Momentum SVGD-EM for Accelerated Maximum Marginal Likelihood Estimation

Adam Rozzio[⋆,1] **Rafael Athanasiades**[⋆,2] **O. Deniz Akyildiz**[2]
[1]Department of Mathematics, Ecole Normale Supérieure Paris-Saclay, France
[2]Department of Mathematics, Imperial College London, UK

Abstract

Maximum marginal likelihood estimation (MMLE) can be formulated as the optimization of a free energy functional. From this viewpoint, the Expectation–Maximisation (EM) algorithm admits a natural interpretation as a coordinate descent method over the joint space of model parameters and probability measures. Recently, a significant body of work has adopted this perspective, leading to interacting particle algorithms for MMLE. In this paper, we propose an accelerated version of one such procedure, based on Stein variational gradient descent (SVGD), by introducing Nesterov acceleration in both the parameter updates and in the space of probability measures. The resulting method, termed Momentum SVGD-EM, consistently accelerates convergence in terms of required iterations across various tasks of increasing difficulty, demonstrating effectiveness in both low- and high-dimensional settings.

1 INTRODUCTION

Latent variable models (LVMs) are ubiquitous in machine learning and statistics, as they provide a powerful framework for modeling complex data by capturing the underlying latent structure (Whiteley et al., 2025). A typical LVM is defined by a joint probability density $p_\theta(x, y) : \mathcal{X} \times \Theta \to \mathbb{R}$, where y is the (fixed) observed data, x is the unobserved latent variable, and $\theta \in \Theta$ is the model's parameter. Given some data y, the typical goal is to find the parameter that maximizes the marginal likelihood $p_\theta(y)$, which is the probability of observing the data under the model - termed the maximum marginal likelihood estimation (MMLE). This results in the following optimization problem:

$$\theta_\star \in \arg\max_{\theta\in\Theta} \log p_\theta(y), \tag{1}$$

where $p_\theta(y) := \int_{\mathcal{X}} p_\theta(x, y)\mathrm{d}x$. The standard approach for performing MMLE is to use the celebrated Expectation-Maximisation (EM) algorithm (Dempster et al., 1977). At iteration t, the EM algorithm alternates between two steps: the E-step, which computes the expectation $Q(\theta, \theta_t) := \mathbb{E}_{p_{\theta_t}(x|y)}[\log p_\theta(x, y)]$, and the M-step, which updates the model parameters by maximising the expected log-likelihood, i.e. $\theta_{t+1} \in \arg\max_{\theta\in\Theta} Q(\theta, \theta_t)$. The EM algorithm is typically intractable (no closed form) to implement, resulting in approximations for its E and M steps, see, e.g. Wei and Tanner (1990), Meng and Rubin (1993), Liu and Rubin (1994), and Lange (1995). In recent years, diffusion based approaches gained traction, in particular, algorithms based on unadjusted Langevin algorithm (ULA) have been proposed, see, e.g., De Bortoli et al. (2021). These approaches are computationally expensive, as they require Markov chain Monte Carlo (MCMC) chains for the E-step, which can be slow to converge and hard to understand theoretically.

An emerging computational paradigm for MMLE is to use the coordinate descent perspective of EM to recast the problem as a minimisation of the so-called free energy (Neal and Hinton, 1998). In particular, Neal and Hinton (1998) showed that the EM algorithm can be interpreted as a coordinate descent scheme on the free energy functional:

$$\mathcal{F}(\theta, q) := \int_{\mathcal{X}} q(x) \log q(x)\mathrm{d}x - \int_{\mathcal{X}} q(x) \log p_\theta(x, y)\mathrm{d}x. \tag{2}$$

Based on this perspective, Kuntz et al. (2023) proposed Particle Gradient Descent (PGD), a particle-based algorithm for MMLE that uses the free energy as an objective function. The PGD runs an optimiser for θ-updates and a particle system for q-updates. This

⋆ Equal contribution.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

is a significant departure from the classical algorithms for EM as the particle-based approach allows for simultaneous updates of the latent variables and the model parameters. This approach was justified theoretically (Caprio et al., 2025) and led to a number of follow-up works, see, e.g., Lim et al. (2024), Sharrock et al. (2024), Akyildiz et al. (2025), Oliva and Akyildiz (2024), and Encinar et al. (2025).

Among the works that followed Kuntz et al. (2023), there are two particular works that are of special interest to us in this work. In the first work, Sharrock et al. (2024) proposed the Stein variational gradient descent-expectation maximisation (SVGD-EM) algorithm, which uses the Stein variational gradient descent (SVGD) (Liu and Wang, 2016) to evolve the particles for the latent variables. This contrasts with the approach followed in Kuntz et al. (2023), where the particles are evolved using (independent) Langevin dynamics. The second work is Lim et al. (2024), which proposed a momentum-based approach to PGD called Momentum Particle Gradient Descent (MPGD). The MPGD algorithm uses an alternative (enriched) free energy to develop a momentum-based algorithm for MMLE. Lim et al. (2024) showed that the MPGD algorithm converges faster than the PGD method on a variety of settings, demonstrating the acceleration provided by the momentum-based approach. In this work, we combine these two approaches. In particular, our contributions are summarised as follows.

Contributions. In this work, we provide an accelerated version of the SVGD-EM algorithm, which we term Momentum SVGD-EM (M-SVGD-EM). Our approach integrates two distinct Nesterov-inspired acceleration schemes: one applied to the sampling phase using the Wasserstein-Nesterov Stein variational gradient descent (SVGD-WNes) as introduced in Liu et al. (2019), and the other applied to the parameter updates following classical momentum techniques (Nesterov, 1983). The resulting method bears similarities to MPGD and can be seen as an analogue of MPGD for the SVGD-EM algorithm. We derive the M-SVGD-EM algorithm that combines the SVGD-WNes with Nesterov momentum for the parameter updates, which is achieved by leveraging the free energy perspective of MMLE and the connection between SVGD-EM and Wasserstein gradient flows. To demonstrate the effectiveness of our approach, we conduct numerical experiments on a variety of tasks, including a Toy Hierarchical Model, a Bayesian Logistic Regression task on the *Wisconsin Breast Cancer Dataset*, and a Bayesian Neural Network model applied to the MNIST dataset. Our results show that M-SVGD-EM consistently outperforms SVGD-EM in terms of convergence speed across all tasks, hence providing a fast and efficient method for MMLE.

The paper is organized as follows. In Section 2, we recall our problem setup and the SVGD-EM algorithm briefly. In Section 3, we introduce our method, M-SVGD-EM. In Section 4, we provide a thorough demonstration of acceleration properties of M-SVGD-EM, showing that the new method consistently accelerates the SVGD-EM. Finally, we conclude with Section 5.

Notation. We denote the latent space, the observation space, and the parameter space with $\mathcal{X} := \mathbb{R}^{d_x}$, $\mathcal{Y} := \mathbb{R}^{d_y}$, and $\Theta := \mathbb{R}^{d_\theta}$ respectively. For any finite measure q on $\mathcal{X}$ we note $T_{\#}q$ the pushforward measure of q by the measurable map $T : \mathcal{X} \to \mathcal{X}$. The map denoted by $\mathrm{id} : \mathcal{X} \to \mathcal{X}$ refers to the identity map. We note $\mathcal{P}_{2,ac}(\mathcal{X})$ the set of absolutely continuous measures w.r.t the Lebesgue measure with finite variance and for $q_1, q_2 \in \mathcal{P}_{2,ac}(\mathcal{X})$, the Wasserstein-2 distance (Ambrosio et al., 2008, Chapter 7.1) between q_1 and q_2 is defined by $W_2^2(q_1, q_2) = \inf_{\gamma \in \Gamma(q_1,q_2)} \int_{\mathcal{X}\times\mathcal{X}} \|x_1 - x_2\|^2 \gamma(\mathrm{d}x_1, \mathrm{d}x_2)$ where $\Gamma(q_1, q_2)$ is the set of couplings between q_1 and q_2. We also note $\mathcal{T}_{q_1}\mathcal{P}_{2,ac}(\mathcal{X})$ the tangent space at q_1 and we refer to the metric space $(\mathcal{P}_{2,ac}(\mathcal{X}), W_2)$ as the Wasserstein space. Let $\mathcal{F} : \Theta \times \mathcal{P}_{2,ac}(\mathcal{X}) \mapsto \mathbb{R} \cup \{\infty\}$, we note $\frac{\delta\mathcal{F}}{\delta q}(\theta, q) : \mathcal{X} \to \mathbb{R}$ the first variation of $\mathcal{F}$ w.r.t q defined (Santambrogio, 2015, Definition 7.12) as the unique function (up to an additive constant) such that $\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{F}(\theta, q + tm)|_{t=0} = \int_{\mathcal{X}} \frac{\delta\mathcal{F}}{\delta q}(\theta, q)(x)\mathrm{d}m(x)$ for any $m : \mathcal{X} \to \mathbb{R}$ satisfying $\int_{\mathcal{X}} m(x)\mathrm{d}x = 0$. The only Reproducing Kernel Hilbert Space (RKHS) associated with the real positive definite kernel $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is noted $\mathcal{H}$ and we note $\nabla_1 k$ the gradient of $k(\cdot, \cdot)$ with respect to its first component. In the following, vector-valued RKHS will be spaces of the form $\mathcal{H}^d := \mathcal{H} \times \cdots \times \mathcal{H}$.

2 BACKGROUND

Let $p_\theta(x, y)$ be a joint probability distribution differentiable in θ and x of an LVM where $y \in \mathcal{Y}$ is a fixed observation (dataset), $x \in \mathcal{X}$ denotes latent variables, $\theta \in \Theta$ is the parameter. In this paper, we are interested in fitting θ given the observed data y by solving the problem

$$\theta_\star \in \arg\max_{\theta\in\Theta} \log p_\theta(y), \tag{3}$$

where $p_\theta(y) := \int_{\mathcal{X}} p_\theta(x, y)\mathrm{d}x$. We note that, given that y is fixed, $p_\theta(y)$ is only a function of θ, i.e., $p_\theta(y) : \Theta \to \mathbb{R}$. The problem in (3) corresponds to the maximum marginal likelihood estimation (MMLE) problem as introduced above, by maximizing the probability of the data under the given model, after integrating out latent variables.

To provide the background for our method for solving the MMLE, we first introduce the EM algorithm below

from a variational perspective, then move on to describe the SVGD-EM. For brevity, let us define $\ell(\theta, x) = \log p_\theta(x, y)$ as the joint log-likelihood function where y is hidden from the notation $\ell(\theta, x)$ throughout as it is fixed.

2.1 A variational view of EM

The EM algorithm is the de facto recipe that is applied for fitting LVMs (Dempster et al., 1977). The EM is an iterative scheme that consists of two steps at iteration t (i) E-step is the task of computing the expectation $Q(\theta, \theta_t) = \mathbb{E}_{p_{\theta_t}(x|y)}[\log p_\theta(x, y)]$ and (ii) M-step is the task of computing the maximizer of $Q(\theta, \theta_t)$ w.r.t. θ (leading to the next iterate θ_{t+1}). Neal and Hinton (1998) showed that the EM algorithm can also be seen as a coordinate descent method on free energy $\mathcal{F} : \Theta \times \mathcal{P}(\mathcal{X}) \to \mathbb{R} \cup \{\infty\}$ defined for $(\theta, q) \in \Theta \times \mathcal{P}(\mathcal{X})$ by

$$\mathcal{F}(\theta, q) := \int_{\mathcal{X}} \log q(x) q(x) \mathrm{d}x - \int_{\mathcal{X}} \ell(\theta, x) q(x) \mathrm{d}x.$$

Using this free energy, the steps of EM can be reformulated as

- E-step : $q_t := \arg\min_{q \in \mathcal{P}(\mathcal{X})} \mathcal{F}(\theta_t, q)$
- M-step : $\theta_{t+1} := \arg\min_{\theta \in \Theta} \mathcal{F}(\theta, q_t)$

An important implication here is that for any $\theta \in \Theta$ the posterior $p_\theta(\cdot|y)$ minimizes $q \mapsto \mathcal{F}(\theta, q)$ (Kuntz et al., 2023). Moreover the marginal likelihood $p_\theta(y)$ has a global maximum at θ if and only if $\mathcal{F}(\theta, q)$ has a global maximum at $(\theta_\star, p_{\theta_\star}(\cdot|y))$. Thus, the idea is now to minimize $\mathcal{F}$ and this will be done in the Wasserstein space which gives a well studied notion of gradient (Otto, 2001; Ambrosio et al., 2008; Villani et al., 2009).

2.2 Euclidean-Wasserstein gradient flow of $\mathcal{F}$ on $\Theta \times \mathcal{P}_{2,ac}(\mathcal{X})$

In order to minimize the free energy $\mathcal{F}$, a viable approach is to formalize its gradient flow on $\Theta \times \mathcal{P}_{2,ac}(\mathcal{X})$ where Θ is equipped with the standard Euclidean metric and $\mathcal{P}_{2,ac}(\mathcal{X})$ with the Wasserstein-2 metric. In this scenario, as presented in Kuntz et al. (2023, Appendix B.1) we have $\nabla\mathcal{F}(\theta, q) = (\nabla_\theta \mathcal{F}(\theta, q), \nabla_q \mathcal{F}(\theta, q))$ with

$$\nabla_\theta \mathcal{F}(\theta, q) = -\int_{\mathcal{X}} \nabla_\theta \ell(\theta, x)\, q(x)\, \mathrm{d}x$$
$$\nabla_q \mathcal{F}(\theta, q) = -\mathrm{div}\big(q \mathbb{W}\mathcal{F}(\theta, q)\big).$$

The term $\mathbb{W}\mathcal{F}(\theta, q) : \mathcal{X} \to \mathcal{X}$ is the velocity vector field commonly called the Wasserstein gradient of $\mathcal{F}$. It is defined by $\mathbb{W}\mathcal{F}(\theta, q) := \nabla_x(\frac{\delta \mathcal{F}}{\delta q}(\theta, q))$, using Lemma 1 of Appendix A.3 in Kuntz et al. (2023) leads to the equality $\mathbb{W}\mathcal{F}(\theta, q) = \nabla_x \log(q/p_\theta(\cdot, y))$ and thus the *Euclidean-Wasserstein* gradient flow of $\mathcal{F}$ can be written as

$$\partial_t \theta_t = \int_{\mathcal{X}} \nabla_\theta \ell(\theta_t, x) q_t(x) \mathrm{d}x \tag{4}$$
$$\partial_t q_t = \mathrm{div}\left(q_t \nabla_x \log\left(\frac{q_t}{p_{\theta_t}(\cdot, y)}\right)\right). \tag{5}$$

Kuntz et al. (2023) develop a method based on the flow (4)–(5) to perform MMLE. The method is called PGD and is based on discretizing a stochastic differential equation based on the flow (4)–(5) using the Euler scheme with a time step $\gamma > 0$ and approximating $q_t \approx (1/N)\sum_{i=1}^N \delta_{x_t^{(i)}}$ where $\{x_t^{(i)}\}_{0 \le i \le N}$ are the particles at time t. This leads to a particle-based algorithm for MMLE (Kuntz et al., 2023), where particles are evolved with respect to a (time-discretized) SDE solving (4), independently.

Let us consider the flow (4)–(5) discretized by a semi-implicit Euler scheme with a time step $\gamma > 0$ and obtain the following update rules:

$$\theta_{t+1} = \theta_t + \gamma \int_{\mathcal{X}} \nabla_\theta \ell(\theta_t, x)\, q_t(x)\, \mathrm{d}x \tag{6}$$
$$q_{t+1} = \Big(\mathrm{id} - \gamma \nabla_x \log\Big(\frac{q_t}{p_{\theta_{t+1}}(\cdot, y)}\Big)\Big)_{\#} q_t. \tag{7}$$

Now we can make the important observation that after sampling q_t with the particles $\{x_t^{(i)}\}_{0 \le i \le N}$, (7) can be seen as the update $x_{t+1}^{(i)} = x_t^{(i)} - \gamma \mathbb{W}\mathcal{F}(\theta_{t+1}, q_t)(x_t^{(i)})$ which means the particles are pushed according to $-\mathbb{W}\mathcal{F}(\theta_{t+1}, q_t)$. This reveals the role of $-\mathbb{W}\mathcal{F}(\theta, q)(\cdot)$ as being the velocity vector fields driving the particles of a density following the Wasserstein Gradient Flow of $\mathcal{F}$ (with a first order approximation when we consider (7) instead of (5) – see Ambrosio et al. (2008, Proposition 8.4.6)).

2.3 SVGD-EM

The SVGD-EM (Sharrock et al., 2024) algorithm relies on a closely relevant but different flow compared to (4)–(5). In particular, the parameter component (4) stays the same but the density q_t in (5) is driven by a different velocity vector field referred to as a *Wasserstein gradient in the RKHS* $\mathcal{H}_k$. This is expressed as $\mathbb{W}_{H_k}\mathcal{F}(\theta, q) : \mathcal{X} \to \mathcal{X}$, defined by $\mathbb{W}_{H_k}\mathcal{F}(\theta, q)(\cdot) := -\int_{\mathcal{X}} [\nabla_x \ell(\theta, x)\, k(x, \cdot) + \nabla_1 k(x, \cdot)]\, q(x)\, \mathrm{d}x$ and which can be seen as an image of the standard Wasserstein gradient through an integral operator (Sharrock et al., 2024). It is the same velocity vector field that drives the particles in the SVGD method, hence the name of SVGD-EM. The resulting semi-implicit time-

discretized flow is then given by

$$\theta_{t+1} = \theta_t + \gamma \int_{\mathcal{X}} \nabla_\theta \ell(\theta_t, x) q_t(x) \mathrm{d}x$$
$$q_{t+1} = \Big(\mathrm{id} + \gamma \int_{\mathcal{X}} \big[k(x, \cdot) \nabla_x \ell(\theta_{t+1}, x) + \nabla_1 k(x, \cdot) \big] q_t(x) \, \mathrm{d}x \Big)_{\#} q_t.$$

An implementable scheme can be obtained by using the particle approximation $q_t \approx (1/N) \sum_{i=1}^N \delta_{x_t^{(i)}}$. The resulting method takes the form of update rules:

$$\theta_{t+1} = \theta_t + \frac{\gamma}{N} \sum_{j=1}^N \nabla_\theta \ell(\theta_t, x_t^j) \tag{8}$$

$$x_{t+1}^i = x_t^i + \frac{\gamma}{N} \sum_{j=1}^N [k(x_t^j, x_t^i) \nabla_x \ell(\theta_{t+1}, x_t^j) + \nabla_{x_t^j} k(x_t^j, x_t^i)]. \tag{9}$$

This algorithm has a few important differences with respect to PGD. First, the algorithm is deterministic (compared to stochastic latent variable updates of PGD), second, the x-particles are interacting through the kernel k (non-local).

3 MOMENTUM SVGD-EM

In this section, we introduce our method, M-SVGD-EM, which is an accelerated version of the SVGD-EM algorithm. This method accelerates both parameter and latent variable updates and in the following sections we derive the accelerated updates first on the parameter space Θ and then on the space of measures $\mathcal{P}_{2,ac}(\mathcal{X})$.

3.1 Acceleration in Θ

We first introduce the (easier) acceleration scheme in the parameter space Θ. This is taken from the standard optimization literature of accelerated gradient descent, see, e.g. Nesterov (1983), Nesterov (1988), and Nesterov (2005). The motivation behind this choice is that the accelerated gradient descent can achieve $\mathcal{O}(1/t^2)$ convergence rate for smooth convex functions compared to gradient descent which only attains $\mathcal{O}(1/t)$, this acceleration is known to be a trade-off with stability as shown by Attia and Koren (2021) but this was carefully monitored in our experiments. We adopt this strategy to replace the θ-update in (8). Recall that the standard gradient update for a loss function f, $\theta_{t+1} = \theta_t - \gamma \nabla_\theta f(\theta_t)$, can be accelerated by introducing a momentum term $\tilde{\theta}_t$ in the original scheme $\theta_{t+1} = \tilde{\theta}_t - \gamma \nabla_\theta f(\tilde{\theta}_t)$ where $\tilde{\theta}_t$ is updated by $\tilde{\theta}_{t+1} = \theta_{t+1} + \alpha_\theta(\theta_{t+1} - \theta_t)$ where α_θ is a momentum parameter (Nesterov, 2013).Applying this to our case, given a set of particles $\{x_t^{(i)}\}_{0 \le i \le N}$ at time t with the parameter update θ_t, we can replace the update (8) with the following update:

$$\theta_{t+1} = \tilde{\theta}_t + \frac{\gamma}{N} \sum_{i=1}^N \nabla_\theta \ell(\tilde{\theta}_t, x_t^{(i)}), \tag{10}$$

$$\tilde{\theta}_{t+1} = \theta_{t+1} + \alpha_\theta(\theta_{t+1} - \theta_t). \tag{11}$$

This is the standard Nesterov scheme, applied to the parameter update of SVGD-EM. Equipped with (10)–(11), we now turn to the task of accelerating SVGD-EM in the space of probability measures.

3.2 Acceleration in $\mathcal{P}_{2,ac}(\mathcal{X})$

To accelerate the latent variable update in (9) on SVGD, we adopt the acceleration scheme SVGD-WNes proposed by Liu et al. (2019) which uses a Nesterov scheme on $\mathcal{P}_{2,ac}(\mathcal{X})$ to accelerate the interacting particle sampling algorithm SVGD and builds upon the Riemannian version of Nesterov's Accelerated Gradient Descent (RAGD) method of Zhang and Sra (2018). The idea of RAGD is similar as in the Euclidean case with accelerated gradient descent, but with the key difference that, instead of linearly combining vectors, each iterate update is performed using exponential maps. The exponential map at q is defined by $\mathrm{Exp}_q(v) := (\mathrm{id} + v)_{\#} q$. This map assigns to a vector tangent in q noted $v \in \mathcal{T}_q \mathcal{P}_{2,ac}(\mathcal{X})$, an associated measure $(\mathrm{id} + v)_{\#} q$ by moving along the geodesic with velocity v in q for a unit of time (Ambrosio et al., 2008, Prop. 8.4.6). This velocity vector field $v(\cdot)$ serves a role analogous to the standard gradient in the accelerated gradient descent. Using the exponential map, applied to the SVGD dynamics driven by $\mathbb{W}_{H_k} \mathcal{F}(\theta, \cdot)$ with fixed θ, the RAGD method proposed by Zhang and Sra (2018) can be written as:

$$q_{t+1} = \mathrm{Exp}_{\tilde{q}_t}(-\gamma \mathbb{W}_{H_k} \mathcal{F}(\theta, \tilde{q}_t)) \tag{12}$$

$$\tilde{q}_{t+1} = \mathrm{Exp}_{q_{t+1}}\{c_1 \mathrm{Exp}_{q_{t+1}}^{-1}[\mathrm{Exp}_{\tilde{q}_t}(\mathrm{Exp}_{\tilde{q}_t}^{-1}(q_{t+1}) + (c_2 - 1)(\mathrm{Exp}_{\tilde{q}_t}^{-1}(q_{t+1}) - \mathrm{Exp}_{\tilde{q}_t}^{-1}(q_t)))]\}, \tag{13}$$

where $c_1, c_2 \in \mathbb{R}^+$ are constants. The eq. (12) is analogous to the first step of the accelerated gradient descent and the eq. (13) is a momentum update as in the accelerated gradient descent. The main bottleneck of implementing (12)–(13) is the computation of the exponential map and its inverse (Pele and Werman, 2009). To alleviate this issue and manage to have a time complexity inferior to the $O(N^2)$ of the Sinkhorn method (Cuturi, 2013; Xie et al., 2020), Liu et al. (2019) introduce an approximation to this exponential map inverse and obtain the algorithm termed Wasserstein-Nesterov Stein variational gradient descent (SVGD-WNes). Given two empirical measures $q_t \approx \sum_i^N \delta_{x_t^{(i)}}$

and $\tilde{q}_t \approx \sum_i^N \delta_{\tilde{x}_t^{(i)}}$, to provide a computable method, Liu et al. (2019) assumes a closeness relationship between particles and momentum particles and between particles of successive iterations leading to the approximation

$$\left(\mathrm{Exp}_{q_t}^{-1}(\tilde{q}_t)\right)(x_t^{(i)}) \approx \tilde{x}_t^{(i)} - x_t^{(i)}, \quad 1 \leq i \leq N. \quad (14)$$

Using the approximation (14) results in the SVGD-WNes method which takes the form of update rules:

$$x_{t+1}^{(i)} = \tilde{x}_t^{(i)} + \frac{\gamma}{N}\sum_{j=1}^{N} k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)})\nabla_x \ell(\theta, \tilde{x}_t^{(j)}) \quad (15)$$
$$+ \nabla_1 k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)})$$
$$\tilde{x}_{t+1}^{(i)} = x_{t+1}^{(i)} + c_1(c_2 - 1)(x_{t+1}^{(i)} - x_t^{(i)}) \quad (16)$$

where $c_1, c_2 \in \mathbb{R}^+$ are constants. We provide a self-contained derivation of eqs. (15)–(16) from (12)–(13) in Appendix A.

3.3 Momentum SVGD-EM

Having derived acceleration schemes for both the parameter and latent variable updates, we can now combine them to obtain our final method. The resulting algorithm is a combination of the two schemes described in Sections 3.1 and 3.2. The resulting method is called M-SVGD-EM and is summarized in Algorithm 1. The main difference with respect to SVGD-EM is that we replace the update (8) with the accelerated update (10)–(11) and the update (9) with the accelerated update (15)–(16).

Algorithm 1 M-SVGD-EM

Input: momentum constants $\alpha_X, \alpha_\theta \in \mathbb{R}$, initial particles $\{x_0^{(i)}\}_{i=1}^N \sim q_0$, initial parameter θ_0, joint log-likelihood ℓ, kernel k, learning rate γ.
Initialize: $\theta_0 = \tilde{\theta}_0$ and $\{x_0^{(i)}\}_{i=1}^N = \{\tilde{x}_0^{(i)}\}_{i=1}^N$
for $t = 0, 1, \ldots, T-1$ **do**
  $\theta_{t+1} = \tilde{\theta}_t + \frac{\gamma}{N}\sum_{j=1}^N \nabla_\theta \ell(\tilde{\theta}_t, x_t^{(j)})$
  $\tilde{\theta}_{t+1} = \theta_{t+1} + \alpha_\theta(\theta_{t+1} - \theta_t)$
  for $i = 1, 2, \ldots, N$ **do**
    $x_{t+1}^{(i)} = \tilde{x}_t^{(i)} + \frac{\gamma}{N}\sum_{j=1}^N k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)})\nabla_x \ell(\theta_{t+1}, \tilde{x}_t^{(j)})$
      $+\nabla_1 k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)})$
    $\tilde{x}_{t+1}^{(i)} = x_{t+1}^{(i)} + \alpha_X(x_{t+1}^{(i)} - x_t^{(i)})$
  end for
end for
Output: θ_T and $\{x_T^i\}_{i=1}^N$

4 NUMERICAL EXPERIMENTS

We now compare the performance of M-SVGD-EM with a number of other algorithms, including PGD (Kuntz et al., 2023), MPGD (Lim et al., 2024), Stochastic Optimization via Unadjusted Langevin algorithm (SOUL) (De Bortoli et al., 2021), and SVGD-EM (Sharrock et al., 2024). The code used to run the following experiments is publicly available[1]. We benchmark these methods on a number of examples, specifically, on a toy hierarchical model, a Bayesian logistic regression model, and a Bayesian neural network. The goal of these experiments is to better understand the acceleration properties of M-SVGD-EM compared to non-accelerated variants. In the original implementation of SVGD-EM (Sharrock et al., 2024), the kernel of choice is Median RBF kernel $k(x, x') = \exp(-h^{-1}||x - x'||^2)$ with the median heuristic introduced by Liu and Wang (2016). In the results that follow, we have used AutoRBF provided in the codebase of the original paper (Sharrock et al., 2024). The AutoRBF kernel is defined as $k(x, x') = \exp\left(-2h^{-2}\|x - x'\|^2\right)$ where we use JAX (Bradbury et al., 2018) to handle gradient calculations. Here, $h > 0$ is the bandwidth parameter that was chosen to be $h = 5.0$ in our experiments. We selected this kernel over MedianRBF as it demonstrated better performance and more pronounced acceleration effects in our empirical tests. Results using MedianRBF can be found in the Appendix B. All experiments were run on a customer grade laptop with CPU 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz 32 GB RAM.

4.1 Toy Hierarchical Model

We first evaluate the algorithms on a toy hierarchical model. We replicate the experimental settings of Kuntz et al. (2023) and Sharrock et al. (2024). The model under consideration can be described as

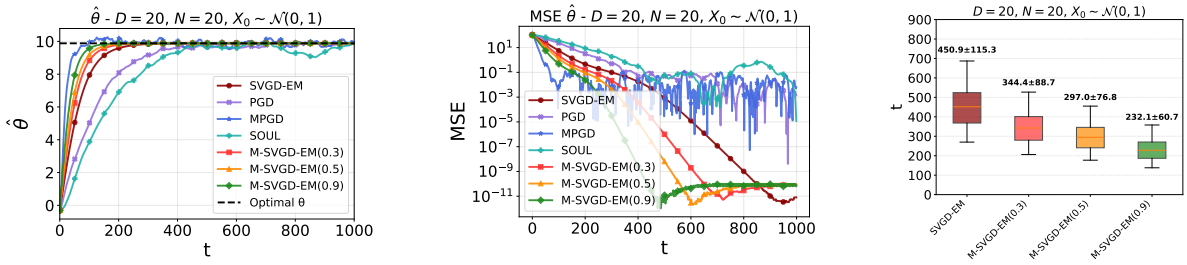
$$p_\theta(x, y) = \prod_{i=1}^{d_y} \mathcal{N}(y_i; x_i, 1)\,\mathcal{N}(x_i; \theta, \sigma^2).$$

The model assumes that we observe the data $(y_1, \ldots, y_{d_y}) \in \mathbb{R}^{d_y}$ generated via first sampling the latent variables $x_i \sim \mathcal{N}(x_i; \theta, \sigma^2)$ and then $\mathcal{N}(y_i; x_i, 1)$. In this model the marginal likelihood $\theta \mapsto p_\theta(y)$ has a unique maximizer $\hat{\theta}_{d_y} = d_y^{-1}\sum_{i=1}^{d_y} y_i$ and one can even find an explicit expression of the posterior $p_{\hat{\theta}_{d_y}}(.|y_i)$ (see Kuntz et al. (2023) Sect E.1).

For the sake of comparison, we use an alternative version of this model that can be found in Lim et al. (2024), ToyHM$(10, 12)$ *i.e.* we fix $\sigma = 12$ and generate the data using $\theta = 10$, where we generate a single observation of $d_y = 20$ dimensions, particle cloud is initialized by $X_0 \sim \mathcal{N}(0, 1)$ using $N = 20$ particles.

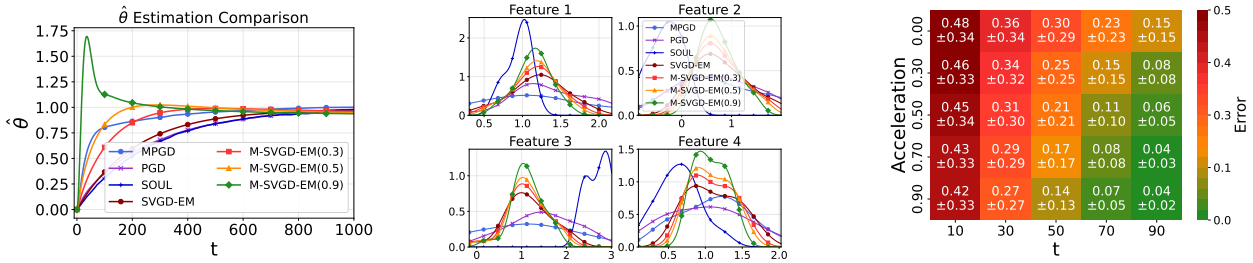
All algorithms are initialized with parameter $\theta_0 \sim \mathcal{U}(-3, 3)$. To determine the optimal learning rates γ,

[1] https://github.com/Farattel/m-svgd-em



(a) Parameter Estimation (b) MSE with $\hat{\theta}_{d_y}$ (c) Average iterations

Figure 1: Comparison of M-SVGD-EM with the SVGD-EM. Part (a) shows the performance on parameter estimation on a single iteration on the ToyHM(10, 12) with different acceleration parameters $\alpha_\theta = \alpha_X = (0.3, 0.5, 0.9)$.(b) displays MSE with respect to the empirical minimizer for each algorithm. (c) shows the average number of iterations until convergence, defined as being within a threshold of 0.05 from the empirical minimizer. The results are averaged over 20 independent trials.



(a) Parameter Estimation (b) Posterior Estimates (c) Test error Heatmap

Figure 2: (a) Parameter estimation convergence on BLR varying acceleration. (b) Posterior distributions across test features, showing acceleration improves convergence. (c) Test error vs. acceleration for different t values, showing error reduction with increased acceleration.

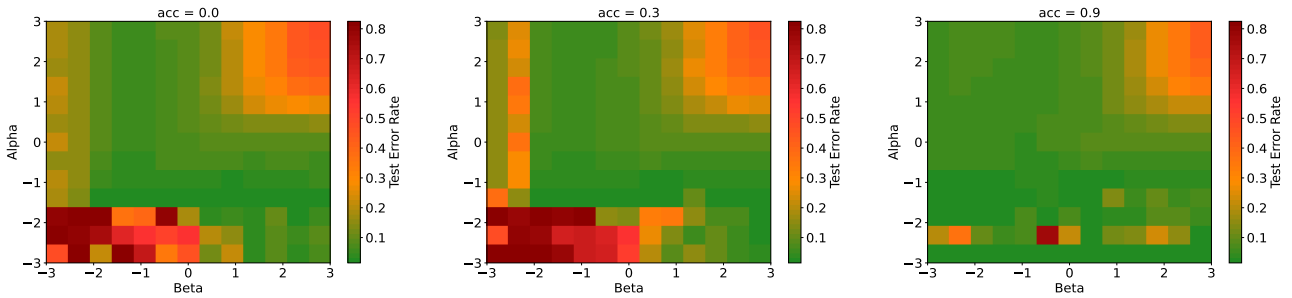
we conduct a fine grid of 100 logarithmically spaced values in $\gamma \in [10^{-3}, 10^2]$. For each γ we run the algorithms for a fixed number of iterations $T = 1000$ and select the γ which minimizes the MSE for each algorithm over 10 different trials. In Figure 1 results displayed use standard gradient descent with $\gamma = 0.01$ for SOUL,$\gamma = 0.2$ for both PGD and MPGD,$\gamma = 0.3$ for both SVGD-EM and M-SVGD-EM. In Figure 1a we observe that M-SVGD-EM consistently outperforms the SVGD-EM across different acceleration parameters $\alpha = \alpha_\theta = \alpha_X$. Figure 1b shows faster drop on the MSE for all accelerations and in particular acceleration $\alpha = 0.9$ achieves the same MSE as SVGD-EM in approximately 50% of its iterations. Figure 1c demonstrates substantial computational savings, with M-SVGD-EM(0.9) requiring only 232 ± 60.7 iterations to converge compared to 450.9 ± 115.1 for SVGD-EM, representing almost 50% number of iterations on average and standard error. These results are averaged over 20 seeds indicating M-SVGD-EM consistent improvement from SVGD-EM the ToyHM(10, 12) model.

4.2 Bayesian Logistic Regression

We now consider a Bayesian Logistic Regression Model on the *Wisconsin Breast Cancer Dataset* (Wolberg et al., 1993) with Gaussian priors on the latent variables represented by the regression weights. The prior on the weights is $\mathcal{N}(\theta \mathbf{1}_{d_x}, 5I_{d_x})$ and we consider the labels l are independent conditioned on the weights and features. The Bayesian Logistic regression model is used to predict the labels based on the data and latent variables, the model supposes $p_\theta(\mathcal{Y}|x) = \prod_{(f,l)\in\mathcal{Y}} s(f^T x)^l [1 - s(f^T x)]^{1-l}$. The joint model density is then:

$$\begin{aligned} p_\theta(x, \mathcal{Y}) &= p_\theta(x) p(\mathcal{Y}|x) \\ &= \mathcal{N}(\theta \mathbf{1}_{d_x}, 5I_{d_x}) \prod_{(f,l)\in\mathcal{Y}} s(f^T x)^l [1 - s(f^T x)]^{1-l}, \end{aligned}$$

where f denotes the features, l the labels, s is the sigmoid function $s(x) = (1 + e^{-x})^{-1}$ and $\mathcal{Y}$ is the data set. In Figure 2, we present the results using standard gradient descent with $N = 20$ particles, a step-size of $\gamma = 0.01$ for all methods determined by a fine-grid search over 100 logarithmically spaced ranged from



(a) SVGD-EM (b) M-SVGD-EM(0.3) (c) M-SVGD-EM(0.9)

Figure 3: Different Initilisations of α and β for accelerations $(0, 0.3, 0.9)$ with $N = 10$ for $T = 500$.

$\gamma \in [10^{-7}, 10^{2}]$ and averaged over 5 distinct seeds for a fixed number of $T = 500$ iterations.

We notice in Figure 2a.(a) that for all α (acceleration) parameters M-SVGD-EM outperforms the non-accelerated methods SVGD-EM, SOUL, and PGD. However, the accelerated method MPGD is a competitive approach for which the number of time-steps required to converge is similar to M-SVGD-EM.

Figure 2b presents the kernel density estimates (KDEs) of their final particle distributions showing the distribution of the first 4 features. We see that as α grows M-SVGD-EM the posterior estimates reach a higher peak and tighter distribution implying a smaller variance, showing that acceleration provides a more confident estimate rather than SVGD-EM. In Figure 2c, we demonstrate the average test error over 20 independent $(80\%, 20\%)$ train and test respectively splits using a step size $\gamma = 10^{-5}$ which is again within the optimal range as shown in Appendix D.

We use $N = 20$ particles and we vary the acceleration and the number of iterations T. The higher the acceleration the more rigorously the test error drops achieving a better performance with less iterations.

4.3 Bayesian Neural Network MNIST

We consider the following setup that is also described in Kuntz et al. (2023) and Sharrock et al. (2024) for the binary classification task on the MNIST dataset between digits 4 and 9. The dataset can be described as $\mathcal{Y} = \{(z_i, y_i)\}_{i=1}^{N_{\mathcal{Y}}}, z_i \in \mathbb{R}^{784}$, $y_i \in \{0, 1\}$, $N_{\mathcal{Y}} = 1000$ where each feature meaning each pixel-dimension has been standardized across the dataset. The model parameters are $x = (w, v)$ where $w \in \mathbb{R}^{40\times 784}$, $v \in \mathbb{R}^{2\times 40}$. We assume the likelihood

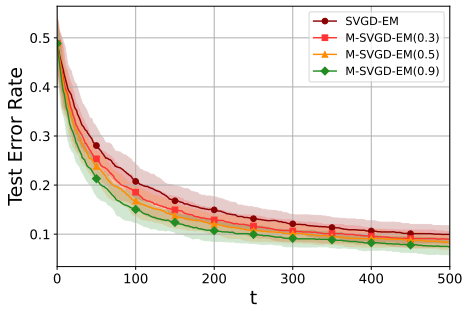
$$p(y_i|z_i, x) \propto \exp\left(\sum_{j=1}^{40} v_{ij} \tanh\left(\sum_{k=1}^{784} w_{jk} z_{ik}\right)\right).$$

We place independent Gaussian priors with log-variances $\theta = (\alpha, \beta)$, i.e. $w_{jk} \sim \mathcal{N}(0, e^{2\alpha})$ and $v_{ij} \sim \mathcal{N}(0, e^{2\beta})$. Thus the joint density is

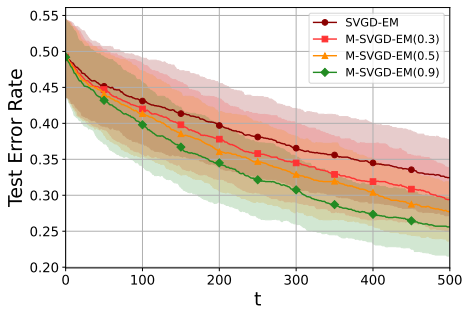
$$p_\theta(x, \mathcal{Y}) = \Big[\prod_{j=1}^{40}\prod_{k=1}^{784} \mathcal{N}(w_{jk}; 0, e^{2\alpha})\Big] \times \Big[\prod_{c=0}^{1}\prod_{j=1}^{40} \mathcal{N}(v_{cj}; 0, e^{2\beta})\Big] \prod_{i=1}^{N} p(y_i \mid z_i, x).$$

We run our method with $N = 10$. The optimizer used for the both updates in this experiment is AdaGrad (Duchi et al., 2011) for the gradient descent step, with our additional momentum acceleration term applied post-update, effectively forming a Momentum-AdaGrad hybrid can be found in the Appendix 2. All methods are run for $T = 500$ iterations and $\gamma = 0.1$. We run the experiment for 10 independent trials and used the data split $(80\%, 20\%)$ for train and test respectively. In Figure 4, we present the test error between SVGD-EM and M-SVGD-EM using two different initializations of θ. In the case $\theta_0 = (0, 0)$ illustrated in Figure 4a M-SVGD-EM demonstrates better performances than SVGD-EM. This is also the case in Figure 4b where $\theta_0 = (2, 2)$. This last case is included in the study as we observed that the predictive performance is heavily affected by different initializations of $\theta_0 = (\alpha_0, \beta_0)$, which can cause the algorithms to converge to local minima. Similarly, in Figure 5(a)–(b), we compare the log predictive probability density with M-SVGD-EM consistently outperforming SVGD-EM across all accelerations. More results can be found in the Appendix B with $N \in \{5, 10, 20\}$ showing consistency of M-SVGD-EM to provide better predictive performance in different settings.

The heatmaps in Figure 3 show how acceleration behaves with different initializations of $\theta = (\alpha, \beta)$ for a fixed number of iterations $T = 500$ and a learning rate $\gamma = 0.1$ using Adagrad as an optimizer. In Figures 3a–3b, where acceleration is 0.3 we can see minimal difference between the test errors in SVGD-EM

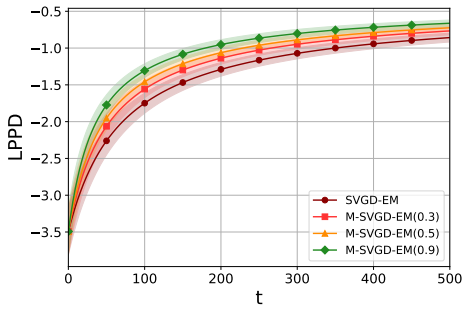


(a) Test error, $(\alpha_0, \beta_0) = (0, 0)$

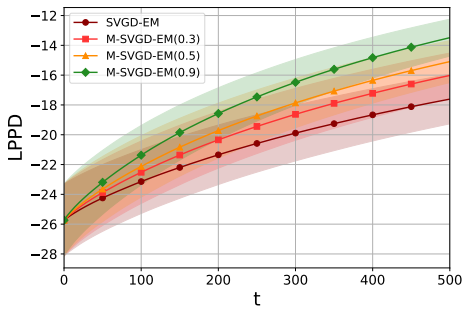


(b) Test error, $(\alpha_0, \beta_0) = (2, 2)$

Figure 4: Test Error w.r.t iterations t for the BNN on MNIST dataset



(a) Lppd, $(\alpha_0, \beta_0) = (0, 0)$



(b) Lppd, $(\alpha_0, \beta_0) = (2, 2)$

Figure 5: Log predictive probability distribution vs iterations t for the BNN on MNIST dataset

and M-SVGD-EM(0.3), whereas for a high acceleration value M-SVGD-EM(0.9) remains stable and dominates providing a much lower test-error for almost all combinations of $\theta = (\alpha, \beta)$, demonstrating the potential of using a high-acceleration to escape local minima on the parameter estimation.

5 CONCLUSION

We have proposed a new method for accelerated MMLE using accelerated SVGD and optimization methods. The resulting method, M-SVGD-EM, results in a significantly faster method in terms of the number of iterations required for convergence. On a number of examples, we have demonstrated the speedup provided by M-SVGD-EM over the standard SVGD-EM method, as well as other state-of-the-art methods such as PGD and SOUL.Furthermore, we compared M-SVGD-EM with the accelerated algorithm MPGD (Lim et al., 2024). On the Toy Hierarchical Model M-SVGD-EM is slower but achieves a lower and more stable MSE, while on Bayesian Logistic Regression it performs competitively, converging in a similar number of iterations. All methods were compared after tuning.

Limitations and future work. Our method is derived using the acceleration ideas from optimization and notably in the space of measures $\mathcal{P}_{2,ac}(\mathcal{X})$. It accounts for the interactions between the particles, leading to $\mathcal{O}(N^2)$ operations to update the particle cloud. This fact limits the scalability of the method with respect to the number of particles. However, the proposed acceleration method partially addresses this issue by reducing the number of iterations required for convergence by up to 50% in the conducted experiments. A limitation of the method is it is based on an approximation introduced in Liu et al. (2019) relying on a heuristic and the theoretical explanation of the performance of our method hence is not provided here. We believe that investigating potential applications of the work in Stein and Li (2025) to our setting can overcome this problem.

We also highlight potential applications of M-SVGD-EM beyond those considered in this paper. By leveraging existing interacting particle methods and their formulations, our approach can be adapted to accelerate inverse problem solvers (Glyn-Davies et al., 2025), to train energy-based generative models (Marks et al., 2025), and to train latent diffusion models (Wang et al., 2025). These directions represent promising avenues for future work.

Broader impact. The proposed method is a novel MMLE scheme for LVMs which find applications in all

areas of science. A potential positive societal impact of our method is it provides a significant speedup over existing methods, which can be beneficial in many applications as the computation times could be slashed, resulting in significant savings in terms of computational resources. Like many other training methods for probabilistic machine learning models, our method can be used to train classifiers or generative models which can be used for harmful purposes. However, this risk is not amplified here by our work and is the same as standard algorithms to train probabilistic models.

Acknowledgements

Part of this work was completed while A. R. was visiting O. D. A.'s research group in the Department of Mathematics at Imperial College London as an intern. R. A. is supported by the EPSRC through the Modern Statistics and Statistical Machine Learning (StatML) CDT programme under grant EP/S023151/1. O. D. A. is grateful to Department of Statistics, London School of Economics and Political Science (LSE) for the hospitality during the preparation of this work.

References

Akyildiz, Ö. Deniz et al. (2025). "Interacting Particle Langevin Algorithm for Maximum Marginal Likelihood Estimation". In: *ESAIM: Probability and Statistics*.

Ambrosio, Luigi, Nicola Gigli, and Giuseppe Savaré (2008). *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media.

Attia, Amit and Tomer Koren (2021). "Algorithmic instabilities of accelerated gradient descent". In: *Advances in Neural Information Processing Systems* 34, pp. 1204–1214.

Bradbury, James et al. (2018). *JAX: Composable transformations of Python+NumPy programs*. Version 0.3.13.

Caprio, Rocco et al. (2025). "Error bounds for particle gradient descent, and extensions of the log-Sobolev and Talagrand inequalities". In: *Journal of Machine Learning Research* 26.103, pp. 1–38.

Cuturi, Marco (2013). "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Advances in neural information processing systems* 26.

De Bortoli, Valentin et al. (2021). "Efficient stochastic optimisation by unadjusted Langevin Monte Carlo: Application to maximum marginal likelihood and empirical Bayesian estimation". In: *Statistics and Computing* 31, pp. 1–18.

Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22.

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7.

Encinar, Paula Cordero, Francesca R Crucinio, and O Deniz Akyildiz (2025). "Proximal Interacting Particle Langevin Algorithms". In: *Uncertainty in Artificial Intelligence (UAI)*.

Glyn-Davies, Alex et al. (2025). "Statistical finite elements via interacting particle Langevin dynamics". In: *SIAM/ASA Journal on Uncertainty Quantification* 13.3, pp. 1200–1227.

Kuntz, Juan, Jen Ning Lim, and Adam M Johansen (2023). "Particle algorithms for maximum likelihood training of latent variable models". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 5134–5180.

Lange, Kenneth (1995). "A gradient algorithm locally equivalent to the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 57.2, pp. 425–437.

Lim, JN et al. (2024). "Momentum particle maximum likelihood". In: *Proceedings of 41st International Conference on Machine Learning (ICML)*. Vol. 235, pp. 29816–29871.

Liu, Chang et al. (2019). "Understanding and accelerating particle-based variational inference". In: *International Conference on Machine Learning*. PMLR, pp. 4082–4092.

Liu, Chuanhai and Donald B Rubin (1994). "The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence". In: *Biometrika* 81.4, pp. 633–648.

Liu, Qiang and Dilin Wang (2016). "Stein variational gradient descent: A general purpose bayesian inference algorithm". In: *Advances in neural information processing systems* 29.

Marks, Joanna, Tim YJ Wang, and O Deniz Akyildiz (2025). "Learning latent energy-based models via interacting particle Langevin dynamics". In: *arXiv preprint arXiv:2510.12311*.

Meng, Xiao-Li and Donald B Rubin (1993). "Maximum likelihood estimation via the ECM algorithm: A general framework". In: *Biometrika* 80.2, pp. 267–278.

Neal, Radford M and Geoffrey E Hinton (1998). "A view of the EM algorithm that justifies incremental, sparse, and other variants". In: *Learning in graphical models*. Springer, pp. 355–368.

Nesterov, Yu (2005). "Smooth minimization of non-smooth functions". In: *Mathematical programming* 103, pp. 127–152.

Nesterov, Yurii (1983). "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$". In: *Dokl. Akad. Nauk. SSSR.* Vol. 269, p. 543.

– (1988). "On an approach to the construction of optimal methods of minimization of smooth convex functions". In: *Ekonomika i Mateaticheskie Metody* 24.3, pp. 509–517.

– (2013). *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media.

Oliva, Paul Felix Valsecchi and O Deniz Akyildiz (2024). "Kinetic interacting particle Langevin Monte Carlo". In: *arXiv preprint arXiv:2407.05790*.

Otto, Felix (2001). "The geometry of dissipative evolution equations: the porous medium equation". In.

Pele, Ofir and Michael Werman (2009). "Fast and robust earth mover's distances". In: *2009 IEEE 12th international conference on computer vision*. IEEE, pp. 460–467.

Santambrogio, Filippo (2015). *Optimal transport for applied mathematicians*. Vol. 87. Springer.

Sharrock, Louis, Daniel Dodd, and Christopher Nemeth (2024). "Tuning-Free Maximum Likelihood Training of Latent Variable Models via Coin Betting". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1810–1818.

Stein, Viktor and Wuchen Li (2025). "Accelerated Stein variational gradient flow". In: *International Conference on Geometric Science of Information*. Springer, pp. 80–90.

Villani, Cédric et al. (2009). *Optimal transport: old and new*. Vol. 338. Springer.

Wang, Tim YJ, Juan Kuntz, and O Deniz Akyildiz (2025). "Training Latent Diffusion Models with Interacting Particle Algorithms". In: *arXiv preprint arXiv:2505.12412*.

Wei, Greg CG and Martin A Tanner (1990). "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms". In: *Journal of the American statistical Association* 85.411, pp. 699–704.

Whiteley, Nick, Annie Gray, and Patrick Rubin-Delanchy (2025). "Statistical exploration of the Manifold Hypothesis". In: *Journal of the Royal Statistical Society: Series B*.

Wolberg, William et al. (1993). *Breast Cancer Wisconsin (Diagnostic)*. UCI Machine Learning Repository.

Xie, Yujia et al. (2020). "A fast proximal point method for computing exact wasserstein distance". In: *Uncertainty in artificial intelligence*. PMLR, pp. 433–453.

Zhang, Hongyi and Suvrit Sra (2018). "An estimate sequence for geodesically convex optimization". In: *Conference On Learning Theory*. PMLR, pp. 1703–1723.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   Details on mathematical settings for every equation can be found in the Notation paragraph in the Introduction. M-SVGD-EM is defined in 1. The pairwise closeness assumption for the derivation in the appendix is defined by 18.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   Time complexity is analyzed in Section 4. Different sample sizes (numbers of particles) are displayed in the Appendix B.2 and B.3.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
   The source code is in the supplementary material ZIP file

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   The code is provided in the supplementary material to reproduce the results.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   Training details can be found in Section 4 and in the Appendix for more details.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]

Details for these are in every experiment and can be found in Section 4 and in any further experiments or tuning presented in the Appendix.

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
Stated in introduction of Section 4

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Yes]
   We extend the code available at `https://github.com/chris-nemeth/coinem`.

   (b) The license information of the assets, if applicable. [Yes]

   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

   (d) Information about consent from data providers/curators. [Yes]
   We cite Wisconsin Breast Cancer Dataset (Wolberg et al., 1993) in 4.2. The MNIST dataset used in 4.3 is an open access dataset.

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Momentum SVGD-EM for Accelerated Maximum Marginal Likelihood Estimation
Supplementary Material

A DERIVATION OF SVGD-WNES FROM SVGD-RAGD

A.1 Practical tools for the derivation

In the following, we derive SVGD-WNEs from the Riemann-Nesterov Stein variational gradient descent (SVGD-RAGD). To facilitate this derivation, we introduce a set of practical tools that make the exponential map and its inverse tractable and composable:

- Let $\{\tilde{x}^{(i)}\}_{1\leq i\leq N}$ be a set of particles sampled from $\tilde{q} \in \mathcal{P}_{2,ac}(\mathcal{X})$, then the particle approximation of the exponential map at q defined by $\mathrm{Exp}_{\tilde{q}}(v) := (\mathrm{id} + v)_{\#}\,\tilde{q}$ where $v \in \mathcal{T}_{\tilde{q}}\mathcal{P}_{2,ac}(\mathcal{X})$ is the application of the map $x \mapsto x + v(x)$ on each particle (Liu et al., 2019), thus

$$\mathrm{Exp}_{\tilde{q}}(v)(\tilde{x}^{(i)}) = \tilde{x}^{(i)} + v(\tilde{x}^{(i)}). \tag{17}$$

- Consider the density $q \approx \sum_i^N \delta_{x^{(i)}}$ and the momentum density $\tilde{q} \approx \sum_i^N \delta_{\tilde{x}^{(i)}}$. The pairwise closeness assumption between $\{x^{(i)}\}_{1\leq i\leq N}$ and $\{\tilde{x}^{(i)}\}_{1\leq i\leq N}$ is $d(x^{(i)}, \tilde{x}^{(i)}) \ll \min\left\{\min_{j\neq i} d(x^{(i)}, x^{(j)}), \min_{j\neq i} d(\tilde{x}^{(i)}, \tilde{x}^{(j)})\right\}$ where d is the euclidean distance in $\mathcal{X}$. As shown in Liu et al. (2019), this leads to

$$\mathrm{Exp}_{\tilde{q}}^{-1}(q)(\tilde{x}^{(i)}) \approx x^{(i)} - \tilde{x}^{(i)} \text{ with } 1 \leq i \leq N, \tag{18}$$

  where $x^{(i)} - \tilde{x}^{(i)} = v(\tilde{x}^{(i)})$ with $v(x) := x^{(i)} - x$. In the derivation, we use the assumption (18) on the particles sampled from the pairs $(\tilde{q}_{t-1}, q_t)$ and $(\tilde{q}_{t-1}, q_{t-1})$.

- Finally we will also encounter expressions of the general form $\mathrm{Exp}_q(v(x^{(i)}))$ where the approximation resulting from the usage of (18) serves as the input of the exponential map. In these cases we simply use (17) and deduce

$$\mathrm{Exp}_q(v(x^{(i)})) = x^{(i)} + v(x^{(i)}). \tag{19}$$

Before proceeding with the derivation, we highlight that the particle-level versions of the exponential map and its inverse yield an analogue of the identity $\mathrm{Exp}_{\tilde{q}}(\mathrm{Exp}_{\tilde{q}}^{-1}(q)) = q$. Specifically, with $1 \leq i \leq N$ we have:

$$\begin{aligned}
\mathrm{Exp}_{\tilde{q}}(\mathrm{Exp}_{\tilde{q}}^{-1}(q)(\tilde{x}^{(i)})) &= \tilde{x}^{(i)} + \mathrm{Exp}_{\tilde{q}}^{-1}(q)(\tilde{x}^{(i)}) \quad \text{using (17)} \\
&\approx \tilde{x}^{(i)} + x^{(i)} - \tilde{x}^{(i)} \text{ using (18)} \\
&= x^{(i)},
\end{aligned}$$

where we emphasize that the inverse exponential map is taken at $\tilde{q}$ and thus evaluated at the particle $\tilde{x}^{(i)}$.

A.2 Derivation

We are now equipped to derive one iteration of SVGD-WNEs from the corresponding iteration of SVGD-RAGD given by:

$$\begin{aligned}
q_{t+1} &= \mathrm{Exp}_{\tilde{q}_t}(-\gamma \mathbb{W}_{H_k} \mathcal{F}(\theta, \tilde{q}_t)) \\
\tilde{q}_{t+1} &= \mathrm{Exp}_{q_{t+1}}\{c_1 \mathrm{Exp}_{q_{t+1}}^{-1}[\mathrm{Exp}_{\tilde{q}_t}(\mathrm{Exp}_{\tilde{q}_t}^{-1}(q_{t+1}) + (c_2 - 1)(\mathrm{Exp}_{\tilde{q}_t}^{-1}(q_{t+1}) - \mathrm{Exp}_{\tilde{q}_t}^{-1}(q_t)))]\}.
\end{aligned}$$

Letting $q_t \approx \sum_i^N \delta_{x_t^{(i)}}$ and $\tilde{q}_t \approx \sum_i^N \delta_{\tilde{x}_t^{(i)}}$, we obtain the particle-level formulation of SVGD-RAGD :

$$x_{t+1}^{(i)} = \mathrm{Exp}_{\tilde{q}_t}(-\gamma \mathbb{W}_{H_k} \mathcal{F}(\theta, \tilde{q}_t))(\tilde{x}_t^{(i)}) \tag{20}$$

$$\tilde{x}_{t+1}^{(i)} = \mathrm{Exp}_{q_{t+1}}\{c_1 \mathrm{Exp}_{q_{t+1}}^{-1}[\mathrm{Exp}_{\tilde{q}_t}(\mathrm{Exp}_{\tilde{q}_t}^{-1}(q_{t+1})(\tilde{x}_t^{(i)}) + (c_2 - 1)(\mathrm{Exp}_{\tilde{q}_t}^{-1}(q_{t+1})(\tilde{x}_t^{(i)}) - \mathrm{Exp}_{\tilde{q}_t}^{-1}(q_t)(\tilde{x}_t^{(i)})))]\}. \tag{21}$$

Recalling (17), we can rewrite (20) as:

$$\begin{aligned} x_{t+1}^{(i)} &= \tilde{x}_t^{(i)} - \gamma \mathbb{W}_{H_k} \mathcal{F}(\theta, \tilde{q}_t)(\tilde{x}_t^{(i)}) \\ &= \tilde{x}_t^{(i)} + \gamma \frac{1}{N} \sum_{j=1}^{N} \left[k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)}) \nabla_x \ell(\theta, \tilde{x}_t^{(j)}) + \nabla_1 k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)}) \right], \end{aligned}$$

which corresponds to (15). Next we use (18) in (21), yielding:

$$\tilde{x}_{t+1}^{(i)} = \mathrm{Exp}_{q_{t+1}}\{c_1 \mathrm{Exp}_{q_{t+1}}^{-1}[\mathrm{Exp}_{\tilde{q}_t}(x_{t+1}^{(i)} - \tilde{x}_t^{(i)} + (c_2 - 1)(x_{t+1}^{(i)} - x_t^{(i)}))]\}.$$

This corresponds to the setting in (19), where the velocity vector field is given by $v_1(x) = x_{t+1}^{(i)} - x + (c_2 - 1)(x_{t+1}^{(i)} - x_t^{(i)})$. Hence, we obtain:

$$\tilde{x}_{t+1}^{(i)} = \mathrm{Exp}_{q_{t+1}}\{c_1 \mathrm{Exp}_{q_{t+1}}^{-1}[x_{t+1}^{(i)} + (c_2 - 1)(x_{t+1}^{(i)} - x_t^{(i)})]\} \tag{22}$$

Applying (18) in (22) and then (19) in (23), we further simplify:

$$\begin{aligned} \tilde{x}_{t+1}^{(i)} &= \mathrm{Exp}_{q_{t+1}}\{c_1[x_{t+1}^{(i)} + (c_2 - 1)(x_{t+1}^{(i)} - x_t^{(i)}) - x_{t+1}^{(i)}]\} \\ &= \mathrm{Exp}_{q_{t+1}}\{c_1(c_2 - 1)(x_{t+1}^{(i)} - x_t^{(i)})\} \\ &= x_{t+1}^{(i)} + c_1(c_2 - 1)(x_{t+1}^{(i)} - x_t^{(i)}), \end{aligned} \tag{23}$$

where (19) is applied with the velocity vector field $v_2(x) = c_1(c_2 - 1)(x - x_t^{(i)})$. We thus obtain the final SVGD-WNes update:

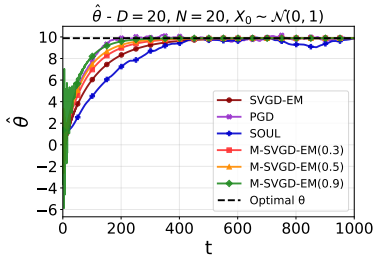
$$\begin{aligned} x_{t+1}^{(i)} &= \tilde{x}_t^{(i)} + \gamma \frac{1}{N} \sum_{j=1}^{N} k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)}) \nabla_x \ell(\theta, \tilde{x}_t^{(j)}) + \nabla_1 k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)}) \\ \tilde{x}_{t+1}^{(i)} &= x_{t+1}^{(i)} + c_1(c_2 - 1)(x_{t+1}^{(i)} - x_t^{(i)}) \end{aligned}$$

B RESULTS WITH MEDIANRBF

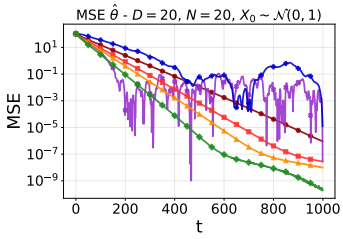
We reproduce the same results presented in the main paper using the MedianRBF kernel instead of the AutoRBF kernel.

B.1 Toy Hierarchical Example MedianRBF

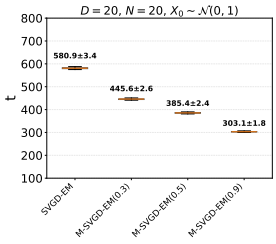
In Figure 6a, we observe that M-SVGD-EM consistently outperforms SVGD-EM across different acceleration parameters $\alpha = \alpha_\theta = \alpha_X$. As shown in Figure 6b, the mean squared error (MSE) decreases for all values of α, with $\alpha = 0.9$ achieving the lowest MSE within $T = 1000$ iterations. Similarly, Figure 6c demonstrates substantial computational savings with $\alpha = 0.9$ achieving again achieving nearly a 50% reduction in iterations. Although the average number of iterations is higher than in Figure 1c, the variance is significantly lower. These results are averaged over 20 independent trials, using standard gradient descent. For SOUL and PGD, we use optimal learning rates $\gamma = (0.01, 0.2)$ (see Figures 7a–7b), while for both SVGD-EM and M-SVGD-EM, we set $\gamma = 1.15$.



(a) Parameter Estimation

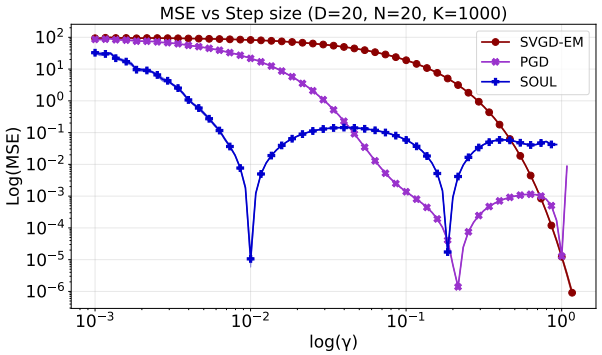


(b) MSE($\hat{\theta}$) vs t

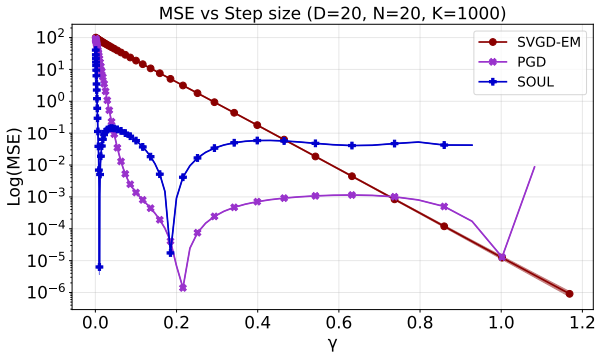


(c) Average Iterations

Figure 6: Comparison between M-SVGD-EM and SVGD-EM. Part (a) shows the performance on parameter estimation on a single run on the ToyHM(10, 12) with different acceleration parameters $\alpha_\theta = \alpha_X = (0.3, 0.5, 0.9)$. Part (b) displays MSE relative to the empirical minimizer for each algorithm. (c) shows the average number of iterations until convergence, defined as being within a threshold of 0.05 from the empirical minimizer. The results are averaged over 20 independent trials.



(a) Log-Log Plot MSE vs LR

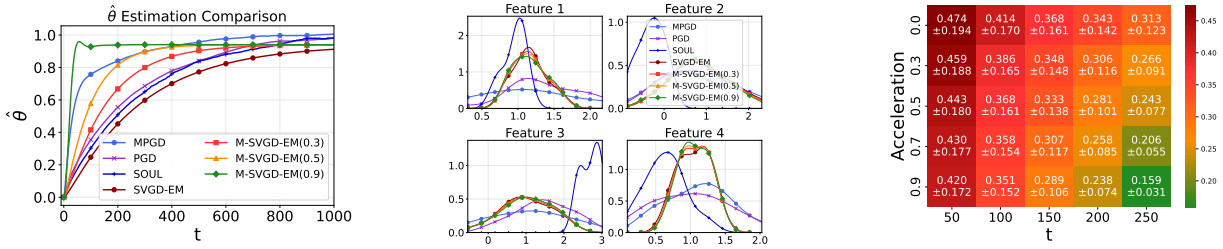


(b) Log-Log Plot MSE vs LR

Figure 7: MSE vs Learning Rate

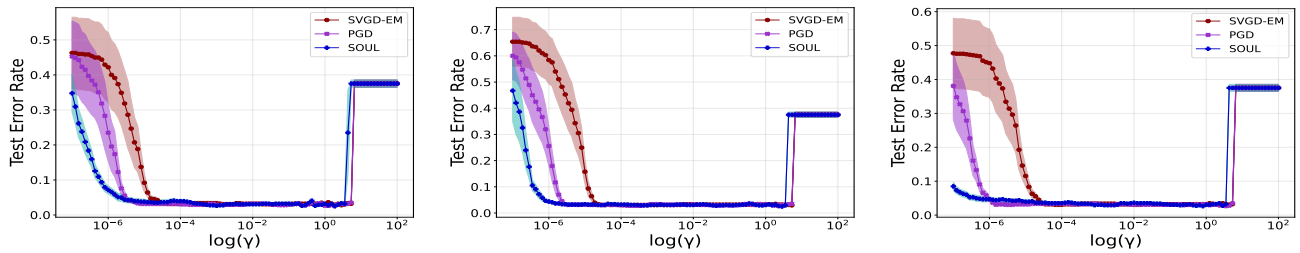
B.2 Bayesian Logistic Regression MedianRBF

In Figure 8, we again present the results on BLR using standard gradient descent with $N = 20$ particles, a step-size of $\gamma = 0.01$ for all methods determined by a fine-grid search over 100 logarithmically spaced ranged from $\gamma \in [10^{-7}, 10^{2}]$ and averaged over 5 distinct seeds for a fixed number of iterations $T = 500$. As shown in Figure 9b the value of γ was chosen based on test error performance. Figure 8b presents the kernel density estimates (KDEs) of their final particle distributions for the the first 4 features. In comparison to using the AutoRBF in Figure 2b, the posterior estimates here overlap between accelerations and are not significantly different from each other. In Figure 8c, we demonstrate the average test error over 20 independent train (80%) and test (20%) splits using a step size $\gamma = 10^{-5}$ which is again within the optimal range as shown in Figure 9b. Using MedianRBF takes longer (Figure 8c) to reach a low test error compared to using the AutoRBF, as shown in Figure 2c. This highlights also the effect of kernels for convergence. Overall, we still see acceleration outperforms standard SVGD-EM when considering $N = 20$ particles, over $T = 500$ iterations. The higher the acceleration the more rigorously the test error drops achieving a better performance with less iterations.



(a) Parameter Estimation (b) Posterior Estimates (c) Test Error Heatmap

Figure 8: (a) Convergence of parameter estimates on BLR under varying acceleration parameters. (b) Posterior distributions across 4 test features. (c) Test error as a function of the acceleration parameter for different values of t, showing consistent error reduction with increased acceleration.



(a) $N = 10$ (b) $N = 20$ (c) $N = 100$

Figure 9: Test Error (Accuracy) for Bayesian Logistic Regression with different numbers of particles.

B.3 Bayesian Neural Network

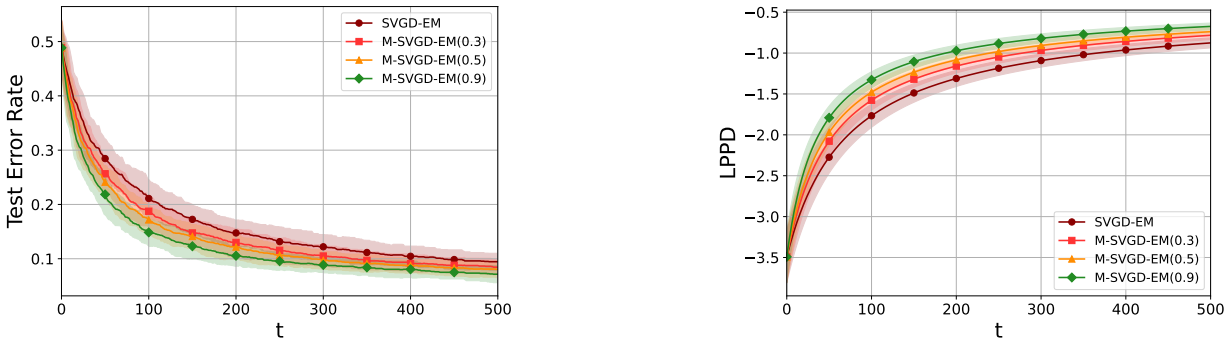
We use in this section a modification of our method, provided in Algorithm 2.

Algorithm 2 M-SVGD-EM with AdaGrad

Input: momentum constants $\alpha_X, \alpha_\theta \in \mathbb{R}$, initial particles $\{x_0^{(i)}\}_{i=1}^N \sim q_0$, initial parameter θ_0, joint log-likelihood ℓ, kernel k, learning rate η, stability constant ϵ.
Initialize: $\theta_0 = \tilde{\theta}_0$, $\{x_0^{(i)}\}_{i=1}^N = \{\tilde{x}_0^{(i)}\}_{i=1}^N$, $G_\theta^{(0)} = 0$, $G_x^{(i,0)} = 0$ for $i = 1, \dots, N$
for $t = 0, 1, \dots, T-1$ **do**
  Update θ with AdaGrad
  $g_\theta^{(t)} = \frac{1}{N} \sum_{j=1}^N \nabla_\theta \ell(\tilde{\theta}_t, x_t^{(j)})$
  $G_\theta^{(t)} = G_\theta^{(t-1)} + (g_\theta^{(t)})^2$
  $\theta_{t+1} = \tilde{\theta}_t + \eta \cdot \frac{g_\theta^{(t)}}{\sqrt{G_\theta^{(t)} + \epsilon}}$
  $\tilde{\theta}_{t+1} = \theta_{t+1} + \alpha_\theta(\theta_{t+1} - \theta_t)$
  Update particles with AdaGrad
  for $i = 1, 2, \dots, N$ **do**
    $g_x^{(i,t)} = \frac{1}{N} \sum_{j=1}^N k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)}) \nabla_x \ell(\theta_{t+1}, \tilde{x}_t^{(j)})$
      $+ \nabla_1 k(\tilde{x}_t^{(j)}, \tilde{x}_t^{(i)})$
    $G_x^{(i,t)} = G_x^{(i,t-1)} + (g_x^{(i,t)})^2$
    $x_{t+1}^{(i)} = \tilde{x}_t^{(i)} + \eta \cdot \frac{g_x^{(i,t)}}{\sqrt{G_x^{(i,t)} + \epsilon}}$
    $\tilde{x}_{t+1}^{(i)} = x_{t+1}^{(i)} + \alpha_X(x_{t+1}^{(i)} - x_t^{(i)})$
  end for
end for

Below we present the results using a MedianRBF kernel on the BNN Example. Figures 10a–10b display the

test error and log-predictive-probability-density w.r.t t for the initialization $\alpha = \beta = 0.0$ using $N = 10$. The performance is comparable to that obtained with the AutoRBF kernel (Figures 4a–5a), where M-SVGD-EM outperforms SVGD-EM across both kernels. We present the same experiment varying the number of particles $N = 5, 10, 20$ for $\alpha = \beta = 0.0$ and $\alpha = \beta = 2.0$ (see Figures 11–14), it is clear that increasing the number of particles with MedianRBF does not offer a better performance especially when the initializations are poor. In Figures 13–14 for $N = 10$ results are heavily affected by poor initialization showing that the model is as good as a coin-flip, whereas in AutoRBF Figures 4b–5b, even though there still is a decline in the overall performance due to the poor initialization, AutoRBF is more robust providing a lower test error and a immediate decreasing on its trend.



(a) Test Error vs. iterations t

(b) Log Predictive Probability Density (LPPD)

Figure 10: Performance metrics for the BNN on MNIST dataset: (a) Test Error across iterations t, and (b) Log Predictive Probability Density (LPPD).

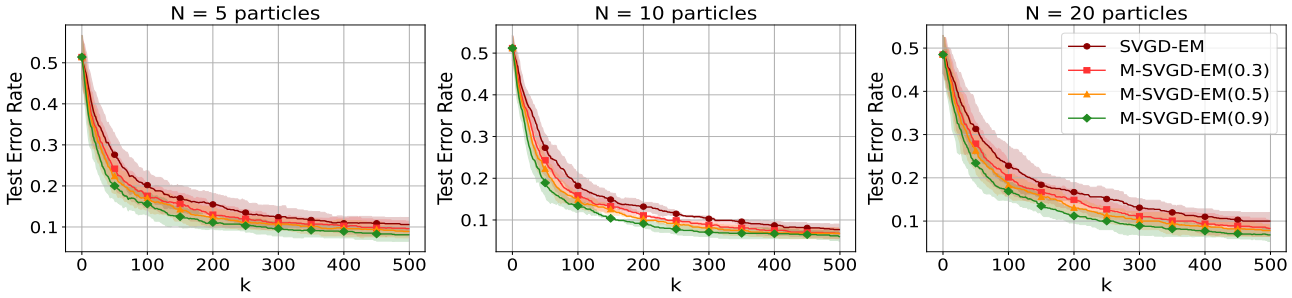


Figure 11: Test Error using $N = 5, 10, 20$ with initial $\alpha = \beta = 0.0$.

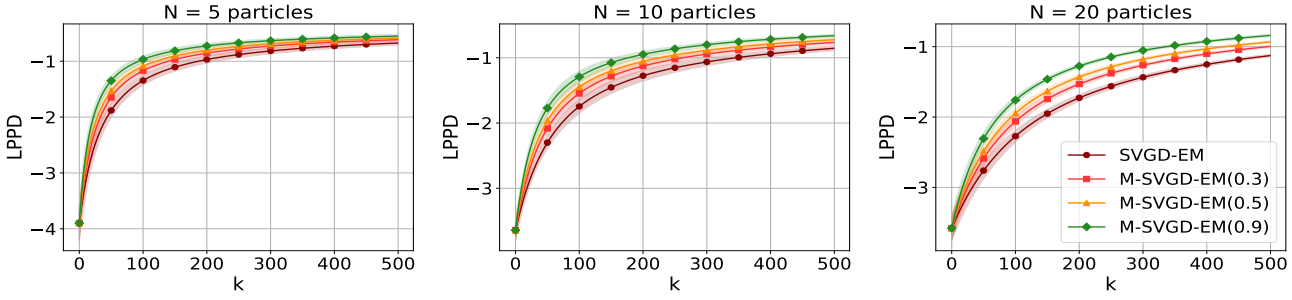


Figure 12: Log Predictive Probability Density using $N = 5, 10, 20$ with initial $\alpha = \beta = 0.0$.

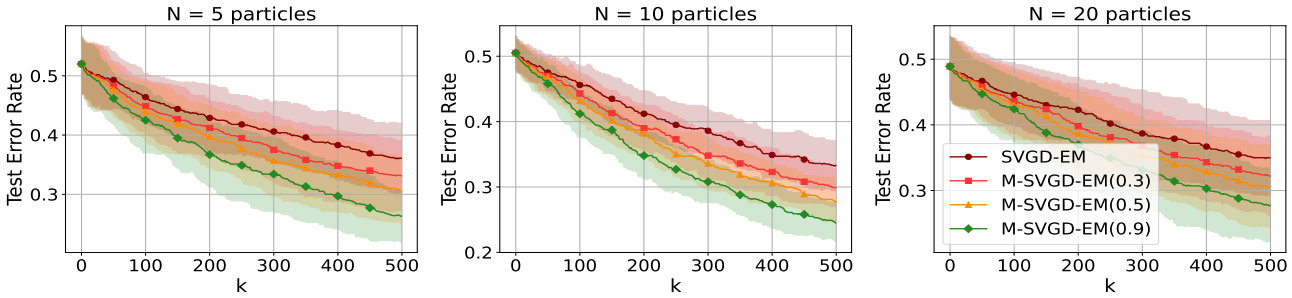


Figure 13: Test Error using $N = 5, 10, 20$ with initial $\alpha = \beta = 2.0$.

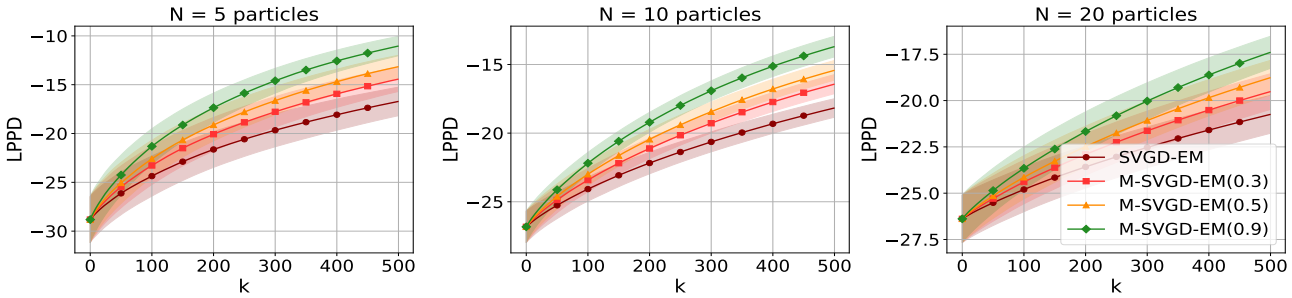
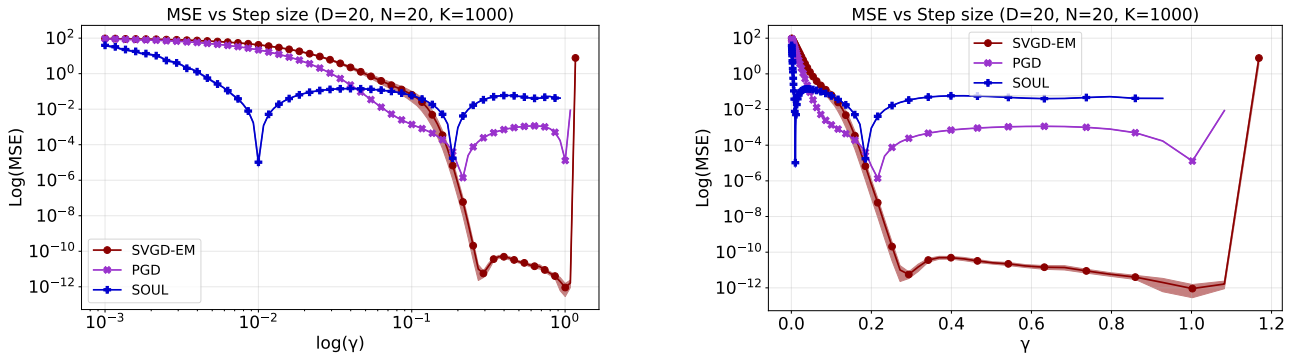


Figure 14: Log Predictive Probability Density using $N = 5, 10, 20$ with initial $\alpha = \beta = 2.0$.

C ADDITIONAL RESULTS ON TOY HIERARCHICAL MODEL

Figures 15a–15b show the selection of the optimal learning rates with respect to the lowest MSE for the results in section 4.1. Each method is run for a fixed number of iterations $T = 1000$ and averaged over 5 independent trials.



(a) Log-Log Plot MSE vs $\log(\gamma)$

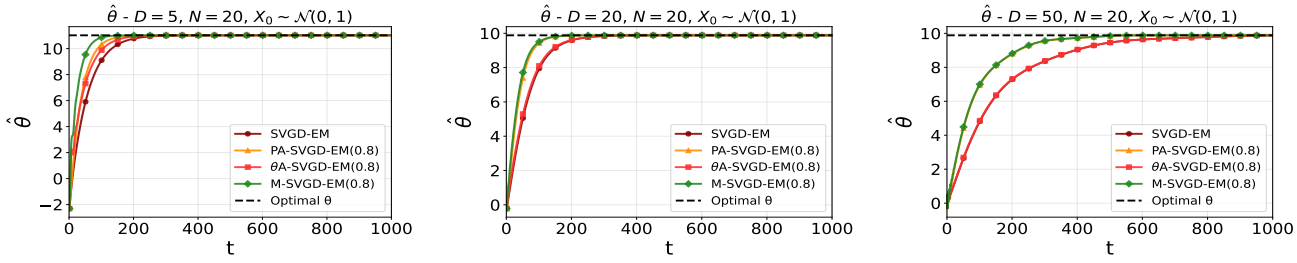
(b) Log(MSE) vs γ

Figure 15: MSE w.r.t Learning Rate over 10 independent trials.

C.1 Accelerations in Θ vs. in $\mathcal{P}_{2,ac}(\mathcal{X})$

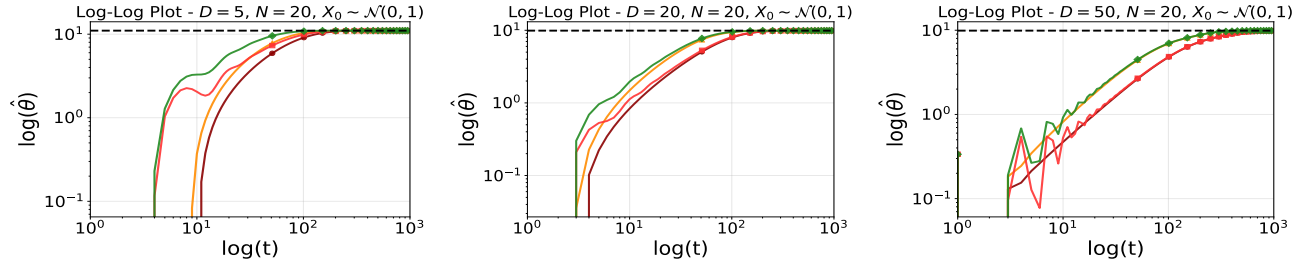
Here we present the results on the Toy Hierarchical model when accelerating only one part of SVGD-EM, namely the acceleration on the particles (PA) or the acceleration on the parameter (θA). We vary the dimensionality of the Toy Hierarchical Model with $D = 5, 20, 50$ to better assess and understand the effect of different acceleration mechanisms. In Figures 16–17, we display the parameter estimation against t iterations on linear and loglog scales respectively. Figure 18 shows the MSE with respect to the empirical minimizer for each method and finally

in Figure 19 we present the average iterations to converge with a threshold of 0.05 for 20 independent runs. All figures show that as we increase the dimensions of the problem, θA offers no improvement to SVGD-EM whereas PA is the only one responsible for accelerating the method.



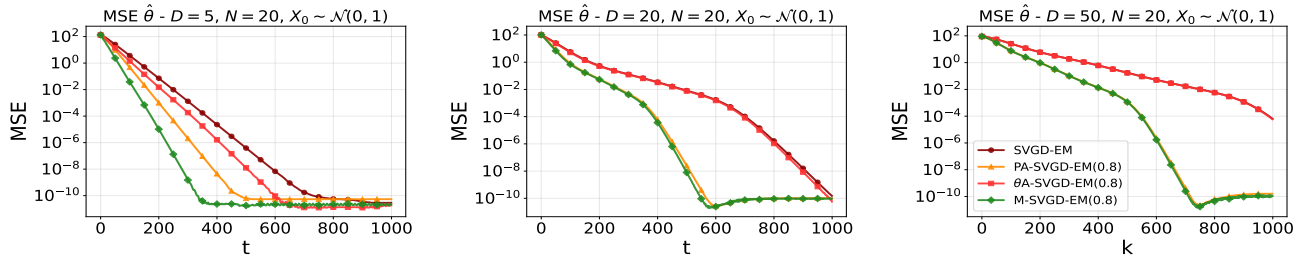
(a) $D = 5$ (b) $D = 20$ (c) $D = 50$

Figure 16: Parameter estimation of a single run varying dimensions $D = 5, 20, 50$.



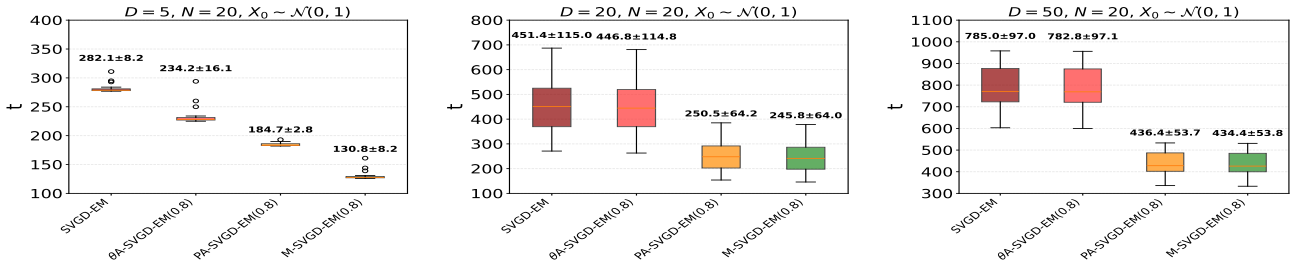
(a) $D = 5$ (b) $D = 20$ (c) $D = 50$

Figure 17: Log-Log Parameter estimation of a single run varying dimensions $D = 5, 20, 50$.



(a) $D = 5$ (b) $D = 20$ (c) $D = 50$

Figure 18: MSE vs iterations calculated w.r.t the empirical minimizer $D = 5, 20, 50$.

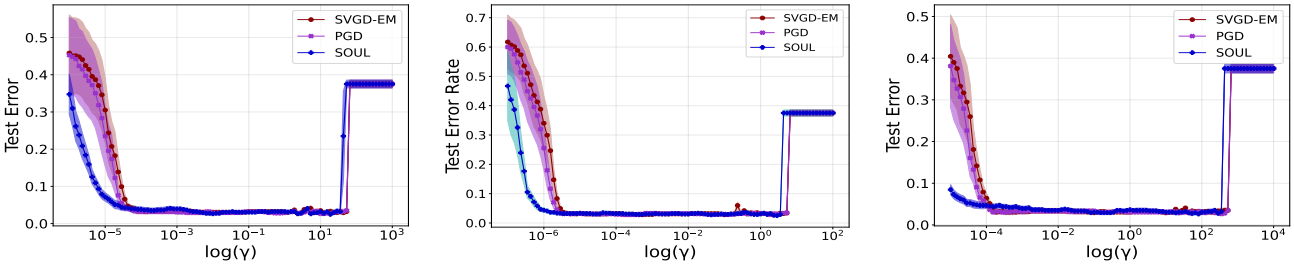


(a) $D = 5$ (b) $D = 20$ (c) $D = 50$

Figure 19: Average number of iterations ±se for 20 independent trials $D = 5, 20, 50$.

D ADDITIONAL RESULTS ON BAYESIAN LOGISTIC REGRESSION

We present here how we choose the optimal learning rate γ within the optimal range that minimizes the test error for different number of particles $N = 10, 20, 100$.



(a) $N = 10$ (b) $N = 20$ (c) $N = 100$

Figure 20: Test Error for Bayesian Logistic Regression with different numbers of particles.

E MPGD TUNING

Following the approach found in the original implementation (Lim et al., 2024). We do a grid search over 10 independent runs with the hyper-parameters ε damping, and η mass.

(Lim et al., 2024) We set the joint effect to be captured by momentum $\mu = 1 - \varepsilon\gamma\eta$, which is the velocity kept per update.

We fix (μ, ε) and choosing a familiar γ this sets $\mu = e^{-\gamma\eta\varepsilon} \to \eta = \frac{1-\mu}{\gamma\varepsilon}$

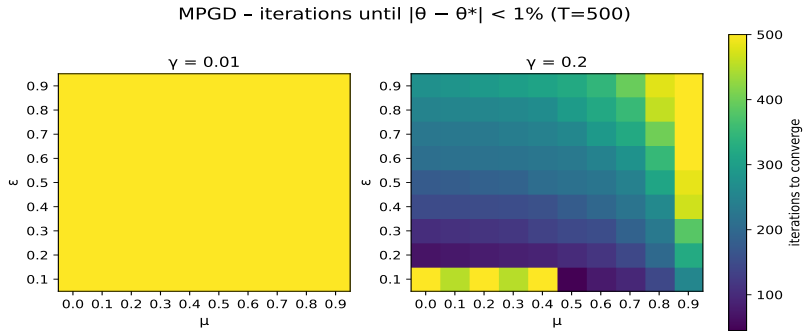


Figure 21: Heatmap Toy Hierarchical Model γ vs μ,with the darker color giving the least iterations needed until convergence

For the sake of consistency we fix the optimal learning rate found in 7a for PGD $\gamma = (0.01, 0.2)$ to provide a direct comparison against accelerated algorithm MPGD. In Figure 21 displays a Heatmap for each learning rate γ and the grid of $\varepsilon \in \{0.1k : k = 1, \ldots, 9\}, \mu \in \{0.1k : k = 0, \ldots, 9\}$, the darker the color the less iterations required for MPGD to converge. In Figure 1 hyper-parameters used were $(\epsilon, \mu) = (0.1, 0.2)$ or equivalently $(\epsilon, \eta) = (0.1, 20)$.

Similarly, in the BLR example Figure 2 we consider the optimal learning rate $\gamma = 0.01$ found in Figure 20 for PGD to provide a fair comparison against accelerated algorithm MPGD.

In Figure 22 we present the hyper-parameters tuning where we chose the algorithm with the least iterations to converge over 10 independent trials and $T = 1000$ steps, namely $(\epsilon, \mu) = (0.3, 0.1)$ or equivalently $(\epsilon, \eta) = (0.3, 300)$.

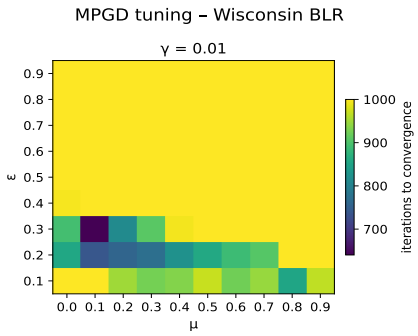


Figure 22: Heatmap Wisconsin Bayesian Logistic Regression Model γ vs μ,with the darker color giving the least iterations needed until convergence

F EXPERIMENTAL DETAILS

We report the runtime required to obtain the main results presented in the paper, averaged over five independent runs for each experiment. All experiments were conducted on a machine with a 12th Gen Intel(R) Core(TM) i7-12700H CPU @ 2.30 GHz and 32 GB of RAM.

Table 1: Average Runtime for each model's notebook

Notebook	**Time (s)**
Toy Hierarchical Model	497.09
Bayesian Logistic Regression	423.35
Bayesian Neural Network	578.71