Draft version January 24, 2023
Typeset using LaTeX **twocolumn** style in AASTeX631

`ripple`: 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]
Daniel Foreman-Mackey,[4] Maximiliano Isi,[4] and Aaron Zimmerman[8]

[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, NY 10010, USA*
[5] *Department of Physics, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong*
[6] *Département de Physique, Université de Montréal, 1375 Avenue Thérèse-Lavoie-Roux, Montréal, QC H2V 0B3, Canada*
[7] *Mila – Quebec AI Institute, 6666 St-Urbain, #200, Montreal, QC, H2S 3H1*
[8] *Center for Gravitational Physics, University of Texas at Austin, Austin, TX 78712, USA*

ABSTRACT

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 GPU-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 10% without any additional overhead. Secondly, we show that Fisher forecasting calculations can be sped up by $\sim 100\times$ (on a CPU) with no loss in accuracy. This increased speed makes population forecasting 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 at github.com/tedwards2412/ripple, and will continue to be updated with new waveforms as they are implemented.

1. INTRODUCTION

The discovery of gravitational waves (GWs) (Abbott et al. 2016a) from inspiraling and merging compact objects (COs) has revolutionized our understanding of both fundamental physics and astronomy (e.g. Abbott et al. 2021a,b,c). Although the data volumes from GW detectors such as Advanced LIGO (Aasi et al. 2015) and Virgo (Acernese et al. 2015) 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 (Maggiore et al. 2020; Reitze et al. 2019; Evans et al. 2021).

The complexity begins even before data taking, since GW searches using the matched-filtering technique (Owen & Sathyaprakash 1999; Owen 1996) 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 (Christensen & Meyer 2022; Speagle 2020a; Ashton et al. 2019; Romero-Shaw et al. 2020; Veitch et al. 2015; Biwer et al. 2019). For binary black holes with non-aligned 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.**(TE: add citations)** **(MI: assuming you don't mean TGR, since it's mentioned below, do you mean calibration and tides?: for cal we can cite W. Farr (2014); Vitale et al. (2021), and for tides maybe the BNS properties paper Abbott et al. (2019), unless you**

want to cite a bunch of tidal waveform papers.) Beyond these simple scenarios, more complex waveform models with additional parameters may be used to test for deviations from General Relativity (Arun et al. 2006; Agathos et al. 2014; Yunes et al. 2016; Abbott et al. 2016b, 2021d; Krishnendu & Ohme 2021). Finally, using the results of PE, population synthesis models constrain the progenitors systems from which the black holes we see merging today began their journey (Abbott et al. 2021e,c; Wong et al. 2022). [KW: Need more reference and some rewording]**(TE: added refs, happy?)** 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 significant 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 towards higher-likelihood values (e.g. using gradient descent (Ruder 2016)) or allow a sampler to rapidly explore parameter space (e.g. using Hamiltonian Monte Carlo (Neal 2011; Betancourt 2017)). 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 (Keppel et al. 2013). With waveforms only increasing in complexity, calculating analytic derivatives will become ever more tricky. 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 (Coogan et al. 2022b) and computing the Fisher information for forecasting (Iacovelli et al. 2022a,b).

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. There are now a variety of AD implementations, most notably in deep learning frameworks such as `pytorch` (Paszke et al. 2019) and `tensorflow` (Abadi et al. 2015). More general frameworks exist in `julia` (Innes 2018; Revels et al. 2016), although `julia`'s limited use in GW analysis software precludes its general use. Here we make use of `JAX` (Bradbury et al. 2018) due to its easy integration with `python` libraries, seamless support for running code on different hardware accelerators (e.g. graphical processing units) and its just-in-time (JIT) compiler, which can substantially accelerate code.

[AZ: Add refs here:]**(TE: Added a bunch, let me know if more are needed!)** 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) (Damour 2008; Buonanno et al. 2006; Buonanno & Damour 2000, 1999; Damour et al. 2000), the phenomenological inspiral-merger-ringerdown (IMRPhenom) (Husa et al. 2016; Khan et al. 2016; Hannam et al. 2014; Pratten et al. 2021), and numerical relativity surrogate (NRsurrogate) (Blackman et al. 2017; Varma et al. 2019b,a). Of these, the IMRPhenom family serves as a natural starting point for an AD implementation in `JAX`. Models like the non-precessing IMRPhenomD (Khan et al. 2016) model studied here and the precessing, higher-mode model IMRPhenomXPHM (Pratten et al. 2020, 2021) 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 straightfoward to implement in `JAX`. 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`. For EOB waveforms, frequency-domain reduced-order models may be a convenient target (e.g. Cotesta et al. 2020).

In this paper we argue that differentiable waveforms will be a vital component 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. 2 we discuss the differentiable IMRPhenomD waveform implemented in `ripple` and perform some benchmarks to demonstrate its speed and accuracy. In Sec. 3 we discuss three distinct applications using differentiable waveforms. Firstly, we illustrate how the fit coefficients that form part of the IMRPhenom waveform models could be improved by high

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

dimensional fitting enabled by a 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 Hamiltonian Monte Carlo to demonstrate that the autocorrelation of derivative samplers is substantially smaller than that of traditional MCMCs. Practically, this translates into much faster PE with no sacrifice in accuracy. The associated code can be found at `ripple` (Coogan et al. 2022a).

2. DIFFERENTIABLE WAVEFORMS

A variety of waveform families have been developed to accurately model the GW emission from COs (Schmidt 2020). When the COs are relatively well separated, the dynamics of the system can be well approximated by a post-Newtonian 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. Note that unlike NRsurrogate, both the EOB and IMRPhenom waveforms must be calibrated to NR waveforms. **(MI: I think this is weird to say: the surrogate is fully trained on ("calibrated to") NR)** 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 (Husa et al. 2016; Khan et al. 2016).

The implementation of IMRPhenomD in `lalsuite` is in C, and therefore needs to be rewritten natively into `python` to be compatible with `JAX`. We have re-written 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. 3).

To demonstrate 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. **(MI: I think it'd be good to define the mismatch $\mathcal{M} \equiv 1-\mathrm{m}$, since it's the standard quantity to discuss, and it's what we plot)**

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 (Abbott et al. 2021f) from the Livingston detector (the most sensitive of the detectors).[2] This is shown in Fig. 1, where we have calculated the $\mathrm{m}(h_1|h_2)$ across the entire parameter space.[3] 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. [AZ: strictly $= 0$? That seems impossible. Fmla has wrong labels for h's] In particular, the gray points at low masses are where $1-\mathrm{m}(h_1|h_2) = 0$ causing points to not appear in the log scale colorbar. **(MI: to Aaron's point, maybe say "0 up to numerical precision"?)**

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

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

[3] Specifically, we use 10^4 points 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 Hz to 1024 Hz with frequency spacing $\Delta f = 0.0125$ Hz.

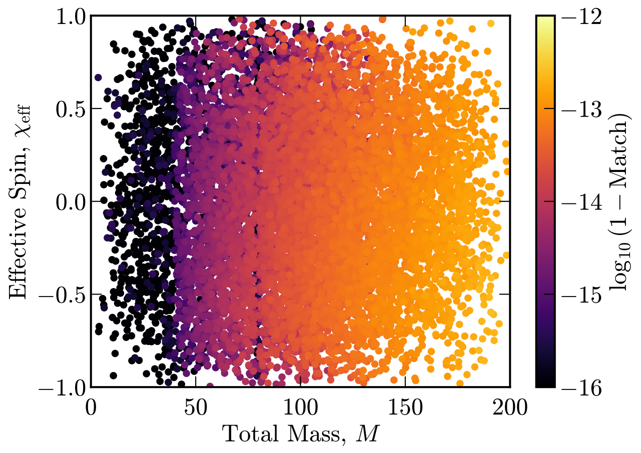

Figure 1. Match between the `ripple` and `lalsuite` implementations of the IMRPhenomD waveform as a function of total mass and effective spin. The gray points are where the match is exactly 1.0 and therefore would not appear in the log scale. It is clear that the `ripple` waveform matches the `lalsuite` implementation many of orders of magnitude more than necessary for all data analysis tasks across the entire parameter space. **(TE: Current plot looking better?)** [DFM: I think this figure could use some work. Maybe the points could be smaller to make them easier to resolve. And it could be interesting to clip the mismatch to something like 10^{-16} so that we don't lose the points with zero mismatch. Right now it looks less impressive!] **(MI: I agree. Furthermore, it looks like there's a subpopulation of darker points hiding under the brighter ones up to and around $M \sim 100$? is there a dependence on the mass ratio? it'd be good to comment on this, and maybe show a corner plot adding q to the mix. Also, M is lacking units (I assume $M_\odot$, but it should be shown in the label))**

by the inspiral but at high masses it is dominated by the merger. **(MI: it's important to say what PSD was used)**

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 within IMRPhenomD to calculate the ringdown and damping frequencies, are not currently supported in `JAX`. [DFM: What dimension of interpolation do you need? Maybe this is something we could implement...] **(TE: Its just one dimension, so is pretty easy. I tried the one from the cosmoJAX people and found that the match did slightly increase but it was a decent chunk slower (maybe a factor of 2 or so). So I just left it out. Would be happy to code up something and put it in if you think its necessary!)** 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.

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. Benchmarking on a MacBook Pro with an M1 Max Apple Silicon processor, we find that a single waveform evaluation takes ~ 0.5 ms for `lalsuite` (interfaced with python) and ~ 0.4 ms for `ripple`.[4] Although similar on a CPU benchmark, `JAX` has a few key advantages. First, its ability to JIT compile allows for significant performace gains (the above benchmark is already JIT compiled). Second, automatic vectorization can be achieved using the `vmap` function. Using `vmap` and performing the same benchmark as described above reduces the average evaluation speed to ~ 0.14 ms. Finally, `JAX` can natively on a GPU which allows for highly parallellized waveform evaluations.

Again, performing the same benchmark as above on a NVIDIA Quadro 6000, we find that on average waveform evaluations take $\sim$ 0.02, over an of magnitude faster than serial CPU evaluation. Generalizing `lalsuite` waveforms to run on a GPU would be a significant undertaking.

One of the primary argument of this paper is 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 are 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 in-

[4] For this benchmark we used 24 Hz and 512 Hz for the lowest and highest frequencies respectively. In addition, we used a frequency spacing of 0.2 Hz. We performed this benchmark by evaluating the waveform 10^4 times and taking the average evaluation time.

put parameters. For reverse-mode AD, one only needs two function calls to evaluate the derivative of all input parameters, regardless of dimension.[5] [KW: Maybe a bit more background on AD, the way we phrase it here sounds a bit too magical too me now.] Since the parameter space of GWs in GR has $\mathcal{O}(10)$ dimensions, the speed of derivative evaluation is less crucial than the 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 have demonstrated that the IMRPhenom waveform family is ideally suited for AD. 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 of potential use cases of differentiable waveforms.

3. APPLICATIONS

Here, we illustrate how three core tasks in GW science can be substantially improved through the use of differentiable waveforms. In the paper we primarily look at toy examples, leaving a 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.

3.1. *Fine-Tuning Waveform Coefficients*

(TE: We use in total 11 waveforms for the optimization. Original PhenomD uses 19 @Kelvin add details of where to find this.) (TE: Fig 2, plot was made after optimizing for ALL waveforms) (TE: 536 waveforms are subset of the SXS catalogue that are not precessing or eccentric. @Kelvin add details of where to find these things) (TE: None of the 536 are part of the training set) (TE: Used ALIGO design sensitivity for PSD)

[AZ: Broad comments: need to cite which catalog the waveforms came from if any. The number of waveforms used seems much larger than the number originally used to calibrate PhenomD. If that is true, we can't make a big deal about having a better mismatch compared to the original version, because of course optimizing over another set will improve performance. Say how many waveforms we use to optimize, how many we compare to, where they came from.]

(MI: this paragraph is redundant, I'd remove it) Having an accurate waveform model is essential for many data analysis tasks in GW, such as searching for signals and estimating source parameters given data. While waveforms generated using NR simulations are ~~commonly regarded as the most accurate,~~ [AZ: in principle the highest fidelity signal model,] they are too computationally expensive to be used in any practical data analysis tasks. To create waveform models that can be used in data analysis, there are a number of groups creating "approximants" of the full NR waveforms, which are essentially simpler ansatzes that ~~are only approximations but~~ can be evaluated much faster [AZ: and in regions not covered by numerical simulations, such as extremes of parameter space or earlier phases of the binary inspiral]. [AC: redundant – remove or merge with earlier material.]

[AZ: Waveform approximants generally have free coefficients which are calibrated to NR waveforms to achieve high accuracy.] ~~fitting coefficients that are tuned during the process of calibrating~~ 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. ~~The accuracy of fitting coefficients determines how well the approx~~ [AZ: Any] inaccuracy in obtaining the fitting coefficients means misrepresentation of the NR waveform, which can translate to systematic error in downstream data analysis task. 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 (Khan et al. 2016), waveform coefficients were fitted in segments with amplitude and phase separated. **(MI: this previous sentence sounds odd: do we mean that amplitude and phase were fitted independently?)** Furthermore, IMRPhenomD is divided in three fitting segments, inspiral, merger and ringdown. Each of these segments has their own set of fitting coefficients. After obtaining the fitting coefficient for individual segments, they are then "stitched" together such that they are continuous in the first derivative. The process of stitching introduces some additional inaccuracy in the waveform model, as the connections would affect the originally fitted segments.

The coefficients of the current IMRPhenomD implementation are tuned in subsets of parameters instead of all together. This means the tuning process ignores the correlation between different subsets of parameters, so

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

the best-fit solution obtained by fitting subsets of parameters may not be the global optimum compared to fitting all coefficients at once. By jointly fitting all coefficients, the correlation in fitting individual and connecting different segments can be accounted for during the optimization. Therefore, there is potential improvement of the accuracy by fitting all coefficients at once.

[AZ: Maybe comment on optimization method used for EOB waveforms? I think they do interesting Monte Carlo etc. Check what method is used for IMRPhenomXPHM?] **(MI: I agree. it'd be good to comment on whether this is still done like this in modern waveforms.)**

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 find the optimal fitting parameters all at once instead of fitting them in subsets.

The first step in fitting all coefficients is to define a loss function that measures the goodness-of-fit of the optimization procedure, which we choose it to be the mismatch between the NR waveform and the approximant waveform:

$$\mathcal{M}(\lambda) = 1 - \mathrm{m}(h_{\mathrm{IMR}}(\lambda)|h_{\mathrm{NR}}(\lambda)), \tag{4}$$

where λ is a vector of the fitting coefficients, h_{IMR} is the waveform generated by the IMRPhenomD and h_{NR} is the waveform generated by the NR simulation. **(MI: I think the mismatch should be defined above, right after the match is defined. We should also state what PSD and sampling parameters are used)** Given the loss function, we use gradient descent to find the optimal fitting coefficients.

$$\lambda \leftarrow \lambda - \alpha\nabla\mathcal{M}, \tag{5}$$

where α is the learning rate. We set α to be a very small number, i.e. 10^{-6}, since the initial guess already lies close to the desired optimum. We terminate the program after 30000 steps, in which λ has converged to the optimum. [DFM: We can't possibly know that it has converged to "the optimum". I'd sat just leave it at "We terminate the program after 30000 steps."] [AC: say something to explain why this is working. I.e.: does the loss initially drop and then plateau?]

As a demonstration, Fig. 2 shows the relative error of the original and optimized waveforms at different frequencies **(MI: for a particular set of parameters?)**. In this optimization procedure, we took NR waveforms h_{NR} stated in (Khan et al. 2016) as training data to fit waveform coefficients. [AC: how many are there?] We

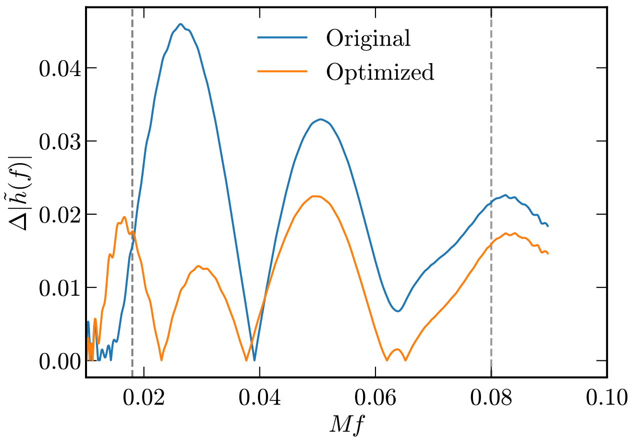

Figure 2. Relative error between the numerical relativity and IMRPhenomD waveform amplitudes. The blue line shows the relative error of the IMRPhenomD model with the original model coefficients where as the orange line uses the IMRPhenomD model after optimization using gradient descent. Interestingly, the error is reduced across the frequency range, demonstrating that high dimensional optimization is useful for all parts of the waveform.

can see the error of the optimized waveform is lower than that of the original waveform for most of the domain. In particular, the error in the early inspiral region is decreased by half while other regions also show good improvement in accuracy. [AC: so this is a waveform in the training set? And I don't follow exactly what you did. Is this just fitting a single waveform?] **(MI: I was also confused by this)**

We can generalize the fitting procedure described above to a collection of waveforms. In that case, the loss function is defined as the average of the mismatch of individual waveforms, given by

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{M}_i, \tag{6}$$

where $\mathcal{M}_i$ is the mismatch of individual training waveforms and N is the total number of training waveforms used in optimization. This optimization is more difficult since we are applying the same set of coefficients to waveforms with different intrinsic parameters, such as mass ratio and spins. From Eq. (6), we can see the averaging between waveforms with different intrinsic parameters implies there are trade-offs in performance in different regions of the intrinsic parameter space. [AC: this means your fit generally will depend on how the training waveforms are distributed over parameter space.]

In Fig. 3, we show the distribution of log mismatches. **(MI: for all parameters in the training set?)** One can see the distribution of mismatch **(MI: mean?)** after optimization has smaller mismatch compared to

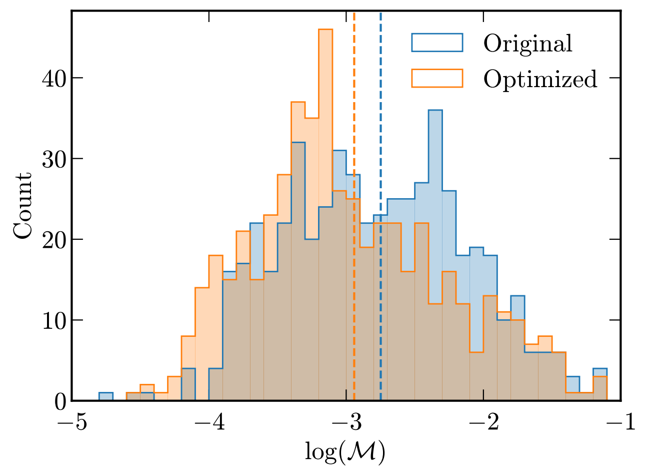

Figure 3. Distribution of 536 log mismatches (see Eq. 4) for the original (blue) and optimized (orange) IMRPhenomD. Overall, the optimized distribution shifts to lower mismatches and therefore a better waveform model. More quantitatively, the mean mismatch (shown as vertical dashed lines) is reduced by $\sim 11\%$.

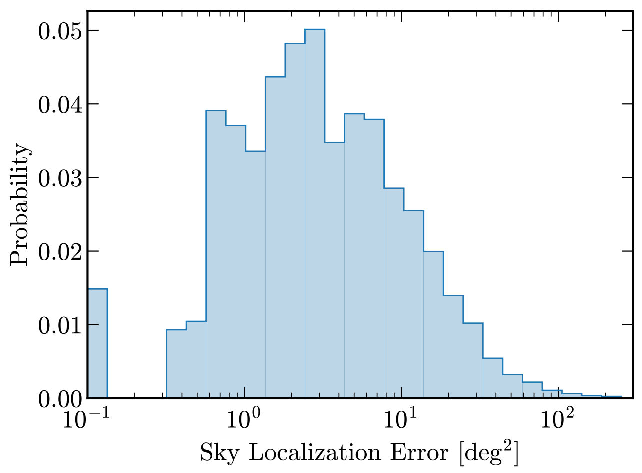

Figure 4. Distribution of Fisher information sky localization error for a population of nearby binaries. Automatic differentiation (AD) was used to compute the derivatives in Eq. 7. As emphasized in the text, after JIT compilation, each error calculation was around 100 times faster than a similar forecasting code `GWbench` (Borhanian 2021).

before optimization, showing the optimization improves the accuracy of the waveform model **(MI: on average?)**. While many waveforms improved their accuracy, some retain similar mismatches **(MI: "waveforms" is overused and obfuscates this discussion: it's indistinctly used to refer to the NR data, the approximants in general, and specific sets of parameters—I think some rewriting is needed, e.g., here we could say "While the mismatch generally improves for several parameter choices after training, it remains the same for others.")**. [AZ: I don't understand this claim. The spin degeneracy is present in the physics of the waveform; $\chi_{\rm eff}$ is just one way of parametrizing the spins that tries to reduce the degeneracy.] This phenomenon is mainly due to the reduced spin approximation used in IMRPhenomD, where spin degeneracy is introduced, hence restricting further improvements. In an upcoming paper, we will give a detailed discussion of how such an approximation scheme affects the optimization procedure. Nevertheless, the peak of the original waveform distribution has moved to the left end of the distribution, indicating that ~~the~~ [AZ: our AD-assisted] optimization procedure [AZ: provides an improved implementation of the model]. ~~Therefore, the new set of waveform coefficients can produce better waveforms for GW analyses.~~

3.2. *Fisher Forecasting*

Forecasting the sensitivity of future experiments is an essential task in GW science. [KW: LOL I won't say it is essential.] **(MI: routine?)** Due to its theoretical simplicity and evaluation speed, the Fisher matrix formalism 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. 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 (Iacovelli et al. 2022a,b). 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 [AC: I'm guessing in practice you compute a Hessian, so maybe write that out?]

$$\mathcal{I}^k_{ij} = (\partial_i h^k | \partial_j h^k)\,, \tag{7}$$

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

$$h^k(\theta) = F_+(\phi)h^k_+(\Xi) + F_\times(\phi)h^k_\times(\Xi)\,. \tag{8}$$

Note that here we have separated out the extrinsic (ϕ) and intrinsic (Ξ) variables as well as introducing the

m_1, m_2	$\boldsymbol{U}[1.0, 50]\,\mathrm{M}_\odot$
χ_1, χ_2	[-0.99, 0.99]
D	[500, 1000] Mpc
t_c	0.0
ϕ_c	0.0
Inclincation Angle, ι	$\boldsymbol{U}[0, 1]$
Polarization Angle, ψ	$\boldsymbol{U}[0, 1]$
Right Ascension, δ	$\boldsymbol{U}[0, 1]$
Declination, α	$\boldsymbol{U}[0, 1]$

Table 1. Priors for the 11 dimensional parameter space used for the Fisher forecasting population analysis in Sec. 3.2. $\boldsymbol{U}$ indicates a uniform distribution between the two variables in the brackets. [AZ: $\cos\iota \in (-1, 1)$, $\psi \in (0, 2\pi)$ (or π), RA is $\alpha \in (0, 2\pi)$, Dec is δ, let $\vartheta = \pi/2 - \delta$, then $\cos(\vartheta) \in (-1, 1)$]

detector projection functions for the plus and cross polarizations as F_+^k and $F_\times^k$ 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:

$$\mathcal{I}_{ij} = \mathcal{I}_{ij}^{\mathrm{Hanford}} + \mathcal{I}_{ij}^{\mathrm{Livingston}} + \mathcal{I}_{ij}^{\mathrm{Virgo}} . \tag{9}$$

Finally, we invert the Fisher matrix to calculate the covariance matrix, [AZ: which proves forecasted measurement errors and parameter covariances for a signal with parameters θ observed by the given detector network]. **(MI: I think Aaron means "provides" not "proves"; also, we might add "in the high SNR limit")** [AZ: Again we need to state and cite the PSDs we use for the three detectors.]

To illustrate the computational speed of computing Fisher matrices with AD, we consider a population of binaries and compute the sky localization error following Eq. (28) in Iacovelli et al. (2022a,b). **(MI: should we say a word about computing FMs with multiple detectors?)** 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 systems. A full list of the distributions used to generate the various parameters in our population are given in Tab. 1. The resulting population produces binaries with signal-to-noise ratios ranging from $\mathcal{O}(10 - 10^2)$.

The distribution of sky localization errors from a population of 10^3 binaries can be seen in Fig. 4. We have verified that our errors agree with a separate dedicated Fisher forecasting code (Borhanian 2021) to within X percent. This demonstrates that AD can be used to accurately produce population-level forecasts.

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 error calculation takes approximately half a second on a single computing core. `GWbench` (Borhanian 2021), on the other hand, takes $\mathcal{O}$(minutes) for each Fisher calculation using the same detector setup and frequency grid. This factor of over 100 speed up is substantial considering the fact that a single core evaluation of the `ripple` waveform is slower than the `lalsuite` call [AZ: used by `GWbench`]. [KW: Want to pull a number on the total time it took on generating the population result? I imagine this should turn the theorists on and send them into a fishing frenzy.] 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.

3.3. *Derivative Based Samplers - Hamiltonian Monte Carlo*

[AC: most acronyms were already introduced.] After the search algorithms have constructed a list of confidently detected binaries, the next step is to sample from the posterior of each sources parameters - so called PE. To do this, one typically uses a Markov Chain Monte Carlo (MCMC) or nested sampler (Skilling 2004; Feroz et al. 2009; Speagle 2020b). 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 BBH system can take $\mathcal{O}(10)$ hours, while BNS 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` Singer & Price 2016). [AZ: We should be careful here. Many new algorithms have sped this a lot. Cogwheel can do fast PE, DINGO is a normalizing flow code that is very fast when trained. There are likelihood acceleration methods like ROQ, multibanding, relative binning/heterodyning that accelerate the process compared to the numbers quoted above.] 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 derivative and must therefore randomly walk towards 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

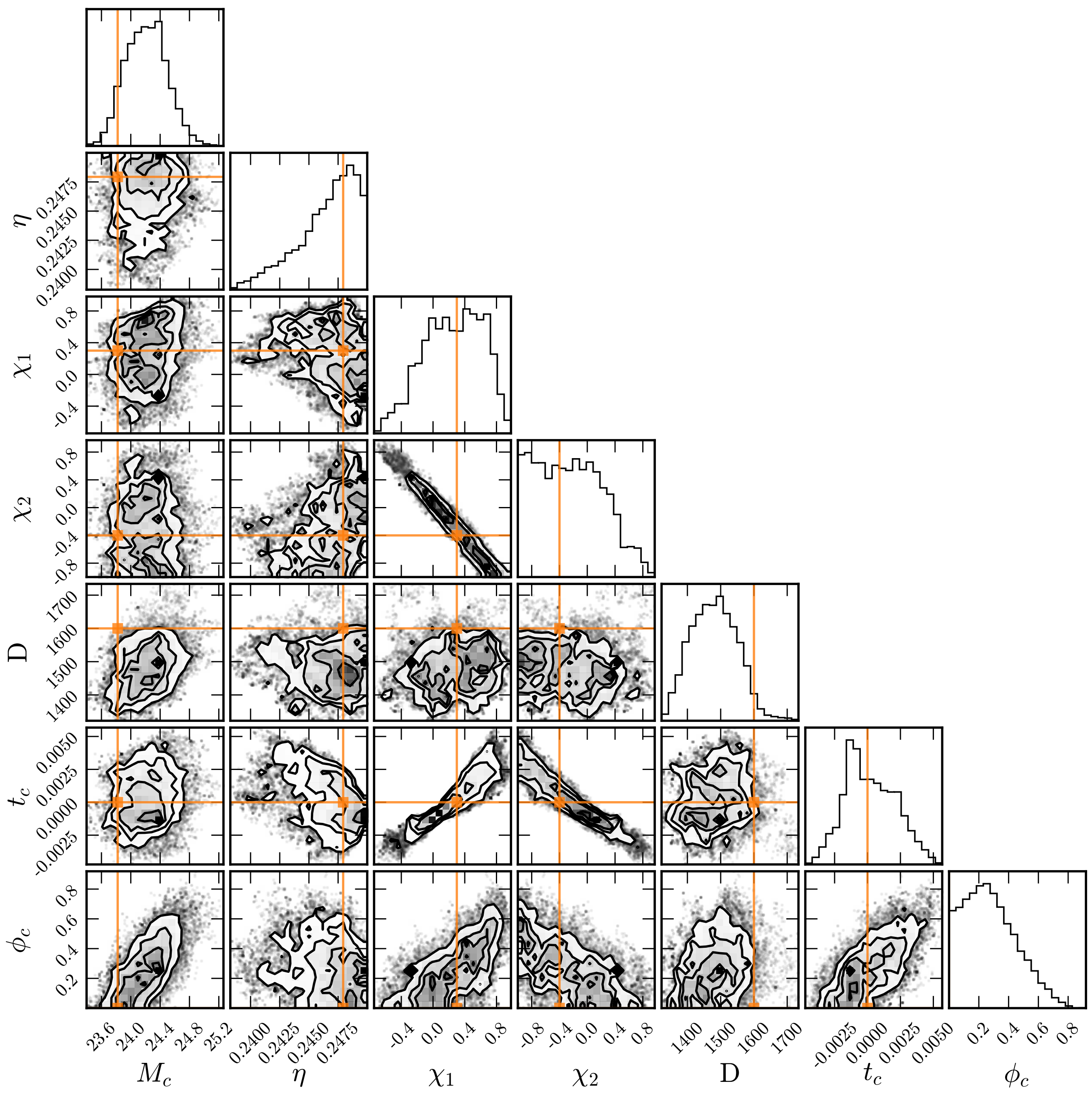

Figure 5. Corner plot for the posteriors from an HMC (see text for details) on simulated noise with injected signal. Orange lines indicate the true values of the injection. Although not fully converged, it is clear that we find posteriors consistent with the injected parameters. [AC: could apply a mild Gaussian kernel to smooth this for the sake of presentation. Are the parameter ranges due to your priors? How did you handle the boundaries in the HMC?]

demonstrate the utility of a derivative based sampler and their efficiency on a small test problem. In particular, we will show that the autocorrelation of a Hamiltonian Monte Carlo (HMC) sampler is significantly lower than a traditional MCMC algorithm. [AZ: Add citations to other HMC papers in GW land]

For our basic example we perform an injection recovery test on a seven dimensional parameter space with the same three detector network setup considered in § 3.2. [AZ: As before, say what PSD and cite.] More specifically, we generate Gaussian noise consistent with the measured PSDs for each detector and then inject a BBH signal with parameters: chirp mass $M_c = 23.82\,\mathrm{M}_\odot$, symmetric mass ratio $\eta = 0.248$, primary spin parameter $\chi_1 = 0.3$, secondary spin parameter [AZ: $\chi_2 = -0.4$], luminosity distance $D = 1.6\,\mathrm{Gpc}$, coalescence time $t_c = 0.0$, and coalescence phase $\phi_c = 0.0$.[6] Using a standard Gaussian likelihood, we then run the HMC sampler implemented in `flowMC` for ~~200000~~ 2×10^5 steps and the Gaussian random walk (GRW) sampler for ~~1500000~~ 1.5×10^6 steps (each with four randomly initialized independent chains). [AZ: What implementation of GRW? Can we cite something?] [AC: briefly summarize and give refs for flowMC and GRW. Is GRW Metropolis-Hastings with a Gaussian transition kernel? How'd you tune this? Why not use NUTS for HMC?] The number of steps and and mass matrix used for each example was hand tuned to give good performance for the specific sampler. [AC: not a huge deal, but why hand tune? There are automated ways to do this in e.g. numpyro with a warm-up adaptation phase.] The additional steps for the GRW sampler were required to achieve a similarly converged posterior. [AC: Why not plot its posterior?]

In Fig. 5, the grey contours show the posterior recovered using the best chain (i.e. one that reached the highest log-likelihood values). **(MI: why show a single chain?)** The orange shows 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. This is expected since the injected binary is relatively nearby. [AZ: Can we give the SNR?]

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. 6 we plot the autocorrelation as a function of lag for both the HMC and GRW samplers. The HMC autocorrelation is substantially lower than that of the GRW. 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 [AC: provide ref?] (a measure of the number of indepedent samples) is between 2 and 7 times larger for HMC across the different dimensions of the parameter space.

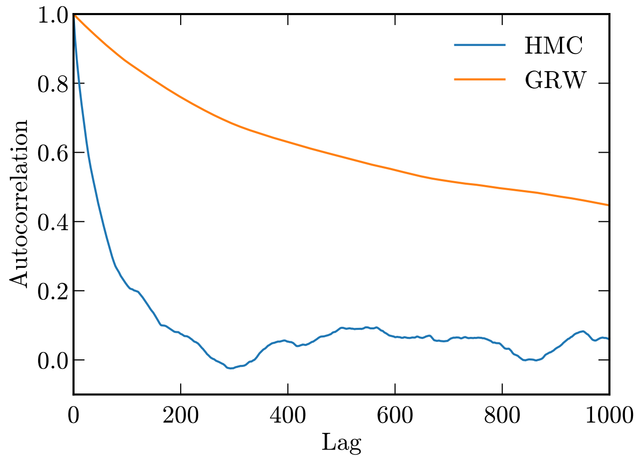

Figure 6. Autocorrelation as a function of lag for HMC and GRW on the same simulated data as discussed in Fig. 5. The smaller autocorrelation of the HMC leads to a larger number of independent samples and therefore a faster converging Monte Carlo.

In a follow up paper we will demonstrate that minute scale PE can be achieved by combining normalizing flows (Wong et al. 2022; Gabrié et al. 2022), GPU acceleration, and a derivative based sampler (Wong et al. in prep.). We therefore expect `JAX` waveforms to be ~~crucial~~ [AZ: highly beneficial] to future PE efforts in GW astronomy, especially for low-latency pipelines. [KW: Not just low latency, but also high dimensional analysis, such as testing GR and lensing.]

4. DISCUSSION AND CONCLUSION

In this paper we introduced and discussed the various benefits of differentiable waveforms in `JAX` for GW data 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 a NVIDIA Quadro 6000 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 utiliz-

[6] The remaining parameters are set to $\pi/3$. [AZ: Can we say what they are? ι, ψ, α, δ?]

ing derivative information of the waveform. 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 (Wong et al. in prep.). Differentiable waveforms therefore represent a crucial advancement to perform efficient GW science.

In this paper, we have primarily focussed on the IMRPhenom family of waveforms as their closed form expression is perfectly suited for implementation in JAX. Two other waveform families are commonly used in GW data analysis: the effective-one-body (EOB) and numerical relativity surrogate (NRSurrogate). Some progress has been made towards the construction of a differentiable NRsurrogate **(MI: cite our paper in prep?)**, but currently it seems difficult to implement EOB waveforms in JAX. In particular, the evolution of the Hamiltonian required to evaluate an EOB waveform is both inherently slow to differentiate and difficult to implement in JAX. 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 (Pratten et al. 2021). Ideally, future waveforms should be implemented either in JAX compatible python directly. **(MI: we could just say "directly implemented under an autodifferentiation framework" to be generic)** This would ensure that the community can easily utilize differentiability and hardware acceleration in the future.

5. ACKNOWLEDGMENTS

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.E. is supported by the Horizon Postdoctoral Fellowship. A.Z. is supported by NSF Grant PHY-2207594. A. C. acknowledges funding from the Schmidt Futures Foundation.

REFERENCES

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

Abadi, M., Agarwal, A., Barham, P., et al. 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/

Abbott, B. P., et al. 2016a, Phys. Rev. Lett., 116, 061102, doi: 10.1103/PhysRevLett.116.061102

—. 2016b, Phys. Rev. Lett., 116, 221101, doi: 10.1103/PhysRevLett.116.221101

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

Abbott, R., et al. 2021a. https://arxiv.org/abs/2111.03606

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

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

—. 2021d, Phys. Rev. D, 103, 122002, doi: 10.1103/PhysRevD.103.122002

—. 2021e, Astrophys. J. Lett., 913, L7, doi: 10.3847/2041-8213/abe949

—. 2021f, Phys. Rev. X, 11, 021053, doi: 10.1103/PhysRevX.11.021053

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

Agathos, M., Del Pozzo, W., Li, T. G. F., et al. 2014, Phys. Rev. D, 89, 082001, doi: 10.1103/PhysRevD.89.082001

Arun, K. G., Iyer, B. R., Qusailah, M. S. S., & Sathyaprakash, B. S. 2006, Class. Quant. Grav., 23, L37, doi: 10.1088/0264-9381/23/9/L01

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

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

Blackman, J., Field, S. E., Scheel, M. A., et al. 2017, Phys. Rev. D, 96, 024058, doi: 10.1103/PhysRevD.96.024058

Borhanian, S. 2021, Class. Quant. Grav., 38, 175014, doi: 10.1088/1361-6382/ac1618

Bradbury, J., Frostig, R., Hawkins, P., et al. 2018, JAX: composable transformations of Python+NumPy programs, 0.2.5. http://github.com/google/jax

Buonanno, A., Chen, Y., & Damour, T. 2006, Phys. Rev. D, 74, 104005, doi: 10.1103/PhysRevD.74.104005

Buonanno, A., & Damour, T. 1999, Phys. Rev. D, 59, 084006, doi: 10.1103/PhysRevD.59.084006

—. 2000, Phys. Rev. D, 62, 064015, doi: 10.1103/PhysRevD.62.064015

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

Coogan, A., Edwards, T., Foreman-Mackey, D., et al. 2022a, ripple, 0.0.1. https://github.com/tedwards2412/ripple

Coogan, A., Edwards, T. D. P., Chia, H. S., et al. 2022b. https://arxiv.org/abs/2202.09380

Cotesta, R., Marsat, S., & Pürrer, M. 2020, Phys. Rev. D, 101, 124040, doi: 10.1103/PhysRevD.101.124040

Damour, T. 2008, Int. J. Mod. Phys. A, 23, 1130, doi: 10.1142/S0217751X08039992

Damour, T., Jaranowski, P., & Schaefer, G. 2000, Phys. Rev. D, 62, 084011, doi: 10.1103/PhysRevD.62.084011

Evans, M., et al. 2021. https://arxiv.org/abs/2109.09882

Feroz, F., Hobson, M. P., & Bridges, M. 2009, Mon. Not. Roy. Astron. Soc., 398, 1601, doi: 10.1111/j.1365-2966.2009.14548.x

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

Hannam, M., Schmidt, P., Bohé, A., et al. 2014, Phys. Rev. Lett., 113, 151101, doi: 10.1103/PhysRevLett.113.151101

Husa, S., Khan, S., Hannam, M., et al. 2016, Phys. Rev. D, 93, 044006, doi: 10.1103/PhysRevD.93.044006

Iacovelli, F., Mancarella, M., Foffa, S., & Maggiore, M. 2022a. https://arxiv.org/abs/2207.02771

—. 2022b. https://arxiv.org/abs/2207.06910

Innes, M. 2018, CoRR, abs/1810.07951. https://arxiv.org/abs/1810.07951

Keppel, D., Lundgren, A. P., Owen, B. J., & Zhu, H. 2013, Phys. Rev. D, 88, 063002, doi: 10.1103/PhysRevD.88.063002

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

Krishnendu, N. V., & Ohme, F. 2021, Universe, 7, 497, doi: 10.3390/universe7120497

Maggiore, M., et al. 2020, JCAP, 03, 050, doi: 10.1088/1475-7516/2020/03/050

Neal, R. 2011, in Handbook of Markov Chain Monte Carlo, 113–162, doi: 10.1201/b10905

Owen, B. J. 1996, Phys. Rev. D, 53, 6749, doi: 10.1103/PhysRevD.53.6749

Owen, B. J., & Sathyaprakash, B. S. 1999, Phys. Rev. D, 60, 022002, doi: 10.1103/PhysRevD.60.022002

Paszke, A., Gross, S., Massa, F., et al. 2019, in Advances in Neural Information Processing Systems 32, ed. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Curran Associates, Inc.), 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pratten, G., Husa, S., Garcia-Quiros, C., et al. 2020, Phys. Rev. D, 102, 064001, doi: 10.1103/PhysRevD.102.064001

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

Reitze, D., et al. 2019, Bull. Am. Astron. Soc., 51, 035. https://arxiv.org/abs/1907.04833

Revels, J., Lubin, M., & Papamarkou, T. 2016, arXiv:1607.07892 [cs.MS]. https://arxiv.org/abs/1607.07892

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

Ruder, S. 2016, arXiv e-prints, arXiv:1609.04747. https://arxiv.org/abs/1609.04747

Schmidt, P. 2020, Frontiers in Astronomy and Space Sciences, 7, doi: 10.3389/fspas.2020.00028

Singer, L. P., & Price, L. R. 2016, Phys. Rev. D, 93, 024013, doi: 10.1103/PhysRevD.93.024013

Skilling, J. 2004, in AIP Conference Proceedings (AIP), doi: 10.1063/1.1835238

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

—. 2020b, MNRAS, 493, 3132, doi: 10.1093/mnras/staa278

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

Vitale, S., Haster, C.-J., Sun, L., et al. 2021, Phys. Rev. D, 103, 063016, doi: 10.1103/PhysRevD.103.063016

W. Farr, B. Farr, T. L. 2014, Modelling Calibration Errors In CBC Waveforms, Tech. rep., LIGO. https://dcc.ligo.org/LIGO-T1400682/public

Wong, K. W. K., Breivik, K., Farr, W. M., & Luger, R. 2022. https://arxiv.org/abs/2206.04062

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

Wong, K. W. K., Isi, M., & Edwards, T. D. P. in prep.

Yunes, N., Yagi, K., & Pretorius, F. 2016, Phys. Rev. D, 94, 084002, doi: 10.1103/PhysRevD.94.084002