Draft version February 20, 2023
Typeset using LaTeX **twocolumn** style in AASTeX631

Fast gravitational wave parameter estimation without compromises

Kaze W. K. Wong,[1] Maximiliano Isi,[1] and Thomas D. P. Edwards[2]

[1] *Center for Computational Astrophysics, Flatiron Institute, New York, NY 10010, USA*
[2] *William H. Miller III Department of Physics and Astronomy, Johns Hopkins University, Baltimore, Maryland 21218, USA*

ABSTRACT

We present a lightweight, flexible, and high-performance framework for inferring the properties of gravitational-wave events. By combining likelihood heterodyning, automatically-differentiable and accelerator-compatible waveforms, and gradient-based Markov chain Monte Carlo (MCMC) sampling enhanced by normalizing flows, we achieve full Bayesian parameter estimation for real events like GW150914 and GW170817 within a minute of sampling time. Our framework does not require pre-training or explicit reparameterizations and can be generalized to handle higher dimensional problems. We present the details of our implementation and discuss trade-offs and future developments in the context of other proposed strategies for real-time parameter estimation. Our code for running the analysis is publicly available on GitHub https://github.com/kazewong/jim.

1. INTRODUCTION

Parameter estimation (PE) underpins all of gravitational-wave physics and astrophysics, and is one of the most commonly performed tasks in gravitational-wave (GW) data analysis (Christensen & Meyer 2022; Thrane & Talbot 2019). The central goal of PE is to infer the parameters of a particular GW source given the strain data recorded by instruments like LIGO (Aasi et al. 2015; Abbott et al. 2021a,b), Virgo (Acernese et al. 2015) and KAGRA (Akutsu et al. 2021). In the standard compact binary coalescence (CBC) scenario, this could mean inferring intrinsic parameters such as the masses and spins of the compact objects, as well as extrinsic parameters such as their sky localization and distance from Earth. PE is also applied to test general relativity (GR) (Abbott et al. 2016, 2019a, 2021c), and constrain the properties of nuclear matter (Abbott et al. 2019b). PE is a crucial step in GW science, since it translates characteristics of the strain data into astrophysically relevant quantities that can be used to constrain astrophysical phenomena, including informing theories of binary evolution (Abbott et al. 2021d).

There exist a number of prominent, community-developed PE codes, including LALInference (Veitch et al. 2015), PyCBC Inference (Biwer et al. 2019), and Bilby (Ashton et al. 2019; Romero-Shaw et al. 2020). These packages have been tested by a number of groups and are well regarded as standard tools. However, while these tools have passed many robustness tests, they are known to be computationally intensive. The exact amount of time needed to analyze one event depends on factors like the duration and frequency of the signal, as well as features of the specific waveform model. Typical runtimes for production-level analyses can range from hours to weeks. This expense precludes iterating quickly on results, launching large scale measurement simulations, or obtaining results in low latency to inform astronomers for potential followup in real time.

Additionally, in the coming decade, there are planned upgrades for existing facilities, as well as plans for next-generation detectors such as the Einstein Telescope (ET) (Punturo et al. 2010) and the Cosmic Explorer (CE) (Abbott et al. 2017). These upgrades will increase the instrument's sensitivity and allow for the detection of more events with a better signal-to-noise ratio (SNR). The number of events that will be detected in the coming decade is expected to grow from around a thousand per year to over a million per year (Baibhav et al. 2019). This will put a significant strain on the current PE tools.

In order to address this, there are efforts from multiple groups to speed up the PE process. This includes methods that employ techniques such as reduced-order quadrature (Canizares et al. 2015; Smith et al. 2016), adaptive proposal distributions in nested sampling (Williams et al. 2021), deep learning networks pre-trained on large collections of waveforms (Dax et al. 2021, 2022), as well as methods that reduce the computational expense of classical PE by leveraging our knowl-

kwong@flatironinstitute.org

edge of GW signals (Veitch et al. 2015; Ashton & Talbot 2021; Cornish 2021a; Islam et al. 2022; Roulet et al. 2022; Lee et al. 2022; Lange et al. 2018; Wofford et al. 2022). While these avenues are promising for standard GW problems, they rely on assumptions that may not hold for analyses targeting additional physical effects beyond standard CBCs in GR, such as matter effects, lensing and deviations from GR.

In this work, we present a lightweight, flexible, and robust framework to infer GW event parameters in a fully-Bayesian analysis. Our approach relies on the following techniques to achieve its performance:

1. likelihood heterodyning,
2. differentiable waveform models,
3. a normalizing-flow enhanced Markov chain Monte-Carlo (MCMC) sampler, and
4. native support for hardware accelerators.

All these components, working jointly, come together into a high-performance and high-fidelity PE pipeline, which can achieve $\gtrsim 10^3\times$ speed ups relative to traditional tools without compromising accuracy or generality for efficiency, and without making limiting assumptions about the target parameter space.

The rest of the paper is structured as follows: we review the basics of PE and introduce our framework in Sec. 2; we present benchmarking results on both simulated and real data in Sec. 3; and, finally, we discuss the implications of this work and directions for future development in Sec. 4.

2. GRAVITATIONAL WAVE PARAMETER ESTIMATION

2.1. *Likelihood function*

The main objective of PE is to obtain a multidimensional posterior distribution $p(\theta \mid d)$ on parameters θ given strain data d. This probability density represents our best inference of the source properties, and encodes all relevant information contained in the observed data. To compute this object, we use Bayes' theorem to write

$$p(\theta \mid d) = \frac{\mathcal{L}(d \mid \theta)\pi(\theta)}{p(d)}\,, \tag{1}$$

where $\mathcal{L}(d \mid \theta)$ is the likelihood function, $\pi(\theta)$ is the prior distribution, and $p(d)$ is the evidence. Since the evidence is a normalization constant that does not depend on the source parameters, it is often omitted if we are only interested in the posterior distribution. The prior distribution is often chosen to be something simple (e.g., uniform in the component masses or a Gaussian distribution in the spins), or it could directly encode astrophysical information. Assuming the noise is drawn from a Gaussian process, the log-likelihood for GW data is given by

$$\log \mathcal{L}(d \mid \theta) = -\frac{1}{2}\left\langle d - h(\theta) \mid d - h(\theta)\right\rangle, \tag{2}$$

where d is the observed strain data, and $h(\theta)$ is the signal predicted by a waveform model with a specific set of source parameters θ. The right hand side of Eq. (2) can be evaluated in either the time or frequency domains. For stationary noise, it is computationally cheaper to compute the likelihood in the frequency domain, with a noise-weighted inner product given by

$$\langle a \mid b\rangle = 4\Re \int_0^\infty \frac{a^*(f)b(f)}{\mathcal{S}_n(f)}\,\mathrm{d}f\,, \tag{3}$$

where $\mathcal{S}_n(f)$ is the one-sided power spectral density (PSD) of the noise. In practice, the integral becomes a discrete sum over a finite number of bins determined by the sampling rate of the detector data and duration of the observation.

To compute the integral in Eq. (3), we need to evaluate a waveform model $h(\theta)$ at a number of frequency bins. This makes evaluating the likelihood function often the most computationally intensive part of PE. The most accurate waveforms are obtained via numerical relativity (NR) simulations (Baumgarte & Shapiro 2010), which directly solve the Einstein equations numerically for a given system. Unfortunately, such simulations can take anywhere from a day to half a year, making direct evaluation through NR prohibitively expensive. To circumvent this, there are several families of waveform "approximants", including the IMRPhenom family (Khan et al. 2016; García-Quirós et al. 2020), the SEOB family (Taracchini et al. 2014), and the NR surrogate family (Varma et al. 2019a). Since PE requires millions of likelihood evaluations during sampling, the computational cost in evaluating the waveform is a major contributor to the long runtimes of GW PE.

2.2. *Heterodyned likelihood*

Since the cost of evaluating a waveform model scales linearly with the number of time or frequency bins, the computational burden for longer-duration signals is often quite large. To mitigate this, there are a number of methods to reduce the number of basis points needed to compute the likelihood faithfully (Field et al. 2011, 2014; Smith et al. 2016; Vinciguerra et al. 2017; Morisaki & Raymond 2020; Morisaki 2021). In this work, we take advantage of likelihood heterodyning (Cornish 2010, 2021b) (also known as relative binning (Zackay et al. 2018)).

The idea behind the heterodyned likelihood can be summarized as follows: since the integrand in Eq. (3) is a highly oscillatory function, one has to sample it at a high rate to compute the integral faithfully; however, the number of sample points needed would be much smaller if the integrand was smooth. For a pair of points θ and θ_0 that are close to each other in parameter space, the corresponding waveforms $h(\theta)$ and $h(\theta_0)$ will necessarily be similar; this means that the ratio between waveforms is a smooth function of frequency. Given a reference waveform $h(\theta_0)$, we can exploit this fact to reduce the number of frequency bins needed to compute the likelihood for the set of θ in the neighborhood of θ_0.

To do this, we decompose the integrand into two parts: (1) a highly oscillatory part that depends only on the reference waveform given by θ_0 and the data, and which need only be evaluated once; and, (2) a smoothly varying part that depends on the target waveform parameters θ, which must be evaluated for every likelihood computation. Because the part that depends on the target waveform parameters is smooth, we can use far fewer frequency samples to compute the integral with sufficient accuracy.

One may be concerned about the accuracy of this scheme, especially in the region where the generated waveform is significantly different from the reference waveform. However, given that we are interested in the most probable set of parameters, if we choose the reference waveform to be close to the data, the waveforms that are different from the reference waveform will necessarily also differ significantly from the data. This means that the likelihood value for parameters far from the reference will be significantly smaller than the likelihood of those close to it, and hence will not be relevant for the PE result. To ensure that this is the case, we always pick reference parameters known to lie close to the target, e.g., by first maximizing the likelihood function using the highest frequency resolution available, which can be run at a much lower cost than full PE.

We now give a concise description of the implementation of this approach in our code; for a more extensive derivation of heterodyned likelihood, we refer the reader to Zackay et al. (2018). Let $h(f)$ and $h_0(f)$ represent the target and reference waveforms, respectively; then, for a given sparse binning of the frequency axis, the ratio $r(f) = h(f)/h_0(f)$ can be well approximated by a linear interpolation over the bin,

$$r(f) \approx r_0(h,b) + r_1(h,b)(f - f_m(b)) + \cdots , \qquad (4)$$

where b is the index of a particular bin, $r_0(h,b)$ and $r_1(h,b)$ are respectively the value and slope of the ratio at the center of the bin, and $f_m(b)$ is the central frequency of the bin. Since we have access to both $h(f)$ and $h_0(f)$, we can compute r_0 and r_1 by evaluating $r(f)$ at the edge of the bin and inverting Eq. (4).

With this definition, the two terms involving h obtained by expanding Eq. (2) can be approximated as

$$\langle d \mid h\rangle \approx \sum_b \left[A_0(b)\, r_0^*(h,b) + A_1(b)\, r_1^*(h,b)\right], \qquad (5a)$$

$$\langle h \mid h\rangle \approx \sum_b \big[B_0(b)\, |r_0(h,b)|^2 + 2B_1(b)\, \Re\{r_0(h,b)\, r_1(h,b)\}\big] \qquad (5b)$$

where $A_0(b)$, $A_1(b)$, $B_0(b)$, and $B_1(b)$ are heterodyning coefficients computed using the data and the reference waveform. These are defined to be

$$A_0(b) = 4\sum_{f\in b} \frac{d(f)h_0^*(f)}{S_n(f)}\Delta f, \qquad (6a)$$

$$A_1(b) = 4\sum_{f\in b} \frac{d(f)h_0^*(f)(f - f_m(b))}{S_n(f)}\Delta f, \qquad (6b)$$

$$B_0(b) = 4\sum_{f\in b} \frac{|h_0(f)|^2}{S_n(f)}\Delta f, \qquad (6c)$$

$$B_1(b) = 4\sum_{f\in b} \frac{|h_0(f)|^2(f - f_m(b))}{S_n(f)}\Delta f\,, \qquad (6d)$$

where the sums within each bin ($f \in b$) should be done with the same, dense sampling rate as the original data (with thin frequency bins of width Δf).

To evaluate Eq. (5), we need to first choose a binning scheme, then evaluate the coefficients in Eq. (6) given the data and the reference waveform, and at last the ratio between the target waveform and the reference waveform at the center of each bin via Eq. (4).

The phasing of an inspiral waveform is denoted by a power series $\Psi(f) = \sum_i \alpha_i f^{\gamma_i}$, where α_i are some coefficients depending on the waveform parameters and γ_i are powers motivated by post-Newtonian theory. For example, for the term $\gamma_i = -5/3$, α_i is related to the chirp mass. The maximum dephasing one can have within a frequency interval $[f_{\min}, f_{\max}]$ is

$$\delta\Psi_{\max}(f) = 2\pi\sum_i (f/f_{*,i})^{\gamma_i}\mathrm{sgn}(\gamma_i), \qquad (7)$$

where $f_{*,i} = f_{\max}$ for $\gamma_i \geq 0$ and $f_{*,i} = f_{\min}$ for $\gamma_i < 0$. Given the relation shown in Eq. (7), we can choose the binning scheme to divide the entire frequency band of interest into a set of bins such that the maximum dephasing within each bin is smaller than a certain threshold ϵ, i.e., $|\delta\Psi_{\max}(f_{\max}) - \delta\Psi_{\max}(f_{\min})| < \epsilon$.

To obtain a reference waveform, we currently use the DIFFERENTIAL EVOLUTION algorithm (Storn & Price 1997) available in the SCIPY package (Virtanen et al. 2020) to find the waveform parameters which maximize the likelihood. The reference waveform could also be produced from trigger parameters precomputed by a search pipeline without additional computation. Once we have obtained a reference waveform, we can check the accuracy of the heterodyned likelihood by comparing its value to the original likelihood at several points in the parameter space. We can then choose the number of bins such that the difference between the values of the two likelihoods is smaller than chosen tolerance threshold.

2.3. *MCMC with Gradient-based sampler*

Given Eq. (2) and the prior, one can evaluate the posterior density, Eq. (1), over the entire parameter space of interest to obtain the most probable set of values that are consistent with the data. However, directly sampling this posterior quickly becomes intractable as the dimensionality of the parameter space increases beyond a few dimensions. Markov chain Monte Carlo (Gelman et al. 2004) is a common method employed to generate samples from the target posterior when direct sampling is not possible.

In MCMC, the posterior distribution is approximated by a Markov chain that eventually converges to the target distribution (Tierney 1994). The chain is constructed by iteratively proposing a new point in the parameter space based on the current location of the chain. The proposed point is accepted with a probability that is usually set to be proportional to the ratio of the posterior density evaluated at the proposed point and the current point. The chain can either accept the proposal and move to the new location, or reject the proposal and stay at the current location. This process is repeated until the chain converges to the target distribution. The samples generated by the chain are then used as a fair sample to estimate the quantities of interest, such as the mean and credible intervals of the source parameters. In practice, since we do not know the target distribution ahead of time, the MCMC process is usually repeated until a certain criterion is met, such as a Gelman-Rubin convergence statistic (Gelman & Rubin 1992) lower than certain threshold, or simply after a fixed number of iterations.

Compared to direct sampling, MCMC algorithms only explore regions that are highly probable, thus reducing the computational cost by not wasting resources on parameters that are unlikely to generate the observed data. However, MCMC algorithms come with their own set of issues. To illustrate what difficulties MCMC may face, we can examine one of the most standard MCMC algorithms: the Metropolis-Hastings algorithm with a Gaussian kernel. Starting at some initial point, one can draw a proposed point from a Gaussian transition kernel, defined as

$$q(\mathbf{x}, \mathbf{x_0}) = \mathcal{N}(\mathbf{x} \mid \mathbf{x_0}, \mathbf{C}), \tag{8}$$

where $\mathbf{x_0}$ is the current location of the chain, $\mathbf{x}$ is the proposed location, and $\mathbf{C}$ is the covariance matrix of the Gaussian. In the simplest case, we can pick $\mathbf{C}$ to be a diagonal matrix with a constant value, which corresponds to an isotropic Gaussian center around the current location and with a fixed variance. The acceptance criterion is defined as

$$\alpha(\mathbf{x}, \mathbf{x_0}) = \min\left(1, \frac{p(\mathbf{x})q(\mathbf{x_0}, \mathbf{x})}{p(\mathbf{x_0})q(\mathbf{x}, \mathbf{x_0})}\right). \tag{9}$$

We can see from Eq. (9) that the acceptance rate is proportional to the fraction of volume where the posterior density at the proposed location is higher than the current location within the Gaussian transition kernel. If we choose the variance of the transition kernel to be too large, this fraction will be small hence the acceptance rate will be poor. On the other hand, if one chooses the variance to be too small, nearby samples will be correlated, and it will take a long time for the chain to wander. In both cases, the efficiency in constructing a chain with a target number of independent samples is suboptimal. Consequently, there is often a tuning process before we run the MCMC algorithm to find the optimal sampling settings (in this example, the variance of the Gaussian) to ensure the best possible performance.

However, as we often deal with high-dimensional problems, even the optimally tuned Gaussian transition kernel does not guarantee good performance. In order to have a reasonable acceptance rate, the variance of the Gaussian has to be smaller in a higher dimensional space, which means that the transition kernel will generally make smaller and smaller steps as we increase the dimensionality of the problem (Betancourt 2017).

Transition kernels that leverage gradient information of the target distribution can help address this issue of shortening steps in a high dimensional space. Instead of proposing a new point by drawing from a Gaussian, one can use the gradient evaluated at the current location to propose a new point, so that the evolution of the chain is preferentially directed to regions of higher probability. For example, the Metropolis-adjusted Langevin algorithm (MALA) (Grenander & Miller 1994) places a unit Gaussian at the tip of the gradient vector at the

current position,

$$\mathbf{x} = \mathbf{x_0} + \tau \nabla \log p(\mathbf{x_0}) + \sqrt{2\tau} N(0, \mathbf{I}) \,, \tag{10}$$

where τ is a step size chosen during the tuning stage. Compared to a Gaussian centered at the current location, the MALA transition kernel is more likely to propose a point in the higher posterior density region because of the gradient term, which helps boost the acceptance rate. Hamiltonian Monte Carlo (Betancourt 2017) is another gradient-based algorithm that has been explored for neutron star inspirals (Bouffanais & Porter 2019).

While transition kernels that use gradient information can help improve the acceptance rate, computing the gradient of the posterior density function introduces an additional computational cost, which is not necessarily beneficial in terms of sampling time. If one wants to compute the gradient through finite differencing, the additional computational cost goes as at least $\sim\mathcal{O}(2n)$, where n is the dimension of the problem. On the other hand, schemes like JAX (Bradbury et al. 2018) allow us to compute the gradient of the likelihood function with respect to the parameters through automatic differentiation, which gives the gradient information down to machine precision at around the same order of time compared to evaluating the posterior itself. Thus, having access to gradient information through automatic differentiation is crucial to making gradient-based transition kernels favorable in terms of computing cost.

To leverage automatic differentiation, the entire posterior function must be implemented within JAX or a similar framework; this includes the likelihood and, therefore, the waveform approximant. This means that the development of differentiable approximants is essential for leveraging gradient-based sampling in GW applications. We currently make use of the waveforms implemented in RIPPLE (Edwards et al. in prep.).

2.4. *Normalizing Flow enhanced sampling*

While gradient-based samplers have been shown to outperform gradient-free algorithms in many practical applications, there remain classes of problems that most gradient-based samplers do not solve well. For example, first-order gradient-based algorithms struggle with target distributions that exhibit locally-varying correlations, since they assume a single mass matrix that does not depend on the location of the chain by construction (Betancourt 2017).[1] Another example is multimodality: if there are multiple modes in the target distribution, individual chains will likely be trapped in one mode and take an extremely long time to transverse between the modes (Mangoubi et al. 2018). This means that the relative weights between modes will take much longer to estimate than the shape of each mode.

Moreover, before we can use the sampling chain to estimate the posterior quantities we care about, the sampler often needs to first find the most probable region in the target space (known as the *typical set*); this is a common process often referred to as "burn-in" in the literature. As a consequence, one would discard a certain amount of data generated from the beginning of the sampling process, and only use the later part of the chain to estimate the quantities of interest. The burn-in phase of a gradient-based sampler is often as long as the sampling phase, which means that a good portion of the computation is not directly devoted to estimating the target quantities.

All the above issues can be mitigated by normalizing flows. Normalizing flows are a technique based on neural networks that aims at learning a mapping from a simple distribution, such as a Gaussian, to a complex distribution, often given in the form of samples (Kobyzev et al. 2019; Papamakarios et al. 2019). Once the network is trained, one can evaluate the probability density of the complex distribution and sample from it very efficiently, by first evaluating the simple distribution and then applying the learned mapping. The core equation of normalizing flows is the coordinate transformation of probability distributions via a Jacobian, as given by

$$p_x(X) = p_z(Z) \left| \frac{\partial f}{\partial z} \right|^{-1} , \tag{11}$$

where $p_x(X)$ is the complex target distribution, $p_z(Z)$ is the simple latent distribution and f is an invertible parameterized transform that connects the two distributions, $x = f(z)$, to be learned by the normalizing flow. See Kobyzev et al. (2019); Papamakarios et al. (2019) for a detailed discussion of the algorithm.

Working in tandem, gradient-based MCMC and normalizing flows can efficiently explore posteriors with local and global correlations, as well as multiple separate modes. The scheme relies on iteratively using draws from the gradient-based MCMC to train a normalizing

[1] Sampling algorithms that use higher order derivatives such as manifold-MALA and Riemannian-HMC (Girolami & Calderhead 2011) can in principle handle local correlations in the target distribution; however, they often encounter instabilities when used in real-life applications, so their use is a rare practice.

flow, which is then itself used as a proposal for another stage of MCMC sampling.

Concretely, we begin by producing initial training data for the normalizing flow by running multiple independent chains of the gradient-based algorithm for a fixed number of steps. From the resulting pool of samples, the normalizing flow can begin to learn the landscape of the target distribution. However, since the independent chains contain the same number of samples, the relative weight assigned to each chain will not represent the true target distribution (e.g., the relative importance of separate modes will not be correctly calibrated). This is mitigated by a second stage of gradient-based MCMC sampling that uses the distribution learned by the normalizing flow as a *proposal*.

Given a trained normalizing flow model, we can generate the proposed jump in the target space by sampling from the latent distribution $z \sim p_z(Z)$, usually a Gaussian, and then pushing it through the learned map given by the normalizing flow model $x = f(z)$. The acceptance criterion is then set to be

$$\alpha(\mathbf{x}, \mathbf{x_0}) = \min\left[1, \frac{\hat{\rho}(\mathbf{x_0})\rho_*(\mathbf{x})}{\hat{\rho}(\mathbf{x})\rho_*(\mathbf{x_0})}\right], \tag{12}$$

where $\hat{\rho}$ is the probability density estimated by the normalizing flow model, ρ_* is the probability density evaluated using the target function, and x_0 is the current position.

From Eq. (12), we can see that the flow distribution is the target distribution when the accepting probability is 1. When the normalizing flow model has not converged to the target distribution, only a portion of the proposed jumps will be accepted. This means an MCMC process using the normalizing flow model as the proposal distribution can adjust the normalization across different regions of the target parameter space by rejecting jumps into less likely regions. The training and sampling are then repeated until certain criteria are met, at each step combining global and local MCMC sampling which respectively do and do not use the normalizing flow as proposal.

Note that every time we retrain the network, we are breaking the Markov properties since we are changing the proposal distribution. To produce final samples that can be used to estimate target quantities, one has to freeze the normalizing flow model and not retrain during the final sampling phase in order to satisfy the detailed balance condition. We use the package flowMC (Wong et al. 2022; Gabrié et al. 2022), with MALA as the gradient-based sampler. The pseudocode of the algorithm is given in Algorithm 1.

Algorithm 1: flowMC pseudocode

Input: initial position ip
Parameters: number of training loops nt, number of production loops np
Variables: current chains cc, current position cp, current NF parameters Θ, chains from local sampler c_{local}, chains from global sampler c_{global}
Result: output chains $chains$

$cp \leftarrow ip$
`/* Training loop */`
for $i < nt$ **do**
  $cc, cp \leftarrow LocalSampling(cp)$
  $\Theta \leftarrow TuneNF(cc)$
  $c_{global}, cp \leftarrow GlobalSampling(cp, \Theta)$
  $cc \leftarrow Append(cc, c_{global})$
`/* Production loop */`
for $i < np$ **do**
  $c_{local}, cp \leftarrow LocalSampling(cp)$
  $c_{global}, cp \leftarrow GlobalSampling(cp, \Theta)$
  $chains \leftarrow Append(chains, c_{local}, c_{global})$
return $chains$

2.5. *Accelerators*

Modern hardware accelerators, such as graphics processing units (GPUs) and tensor processing units (TPUs), are designed to execute large-scale, dense computation. They are often much more cost-efficient than using many central processing units (CPUs) when it comes to solving problems that can be benefited from parallelization. The downside of these accelerators compared to CPUs is that they can only perform a more restricted set of operations and are often less performant when dealing with serial problems. Parameter estimation with MCMC is a serial problem since each new sample generated from a chain depends on the last sample in the chain. This means that naively putting the problem on an accelerator is more likely to reduce performance than increase it.

Yet, in our work, the use of accelerators provides two independent advantages that tremendously benefit the parameter estimation process. First, using accelerators allows us to run many independent MCMC chains simultaneously, which benefits the training of the normalizing flow. Since we generate the data we use to train the normalizing flow on the fly, the more independent data we can feed to the training process, the higher chance the normalizing flow can learn a reasonable representation of the global landscape of the target distribution. If we only used a small number of chains, we would be limited to the correlated samples from each chain and we would have to run more sequential steps to obtain the same amount of independent samples in the end—with more

chains the problem becomes parallelizable and we can obtain the same number of training samples sooner. In other words, being able to use many independent chains helps the normalizing flow learn the global landscape faster in wall time.

Another benefit of accelerators is the parallel evaluation of waveforms. Since the waveform model we use can be evaluated at any given time or frequency independently, this means computing a waveform can be trivially parallelized over frequency bins. On an NVIDIA A100 GPU, we can evaluate the waveform model $\sim \mathcal{O}(10^9)$ times in a second for different frequencies or source parameters. Together with the heterodyned likelihood, this allows us to run thousands of parallel chains in a PE run. The high throughput of waveform evaluations unlocks the potential of the FLOWMC sampling algorithm.

3. RESULT

3.1. *Injection-recovery test*

To demonstrate the robustness of our pipeline, we use it to recover the parameters of a set of simulated signals injected into different instantiations of synthetic stationary Gaussian noise. Then we run our pipeline on the simulated data, and determine the credible interval at which the true parameters of the injected signals are recovered. From the set of credible values, we can check whether the truth lies within a certain credible interval at the expected frequency: if our pipeline is working as expected, we should find that the true parameters lie within $x\%$ credible interval $x\%$ of the time, e.g., the true value should lie within the 50% credible interval 50% of the time. In other words, the recovered percentiles of the true parameters should be uniformly distributed. Deviation from this behavior would suggest the pipeline is either over-confident or too conservative (Cook et al. 2006; Talts et al. 2018).

We sample 1200 events from the distribution of parameters detailed in Table 1; the same distributions are used as the prior in the PE process. We simulate signals over 16 s of data, with a minimum frequency cutoff of 30 Hz and a sampling rate of 2048 Hz. We draw noise from a set of projected design PSDs for the LIGO Hanford, LIGO Livingston (`SimNoisePSDaLIGOZeroDetHighPower`) and Virgo (`SimNoisePSDAdvVirgo`) detectors (LIGO Scientific Collaboration 2018; Shoemaker 2009; Manzotti & Dietz 2012). For both injection and recovery, we make use of the IMRPHENOMD waveform (Khan et al. 2016) via the fully-differentiable implementation presented in the RIPPLE package (Edwards et al. in prep.). We use a neural spline flow model (Durkan et al. 2019) with 10 layers, each with 128 hidden units, and 8 bins per layer as our normalizing flow model.

We summarize the result of this injection-recovery campaign in Fig. 1. This shows the cumulative distribution over injections of the quantile at which the true value lies in the marginalized distribution of each parameter. The shaded band denotes the 95%-confident variation expected from draws from a uniform distribution with the same number of events. We can see that most of the measured curves lie within this band, showing that our inference results agree well with a uniform distribution.

To further quantify how well our result agrees with a uniform distribution, we can compute the Kolmogorov-Smirnov p-values for each marginalized distribution (Karson 1968). A low p-value (with a threshold often chosen to be $p = 0.05$) could indicate that our result is in tension with a uniform distribution. The p-values obtained for each parameter are shown in the legend of Fig. 1. Most of them are well above the $p = 0.05$ threshold, except for ψ, which is close to the threshold. Once again, assuming these p-values are drawn from a uniform distribution, given 11 draws (the number of parameters in our inference), it is not abnormal to have one of the parameters lying slightly outside the threshold. To assess whether this is expected, we can compute the combined p-value for these 11 parameters, and find it to be $p = 0.70$. This shows our inference pipeline performs properly on simulated data at a similar level as standard tools (Veitch et al. 2015; Romero-Shaw et al. 2020).

3.2. *Real event parameter estimation*

To demonstrate the performance of our parameter estimation pipeline, we apply it to two real LIGO-Virgo events: GW150914 and GW170817. We use the priors shown in Table 1, and take 4 s of data sampled at 2048 Hz starting at 20 Hz for GW150914, and 128 s of data sampled at 4096 Hz starting at 23 Hz for GW170817; strain data and PSDs for both events are fetched from GWOSC (Abbott et al. 2021e). We use the same normalizing flow model as the injection-recovery study. For our specific choice of sampler settings, we produce ~2500 and 3500 *effective samples*[2] for GW150914 and GW170817 respectively. Running on an NVIDIA A100 GPU, the wall time for both events is around 3 minutes.

[2] Effective samples here refers to the number of independent samples, which is the total number of generated samples divided by their correlation length; we compute the effective sample size using ARVIZ (Kumar et al. 2019), https://python.arviz.org/en/stable/api/generated/arviz.ess.html.

Parameter	Description	Injection	GW150914	GW170817
M_c	chirp mass $[M_\odot]$	$[10, 50]$	$[10, 80]$	$[1.18, 1.21]$
q	mass ratio	$[0.5, 1]$	$[0.125, 1]$	$[0.125, 1]$
χ_1	primary dimensionless spin	$[-0.5, 0.5]$	$[-1, 1]$	$[-0.05, 0.05]$
χ_2	secondary dimensionless spin	$[-0.5, 0.5]$	$[-1, 1]$	$[-0.05, 0.05]$
d_L	luminosity distance [Mpc]	$[300, 2000]$	$[0, 2000]^\dagger$	$[1, 75]^\dagger$
t_c	coalescence time [s]	$[-0.5, 0.5]$	$[-0.1, 0.1]$	$[-0.1, 0.1]$
ϕ_c	coalescence phase	$[0, 2\pi]$	$[0, 2\pi]$	$[0, 2\pi]$
$\cos\iota$	cosine of inclination angle	$[-1, 1]$	$[-1, 1]$	$[-1, 1]$
ψ	polarization angle	$[0, \pi]$	$[0, \pi]$	$[0, \pi]$
α	right ascension	$[0, 2\pi]$	$[0, 2\pi]$	$[0, 2\pi]$
$\sin\delta$	sine of declination	$[-1, 1]$	$[-1, 1]$	$[-1, 1]$

Table 1. Prior ranges for parameters varied in the injection-recovery test, as well as the GW150914 and GW170817 analyses. All priors are uniform over the ranges shown, except for the luminosity distance prior in the GW150914 and GW170817 analyses (†) for which we apply a prior unform in comoving volume. The coalescence time refers to a shift relative to the geocenter trigger time, and M_c refers to the redshifted (detector-frame) chirp mass.

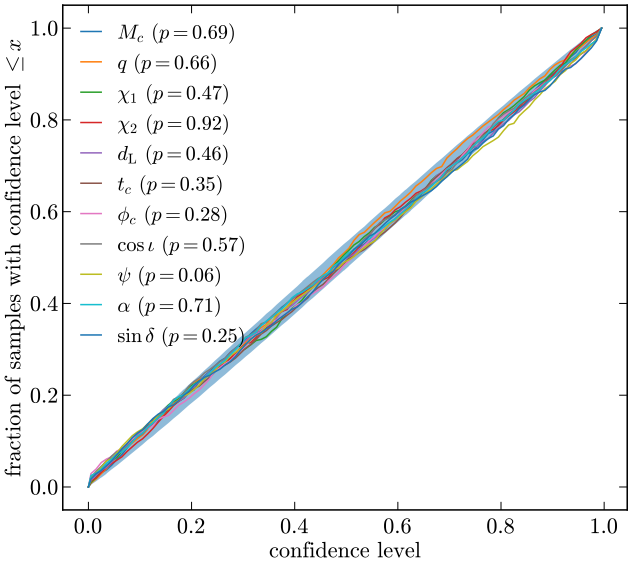

Figure 1. Cumulative distribution of the quantile of which the true value lies for each marginalized distribution. The shadow band denotes the 95% credible interval drawn from a uniform distribution with the same number of events as the injection campaign. The legend shows the p-values for each marginalized distribution, with a combined value of $p = 0.70$.

Most of this time is spent on just-in-time (JIT) compilation of the code; the actual sampling time is only ∼40 s. We pre-compute the reference waveform parameters used to hetetrodyne the likelihood for the two events, which is omitted in the wall-time calculation.

For comparison, we produce equivalent runs with Bilby, using the same exact data and priors. We use the dynesty sampler (Speagle 2020; Koposov et al. 2022), with 1000 live points and other settings as in the configurations files available on github https://github.com/kazewong/TurboPE/tree/main/src. We carry out these runs using parallel Bilby (pBilby) (Smith et al. 2020) to distribute the computation over 400 Intel Skylake CPUs for each event. For the specific settings chosen, the wall-time duration of each run was ∼2 h for GW150914 and ∼1 day for GW170817.

Figs. 2 and 3 show that our posteriors are consistent with those produced by Bilby. For a quantitative comparison, we compute the Jensen-Shannon divergence between our code and Bilby for the marginalized distribution for each parameter. The Jensen-Shannon divergence (JSD) is a symmetric measure of the distance between two probability distributions, with a value of 0 indicating identical distributions and a value of $\ln 2$ nat representing the maximum possible divergence between two distributions. The JSD values for the two events are shown in Table 2. The maximum JSDs for GW150914 and GW170817 are 0.0172 and 0.0026, and the mean JSDs are 0.0031 and 0.0012, respectively. The JSD values are comparable to those reported in Romero-Shaw et al. (2020), which show our code agrees with existing tools.

4. DISCUSSION

4.1. *Comparison to other approaches*

There have been several recent efforts to speed up GW parameter estimation, relying on techniques ranging from efficient reparameterizations (Islam et al. 2022; Roulet et al. 2022) to deep learning (Dax et al. 2021, 2022). While all of these methods can achieve minutes-scale parameter estimation with high fidelity under some conditions, our approach possesses unique strengths, and may complement some of those other techniques. Additional approaches for speeding up GW parameter

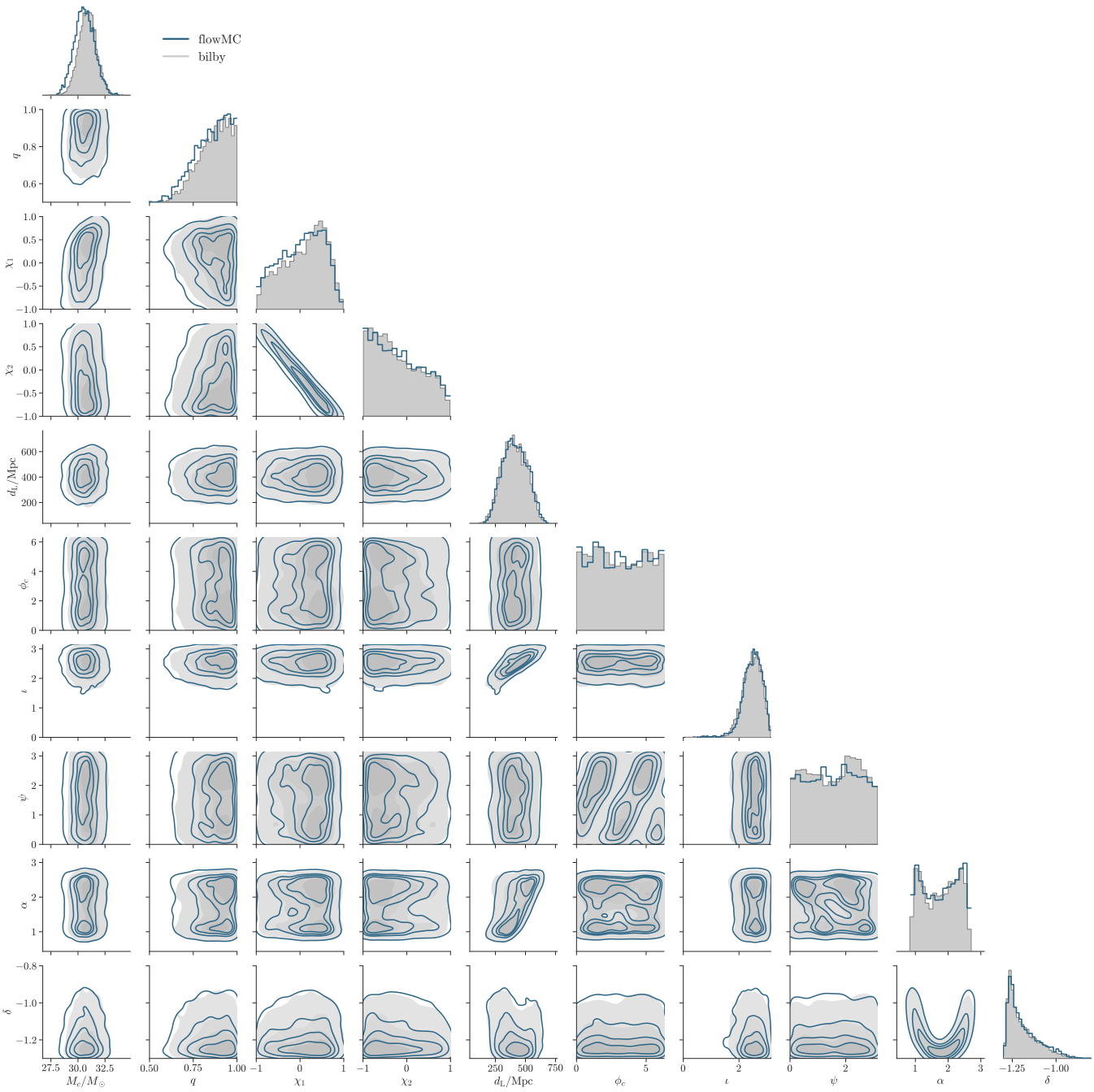

Figure 2. GW150914 posterior computed by our code (blue) and Bilby (gray).

estimation include Canizares et al. (2015); Smith et al. (2016); Lee et al. (2022); Wofford et al. (2022); Lange et al. (2018); Williams et al. (2021); Morisaki (2021); here we discuss those most relevant in the context of our work.

In contrast to Dax et al. (2021, 2022), we do not require pre-training the neural network on a large collection of waveforms and noise realizations. This means that our algorithm can be immediately deployed as soon as new waveform models and noise models are available. Furthermore, our method is at its core an MCMC algorithm, meaning it inherits the merit of convergence measures in MCMC. As we are only using the normalizing flow as a proposal distribution, and the normalizing flow is trained jointly with a local sampler, we do not risk overfitting since our training data is being generated on the fly and is always approaching the target distribution. In this sense, we do not introduce potential extra systematic errors to the inference results.

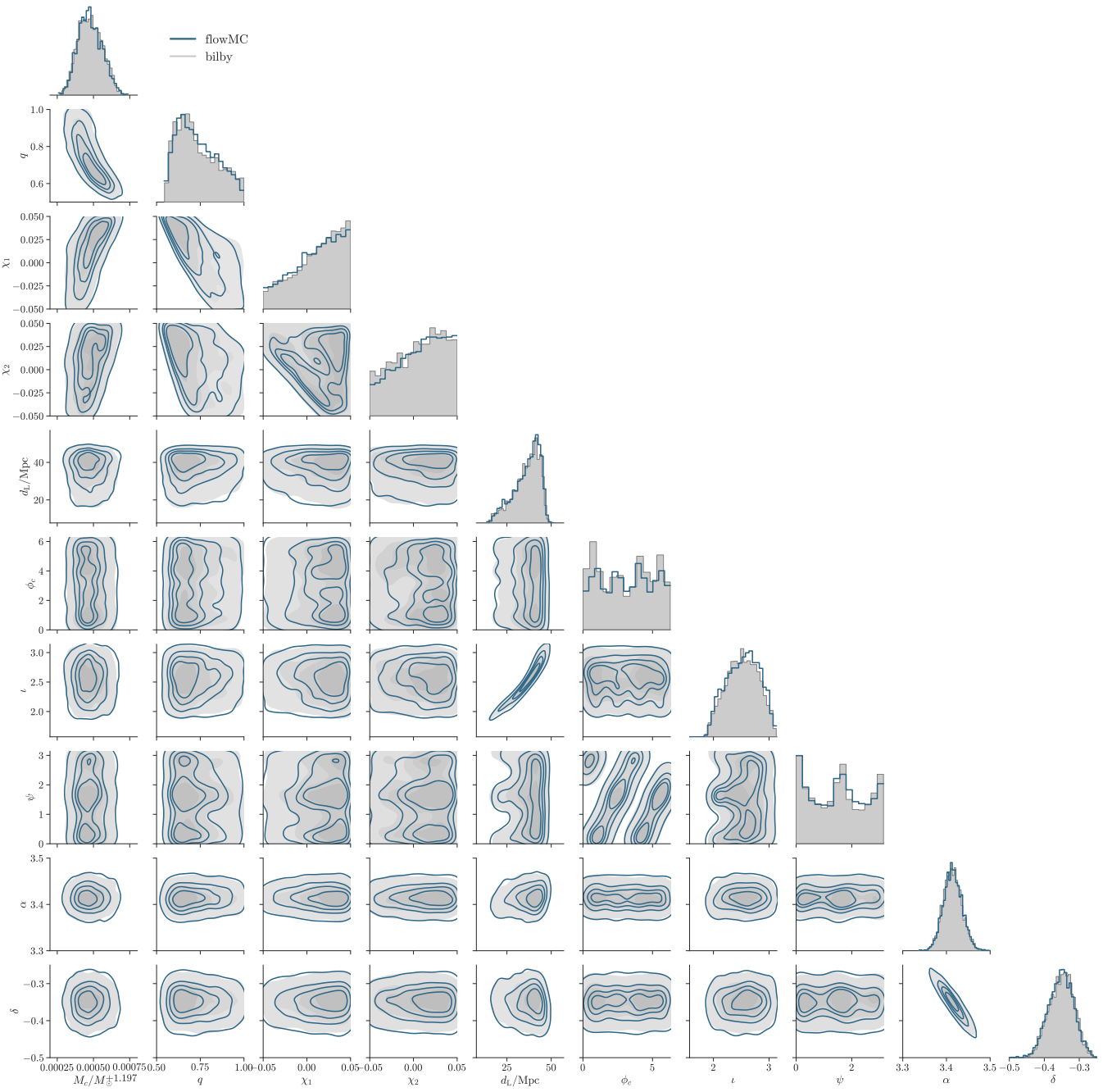

Figure 3. GW170817 posterior computed by our code (blue) and Bilby (gray).

While our pipeline uses samples generated by the local sampler for training, one could also supply a pre-trained normalizing flow to our pipeline to bypass the training stage. This would have the advantage of further reducing the total runtime; however, it could introduce systematic bias in the inference result if the pre-trained network is not able to capture the complexity presented in the data.

In contrast to Islam et al. (2022); Roulet et al. (2022) or other reparameterization schemes, we do not rely on handcrafted coordinate systems for sampling. If a useful coordinate transformation is known ahead of time, it can be trivially implemented within our pipeline, potentially easing convergence. However, tailored reparameterizations rely on specific assumptions about the targeted signal, which cannot always be generalized beyond specific applications. On the other hand, within our pipeline, the normalizing flow effectively discovers reparameterizations that ease sampling automatically without *a priori* knowledge of the structure of the problem. In gen-

	GW150914	GW170817
M_c	**0.01716**	0.00079
q	0.00361	0.00114
χ_1	0.00234	0.00055
χ_2	0.00099	0.00088
$d_{\rm L}$	0.00095	0.00108
ϕ_c	0.00032	**0.00257**
ι	0.00262	0.00155
ψ	0.00056	0.00061
α	0.00073	0.00176
δ	0.00202	0.00150

Table 2. JSD values for the marginalized distribution of each parameter for GW150914 and GW170817 between our code and Bilby. The bold values indicate the parameters with the largest JSD.

eral, the transformation discovered by the normalizing flow will only be approximate and hence not as efficient as an explicit reparameterization of the problem; yet, our approach applies to a much border class of problems where clever coordinate transformations are not known ahead of time, such as parameter estimation with precessing waveforms, calibration parameters, testing GR, and multi-event joint inference.

It is always beneficial to reparameterize if a convenient mapping is known ahead of time. For the class of problems treated in Islam et al. (2022); Roulet et al. (2022), we can incorporate those reparameterizations directly into our MCMC pipeline to reduce the complexity of the problem, hence speeding up the training phase. If there are limitations to the reparameterization that mean it cannot properly encompass part of the target posterior, the normalizing flow should still be able to learn to produce accurate samples efficiently.

The two avenues discussed here (machine learning and reparameterizations) represent two orthogonal directions one can take in building next generation PE tools. On the one hand, there are modern techniques such as deep learning that are very flexible and powerful, but rely on having highly robust training data. On the other hand, there are traditional tools that make use of our understanding of the underlying physics to ease sampling, which relies on having good intuition of the problem and tends to not generalize. Our method achieves the key advantages of both approaches, without many of their limitations.

The main difference between methods used in industrial products and scientific problems is that the latter must address questions that may not have been answered before, hence requiring techniques that generalize beyond the current state of knowledge robustly. Our work leverages both reparameterization and machine learning, yet our method can be trivially extended to problems beyond standard CBC analyses that would be unsuitable for reparameterizations or deep learning alone. Beyond efficiency, such flexibility and robustness are crucial for building scientific tools.

4.2. *Future development*

We are currently working on a number of improvements and extensions to our current infrastructure. While the IMRPhenomD waveform approximant is a reasonable start, it lacks some qualitative features that other state-of-the-art models have, such as precession, subdominants moments of the radiation, tides, or eccentricity. It also has a higher mismatch with reference numerical relativity waveforms compared to more recent waveform models. Currently, we are working on building differentiable implementations of IMRPhenomXPHM (Pratten et al. 2021), a precessing successor to IMRPhenomD including subdominant harmonics, as well as the numerical relativity surrogate waveforms, including NRSur7dq4 (Varma et al. 2019b). Going forward, we expect the use of autodifferentiation environments like Jax to become more prevalent in the waveform development community, increasing the number of differentiable waveform models available. This would not only be beneficial for parameter estimation, but also for a number of other applications such as Fisher matrix computations, template placement and calibrating waveforms to numerical relativity (Coogan et al. 2022; Iacovelli et al. 2022a,b; Edwards et al. in prep.).

While standard CBC analyses go up to 17 dimensions, non-standard GW PE problems can have more parameters, which could potentially lead to more complicated geometries in the target posterior that is hard to reparameterize. For example, Abbott et al. (2021c) introduces 10 extra parameters controlling deviations in the post-Newtonian coefficients predicted in GR. On top of the increase of dimensionality, these parameters often introduce non-trivial degeneracies such as local correlation and multi-modality. Therefore, currently testing GR is limited in practice to varying these modifications one at a time, partially due to the bottleneck in the sampler. Given the gradient-based and normalizing flow-enhanced sampler, our code shows promise in tackling this problem.

Our current code can perform parameter estimation for any combination of ground-based detectors, under the assumption that signals are transient and their wavelength is short. The first condition guarantees that the effect of Earth's rotation can be ignored when comput-

ing antenna patterns, while the second means that we can treat the antenna patterns as frequency independent constants. These assumptions break for next-generation detectors, whether on Earth or in space, like Cosmic Explorer, Einstein Telescope and LISA; differentiable implementations of antenna patterns for those detectors is work in progress.

Furthermore, our current implementation is minimal and we do not make use of most standard "tricks" to accelerate sampling. In particular, we do not incorporate (semi)analytic marginalization schemes over parameters such as time and phase (Thrane & Talbot 2019). As the performance of our implementation is not significantly impacted by having two extra dimensions, time and phase marginalization are not crucial for us; however, their implementation within our framework would be trivial.

For simplicity, we did not include calibration uncertainties in this study, but implementing this is also straightforward. Additionally, we took the noise PSD as an input for our analysis, as is the case for most traditional PE pipelines. However, since we are planning to deploy this pipeline to perform nearly-online PE, we also need to consider estimating the PSD on the same timescale, for which Cornish (2021a) has proposed a solution. We are looking into incorporating this, as well as analytic marginalization schemes and calibration uncertainties, into our pipeline.

When it comes to wall time, the just-in-time compilation of our code is the current limiting factor. While JAX's JIT compilation drastically accelerates likelihood evaluations, it comes with a significant compilation overhead before the first evaluation. We observe that the compilation time depends on the device where the code is run; this is expected since JAX leverages the ACCELERATED LINEAR ALGEBRA (XLA) compiler to take advantage of hardware accelerators, which means that JAX needs to compile the code for each specific device according to its architecture. On an NVIDIA A100 GPU, the compilation overhead could go up to 3 minutes for our current waveform. Meanwhile, for the cases we have studied, the time needed to obtain converging sampling on an A100 is about $\sim 40\,\mathrm{s}$. This means the compilation overhead dominates the wall-clock time of our current PE runs. To maximize the potential of our code, we are looking into ways to reduce the compilation overhead or to cache the compilation results to avoid paying the compilation overhead for every event.

Besides compilation, there is in principle also overhead from finding the reference waveform used in for heterodyning the likelihood. Since the DIFFERENTIAL EVOLUTION algorithm we currently use has not been implemented in JAX, and the JAX waveform we use is not compatible with the parallelization scheme in the SCIPY library, maximizing the likelihood currently takes us around 1 minute for GW170817. There are two ways to reduce this time.

First, we can explore a different optimization strategy that takes full advantage of the strengths of our pipeline, in particular the differentiability of our likelihood and the possibility to evaluate many waveforms in parallel with a GPU. Particle swarm (Bonyadi & Michalewicz 2017) and stochastic gradient descent methods (Bottou 1999) are promising candidates we are investigating.

Second, we may marginalize extrinsic parameters to reduce the dimensionality of the optimization problem. Currently, we simultaneously maximize all 11 CBC parameters in our problem numerically, which is unnecessary. There are long-existing, efficient maximization schemes for extrinsic parameters, such as the merger time and phase, which can find the corresponding maximum likelihood waveform much more efficiently when compared to differential evolution. We expect implementing these schemes will reduce the time needed to find the reference waveform parameters by fixing the extrinsic parameters and by reducing the dimensionality of the optimization problem. Furthermore, the search pipeline provides a subset of the parameters such as the masses, which can be fixed during the optimization to further reduce the dimensionality of the problem.

Finally, one important aspect of modern computing is scalability, meaning it is generally favorable if one can simply devote more computing units to the same problem in order to reduce the wall time. In our case, this means that we would like to use more than one GPU for the same PE process. More GPUs can help in the following ways: first, we can run more independent chains at the same time, which can generate more samples faster; second, and more importantly, as shown in this work and Wong et al. (2022), more independent chains also help reduce the burn-in time. Parallelizing over the number of chains dimension is trivial and does not require much change to the current infrastructure. Additional GPUs can also help by enabling faster training of larger flow models. While the training time is not the biggest bottleneck given the flow model used in this study, more GPUs means we can increase the bandwidth of the flow model by increasing its size while keeping the training time the same. This would help capture more complex geometries in the target space, which can lead to better convergence in general.

5. CONCLUSION

In this work, we presented a PE pipeline for GW events that is efficient, flexible and reliable. Our package brings together a number of innovations, including differentiable waveform models, likelihood heterodyning, and normalizing-flow enhanced gradient-based sampling. We tested the robustness of our pipeline, currently built upon RIPPLE and FLOWMC, on a set of 1200 synthetic GW events, showing it is robust, unbiased and efficient enough to handle the large catalogs of detections that will be available in the near future. We also show that our pipeline can estimate the parameters of GW150914 and GW170817 within a couple minutes, demonstrating the potential of our implementation on real data.

6. ACKNOWLEDGEMENTS

We thank Will M. Farr, Aaron Zimmerman, Daniel Foreman-Mackey and Marylou Gabrié for helpful discussions; we also thank Carl-Johan Haster, Neil J. Cornish and Thomas Dent for comments on the draft. The Flatiron Institute is a division of the Simons Foundation. This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation. This research has made use of data or software obtained from the Gravitational Wave Open Science Center (gw-openscience.org), a service of LIGO Laboratory, the LIGO Scientific Collaboration, the Virgo Collaboration, and KAGRA. LIGO Laboratory and Advanced LIGO are funded by the United States National Science Foundation (NSF) as well as the Science and Technology Facilities Council (STFC) of the United Kingdom, the Max-Planck-Society (MPS), and the State of Niedersachsen/Germany for support of the construction of Advanced LIGO and construction and operation of the GEO600 detector. Additional support for Advanced LIGO was provided by the Australian Research Council. Virgo is funded, through the European Gravitational Observatory (EGO), by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale di Fisica Nucleare (INFN) and the Dutch Nikhef, with contributions by institutions from Belgium, Germany, Greece, Hungary, Ireland, Japan, Monaco, Poland, Portugal, Spain. The construction and operation of KAGRA are funded by Ministry of Education, Culture, Sports, Science and Technology (MEXT), and Japan Society for the Promotion of Science (JSPS), National Research Foundation (NRF) and Ministry of Science and ICT (MSIT) in Korea, Academia Sinica (AS) and the Ministry of Science and Technology (MoST) in Taiwan. T.E. is supported by the Horizon Postdoctoral Fellowship.

REFERENCES

Aasi, J., et al. 2015, Class. Quant. Grav., 32, 074001, doi: 10.1088/0264-9381/32/7/074001

Abbott, B. P., et al. 2016, Phys. Rev. Lett., 116, 221101, doi: 10.1103/PhysRevLett.116.221101

—. 2017, Class. Quant. Grav., 34, 044001, doi: 10.1088/1361-6382/aa51f4

—. 2019a, Phys. Rev. Lett., 123, 011102, doi: 10.1103/PhysRevLett.123.011102

—. 2019b, Phys. Rev. X, 9, 011001, doi: 10.1103/PhysRevX.9.011001

Abbott, R., et al. 2021a. https://arxiv.org/abs/2108.01045

—. 2021b. https://arxiv.org/abs/2111.03606

—. 2021c. https://arxiv.org/abs/2112.06861

—. 2021d. https://arxiv.org/abs/2111.03634

—. 2021e, SoftwareX, 13, 100658, doi: 10.1016/j.softx.2021.100658

Acernese, F., et al. 2015, Class. Quant. Grav., 32, 024001, doi: 10.1088/0264-9381/32/2/024001

Akutsu, T., et al. 2021, PTEP, 2021, 05A101, doi: 10.1093/ptep/ptaa125

Ashton, G., & Talbot, C. 2021, Mon. Not. Roy. Astron. Soc., 507, 2037, doi: 10.1093/mnras/stab2236

Ashton, G., et al. 2019, Astrophys. J. Suppl., 241, 27, doi: 10.3847/1538-4365/ab06fc

Baibhav, V., Berti, E., Gerosa, D., et al. 2019, Phys. Rev. D, 100, 064060, doi: 10.1103/PhysRevD.100.064060

Baumgarte, T. W., & Shapiro, S. L. 2010, Numerical Relativity: Solving Einstein's Equations on the Computer (Cambridge University Press), doi: 10.1017/CBO9781139193344

Betancourt, M. 2017. https://arxiv.org/abs/1701.02434

Biwer, C. M., Capano, C. D., De, S., et al. 2019, Publ. Astron. Soc. Pac., 131, 024503, doi: 10.1088/1538-3873/aaef0b

Bonyadi, M. R., & Michalewicz, Z. 2017, Evolutionary Computation, 25, 1, doi: 10.1162/EVCO_r_00180

Bottou, L. 1999, On-Line Learning and Stochastic Approximations (USA: Cambridge University Press), 9–42

Bouffanais, Y., & Porter, E. K. 2019, Phys. Rev. D, 100, 104023, doi: 10.1103/PhysRevD.100.104023

Bradbury, J., Frostig, R., Hawkins, P., et al. 2018, JAX: composable transformations of Python+NumPy programs, 0.4.3. http://github.com/google/jax

Canizares, P., Field, S. E., Gair, J., et al. 2015, Phys. Rev. Lett., 114, 071104, doi: 10.1103/PhysRevLett.114.071104

Christensen, N., & Meyer, R. 2022, Rev. Mod. Phys., 94, 025001, doi: 10.1103/RevModPhys.94.025001

Coogan, A., Edwards, T. D. P., Chia, H. S., et al. 2022, Phys. Rev. D, 106, 122001, doi: 10.1103/PhysRevD.106.122001

Cook, S. R., Gelman, A., & Rubin, D. B. 2006, Journal of Computational and Graphical Statistics, 15, 675. http://www.jstor.org/stable/27594203

Cornish, N. J. 2010. https://arxiv.org/abs/1007.4820

—. 2021a, Phys. Rev. D, 103, 104057, doi: 10.1103/PhysRevD.103.104057

—. 2021b, Phys. Rev. D, 104, 104054, doi: 10.1103/PhysRevD.104.104054

Dax, M., Green, S. R., Gair, J., et al. 2021, Phys. Rev. Lett., 127, 241103, doi: 10.1103/PhysRevLett.127.241103

—. 2022. https://arxiv.org/abs/2210.05686

Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. 2019, arXiv e-prints, arXiv:1906.04032, doi: 10.48550/arXiv.1906.04032

Edwards, T. D. P., Wong, K. W. K., Lam, K. K. H., et al. in prep., RIPPLE: Differentiable and Hardware-Accelerated Waveforms for Gravitational Wave Data Analysis. https://github.com/tedwards2412/ripple

Field, S. E., Galley, C. R., Herrmann, F., et al. 2011, Phys. Rev. Lett., 106, 221102, doi: 10.1103/PhysRevLett.106.221102

Field, S. E., Galley, C. R., Hesthaven, J. S., Kaye, J., & Tiglio, M. 2014, Phys. Rev. X, 4, 031006, doi: 10.1103/PhysRevX.4.031006

Gabrié, M., Rotskoff, G. M., & Vanden-Eijnden, E. 2022, Proc. Nat. Acad. Sci., 119, e2109420119, doi: 10.1073/pnas.2109420119

García-Quirós, C., Colleoni, M., Husa, S., et al. 2020, Phys. Rev. D, 102, 064002, doi: 10.1103/PhysRevD.102.064002

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. 2004, Bayesian Data Analysis, 2nd edn. (Chapman and Hall/CRC)

Gelman, A., & Rubin, D. B. 1992, Statistical Science, 7, 457. http://www.jstor.org/stable/2246093

Girolami, M., & Calderhead, B. 2011, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73, 123, doi: https://doi.org/10.1111/j.1467-9868.2010.00765.x

Grenander, U., & Miller, M. I. 1994, Journal of the Royal Statistical Society. Series B (Methodological), 56, 549. http://www.jstor.org/stable/2346184

Iacovelli, F., Mancarella, M., Foffa, S., & Maggiore, M. 2022a, Astrophys. J., 941, 208, doi: 10.3847/1538-4357/ac9cd4

—. 2022b, Astrophys. J. Supp., 263, 2, doi: 10.3847/1538-4365/ac9129

Islam, T., Roulet, J., & Venumadhav, T. 2022. https://arxiv.org/abs/2210.16278

Karson, M. 1968, Journal of the American Statistical Association, 63, 1047, doi: 10.1080/01621459.1968.11009335

Khan, S., Husa, S., Hannam, M., et al. 2016, Phys. Rev. D, 93, 044007, doi: 10.1103/PhysRevD.93.044007

Kobyzev, I., Prince, S. J. D., & Brubaker, M. A. 2019, arXiv e-prints, arXiv:1908.09257. https://arxiv.org/abs/1908.09257

Koposov, S., Speagle, J., Barbary, K., et al. 2022, joshspeagle/dynesty: v2.0.3, v2.0.3, Zenodo, doi: 10.5281/zenodo.7388523

Kumar, R., Carroll, C., Hartikainen, A., & Martin, O. 2019, Journal of Open Source Software, 4, 1143, doi: 10.21105/joss.01143

Lange, J., O'Shaughnessy, R., & Rizzo, M. 2018. https://arxiv.org/abs/1805.10457

Lee, E., Morisaki, S., & Tagoshi, H. 2022, Phys. Rev. D, 105, 124057, doi: 10.1103/PhysRevD.105.124057

LIGO Scientific Collaboration. 2018, LIGO Algorithm Library - LALSuite, free software (GPL), doi: 10.7935/GT1W-FZ16

Mangoubi, O., Pillai, N. S., & Smith, A. 2018, arXiv e-prints, arXiv:1808.03230. https://arxiv.org/abs/1808.03230

Manzotti, A., & Dietz, A. 2012, arXiv e-prints, arXiv:1202.4031, doi: 10.48550/arXiv.1202.4031

Morisaki, S. 2021, Phys. Rev. D, 104, 044062, doi: 10.1103/PhysRevD.104.044062

Morisaki, S., & Raymond, V. 2020, Phys. Rev. D, 102, 104020, doi: 10.1103/PhysRevD.102.104020

Papamakarios, G., Nalisnick, E., Jimenez Rezende, D., Mohamed, S., & Lakshminarayanan, B. 2019, arXiv e-prints, arXiv:1912.02762. https://arxiv.org/abs/1912.02762

Pratten, G., et al. 2021, Phys. Rev. D, 103, 104056, doi: 10.1103/PhysRevD.103.104056

Punturo, M., et al. 2010, Class. Quant. Grav., 27, 194002, doi: 10.1088/0264-9381/27/19/194002

Romero-Shaw, I. M., et al. 2020, Mon. Not. Roy. Astron. Soc., 499, 3295, doi: 10.1093/mnras/staa2850

Roulet, J., Olsen, S., Mushkin, J., et al. 2022, Phys. Rev. D, 106, 123015, doi: 10.1103/PhysRevD.106.123015

Shoemaker, D. 2009, Advanced LIGO anticipated sensitivity curves (obsolete), Tech. Rep. LIGO-T0900288, LIGO Laboratory. https://dcc.ligo.org/LIGO-T0900288/public

Smith, R., Field, S. E., Blackburn, K., et al. 2016, Phys. Rev. D, 94, 044031, doi: 10.1103/PhysRevD.94.044031

Smith, R. J. E., Ashton, G., Vajpeyi, A., & Talbot, C. 2020, Mon. Not. Roy. Astron. Soc., 498, 4492, doi: 10.1093/mnras/staa2483

Speagle, J. S. 2020, MNRAS, 493, 3132, doi: 10.1093/mnras/staa278

Storn, R., & Price, K. V. 1997, Journal of Global Optimization, 11, 341

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., & Gelman, A. 2018, arXiv e-prints, arXiv:1804.06788. https://arxiv.org/abs/1804.06788

Taracchini, A., Buonanno, A., Pan, Y., et al. 2014, Phys. Rev. D, 89, 061502, doi: 10.1103/PhysRevD.89.061502

Thrane, E., & Talbot, C. 2019, PASA, 36, e010, doi: 10.1017/pasa.2019.2

Tierney, L. 1994, The Annals of Statistics, 22, 1701 , doi: 10.1214/aos/1176325750

Varma, V., Field, S. E., Scheel, M. A., et al. 2019a, Phys. Rev. Research., 1, 033015, doi: 10.1103/PhysRevResearch.1.033015

—. 2019b, Phys. Rev. D, 99, 064045, doi: 10.1103/PhysRevD.99.064045

Veitch, J., et al. 2015, Phys. Rev. D, 91, 042003, doi: 10.1103/PhysRevD.91.042003

Vinciguerra, S., Veitch, J., & Mandel, I. 2017, Class. Quant. Grav., 34, 115006, doi: 10.1088/1361-6382/aa6d44

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: 10.1038/s41592-019-0686-2

Williams, M. J., Veitch, J., & Messenger, C. 2021, Phys. Rev. D, 103, 103006, doi: 10.1103/PhysRevD.103.103006

Wofford, J., et al. 2022. https://arxiv.org/abs/2210.07912

Wong, K. W. K., Gabrié, M., & Foreman-Mackey, D. 2022, arXiv e-prints, arXiv:2211.06397. https://arxiv.org/abs/2211.06397

Zackay, B., Dai, L., & Venumadhav, T. 2018. https://arxiv.org/abs/1806.08792