Diffusion Map Particle Systems for Generative Modeling

Fengyi Li [1] **Youssef Marzouk** [1]

Abstract

We propose a novel diffusion map particle system (DMPS) for generative modeling, based on diffusion maps and Laplacian-adjusted Wasserstein gradient descent (LAWGD). Diffusion maps are used to approximate the generator of the Langevin diffusion process from samples, and hence to learn the underlying data-generating manifold. On the other hand, LAWGD enables efficient sampling from the target distribution given a suitable choice of kernel, which we construct here via a spectral approximation of the generator, computed with diffusion maps. Our method requires no offline training and minimal tuning, and can outperform other approaches on data sets of moderate dimension.

1. Introduction

Generative modeling is a central task in fields such as computer vision (Cai et al., 2020; Ho et al., 2021) and natural language processing (Yogatama et al., 2017; Miao & Blunsom, 2016), and applications ranging from medical image analysis (Yi et al., 2019) to protein design (Wu et al., 2021). Despite their successes, popular generative models such as variational auto-encoders (VAE) (Kingma & Welling, 2013; Rezende et al., 2014), generative adversarial networks (GANs) (Goodfellow et al., 2014), and score-based generative models (SGM) (Song & Ermon, 2019; Song et al., 2020), typically need careful hyperparameter tuning (Ruthotto & Haber, 2021; Song & Ermon, 2020) and may involve long convergence times, e.g., for Langevin-type sampling (Franzese et al., 2022). The performance of such methods also depends on the choice of architecture and parameters of deep neural networks (Salimans et al., 2016; Khandelwal & Krishnan, 2019), which, all too often, require expert knowledge and tuning.

In this paper, we propose a new non-parametric kernel-based approach to generative modeling, based on diffusion maps and interacting particle systems.

Diffusion maps (Coifman et al., 2005; Coifman & Lafon, 2006; Nadler et al., 2005; 2006), along with many other graph-based methods (Belkin & Niyogi, 2003; Tenenbaum et al., 2011; Roweis & Saul, 2000), have mainly been used as a tool for nonlinear dimension reduction. The kernel matrix, constructed using pairwise distances between samples with proper normalization, approximates the generator of a Langevin diffusion process. This approximation becomes exact as the number of samples goes to infinity and the kernel bandwidth goes to zero. Construction of the kernel matrix using smooth kernels (e.g., Gaussians) enables one to compute the inverse of the eigenvalues and the gradients of the eigenfunctions analytically. Separately, the notion of gradient flow (Santambrogio, 2016; Ambrosio et al., 2005; Daneri & Savaré, 2010) underlies a very active field of research and offers a unifying perspective on sampling and optimization (Jordan et al., 1998; Wibisono, 2018). The unadjusted Langevin algorithm (ULA) is a canonical example, and follows from the time discretization of a Langevin SDE. But many other particle systems, particularly interacting particle systems, approximate gradient flows: examples include Stein variational gradient descent (SVGD) (Liu, 2017; Liu & Wang, 2016), affine-invariant interacting Langevin dynamics (Garbuno-Iñigo et al., 2019), and Laplacian-adjusted Wasserstein gradient descent (LAWGD) (Chewi et al., 2020).

Our approach combines these two ideas by using diffusion maps to directly approximate (the gradient of the inverse of) the generator of the Langevin diffusion process from samples, and in turn using this approximation within LAWGD to produce more samples efficiently. Compared to other generative modeling methods, our approach has several advantages. First, the use of diffusion maps facilitates accurate sampling from distributions supported on manifolds, particularly when the dimension of the manifold is lower than that of the ambient space. Second, we demonstrate accurate sampling from distributions with (a priori unknown) bounded support. Both of these features are in contrast with methods driven only by approximations of the local score: we conjecture that such methods are less able to detect the overall geometry of the target distribution, whereas our approach harnesses graph-based methods that are widely used

[1]Massachusetts Institute of Technology, Cambridge, MA 02139 USA. Correspondence to: Fengyi Li <fengyil@mit.edu>.

Accepted to ICML workshop on *Structured Probabilistic Inference & Generative Modeling*, Honolulu, Hawaii, USA.

for nonlinear dimension reduction to approximate the generator as a whole. Finally, our method is quite simple and computationally efficient comparing to training a neural network (e.g., as in score-based generative modeling (Song & Ermon, 2019) or normalizing flows (Caterini et al., 2021; Ho et al., 2019)): the only parameter that needs to be tuned is the kernel bandwidth, and no offline training is required.

2. Diffusion map and kernel construction

We use π to denote the underlying target distribution, i.e., the distribution we would like to sample from, and let V denote the potential, where $\pi \propto \exp(-V)$. Let $\mathcal{P}_2(\mathcal{D})$ be the collection of probability measures on a compact manifold $\mathcal{D}$ that have finite second moments. All distributions are assumed to have densities with respect to Lebesgue measure, and we will abuse notation by using the same symbol to denote a measure and its density. The kernel $K(\cdot,\cdot) : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ is assumed to be differentiable with respect to both arguments, and we use $\nabla_1 K(\cdot,\cdot)$, $\nabla_2 K(\cdot,\cdot)$ to denote the (Euclidean) gradient of the kernel with respect to its first and second arguments, respectively. We assume sufficient regularity to exchange the order of integration and differentiation (Leibniz integral rule) throughout.

Appendix A provides a brief review of Wasserstein gradient flow and recalls the LAWGD algorithm of Chewi et al. (2020). To adapt LAWGD to the generative modeling setting, we must approximate the inverse of the generator of the Langevin diffusion process, $\mathscr{L}^{-1}$, for π; doing so will let us implement the update step (11) of LAWGD.

2.1. Diffusion map approximation of the generator

Diffusion maps (Coifman et al., 2005; Coifman & Lafon, 2006; Nadler et al., 2005; 2006) provide a natural framework for approximating $\mathscr{L}$ using kernels. Consider the Gaussian kernel $K_\epsilon(x,y) = e^{-\frac{\|x-y\|^2}{2\epsilon}}$ under some normalization

$$M_\epsilon(x,y) := \frac{K_\epsilon(x,y)}{\sqrt{\int K_\epsilon(x,y)d\pi(x)}\sqrt{\int K_\epsilon(x,y)d\pi(y)}}.$$

We construct the following two quantities,

$$P^f_\epsilon(x,y) := \frac{M_\epsilon(x,y)}{\int M_\epsilon(x,y)d\pi(x)},$$
$$P^b_\epsilon(x,y) := \frac{M_\epsilon(x,y)}{\int M_\epsilon(x,y)d\pi(y)},$$

by normalizing with respect to the first or second argument. Here f and b stand for 'forward' and 'backward,' respectively. Their actions on a function g are defined as

$$T^{f,b}_\epsilon g(\cdot) = \int P^{f,b}_\epsilon(\cdot,y)g(y)d\pi(y).$$

We also define the associated operators

$$L^{f,b}_\epsilon := \frac{T^{f,b}_\epsilon - \mathrm{Id}}{\epsilon}.$$

As studied in Nadler et al. (2006), both the forward and backward operators converge to the generator,

$$\lim_{\epsilon\to 0} L^f_\epsilon = \lim_{\epsilon\to 0} L^b_\epsilon = \mathscr{L}. \tag{1}$$

Combining the previous results, we have

$$\lim_{\epsilon\to 0} \frac{\int P^{f,b}_\epsilon(\cdot,y)g(y)d\pi(y) - g(\cdot)}{\epsilon} = \mathscr{L}g(\cdot). \tag{2}$$

Note that this approximation holds only when data lie on a compact manifold (Nadler et al., 2006; Hein et al., 2005; Singer, 2006). In practice, however, this assumption can be relaxed. In addition, although $L^{f,b}_\epsilon$ converges to a symmetric operator in the continuum limit, neither L^f_ϵ nor L^b_ϵ is symmetric. However, they satisfy $P^f_\epsilon(x,y) = P^b_\epsilon(y,x)$ and $L^f_\epsilon = (L^b_\epsilon)^*$. Therefore, one way to get a symmetric operator is to take the average of the two

$$L_\epsilon = \frac{1}{2}(L^f_\epsilon + L^b_\epsilon).$$

L_ϵ inherits all the properties of the forward and the backward kernel, hence converging to $\mathscr{L}$ in the limit. We now consider a finite sample approximation of the operator $\mathscr{L}$. Given samples $\{z^i\}_{i=1}^N \sim \pi$, the above construction can be approximated by samples, namely by replacing the integral by its empirical average. Details of the finite-sample construction are in Appendix C. We add another subscript N, i.e., $M_{\epsilon,N}, P^{f,b}_{\epsilon,N}, T^{f,b}_{\epsilon,N}$ and $L^{f,b}_{\epsilon,N}$, to signify N-sample approximations of the objects above.

2.2. Spectral approximation of $\nabla\mathscr{L}^{-1}f(x)$

We now exploit the spectral properties of the kernel. Recall from above that $\mathscr{L} = \lim_{\epsilon\to 0}\frac{T_\epsilon - \mathrm{Id}}{\epsilon}$, where $T_\epsilon g(\cdot) = \int P_\epsilon(\cdot,y)g(y)\pi(y)$. Also, $P_\epsilon = \frac{1}{2}(P^f_\epsilon + P^b_\epsilon)$ is a symmetric positive definite kernel. Then Mercer's theorem states that

$$P_\epsilon(x,y) = \sum_{i=1}^\infty \lambda_i\phi_i(x)\phi_i(y). \tag{3}$$

Therefore, we can write

$$\mathscr{L}f(x) = \lim_{\epsilon\to 0}\int K_{\mathscr{L},\epsilon}(x,y)f(y)d\pi(y),$$

where $K_{\mathscr{L},\epsilon}(x,y) = \sum_{i=1}^\infty \frac{\lambda_i - 1}{\epsilon}\phi_i(x)\phi_i(y)$. Hence,

$$K_{\mathscr{L}^{-1},\epsilon}(x,y) = K^{-1}_{\mathscr{L},\epsilon}(x,y) = \sum_{i=1}^\infty \frac{\epsilon}{\lambda_i - 1}\phi_i(x)\phi_i(y),$$

and we obtain

$$\nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x,y) = \sum_{i=1}^{\infty} \frac{\epsilon}{\lambda_i - 1} \nabla\phi_i(x)\phi_i(y),$$

and

$$\nabla L_\epsilon^{-1} f(x) = \int \nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x,y) f(y) d\pi(y).$$

Under regularity assumptions (see Appendix E), $\lim_{\epsilon\to 0} \nabla L_\epsilon f(x) = \nabla \mathscr{L} f(x)$ and $\lim_{\epsilon\to 0} \nabla L_\epsilon^{-1} f(x) = \nabla \mathscr{L}^{-1} f(x)$.

3. The generative model

We can now write the discrete-time update step of LAWGD (11) using an ϵ-kernel approximation

$$x_{t+1}^i = x_t^i - \frac{h}{M} \sum_{j=1}^{M} \nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x_t^i, x_t^j), \quad i = 1,\dots,M$$

This update describes the dynamics of M interacting particles. Note that the kernel $K_{\mathscr{L}^{-1},\epsilon}$ is constructed only at the locations of the training samples $\{z^i\}_{i=1}^N \sim \pi$. To obtain an implementable algorithm, we need to be able to compute $\nabla_1 K_{\mathscr{L}^{-1},\epsilon}(\cdot,\cdot)$ for arbitrary points x^*, y^*. One way is to interpolate the eigenfunctions ϕ and their gradients $\nabla\phi$ at x^* and y^*. However, this is restricted by the number of training samples for learning the kernel, as well as the interpolation method. To overcome this problem, we propose yet another natural way of computing $\nabla_1 K_{\mathscr{L}^{-1},\epsilon}(\cdot,\cdot)$ by taking advantage of the eigendecomposition of the kernel, avoiding interpolation of eigenfunctions.

3.1. Computing $\nabla_1 \mathcal{K}_{\mathscr{L}^{-1}}(x,y)$ for arbitrary points x, y

Set $\sigma_i = \frac{\lambda_i - 1}{\epsilon}$, and recall from (3) that $P_\epsilon(x,y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(y)$. Consider the following eigendecomposition of the kernel:

$$\begin{aligned}
&\nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x^*, y^*) \\
&= \int_{\mathcal{Z}} \int_{\mathcal{W}} \left(\sum_{k=1}^{\infty} \sigma_k^{-1} \nabla\phi_k(x^*)\phi_k(y^*) \right) \\
&\quad \times \left(\sum_{j=1}^{\infty} \lambda_j^{-1}\sigma_j^{-1}\lambda_j^{-1}\phi_j(w)\phi_j(z) \right) \\
&\quad \times \left(\sum_{i=1}^{\infty} \lambda_i \phi_i(z)\phi_i(y^*) \right) d\pi(w)\, d\pi(z) \\
&= \int_{\mathcal{Z}} \int_{\mathcal{W}} \nabla_1 P_\epsilon(x^*, w) \left(\sum_{j=1}^{\infty} \lambda_j^{-1}\sigma_j^{-1}\lambda_j^{-1}\phi_j(w)\phi_j(z) \right) \\
&\quad \times P_\epsilon(z, y^*) d\pi(w)\, d\pi(z).
\end{aligned} \tag{4}$$

Algorithm 1 DMPS

Input: Training samples $\{z^i\}_{i=1}^N \sim \pi$ and initial particles $\{x_0^i\}_{i=1}^M$, tolerance tol, bandwidth ϵ, step size h
Output: $\{x_T^i\}_{i=1}^M$
Construct $P_{\epsilon,N}$ using $\{z^i\}_{i=1}^N$ as in (12).
Compute the eigenpairs $\{\lambda_i, \phi_i\}$ such that $P_{\epsilon,N}(x,y) = \sum_{i=1}^N \lambda_i \phi_i(x)\phi_i(y)$, i.e., performing singular value decomposition on the kernel matrix $P_{\epsilon,N}$.
while tol not met **do**
  for $i = 1,\dots,M$ **do**
    $x_{t+1}^i = x_t^i - \frac{h}{M} \sum_{j=1}^M \nabla_1 K_{\mathscr{L}^{-1},\epsilon,N}(x_t^i, x_t^j)$ as in (5).
  end for
end while

where we have used the orthogonality of the eigenfunctions.

As noted previously, we use $\{z^i\}_{i=1}^N$ to represent the training samples and use $\{x_t^i\}_{i=1}^M$ to represent the generated samples at time t. Focusing on a single time step, we drop the dependence on t. Then the empirical approximation of (4) is as follows:

$$\begin{aligned}
&\nabla_1 K_{\mathscr{L}^{-1},\epsilon,N}(x^i, x^k) \\
&= \sum_{j_1=1}^{N} \sum_{j_2=1}^{N} \nabla_1 P_{\epsilon,N}(x^i, z^{j_1}) \\
&\quad \times \left(\sum_{j_3=1}^{N} \phi_{j_3}(z^{j_1}) \lambda_{j_3}^{-1}\sigma_{j_3}^{-1}\lambda_{j_3}^{-1}\phi_{j_3}(z^{j_2}) \right) P_{\epsilon,N}(z^{j_2}, x^k).
\end{aligned} \tag{5}$$

In a matrix representation, the three matrices (from left to right) above are of size $M \times N$, $N \times N$, and $N \times M$. The ingredients for computing the expression above are $P_{\epsilon,N}(\cdot,\cdot), \nabla_1 P_{\epsilon,N}(\cdot,\cdot), \lambda_i, \sigma_i, \phi_i$. Since $P_{\epsilon,N}$ is constructed using Gaussian kernels, its derivative with respect to the first argument $\nabla_1 P_{\epsilon,N}(\cdot,\cdot)$ can be computed in closed form (see Appendix C.1).

Algorithm 1 summarizes our proposed scheme, called a diffusion map particle system (DMPS). We offer several comments. The classical analysis of diffusion maps requires the underlying distribution to have bounded support, but we find that this algorithm works well even when the support of π is (in principle) unbounded. We suggest to initialize the samples $\{x_0^i\}_{i=1}^M$ inside the support of π. Even though initializing samples outside the support would work because of the finite bandwidth ϵ, starting the samples inside the support generally makes the algorithm more stable. We typically choose the bandwidth as $\epsilon = \mathrm{med}^2/(2\log N)$, following the heuristics proposed in Liu (2017), where med is the median of the pairwise distances between training

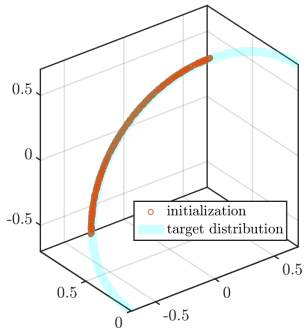
samples. A convergence analysis of Algorithm 1 is given in Appendix D.

4. Numerical experiments

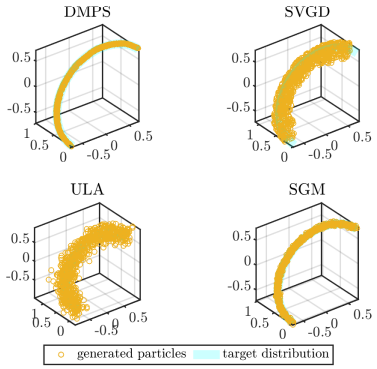
We present five numerical examples (three in the main paper, two in the appendices) exploring the performance of the DMPS algorithm on connected and disconnected domains and on manifolds. To benchmark the performance of DMPS, we compare it with: (i) SVGD and (ii) ULA, where the score ∇V required by both algorithms is replaced by its empirical approximation using the diffusion map, as well as with (iii) the score-based generative model (SGM) of Song & Ermon (2019). We implement SGM using a lightweight notebook from Pidstrigach (2022b;a). To make (i) and (ii) more precise: recall that the a diffusion map can be used as a tool for approximating the Langevin generator from samples. That is, $\mathscr{L}f = \nabla^2 f - \langle \nabla V, \nabla f \rangle$. Then note that by letting f be the identity, i.e., $f(x) = x$, we have that $\mathscr{L}f = \nabla \log \pi$. Therefore, we can use samples to approximate the gradient of the potential. For DMPS and SVGD, we run the algorithm until a prescribed tolerance is met; we run ULA and SGM for a fixed number of iterations. To evaluate the quality of samples generated with each method on synthetic dataset, we compute the regularized optimal transport (OT) distance between the generated samples and a large number of "reference samples" from the target distribution. Details on the evaluation of this error metric are in Appendix F.1. We then present a real world example in the field of high energy physics and show that the generated samples resemble those from the target distribution.

4.1. The arc: one-dimensional manifold embedded in a three-dimensional space

We consider an example where the data lie on a manifold, in this case an arc of radius 1 embedded in $\mathbb{R}^3$. Training data are drawn uniformly from the arc and perturbed in the radial direction only, with $U(0, 10^{-2})$ noise. Results are obtained with both 100 and 1000 training samples. The initial particles and the target distribution are shown in Figure 1 (left). We then run DMPS, SVGD, ULA, and SGM for each batch of training and initialization samples, visualizing an instance in Figure 1. Particles generated by the two deterministic methods, DMPS and SVGD, lie only on the two-dimensional plane of the training data, but the particles generated using SVGD do not fully explore the target distribution. Particles generated by the two stochastic methods, ULA and SGM, span the full three-dimensional space due to the added noise. Errors are plotted in Figure 3, for both choices of training set size. We see that DMPS exhibits the smallest errors, and that this error decreases as we increase the number of generated particles. SGM shows a similar trend, but with larger errors. The performance of



Initialization and target distribution



Generated particles

Figure 1: The arc: 900 generated particles using DMPS, SVGD, ULA, and SGM with 1000 training samples.

ULA does not seem to improve after using more training samples, which might be due to finite discretization timestep (although it was chosen small relative to the width of the target, $h = 5 \times 10^{-4}$). SVGD gives the largest errors, which do not seem to decrease with more particles.

The reason why DMPS performs better might be that the kernel method for approximating the generator relies on diffusion maps, or more broadly, the graph Laplacian, which is widely used for manifold learning due to its flexibility in detecting the underlying geometry. The fact that the generator is approximated as a whole distinguishes it from other score-based methods. SGM also produces generally good results. However, it does require orders of magnitude more computation. For the arc problem, with 1000 training samples and 100 initial particles, running DMPS took 6.21 seconds, while training SGM (running 5000 epochs) took 1633 seconds (27 minutes).

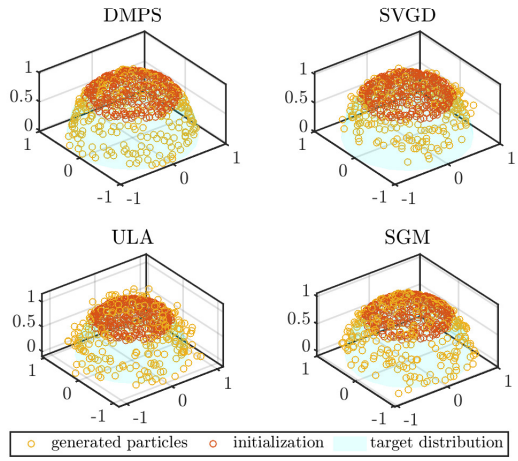


Figure 2: Hyper-semisphere: 300 generated samples using DMPS, SVGD, ULA and SGM in three dimensions with 1000 training samples.

	DMPS	SVGD	ULA	SGM
$d = 3$	**0.018 ± 0.000**	0.146 ± 0.023	0.033 ± 0.001	0.032 ± 0.002
$d = 6$	**0.142 ± 0.000**	0.267 ± 0.017	0.185 ± 0.001	0.170 ± 0.002
$d = 9$	**0.303 ± 0.000**	0.361 ± 0.008	0.378 ± 0.001	0.348 ± 0.003
$d = 12$	**0.441 ± 0.000**	0.811 ± 0.078	0.555 ± 0.002	0.496 ± 0.004
$d = 15$	**0.564 ± 0.001**	0.986 ± 0.005	0.713 ± 0.002	0.608 ± 0.002

Table 1: Hyper-semisphere: error comparison (± standard error) between DMPS, SVGD, ULA, and SGM.

4.2. Hyper-semisphere: 2 to 14-dimensional manifolds

We next study an example where data are uniformly sampled on a half-sphere embedded in ambient dimensions $d \in \{3, 6, 9, 12, 15\}$; in each case, the manifold is thus of dimension $d - 1$. For this problem, the number of training samples and the number of generated particles are fixed to 1000 and 300, respectively. A visualization for $d = 3$ can be seen in Figure 2. We show the error (in OT distance), and the standard error of the mean error over 10 trials, in Table 1. For all dimensions, DMPS enjoys the smallest error and the smallest standard error, followed by SGM and ULA, which also produce relatively small errors and stable results. SVGD has the largest error and does not produce stable results (large variability over the 10 trials).

4.3. High energy physics: gluon jet dataset

We now study a real-world example from high energy physics, where the goal is to generate relative angular coordinates $\eta^{\text{rel}}, \phi^{\text{rel}}$ and relative transverse momenta p_T^{rel} of elementary particles produced in a gluon jet. Details on the dataset are in Kansal et al. (2021). For this experiment, we first normalize the training data so that they have mean 0 and variance 1. We use 1000 samples for training and initialize 100, 300, 900, and 2700 particles respectively from $U(-1, 1)^3$ for the generative process. Target samples and generated samples compare favorably, as shown in Appendix G.3. We also compared DMPS, SVGD, ULA and SGM with the same setup, and the errors (± standard error) are shown in Table 2. We see that DMPS achieves the best performance on both cases in this problem.

# particles	DMPS	SVGD	ULA	SGM
100	**0.0027** $\pm 9.30 \times 10^{-5}$	0.0071 $\pm 2.40 \times 10^{-5}$	0.0049 $\pm 2.61 \times 10^{-4}$	0.0039 $\pm 2.09 \times 10^{-4}$
300	**0.0021** $\pm 6.66 \times 10^{-5}$	0.0068 $\pm 2.73 \times 10^{-5}$	0.0044 $\pm 2.02 \times 10^{-4}$	0.0030 $\pm 0.83 \times 10^{-4}$
900	**0.0020** $\pm 5.78 \times 10^{-5}$	0.0066 $\pm 2.78 \times 10^{-5}$	0.0036 $\pm 1.57 \times 10^{-4}$	0.0028 $\pm 1.34 \times 10^{-4}$
2700	**0.0019** $\pm 5.98 \times 10^{-5}$	0.0066 $\pm 3.04 \times 10^{-5}$	0.0034 $\pm 1.25 \times 10^{-4}$	0.0026 $\pm 1.20 \times 10^{-4}$

Table 2: Gluon jet dataset: error comparison (± standard error) between DMPS, SVGD, ULA, and SGM, for different numbers of generated particles.

5. Conclusion

We introduced DMPS as a *simple-to-implement* and *computationally efficient* kernel method for generative modeling. Our approach combines diffusion maps with the LAWGD approach to construct a generative particle system that adapts to the geometry of the underlying distribution. Our method compares favorably with other competing schemes (SVGD and ULA with learned scores, and diffusion-based generative models) on synthetic datasets, consistently achieving the smallest errors in terms of regularized OT distance. While the examples presented here are of moderate dimension (up to $d = 15$ for the example in Section 4.2), we expect that more sophisticated kernel methods (Li et al., 2017) can slot naturally into the DMPS framework and help extend the method to higher-dimensional problems. Future research will also study the convergence rate of the method in discrete time and with finite samples, and consider more complex geometric domains.

References

Ambrosio, L., Gigli, N., and Savare, G. Gradient flows in metric spaces and in the space of probability measures. 01 2005. doi: 10.1007/978-3-7643-8722-8.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. doi: 10.1162/089976603321780317.

Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S. J., Snavely, N., and Hariharan, B. Learning gradient fields for shape generation. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J. (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings,*

Part III, volume 12348 of *Lecture Notes in Computer Science*, pp. 364–381. Springer, 2020. doi: 10.1007/978-3-030-58580-8_22. URL https://doi.org/10.1007/978-3-030-58580-8_22.

Caterini, A. L., Loaiza-Ganem, G., Pleiss, G., and Cunningham, J. P. Rectangular flows for manifold learning. In *Neural Information Processing Systems*, 2021.

Chewi, S., Gouic, T. L., Lu, C., Maunu, T., and Rigollet, P. SVGD as a kernelized Wasserstein gradient flow of the chi-squared divergence. *ArXiv*, abs/2006.02509, 2020.

Coifman, R. R. and Lafon, S. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2006.04.006. URL https://www.sciencedirect.com/science/article/pii/S1063520306000546. Special Issue: Diffusion Maps and Wavelets.

Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., and Zucker, S. W. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005. doi: 10.1073/pnas.0500334102. URL https://www.pnas.org/doi/abs/10.1073/pnas.0500334102.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf.

Daneri, S. and Savaré, G. Lecture notes on gradient flows and optimal transport, 2010. URL https://arxiv.org/abs/1009.3737.

Franzese, G., Rossi, S., Yang, L., Finamore, A., Rossi, D., Filippone, M., and Michiardi, P. How much is enough? a study on diffusion times in score-based generative models, 2022. URL https://arxiv.org/abs/2206.05173.

Garbuno-Iñigo, A., Nüsken, N., and Reich, S. Affine invariant interacting Langevin dynamics for Bayesian inference. *SIAM J. Appl. Dyn. Syst.*, 19:1633–1658, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Hein, M., Audibert, J.-Y., and von Luxburg, U. From graphs to manifolds - weak and strong pointwise consistency of graph laplacians. In *Annual Conference Computational Learning Theory*, 2005.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *ArXiv*, abs/1902.00275, 2019.

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23:47:1–47:33, 2021.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the fokker–planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998. doi: 10.1137/S0036141096303359. URL https://doi.org/10.1137/S0036141096303359.

Kansal, R., Duarte, J., Su, H., Orzari, B., Tomei, T., Pierini, M., Touranakou, M., Gunopulos, D., et al. Particle cloud generation with message passing generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:23858–23871, 2021.

Khandelwal, A. and Krishnan, R. G. Fine-tuning generative models. 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *CoRR*, abs/1312.6114, 2013.

Knight, P. A. The Sinkhorn-Knopp algorithm: Convergence and applications. *SIAM J. Matrix Anal. Appl.*, 30:261–275, 2008.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. MMD GAN: Towards deeper understanding of moment matching network. In *NIPS*, 2017.

Liu, Q. Stein variational gradient descent as gradient flow. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/17ed8abedc255908be746d245e50263a-Paper.pdf.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *NIPS*, 2016.

Miao, Y. and Blunsom, P. Language as a latent variable: Discrete generative models for sentence compression. pp. 319–328, 01 2016. doi: 10.18653/v1/D16-1031.

Nadler, B., Lafon, S., Kevrekidis, I., and Coifman, R. Diffusion maps, spectral clustering and eigenfunctions of Fokker-Planck operators. In Weiss, Y., Schölkopf, B., and Platt, J. (eds.), *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2005/file/2a0f97f81755e2878b264adf39cba68e-Paper.pdf.

Nadler, B., Lafon, S., Coifman, R. R., and Kevrekidis, I. G. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis*, 21(1):113–127, 2006. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2005.07.004. URL https://www.sciencedirect.com/science/article/pii/S1063520306000534. Special Issue: Diffusion Maps and Wavelets.

Peres, Y. Conditions for convergence of a derivative, given the function itself is convergent. Mathematics Stack Exchange. URL https://math.stackexchange.com/q/4517640. (version: 2022-08-24).

Pidstrigach, J. Score-based generative models detect manifolds. *ArXiv*, abs/2206.01018, 2022a.

Pidstrigach, J. Score-based generative models introduction. https://jakiw.com/sgm_intro, 2022b. Accessed: 2024-01-24.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/rezende14.html.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 5500:2323–6, 2000.

Ruthotto, L. and Haber, E. An introduction to deep generative modeling. *GAMM-Mitteilungen*, 44(2):e202100008, 2021. doi: https://doi.org/10.1002/gamm.202100008. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/gamm.202100008.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. Improved techniques for training GANs. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

Santambrogio, F. Euclidean, Metric, and Wasserstein gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7, 09 2016. doi: 10.1007/s13373-017-0101-1.

Shi, Z. Convergence of Laplacian spectra from random samples. *ArXiv*, abs/1507.00151, 2020.

Singer, A. From graph to manifold laplacian: The convergence rate. *Applied and Computational Harmonic Analysis*, 21:128–134, 2006.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12438–12448. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/92c3b916311a5517d9290576e3ea37ad-Paper.pdf.

Song, Y., Sohl-Dickstein, J. N., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *ArXiv*, abs/2011.13456, 2020.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. Reduction a global geometric framework for nonlinear dimensionality. 2011.

Villani, C. Topics in optimal transportation. 2003.

Wibisono, A. Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem. In *Annual Conference Computational Learning Theory*, 2018.

Wu, Z., Johnston, K. E., Arnold, F. H., and Yang, K. K. Protein sequence design with deep generative models. *Current Opinion in Chemical Biology*, 65:18–27, 2021. ISSN 1367-5931. doi: https://doi.org/10.1016/j.cbpa.2021.04.004. URL https://www.sciencedirect.com/science/article/pii/S136759312100051X. Mechanistic Biology * Machine Learning in Chemical Biology.

Yi, X., Walia, E., and Babyn, P. Generative adversarial network in medical imaging: A review. *Medical Image Analysis*, 58:101552, 2019. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2019.101552. URL https://www.sciencedirect.com/science/article/pii/S1361841518308430.

Yogatama, D., Dyer, C., Ling, W., and Blunsom, P. Generative and discriminative text classification with recurrent neural networks. *ArXiv*, abs/1703.01898, 2017.

A. Details of Wasserstein gradient flow and LAWGD algorithm

A.1. Gradient flow on Wasserstein space

Let $F(\mu)$ be a functional over the space of probability measures, i.e., $F : \mathcal{P}_2(\mathbb{R}^d) \to \mathbb{R}$. We seek to steer the measure μ_t (at time t) in the direction of steepest descent, defined by F and a chosen metric. That is, $\frac{\partial \mu_t}{\partial t} = -\nabla_{W_2} F(\mu_t)$, where ∇_{W_2} denotes the general gradient in Wasserstein metric. Under some smoothness assumptions, we can write this as

$$\frac{\partial \mu_t}{\partial t} = \operatorname{div}(\mu_t \nabla \delta F(\mu_t)), \tag{6}$$

where $\delta F(\mu)$ is the first variation of F evaluated at μ (Villani, 2003). If we choose the functional F to be the Kullback–Leibler (KL) divergence, $F(\mu) = \mathrm{D}_{\mathrm{KL}}(\mu||\pi)$, then (6) becomes

$$\frac{\partial \mu_t}{\partial t} = \operatorname{div}\left(\mu_t \nabla \log \frac{d\mu_t}{d\pi}\right),$$

which is the Fokker–Planck equation (Jordan et al., 1998). The measure μ_t can be approximated by particles $\{x^i(t)\}$ evolving according to the following dynamics,

$$\dot{x} = -\nabla \log \frac{d\mu_t}{d\pi}(x).$$

Forward Euler discretization with stepsize h then yields the following numerical scheme,

$$x_{t+1} = x_t - h\nabla \log \frac{d\mu_t}{d\pi}(x_t).$$

A.2. LAWGD algorithm

One challenge with the scheme above is that the measure μ_t is intractable at time t. To solve this problem, SVGD (Liu & Wang, 2016) implements the following kernelized dynamics (in the continuum limit),

$$\dot{x} = -\int K(x,y)\nabla \log \frac{d\mu_t}{d\pi}(y) d\mu_t(y).$$

The expression above can be equivalently written as

$$\dot{x} = -\int K(x,y)\nabla \frac{d\mu_t}{d\pi}(y) d\pi(y). \tag{7}$$

Define

$$\mathcal{K}_\pi f(x) \coloneqq \int K(x,y) f(y) d\pi(y).$$

Then we write (7) as

$$\dot{x} = -\mathcal{K}_\pi \nabla \frac{d\mu_t}{d\pi}(x),$$

and under SVGD, the density evolves according to

$$\partial_t \mu_t = \operatorname{div}\left(\mu_t \mathcal{K}_\pi \nabla \frac{d\mu_t}{d\pi}\right).$$

On the other hand, LAWGD makes the JKO scheme implementable by considering the following kernelization

$$\dot{x} = -\nabla \mathcal{K}_\pi \frac{d\mu_t}{d\pi}(x),$$

and expanding it to obtain

$$\begin{aligned}\dot{x} &= -\int \nabla_1 K(x,y)\frac{d\mu_t}{d\pi}(y)d\pi(y)\\ &= -\int \nabla_1 K(x,y)d\mu_t(y).\end{aligned} \tag{8}$$

The kernel $\mathcal{K}_\pi$ is specifically chosen to be $-\mathscr{L}^{-1}$, where $\mathscr{L} = \nabla^2 - \langle \nabla V, \nabla \cdot \rangle$ is generator of the Langevin diffusion process. Here we assume $\mathscr{L}$ has discrete spectrum (see Appendix B). This choice is motivated by the rate of change of KL divergence,

$$\partial_t \, \mathrm{D}_{\mathrm{KL}}(\mu_t||\pi) = -\mathbb{E}_\pi \left[\frac{d\mu_t}{d\pi} \mathscr{L}\mathcal{K}_\pi \frac{d\mu_t}{d\pi} \right]. \tag{9}$$

Indeed, such a choice yields

$$\begin{aligned}\partial_t \, \mathrm{D}_{\mathrm{KL}}(\mu_t||\pi) &= -\mathbb{E}_\pi \left[\left(\frac{d\mu_t}{d\pi} \right)^2 \right]\\ &\leq -\mathbb{E}_\pi \left[\left(\frac{d\mu_t}{d\pi} - 1 \right)^2 \right] = -\chi^2(\mu_t||\pi).\end{aligned} \tag{10}$$

The evolution of the density under LAWGD thus follows

$$\partial_t \mu_t = \mathrm{div} \left(\mu_t \nabla \mathscr{L}^{-1} \frac{d\mu_t}{d\pi} \right).$$

Now suppose we initialize $\{x_0^i\}_{i=1}^M \sim \mu_0$. We then obtain a discrete algorithm from (8), where the update step reads

$$x_{t+1}^i = x_t^i - \frac{h}{M} \sum_{j=1}^M \nabla_1 K_{\mathscr{L}^{-1}}(x_t^i, x_t^j). \tag{11}$$

Here $K_{\mathscr{L}^{-1}}$ can be understood as a kernelized version of $\mathscr{L}^{-1}$, satisfying $\mathscr{L}^{-1} f(x) = \int K_{\mathscr{L}^{-1}}(x,y) f(y) d\pi(y)$. In particular, setting $f = \frac{d\mu_t}{d\pi}$, we have $\mathscr{L}^{-1} \frac{d\mu_t}{d\pi}(x) = \int K_{\mathscr{L}^{-1}}(x,y) \frac{d\mu_t}{d\pi}(y) d\pi(y) = \int K_{\mathscr{L}^{-1}}(x,y) d\mu_t(y)$. More details can be found in Chewi et al. (2020).

B. Spectral properties of $\mathscr{L}$

Suppose $\mathcal{D}$ is compact and $\mathscr{L} = \nabla^2 - \langle \nabla V, \nabla \cdot \rangle$ has discrete spectrum $\{\sigma_i\}_{i=1}^\infty$. Then the spectrum is bounded $\sigma_0 = 0 \leq \sigma_1 \leq \sigma_2 \leq \cdots \leq B < \infty$ (Shi, 2020). Let ϕ_i be their corresponding eigenfunctions on $\mathcal{D}$. The action of $\mathscr{L}$ on a function $f \in L^2(\pi)$ reads

$$\mathscr{L} f(x) = \sum_{i=1}^\infty \sigma_i \langle \phi_i, f \rangle_{L^2(\pi)} \phi_i(x),$$

and its inverse has the following expression

$$\mathscr{L}^{-1} f(x) = \sum_{i=1}^\infty \sigma_i^{-1} \langle \phi_i, f \rangle_{L^2(\pi)} \phi_i(x).$$

C. Finite sample approximation of the operator

In this section, we introduce the finite sample counterpart to 2.1. We consider the approximation of the generator of the Langevin diffusion process $\mathscr{L}$ from finite samples $\{z^i\}_{i=1}^N \sim \pi$. In this case, we have

$$M_{\epsilon,N}(x,y) = \frac{K_\epsilon(x,y)}{\sqrt{\sum_{i=1}^N K_\epsilon(z^i,y)} \sqrt{\sum_{i=1}^N K_\epsilon(x,z^i)}},$$

and the corresponding $L^f_{\epsilon,N}$ and $L^b_{\epsilon,N}$ can be written as

$$P^f_{\epsilon,N}(x,y) := \frac{M_{\epsilon,N}(x,y)}{\sum_{i=1}^N M_{\epsilon,N}(z^i,y)},$$
$$P^b_{\epsilon,N}(x,y) := \frac{M_{\epsilon,N}(x,y)}{\sum_{i=1}^N M_{\epsilon,N}(x,z^i)},$$

and we set

$$P_{\epsilon,N}(x,y) = \frac{1}{2}\left(P^f_{\epsilon,N}(x,y) + P^b_{\epsilon,N}(x,y)\right). \tag{12}$$

Similarly, its action on a function g writes

$$T^{f,b}_{\epsilon,N}g(x) = \sum_{i=1}^N P^{f,b}_{\epsilon,N}(x,z^i)g(z^i).$$

Let

$$L^{f,b}_{\epsilon,N} := \frac{T^{f,b}_{\epsilon,N} - \mathrm{Id}}{\epsilon}.$$

Similar to their spatial continuum limit, we have

$$\lim_{\epsilon\to 0, N\to\infty} L^f_{\epsilon,N} = \lim_{\epsilon\to 0, N\to\infty} L^b_{\epsilon,N} = \mathscr{L}.$$

Then, similar to (2), we have that

$$\lim_{\epsilon\to 0, N\to\infty} \frac{\sum_{i=1}^N P^{f,b}_{\epsilon,N}(x,z^i)g(z^i) - g(x)}{\epsilon} = \mathscr{L}g(x),$$

and

$$L_{\epsilon,N} = \frac{1}{2}(L^f_{\epsilon,N} + L^b_{\epsilon,N}).$$

is the symmetric kernel.

C.1. Computing $\nabla_1 P_\epsilon(x,y)$

Recall that $P_\epsilon = \frac{1}{2}(P^f_\epsilon + P^b_\epsilon)$. We then compute $\nabla_1 P^f_\epsilon(x,y)$ and $\nabla_1 P^b_\epsilon(x,y)$ separately.

C.1.1. Computing $\nabla_1 P^f_\epsilon(x,y)$

Recall

$$P^f_\epsilon(x,y) := \frac{M_\epsilon(x,y)}{\int_X M_\epsilon(x,y)d\pi(x)},$$

where

$$M_\epsilon(x,y) := \frac{K_\epsilon(x,y)}{\sqrt{\int_{\mathcal{X}} K_\epsilon(x,y)d\pi(x)}\sqrt{\int_{\mathcal{Y}} K_\epsilon(x,y)d\pi(y)}}. \tag{13}$$

Then

$$\nabla_1 P^f_\epsilon(x,y) = \frac{\nabla_1 M_\epsilon(x,y)}{\int_X M_\epsilon(x,y)d\pi(x)}.$$

We then compute $\nabla_1 M_\epsilon(x,y)$. Let $d_\epsilon(x) = \int_{\mathcal{Y}} K_\epsilon(x,y)d\pi(y)$ and $d_\epsilon(y) = \int_{\mathcal{X}} K_\epsilon(x,y)d\pi(x)$, then

$$\nabla_1 M_\epsilon(x,y) = \frac{\nabla_1 K_\epsilon(x,y)\sqrt{d_\epsilon(y)}\sqrt{d_\epsilon(x)} - \left(\frac{\partial}{\partial x}\sqrt{d_\epsilon(x)}\right)\sqrt{d_\epsilon(y)}K_\epsilon(x,y)}{d_\epsilon(x)d_\epsilon(y)},$$

where

$$\nabla_1 K_\epsilon(x,y) = -\left(\frac{x-y}{\epsilon}\right)e^{-\frac{\|x-y\|^2}{2\epsilon}}, \tag{14}$$

$$\frac{\partial}{\partial x}\left(\sqrt{d_\epsilon(x)}\right) = \frac{1}{2}\int_{\mathcal{Y}}\left(-\left(\frac{x-y}{\epsilon}\right)e^{-\frac{\|x-y\|^2}{2\epsilon}}\right)^{-1/2} d\pi(y). \tag{15}$$

C.1.2. COMPUTING $\nabla_1 P^b_\epsilon(x,y)$

On the other hand, we have

$$P^b_\epsilon(x,y) := \frac{M_\epsilon(x,y)}{\int_{\mathcal{Y}} M_\epsilon(x,y)d\pi(y)},$$

and

$$\nabla_1 P^b_\epsilon(x,y) = \frac{\nabla_1 M_\epsilon(x,y)\int_{\mathcal{Y}} M_\epsilon(x,y)d\pi(y) - \int_{\mathcal{Y}}\nabla_1 M_\epsilon(x,y)d\pi(y)M_\epsilon(x,y)}{\left(\int_{\mathcal{Y}} M_\epsilon(x,y)d\pi(y)\right)^2},$$

where all the ingredients are computable from (13), (14), and (15).

C.2. Computing $\nabla_1 P_{\epsilon,N}(x,y)$

The discrete version is obtained by replacing the integral with its empirical average. Similarly, let $\{z^i\}_{i=1}^N \sim \pi$, and define $d_{\epsilon,N}(x) = \sum_{i=1}^N K_\epsilon(x,z^i)$ and $d_{\epsilon,N}(y) = \sum_{i=1}^N K_\epsilon(z^i,y)$. Recall that

$$M_{\epsilon,N}(x,y) = \frac{K_\epsilon(x,y)}{\sqrt{\sum_{i=1}^N K_\epsilon(z^i,y)}\sqrt{\sum_{i=1}^N K_\epsilon(x,z^i)}},$$

then

$$\nabla_1 M_{\epsilon,N}(x,y) = \frac{\nabla_1 K_{\epsilon,N}(x,y)\sqrt{d_{\epsilon,N}(y)}\sqrt{d_{\epsilon,N}(x)} - \left(\frac{\partial}{\partial x}\sqrt{d_{\epsilon,N}(x)}\right)\sqrt{d_{\epsilon,N}(y)}K_\epsilon(x,y)}{d_{\epsilon,N}(x)d_{\epsilon,N}(y)},$$

and

$$\frac{\partial}{\partial x}\left(\sqrt{d_{\epsilon,N}(x)}\right) = \frac{1}{2}\sum_{i=1}^N\left(-\left(\frac{x-z^i}{\epsilon}\right)e^{-\frac{\|x-z^i\|^2}{2\epsilon}}\right)^{-1/2}.$$

Finally, similar to the previous section, we have that

$$\nabla_1 P^f_{\epsilon,N}(x,y) = \frac{\nabla_1 M_{\epsilon,N}(x,y)}{\sum_{i=1}^N M_{\epsilon,N}(z^i,y)},$$

and

$$\nabla_1 P^b_{\epsilon,N}(x,y) = \frac{\nabla_1 M_{\epsilon,N}(x,y)\left(\sum_{i=1}^N M_{\epsilon,N}(x,z^i)\right) - \left(\sum_{i=1}^N \nabla_1 M_{\epsilon,N}(x,z^i)\right)M_{\epsilon,N}(x,y)}{\left(\sum_{i=1}^N M_{\epsilon,N}(x,z^i)\right)^2}.$$

D. Convergence analysis

We comment on the convergence rate of our scheme at the population level. From (9), we see that if the kernel is exact, then the rate of change of the KL divergence is $-\mathbb{E}_\pi\left[\frac{d\mu_t}{d\pi}\mathscr{L}\mathcal{K}_\pi\frac{d\mu_t}{d\pi}\right]$. If we replace $\mathcal{K}_\pi$ by its kernel approximation L_ϵ^{-1}, then the resulting rate of change is

$$\partial_t \,\mathrm{D_{KL}}(\hat{\mu}_t||\pi) = -\mathbb{E}_\pi\left[\frac{d\hat{\mu}_t}{d\pi}\mathscr{L}L_\epsilon^{-1}\frac{d\hat{\mu}_t}{d\pi}\right],$$

where $\hat{\mu}_t$ is the distribution at time t obtained from the following evolution

$$\dot{x} = \int \nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x,y)d\hat{\mu}_t(y).$$

Classical results from diffusion map literature (Hein et al., 2005; Singer, 2006) reveal that the bias $\left|\mathscr{L}f(x) - L_\epsilon f(x)\right| \sim \mathcal{O}(\epsilon)$ if data lie on a compact manifold. Using the same assumptions, we state the following theorem.

Theorem D.1. *Suppose the target distribution π is supported on a compact manifold. Let μ_0 be the initial distribution of the particles and $\hat{\mu}_t$ be the distribution of the generated process at time t, and assume that $\frac{d\hat{\mu}_t}{d\pi}$ is finite and twice differentiable for all t. Then we have*

$$\mathrm{D_{KL}}(\hat{\mu}_t||\pi) \leq \left(\mathcal{O}(\epsilon) + \mathrm{D_{KL}}(\mu_0||\pi)\right)e^{-t}.$$

Proof. From diffusion map approximation, we have $\left|\mathscr{L}\frac{d\hat{\mu}_t}{d\pi}(x) - L_\epsilon\frac{d\hat{\mu}_t}{d\pi}(x)\right| \sim \mathcal{O}(\epsilon)$. Then obtain that ,

$$\mathscr{L}\frac{d\hat{\mu}_t}{d\pi}(x) = \left(L_\epsilon + \mathcal{O}(\epsilon)\right)\frac{d\hat{\mu}_t}{d\pi}(x)$$

by factoring out $\frac{d\hat{\mu}_t}{d\pi}(x)$. Then using Neumann series or binomial expansion

$$\begin{aligned}\mathscr{L}^{-1}\frac{d\hat{\mu}_t}{d\pi}(x) &= \left(L_\epsilon + \mathcal{O}(\epsilon)\right)^{-1}\frac{d\hat{\mu}_t}{d\pi}(x)\\ &= \left(L_\epsilon^{-1} - \mathcal{O}(\epsilon)L_\epsilon^{-2} + \mathcal{O}(\epsilon)^2\right)\frac{d\hat{\mu}_t}{d\pi}(x)\\ &= \left(L_\epsilon^{-1} + \mathcal{O}(\epsilon)\right)\frac{d\hat{\mu}_t}{d\pi}(x).\end{aligned}$$

This follows from the fact that the inverse L_ϵ^{-1} is bounded. We then have

$$\mathscr{L}^{-1} = L_\epsilon^{-1} + \mathcal{O}(\epsilon).$$

We can then write

$$\begin{aligned}&\mathbb{E}_\pi\left[\frac{d\hat{\mu}_t}{d\pi}\mathscr{L}L_\epsilon^{-1}\frac{d\hat{\mu}_t}{d\pi}\right]\\ =&\mathbb{E}_\pi\left[\frac{d\hat{\mu}_t}{d\pi}\mathscr{L}\left(\mathscr{L}^{-1}\frac{d\hat{\mu}_t}{d\pi} + \mathcal{O}(\epsilon)\right)\right]\\ =&\mathbb{E}_\pi\left[\left(\frac{d\hat{\mu}_t}{d\pi}\right)^2\right] + \mathcal{O}(\epsilon),\end{aligned}$$

and consequently by (10),

$$\partial_t \,\mathrm{D_{KL}}(\hat{\mu}_t||\pi) = -\chi^2(\hat{\mu}_t||\pi) + \mathcal{O}(\epsilon).$$

Then using the results (Theorem 1) in Chewi et al. (2020) and Gronwall's inequality, we have

$$\mathrm{D_{KL}}(\hat{\mu}_t||\pi) \leq \left(\mathcal{O}(\epsilon) + \mathrm{D_{KL}}(\mu_0||\pi)\right)e^{-t}.$$

□

E. Regularity assumptions

We state here the regularity assumptions needed for the gradient to converge. The statement and the proof are adapted from Peres.

Theorem E.1. *Suppose $L_\epsilon f(x)$ is a family of bounded differentiable functions from $\mathcal{D}$ to $\mathbb{R}$ converging pointwise to $\mathscr{L} f(x)$ as $\epsilon \to 0$. Furthermore, suppose $\nabla L_\epsilon f(x)$ is a family of uniformly equicontinuous functions. Then $\mathscr{L} f(x)$ is differentiable on $\mathcal{D}$ and $\nabla L_\epsilon f(x)$ converges to $\nabla \mathscr{L} f(x)$ uniformly.*

Proof. We first choose a countable set of ϵ, say, $\epsilon = \{1/n\}_{i=1}^{\infty}$, and we use L_n to denote $L_{\epsilon=1/n}$ for convenience. Since $L_n f(x)$ are uniformly bounded and $\nabla L_n f(x)$ are uniformly equicontinuous, $\nabla L_n f(x)$ are uniformly bounded. Then $\nabla L_n f(x)$ has a subsequence $\nabla L_{n(k)} f(x)$ that converges uniformly to some function $g \in C(\mathcal{D})$ by Arzela-Ascoli theorem. We then show that $g = \nabla \mathscr{L} f(x)$ by contradiction. Suppose $\nabla L_n f(x)$ does not converge uniformly to $\nabla \mathscr{L} f(x)$. Then there exists $\epsilon > 0$ and another subsequence $\nabla L_{m(k)} f(x)$ of $\nabla L_n f(x)$ such that $\|\nabla L_{m(k)} f - \nabla \mathscr{L} f\|_\infty > \epsilon$ for all k. But by Arzela-Ascoli theorem, $\nabla L_{m(k)}$ has a subsequence converging uniformly to $\nabla \mathscr{L} f(x)$: contradiction. Therefore, $\nabla L_n f(x)$ converges to $\nabla \mathscr{L} f(x)$ uniformly on $\mathcal{D}$. □

F. Numerical results

F.1. Measuring the error using OT distance

We compute the OT distance using the Sinkhorn–Knopp algorithm (Cuturi, 2013; Knight, 2008). The cost matrix is set to be the pairwise distance between the reference samples and generated particles, and each sample is assigned equal weight marginally. The number of reference samples is chosen to be large to mitigate error in the OT distance resulting from discretization of the target: 20000 for the Mickey mouse, two moons, the arc problems, the high energy physics example, and 50000 for the hyper-semisphere example due to its higher dimension. The entropic regularization penalty $1/\lambda$ is set to be $O(10^{-2})$ in Mickey mouse, two moons, the arc problems, and hyper-semisphere example, and $O(10^{-3})$ in the high energy physics example. Each experiment is repeated 10 times for reproducibility; this replication involves sampling new training data and repeating all steps of each algorithm. For the the Mickey mouse, two moons, the arc problems, the number of generated particles is varied over $\{100, 300, 900, 2700\}$; for the hyper-semisphere example, it is fixed to 300.

F.2. The arc: one-dimensional manifold embedded in a three-dimensional space

Figure 1 shows the initial particles and the target distribution, and the comparison between different methods. Figure 3 shows the error comparison between different methods.

G. Additional numerical results

G.1. Mickey mouse: two-dimensional connected domain

In this example, the target distribution is uniform over a compactly supported Mickey mouse-shaped domain. The generative process is initiated uniformly inside a circle. Results are obtained with both 1000 and 2000 training samples. In Figure 4, we show the initial particles, the generated particles and the target distribution. Both methods capture the shape relatively well. However, particles generated from SVGD move out of the domain, while most of the particles generated using DMPS stay inside. In some cases, the SVGD-generated particles exhibit a non-uniform pattern; see Figure 8. Figure 9 shows quantitative comparisons of the error.

G.2. Two moons: two-dimensional disconnected domain

In this example, the target distribution is uniform and compactly supported on a two-moon-shaped domain. In contrast with the previous example, the domain is disconnected. Though the underlying distribution has zero density outside the support, the finite kernel bandwidth enables the methods to be implementable in this case. Results are obtained with 500 and 1000 training samples. We show the initial particles, target distribution, and particles generated with DMPS, SVGD, and ULA in Figure 6 and the regularized OT distance in Figure 7. As we can see in Figure 6, SVGD does not explore the very end of the domain and ULA has mny samples that diffuse out of the support. The error plot (Figure 7) shows that DMPS enjoys the smallest error in terms of OT distance, and that this the error decreases with more generated particles. While ULA shows a similar convergence (with larger values of error), the error of SVGD fluctuates as more particles are included.

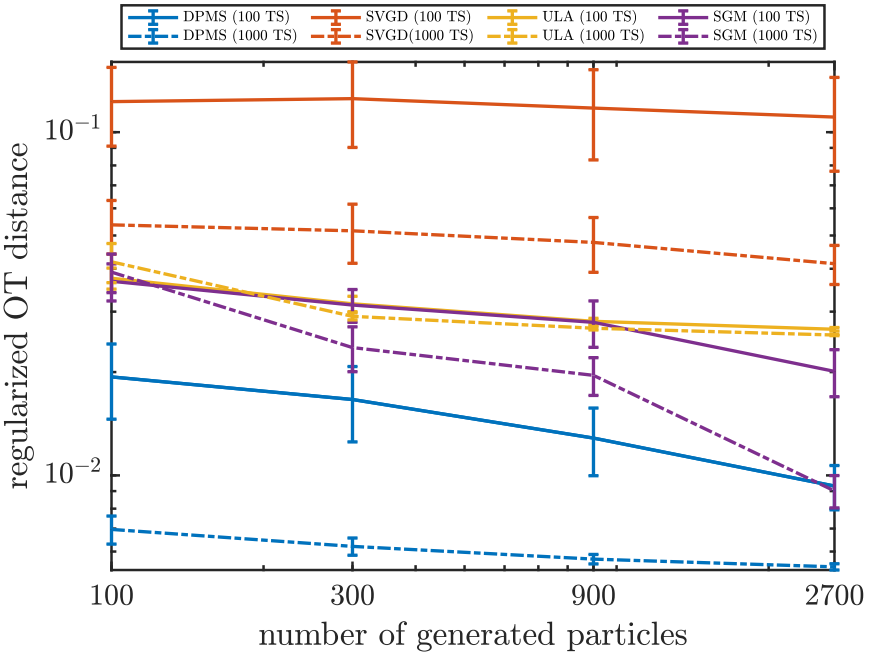


Figure 3: The arc: error comparison between DMPS, SVGD, ULA, and SGM. Solid lines use 100 training samples; dashed lines use 1000.

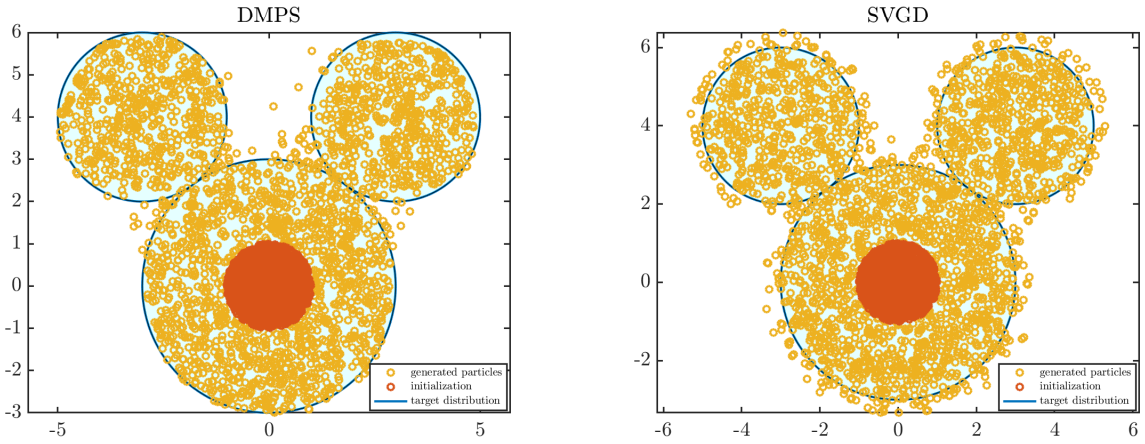


Figure 4: Mickey mouse: 2700 generated particles using DPMS and SVGD, with 2000 training samples

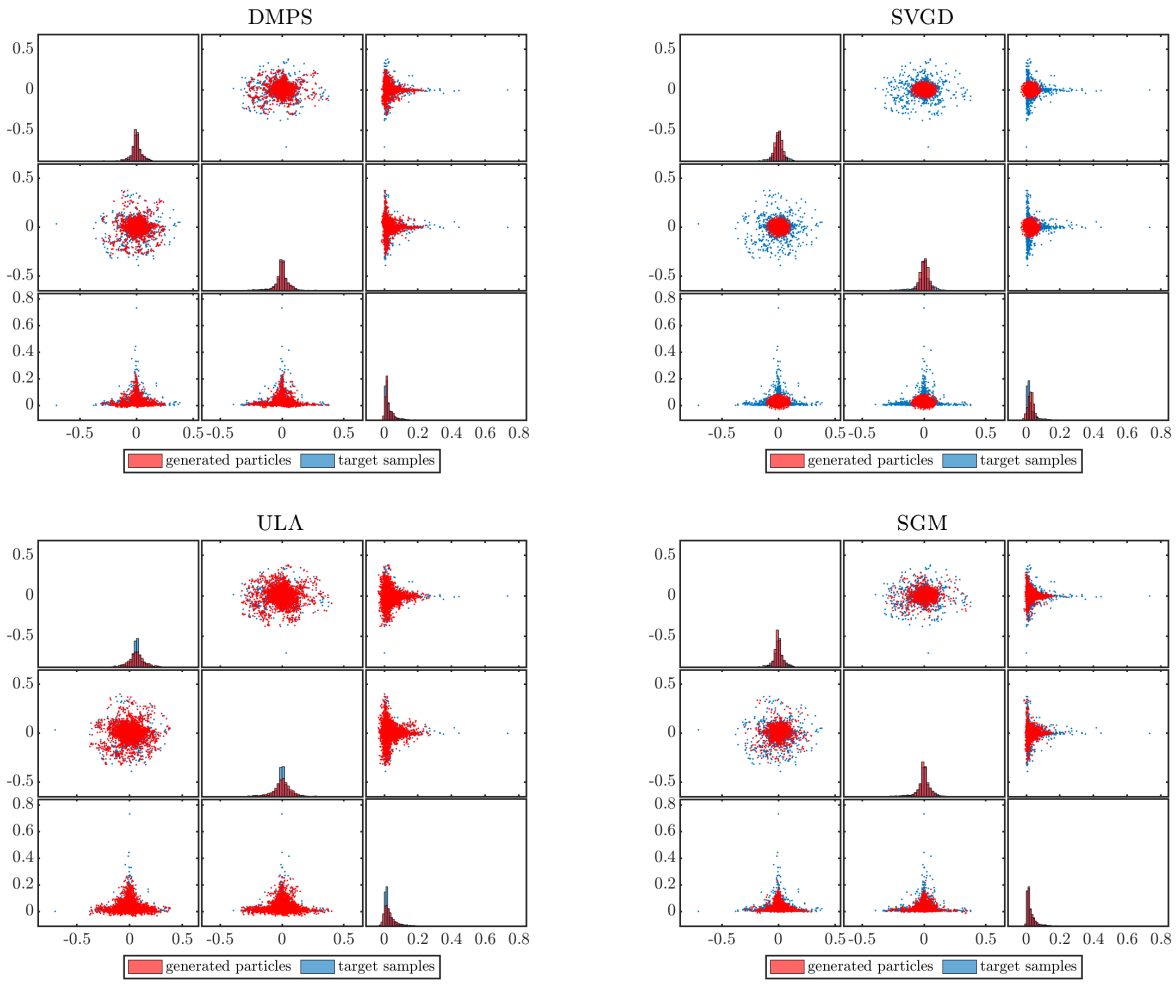


Figure 5: Distributions of the target samples and 2700 generated particles using DMPS, SVGD, ULA, and SGM in the gluon jet example. Coordinates are η^{rel}, ϕ^{rel}, and p_T^{rel}, respectively.

G.3. High energy physics: gluon jet dataset

We present the distribution of the generated particles using different methods. For this comparison, the number of training samples is 1000 and the number of generated particles is 2700. In each figure, we plot 2000 target samples as a reference. As we can see in Figure 5, particles generated using DMPS, ULA, and SGM resemble those from the target distribution. Nevertheless, particles generated using SVGD fail to capture the target distribution.

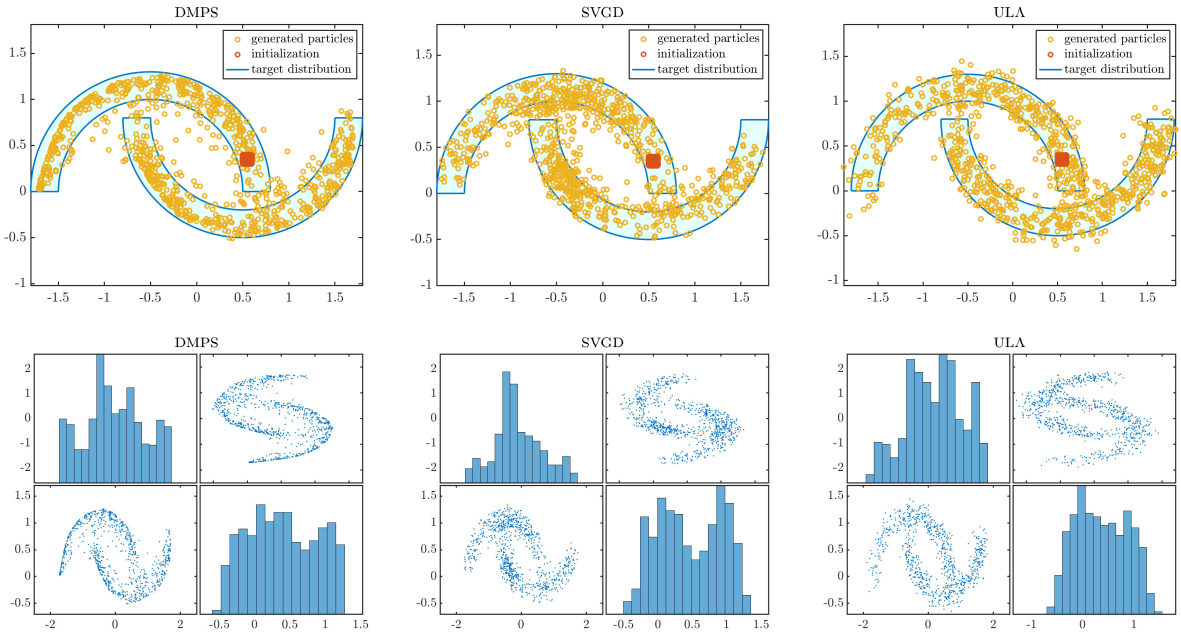


Figure 6: Two moons: 900 generated particles from DMPS, SVGD, and ULA with 500 training samples

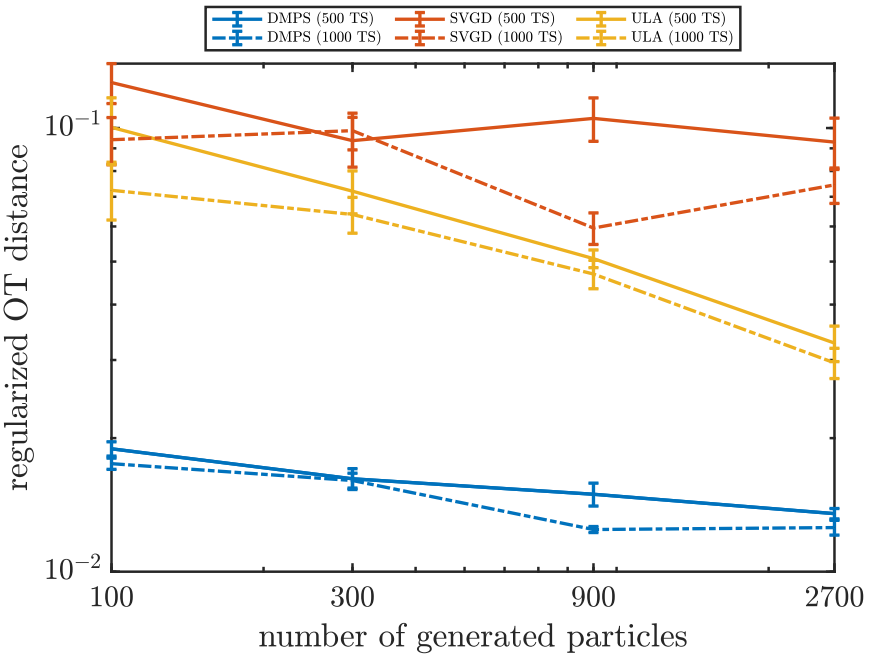


Figure 7: Two moons: error comparison between DMPS, SVGD, and ULA. Solid lines use 500 training samples, dashed lines use 1000.

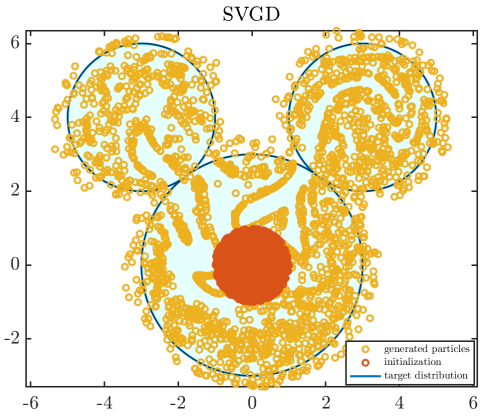


Figure 8: Mickey mouse: an instance of running the SVGD generative model shows strange non-uniform pattern with 1000 training samples and 2700 generated particles

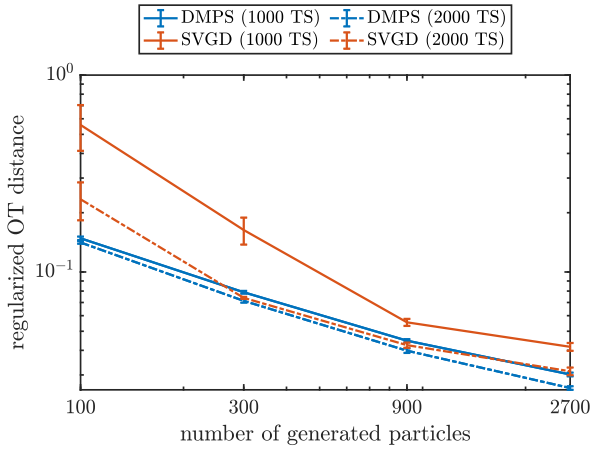


Figure 9: Mickey mouse: error comparison between DMPS, SVGD

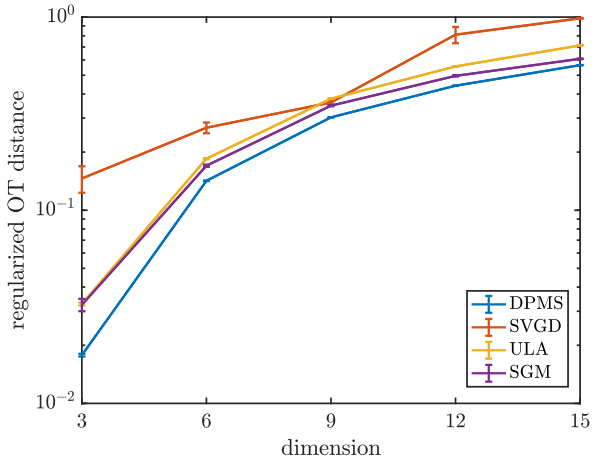


Figure 10: Hyper-semisphere: error comparison between DMPS, SVGD, ULA and SGM