ & & (n-d) \times n \text{ matrix} & & \end{pmatrix},$$

and another vectors $\{e_i\}_{n+1 \le i \le 2n}$ as $e_i := -e_{i-n}$ for $n+1 \le i \le 2n$.

Theorem (Victoir (2005))

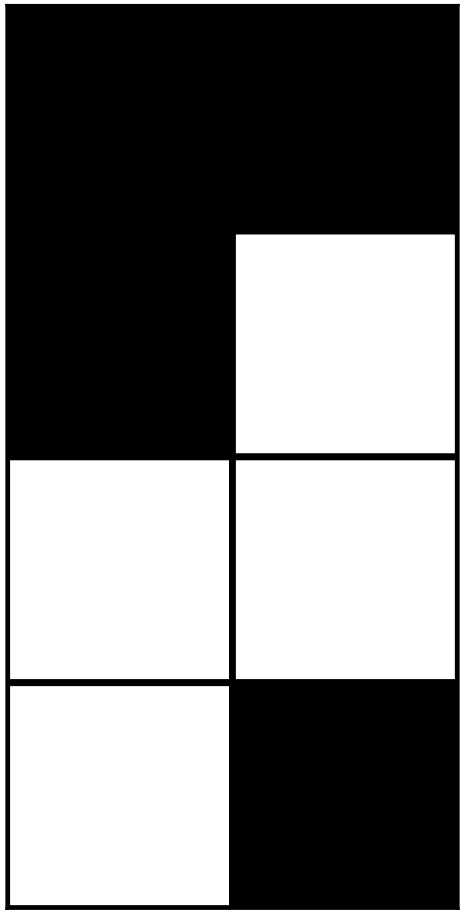
Let $X \sim \mathit{Uniform}\big(\{e_i\}_{1 \le i \le 2n}\big)$. Then X is a symmetric random vector with covariance matrix:

$$\mathbb{E}\big[X^{\otimes 2}\big] = I_d\,.$$

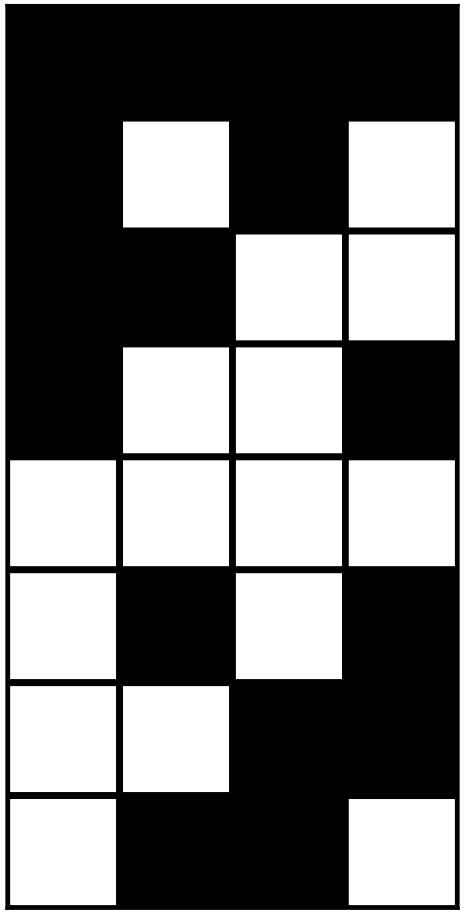
Thus, X matches the first 3 moments of the standard normal distribution.

One-step cubature formula via Hadamard matrices

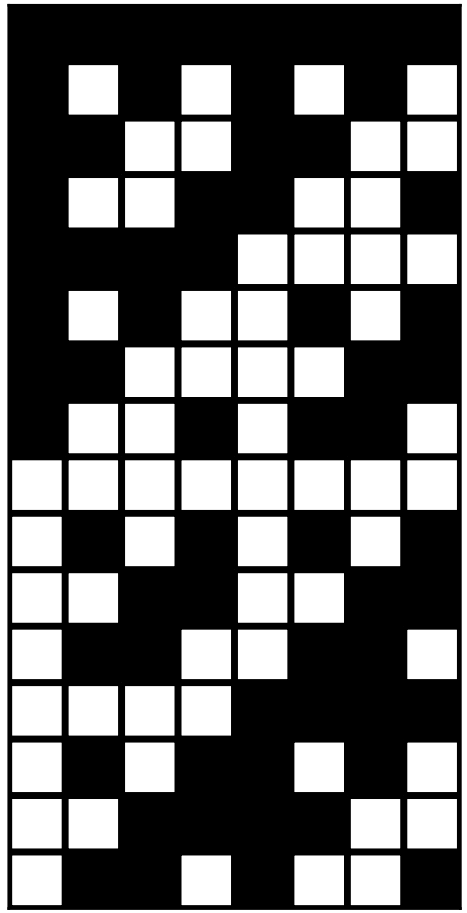
H_2



H_4



H_8



One-step cubature formula via Hadamard matrices

Therefore, in each step, we define $2^{\lceil \log d \rceil + 1}$ new cubature particles as

$$y_i^{\text{new}} := y_s - \nabla f(y_s)h + \sqrt{2h}e_i, \tag{4}$$

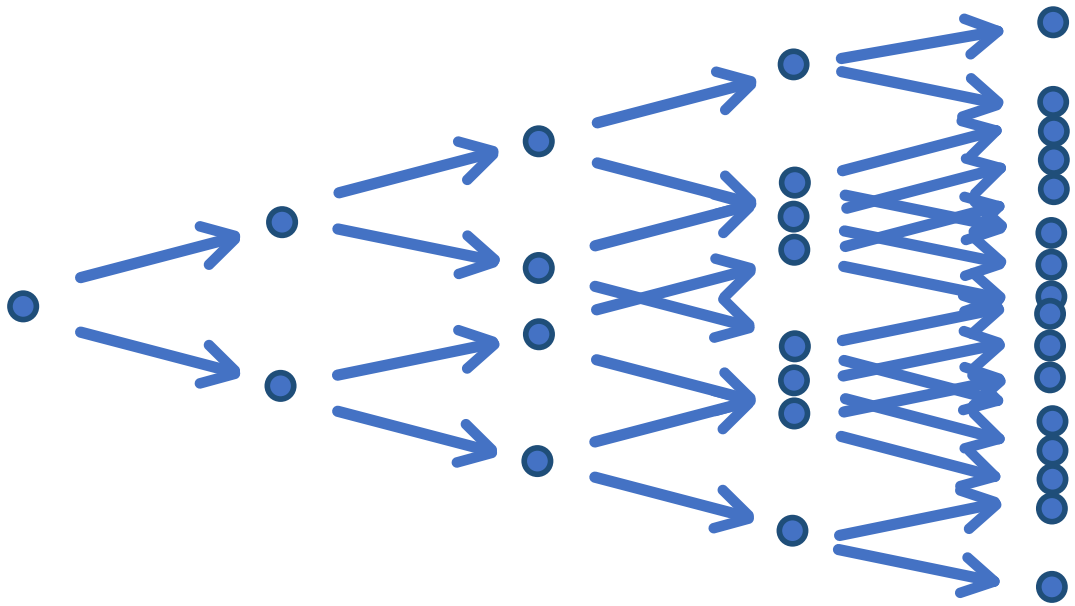
each with equal weight. The moments of (4) will then have $O(h^2)$ error. Equation (4) is a particular example of "Cubature on Weiner Space" [4]. However, even in the one-dimensional setting, there is a clear problem!

One-step cubature formula via Hadamard matrices

Therefore, in each step, we define $2^{\lceil \log d \rceil + 1}$ new cubature particles as

$$y_i^{\text{new}} := y_s - \nabla f(y_s)h + \sqrt{2h}e_i, \tag{4}$$

each with equal weight. The moments of (4) will then have $O(h^2)$ error. Equation (4) is a particular example of "Cubature on Weiner Space" [4]. However, even in the one-dimensional setting, there is a clear problem!



After each step, the number of particles increases by a factor of $\approx 2d$.

Outline

1. Markov Chain Monte Carlo
2. One-step cubature formula
3. Distribution compression
4. Error analysis of cubature
5. Numerical experiments
6. Conclusion and ongoing work
7. References

Distribution compression

Thus, for SDE cubature to be practical, we require an algorithm that can "compress" (or reduce the support of) discrete probability distributions. In addition to just resampling particles, there are a variety of algorithms where the compressed measure accurately integrates certain functions.

Distribution compression

Thus, for SDE cubature to be practical, we require an algorithm that can "compress" (or reduce the support of) discrete probability distributions. In addition to just resampling particles, there are a variety of algorithms where the compressed measure accurately integrates certain functions.

Test functions are explicitly specified by the user (e.g. polynomials)

- High order Recombination (Litterer and Lyons (2012), Tchernychova and Lyons (2016), Cosentino et al. (2020))

Distribution compression

Thus, for SDE cubature to be practical, we require an algorithm that can "compress" (or reduce the support of) discrete probability distributions. In addition to just resampling particles, there are a variety of algorithms where the compressed measure accurately integrates certain functions.

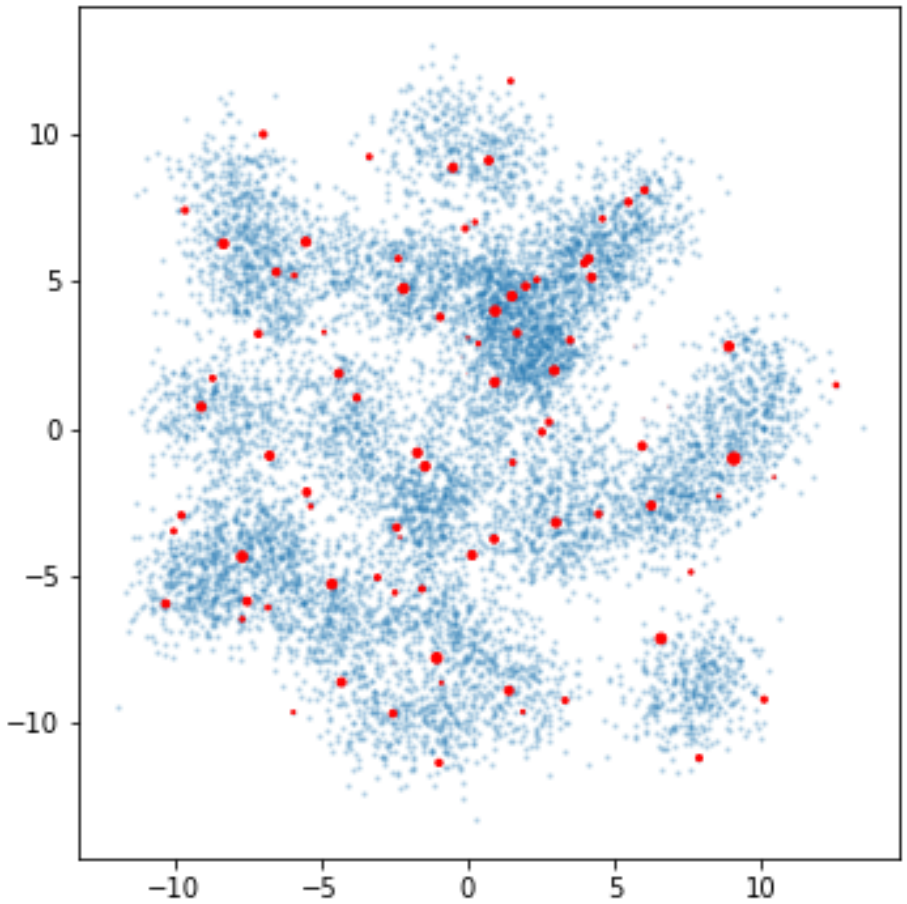
Test functions are explicitly specified by the user (e.g. polynomials)

- High order Recombination (Litterer and Lyons (2012), Tchernychova and Lyons (2016), Cosentino et al. (2020))

Test functions come from a reproducing kernel (i.e. MMD distance)

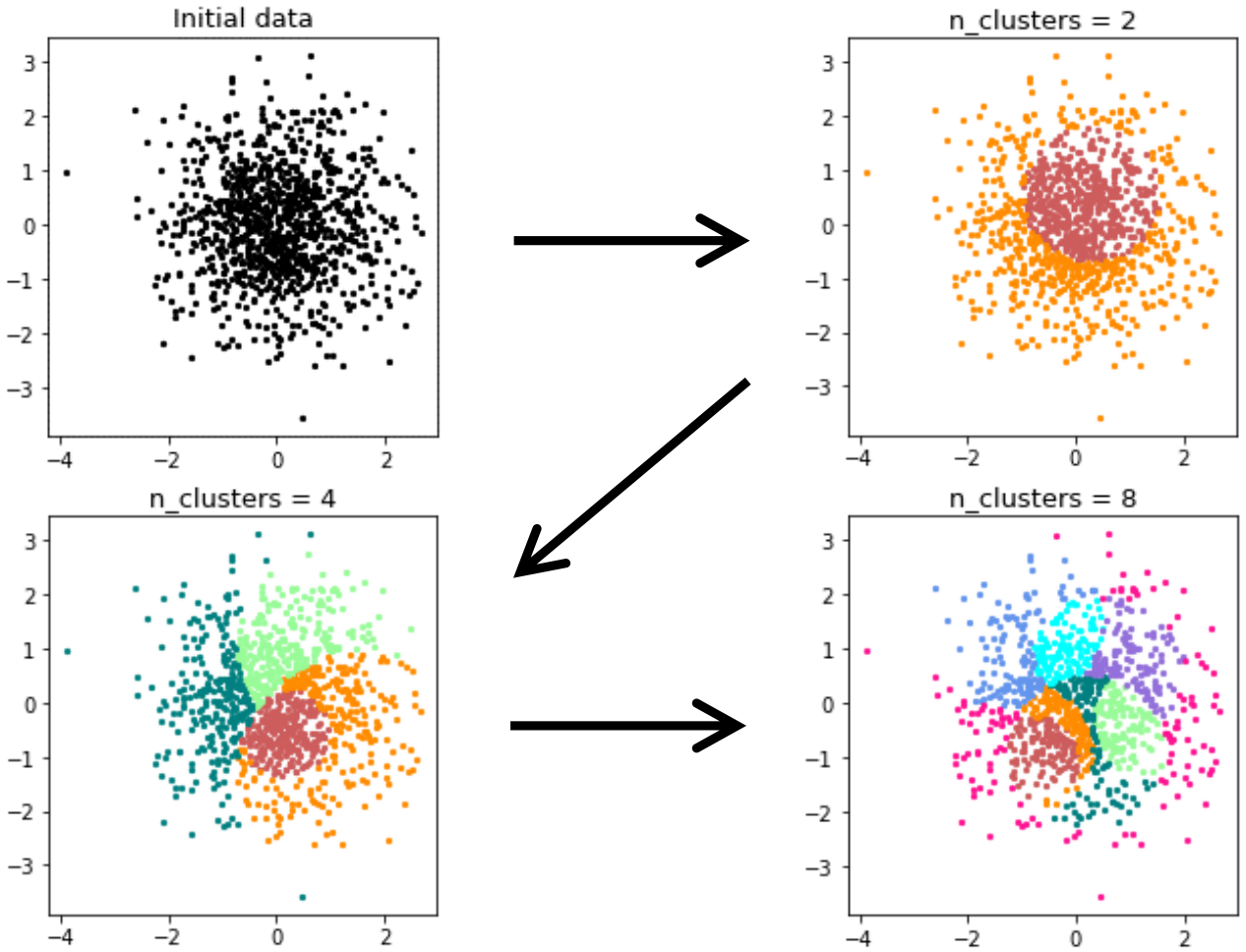
- Kernel Herding (Chen et al. (2010))
- Kernel Thinning (Dwivedi and Mackey, (2021))
- Kernel Recombination (Hayakawa et al. (2022, 2023))
- Stein Thinning (Riabiz et al. (2022))

Distribution compression



Example of Stein Thinning [14], which is designed to thin MCMC samples

Applying distribution compression locally



A standard ball tree algorithm for partitioning points is implemented in

```
from sklearn.neighbors import BallTree
```

A basic cubature algorithm

Step 0. Generate a cloud of N points Y_0 with uniform weights $w_0 = \frac{1}{N}$ from a prior distribution.

Step 1. In each step, generate a new cloud of $2^{1+\lceil \log d \rceil} N$ points by

$$Y_{n+1,i} := Y_n - \nabla f(Y_n) h + \sqrt{2h} e_i,$$

with weights $w_{n+1,i} := 2^{-(1+\lceil \log d \rceil)} w_n$.

The vectors $\{e_i\}$ come from the "Hadamard" cubature formula.

Step 2. Put the points into N smaller "patches" via a ball tree algorithm.

Step 3. On each patch, resample a point (or reduce small clouds of points/weights using a distribution compression algorithm).

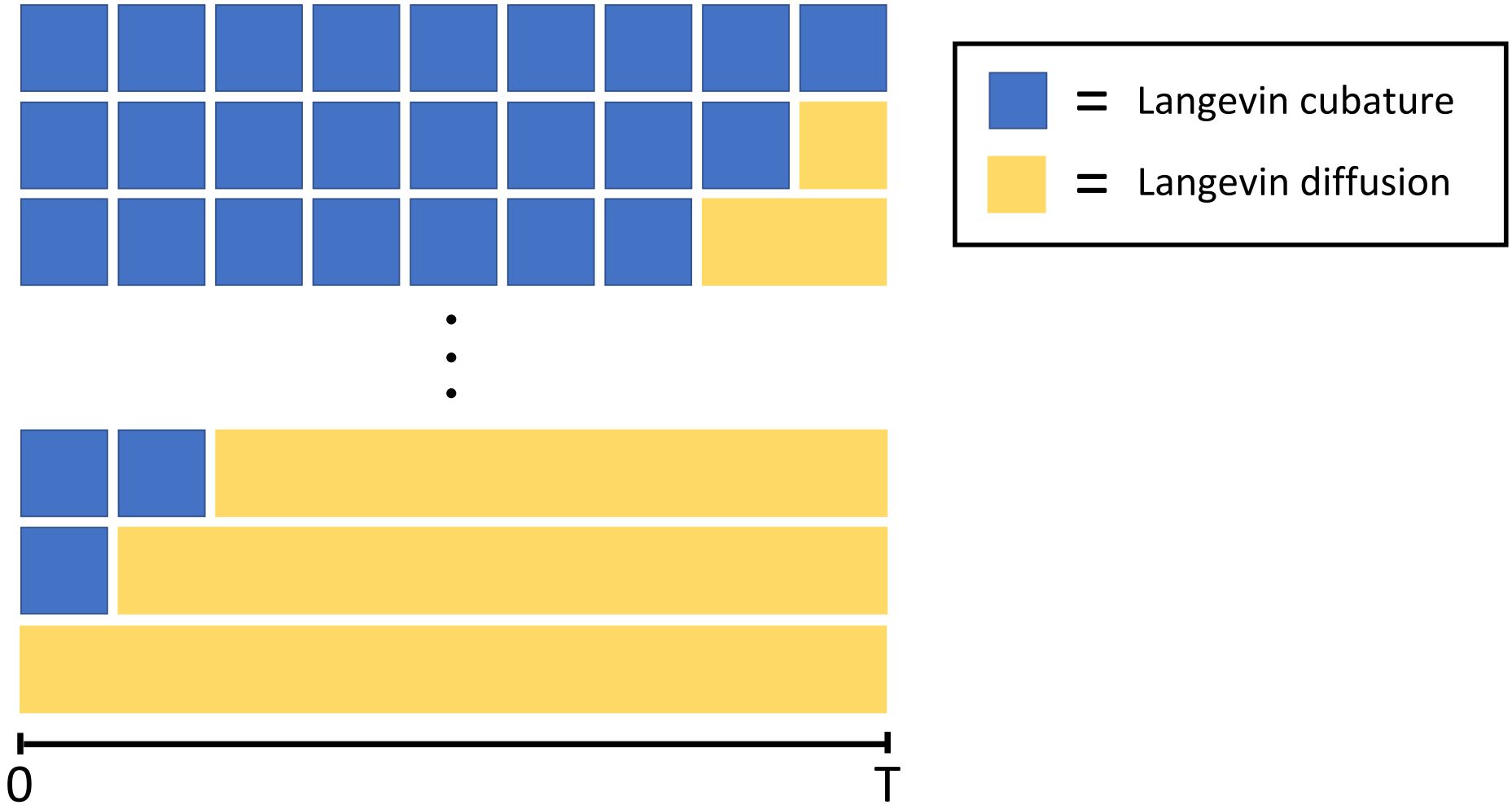
Step 4. Repeat steps 1 – 3.

Outline

1. Markov Chain Monte Carlo
2. One-step cubature formula
3. Distribution compression
4. Error analysis of cubature
5. Numerical experiments
6. Conclusion and ongoing work
7. References

Error analysis of cubature

We want to compare n steps of Langevin cubature to the SDE at $T = nh$.



Error analysis of cubature

For a smooth test function $F : \mathbb{R}^d \to \mathbb{R}$, let

$$P_t F : y \mapsto \mathbb{E}\big[F(y_t) \,|\, y_0 = y \big],$$
$$Q_k F : Y \mapsto \mathbb{E}\big[F(Y_k) \,|\, Y_0 = Y \big],$$

be the semigroups associated with the Langevin diffusion and cubature (Y_k is ULA with Hadamard random vectors). By a telescoping sum trick,

$$\begin{aligned} P_T F - Q_n F &= \sum_{k=0}^{n-1} Q_{n-(k+1)} \big(P_{t_{k+1}} F \big) - Q_{N-k} \big(P_{t_k} F \big) \\ &= \sum_{k=0}^{n-1} Q_{n-(k+1)} \big(P_h - Q_1 \big) \big(P_{t_k} F \big), \end{aligned}$$

where the cubature algorithm uses a step size of $h > 0$ and $t_k := kh$.

Error analysis of cubature

Since Q_1 is a Markov operator, it follows from [15, Theorem 13.2] that

$$\begin{aligned}
\left\|P_T F - Q_n F\right\|_{L^2_\mu} &\leq \sum_{k=0}^{n-1} \left\|Q_{n-(k+1)}\left(P_h - Q_1\right)\left(P_{t_k} F\right)\right\|_{L^2_\mu} \\
&= \sum_{k=0}^{n-1} \Big\| \underbrace{Q_1 \cdots Q_1}_{\substack{n-(k+1) \\ \text{times}}} \left(P_h - Q_1\right)\left(P_{t_k} F\right)\Big\|_{L^2_\mu} \\
&\leq \sum_{k=0}^{n-1} \left\|\left(P_h - Q_1\right)\left(P_{t_k} F\right)\right\|_{L^2_\mu},
\end{aligned}$$

where μ is the prior distribution (that is, $y_0 \sim \mu$) and $\|\cdot\|_{L^2_\mu}$ is the norm

$$\|G\|_{L^2_\mu} := \left(\int_{\mathbb{R}^d} \|G(x)\|^2 \, d\mu(x)\right)^{\frac{1}{2}}.$$

Error analysis of cubature

Since Q_1 is a Markov operator, it follows from [15, Theorem 13.2] that

$$\begin{aligned}
\left\|P_T F - Q_n F\right\|_{L^2_\mu} &\leq \sum_{k=0}^{n-1} \left\|Q_{n-(k+1)}\left(P_h - Q_1\right)\left(P_{t_k} F\right)\right\|_{L^2_\mu} \\
&= \sum_{k=0}^{n-1} \Big\| \underbrace{Q_1 \cdots Q_1}_{\substack{n-(k+1) \\ \text{times}}} \left(P_h - Q_1\right)\left(P_{t_k} F\right)\Big\|_{L^2_\mu} \\
&\leq \sum_{k=0}^{n-1} \left\|\left(P_h - Q_1\right)\left(P_{t_k} F\right)\right\|_{L^2_\mu},
\end{aligned}$$

where μ is the prior distribution (that is, $y_0 \sim \mu$) and $\|\cdot\|_{L^2_\mu}$ is the norm

$$\|G\|_{L^2_\mu} := \left(\int_{\mathbb{R}^d} \|G(x)\|^2 \, d\mu(x)\right)^{\frac{1}{2}}.$$

Hence, there are two different operators to estimate: $(P_h - Q_1)$ and P_{t_k}.

Error analysis of cubature (based on Talay-Tubaro [16])

Using the notation $F_t := P_t F$, we can Taylor expand $(P_h - Q_1)(P_t F)$ since

$$\mathbb{E}\big[F_t(y_h)\,|\,y_0 = y\big] = F_t(y) + \big(\Delta F_t(y) - \nabla F_t(y)\nabla f(y)\big)h + R_1(t,y)h^2,$$
$$\mathbb{E}\big[F_t(Y_1)\,|\,Y_0 = y\big] = F_t(y) + \big(\Delta F_t(y) - \nabla F_t(y)\nabla f(y)\big)h + R_2(t,y)h^2,$$

where each remainder $R_i(t,y)$ can bounded by a function of the form $C(1+\|y\|^m)e^{-\alpha t}$, due to some classical SDE theory [1, 17].

Error analysis of cubature (based on Talay-Tubaro [16])

Using the notation $F_t := P_t F$, we can Taylor expand $(P_h - Q_1)(P_t F)$ since

$$\mathbb{E}\big[F_t(y_h)\,|\,y_0 = y\big] = F_t(y) + \big(\Delta F_t(y) - \nabla F_t(y)\nabla f(y)\big)h + R_1(t,y)h^2,$$
$$\mathbb{E}\big[F_t(Y_1)\,|\,Y_0 = y\big] = F_t(y) + \big(\Delta F_t(y) - \nabla F_t(y)\nabla f(y)\big)h + R_2(t,y)h^2,$$

where each remainder $R_i(t,y)$ can bounded by a function of the form $C(1+\|y\|^m)e^{-\alpha t}$, due to some classical SDE theory [1, 17]. Therefore

$$\begin{aligned}\mathbb{E}_{y_0\sim\mu}\Big[\big\|\big(P_T F - Q_n F\big)(y_0)\big\|^2\Big]^{\frac{1}{2}} &\le \sum_{k=0}^{n-1} \mathbb{E}_{y_0\sim\mu}\Big[\big(C(1+\|y_0\|^m)e^{-\alpha t_k}\big)^2\Big]^{\frac{1}{2}} h^2 \\ &\le \sum_{k=0}^{n-1} \widetilde{C} h^2 e^{-\alpha t_k} \\ &\le \widetilde{C}\Big(\frac{1}{\alpha} + h\Big)h.\end{aligned}$$

Error analysis of cubature (based on Talay-Tubaro [16])

Using the notation $F_t := P_t F$, we can Taylor expand $(P_h - Q_1)(P_t F)$ since

$$\mathbb{E}\big[F_t(y_h)\,|\,y_0 = y\big] = F_t(y) + \big(\Delta F_t(y) - \nabla F_t(y)\nabla f(y)\big)h + R_1(t,y)h^2,$$
$$\mathbb{E}\big[F_t(Y_1)\,|\,Y_0 = y\big] = F_t(y) + \big(\Delta F_t(y) - \nabla F_t(y)\nabla f(y)\big)h + R_2(t,y)h^2,$$

where each remainder $R_i(t,y)$ can bounded by a function of the form $C(1+\|y\|^m)e^{-\alpha t}$, due to some classical SDE theory [1, 17]. Therefore

$$\begin{aligned}
\mathbb{E}_{y_0\sim\mu}\Big[\big\|\big(P_T F - Q_n F\big)(y_0)\big\|^2\Big]^{\frac{1}{2}} &\le \sum_{k=0}^{n-1} \mathbb{E}_{y_0\sim\mu}\Big[\big(C(1+\|y_0\|^m)e^{-\alpha t_k}\big)^2\Big]^{\frac{1}{2}} h^2 \\
&\le \sum_{k=0}^{n-1} \widetilde{C} h^2 e^{-\alpha t_k} \\
&\le \widetilde{C}\Big(\frac{1}{\alpha} + h\Big)h. \quad \boxed{O(h) \text{ error for all } n \ge 1}
\end{aligned}$$

Error analysis of cubature (based on Talay-Tubaro [16])

Classical SDE Theory ([1, Prop 4.2], [17, Prop 3.1 and Thm 3.4])

Suppose f is a *confining potential* ($\lim_{|x|\to\infty} f(x) = \infty$ and $\int_{\mathbb{R}^d} e^{-\beta f(x)} dx < \infty$ for $\beta > 0$), ∇f is *smooth with bounded derivatives* and $\exists c > 0$ such that $\langle x, \nabla f(x)\rangle \geq c\|x\|^2$. Then

- For $m \geq 1$, there exists constants $C_m, \alpha_m > 0$ such that for all $y_0 \in \mathbb{R}^d$ and $t \geq 0$,

$$\mathbb{E}\left[\|y_t\|^m\right] \leq C_m\left(1 + \|y_0\|^m e^{-\alpha_m t}\right). \tag{5}$$

- Suppose $F \in C^\infty(\mathbb{R}^d)$ and its derivatives have at most polynomial growth at ∞. Define $u(t,y) := \mathbb{E}\left[F(y_t)|y_0 = y\right]$. Then for any given multi-index $I = (i_1, \cdots, i_k)$, there exists $C_I, \alpha_I > 0$ and a positive integer m_I such that for $y \in \mathbb{R}^d$ and $t \geq 0$,

$$\left|\frac{\partial^{|i|}}{\partial^{i_1} y^1 \cdots \partial^{i_n} y^{i_k}}\left(u(t,y)\right)\right| \leq C_I\left(1 + \|y\|^{m_I}\right)e^{-\alpha_I t}. \tag{6}$$

  In addition, there exists $C, \alpha > 0$ and $m \geq 1$ such that for all $y \in \mathbb{R}^d$ and $t \geq 0$,

$$\left|u(t,y) - \mathbb{E}_{x\sim\pi}\left[F(x)\right]\right| \leq C\left(1 + \|y\|^m\right)e^{-\alpha t}, \tag{7}$$

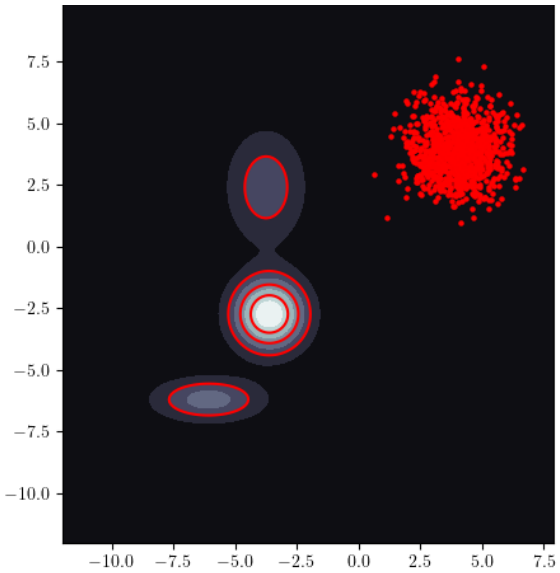
  where $\pi(x) \propto e^{-f(x)}$ is the *stationary distribution* of the Langevin diffusion $\{y_t\}$.

Outline

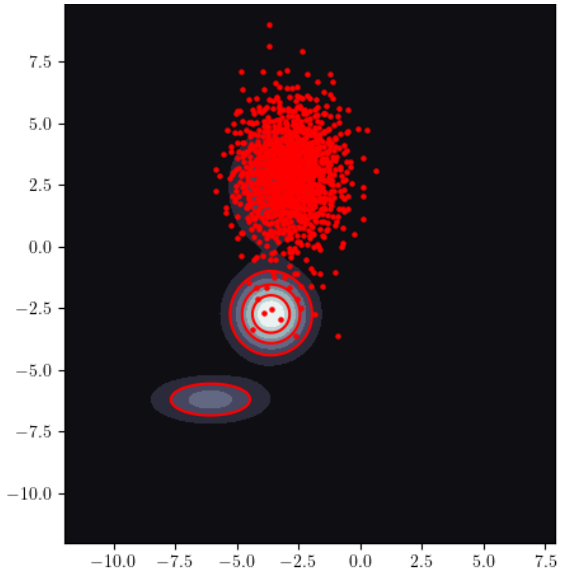
1. Markov Chain Monte Carlo
2. One-step cubature formula
3. Distribution compression
4. Error analysis of cubature
5. Numerical experiments
6. Conclusion and ongoing work
7. References

Langevin cubature on a Gaussian mixture model

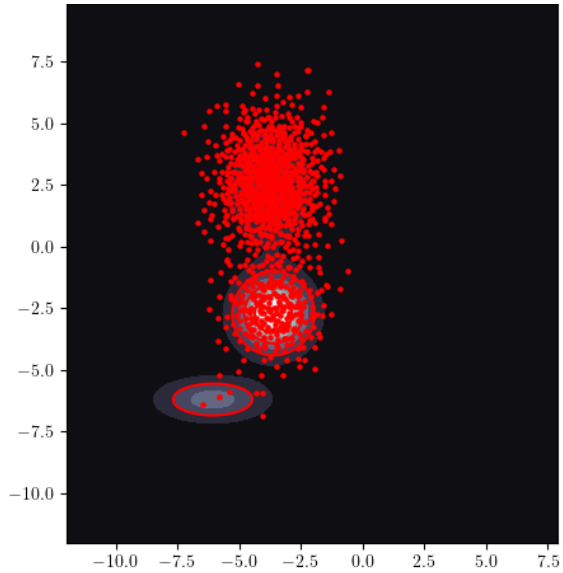
Iter 0; $\|\mu - \hat{\mu}\| = 10.036$; $\|\Sigma - \hat{\Sigma}\| = 10.019$



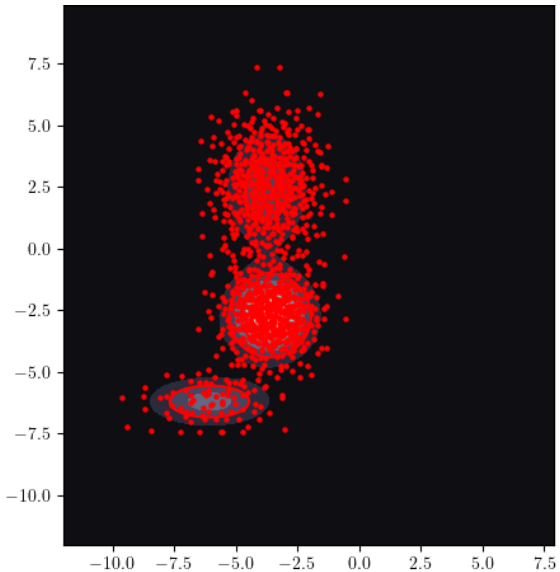
Iter 25; $\|\mu - \hat{\mu}\| = 4.971$; $\|\Sigma - \hat{\Sigma}\| = 8.144$



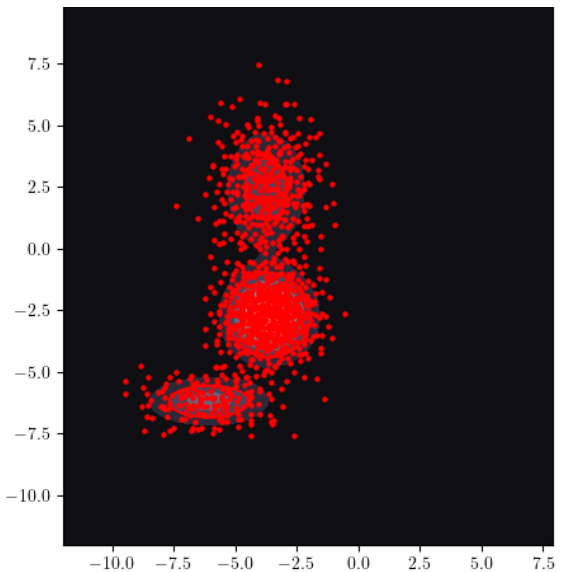
Iter 100; $\|\mu - \hat{\mu}\| = 3.288$; $\|\Sigma - \hat{\Sigma}\| = 3.767$



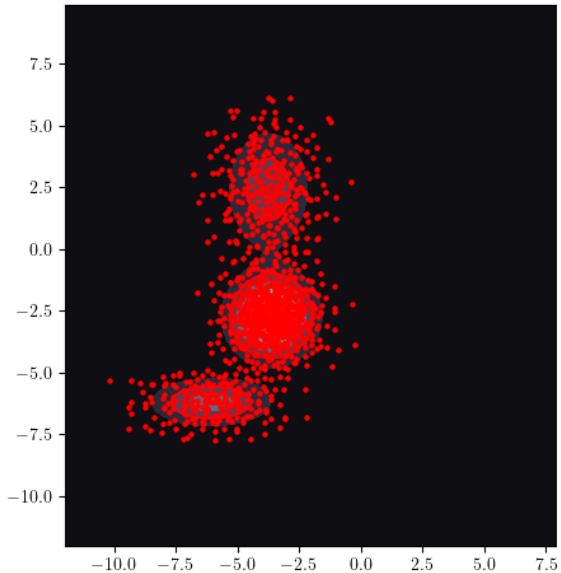
Iter 250; $\|\mu - \hat{\mu}\| = 1.590$; $\|\Sigma - \hat{\Sigma}\| = 0.640$



Iter 500; $\|\mu - \hat{\mu}\| = 0.488$; $\|\Sigma - \hat{\Sigma}\| = 0.311$



Iter 1000; $\|\mu - \hat{\mu}\| = 0.016$; $\|\Sigma - \hat{\Sigma}\| = 0.227$



Cubature vs MCMC on a Gaussian mixture model

All algorithms were implemented in Python and ran on a laptop (not in parallel)

Algorithm	Time (s)	Iterations	$\|\mu - \hat{\mu}\|$ (3.d.p)
Langevin cubature	32.9	1000	0.016
Single MCMC chain (ULA)	362.5	1000000 (+ 1000 burn-in)	0.047

Cubature parameters:

- Step size, $h = 0.1$
- Number of particles, $N = 1024$

MCMC parameters:

- Step size, $h = 0.1$

Cubature vs Stein Variational Gradient Descent

Stein Variational Gradient Descent (SVGD), due to Liu & Wang (2016), is a very popular interacting particle algorithm for Bayesian inference.

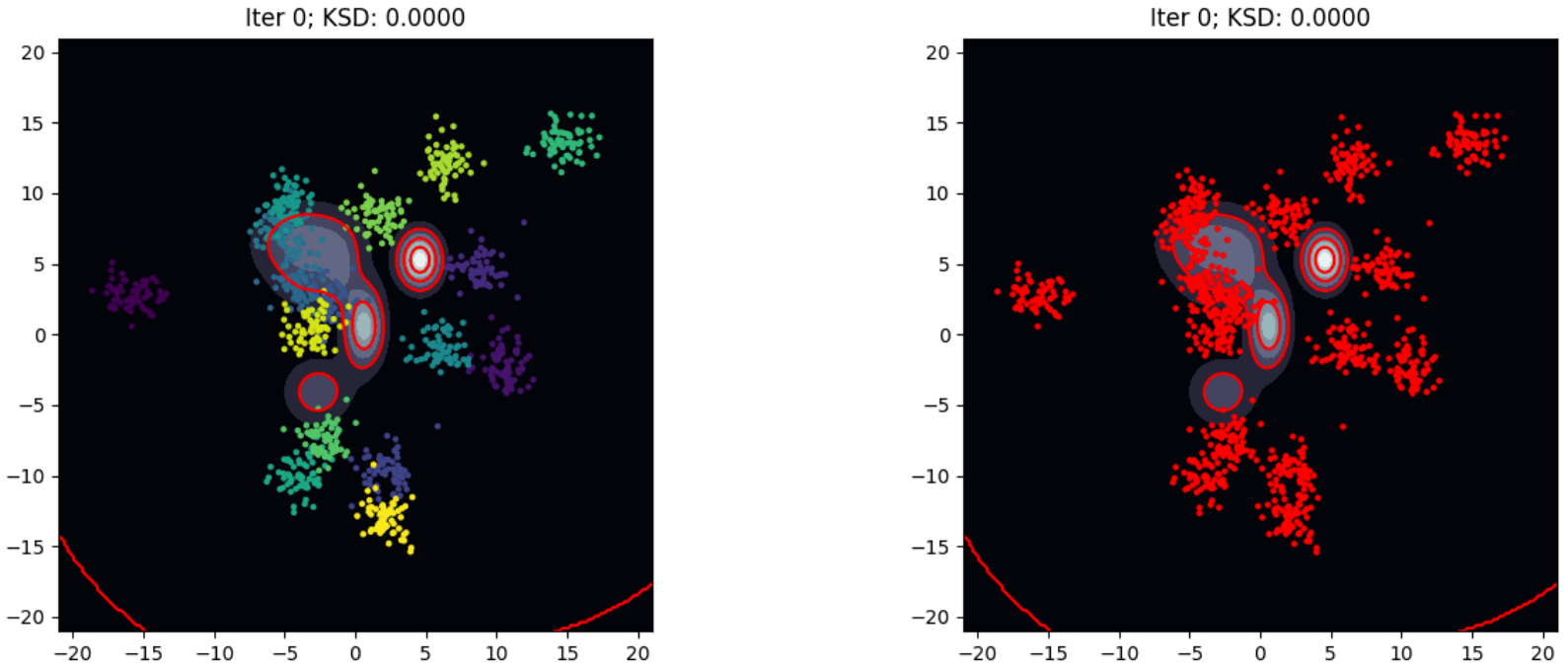
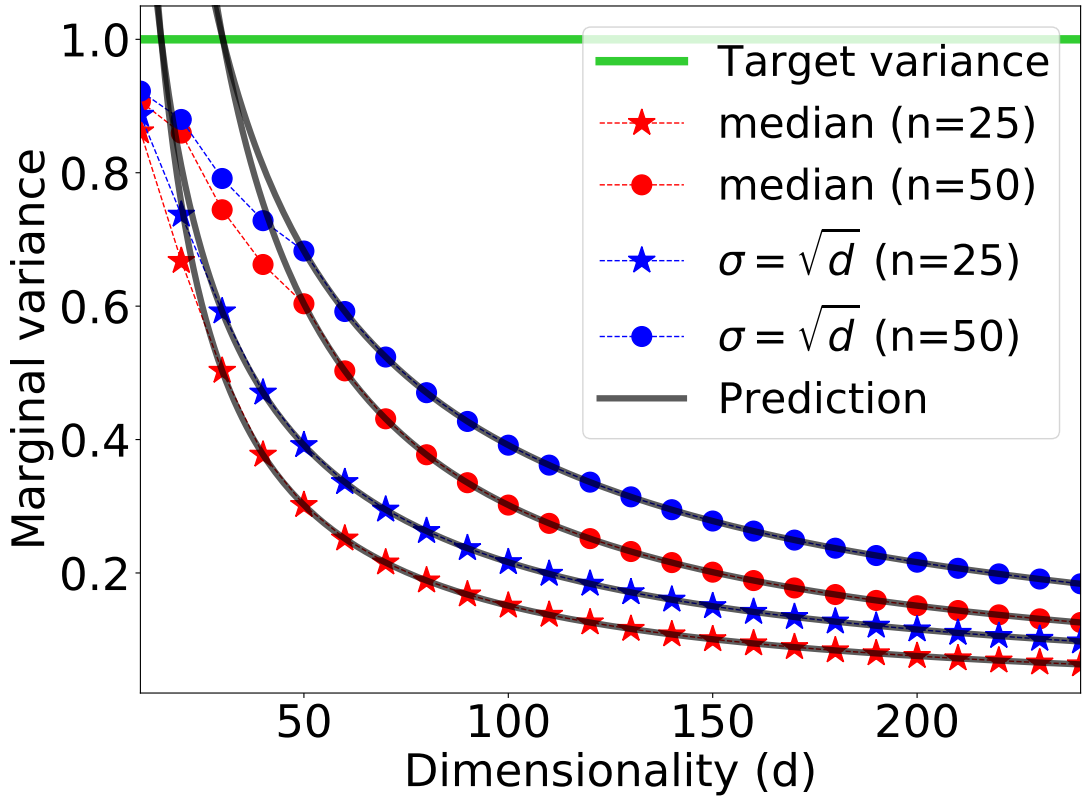


Figure: (left) Langevin Cubature. (right) Stein Variational Gradient Descent.

The variance collapse of SVGD

However, without enough particles, SVGD exhibits “variance collapse” and behaves similar to gradient-based single particle optimisation [19].



Furthermore, SVGD scales quadratically with the number of particles.

Cubature vs SVGD on Bayesian logistic regression

Forest Covertype dataset [20]: 581,012 data points and 54 features.

Algorithm	Iterations	Test accuracy	Log-likelihood
Langevin Cubature (with Tamed Euler [21])	1500	75.9%	-0.564
Stein Variational Gradient Descent	6000	75.7%	-0.521

Langevin Cubature ran for 10.7 hours whereas SVGD ran for 7.1 hours.
SVGD has similar test accuracy, but is much faster with fewer particles!

Cubature parameters:

- Step size, $h = 0.01$
- Particles, $N = 8192$

SVGD parameters:

- Step size, $h = 0.05$
- Particles, $N = 8192$
- RBF kernel, $\sigma^2 = \mathrm{med}^2 / \log N$

Both algorithms use the same prior distribution and a batch size of 100.

Outline

1. Markov Chain Monte Carlo
2. One-step cubature formula
3. Distribution compression
4. Error analysis of cubature
5. Numerical experiments
6. Conclusion and ongoing work
7. References

Conclusion and ongoing work

Conclusion

- Cubature is a way of solving SDEs as a cloud of (dependent) particles.
- Langevin cubature can outperform MCMC on a 2D Gaussian mixture and also perform well on a non-toy example, but is currently too slow.
- Unlike SVGD, it can avoid $O(N^2)$ complexity in the number of particles.

Conclusion and ongoing work

Conclusion

- Cubature is a way of solving SDEs as a cloud of (dependent) particles.
- Langevin cubature can outperform MCMC on a 2D Gaussian mixture and also perform well on a non-toy example, but is currently too slow.
- Unlike SVGD, it can avoid $O(N^2)$ complexity in the number of particles.

Ongoing work

(currently being developed by Thomas Coxon)

- Error analysis for Langevin cubature with compression (my work)

- Easy-to-use GPU-capable JAX implementation

```
# 1. Construct the cubature components
measure_propagator = LangevinDiffusionPropagator()
measure_partitioner = BinaryTreePartitioner(number_of_partitions=16, tree="BallTree")
measure_compressor = MonteCarloCompressor(
    key=compressor_key,
    with_replacement=True,
)

# 2. Construct the cubature
cubature_visualizer = TensorboardVisualizer(logdir="examples")
cubature = CubatureRule(
    cubature_step=StandardCubatureStep(
        measure_propagator, measure_partitioner, measure_compressor
    ),
    prior_measure=lambda particles: prior_measure.sample(particles, seed=prior_key),
)
cubature_options = CubatureOptions(
    number_of_seed_points=1024,
    step_size=0.1,
    number_of_cubature_epochs=250,
)
```

Thank you
for your attention!

Outline

1. Markov Chain Monte Carlo
2. One-step cubature formula
3. Distribution compression
4. Error analysis of cubature
5. Numerical experiments
6. Conclusion and ongoing work
7. References

References I

- G. A. Pavliotis. *Stochastic Processes and Applications*, Springer, New York, 2014.
- S. Vempala and A. Wibisono. *Rapid Convergence of the Unadjusted Langevin Algorithm: Isoperimetry Suffices*, NeurIPS 2019.
- N. Victoir. *Asymmetric Cubature Formulae with Few Points in High Dimension for Symmetric Measures*, SIAM Journal on Numerical Analysis, vol. 42, no. 1, pp. 209–227, 2005.
- T. Lyons and N. Victoir. *Cubature on Wiener space*, Proceedings of the Royal Society A, vol. 460, no. 2041, pp. 169–198, 2004.
- C. Litterer and T. Lyons. *High order recombination and an application to cubature on Wiener space*, Annals of Applied Probability, vol. 22, no. 4, pp. 1301–1327, 2012.
- M. Tchernychova. *Caratheodory cubature measures*, DPhil thesis, University of Oxford, 2016.

References II

- F. Cosentino, H. Oberhauser and A. Abate. *A randomized algorithm to reduce the support of discrete measures*, NeurIPS 2020.
- Y. Chen, M. Welling and A. Smola. *Super-Samples from Kernel Herding*, Conference on Uncertainty in Artificial Intelligence, 2010.
- R. Dwivedi and L. Mackey. *Kernel Thinning*, Proceedings of Thirty Fourth Conference on Learning Theory, 2021.
- S. Hayakawa, H. Oberhauser and T. Lyons. *Positively Weighted Kernel Quadrature via Subsampling*, NeurIPS 2022.
- S. Hayakawa, H. Oberhauser and T. Lyons. *Sampling-based Nyström Approximation and Kernel Quadrature*, arxiv:2301.09517, 2023.
- M. Adachi, S. Hayakawa, M. Jørgensen, H. Oberhauser and M. Osborne. *Fast Bayesian Inference with Batch Bayesian Quadrature via Kernel Recombination*, NeurIPS 2022.

References III

- M. Adachi, S. Hayakawa, S. Hamid, M. Jørgensen, H. Oberhauser and M. Osborne. *SOBER: Scalable Batch Bayesian Optimization and Quadrature using Recombination Constraints*, arxiv:2301.11832.
- M. Riabiz, W. Ye Chen, J. Cockayne, P. Swietach, S. Niederer, L. Mackey and C. Oates. *Optimal thinning of MCMC output*, Journal of Royal Statistical Society B (Statistical Methodology), 2022.
- T. Eisner, B. Farkas, M. Haase and R. Nagel, *Operator Theoretic Aspects of Ergodic Theory*, Graduate Texts in Maths, Springer, 2015.
- D. Talay and L. Tubaro. *Expansion of the global error for numerical schemes solving stochastic differential equations*, Stochastic Analysis and Applications, vol. 8, no. 4, pp. 483–509, 1990.
- D. Talay. *Second-order discretization schemes of stochastic differential systems for the computation of the invariant law*, Stochastics and Stochastic Reports, vol. 29, no. 1, pp. 13–36, 1990.

References IV

- B. Leimkuhler, C. Matthews and M. Tretyakov. *On the long-time integration of stochastic gradient systems*, Proceedings of the Royal Society A, vol. 470, no. 2170, 2014.
- J. Ba, M.Erdogdu, M. Ghassemi, S. Sun, T. Suzuki, D. Wu, T. Zhang. *Understanding the Variance Collapse of SVGD in High Dimensions*, International Conference on Learning Representations (ICLR) 2022.
- M. Lichman. *UCI machine learning repository*, https://archive.ics.uci.edu/ml, 2013. Binary covertype dataset: www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html
- D.-Y. Lim and S. Sabanis. *Polygonal Unadjusted Langevin Algorithms: Creating stable and efficient adaptive algorithms for neural networks*, arxiv:2105.13937, 2021.