Draft version January 23, 2023
Typeset using LaTeX **twocolumn** style in AASTeX631

Turbo fast no compromise 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 light-weightedlightweight, flexible, and high-performance framework to estimate gravitational wave event parametersfor inferring the properties of gravitational-wave events. By combining heterodyned likelihood , automatically differentiable and accelerators compatible waveforms, normalizing flow enhanced likelihood heterodyning, automatically-differentiable and accelerator-compatible waveforms, and gradient-based Markov Chain 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 reparameterization reparameterizations and can be generalized to handle higher dimensional problems. We also discuss as near real-time parameter estimation has been shown to be possible by multiple groups, what are the present the details of our implementation and discuss trade-offs and future developments the community should be aware of in the context of other proposed strategies for real-time parameter estimation. Our code for generating the manuscript and running the analysis is publically publicly available on GitHub.

1. INTRODUCTION

Parameter estimation (PE) underpins all of gravitational-wave physics and astrophysics, and is one of the most commonly performed data analysis tasks in gravitational wave tasks in gravitational-wave (GW) data analysis Christensen & Meyer (2022); Thrane & Talbot (2019). The central question goal of PE is to infer the parameters of a particular GW model source given the strain data Christensen & Meyer (2022); Thrane & Talbot (2019)recorded by instruments like LIGO ?, Virgo ? and KAGRA ?. In the standard compact binary coalescence (CBC) scenario, this could mean inferring intrinsic parameters such the masses and spins of the eventcompact objects, as well as extrinsic parameters such as their sky localization and distance from Earthto the event. PE is also used in testing applied to test general relativity (GR) , which the main question is then to check for deviations in parameters that are related to non-GR modifications and constrain deviations away from its predictions in observed data Abbott et al. (2016, 2019, 2021a). PE is a crucial step in GW science, since it translate translates characteristics of the strain data into astrophysically relevant quantities that can be used to constrain astrophysical phenomenon such as phenomena, including informing theories of binary evolution Abbott et al. (2021b) and measuring the properties of nuclear matter.

The PE results in the most recent catalog of GW events are powered by several There exist a number of prominent, community-developed PE codes, including `lalinference` Veitch et al. (2015), `pycbc` LALInference Veitch et al. (2015), PyCBC Inference Biwer et al. (2019), and `bilby` Bilby Ashton et al. (2019). These packages have been tested by a number of groups and are well regarded as the standard tools. While the standard 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 length of the event. These community tools take about a week to analyze shorter events such as binary black hole (BBH) mergers, whereas they can take up to a month on a computer cluster to analyze longer events such as binary neutron star (BNS) mergers. In the coming decade, there are planned upgrade upgrades for existing facilitiesand , as well as plans for next-generation detector detectors such as the Einstein Telescope (ET) Punturo et al. (2010) and the Cosmic

kwong@flatironinstitute.org

Explorer (CE) Abbott et al. (2017). These upgrades will increase the sensitivity of the detectors instruments 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, which are not designed to handle this many events.

In order to handle the future catalogaddress this, there are efforts from multiple groups developing tools to speed up the PE process. This includes methods that employ modern tools such as deep learning-based method methods based on deep learning networks pre-trained on a large collection large collections of waveforms Dax et al. (2021, 2022), as well as methods that simplify the computational challenge reduce the computational expense of classical PE by leveraging our knowledge about of GW signals Islam et al. (2022); Roulet et al. (2022). While these methods are promising avenues to tackle standard GW analysis tasks in the coming decade, particularly about events involving compact binary coalescence (CBC) for standard GW problems, particularly for CBCs in GR, they capitalize rely on assumptions that may not be valid for analysis hold for analyses involving additional physical effects such as lensing and beyond-GR analysisdeviations from GR, or may use approximations that do not compute the Bayesian likelihood exactly.

In this work, we present a light-weightedlightweight, flexible, and high performance framework to estimate infer GW event parameters in a fully-Bayesian analysis. Our framework implements the following major features techniques to achieve its performance:

1. Differential waveform modelsdifferentiable waveform models,
2. Normalizing flow enhanced MCMC sampler normalizing-flow enhanced Markov chain Monte-Carlo (MCMC) sampler,
3. Heterodyned likelihoodheterodyned likelihood,
4. Native support for accelerator native support for hardware accelerators.

The main advantage of our framework is it does not rely on specific models or assumptions about the problem to achieve its performance. This makes our method extensible to problems beyond the standard CBC analysis, without sacrificing accuracy for efficiency.

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

2. GRAVITATIONAL WAVE PARAMETER ESTIMATION

2.1. *Likelihood function*

The main objective in PE can be summarized as given data in the form of a time series of strain $\mathbf{d}$, find the distribution of the source parameters θ that is compatible with the data, i. e. $p(\theta|\mathbf{d})$of PE is to obtain a multidimensional posterior distribution $p(\theta \mid d)$ on parameters θ given strain data d. Such probability density represents our best inference of the source properties, an encodes all relevant information contained in the observed data. To compute this object, we can rewrite it using use Bayes' theorem , to write

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

where $\mathcal{L}(d|\theta)$ $\mathcal{L}(d \mid \theta)$ 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 some simple distribution, such as uniformly distributed in the component masses or a Gaussian distribution in the spins, or it could encode astrophysical information. Assuming the noise follows from a stationary is drawn from a Gaussian process, the log-likelihood in the case of GW for GW data is given by

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

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

$$\langle a||b\rangle = 4\Re \int \frac{a^*(f)b(f)}{\mathcal{S}_n(f)} df\ \mathrm{df}\,, \tag{3}$$

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

To compute the integral shown in eq.3Eq. (3), we need to evaluate a chosen waveform model $h(\theta)$ at a number of frequnecy frequency sample points. Evaluating This makes evaluating the likelihood function is often the most computationally intensive part of the PE. The most accurate waveform model is numerical relativity (NR), which

obtains waveforms by directly solving the Einstein equations numerically for a given system. However, depending on the source parameters, generating one time series of strain can take a day to half a year, which makes NR prohibitively expensive for PE. To circumvent this problem, there are several waveform "approximant" families families of waveform "approximants", including the IMRPhenom IMRPHENOM family Khan et al. (2016); García-Quirós et al. (2020), SEOB the SEOB family Taracchini et al. (2014), and the NR surrogate familyVarma et al. (2019a). For shorter events, such as a $30-30\ M_\odot$ BBHbinary black hole, one waveform call could take 10ms to $\sim 1s$ $\sim 1s$ [KW: verify number in lalsuite]. For longer events, such as a $1.4-1.4\ M_\odot$ BNS eventbinary neutron star, the evaluation time could go up to [KW: Fill]. Since one needs to evaluate the likelihood many millions of times during sampling[1],[1] the computational cost in evaluating the waveform accumulate accumulates and is the main reason of the long runtime of GW PE.

2.2. *Heterodyned likelihood*

Since the computational cost of evaluating a waveform model scales linearly with the number of sample points either in the time or frequency domain, the computational burden for longer-duration signals is often quite large. To reduce the computational cost, there are a number of methods to reduce the number of basis points one would need to compute the likelihood faithfully Field et al. (2011, 2014); Smith et al. (2016); Vinciguerra et al. (2017). We use heterodyned likelihood In this work, we use likelihood heterodyning Cornish (2021) (also named relative binning in Zackay et al. (2018))in this work.
.

The idea behind the heterodyned likelihood can be summarized as the followingfollows: the integrand in eq.3 Eq. (3) is a highly oscillatory function, so one has to sample the integrand with sufficiently dense sampling to compute the integral faithfully. The number of sample points needed would be much smaller if the integrand was smooth. Given a pair of waveform parameters θ and θ_0 that are close to each other, the waveforms generated using the pair of parameters are similar to each other, this means the ratio between the waveforms is a smoothly varying function. Given a reference waveform $h(\theta_0)$, we can exploit this similarity between waveforms to reduce the number of sample points needed to compute the likelihood for the set of θ that is similar to θ_0. We decompose the integrand into two parts: (1) a highly oscillatory part that depends only on the reference waveform given by θ and the data, hence it and hence only needs to be evaluated once; and, (2) a smoothly varying part that depends on the target waveform parameters θ, that needs to be evaluated for every new likelihood evaluation. Because the part that depends on the target waveform parameters is smooth, we can use far fewer sample points to compute the integral with sufficient accuracy.

One may be concerned by the accuracy of this approximation over the target parameter space, 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 should necessarily also differ significantly from the data. This means that the likelihood value for these waveforms should be significantly smaller than the likelihood of the waveforms that are similar to the reference waveform, and hence will not be relevant for the PE result. In practice, one will first optimize the likelihood function with full frequency resolution to obtain the reference waveform parameters, which can be run at a much lower cost compared to PE.

We now give a concise description of what we implement the implementation of this approach in our codehere. For ; for a more extensive derivation of heterodyned likelihood, we refer the reader to the reference Zackay et al. (2018). In the heterodyned likelihood framework, the two terms in eq 2 are given by

$$\begin{aligned}\langle d|h\rangle &\approx \textstyle\sum_b A_0(b) r_0^*(h,b) + A_1(b) r_1^*(h,b),\\ \langle h|h\rangle &\approx \textstyle\sum_b B_0(b)|r_0(h,b)|^2 + 2B_1(b)\Re[r_0(h,b) r_1(h,b)],\end{aligned}$$

involving h obtained by expanding Eq. (2) can be approximated as

$$\langle \mathsf{d} \mid \mathsf{h}\rangle \approx \sum_{\mathsf{b}} \left[\mathsf{A}_0(\mathsf{b})\, \mathsf{r}_0^*(\mathsf{h},\mathsf{b}) + \mathsf{A}_1(\mathsf{b})\, \mathsf{r}_1^*(\mathsf{h},\mathsf{b})\right], \tag{4a}$$

$$\begin{aligned}\langle \mathsf{h} \mid \mathsf{h}\rangle \approx \sum_{\mathsf{b}} \big[&\mathsf{B}_0(\mathsf{b})\, |\mathsf{r}_0(\mathsf{h},\mathsf{b})|^2 + \\ &2\mathsf{B}_1(\mathsf{b})\, \Re\{\mathsf{r}_0(\mathsf{h},\mathsf{b})\, \mathsf{r}_1(\mathsf{h},\mathsf{b})\}\big]\end{aligned} \tag{4b}$$

where b denotes the index of a *sparse* set of bins where over which the integrand will be computedon, ; $A_0(b)$, $A_1(b)$, $B_0(b)$, and $B_1(b)$ are the heterodyning coefficients computed using the data and the reference waveform, and; and, finally, $r_0(h,b)$ and $r_1(h,b)$ are the ratio ratios

[1] A typical PE run with Bilby takes $> 10^6$ likelihood evaluation to converge.

[1] A typical PE run with BILBY takes $>10^6$ likelihood evaluations to converge.

between the target waveform and the reference waveform at the center of the bin . To evaluate eq.4and its first derivative. For sufficiently fine bins, the ratio between the target waveform and the reference within a bin can be approximated by linear interpolation,

$$\mathsf{r(f)} = \frac{\mathsf{h(f)}}{\mathsf{h_0(f)}} = \mathsf{r_0(h,b) + r_1(h,b)(f - f_m(b)) + \cdots}, \quad (5)$$

where b is the index of a particular bin, $r_0(h,b)$ and $r_1(h,b)$ are the value and slope of the ratio at the center of the bin respectively, and $f_m(b)$ is the center 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 the value of $r(f)$ at the edge of the bin and inverting Eq. (5). To evaluate Eq. (4), we need to be able to first choose a binning scheme, then evaluate the coefficients given the data and the reference waveformand , and at last the ratio between the target waveform and the reference waveform at the center of the each bin.

Considering the phasing of a 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 $[f_{\rm min}, f_{\rm max}]$ is given by

$$\delta\Psi_{\rm max}(f) = 2\pi \sum_i (f/f_{*,i})^{\gamma_i} {\rm sgn}(\gamma_i), \quad (6)$$

where $f_{*,i} = f_{\rm max}$ for γ_i >= 0 $\gamma_i \geq 0$ and $f_{*,i} = f_{\rm min}$ for $\gamma_i < 0$. Given the relation shown in eq.6Eq. (6), we can choose the binning scheme which to divide the entire frequency domain of interests 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., $|\delta\Psi_{\rm max}(f_{\rm max}) - \delta\Psi_{\rm max}(f_{\rm min})| < \epsilon$.

Given a sufficiently dense binning scheme, the ratio between the target waveform and the reference within a bin can be approximated by linear interpolation

$$r(f) = \frac{h(f)}{h_0(f)} = r_0(h,b) + r_1(h,b)(f - f_m(b)) + \cdots,$$

where b is the index of a particular bin, $r_0(h,b)$ and $r_1(h,b)$ are the value and slope of the ratio at the center of the bin respectively, and $f_m(b)$ is the center 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 the value of $r(f)$ at the edge of the bin and invert eq. 5.

The final ingredient we need is the heterodyning coefficients given for the data and the reference waveform on the sparse bins, which are given by explicitly given by

$$A_0(b) = 4 \sum_{f \in b} \frac{d(f) h_0^*(f)}{S_n(f)} df\,\Delta\mathsf{f}, \quad (7)$$

$$A_1(b) = 4 \sum_{f \in b} \frac{d(f) h_0^*(f)(f - f_m(b))}{S_n(f)} df\,\Delta\mathsf{f}, \quad (8)$$

$$B_0(b) = 4 \sum_{f \in b} \frac{|h_0(f)|^2}{S_n(f)} df\,\Delta\mathsf{f}, \quad (9)$$

$$B_1(b) = 4 \sum_{f \in b} \frac{|h_0(f)|^2 (f - f_m(b))}{S_n(f)} df\,\Delta\mathsf{f}. \quad (10)$$

Note that the sum within each bin should be done with the same sampling rate as the data, i.e., the same one would do in the case without using without using the heterodyned likelihood.

To obtain a reference waveform, we currently use the DIFFERENTIAL EVOLUTION algorithm Storn & Price (1997) available in the SCIPY package ? 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.

2.3. *MCMC with Gradient-based sampler*

Given eq.2 Eq. (2) and the prior, one can evaluate the posterior density function, Eq. (1), over the entire parameter space of interest to obtain the most probable set of points values that are consistent with the data. However, direct sampling directly sampling this posterior quickly becomes intractable as the dimensionality of the parameter space increases beyond a few dimensions. Markov chain Monte Carlo (MCMC) 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 Markov 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 pro-

cess is usually repeated until a certain criterion is met, such as a Gelman-Rubin convergence statistic Gelman & Rubin (1992) lower than a thresholdcertain threshold, or simply after a certain 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 in regions where it is unlikely to generate the observed data. However, MCMC algorithms come with their own set of issues. To illustrate what difficulties an MCMC algorithm MCMC may face, we can examine one of the most vanilla MCMC algorithm, 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

$$q(\mathbf{x}, \mathbf{x_0}) = \mathcal{N}(\mathbf{x}||\mathbf{x_0}, \mathbf{C}), \tag{11}$$

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 with a constant variance. And the acceptance criteria and with a fixed variance. The acceptance criterion is defined as

$$\alpha(\mathbf{x}, \mathbf{x_0}) = \min\left(1, \frac{p(\mathbf{x})q(\mathbf{x_0}, \mathbf{x})}{p(\mathbf{x_0})q(\mathbf{x}, \mathbf{x_0})}\right). \tag{12}$$

We can see from eq.12 Eq. (12) that the acceptance rate is proportion 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 the chain with a target number of independent samples is suboptimal. There Consequently, there is often a tuning process before we run the MCMC algorithm to find the optimal tuning parameters, settings for the algorithm (in this example, the variance of the Gaussian, to ensure close to optimal) to ensure the best possible performance.

However, as we are dealing with higher dimensional 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 is in general making 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 to 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 evaluation 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, Metropolis-adjusted Langevin algorithm (MALA) Grenander & Miller (1994) place a unit Gaussian at the tip of the gradient vector at the current position,

$$\mathbf{x} = \mathbf{x_0} + \tau \nabla \log p(\mathbf{x_0}) + \sqrt{2\tau} N(0, \mathbf{I}), \tag{13}$$

where τ is the step size , which is a tuning parameter a step size chosen during the tuning stage. Compared to a Gaussian centers centered at the current location, the MALA transition kernel is more likely to propose a point into in the higher posterior density region because of the gradient term, which helps to boost the acceptance rate.

While transition kernels that use gradient information can help to improve the acceptance rate, computing the gradient of the posterior density function introduce introduces an additional computational cost, which is not necessarily beneficial in terms of sampling time. If one wants to compute the gradient information through finite differencing, the additional computation computational cost goes as at least $\sim \mathcal{O}(2n) \sim \mathcal{O}(2n)$, where n is the dimension of the problem. Jax allows On the other hand, automatic differentiation schemes like Jax 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. Having Thus, having access to gradient information through automatic differentiation is crucial to make the trade-off between using making gradient-based transition kernels and additional computation cost favorable favorable in terms of computing cost.

2.4. *Normalizing Flow enhanced sampling*

While gradient-based samplers have been shown to be superior in terms of performance when compared to other outperform gradient-free algorithms in many practical examples, there are still a number of applications, there remain classes of problems that most gradient-based samplers do not

solve well. [2] For example, target distribution that exhibits local correlation is hard to deal with, since by construction first order first-order gradient-based algorithms can only handle global correlation structure Betancourt (2017). 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). [2] Another example is multi-modality. If : if there are multiple modes in the target distribution, an independent chain 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 sample compared to estimate than the shape of each mode.

Moreover, before we can use the sampling chain to estimate the quantity posterior quantities we care about, the sampler often needs to first find the most probable region in the target space , which (known as the *typical set*); this is a common process that is often referred to as "“burn-in” " in the literature. This means 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 necessarily helpful in directly devoted to estimating the target quantities.

Normalizing flow is a neural-network based technique All the above issues can be mitigated by normalizing flows. Normalizing flows is a technique based on neural networks that aims at learning a mapping between from a simple distribution, such as a Gaussiandistribution, to a complex distribution, often given in the form of data samples from a target distribution 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 samples sample from it very efficiently, by first evaluating the simple distribution and then applying the learned mapping. The core equation of normalizing flow is given by flows is the coordinate transformation of probability distributions via a Jacobian, as given by

$$p_x(X) = p_z(Z) \left| \frac{\partial f}{\partial z} \right|^{-1}, \tag{14}$$

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 connect connects the two distributions, $x = f(z)$, to be learned by the normalizing flow. For a detailed discussion of the algorithm, we refer the readers to Kobyzev et al. (2019); Papamakarios et al. (2019).

As mentioned in the previous subsection, one of the main issue Working in tandem, gradient-based sampler is it does not explore locally correlated features and multi-modality well . This is exactly where the normalizing flow model can help. Once 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 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 have generated samples, we can combine the samples and feed them to the normalizing flow , which learns the global 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. Note that However, since the independent chains are generated with contain the same number of samples, the normalization between posterior density represented by different chains is in general different from relative weight assigned to each chain will not represent the true target distribution . To adjust for this inaccuracy, we use the normalizing flow model (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*distribution. .

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

$$\alpha(\mathbf{x}, \mathbf{x_0}) = \min \left[1, \frac{\hat{\rho}(\mathbf{x_0})\rho_*(\mathbf{x})}{\hat{\rho}(\mathbf{x})\rho_*(\mathbf{x_0})} \right], \tag{15}$$

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. We can see

[2] To be specific, we are referring to sampling algorithms that use first-order derivatives here. Sampling algorithms that use the information of higher order derivatives such as manifold-MALA and Riemannian-HMC Girolami & Calderhead (2011) can in principle decorrelate local correlation in the target distribution, however, they often have instability issues when they are used on real-life applications, so they are not used often in practice.

[2] Sampling algorithms that use the information of 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.

From Eq. (15), 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 criteriaare 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` flowMC Wong et al. (2022). The pseudocode of the algorithm is given in Algorithm 1.

Algorithm 1: flowMC flowMC pseudocode

Input: initial position ip
Parameters: number of training loops nt, number of production loops np
Variables: current chain cc, current position cp, current NF parameters Θ
Result: *chains*

$cp \leftarrow ip$
`/* Training loop */`
for $i < nt$ **do**
  $cc, cp \leftarrow LocalSampling(cp)$
  $\Theta \leftarrow TuneNF(cc)$
  $chains, cp \leftarrow GlobalSampling(cp, \Theta)$
  $cc \leftarrow Append(cc, chains)$
`/* 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 accelerators such as GPUs and TPUs hardwares 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 CPUs 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 they are 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 harm the performance instead of improving performance than improve it.

In this Yet, in our work, the use of accelerators provides two independent perks that tremendously benefit the parameter estimation process. First, using accelerators allow us to run many independent chains at the same timeMCMC 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 use used a small number of chains, we are would be limited to the correlated samples from each chain , so we and we would have to run more sequential steps to get the same amount of independent data as compared to running more chains , where the former option does not benefit from parallelization but the latter does. In another wordsamples—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 help helps the normalizing flow learn the global landscape faster in Wall wall time.

Another benefit accelerator brings of accelerators is the parallel evaluation of waveforms. Since the waveform models being used in PE are approximants, they model we use can be evaluated at any given time or frequency independent of what has been generated before the time or frequency. This means predicting the independently, this means computing a waveform can be trivially parallelized over frequency bins. Together with the heterodyned likelihood, we can evaluate the likelihood at $\mathcal{O}(10^7)$ different locations on an Nvidia NVIDIA A100 GPU. The high throughput of likelihood evaluations unlocks the potential of the `flowMC` flowMC sampling algorithms.

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. We create a set of simulated signals and inject them into simulated noise, generated under the Gaussian noiseassumption using the designed PSD for every detectorstationary Gaussian noise. Then we run our pipeline given on the simulated data, and determine the credible interval at which the true parameters of the injected signals . Given are re-

covered. From the set of credible values, we can check whether the true value lies within a certain credible interval at a reasonable frequency. If the expected frequency: if our pipeline is perfectworking as expected, we should find 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 word, the percentiles of the true parameters should be uniformly distributed. Deviation from this behavior suggests the pipeline is either over-confident or too conservative ??.

We sample 1200 events from the following distribution of parameters detailed in table 1. The same distribution of parameters Table 1; the same distributions are used as the prior in the PE process. The simulated signals have a We simulate signals over 16 seconds segment length, and we project the signal on the LIGO Hanford, Livingston and Virgo detector. We used the designed sensitivity curve for each detector to generate the noise . We use of data, with a minimum frequency cutoff of 30 Hz with and a sampling rate of 2048 Hz. We draw noise from the design PSDs for the LIGO Hanford, LIGO Livingston (SimNoisePSDaLIGOZeroDetHighPower) and Virgo (SimNoisePSDAdvVirgo) detectors . 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 .

The We summarize the result of this injection-recovery campaign is shown in fig.1. We computed the quantile of in Fig. 1. This shows the cumulative distribution over injections of the quantile at which the true value lies for each marginalized distribution , then plotted the cumulative distribution of these quantiles. The shadow in the marginalized distribution of each parameter. The shaded band denotes the 95%credible interval drawn from -confident variation expected from draws from a uniform distribution with the same number of events. We can see most the injection-recovery result lies within the banddrawn from a uniform distribution, showing that most of the measured curves lie within this band, showing that our inference results agree well with a uniform distribution. There is a small deviation from a uniform distribution for the secondary spin χ_2, which is not too alarming since alarming given that we are computing the quantile for 11 parameters. Having one of them lying briefly outside the 95% band does not mean the inference is biasThis is what we expect if our pipeline is working as expected.

To further quantify how well our result agrees with a uniform distribution, we can compute the Kolmogorov-Smirnov test p-values p-values for each marginalized distribution . If the p-value is too low , . If this p-value were is low (with a threshold often chosen to be 0.05, then it means our result does not agree $p = 0.05$), then our result could be in tension with a uniform distribution, hence it could be biased.

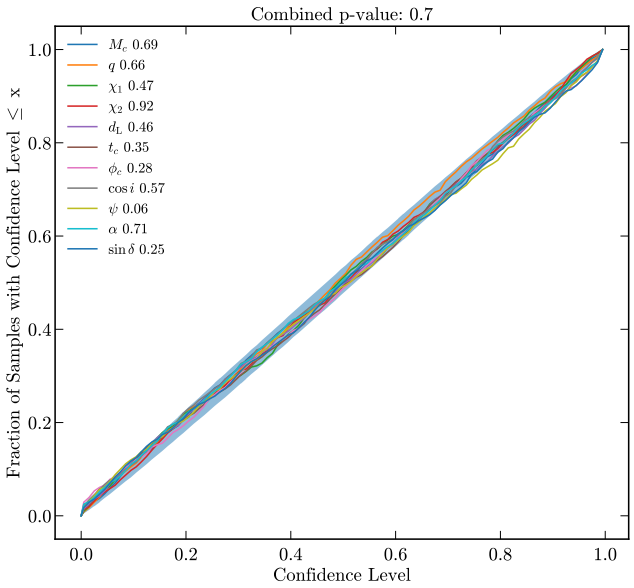

Figure 1. Cumulative distribution of the quantile of which the true value lies for each marginalized distribution. The shadow band denotes the 95% credible interval drawn from a uniform distribution with the same number of events as the injection campaign. The legend shows the p-values for each marginalized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

The p-values for each marginalized distribution . The p-values obtained for each parameter are shown in the legend of fig.1. We can see most Fig. 1. Most of them are well above the 0.05 $p = 0.05$ threshold, except for χ_2, which is mildly below the threshold. Once again, assuming these p-values p-values are drawn from a uniform distribution, given 11 11 draws (the number of parameters in our inference), it is not abnormal to have one of the parameters lies lying slightly outside the threshold. To assess whether this is expected, we can compute the combined p-value p-value for these 11 parameters, which is 0.47and find it to be $p = 0.47$. This shows our inference pipeline performs properly on simulated data . Indeed, the accuracy of our inference pipeline is consistent with other works. 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 our pipeline to analyze it to two real LIGO-Virgo events: GW150914 and GW170817. The prior used for the two events are shown in table 1. We use a We use the priors shown in Table 1, and take 4 seconds long segment s of data sampled at 2048 Hz for the GW150914 analysis, and a 128 seconds long segment s of data sampled at 4096 Hz for the GW170817 analysis. Both data segment ; strain data and PSDs for both events are fetched from GWOSC . Given the specific sampler setting we use?. For our

Parameters Parameter	Description	Injection	GW150914	GW170817
M_c	Chirp chirp mass $[M_\odot]$	$[10, 50]$	$[10, 80]$	$[1.18, 1.21]$
q	Mass mass ratio	$[0.5, 1]$	$[0.125, 1]$	$[0.125, 1]$
χ_1	Primary primary dimensionless spin	$[-0.5, 0.5]$	$[-1, 1]$	$[-0.3, 0.3]$
χ_2	Secondary secondary dimensionless spin	$[-0.5, 0.5]$	$[-1, 1]$	$[-0.3, 0.3]$
d_L	Luminosity luminosity distance [Mpc]	$[300, 2000]$	$[0, 2000]^\dagger$	$[1, 75]^\dagger$
t_c	Coalescence coalescence time [s]	$[-0.5, 0.5]$	$[-0.1, 0.1]$	$[-0.1, 0.1]$
ϕ_c	Coalescence coalescence phase	$[0, 2\pi]$	$[0, 2\pi]$	$[0, 2\pi]$
$\cos\iota$	Cosine cosine of inclination angle	$[-1, 1]$	$[-1, 1]$	$[-1, 1]$
ψ	Polarization polarization angle	$[0, \pi]$	$[0, \pi]$	$[0, \pi]$
α	Right right ascension	$[0, 2\pi]$	$[0, 2\pi]$	$[0, 2\pi]$
$\sin\delta$	Sine sine of declination	$[-1, 1]$	$[-1, 1]$	$[-1, 1]$

Table 1. Parameters used and the range of prior Prior ranges for parameters varied in the injection-recovery test, as well as the GW150914 , and GW170817 analyses. All priors assume a are uniform distribution with over the range tabulated ranges shown, except for the luminosity distance prior in the tableGW150914 and GW170817 analyses (†) for which we apply a prior unform in comoving volume. The coalescence time refers to a shift in time around relative to the geocenter trigger time. † Instead of uniform in luminosity distance, the distance prior used for the GW150914 and GW170817 analysis are uniform in comoving distance, with M_c refers to the range denote by the value in the table in MPCredshifted (detector-frame) chirp mass.

specific choice of sampler settings, we produce around an average of ∼2500 and 3500 **effective samples***effective samples*[3] for GW150914 and GW170817 in each analysis respectively, which is run on an Nvidia respectively. Running on an NVIDIA A100 GPU. The `wall` , the wall time for both event events is around 10 minutes. The chain data and the analysis scripts which generate the chains can be found on . Most of the Most of this time is spent on jit just-in-time (JIT) compilation of the code, and ; the actual sampling time was about 150s. Note that we have pre-computed is only ∼150 s. We pre-compute the reference waveform parameters used in hetetrodyne likelihood for the two events, so the time spent on solving for the reference waveform parameters the time for which is omitted in the `wall` time calculation. The chain data and the analysis scripts which generate the chains can be found in .

Since we are using the `IMRPhenomD` waveform and For comparison, we produce equivalent runs with BILBY, using the same exact data and priors. We use the DYNESTY sampler ??, with 1000 live points and other settings as shown in ?. We carry out these runs using PARALLEL BILBY (PBILBY) ? to distribute the computation over 400 Intel Skylake CPUs for each event. For the specific settings chosen, the open data release available on GWOSC does not provide the posterior

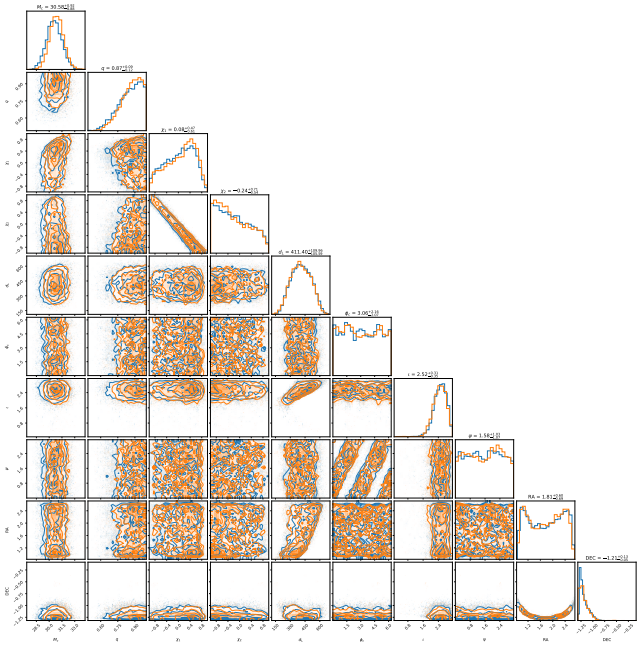

Figure 2. Posterior of GW150914 sampled using flowMC posterior computed by our code (blue) and Bilby BILBY (orange).

samples for this waveform, we run Bilby using the same prior and waveform model as a reference for comparison. From fig wall-time duration of each run was ∼2 h for GW150914 and ∼1 day for GW170817.

Figs. 2 and 3 , we can see the 11-dimensional posteriors results produced by our pipeline show that our posteriors are consistent with the results obtained using Bilby .

those produced by BILBY. For a quantitative comparisonbetween results obtained through our code and other existing

[3] Note that effective sample measures Effective samples here refers to the number of independent samples, which takes correlation between step into account, and it is not the same as the total number of generated samples . The divided by their correlation length; we compute the effective sample size is computed using `arviz`ARVIZ ?,https://python.arviz.org/en/stable/api/generated/arviz.ess.html.

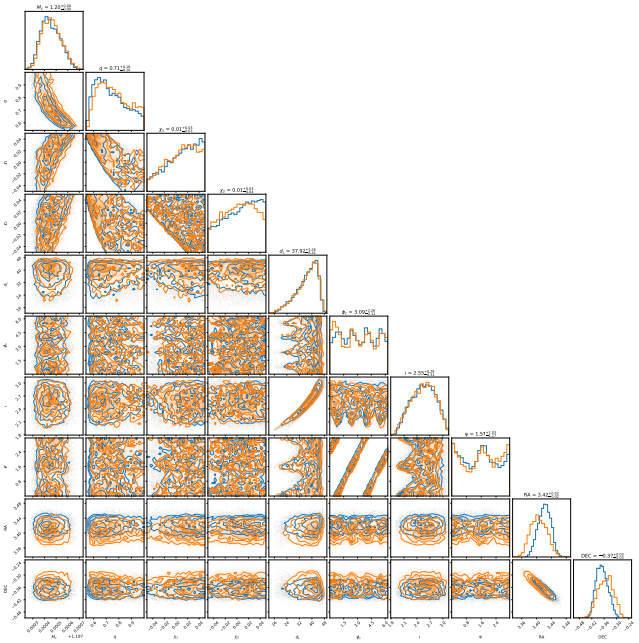

Figure 3. Posterior of GW170817 sampled using flowMC posterior computed by our code (blue) and Bilby BILBY (orange).

tools, we compute the Jensen-Shannon divergence of the marginalized distribution between our code and `Bilby`BILBY for the marginalized distribution for each parameter. The Jensen-Shannon divergence is a symmetric measure of the distance between two probability distributions, with a value of 0 indicating identical distributions and a value of ln 2nat representing the maximum possible divergence between two distributions.

[KW: Report JSO]

4. DISCUSSION

In this work, we present a PE pipeline for GW events that is efficient and flexible. Our package put together a number of innovations, including differentiable waveform models from `ripple` , heterodyned likelihood and normalizing flow enhanced sampler `flowMC`. We tested the robustness of our pipeline on a set of 1200 synthetic GW events, showing it is capable of handling catalog that will be available in the near future. We also demonstrate our pipeline can estimate the parameters of GW150914 and GW170817 in .

4.1. Comparison to other approaches

There are recent works from different groups on speeding have been several recent efforts to speed up parameter estimation of gravitational wave, including efficient reparameterization Islam et al. (2022); Roulet et al. (2022) and 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 get down to achieve minutes-scale parameter estimation with high fidelity, we would like to highlight the unique strength of this work, and the potential interplay between our work and others' workour approach possesses unique strengths, and may complement some of those other techniques.

Compared In contrast to Dax et al. (2021, 2022), we do not rely on require pre-training the neural network on a large collection of waveforms and noise realizationrealizations. This means that our algorithm can be immediately deployed as soon as new waveform models and noise models are available, our algorithm can already be deployed. Furthermore, our method is essentially at its core an MCMC algorithm, which 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 suffer from 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 error to errors to the inference results.

While our pipeline uses the sample samples generated by the local sampler for training, one can could also supply a pre-trained normalizing flow model to our pipeline to bypass the training stage. This can further reduce would have the advantage of further reducing the total runtimeof our inference pipeline. However, this may introduce potential ; 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.

Compared In contrast to Islam et al. (2022); Roulet et al. (2022), we do not rely on handcrafted reparameterization reparameterizations of the coordinate systems . If the used for sampling. If a useful reparameterization scheme is known ahead of time, a nice reparameterization is also encouragedit could trivially be implemented within our pipeline, potentially easing convergence. However, handcrafted reparameterization depends on the assumptions used in deriving the reparameterization scheme, which would inevitably run into limitations of use cases. Our work can be viewed as an automatic reparameterization powered by normalizing flow, while not being exact hence 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 general, the transformation discovered by the normalizing flow will only be approximate and hence not as efficient as an explicit reparameterization , the method proposed in this work of the problem; yet, our approach applies to a much border class of problems where clever reparameterizations 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 the reparameterization is known ahead of time. For the class of problems where Islam et al. (2022); Roulet et al. (2022)appliestreated in Islam et al. (2022); Roulet et al. (2022), we can incorporate the reparameterization scheme those reparameterizations directly into our MCMC pipeline to reduce the complexity of the problem, hence speeding up the training phase. On the other hand, if there is If there are limitations to the reparameterization that mean it cannot properly encompass part of the target posteriorthat is not included in the reparameterization scheme, the normalizing flow should still be able to learn to produce accurate samples efficiently.

The two relevant works avenues discussed here (machine learning and reparameterizations) represent two orthogonal directions one can take in building next generation tools in general. On PE tools. On the one hand, there are modern tools techniques such as deep learning that are very flexible and powerful, but may need to 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 simplify the problem, which relies on having good intuition of the problem. Both approaches rely on having a somewhat reasonable prior on how to approach the problem. The main difference between methods used in industrial products and scientific problems is scientific problems often try to address questions one may have not been answered before, hence it is supposed to be far away eventually depart from our prior knowledge, meaning one need to take capability to its desirable to design methods that generalize beyond the current problem into account while building the toolstate of knowledge robustly. Our work utilizes both reparameterization and deep machine learning, yet our method can be trivially extended to problems beyond standard GW analysis that reparameterization or deep learning alone may have trouble dealing with. We Beyond efficiency, we believe such flexibility is and robustness are crucial for building scientific tools.

There are some future developments we are working on . While IMRPhenomD Khan et al. (2016)

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, higher modesubdominants moments of the radiation, and 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 IMRPhenomPv2 Khan et al. (2019)and three-parameters NRSurrogate waveformsimplementations of IMRPhenomPv2 Khan et al. (2019), the precessing successor to IMRPhenomD, as well as the numerical relativity surrogate waveforms, including NRSur7dq4 Varma et al. (2019b). Going forward, we encourage the expect the use of autodifferentiation environments like Jax to become more prevalent in the waveform development communityto leverage autodiff environments such as Jax when constructing new waveform models . Having a differentiable waveform model is not only , 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 results, as detailed in [KW: cite].

While standard GW analysis goes CBC analyses go up to 17 dimensions. , non-standard GW PE problems can have more parameters, which could potentially lead to more complicated geometry in the target posterior that is hard to reparameterize. For example, Abbott et al. (2021a) introduces 10 extra parameters to modify controlling deviations in the post-Newtonian coefficients predicted in GR, which should be taken care at the same time during inference. 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 the potential in solving this problemin full at oncepromise in tackling this problem.

To perform realistic PE with our pipeline, the antenna pattern of the detector network needed to be taken into account. Our current code can perform parameter estimation for any combination of the Hanford, Livingston, and Virgo detectors ground-based detectors detectors, under the assumption of short signals such that signals are transient and their wavelength is short. The first condition guarantees that the effect of Earth's rotation can be ignored . For when computing antenna patterns, while the second means that we can treat the antenna patterns as frequency independent constants. These assumptions break for next-generation detector networks such as the Einstein Telescope and the detectors, whether on Earth or in space, like Cosmic Explorer, a differentiable version of their antenna pattern is neededEinstein Telescope and LISA; differentiable implementations antenna patterns for those detectors is work in progress.

The features we have implemented are the barebone version of parameter estimation. We do not include marginalization schemes 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, phase, and calibration lines marginalization parameters Thrane & Talbot (2019). Because of the already reasonable performance of the sampler on our sampler thanks to hardware accelerators, time and phase marginalization is not necessarycrucial, as the performance of our implementation is not significantly impacted by having two extra dimensions. Nevertheless, having less parameters in a PE is always appreciated, which fewer parameters cannot hurt in the future, so we are looking into including incorporating analytic marginalization schemes in our code as wellas other marginalization modes.

JaxWhen it comes to wall time, the just-in-time compilation of our code is the current limiting factor. While JAX's JIT compilation drastically reduces the computational time to evaluate the likelihood . Howeverquickens likelihood evaluations, it comes with a compilation overhead when the likelihood is evaluated for the first timesignificant compilation overhead before the first evaluation. We observe that the compilation time could depend depends on the device where the code will be run. This is run; this is expected since Jax leverage Accelerated Linear Algebra JAX leverages the ACCELERATED LINEAR ALGEBRA (XLA) takes advantage of compiler to take advantage of hardware accelerators, which means Jax that JAX needs to compile the code for the each specific device according to its architecture. On an Nvidia NVIDIA A100 GPU, the compilation overhead could go up to 8 minutes for the waveformwe are using. For our current waveform. Meanwhile, for the cases we have studied, the time needed to obtain converging results sampling on an A100 is about 2-3 minutes. This means the compilation overhead is dominating dominates the wall-clock time of the specific PE run we considered. To utilize our implementation to its full potential 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.

Another overhead we are looking into reducing is the time needed to find the reference waveform Besides compilation, there is in principle also overhead from finding the reference waveform used in for heterodyning the likelihood. Currently, we are using `differential evolution` Storn & Price (1997) available in `scipy` to find the waveform parameters which maximize the likelihood. Since `differnetial evolution` Since the DIFFERENTIAL EVOLUTION algorithm we currently use has not been implemented in JaxJAX, and the Jax JAX waveform we use is not compatible with the parallelization scheme in the `scipy` library, it takes SCIPY library, maximizing the likelihood currently takes us around 5 minutes [KW: Benchmark it in the code and make this number precise.] to find the reference waveform parameters for GW170817. There are two ways to improve the performance in finding the parameters of the reference waveform. reduce this time.

First, we can explore a different optimization strategy that is more compatible with the strength 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 which we intend to investigate in the futurewe are investigating.

Another way to reduce the time of finding the maximum likelihood waveform is to incorporate marginalization of Second, we may marginalize extrinsic parameters to reduce the dimensionality of the optimization problem. Currently, we simultaneously maximize all 11 GW parameters CBC parameters in our problem numerically, which is unnecessarily complicated and expensiveunnecessary. There are long-existingmarginalization schemes over extrinsic parameters, 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 marginalization schemes will reduce the time needed to find the reference waveform parameters by obtaining fixing the extrinsic parameters and by reducing the dimensionality of the optimization problemfor the intrinsic parameters.

One Finally, one important aspect of modern computing is scalability, meaning it is generally favorable if one can simply put more computing units in the same problem and reduce the Wall 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, more GPUs means we can run more independent chains at the same time, which can generate more samples faster. But ; second, and more importantly, as shown in this work and [KW: cite flowMC?], more independent chains also help with reducing the burn-in time. Parallelizing over the number of chain dimension is trivial and does not require much change to the current infrastructure. Another way more GPUs can helps is by allowing 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 helps would help

capture more complex geometry geometries in the target space, which can lead to more accurate results 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 in [KW: Quote speed], demonstrating the potential of our implementation on real data.

6. ACKNOWLEDGEMENTS

Thanks Will We thank Will M. Farr for illuminating discussions. The Flatiron Institute is a division of the Simons Foundation, supported through the generosity of Marilyn and Jim Simons. 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 (gwopenscience.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.

REFERENCES

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

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

Abbott, R., et al. 2021a. https://arxiv.org/abs/2112.06861

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

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

Betancourt, M. 2017, arXiv e-prints, arXiv:1701.02434. 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

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

Cornish, N. J. 2021, 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

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

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

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

Khan, S., Chatziioannou, K., Hannam, M., & Ohme, F. 2019, Phys. Rev. D, 100, 024059, doi: 10.1103/PhysRevD.100.024059

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

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

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

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

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

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

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

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