# Improved Marginal Unbiased Score Expansion (MUSE) via Implicit Differentiation

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

We apply the technique of implicit differentiation to boost performance, reduce numerical error, and remove required user-tuning in the Marginal Unbiased Score Expansion (MUSE) algorithm for hierarchical Bayesian inference. We demonstrate these improvements on three representative inference problems: 1) an extended Neal's funnel 2) Bayesian neural networks, and 3) probabilistic principal component analysis. On our particular test cases, MUSE with implicit differentiation is faster than Hamiltonian Monte Carlo by factors of 155, 397, and 5, respectively, or factors of 65, 278, and 1 without implicit differentiation, and yields good approximate marginal posteriors. The Julia and Python MUSE packages have been updated to use implicit differentiation, and can solve problems defined by hand or with any of a number of popular probabilistic programming languages and automatic differentiation backends.

## 1 Introduction

MUSE is an algorithm for fast approximate hierarchical Bayesian inference, recently proposed by [1, 2]. The user denotes some subset of model parameters as the "parameters of interest," and the algorithm will approximate their marginal posterior while integrating out remaining "latent" parameters. MUSE is efficient for very high-dimensional latent spaces and can often provide near-exact inference at orders of magnitude lower computational cost than other methods such as Hamiltonian Monte Carlo (HMC) or variational inference (VI) [2].

The requirements for using MUSE on a given problem are that 1) samples can be generated from the prior and 2) gradients of the joint posterior probability distribution can be calculated. The latter requirement is the same as for HMC, VI, and many other tools. The former requirement is not strictly a requirement for some of these, but is generally even easier. All problems defined via a probabilistic programming language satisfy the requirements automatically. Owing to its reliance on prior samples, MUSE can be considered a form of simulation-based inference, extended to use readily available joint posterior gradients, similar to the proposal by [3].

At its core, MUSE is based on an approximation to the marginal score formed from solutions to a series of optimization problems. As part of the algorithm, we must compute derivatives of these solutions, and, in this work, we improve MUSE by making use of implicit differentiation (ID) to perform this calculation. While ID is not a new development, it has recently been shown to be particularly powerful in conjunction with automatic differentiation (AD) [4, 5]. We follow this approach, and demonstrate that it leads to significant improvements in both speed and usability for MUSE, strengthening its case as a generic inference tool.

Under review at the NeurIPS 2022 Workshop on Score-Based Methods. Do not distribute.

## 2 Summary of the MUSE method

Here we give a brief and practical summary of MUSE to help understand where ID fits in (for a comprehensive introduction, see [2]). MUSE is applicable to inference problems where the posterior probability of some parameters of interest, $\theta$, given data, $x$, requires marginalization over a high-dimensional latent space parameterized by $z$,

$$\mathcal{P}(\theta \,|\, x) = \int d^n z \, \mathcal{P}(x, z \,|\, \theta) \, \mathcal{P}(\theta). \tag{1}$$

The algorithm provides a fast estimate of the marginal posterior mean and covariance, which is computed under an approximation to the integral over $z$. This approximation involves solving a series of optimization problems wherein we maximize the joint likelihood, $\mathcal{P}(x, z \,|\, \theta)$, over the latent parameters $z$, given fixed $x$ and $\theta$,

$$\hat{z}(\theta, x) \equiv \underset{z}{\text{argmax}} \; \log \mathcal{P}(x, z \,|\, \theta). \tag{2}$$

These correspond to maximum a posteriori (MAP) estimates of $z$, and they are used to define the score at the MAP,

$$s_i^{\text{MAP}}(\theta, x) \equiv \frac{d}{d\theta_i} \log \mathcal{P}(x, \hat{z}(\theta, x) \,|\, \theta). \tag{3}$$

The MUSE estimate of the posterior mean, $\bar{\theta}$, is then implicitly defined as the solution to

$$s_i^{\text{MAP}}(\bar{\theta}, x) = \Big\langle s_i^{\text{MAP}}(\bar{\theta}, x) \Big\rangle_{x \sim \mathcal{P}(x \,|\, \bar{\theta})}, \tag{4}$$

and the posterior covariance is $\Sigma = H^{-1} J H^{-\dagger}$, with

$$J_{ij} = \Big\langle s_i^{\text{MAP}}(\bar{\theta}, x) \, s_j^{\text{MAP}}(\bar{\theta}, x) \Big\rangle_{x \sim \mathcal{P}(x \,|\, \bar{\theta})} - \Big\langle s_i^{\text{MAP}}(\bar{\theta}, x) \Big\rangle \Big\langle s_j^{\text{MAP}}(\bar{\theta}, x) \Big\rangle_{x \sim \mathcal{P}(x \,|\, \bar{\theta})} \tag{5}$$

$$H_{ij} = \frac{d}{d\theta_j} \left[ \Big\langle s_i^{\text{MAP}}(\bar{\theta}, x) \Big\rangle_{x \sim \mathcal{P}(x \,|\, \theta)} \right] \Bigg|_{\theta = \bar{\theta}}. \tag{6}$$

This definition gives MUSE a number of useful properties (see [2] for proofs): 1) it is an asymptotically unbiased estimate of $\theta$ irregardless of any non-Gaussianity in the likelihood, 2) it is asymptotically optimal for a Gaussian likelihood, where it becomes equivalent to the marginal maximum likelihood estimate and the covariance becomes the inverse Fisher information matrix, 3) no dense operators of the dimensionality of $z$ ever need to be computed, meaning it is well-suited for high-dimensional problems and 4) it requires few tuning parameters, setting it apart from HMC, VI, or many other simulation-based inference methods, which need user-provided mass matrices, surrogate distributions, or neural network architectures to work or to achieve optimal performance on complicated latent spaces. MUSE is approximate, so it does not aim to generically replace exact algorithms like HMC, but in many cases, its speed and aforementioned properties make it a very advantageous alternative.

In practice, the optimization problem in Eq. (2) is performed with LBFGS using user-provided or AD gradients. An existing challenge for MUSE is that naively computing Eq. (6) with AD would require propagating second-order derivatives through the optimizer, since a chain rule term involving $d\hat{z}/d\theta$ arises. With few or no AD libraries robustly supporting second-order AD through an optimizer, we have previously resorted to computing this term with finite differences (FD). This has not been completely prohibitive as FD are needed only over the low-dimensional $\theta$ and not over the high-dimensional $z$, so the solution remains tractable despite a linear computational scaling with the dimensionality of $\theta$. However, it requires tuning the FD step size for each dimension of $\theta$, and can at times incur large numerical errors. The main development of this paper is to demonstrate that this term can instead be computed more simply and exactly with ID.

## 3 Using implicit differentiation

To compute $H$ with ID, first note that Eq. (6) can be written as $H_{ij} = \frac{1}{N} \sum_{\alpha=1}^{N} h_{ij}(\Omega_\alpha)$, where $\Omega_\alpha$ are some independent random states, and

$$h_{ij}(\Omega) = \frac{d}{d\theta'_j} \frac{d}{d\theta_i} \log \mathcal{P}\Big( x\big(\Omega, \theta'\big), \hat{z}\big(x(\Omega, \theta'), \bar{\theta}\big) \,\Big|\, \theta \Big) \Bigg|_{\theta = \theta' = \bar{\theta}}. \tag{7}$$

Here, we consider a single realization of $x$ as dependent on $\theta$ in the sense that any simulated $x$ can be written as a deterministic function of $\theta$ and a random state (think of $\Omega$ as the machine's pseudo random number generator). Expanding the chain rule once and omitting $\Omega$ and the final evaluation at $\bar{\theta}$ for brevity yields

$$\frac{d}{d\theta'_j}\frac{d}{d\theta_i}\log\mathcal{P}\Big(x\big(\theta'\big), \hat{z}\big(x(\bar{\theta}), \bar{\theta}\big)\,\Big|\,\theta\Big) \;+\; \frac{d}{dz_n}\frac{d}{d\theta_i}\log\mathcal{P}\Big(x\big(\bar{\theta}\big), z\,\Big|\,\theta\Big)\Big|_{z=\hat{z}}\frac{d\hat{z}_n\big(x(\theta'), \bar{\theta}\big)}{d\theta'_j}. \tag{8}$$

The first term can be computed with second-order AD through the likelihood and through the prior samples of $x$. In practice, this means simply using the same random state on the forwards and/or backwards AD passes and otherwise considering random number generation constant (this is the default in most AD libraries). The second term, where we will use ID, involves a derivative of the MAP solution, $d\hat{z}/d\theta$. The MAP solution by definition obeys

$$\frac{d}{dz}\log\mathcal{P}\Big(x\big(\theta'\big), z\,\Big|\,\bar{\theta}\Big)\Big|_{z=\hat{z}(x(\theta'),\bar{\theta})} = 0. \tag{9}$$

Taking a $\theta'$ derivative of this equation and solving the resulting equation for $d\hat{z}/d\theta'$ yields

$$\frac{d\hat{z}_n}{d\theta'_j} = \left[\frac{d^2}{dz_m dz_n}\log\mathcal{P}\Big(x\big(\bar{\theta}\big), z\,\Big|\,\bar{\theta}\Big)\right]^{-1}\frac{d}{d\theta'_j}\frac{d}{dz_m}\log\mathcal{P}\Big(x\big(\theta'\big), z\,\Big|\,\bar{\theta}\Big)\Big|_{z=\hat{z}}. \tag{10}$$

This quantity now only requires derivatives through the likelihood rather than through an optimizer; in fact, it is independent of the particular optimizer used to obtain $\hat{z}$. Computing it involves solving a linear problem with the same dimensionality as $z$. Because $z$ is assumed high-dimensional where forming an explicit matrix is impossible, we solve the system iteratively, with the action of the quantity in brackets above given by a jacobian-vector product. Note, however, that the linear operator is symmetric since it is a Hessian, and, by definition if the MAP exists (which is a requirement for MUSE anyway), it is positive definite. Thus, we can use an efficient conjugate gradient solver which exploits this structure, as opposed to generic linear solvers which must be used in more general ID problems.

# 4 Results

We compare HMC and MUSE with or without ID on three representative inference problems:

**Funnel problem** We consider an embedding of several Neal's funnels into a toy hierarchical problem [6, 2]. The model is:

$$\theta_i \sim \mathrm{Normal}(0, 3) \qquad z_{ij} \sim \mathrm{Normal}(0, \exp(\theta_i/2)) \qquad x_{ij} \sim \mathrm{Normal}(\tanh(z_{ij}), 1) \tag{11}$$

with $i \in 1{:}10$ parameters and $j \in 1{:}500$ latent dimensions per parameter. Although the embedded funnels are independent, for demonstration, we solve the entire problem as one large system when running either NUTS or MUSE.

**Bayesian Neural Network** Following the example given in [7], we consider a Bayesian neural network (BNN) analysis, where we interpolate some noisy one-dimensional data with a three-layer neural network. The model is:

$$\begin{aligned} \sigma_i &\sim \mathrm{LogNormal}(0, 1) & W_i &\sim \mathrm{Normal}(0, \sigma_i) \\ \tau &\sim \mathrm{Gamma}(3, 1) & Y_j &\sim \mathrm{Normal}(\mathrm{NN}(W_i), 1/\tau) \end{aligned} \tag{12}$$

where $i \in 1{:}3$ layers, the layer weights, $W_i$, which parameterize the network, NN, contain 45 latent dimensions and map the data coordinates to 5 hidden units and finally to the data space, and the data $Y_j$ consists of $j \in 1{:}500$ data points. The goal is to infer the $\sigma_i$ and $\tau$. We note that each internal optimization solution in Eq. (2) involves training the network given some prior on the weights. For this simple example we use our standard LBFGS solver, but other more machine learning oriented solvers can readily be used for the internal MUSE optimization step as well.

**Probabilistic Principal Component Analysis** Finally, we consider a probabilistic principal component analysis (PPCA) with automatic relevance determination [8]. The model is:

$$\begin{aligned} \alpha_i &\sim \mathrm{InverseGamma}(1, 1) & W_{ki} &\sim \mathrm{Normal}(0, \sqrt{\alpha}_i) \\ Z_{ij} &\sim \mathrm{Normal}(0, 1) & X_{kjl} &\sim \mathrm{Normal}(W_{ki} Z_{ij}, 1) \end{aligned} \tag{13}$$

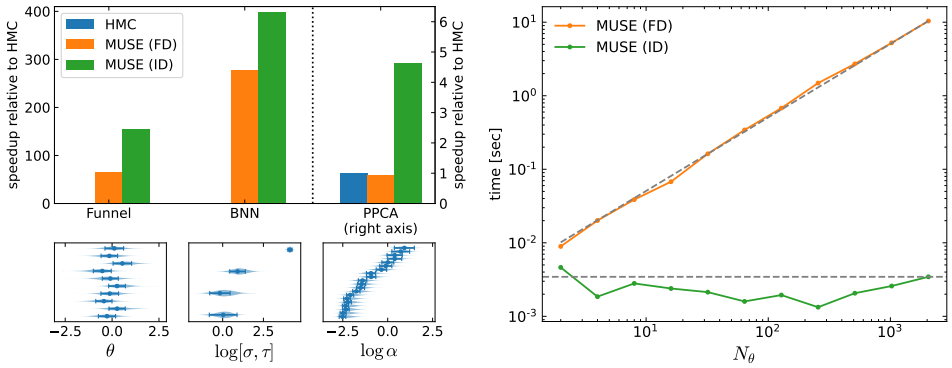


Figure 1: *(Top left)* Speedups which are possible with MUSE both with or without ID as compared to HMC on a variety of hierarchical Bayesian inference problems (described in Sec. 4). *(Bottom left)* HMC posteriors as violin plots, compared to MUSE results as error bars. *(Right)* Empirical check of the asymptotical scaling of the $H$ computation with FD or ID.

with $i \in 1{:}10$ principal components, $j, k \in 1{:}100$ observations, and $l \in 5$ batches. The goal is to find the largest principal component amplitudes, $\alpha$, given observations of $X$, while marginalizing over the entries in the $Z$ and $W$ matrices.

Our benchmarks compare the number of posterior gradient evaluations needed such that for all parameters of interest, we reach 1) a 10% error on the mean relative to the standard deviation and 2) a 10% relative error on the standard deviation. Given a Gaussian sampling distribution, these criteria impose the same constraint. For HMC, this corresponds to achieving an effective sample size of 100 for all parameters. We use NumPyro to implement each model [9, 10], and sample with NumPyro's NUTS implementation with default parameters. For MUSE this corresponds to running MUSE with 100 simulations and setting the $\theta$ tolerance to 10%. We use the existing Jax [11] MUSE implementation to run MUSE on the same NumPyro model.

The results are summarized in the left panels in Fig. 1. We see that for each of the three problems, ID outperforms the previous FD approach by as much as a factor of 5. In all cases, MUSE with ID significantly outperforms HMC, including by a factor of 391 in the most dramatic case (the BNN). The bottom panel shows a comparison of the inferred values of the parameters of interest, confirming the quality of the MUSE approximation.

We also expect a more favorable computational scaling for ID over FD as we increase the dimensionality of $\theta$. This is because computing $h_{ij}$ with FD requires perturbing each element of $\theta$ and recomputing a MAP each time, whereas ID requires just one MAP that is then used evaluating all terms in Eqn. (8), with the tradeoff of also needing to solve a linear problem. To confirm this tradeoff is beneficial, we modify our funnel problem to increase the dimensionality of $\theta$ (while keeping the latent dimensionality the same), and plot resulting timings for the $H$ computation in the right panel of Fig. 1. We find that FD scales linearly with the dimensionality of $\theta$ as expected, but that ID is nearly constant, meaning the cost of the linear solver is subdominant. For the configurations considered, we reach multiple orders of magnitude speedups over FD.

## 5 Conclusions

In this work, we have shown that ID makes the MUSE algorithm faster and removes reliance on numerically-noisy FD. It requires second-order derivatives through the joint likelihood, but not through any optimizer, and never fully with respect to the latent space, meaning MUSE with ID is still well-suited for very high-dimensional problems. We have also provided examples of MUSE applied to BNNs and PPCA, demonstrating the extended applicability of the algorithm, which had previously been tested only on simpler toy problems or more complex but less general problems in cosmology [12, 2]. Beyond speed and accuracy improvements, removing the need to verify or tweak FD step-sizes represents a significant usability advancement for the algorithm.

# References


[1] Uros Seljak, Grigor Aslanyan, Yu Feng, and Chirag Modi. Towards optimal extraction of cosmological information from nonlinear data. *Journal of Cosmology and Astroparticle Physics*, 2017(12):009–009, December 2017.

[2] Marius Millea and Uroš Seljak. Marginal unbiased score expansion and application to CMB lensing. *Physical Review D*, 105:103531, May 2022.

[3] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, December 2020.

[4] D Duvenaud, J. Z. Kolter, and M. Johnson. Deep Implicit Layers: Neural ODEs, Equilibrium Models, and Differentiable Optimization. https://implicit-layers-tutorial.org.

[5] Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and Modular Implicit Differentiation, May 2022.

[6] Radford M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, June 2003.

[7] NumPyro Documentation: Bayesian Neural Network. https://num.pyro.ai/en/stable/examples/bnn.html.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[9] Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro, December 2019.

[10] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 20(28):1–6, 2019.

[11] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

[12] Benjamin Horowitz, Uros Seljak, and Grigor Aslanyan. Efficient Optimal Reconstruction of Linear Fields and Band-powers from Cosmological Data. *Journal of Cosmology and Astroparticle Physics*, 2019(10):035–035, October 2019.