Differentiable and hardware-accelerated waveforms for gravitational wave data analysis

Thomas D. P. Edwards,[1,2,3] Kaze W. K. Wong,[4] Kelvin K. H. Lam,[4,5] Adam Coogan,[6,7,8] Daniel Foreman-Mackey,[4] Maximiliano Isi,[4] and Aaron Zimmerman[9]

[1]*William H. Miller III Department of Physics and Astronomy, Johns Hopkins University, Baltimore, Maryland 21218, USA*
[2]*The Oskar Klein Centre, Department of Physics, Stockholm University, AlbaNova, SE-106 91 Stockholm, Sweden*
[3]*Nordic Institute for Theoretical Physics (NORDITA), 106 91 Stockholm, Sweden*
[4]*Center for Computational Astrophysics, Flatiron Institute, New York, New York 10010, USA*
[5]*Department of Physics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong*
[6]*Ciela—Computation and Astrophysical Data Analysis Institute, Montréal, Quebec, Canada*
[7]*Département de Physique, Université de Montréal, 1375 Avenue Thérèse-Lavoie-Roux, Montréal, Quebec H2V 0B3, Canada*
[8]*Mila—Quebec AI Institute, 6666 St-Urbain, Suite 200, Montreal, Quebec H2S 3H1, Canada*
[9]*Weinberg Institute, University of Texas at Austin, Austin, Texas 78712, USA*

(Received 23 March 2023; accepted 19 June 2024; published 10 September 2024)

We propose the use of automatic differentiation through the programming framework `JAX` for accelerating a variety of analysis tasks throughout gravitational wave (GW) science. Firstly, we demonstrate that complete waveforms which cover the inspiral, merger, and ringdown of binary black holes (i.e., IMRPhenomD) can be written in `JAX` and demonstrate that the serial evaluation speed of the waveform (and its derivative) is similar to the `lalsuite` implementation in `C`. Moreover, `JAX` allows for graphics processing unit–accelerated waveform calls which can be over an order of magnitude faster than serial evaluation on a CPU. We then focus on three applications where efficient and differentiable waveforms are essential. Firstly, we demonstrate how gradient descent can be used to optimize the ∼200 coefficients that are used to calibrate the waveform model. In particular, we demonstrate that the typical *match* with numerical relativity waveforms can be improved by more than 50%. Secondly, we show that Fisher forecasting calculations can be sped up by ∼3–5× (on a CPU) with no loss in accuracy. This increased speed makes Fisher forecasting for a population of events substantially simpler. Finally, we show that gradient-based samplers like Hamiltonian Monte Carlo lead to significantly reduced autocorrelation values when compared to traditional Monte Carlo methods. Since differentiable waveforms have substantial advantages for a variety of tasks throughout GW science, we propose that waveform developers use `JAX` to build new waveforms moving forward. Our waveform code, `ripple`, can be found on GitHub website and will continue to be updated with new waveforms as they are implemented.

DOI: 10.1103/PhysRevD.110.064028

I. INTRODUCTION

The first detection of gravitational waves (GWs) [1] from inspiraling and merging compact objects (COs) has revolutionized our understanding of both fundamental physics and astronomy (e.g., [2–4]). Although the data volumes from GW detectors such as Advanced LIGO [5] and Virgo [6] are relatively small, analyzing the data is a computationally demanding task. In addition, this computational cost will substantially increase when next generation detectors come online [7–9].

The complexity begins even before data taking, since GW searches using the matched-filtering technique [10,11] require the generation of large banks of template waveforms. Once potential candidates are found, parameter estimation (PE) is performed to extract the detailed source properties of each event [12–19]. For binary black holes with nonaligned spins, this requires a Markov chain Monte Carlo (MCMC) on a 15-dimensional parameter space. More general binary inspirals, such as those involving neutron stars, can lead to a significant increase in dimension. Beyond these simple scenarios, more complex

waveform models with additional parameters may be used to include calibration errors [20,21], model the tides of neutron stars [22], and test for deviations from general relativity [23–28]. Finally, using the results of PE, population synthesis models constrain the progenitor systems from which the black holes we see merging today began their journey [4,29,30]. Overall, GW data analysis therefore requires significant computation. In this paper, we will argue that differentiable waveforms (and more generally differentiable pipelines) can play a useful role in alleviating this computational demand.

Derivatives are ubiquitously useful throughout data analysis tasks. For instance, during PE, derivative information can be used to guide an optimizer toward higher-likelihood values (e.g., using gradient descent [31]) or allow a sampler to rapidly explore parameter space [e.g., using Hamiltonian Monte Carlo (HMC) [32,33]]. Gradients are particularly valuable for high-dimensional spaces. Unfortunately, in the field of GW data analysis, analytic derivatives of the necessary quantities (such as the likelihood) have historically required a significant amount of work to obtain [34]. With waveforms only increasing in complexity, calculating analytic derivatives will become even trickier. Numerical derivatives also suffer from accuracy issues stemming from rounding or truncation errors. However, recent progress in automatic differentiation (AD) has shown promise in allowing general, fast derivative calculations for gravitational waveforms, with applications to constructing template banks [35] and computing the Fisher information for forecasting [36,37].

Automatic differentiation is a family of methods used to compute machine-precision derivatives with little computational overhead. AD's recent ascendance is primarily driven by its use in machine learning, particularly for derivative computations of neural networks which use gradient descent during training. The core idea of AD is that any mathematical function can be broken down into a small set of basic operations, each with a known differentiation rule.[1] The full derivative can then be constructed using the chain rule [38]. There are now a variety of AD implementations, most notably in deep learning frameworks such as `pytorch` [39] and `tensorflow` [40]. More general frameworks exist in `julia` [41,42], although `julia`'s limited use in GW analysis software precludes its general use. Here we make use of `JAX` [43] due to its easy integration with `python` libraries, seamless support for running code on different hardware accelerators, such as graphical processing units (GPUs), and its just-in-time (JIT) compiler, which can substantially accelerate code.

There are a variety of gravitational waveforms currently used in analysis pipelines. They are generally structured into different families, the most common of which are the effective one body (EOB) [44–54], the phenomenological inspiral-merger ringdown (IMRPhenom) [55–61], and numerical relativity surrogate (NRsurrogate) [62–64]. Of these, the IMRPhenom family serves as a natural starting point for an AD implementation in `JAX`. Models like the nonprecessing IMRPhenomD [58] model studied here and the precessing, higher-mode model IMRPhenomXPHM [59,60] are written in the frequency domain using closed-form expressions. This makes a `JAX` implementation that complies with the constraints of JIT compilation simple. NRsurrogate models, which interpolate directly over waveforms produced by numerical relativity (NR) simulations, are in principle also straightforward to implement in `JAX` and will be explored in the future. EOB waveforms on the other hand are produced by evolving the dynamics of an effective one body Hamiltonian, and are therefore more difficult to implement in `JAX`.[2] For EOB waveforms, frequency-domain reduced-order models may therefore be a more convenient target (e.g., [49]).

In this paper we argue that automatically differentiable waveforms may be a useful tool for the future of GW data analysis. In addition, we present `ripple`, a small GW `python` package which, at the time of writing, includes a `JAX` implementation of the IMRPhenomD waveform and will be continually updated with new waveforms as they are implemented. The remainder of this paper is structured as follows. In Sec. II we discuss the automatically differentiable IMRPhenomD waveform implemented in `ripple` and perform some benchmarks to demonstrate its speed and accuracy. In Sec. III we discuss three distinct applications using differentiable waveforms. Firstly, we illustrate how the coefficients that form part of the IMRPhenomD waveform model could be improved by high-dimensional fitting enabled by an automatically differentiable waveform. Secondly, we implement differentiable detector response functions and show that the speed of Fisher matrix calculations can be substantially accelerated using AD. Finally, we run an illustrative injection example using HMC to demonstrate that the autocorrelation of derivative samplers is substantially smaller than that of traditional MCMCs. The associated code can be found at `ripple` [66].

II. IMPLEMENTATION AND BENCHMARKING

A variety of waveform families have been developed to accurately model the GW emission from COs [67]. When the COs are relatively well separated, the dynamics of the system can be well approximated by a post-Newtonian

[1]Of course, nondifferentiable functions exist and care must be taken when treating these special cases.

[2]In particular, the dynamically chosen time step for solving ordinary differential equations (ODEs) accurately makes JIT compilation in `JAX` difficult. However, the `diffrax` package [65] has shown good performance for ODE solving in a variety of settings and could potentially be used to implement EOB waveforms in `JAX`.

expansion. However, close to merger, NR simulations are required to accurately model the binary. Unfortunately, these numerical simulations are computationally expensive and cannot be run in conjunction with data analysis. Approximate, phenomenological waveforms have therefore been constructed to enable relatively fast waveform generation at sufficient accuracy.

As mentioned in the previous section, the three major waveform families that have been developed to date are EOB, NRsurrogate, and IMRPhenom. EOB waveforms require one to model the binary system using a Hamiltonian and are typically slow to evaluate, whereas IMRPhenom waveforms are constructed with simple closed-form expressions. IMRPhenom waveforms are therefore ideally suited for AD, especially using `JAX`. In this paper we focus on the aligned-spin, circular-orbit model IMRPhenomD [57,58].

The implementation of IMRPhenomD in `lalsuite` [68] is in `C`, and therefore needs to be rewritten natively into `python` to be compatible with `JAX`. We have rewritten IMRPhenomD from scratch using a combination of pure `python` and `JAX` derivatives. In addition, we have restructured the code for readability and evaluation speed as well as exposing the internal fitting coefficients to the user (which we will use later in Sec. III).

To demonstrate that our implementation of IMRPhenomD is faithful to the `lalsuite` implementation, we start by defining the noise weighted inner product:

$$(h_1|h_2) \equiv 4\mathrm{Re}\int_0^\infty \mathrm{d}f \frac{h_1^*(f)h_2(f)}{S_n(f)}, \tag{1}$$

where S_n is the (one-sided) noise power spectral density (PSD) and h_1 and h_2 are the frequency-domain waveforms which are to be compared. We can then normalize the inner product through

$$[h_1|h_2] = \frac{(h_1|h_2)}{\sqrt{(h_1|h_1)(h_2|h_2)}}. \tag{2}$$

Now we are ready to define the *match* which is given by

$$\mathrm{m}(h_1|h_2) \equiv \max_{\Delta t_c, \Delta\phi_c} [h_1|h_2] \tag{3}$$

where Δt_c and $\Delta\phi_c$ are, respectively, the differences in time and phase of coalescence between the two waveforms. Finally, we can define the mismatch as

$$\mathcal{M}(h_1|h_2) \equiv 1 - \mathrm{m}(h_1|h_2). \tag{4}$$

Since the match is a measure of the difference between two waveforms, we can use it to demonstrate that the implementation of IMRPhenomD in `ripple` accurately matches the `lalsuite` implementation. For this comparison, we use the S_n presented in GWTC-2 [69] from the Livingston detector (the most sensitive of the detectors).[3] This is shown in Fig. 1, where we have calculated the $\mathcal{M}(h_1|h_2)$ across the entire parameter space.[4] Here, h_1 corresponds to the `ripple` waveform implementation and h_2 is the `lalsuite` implementation, evaluated at the same point in parameter space. From Fig. 1, it is clear that the `ripple` waveform matches with the `lalsuite` waveform close to numerical precision across the entire parameter space. In fact, the scale in Fig. 1 is clipped such that points which give $\mathcal{M} = 0$ (up to numerical precision) are plotted as the lowest value in the color bar.

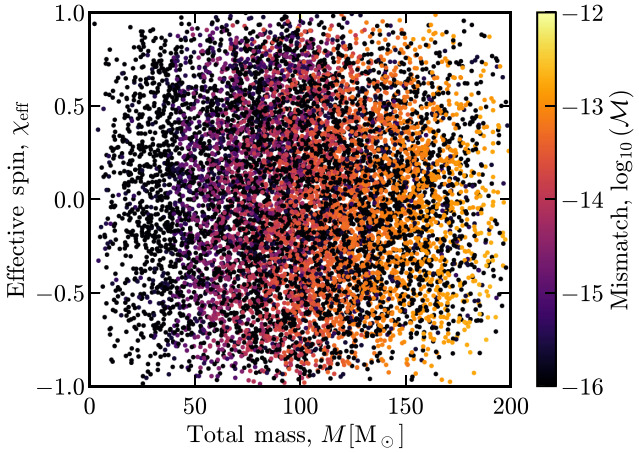

FIG. 1. Match between the `ripple` and `lalsuite` implementations of the IMRPhenomD waveform as a function of total mass and effective spin. Points which give $\mathcal{M} = 0$ (up to numerical precision) are plotted as the lowest value in the color bar. It is clear that the `ripple` waveform matches the `lalsuite` implementation many of orders of magnitude more than necessary for all data analysis tasks across the entire parameter space.

Note that in general relativity (GR) the total mass $M = m_1 + m_2$ of a binary black hole simply serves as an overall scale for the system, so that the frequency evolution can be trivially rescaled. The total mass only impacts the match because the chosen PSD and frequency limits fix a reference scale. If we instead use a flat PSD in Eq. (1) and rescale the frequency grid to units of Mf, all dependence with M seen in Fig. 1 vanishes. This figure instead illustrates a more realistic PSD where at low masses the waveform signal to noise is dominated by the inspiral but at high masses it is dominated by the merger.

There remains some slight deviation between the two waveform implementations at high total mass. This is partly due to the fact that cubic interpolators, which are used

[3]See https://dcc.ligo.org/LIGO-P2000251/public.

[4]Specifically, we use 10^4 points with varying component masses m_1, m_2 and spin parameters χ_1, χ_2 in the ranges $m_{1,2} = (1, 100)M_\odot$ and $\chi_{1,2} = (-1, 1)$. In addition, we evaluated the waveforms on a frequency grid from 32 to 1024 Hz with frequency spacing $\Delta f = 0.0125$ Hz.

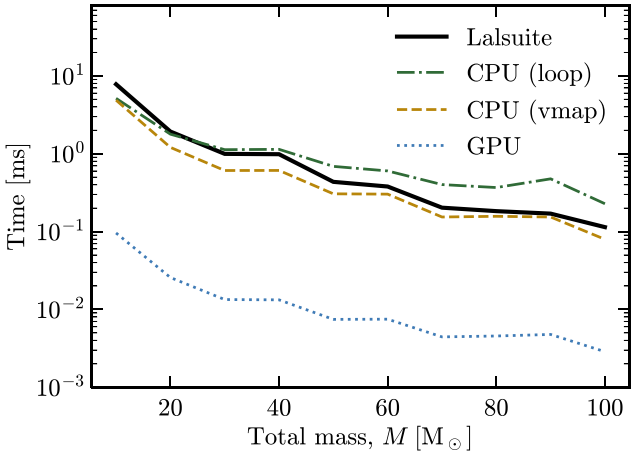

FIG. 2. Benchmarking the evaluation time of a single IMRPhenomD waveform as a function of the total mass. As the total mass increases, the time to merger decreases leading to a coarser frequency grid and shorter evaluation times. The details of the benchmarks are given in the text.

within IMRPhenomD to calculate the ringdown and damping frequencies, are not currently supported in `JAX`. Instead, we initially use `scipy`'s cubic interpolator to create a fine grid of 5×10^5 values, which we then linearly interpolate during waveform generation. Unfortunately, we cannot make the initial cubic interpolation arbitrarily fine as this would add additional computational overhead when loading the data during waveform evaluation. Note, however, that the differences are well below the accuracy requirements of the waveforms and will have no noticeable effect for realistic data analysis tasks. This is demonstrated in Fig. 4 where the mismatch between NR waveforms and IMRPhenomD is always above 10^{-5}.

For maximum utility, a waveform needs to be fast to evaluate. Fortunately, the IMRPhenom waveforms are constructed from simple closed-form expressions which are computationally efficient. Even though the `lalsuite` implementation of IMRPhenomD is written in `C`, the serial evaluation of the waveform in `ripple` is comparably fast. The evaluation time of a single waveform (averaged over 10^3 calls) is shown in Fig. 2 as a function of the total mass. In particular, we use 16 and 1024 Hz for the lowest and highest frequencies respectively. To calculate the frequency spacing, we first estimate the length of time taken for the binary to merge using Eq. (29) from [70]. Following LIGO convention, we take the closest factor of two higher than this time estimate as T. The frequency spacing is then given as $\Delta f = 1/T$.

For the CPU benchmarks we use a MacBook Pro with an M1 Max Apple Silicon processor. Although similar on a CPU benchmark, `JAX` has a few key advantages. First, its ability to JIT compile allows for significant performance gains (the above benchmark is already JIT compiled). Second, automatic vectorization can be achieved using the `vmap` function which leads to a noticeable speedup when evaluating a batch of waveforms. Finally, `JAX` can natively run on a GPU which allows for highly parallellized waveform evaluations.

Again, performing the same benchmark as above on an NVIDIA A100 with 40 GB of memory, we find that waveform evaluations are over an order of magnitude faster than serial CPU evaluations. Generalizing `lalsuite` waveforms to run on a GPU would be a significant undertaking. Note that JIT for both CPU and GPU evaluations adds an additional overhead to the first evaluation of the waveform which is not included in the above benchmarks. Note also that the GPU benchmark does not include the time taken to move the evaluated waveform from the GPU to the CPU.

One of the primary aims of this paper is to show that waveform derivatives will also be highly valuable to data analysis tasks. AD provides two big advantages when it comes to evaluating derivatives compared to numerical differentiation. First, the accuracy of derivatives from AD is significantly more stable than finite-difference methods. In particular, finite differences suffer from both rounding and truncation errors, meaning that the user is required to *tune* the width over which the difference is taken. On the other hand, AD produces machine-precision derivatives with no tuning. Second, AD scales favorably with the dimensionality of the function. In particular, for every input dimension added, D, one would need to evaluate the function at least $2D$ times to calculate finite-difference derivatives for all input parameters. For reverse-mode AD [38], one only needs two function calls to evaluate the derivative of all input parameters, regardless of the number of dimensions.[5] In gravitational wave physics one typically wants to compute derivatives of a scalar quantity (e.g., the likelihood) with respect to the intrinsic parameters of the binary (e.g., the masses and spins). Reverse-mode AD is therefore ideal for this task. For a review of AD methods see [38]. Since the parameter space of GWs in GR has $\mathcal{O}(10)$ dimensions, the speed of derivative evaluation is less crucial than the numerical stability. However, this might change for waveforms in beyond-GR models, waveforms involving equation of state parameters for neutron stars, and models which account for calibration and waveform uncertainties. In these cases many more parameters can be added.

Overall, we argued that the IMRPhenom waveform family is well suited for AD, showing it explicitly for IMRPhenomD. Moreover, we have shown that our implementation of IMRPhenomD in `ripple` is accurate and quick to evaluate, especially when hardware acceleration is available. In the next section, we will discuss a variety

[5]Note that although the number of function calls is small, reverse-mode AD does add memory overhead. We have not found this to be limiting in any of the situations tested so far.

of potential use cases of automatically differentiable waveforms.

III. APPLICATIONS

Here, we illustrate how differentiable waveforms can contribute to the improvement of three core tasks in GW science. In this paper we primarily look at toy examples, leaving more careful analyses to future work. The three tasks discussed here cover a wide range of GW science, starting with waveform development all the way to Fisher forecasting and PE.

A. Fine-tuning waveform coefficients

Having an accurate waveform model is essential for many data analysis tasks throughout GW science. While waveforms generated using NR simulations are in principle the highest fidelity signal model, they are too computationally expensive to be used in any practical data analysis tasks. The community therefore utilizes waveform "approximants" (such as those discussed in the Introduction) which can be evaluated much faster and in regions not covered by numerical simulations, such as extremes of parameter space or earlier phases of the binary inspiral.

Waveform approximants generally have free coefficients which are calibrated to NR waveforms to achieve high accuracy. In the case of IMRPhenomD, there are 209 fitting coefficients used to capture the separate behavior of the amplitude and phase as a function of the mass ratio and spins.

Any inaccuracy in obtaining the fitting coefficients leads to a misrepresentation of the NR waveform, which can translate to systematic error in downstream data analysis tasks. For example, sufficiently large systematic errors in the waveform would cause the recovered source parameters to be biased in the case of PE.

Previously in the construction of IMRPhenomD [58], waveform coefficients for the amplitude and phase were fitted independently. Furthermore, IMRPhenomD is divided into three fitting segments: inspiral, merger, and ringdown. Each of these segments has its own set of fitting coefficients. After obtaining the fitting coefficient for individual segments, they are then "stitched" together such that both the phase and amplitude are continuous in the first derivative. The process of stitching introduces some additional inaccuracy in the waveform model, as the connections affect the originally fitted segments.

Since the coefficients for each segment are tuned individually, the correlations between different parameters are ignored, meaning that the provided best-fit solution may not be the global optimum. We therefore aim to improve the accuracy of the waveform by jointly fitting all coefficients at once.[6]

In general, optimization problems in a high-dimensional space benefit from having access to the gradient of the objective function. Since we can differentiate through the entire waveform model against the fitting parameters, one can use gradient descent to more efficiently find the local best fitting parameters.

The first step is to define a loss function that measures the goodness of fit of the current waveform coefficients. Here, we choose it to be the mismatch between the NR waveform and the approximant waveform:

$$\mathcal{M}(\lambda) = 1 - \mathrm{m}(h_\theta^{\mathrm{IMR}}(\lambda)|h_\theta^{\mathrm{NR}}), \tag{5}$$

where λ is a vector of the fitting coefficients, h_θ^{IMR} is the waveform generated by IMRPhenomD, and h_θ^{NR} is the waveform generated by the NR simulation. Given the loss function, we use gradient descent to update the fitting coefficients:

$$\lambda \leftarrow \lambda - \alpha \nabla \mathcal{M}, \tag{6}$$

where α is the learning rate and we can compute $\nabla\mathcal{M}$ using AD. We set α to be 10^{-6}.

To generalize the loss function to a collection of waveforms we use the average of the mismatch of individual waveforms, given by

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{M}_i, \tag{7}$$

where $\mathcal{M}_i$ is the mismatch of an individual training waveform and N is the total number of training waveforms used in the optimization.[7] This optimization is more difficult since we are now applying the same set of coefficients to waveforms with different intrinsic parameters, such as mass ratio and spins. From Eq. (7), we can see the averaging between waveforms with different intrinsic parameters implies there are trade-offs in performance for different regions of the intrinsic parameter space. Additionally, this means our best-fit points will generally depend on the distribution of training waveforms across the parameter space.

To evaluate Eq. (7) we use a flat PSD and a frequency array which is scaled by the total mass.[8] Since the NR waveforms do not have a total mass associated with them, we assign a fixed total mass to compare with the `ripple` waveform.

[6]For a further breakdown of how the optimization changes across the parameter space, see our accompanying paper [71].

[7]Of course, now we replace $\mathcal{M}$ with $\mathcal{L}$ in the gradient descent update (6).

[8]In particular, we use $Mf_{\mathrm{l}} = 2.5 \times 10^{-3}$, $Mf_{\mathrm{u}} = 1.2 f_{\mathrm{RD}}$ (f_{RD} is defined in [58]), and $M\Delta f = 2.5 \times 10^{-6}$ for the dimensionless lower, upper, and frequency spacing respectively. We fix M to be $50M_\odot$ throughout this section.

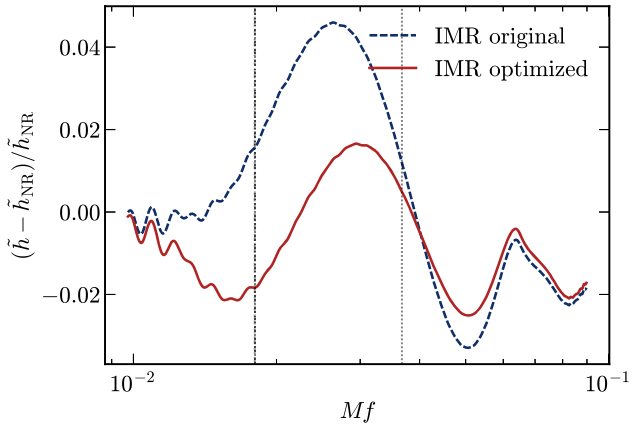

FIG. 3. Relative error between the numerical relativity and IMRPhenomD waveform amplitudes. The blue line shows the relative error of the IMRPhenomD model with the original model coefficients whereas the red line uses the IMRPhenomD model after optimization using gradient descent. Interestingly, the error is reduced across the frequency range, demonstrating that high-dimensional optimization is useful for all parts of the waveform.

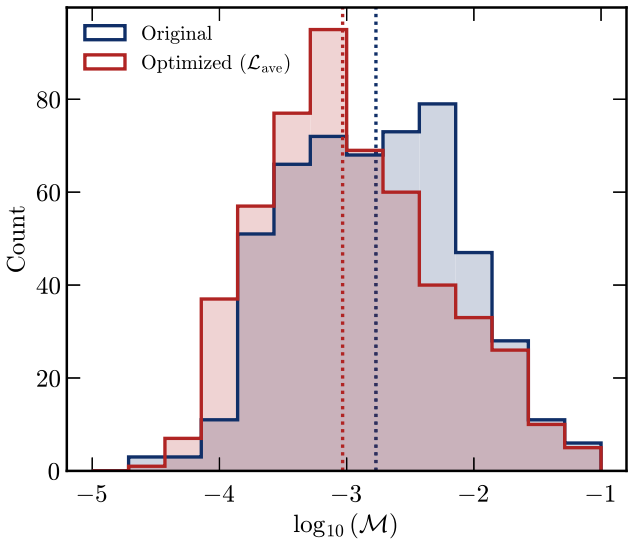

FIG. 4. Distribution of 536 log mismatches [see Eq. (5)] for the original (blue) and optimized (red) IMRPhenomD. Overall, the optimized distribution shifts to lower mismatches and therefore a better waveform model. More quantitatively, the median mismatch (shown as vertical dashed lines) is reduced by ∼50%.

For training, we use the publicly available subset (11 waveforms) of the 19 waveforms used in the original IMRPhenomD paper [58]. These 11 waveforms are taken from the SXS catalog [72]. We then run gradient descent, as described above, until the validation loss stops decreasing (the validation loss is calculated using the NR waveforms discussed below). Figure 3 shows the relative error (against a test NR waveform; see below) of the original and optimized waveform as a function of dimensionless frequency. The vertical dashed lines indicate the stitching points for the phase, i.e., when the inspiral is joined onto the merger. We can see the error of the optimized waveform is lower than that of the original waveform for most of the domain. In particular, the relative error in the merger region (in between the two vertical dashed lines) is decreased by half while other regions also show good improvement in accuracy.

In Fig. 4, we show the distribution of log mismatches for a set of test waveforms. In particular, we use 536 waveforms from the SXS catalog, i.e., all waveforms with aligned spins, $|\chi_{x,y}| < 5 \times 10^{-3}$, and eccentricity $<2 \times 10^{-3}$ [72]. One can see that the distribution of mismatches after optimization is generally shifted to lower mismatch compared to the original waveform. In particular, the median mismatch (shown as vertical dashed lines in Fig. 4) is reduced by ∼50%, indicating that our AD-assisted optimization procedure provides an improved implementation of the model.

While we focused on `IMRPhenomD` here, the ability to apply AD to the calibration parameters may assist in other approaches to calibration, such as that used for the aligned-spin EOB model in Bohé *et al.* [45]. Automatic derivatives, if implemented for EOB waveforms, could allow for the application of other sampling methods such as HMC, or possibly optimization over the entire set of NR waveforms at once.

B. Fisher forecasting

Forecasting the sensitivity of future experiments is a routine task in GW science. Due to its theoretical simplicity and evaluation speed, the Fisher matrix formalism (see, e.g., [73]) is commonly deployed to estimate how well a binary system's parameters could be measured. The Fisher matrix approach is built around the assumption of a Gaussian likelihood [74]. Although in practice this assumption is often violated for realistic detector noise, the results obtained using a Fisher analysis can provide quick and useful diagnostics in evaluating sensitivities for a variety of models and detector configurations.

Computing the Fisher matrix requires one to evaluate derivatives of the likelihood, which in turn involves derivatives of the waveform model and detector projection functions. AD is therefore perfectly suited for computing Fisher matrices accurately and efficiently. Forecasting with Fisher matrices for third generation detectors has already been extensively explored in [36,37]. Here we purely want to illustrate the simplicity and speed of AD for forecasting rather than providing new physics insights. We therefore consider a simple, three-detector setup corresponding to the two LIGO detectors in addition to Virgo.

The Fisher information matrix for a single detector is typically given by

$$\mathcal{I}^k_{ij} = (\partial_i h^k | \partial_j h^k), \tag{8}$$

where k indicates the detector, $\partial_i = \partial/\partial\theta_i$, and h^k is the strain measured by the detector which is given by

$$h^k(\theta) = F^k_+(\phi)h_+(\Xi) + F^k_\times(\phi)h_\times(\Xi). \tag{9}$$

Note that here we have separated out the extrinsic (ϕ) and intrinsic (Ξ) variables as well as introduced the detector projection functions for the plus and cross polarizations as F^k_+ and $F^k_\times$ respectively. Since we are considering a three detector setup we simply add the Fisher matrices from the individual detectors to get the combined Fisher matrix:

$$\mathcal{I}_{ij} = \mathcal{I}^{\text{Hanford}}_{ij} + \mathcal{I}^{\text{Livingston}}_{ij} + \mathcal{I}^{\text{Virgo}}_{ij}. \tag{10}$$

Finally, we invert the Fisher matrix to calculate the covariance matrix, which provides forecasted measurement errors and parameter covariances for a signal with parameters θ observed by the given detector network in the high signal-to-noise limit.

To illustrate the computational speed of computing Fisher matrices with AD, we consider a population of binaries and compute the 1σ chirp mass error.[9] Since the Fisher matrix approach is known to have both theoretical issues as well as numerical instabilities for low signal-to-noise events, we restrict our population to only nearby (high SNR) systems. A full list of the distributions used to generate the various parameters in our population is given in Table I. Additionally, we use 20 Hz, 2048 Hz, and 16 s for the minimum frequency, sampling frequency, and sample length. Our noise curves correspond to the design PSDs for LIGO Hanford, LIGO Livingston (SIMNOISEPSDALIGOZERODETHIGHPOWER), and Virgo (SIMNOISEPSDADVVIRGO).[10] The resulting population produces binaries with signal-to-noise ratios ranging from $\mathcal{O}(10 - 10^2)$.

The distribution of chirp mass errors from a population of 5×10^3 binaries can be seen in Fig. 5. We have verified that our errors agree with a separate dedicated Fisher forecasting code [75] to within 30%.[11] This demonstrates that AD can be used to accurately do Fisher forecasting for a large population of events.

Moreover, each error calculation (including computing the Fisher matrices for each detector and the inversion process) is substantially faster. In particular, we find that after compilation, each Fisher calculation takes approximately 1 s on a single computing core. GWBENCH [75], on the other hand, takes $\mathcal{O}$(3–5 s) for each Fisher calculation using the same detector setup and frequency grid.

[9]Note that we include all 11 parameters when computing the covariance matrix, and only plot the chirp mass error for illustrative purposes.

[10]See https://lscsoft.docs.ligo.org/lalsuite/lalinspiral/psds_8py_source.html.

[11]Note that since Ref. [75] is based on numerical derivatives, the agreement is not expected to be close to machine precision.

TABLE I. Priors for the 11-dimensional parameter space used for the Fisher forecasting population analysis in Sec. III B. $\boldsymbol{U}$ indicates a uniform distribution between the two variables in the brackets. The distance prior ensures that the binaries are uniformly distributed in volume.

m_1, m_2	$\boldsymbol{U}[20, 60]M_\odot$
χ_1, χ_2	$\boldsymbol{U}[-0.8, 0.8]$
D (uniform in volume)	[600, 900] Mpc
t_c	0.0
ϕ_c	0.0
Inclination angle, $\cos(\iota)$	$\boldsymbol{U}[-1, 1]$
Polarization angle, ψ	$\boldsymbol{U}[0, 2\pi]$
Right ascension, α	$\boldsymbol{U}[0, 2\pi]$
Declination, $\sin(\delta)$	$\boldsymbol{U}[-1, 1]$

This factor of 3–5 speedup is substantial considering the fact that a single core evaluation of the `ripple` waveform is similar to `lalsuite` which is used by GWBENCH. On a MacBook Pro with an M1 Max Apple Silicon processor, JIT compilation takes ∼70 s and computing chirp mass errors for the full population takes less than $\mathcal{O}$(2 h). As discussed above, performance can be further improved by utilizing hardware acceleration such as parallel GPU processing. AD therefore represents a fast and accurate way of performing population level analyses, and should be utilized for testing the capabilities of next generation detectors.

C. Derivative-based samplers—Hamiltonian Monte Carlo

After the search algorithms have constructed a list of confidently detected binaries, the next step is to sample

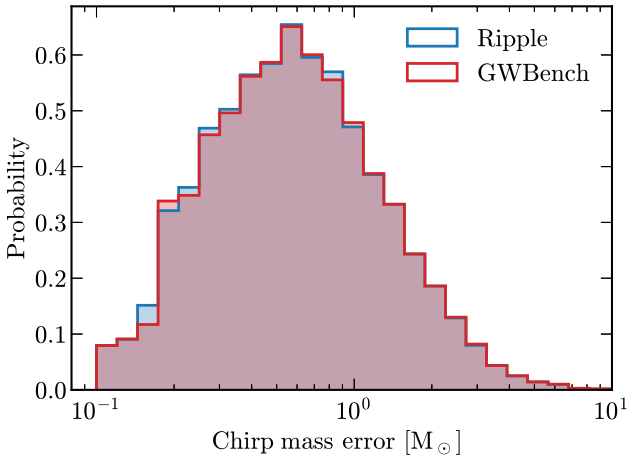

FIG. 5. Distribution of Fisher information chirp mass error for a population of nearby binaries. Automatic differentiation (AD) was used to compute the derivatives in Eq. (8). As emphasized in the text, after JIT compilation, each error calculation was between 3 and 5 times faster than a numerical derivative forecasting code GWBENCH [75].

from the posterior of each source parameter—so-called PE. To do this, one typically uses an MCMC or nested sampler [76–78]. Although robust, both MCMC and nested sampling are slow to converge and are known to perform poorly in high-dimensional parameter spaces. For example, sampling the 15-dimensional parameter space for a binary black hole (BBH) system can take $\mathcal{O}(10)$ h, while binary neutron star systems can take up to weeks. Dedicated fast samplers have been designed to get approximate posteriors on the sky localization to facilitate follow-up electromagnetic observations (e.g., `BAYESTAR` [79]). Moreover, a number of methods have been developed to speed up PE well below the numbers quoted above [80–86]. Nevertheless, these do not present the whole picture; fast, general PE therefore remains a key aim of GW data analysis.

A primary issue with both MCMC and nested sampling is that neither utilizes information about the likelihood's

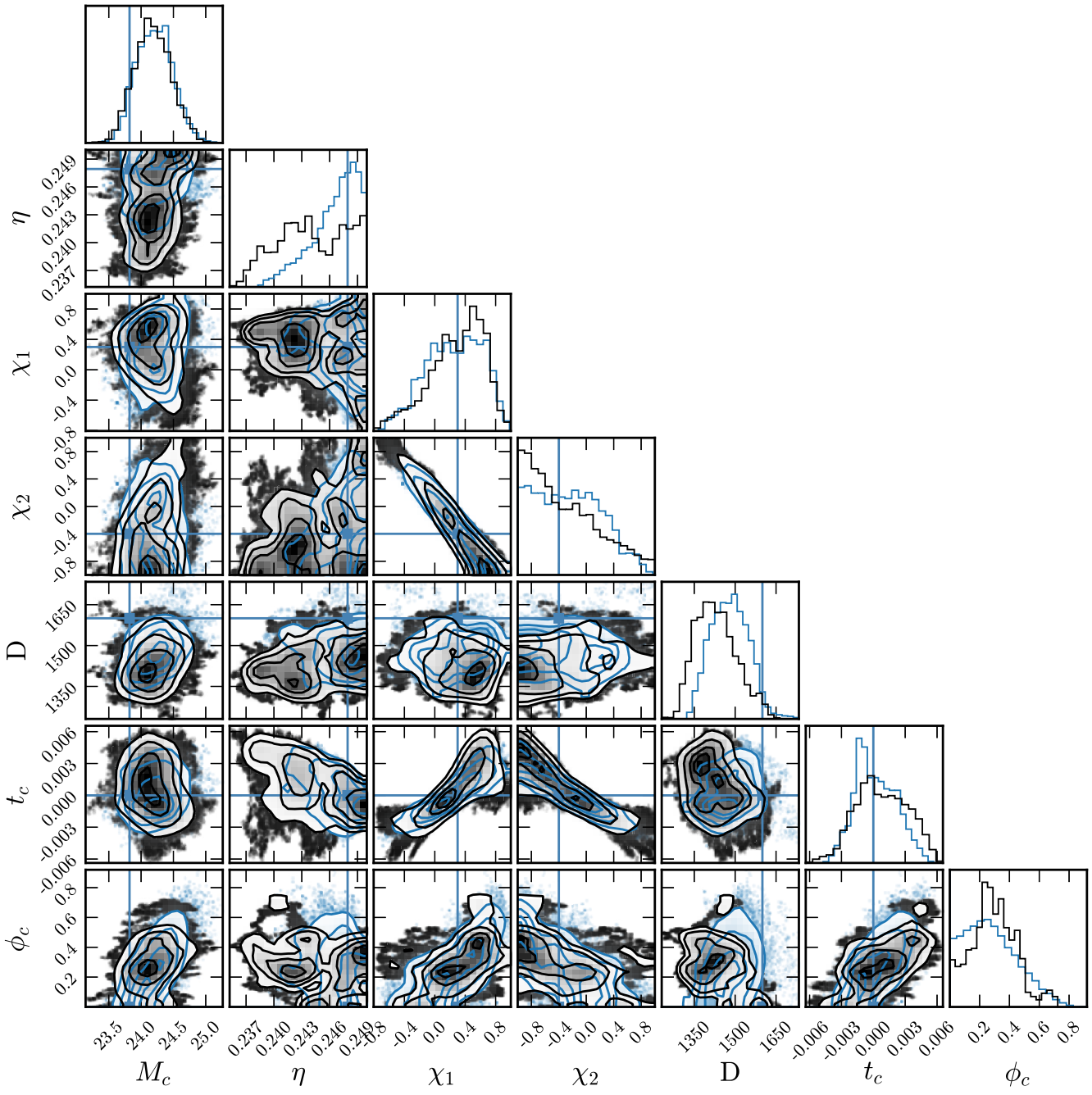

FIG. 6. Corner plot for the posteriors (see text for details) on simulated noise with injected signal using HMC (blue) and RWMH (black). Blue lines indicate the true values of the injection. Although not fully converged, it is clear that we find posteriors consistent with the injected parameters.

derivative and must therefore randomly walk toward areas of highest likelihood. Derivative-based samplers, on the other hand, have been shown to extrapolate well to higher dimensions although they sometimes come with their own drawbacks. Here we simply aim to demonstrate the utility of a derivative-based sampler and its efficiency on a small test problem. In particular, we will show that the autocorrelation of an HMC sampler is significantly lower than a traditional MCMC algorithm [87,88].

For our basic example we perform an injection recovery test on a seven-dimensional parameter space with the two LIGO detectors in our network. Our noise curves correspond to the design PSDs for LIGO Hanford and Livingston (SIMNOISEPSDALIGOZERODETHIGHPOWER). We use 20 Hz, 1024 Hz, and 16 s for the minimum frequency, sampling frequency, and sample length. More specifically, we generate Gaussian noise consistent with the measured PSDs for each detector and then inject a BBH signal with parameters: chirp mass $M_c = 23.82M_\odot$, symmetric mass ratio $\eta = 0.248$, primary spin parameter $\chi_1 = 0.3$, secondary spin parameter $\chi_2 = -0.4$, luminosity distance $D = 1.6$ Gpc, coalescence time $t_c = 0.0$, and coalescence phase $\phi_c = 0.0$.[12] Using a standard Gaussian likelihood, we then run the HMC sampler implemented in `flowMC` [89] for 2×10^5 steps and the random walk Metropolis Hastings (RWMH) sampler [90] for 1.5×10^6 steps (each with four randomly initialized independent chains). The number of steps and mass matrix used for each example was hand tuned to give good performance for the specific sampler. The additional steps for the RWMH sampler were required to achieve a similarly converged posterior.

We note at this point that neither pure HMC nor RWMH are the most modern versions of gradient- and nongradient-based samplers. For example, for gradient-based samplers one could use a No-U-Turn sampler [91] or the Metropolis-adjusted Langevin algorithm [92]. Traditional MCMC methods such as nested sampling [76–78] or the Affine Invariant MCMC Ensemble sampler implemented in `emcee` [93] will also lead to more efficient sampling of the posterior than basic RWMH. Here we instead seek to demonstrate the simplicity with which HMC can be implemented within a differentiable pipeline. In addition, as we discuss below, we find that a basic HMC algorithm will produce significantly more efficient sampling that RWMH, motivating further exploration of gradient-based samplers for GW PE [94].

In Fig. 6, the blue contours show the posterior recovered by HMC using the best chain (i.e., one that reached the highest log-likelihood values). The blue lines show the true parameters of the injected signal. From the one-dimensional histograms along the diagonal, it is clear that we consistently recover all seven parameters apart from ϕ_c (which is known to be measured poorly). This is expected since the injected binary is relatively nearby with an SNR of ∼13.

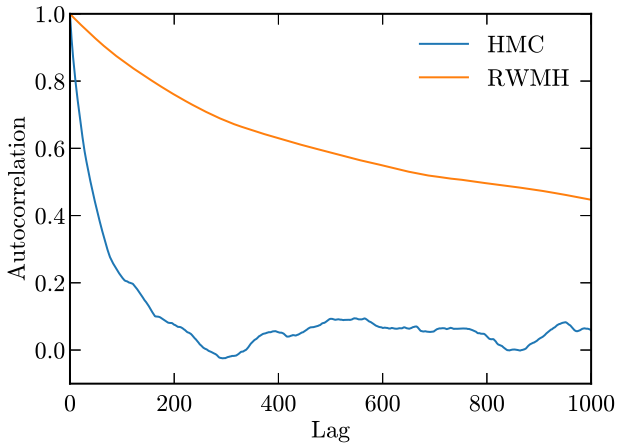

FIG. 7. Autocorrelation as a function of lag for HMC and RWMH on the same simulated data as discussed in Fig. 6. The smaller autocorrelation of the HMC leads to a larger number of independent samples and therefore a faster converging Monte Carlo.

Although further steps would be required to achieve a fully converged posterior, these chains are sufficient to show the increased efficiency associated with HMC. To further illustrate this, in Fig. 7 we plot the autocorrelation as a function of lag[13] for both the HMC and RWMH samplers. The HMC autocorrelation is substantially lower than that of the RWMH. In particular, we see that the autocorrelation reaches close to zero at around 300 lag, meaning that samples become efficiently uncorrelated. This is in contrast to RWMH which remains highly correlated even at 1000 lag. We therefore expect gradient-based samplers to converge significantly faster than typical samplers, especially in higher dimensions. In addition, we found that the effective number of samples [95], (a measure of the number of independent samples)[14] is between 2 and 7 times larger for HMC across the different dimensions of the parameter space.

In our companion paper we demonstrate that minute scale PE can be achieved by combining normalizing flows [89,96], GPU acceleration, and a derivative-based sampler [94]. We therefore expect `JAX` waveforms to be beneficial to future PE efforts in GW astronomy, particularly for low-latency pipelines and higher-dimensional analyses.

IV. DISCUSSION AND CONCLUSION

In this paper we introduced and discussed the various benefits of differentiable waveforms in `JAX` for GW data

[12]The remaining parameters (inclination angle ι, polarization angle ψ, right ascension α, declination δ) are set to $\pi/3$.

[13]Lag here corresponds to the number of steps between samples.

[14]Computed using ARVIZ (https://python.arviz.org/en/stable/api/generated/arviz.ess.html).

analysis. First, we demonstrated the speed and accuracy of our implementation of the aligned-spin IMRPhenomD waveform. In particular, we showed that it matches the `lalsuite` implementation to near machine precision and can be easily parallelized on a GPU. Parallelization on a GPU provides substantial speed increases; on an A100 40 GB GPU we found that waveform evaluations are over an order of magnitude faster than serial CPU evaluations. Second, we discussed three data analysis tasks which can all be substantially improved by utilizing derivative information of the waveform. In particular, we showed that AD can be used to fine-tune waveform coefficients,[15] perform efficient and accurate Fisher forecasting, and enhance the efficiency of sampling algorithms during parameter estimation.

Although we primarily discuss toy examples in this paper, each can be extended to the full data analysis task, some of which will be shown in upcoming papers [71,94]. Differentiable waveforms therefore represent a useful advancement toward efficient GW science.

In this paper, we primarily focused on the IMRPhenom family of waveforms as their closed form expression is perfectly suited for a `JAX` implementation. A differentiable NRsurrogate implementation is under development [97], but it currently seems difficult to implement EOB waveforms in `JAX`. As mentioned in the Introduction, the evolution of the Hamiltonian required to evaluate an EOB waveform is both inherently slow to differentiate and difficult to implement in `JAX`, although some progress has been made [65]. Since EOB methods are used to produce state-of-the-art waveforms for many applications, more work is required to see if a fast, differentiable implementation is possible.

Currently the biggest constraint to adopting differentiable waveforms is the need to rewrite the most commonly used waveforms into `JAX` (or pure `python`). In order to showcase the benefits of differentiable waveforms as quickly as possible, at the time of writing, we have only implemented an aligned-spin GW model (IMRPhenomD). We plan on adding a variety of different waveforms to `ripple` in the near future with the primary goal of reaching a `JAX` version of a fully precessing, higher-order mode waveform such as IMRPhenomXPHM [60]. Ideally, future waveforms should be implemented under an AD framework such as `JAX`. This would ensure that the community can easily utilize differentiability and hardware acceleration in the future.

[15]We note here that studying the extrapolation and bias of the fine-tuned waveform in regions beyond those covered by NR simulations is an essential task for future work.

ACKNOWLEDGMENTS

T. D. P. E. would like to thank Luca Reali extensively for his help with running `GWBENCH`. 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 work was supported by collaborative visits funded by the Cosmology and Astroparticle Student and Postdoc Exchange Network (CASPEN). T. D. P. E. is supported by the Horizon Postdoctoral Fellowship. A. Z. is supported by NSF Grants No. PHY-2207594. A. C. acknowledges funding from the Schmidt Futures Foundation.

[1] B. P. Abbott *et al.*, Phys. Rev. Lett. **116**, 061102 (2016).
[2] R. Abbott *et al.*, Phys. Rev. X **13**, 041039 (2023).
[3] R. Abbott *et al.*, arXiv:2112.06861.
[4] R. Abbott *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), Phys. Rev. X **13**, 011048 (2023).
[5] J. Aasi *et al.*, Classical Quantum Gravity **32**, 074001 (2015).
[6] F. Acernese *et al.*, Classical Quantum Gravity **32**, 024001 (2015).
[7] M. Evans *et al.*, arXiv:2109.09882.
[8] M. Maggiore *et al.*, J. Cosmol. Astropart. Phys. 03 (2020) 050.
[9] D. Reitze *et al.*, Bull. Am. Astron. Soc. **51**, 035 (2019).
[10] B. J. Owen, Phys. Rev. D **53**, 6749 (1996).
[11] B. J. Owen and B. S. Sathyaprakash, Phys. Rev. D **60**, 022002 (1999).
[12] G. Ashton *et al.*, Astrophys. J. Suppl. Ser. **241**, 27 (2019).
[13] C. M. Biwer, C. D. Capano, S. De, M. Cabero, D. A. Brown, A. H. Nitz, and V. Raymond, Publ. Astron. Soc. Pac. **131**, 024503 (2019).
[14] N. Christensen and R. Meyer, Rev. Mod. Phys. **94**, 025001 (2022).
[15] B. Farr and W. M. Farr, https://github.com/bfarr/kombine (2015).
[16] I. M. Romero-Shaw *et al.*, Mon. Not. R. Astron. Soc. **499**, 3295 (2020).
[17] J. S. Speagle, Mon. Not. R. Astron. Soc. **493**, 3132 (2020).
[18] J. Veitch *et al.*, Phys. Rev. D **91**, 042003 (2015).
[19] W. D. Vousden, W. M. Farr, and I. Mandel, Mon. Not. R. Astron. Soc. **455**, 1919 (2015).
[20] S. Vitale, C.-J. Haster, L. Sun, B. Farr, E. Goetz, J. Kissel, and C. Cahillane, Phys. Rev. D **103**, 063016 (2021).
[21] W. Farr, B. Farr, and T. Littenberg, Modelling calibration errors in CBC waveforms, LIGO Technical Report

No. T1400682-v1, 2014, https://dcc.ligo.org/LIGO-T1400682/public.
[22] B. P. Abbott *et al.*, Phys. Rev. X **9**, 011001 (2019).
[23] B. P. Abbott *et al.*, Phys. Rev. Lett. **116**, 221101 (2016).
[24] R. Abbott *et al.*, Phys. Rev. D **103**, 122002 (2021).
[25] M. Agathos, W. Del Pozzo, T. G. F. Li, C. Van Den Broeck, J. Veitch, and S. Vitale, Phys. Rev. D **89**, 082001 2014.
[26] K. G. Arun, B. R. Iyer, M. S. S. Qusailah, and B. S. Sathyaprakash, Classical Quantum Gravity **23**, L37 (2006).
[27] N. V. Krishnendu and F. Ohme, Universe **7**, 497 (2021).
[28] N. Yunes, K. Yagi, and F. Pretorius, Phys. Rev. D **94**, 084002 (2016).
[29] R. Abbott *et al.*, Astrophys. J. Lett. **913**, L7 (2021).
[30] K. W. K. Wong, K. Breivik, W. M. Farr, and R. Luger, Astrophys. J. **950**, 181 (2023).
[31] S. Ruder, arXiv:1609.04747.
[32] M. Betancourt, arXiv:1701.02434.
[33] R. Neal, in *Handbook of Markov Chain Monte Carlo*, edited by S. Brooks, A. Gelman, G. Jones, and X.-L. Meng (Chapman and Hall, New York, 2011), pp. 113–162.
[34] D. Keppel, A. P. Lundgren, B. J. Owen, and H. Zhu, Phys. Rev. D **88**, 063002 (2013).
[35] A. Coogan, T. D. P. Edwards, H. S. Chia, R. N. George, K. Freese, C. Messick, C. N. Setzer, C. Weniger, and A. Zimmerman, Phys. Rev. D **106**, 122001 (2022).
[36] F. Iacovelli, M. Mancarella, S. Foffa, and M. Maggiore, Astrophys. J. **941**, 208 (2022).
[37] F. Iacovelli, M. Mancarella, S. Foffa, and M. Maggiore, Astrophys. J. Suppl. Ser. **263**, 2 (2022).
[38] C. C. Margossian, arXiv:1811.05031.
[39] A. Paszke, S. Gross, F. Massa *et al.*, in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Red Hook, NY, 2019), pp. 8024–8035.
[40] M. Abadi, A. Agarwal, P. Barham *et al.*, `TensorFlow`: Large-scale machine learning on heterogeneous systems, https://www.tensorflow.org (2015).
[41] M. Innes, arXiv:1810.07951.
[42] J. Revels, M. Lubin, and T. Papamarkou, arXiv:1607.07892.
[43] J. Bradbury, R. Frostig, P. Hawkins *et al.*, `JAX`: Composable transformations of `python + NumPy` programs, 0.2.5, http://github.com/google/jax (2018).
[44] A. Albertini, A. Nagar, P. Rettegno, S. Albanesi, and R. Gamba, Phys. Rev. D **105**, 084025 (2022).
[45] A. Bohé *et al.*, Phys. Rev. D **95**, 044028 (2017).
[46] A. Buonanno, Y. Chen, and T. Damour, Phys. Rev. D **74**, 104005 (2006).
[47] A. Buonanno and T. Damour, Phys. Rev. D **59**, 084006 (1999).
[48] A. Buonanno and T. Damour, Phys. Rev. D **62**, 064015 (2000).
[49] R. Cotesta, S. Marsat, and M. Pürrer, Phys. Rev. D **101**, 124040 (2020).
[50] T. Damour, Int. J. Mod. Phys. A **23**, 1130 (2008).
[51] T. Damour, P. Jaranowski, and G. Schaefer, Phys. Rev. D **62**, 084011 (2000).
[52] A. Nagar, A. Bonino, and P. Rettegno, Phys. Rev. D **103**, 104021 (2021).
[53] S. Ossokine *et al.*, Phys. Rev. D **102**, 044055 (2020).
[54] A. Ramos-Buades, A. Buonanno, M. Khalil, and S. Ossokine, Phys. Rev. D **105**, 044035 (2022).
[55] C. García-Quirós, M. Colleoni, S. Husa, H. Estellés, G. Pratten, A. Ramos-Buades, M. Mateu-Lucena, and R. Jaume, Phys. Rev. D **102**, 064002 (2020).
[56] M. Hannam, P. Schmidt, A. Bohé, L. Haegel, S. Husa, F. Ohme, G. Pratten, and M. Pürrer, Phys. Rev. Lett. **113**, 151101 (2014).
[57] S. Husa, S. Khan, M. Hannam, M. Pürrer, F. Ohme, X. J. Forteza, and A. Bohé, Phys. Rev. D **93**, 044006 (2016).
[58] S. Khan, S. Husa, M. Hannam, F. Ohme, M. Pürrer, X. J. Forteza, and A. Bohé, Phys. Rev. D **93**, 044007 (2016).
[59] G. Pratten, S. Husa, C. Garcia-Quiros, M. Colleoni, A. Ramos-Buades, H. Estellés, and R. Jaume, Phys. Rev. D **102**, 064001 (2020).
[60] G. Pratten *et al.*, Phys. Rev. D **103**, 104056 (2021).
[61] R.Smith, S. E. Field, K. Blackburn, C.-J. Haster, M. Pürrer, V. Raymond, and P. Schmidt, Phys. Rev. D **94**, 044031 (2016).
[62] J. Blackman, S. E. Field, M. A. Scheel, C. R. Galley, C. D. Ott, M. Boyle, L. E. Kidder, H. P. Pfeiffer, and B. Szilágyi, Phys. Rev. D **96**, 024058 (2017).
[63] V. Varma, S. E. Field, M. A. Scheel, J. Blackman, D. Gerosa, L. C. Stein, L. E. Kidder, and H. P. Pfeiffer, Phys. Rev. Res. **1**, 033015 (2019).
[64] V. Varma, S. E. Field, M. A. Scheel, J. Blackman, L. E. Kidder, and H. P. Pfeiffer, Phys. Rev. D **99**, 064045 (2019).
[65] P. Kidger, Ph.D. thesis, University of Oxford, 2021.
[66] A. Coogan, T. Edwards, D. Foreman-Mackey *et al.*, `ripple`, 0.0.1, https://github.com/tedwards2412/ripple (2022).
[67] P. Schmidt, Front. Astron. Space Sci. **7**, 3132 (2020).
[68] LIGO Scientific Collaboration, LIGO Algorithm Library—`lalsuite`, free software (GPL), 10.7935/GT1W-FZ16 (2013).
[69] R. Abbott *et al.*, Phys. Rev. X **11**, 021053 (2021).
[70] H. Yu, J. Roulet, T. Venumadhav, B. Zackay, and M. Zaldarriaga, Phys. Rev. D **108**, 064059 (2023).
[71] K. K. H. Lam, K. W. K. Wong, and T. D. P. Edwards, Phys. Rev. D **109**, 124009 (2024).
[72] M. Boyle *et al.*, Classical Quantum Gravity **36**, 195006 (2019).
[73] N. J. Cornish, arXiv:1007.4820.
[74] M. Vallisneri, Phys. Rev. D **77**, 042001 (2008).
[75] S. Borhanian, Classical Quantum Gravity **38**, 175014 (2021).
[76] F. Feroz, M. P. Hobson, and M. Bridges, Mon. Not. R. Astron. Soc. **398**, 1601 (2009).
[77] J. Skilling, AIP Conf. Proc. **735**, 395 (2004).
[78] J. S. Speagle, Mon. Not. R. Astron. Soc. **493**, 3132 (2020).
[79] L. P. Singer and L. R. Price, Phys. Rev. D **93**, 024013 (2016).
[80] P. Canizares, S. E. Field, J. R. Gair, and M. Tiglio, Phys. Rev. D **87**, 124005 (2013).
[81] N. J. Cornish, Phys. Rev. D **104**, 104054 (2021).
[82] M. Dax, S. R. Green, J. Gair, J. H. Macke, A. Buonanno, and B. Schölkopf, Phys. Rev. Lett. **127**, 241103 (2021).
[83] T. Islam, J. Roulet, and T. Venumadhav, arXiv:2210.16278.
[84] N. Leslie, L. Dai, and G. Pratten, Phys. Rev. D **104**, 123030 (2021).

[85] J. Roulet, S. Olsen, J. Mushkin, T. Islam, T. Venumadhav, B. Zackay, and M. Zaldarriaga, Phys. Rev. D **106**, 123015 (2022).
[86] B. Zackay, L. Dai, and T. Venumadhav, arXiv:1806.08792.
[87] Y. Bouffanais and E. K. Porter, Phys. Rev. D **100**, 104023 (2019).
[88] E. K. Porter and J. Carré, Classical Quantum Gravity **31**, 145004 (2014).
[89] K. W. K. Wong, M. Gabrié, and D. Foreman-Mackey, J. Open Source Softwaare **8**, 5021 (2023).
[90] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).
[91] M. D. Hoffman and A. Gelman, arXiv:1111.4246.
[92] T. Xifara, C. Sherlock, S. Livingstone, S. Byrne, and M. Girolami, arXiv:1309.2983.
[93] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, Publ. Astron. Soc. Pac. **125**, 306 (2013).
[94] K. W. K. Wong, M. Isi, and T. D. P. Edwards, Astrophys. J. **958**, 129 (2023).
[95] R. Kumar, C. Carroll, A. Hartikainen, and O. Martin, J. Open Source Softwaare **4**, 1143 (2019).
[96] M. Gabrié, G. M. Rotskoff, and E. Vanden-Eijnden, Proc. Natl. Acad. Sci. U.S.A. **119**, e2109420119 (2022).
[97] T. Islam, K. W. K. Wong, M. Isi, and V. Varma (to be published).