# flowMC: Normalizing flow enhanced sampling package for probabilistic inference in JAX

Kaze Wong, Marylou Gabrié, Daniel Foreman-Mackey

The Journal of Open Source Software

# flowMC: Normalizing flow enhanced sampling package for probabilistic inference in JAX

**Kaze W. K. Wong** [1], **Marylou Gabrié** [2,3], **and Daniel Foreman-Mackey** [1]

**1** Center for Computational Astrophysics, Flatiron Institute, New York, NY 10010, US **2** École Polytechnique, Palaiseau 91120, France **3** Center for Computational Mathematics, Flatiron Institute, New York, NY 10010, US

**DOI:** 10.21105/joss.05021

**Software**

- Review
- Repository
- Archive

**Editor:** Rachel Kurchin

**Reviewers:**

- @matt-graham
- @Daniel-Dodd

**Submitted:** 10 November 2022
**Published:** 09 March 2023



## Summary

Across scientific fields, the Bayesian framework is used to account for uncertainties in inference (Ellison, 2004; Lancaster, 2004; Von Toussaint, 2011). However, for models with more than a few parameters, exact inference is intractable. A frequently used strategy is to approximately sample the posterior distribution on a model's parameters with a Markov chain Monte Carlo (MCMC) method. Yet conventional MCMC methods relying on local updates can take a prohibitive time to converge when posterior distributions have complex geometries (see e.g., Rubinstein & Kroese (2017)).

`flowMC` is a Python library implementing accelerated MCMC leveraging deep generative modelling as proposed by Gabrié et al. (2022), built on top of the machine learning libraries JAX (Bradbury et al., 2018) and Flax (Heek et al., 2020). At its core, `flowMC` uses a combination of Metropolis-Hastings Markov kernels using local and global proposed moves. While multiple chains are run using local-update Markov kernels to generate approximate samples over the region of interest in the target parameter space, these samples are used to train a normalizing flow (NF) model to approximate the samples' density. The NF is then used in an independent Metropolis-Hastings kernel to propose global jumps across the parameter space. The `flowMC` sampler can handle non-trivial geometry, such as multimodal distributions and distributions with local correlations.

The key features of `flowMC` are summarized in the following list:

- Since `flowMC` is built on top of JAX, it supports gradient-based samplers through automatic differentiation such as the Metropolis-adjusted Langevin algorithm (MALA) and Hamiltonian Monte Carlo (HMC).
- `flowMC` uses state-of-the-art NF models such as rational quadratic splines (RQS) to power its global proposals. These models are efficient in capturing important features within a relatively short training time.
- Use of accelerators such as graphics processing units (GPUs) and tensor processing units (TPUs) are natively supported. The code also supports the use of multiple accelerators with SIMD parallelism.
- By default, just-in-time (JIT) compilations are used to further accelerate the sampling process.
- We provide a simple black-box interface for the users who want to use `flowMC` by its default parameters, as well as an extensive guide explaining trade-offs while tuning the sampler parameters.

The tight integration of all the above features makes `flowMC` a highly performant yet simple-to-use package for statistical inference.

## Statement of need

Bayesian inference requires computing expectations with respect to a posterior distribution on parameters $\theta$ after collecting observations $\mathcal{D}$. This posterior is given by

$$p(\theta|\mathcal{D}) = \frac{\ell(\mathcal{D}|\theta)p_0(\theta)}{Z(\mathcal{D})},$$

where $\ell(\mathcal{D}|\theta)$ is the likelihood induced by the model, $p_0(\theta)$ the prior on the parameters and $Z(\mathcal{D})$ the model evidence. For parameter space with more than a few dimensions, it is necessary to resort to a robust sampling strategy such as MCMC. Drastic gains in computational efficiency can be obtained by a careful selection of the MCMC transition kernel, which can be assisted by machine learning methods and libraries.

***Gradient-based sampler*** In a high dimensional space, sampling methods which leverage gradient information of the target distribution aid by proposing new samples likely to be accepted. `flowMC` supports gradient-based samplers such as MALA and HMC through automatic differentiation with JAX.

***Learned transition kernels with NFs*** When the posterior distribution has a non-trivial geometry, such as multiple modes or spatially dependent correlation structures (e.g, (Neal, 2003)), samplers based on local updates are inefficient. To address this problem, `flowMC` also uses a generative model, namely a NF (Kobyzev et al., 2021; Papamakarios et al., 2021), that is trained to mimic the posterior distribution and used as a proposal in Metropolis-Hastings MCMC steps. Variants of this idea have been explored in the past few years (Albergo et al., 2019; Hoffman et al., 2019; e.g., Parno & Marzouk, 2018). Despite the growing interest in these methods, few accessible implementations for practitioners exist, especially with GPU and TPU support. Notably, a version of the NeuTra sampler (Hoffman et al., 2019) is available in Pyro (Bingham et al., 2019), and the PocoMC package (Karamanis et al., 2022) implements a version of sequential Monte Carlo (SMC), including NFs.

`flowMC` implements the method proposed by Gabrié et al. (2022). As individual chains explore their local neighborhood through gradient-based MCMC steps, multiple chains train the NF to learn the global landscape of the posterior distribution. In turn, the chains can be propagated with a Metropolis-Hastings kernel using the NF to propose globally in the parameter space. The cycle of local sampling, NF tuning, and global sampling is repeated until chains of the desired length are obtained. The entire algorithm belongs to the class of adaptive MCMC methods (Andrieu & Thoms, 2008), collecting information from the chains' previous steps to simultaneously improve the transition kernel. Usual MCMC diagnostics can be applied to assess the robustness of the inference results, thereby avoiding the common concern of validating the NF model. If further sampling from the posterior is necessary, the flow trained during a previous run can be reused without further training. The mathematical details of the method are explained in (Gabrié et al., 2021, 2022).

***Use of accelerators*** `flowMC` is built on top of JAX, which supports the use of GPU and TPU accelerators by default. Users can write code the same way as they would on a CPU, and the library will automatically detect the available accelerators and use them at run time. Furthermore, the library leverages JIT compilations to further improve the performance of the sampler.

***Simplicity and extensibility*** We provide a black-box interface with a few tuning parameters for users who intend to use `flowMC` without too much customization on the sampler side. The only inputs we require from the users are (1) a function to evaluate the logarithm of the (unnormalized) density of the posterior distribution of interest and (2) the initial position of the chains. On top of the black-box interface, the package offers automatic tuning for the local samplers to reduce the number of hyperparameters users need to manage.

While we provide a high-level interface suitable for most practitioners, the code is also designed to be extensible. Researchers with knowledge of more appropriate local and/or global sampling kernels for their application can integrate the kernels in the sampler module.

## Acknowledgements

M.G. acknowledges funding from Hi! PARIS.

## References

Albergo, M. S., Kanwar, G., & Shanahan, P. E. (2019). Flow-based generative models for Markov chain Monte Carlo in lattice field theory. *Physical Review D*, *100*(3), 034515. https://doi.org/10.1103/PhysRevD.100.034515

Andrieu, C., & Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, *18*(4), 343–373. https://doi.org/10.1007/s11222-008-9110-y

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P. A., Horsfall, P., & Goodman, N. D. (2019). Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, *20*, 28:1–28:6. http://jmlr.org/papers/v20/18-403.html

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/google/jax

Ellison, A. M. (2004). Bayesian inference in ecology. *Ecology Letters*, *7*(6), 509–520.

Gabrié, M., Rotskoff, G. M., & Vanden-Eijnden, E. (2021). Efficient Bayesian sampling using normalizing flows to assist Markov chain Monte Carlo methods. *Invertible Neural Networks, NormalizingFlows, and Explicit Likelihood Models (ICML Workshop)*. https://arxiv.org/abs/2107.08001

Gabrié, M., Rotskoff, G. M., & Vanden-Eijnden, E. (2022). Adaptive Monte Carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, *119*(10). https://doi.org/10.1073/pnas.2109420119

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., & Zee, M. van. (2020). *Flax: A neural network library and ecosystem for JAX* (Version 0.6.3). http://github.com/google/flax

Hoffman, M. D., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., & Vasudevan, S. (2019). NeuTra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport. *1st Symposium on Advances in Approximate Bayesian Inference, 2018 1–5*. http://arxiv.org/abs/1903.03704

Karamanis, M., Beutler, F., Peacock, J. A., Nabergoj, D., & Seljak, U. (2022). Accelerating astronomical and cosmological inference with preconditioned Monte Carlo. *Monthly Notices of the Royal Astronomical Society*, *516*(2), 1644–1653. https://doi.org/10.1093/mnras/stac2272

Kobyzev, I., Prince, S. J. D., & Brubaker, M. A. (2021). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(11), 3964–3979. https://doi.org/10.1109/TPAMI.2020.2992934

Lancaster, T. (2004). *An introduction to modern bayesian econometrics*. Blackwell Oxford.

Neal, R. M. (2003). Slice sampling. *The Annals of Statistics*, *31*(3), 705–767.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, *22*(57), 1–64. https://jmlr.org/papers/v22/19-1028.html

Parno, M. D., & Marzouk, Y. M. (2018). Transport map accelerated Markov chain Monte Carlo. *SIAM-ASA Journal on Uncertainty Quantification*, *6*, 645–682. https://doi.org/10.1137/17M1134640

Rubinstein, R. Y., & Kroese, D. P. (2017). *Simulation and the Monte Carlo method* (Third edition). John Wiley & Sons, Inc. ISBN: 978-1-118-63238-3

Von Toussaint, U. (2011). Bayesian inference in physics. *Reviews of Modern Physics*, *83*(3), 943.