Addressing sampling problems with adaptive MCMC and normalizing flows

Thibeau Wouters

March 19, 2025

Contents

1 Introduction

In science, we are often interested in understanding the parameters of a certain model after gathering data from experiments. One such example are gravitational waves (GWs), ripples in the fabric of space-time induced by the coalescences of heavy, compact objects like black holes or neutron stars.

When a model has parameters θ and we have a dataset d at our disposal to inform these parameters, we are interested in the posterior distribution of these parameters, $p(\theta|d)$. These distributions are analytically intractable, and therefore, we have to rely on stochastic sampling methods to obtain representational samples from it.

However, this is a computationally expensive and complicated problem in GW analyses, for which many researchers are currently trying to devise a solution. In the field of GWs, this

computational cost will increase once we have more advanced detectors in the next decades, such as the Einstein Telescope (ET) and Cosmic Explorer (CE). This is often referred to as the third-generation (3G) of gravitational wave detectors. To get a sense of the problem, have a look at Ref. [1], where the abstract for instance quotes that analyzing one month of observations made with a network of ET and CE, even with the latest speed-up methods used in the field, will require millions of CPU hours – an enormous economic cost with a high carbon footprint.

We aim to solve this data analysis problems with a more recently developed framework, called Jim. This package uses a more advanced Markov chain Monte Carlo (MCMC) sampler, called flowMC, using a few key ingredients that accelerate the process. First, flowMC uses jax [2], a software package developed by Google that supports just-in-time (JIT) compilation, automatic differentiation, vectorizing capabilities and allows the code to run on graphical processing units (GPUs). These components already have the potential to massively accelerate code execution. Moreover, in order to make the sampling process itself more efficient, a normalizing flow (NF) is trained on-the-fly to approximate the target distribution. Normalizing flows are a class of machine learning methods that are well-suited to approximate probability distributions. Once trained, their probability density can easily be evaluated, and samples from the NF distribution can easily be generated, no matter how complex the distribution looks like. By leveraging these NFs as proposal distributions, the convergence time required by the MCMC process is reduced, the mixing of the chains is improved, and more effective samples of the target distribution can be obtained in a shorter amount of runtime. The idea is summarized by Fig 1. Using this package, it has already been shown that analyzing the GW signals as observed by current detectors can be done in minutes [3, 4].

2 The problem

So, what exactly makes the problem so hard in GW analyses, and how will this change for future GW detectors?

Figure 2 shows a so-called cornerplot of the posterior distributions of the source parameters of the first binary neutron star merger, GW170817. A cornerplot is a way to visualize a high-dimensional distribution[1] by showing the 2D marginalized distributions in the middle panels, while showing the 1D marginals in the top panels, for each parameter. The plot shows two different samplers used: bilby, which uses nested sampling [5], and jim as explained above.

Pay attention to the shape of the posterior. As you can see in some of the 2D panels: the posterior is highly complicated. For instance, there are multiple modes and large correlations.

[1] Here, we are talking about 13 source parameters in total – the plot shows a few interesting combinations of them.

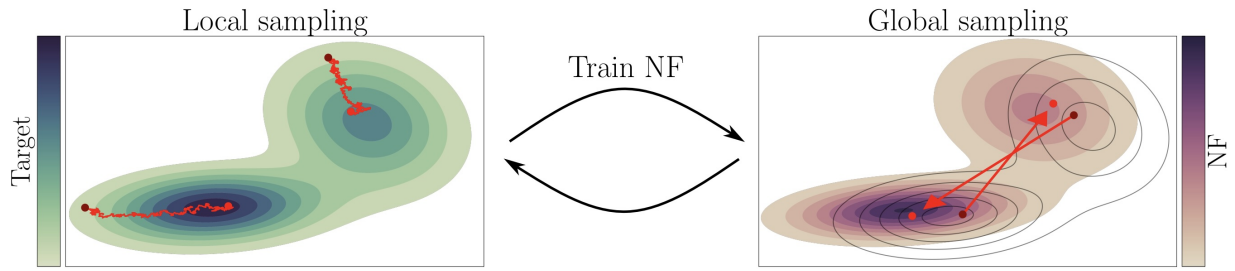

Figure 1: Schematic overview of the training loop of the FLOWMC sampler. Each loop starts with running the local sampler. For the local sampling, we use the MALA algorithm, which exploits the gradient of the target distribution (green and gray lines) to evolve the Markov chains (red). With the samples obtained from the local sampler, a normalizing flow (NF) is trained to approximate the distribution of the Markov chains. During the global sampling phase, we use the density learned by the NF (purple) as a proposal. We accept or reject the proposed samples (red) with a Metropolis-Hastings step, which relies on both the proposal density as well as the target density. This local-global procedure is repeated until the NF has converged. Afterward, we perform a fixed number of production loops, where the weights of the NF are frozen and the local and global sampler output the final production samples.

Sampling such features with MCMC can be tedious and require a long time before the chains fully explore such posterior landscapes. This will only worsen with more advanced detectors, since the posteriors will get more narrowly peaked, making it even more difficult to sample.

In this thesis, we are going to look at toy examples that serve as a proxy for such complicated distributions and assess the performance of FLOWMC, and potentially other samplers as well, in recovering the posterior distributions. By looking at such proxy examples, we have control over the ground truth and can cook up metrics to check how close the obtained result is to the end result (for instance: are all the modes well recovered, with the correct weight? et cetera). Afterward, we will consider the interplay between various components of the FLOWMC sampler and how to find the optimal settings for a given problem. Moreover, we will consider new tools, borrowed from other sampler codes, to improve the exploration of the posterior landscape and improve convergence.

We will discuss specific projects in more detail in the first few weeks, and brainstorm together to define the most interesting and feasible subprojects to consider attacking. However, I highly encourage you to explore on your own (skim a few papers, read a few sampler documentation packages, think about possible questions on how to improve certain aspects of the sampler, etc) – coming up with your own research question (i) is the most challenging but therefore also rewarding aspect of research, (ii) ensures you are maximally invested in the thesis, (iii) might enable new (perhaps better!) questions to come up that I might not have thought of myself before! :)

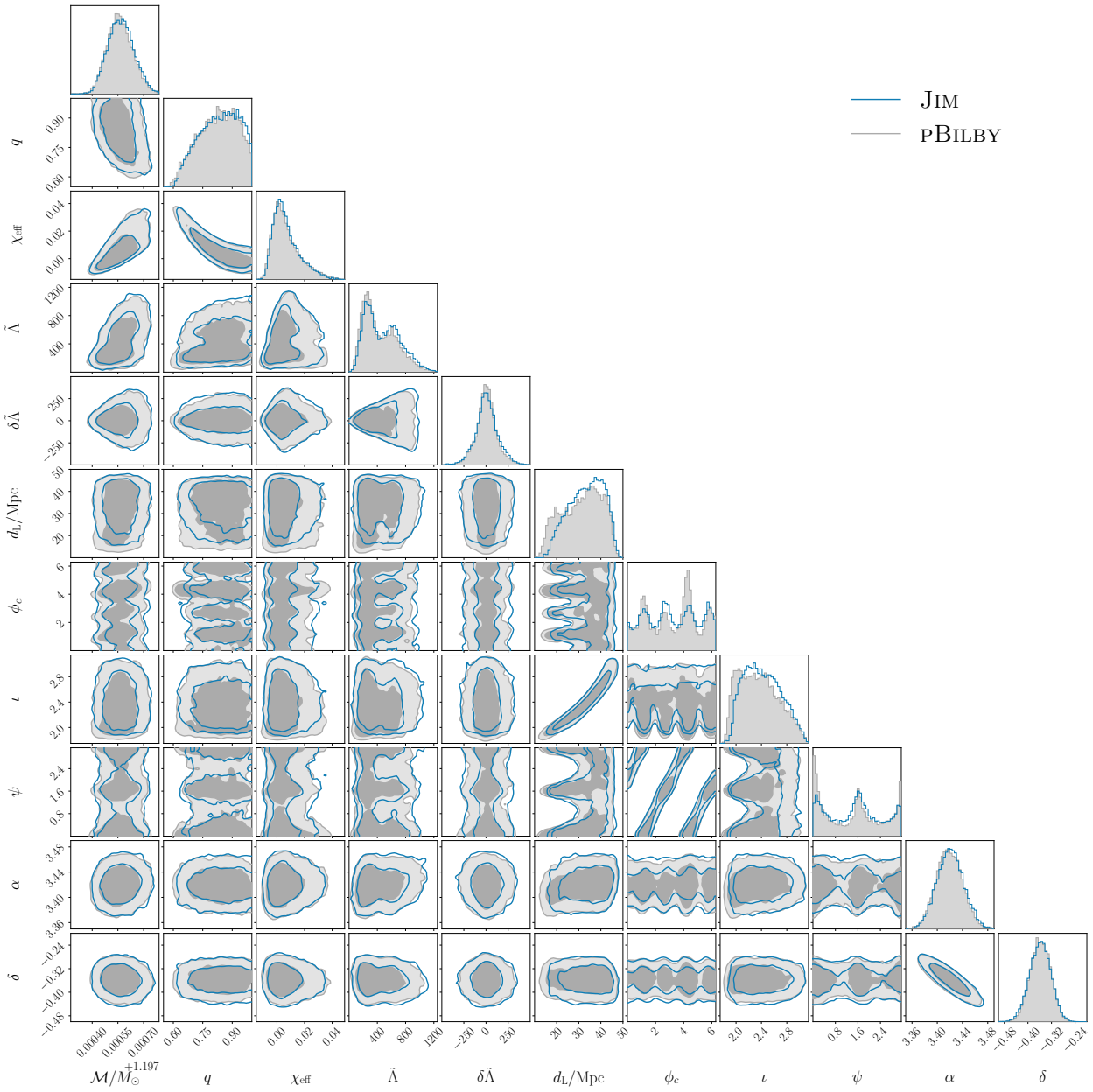

Figure 2: Posterior distribution of the GW170817 event.

3 Recommended sources for further reading

Have a look at the above-mentioned papers a bit. Don't read all of them in the same level of detail, but try to focus on the core problem we will tackle: how to make samplers more efficient, what are the challenges faced,... Therefore, feel free to skip all the GW details as digesting this will require a whole different set of skills.

Here are some more references to get a broader overview.

There are two "schools" to do sampling that are commonly used in our community: nested sampling [5] (a common package used is bilby [6, 7]) and MCMC. For books, I recommend "Data analysis: a Bayesian tutorial" by Sivia and Skilling and "Handbook of Markov Chain Monte Carlo" by Brooks *et al.*

To learn more about our methods used, refer to Ref. [2] to understand how JAX works. Although of course the best way to get to know the software is by using it – have a look at the documentation, for instance. The ideas behind normalizing flow-enhanced MCMC sampling itself are explained in Ref. [8], with the JOSS paper of the package in Ref. [9]. The

For more information regarding deep learning, see the book by Goodfellow *et al.* [10]. A broader introduction to machine learning in general can be found in the book by Murphy [11]. Advanced topics are discussed in a second book by Murphy [12]. Extensive background in probability theory and information theory underpinning machine learning can be read in the book by Mackay [13].

Normalizing flows are well explained by Ref. [14].

Interesting software packages to read about samplers and understand them/get inspiration for the project (some of their documentation pages give good sampling problems and examples how to solve them and can be interesting to read).

- FLOWMC: https://github.com/kazewong/flowMC
- BLACKJAX: https://github.com/blackjax-devs/blackjax
- BAYEUX: https://github.com/jax-ml/bayeux

4 Thesis writing

I recommend to start as early as possible with thesis writing, even if it is not the final version. For instance, make notes while reading the above papers or skimming the software package documentation. What are some common problems you see popping up? What kind

of sampler methods exist out there? What are some proposed solutions? Making notes on these topics while you are reading them (i) makes sure you do not forget these things (as we often underestimate how quickly we forget where exactly we saw a method/problem appear, or where an interesting discussion appeared,... and I talk from experience!), (ii) helps you define the research question of the thesis and the literature overview that will eventually have to be done. In the early stages, just keep writing and do not hold back, you can always edit later on, or take the current notes as a starting point and start from scratch with them!

References

[1] Qian Hu and John Veitch. "Costs of Bayesian Parameter Estimation in Third-Generation Gravitational Wave Detectors: a Review of Acceleration Methods". In: (Dec. 2024). arXiv: 2412.02651 [gr-qc].

[2] Roy Frostig, Matthew James Johnson, and Chris Leary. "Compiling machine learning programs via high-level tracing". In: *Systems for Machine Learning* 4.9 (2018).

[3] Kaze W. K. Wong, Maximiliano Isi, and Thomas D. P. Edwards. "Fast Gravitational-wave Parameter Estimation without Compromises". In: *Astrophys. J.* 958.2 (2023), p. 129. DOI: 10.3847/1538-4357/acf5cd. arXiv: 2302.05333 [astro-ph.IM].

[4] Thibeau Wouters et al. "Robust parameter estimation within minutes on gravitational wave signals from binary neutron star inspirals". In: *Phys. Rev. D* 110.8 (2024), p. 083033. DOI: 10.1103/PhysRevD.110.083033. arXiv: 2404.11397 [astro-ph.IM].

[5] John Skilling. "Nested sampling for general Bayesian computation". In: *Bayesian Analysis* 1.4 (2006), pp. 833–859. DOI: 10.1214/06-BA127.

[6] I. M. Romero-Shaw et al. "Bayesian inference for compact binary coalescences with bilby: validation and application to the first LIGO–Virgo gravitational-wave transient catalogue". In: *Mon. Not. Roy. Astron. Soc.* 499.3 (2020), pp. 3295–3319. DOI: 10.1093/mnras/staa2850. arXiv: 2006.00714 [astro-ph.IM].

[7] Gregory Ashton et al. "BILBY: A user-friendly Bayesian inference library for gravitational-wave astronomy". In: *Astrophys. J. Suppl.* 241.2 (2019), p. 27. DOI: 10.3847/1538-4365/ab06fc. arXiv: 1811.02042 [astro-ph.IM].

[8] Marylou Gabrié, Grant M. Rotskoff, and Eric Vanden-Eijnden. "Adaptive Monte Carlo augmented with normalizing flows". In: *Proc. Nat. Acad. Sci.* 119.10 (2022), e2109420119. DOI: 10.1073/pnas.2109420119. arXiv: 2105.12603 [physics.data-an].

[9] Kaze W. k. Wong, Marylou Gabrié, and Daniel Foreman-Mackey. "flowMC: Normalizing flow enhanced sampling package for probabilistic inference in JAX". In: *J. Open Source Softw.* 8.83 (2023), p. 5021. DOI: 10.21105/joss.05021. arXiv: 2211.06397 [astro-ph.IM].

[10] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*. Vol. 1. MIT press Cambridge, MA, USA, 2017.

[11] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[12] Kevin P Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

[13] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[14] George Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". In: *J. Machine Learning Res.* 22.1 (2021), pp. 2617–2680. DOI: 10.5555/3546258.3546315. arXiv: 1912.02762 [stat.ML].