Addressing sampling problems with adaptive MCMC and normalizing flows

Thibeau Wouters

April 18, 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 – general viewpoint

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-

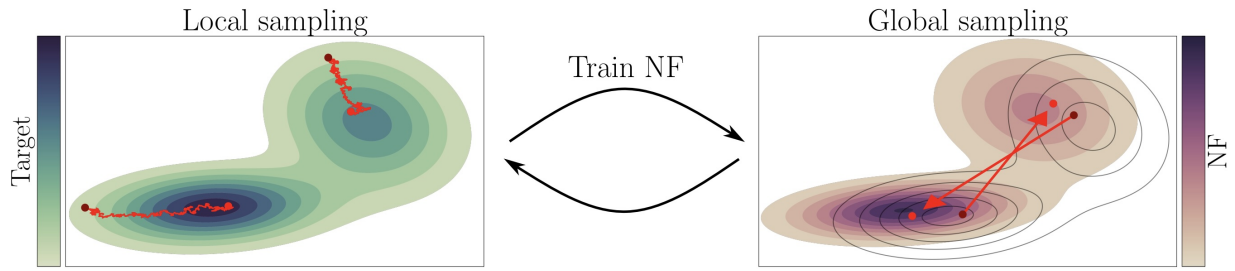

Figure 1: Schematic overview of the training loop of the FLOWMC sampler. Each loop starts with running the local sampler. For the local sampling, we use the MALA algorithm, which exploits the gradient of the target distribution (green and gray lines) to evolve the Markov chains (red). With the samples 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.

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. 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. Ideally (but mainly: "if time and research progress permit it"), we will then apply the newly discovered tools to actual gravitational wave problems!

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

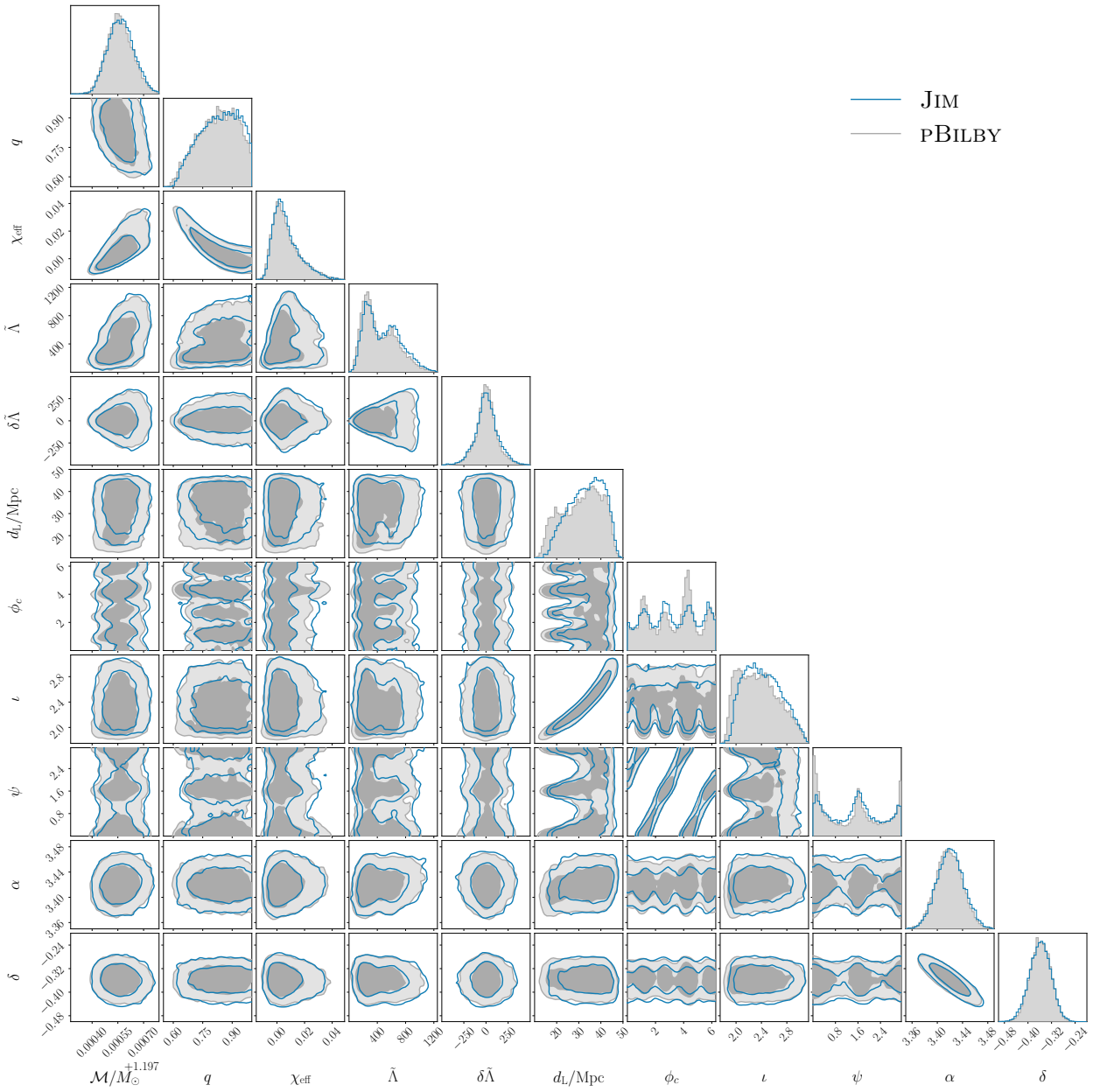

Figure 2: Posterior distribution of the GW170817 event.

3 Thesis project proposals

As explained above, our research group is making giant leaps to solve this problem by leveraging two important, new computational tools in the problem The first is to use gradients of the likelihood function, which we can do thanks to the automatic differentiation capabilities of JAX. For example, this enables us to use gradient-based samplers (common examples are MALA and HMC) which more efficiently explore the likelihood landscape. The second ingredient is normalizing flows: these are generative machine learning models that are ideally suited for approximating complicated probability distributions. In FLOWMC, they are trained on the fly from the MCMC samples and then used as proposal distributions.

In this thesis, we will think about the following research questions to investigate the efficacy of these two novel tools:

1. By using gradients of the likelihood, we can directly perform gradient-based optimization. Using this, we can therefore design a "cheap" algorithm (i.e., not doing the full sampling) that searches for the high-likelihood modes in the parameter space, and uses that prior knowledge to initialize the chains at interesting, high-likelihood locations. This way, we can potentially reduce the time needed to converge to these modes. A related application could be to create training data sampled from these modes for the normalizing flow and "pre-train" it with them – again the question arises: does this increase the usefulness of the normalizing flow from the start, therefore reducing the convergence time? This project therefore focuses more on the optimization aspect that is offered by JAX (which other sampling codes would *not* be able to offer). The exact implementation of such an algorithm is an open question and would be the main focus in this project. To get started, one idea would be to use evolutionary optimizers (see evosax as an example in JAX). However, this project would ideally require researching past literature on gradient-based optimization methods to discover local optima in high-dimensional spaces to see which methods would be well-suited for our use case. For a more general introduction on setting up and calling optimizers in JAX, I have an example notebook in a completely unrelated package here: please ignore the physics details and just focus on the JAX code to use optimizers to get started in this direction.

2. There are many architectures for normalizing flows: for overviews, see Refs. [6, 7]. While some of them are implemented in FLOWMC, other packages deliver additional flow architectures, such as flowjax. In this project, we start by adding support in flowCM to integrate with flowjax, so that users can choose from more flow architectures. In parallel, the student will deeply investigate the details and implementations of the various normalizing flows in the literature, and add new, interesting flow architectures (for instance, a more recent type of normalizing flow proposed is based on Bernstein polynomials [8]). Once these additional flows are coded up, and their trade-offs in computational cost and expressiveness are understood well, we assess their performance

in the context of flowMC: which flow architectures are more suitable to be used in concert with MCMC sampling?

The above projects would extend the existing methodology for flowMC. In particular, they are focusing on specific components in the paradigm of the concurrent MCMC sampling together with using normalizing flow as proposals in the sampler. However, several researchers are now wondering whether there would be other, approximate methods that are potentially even faster (to address the enormous computational costs of future GW detectors which will offer many detections – impossible to keep up with perhaps with Bayesian inference).

One interesting avenue left unexplored so far is variational inference. Variational inference does not make use of sampling (such as MCMC discussed above and used in flowMC). Rather, it posits a parametrized family of densities and then tries to "fit" those parameters in order to approximate the target distribution as well as possible. Here, by "fitting", we usually mean adjusting the parameters in such a way that the KL divergence (a "distance" measure between two probability distributions) is minimized. Therefore, it turns Bayesian inference from a sampling problem into an optimization problem.

Two common questions are often encountered here: (i) how do we choose the ideal family of densities?, and (ii) How do we perform the optimization? For the former, if the families are too simplistic, the final result is guaranteed to be a bad approximation. Say we have a target distribution such as the one in Figure 2. If we choose the parametric family to be a multivariate Gaussian (so that the parameters to optimize are its means and covariance matrix), then clearly this target density will never be a part of this family of densities, and the method will fail. Therefore, one needs flexible models, but they might be high-dimensional. This then ties into the second point: optimization is always hard in high-dimensionality, so how can we make sure it is efficient?

For the first point, perhaps you have already guessed that normalizing flows would be an ideal choice: they are expressive and can model all sorts of target densities! For the second point, of course, JAX is the way out, since it offers "cheap" gradients.

Combining these two then leads us to the field of *neural variational inference*, where we apply the ideas of variational inference on a normalizing flow. Not much research has been done in this direction in GW (or even outside of that field, if I do a few Google searches!), perhaps since it would really only become feasible due to the integration of normalizing flows and JAX, which are both quite novel in scientific computing.

There was a recent paper that used neural variational inference for the first time in the field of GWs [9], although this dealt with the problem of inferring the parameters of *populations* of GW sources, not the parameters of an individual source. Therefore, we still have to assess whether the method works for the latter case. Therefore, a logical research question is the following

3. Can we design a neural variational inference scheme for gravitational wave parameter estimation?

Of course, the first step in this direction would be to understand mainly the idea behind variational inference, and also why normalizing flows would be interesting here. For the former, I recommend the review article by Ref. [10]. For the latter, Ref. [9] is actually a good paper on this. The first step towards this scheme of course requires coding it up in JAX. Luckily, we have the advantage that Ref. [9] implemented its algorithm in JAX: have a look at this function in his Github package for the implementation (for now, don't obsess over its use of the 'bounds' argument, or the exact implementation of the likelihood function calling, since that will be different in our use case). This can serve as an ideal starting point for the code implementation. Also check (as mentioned above in the first research topic above) this example notebook for some JAX code for optimization.

Note that this would be a very experimental research topic as it requires us to somewhat start from scratch, so there can be obstacles and hurdles to overcome. Two possible hurdles are: (i) the variational inference scheme struggles with multimodality in the target distribution (as hinted at by Ref. [9]), (ii) the normalizing flow is not expressive enough/too slow to train,... Notice, however, the following exciting insight: this is exactly what the above two research topics will be about! Therefore, all of the above thoughts and research topics combine into the following overarching research goal

Research goal: How can we optimize Bayesian inference in the field of parameter estimation of gravitational wave data? State-of-the-art methods leverage automatic differentiation and normalizing flows and use them in traditional MCMC sampling. How can we further optimize these methods to deal with multimodality in intractable target densities? Can neural variational inference compete with this method? What are the challenges involved here, and can we use the lessons learned from the state-of-the-art sampler codes to solve these challenges?

4 Plan for first weeks

These tasks were already given during our first meetings, but they should get finished now ASAP so that we can start working on the actual research.

- Choose thesis topic, start background reading, write up the summary of the research topic to be submitted to OSIRIS.
- Scour literature for toy problems/benchmarks for sampling problems (mixture of Gaussians, Rosenbrock function,...): this will be used for testing newly implemented code and benchmark different approaches

- Create a Github where you add code to generate these toy problems (writing up the likelihood function et cetera).
- Start gaining experience with solving these sampling problems with the use of flowMC.

5 Research workflow tips

Now is also the time to set things up to organize your research workflow

- Make sure to write as many notes as possible: during our meetings, during your own personal brainstorming sessions, when gathering first results,... Take a notepad with you to quickly jot down thoughts/sketch ideas/crunch through math. Besides this, start a simple Google slides where you can paste links (to code packages) and figures (if you did a new numerical experiment, and the result is interesting, paste it there with some comments on the setup and thoughts/things to explore). This will make sure you never lose track of your progress, and the slides can be used during our meetings to make them more efficient: it is always much better to have plots and links to code ready so that you don't suddenly discover you cannot find them anymore!
- Get a reference manager: my recommendation is Zotero. The same advice applies here: download any interesting paper to Zotero (there is a plugin for the browser making this easy), read them with taking highlights *and* make sure to add personal notes to papers. These can be simple for papers you don't read in detail ("found when looking for X, seems interesting for Y, need to check that later") or very detailed notes per section if you really want to digest the paper and understand its details. Making a summary in your own words takes time and is hard, but gives you so much insight into the paper. Moreover, make sure to add questions/criticism ("the authors mention X but is this really true? Need to check papers Y and Z mentioned there for comparison").
- Want to find more papers, do literature review? Start with Google scholar or Google search terms, and once you find some interesting papers, make sure to look at (i) its citations, especially if they appear in some "related works" section. Sometimes, there is an actual section with such a title, but other times, it is a bit hidden (but the section starts off with something like "Previous works have similarly investigated,... However, our approach differs because...". This will help you understand similarities and differences between individual papers, but also general ideas more broadly. (ii) Papers that cited that interesting papers later in time: perhaps the ideas are outdated (since sampling/machine learning is a rapidly evolving field, a few years can really make a difference!) and recent work has solved some of the issues mentioned in older papers or just made them more efficient!

6 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 [11, 12]) 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. [13], with the JOSS paper of the package in Ref. [14]. The

For more information regarding deep learning, see the book by Goodfellow *et al.* [15]. A broader introduction to machine learning in general can be found in the book by Murphy [16]. Advanced topics are discussed in a second book by Murphy [17]. Extensive background in probability theory and information theory underpinning machine learning can be read in the book by Mackay [18].

Normalizing flows are well explained by Ref. [6].

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

7 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] 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].

[7] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Trans. Pattern Anal. Machine Intell.* 43.11 (2021), pp. 3964–3979. DOI: 10.1109/tpami.2020.2992934. arXiv: 1908.09257 [stat.ML].

[8] Sameera Ramasinghe et al. *Robust normalizing flows using Bernstein-type polynomials*. 2022. arXiv: 2102.03509 [cs.LG]. URL: https://arxiv.org/abs/2102.03509.

[9] Matthew Mould, Noah E. Wolfe, and Salvatore Vitale. "Rapid inference and comparison of gravitational-wave population models with neural variational posteriors". In: (Apr. 2025). arXiv: 2504.07197 [astro-ph.IM].

[10] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (Apr. 2017), pp. 859–877. ISSN: 1537-274X. DOI: 10.1080/01621459.2017.1285773. URL: http://dx.doi.org/10.1080/01621459.2017.1285773.

[11] 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].

[12] 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].

[13] 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].

[14] 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].

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

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

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

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