Draft version January 18, 2023
Typeset using LaTeX **twocolumn** style in AASTeX631

`ripple`: Differentiable Waveforms for Gravitational Wave Data Analysis

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

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

ABSTRACT

Since their discovery in 2015, gravitational waves (GWs) have developed into a unique observational probe used throughout fundamental and astrophyics. As detections accumulate, data analysis task are becoming increasingly computationally demanding. Here, we propose the use of automatic differentiation through the programming framework `JAX` for accelerating a variety of these analsyis tasks. Firstly, we demonstrate that highly accurate waveforms (here, 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 five orders 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 stochastic gradient descent can be used to optimize the ~ 200 coefficients that make up the waveform. [AC: what does "make up the waveform" mean?] 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 CPU) with no loss in accuracy. This increased speed makes population forecasting substantially simpler, as discussed extensively in (Iacovelli et al. 2022a,b). [AC: I'd personally avoid refs in the abstract] Finally, we show that gradient based samplers like Hamiltonian Monte Carlo lead to significantly reduced autocorrelation values when compared to tradiaitional Monte Carlo methods. Since differentiable waveforms appear to have substantial advantages for a variety of tasks throughtout 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) from inspiraling and mergering compact objects (COs) has revolutionized our understanding of both fundamental physics and astronomy (Abbott et al. 2021). Although the data volumes from GW detectors are relatively small, analyzing the data is a computationally demanding task. In addition, this computational cost will substantially increase when next generation detectors [AC: should reference some here] come online.

The complexity begins before data gathering, where one is required to generate banks of template waveforms which will be used for a matched-filter search (Owen & Sathyaprakash 1999; Owen 1996). Once potential candidates are found, parameter estimation (PE) is performed to extract the detailed source properties of each event. In the most minimal example, this requires a Markov Chain Monte Carlo (MCMC) on an 11-dimensional parameter space for each event. Beyond this simple scenario, more complex waveform models with many additional parameters may be used to test for deviations from General Relativity. Finally, using the results of PE, population synthesis models constrain the progenitors systems from which the black holes we see merg-

ing today began their evolutionary journey (Wong et al. 2022). All of these tasks require significant computing resources. In this paper, we will argue that differentiable waveforms (and more generally differentiable pipelines) can play a significant role in alleaviating this computational demand.

Derivatives are ubiqitously useful in 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) or allow a sampler to rapidly explore parameter space (e.g. using Hamiltonian Monte Carlo (Neal 2011; Betancourt 2017) or gradient descent). [AC: previously said HMC was exploring high-likelihood regions, which isn't quite right.] Gradients are particularly helpful in high-dimensional spaces. Unfortunately, in the field of GW data analysis, analytic derivatives of the necessary quantities (such as the likelihood) have historically been difficult to obtain. 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 calcuations for gravitational waveforms (Coogan et al. 2022b).

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 support in GW physics 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. [AC: GPU support and JIT are separate features.]

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), the phenomenological inspiral-merger-ringerdown (IMRPhenom), and numerical relativity surrogate (NRsurrogate). Of these, the IMRPhenom family is most well suited for an AD implementation in `JAX`. In particular, their closed-form expressions and fixed-size frequency grids make a `JAX` implementation that complies with the constraints of JIT compilation simple to implement. EOB waveforms on the other hand require dynamic frequency grids and are therefore tricky to implement in `JAX`. In addition, EOB waveforms use hypergeometric functions which have limited support in `JAX`. NRsurrogate waveforms have no fundamental issue, but require the user to download relatively large data files, making their use cubersome in practice.**(TE: Max/Kaze, please add to/change this if you think its incorrect)**

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 can serve as a respository for differentiable implementations of some of the main waveforms currently used in LIGO and Virgo analyses. 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 accruacy. 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 can be better-optimized via the high-dimensional fitting methods enabled by differentiable waveforms. Secondly, we implement differentiable detector response functions and show that the speed of Fisher matrix calculations for approximate PE can be substantially improved [AC: 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. 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, numerical relatively (NR) simulations are required to accurately model the binary. Unfortunately, these numerical simulations

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

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 numerical relativity waveforms. 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 rewritten IMRPhenomD from scratch using a combination of pure `python` and `JAX` derivatives. In addition, we have restructed 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 implempation of IMRPhenomD is faithful to the `lalsuite` implementation, we start by defining the noise weighted inner product:

$$(h_{\theta_1}|h_{\theta_2}) \equiv 4\,\mathrm{Re}\int_0^\infty \mathrm{d}f\,\frac{h_{\theta_1}(f)h^*_{\theta_2}(f)}{S_n(f)}\,, \tag{1}$$

where S_n is the (one-sided) noise power spectral density (PSD) and h_{θ_1} is the frequency domain waveform evaluated at the instrinsic parameter point θ_1. We can then normalize the inner product through

$$[h_{\theta_1}|h_{\theta_2}] = \frac{(h_{\theta_1}|h_{\theta_2})}{\sqrt{(h_{\theta_1}|h_{\theta_1})\,(h_{\theta_2}|h_{\theta_2})}}\,. \tag{2}$$

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

$$\mathrm{m}(h_{\theta_1}|h_{\theta_2}) \equiv \max_{t_c,\,\phi_c}[h_{\theta_1}\,|\,h_{\theta_2}] \tag{3}$$

where t_c and ϕ_c are, respectively, the time and phase of coalesence.

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. This is shown in Fig. 1, where we have calculated the $\mathrm{m}(h_{\theta_1}|\bar{h}_{\theta_1})$ across the entire parameter space.[2] Here, h_{θ_1} corresponds to the `ripple` waveform implementation and $\bar{h}_{\theta_1}$ is the `lalsuite` implementation. 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 particular, the reduced density of points at low masses is where the $1 - \mathrm{m}(h_{\theta_1}|h^*_{\theta_1}) = 0$ causing points to not appear in the log scale colorbar.

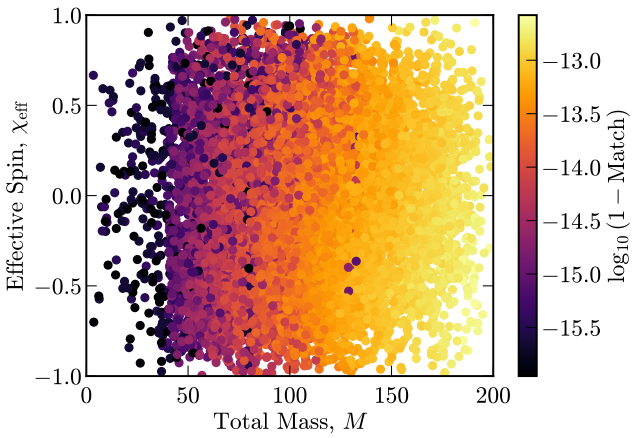

Figure 1. Match between the `ripple` and `lalsuite` implementations of the IMRPhenomD waveform as a function of total mass and effective spin. The lower density of points at low masses is caused by the match being exactly 1.0 and therefore not appearing in the log scale. It is clear that the `ripple` waveform matches the `lalsuite` implementation many of orders of magnitude more than necessary for all data analysis tasks across the entire parameter space.

Note that in General Relativity (GR) the total mass of the system is simply an overall scaling of the waveforms frequency evolution and is therefore not a truly independent parameter. If we instead use a flat PSD in Eq. 1 and scale the frequency grid correctly [AC: what do you mean by that?], 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 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

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

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 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. Since the `lalsuite` implementation of IMRPhenomD is written in `C`, the evaluation of the waveform itself is faster than `ripple`. Benchmarking on a MacBook Pro with an M1 Max Apple Silicon processor, we find that a single waveform evaluation takes $\sim 2-5\,\mathrm{ms}$ for `lalsuite` (interfaced with python) and $\sim 5-8\,\mathrm{ms}$ for `ripple`.[3] Although slower on a CPU benchmark, a key advantage of using `JAX` is its native ability to run on a GPU. Waveform calls can therefore be evaluated in parallel, massively improving performance. [AC: mention `vmap` and/or `pmap` – super trivial to parallelize! More work in C.] Performing the same benchmark as above on a Google Colab notebook with a freely-provided NVIDIA Tesla T4 GPU (far from the current state-of-the-art), we find that on average waveform evaluations take ~ 0.00005ms, nearly five orders of magnitude faster. Generalizing `lalsuite` waveforms to run on a GPU would be a significant undertaking.

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, 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 favourably with the dimensionality of the function. In particular, for every input dimension, D, added one would need to evaluate the function at least $2D$ times to calcuate finite difference derivatives for all input parameters. For reverse-mode AD, one only needs two function calls to evaluate the derivative of all input parameters, regardless of dimension.[4] Since the parameter space of GWs in GR has $\mathcal{O}(10)$ dimensons, the speed of derivative evaluation is less crucial than the stability. However, this might change for waveforms in beyond GR models where many more parameters are 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 will illustrate how three core tasks in GW science can be substantially improved through the use of differentiable waveforms. In the paper we will 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 forecasting and parameter estimation.

3.1. *Fine-Tuning Waveform Coefficients*

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 numerical relativity simulations are commonly regarded as the most accurate, 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 numerical relativity waveforms, which are essentially simpler ansatzes that are only approximations but can be evaluated much faster. [AC: redundant – remove or merge with earlier material.]

Every waveform approximant has some fitting coefficients that are tuned during the process of calibrating the ansatz to NR waveforms. 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 approximant reproduces the NR waveform. The 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, suf-

[3] For this benchmark we used 16 Hz and 1024 Hz for the lowest and highest frequencies respectively. In addition, we used a frequency spacing of 0.0125 Hz. We performed this benchmark by evaluating the waveform 5000 times and taking the average evaluation time.

[4] 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.

ficiently 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. Furthermore, IMRPhenomD is divided in three fitting segments, inpsiral, 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 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 during the optimization. Therefore, there is potential improvement of the accuracy by fitting all coefficients at once.

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 numerical relativity simulation. 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. [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 original and optimized waveform at different frequencies. In this optimization procedure, we took NR waveforms h_{NR} stated in (Khan et al. 2016) as training waveforms to fit waveform coefficients. [AC: how many are there?] We can see the error of 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?]

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 can be a trade-off 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. One can see the distribution of mismatch after optimization has smaller mismatch compared to before optimization, showing the optimization improves the accuracy of the waveform model. While many waveforms improved their accuracy, some remain to have similar mismatches. 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 moved to the left end of the distribution, indicating that the optimization procedure improved the model. Therefore, the new set of waveform coefficients can produce better waveforms for GW analyses.

3.2. *Fisher Forecasting*

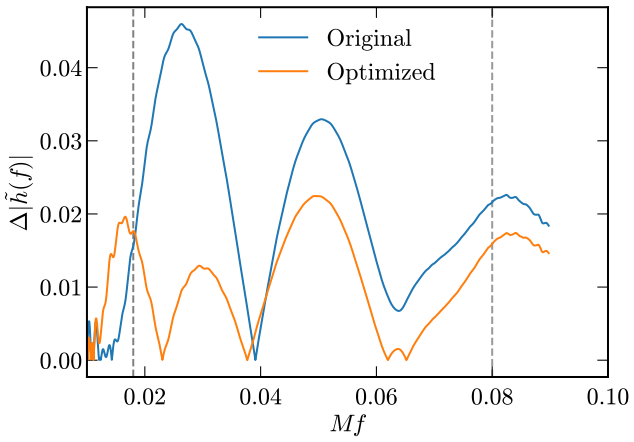

Figure 2. Relative error between the numerical relatively and IMRPhenomD waveform amplitudes. The blue line shows the relative error of the IMRPhenomD model with the original model coefficients where as the orange line uses the IMRPhenomD model after optimization using stochastic gradient descent. Interestingly,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

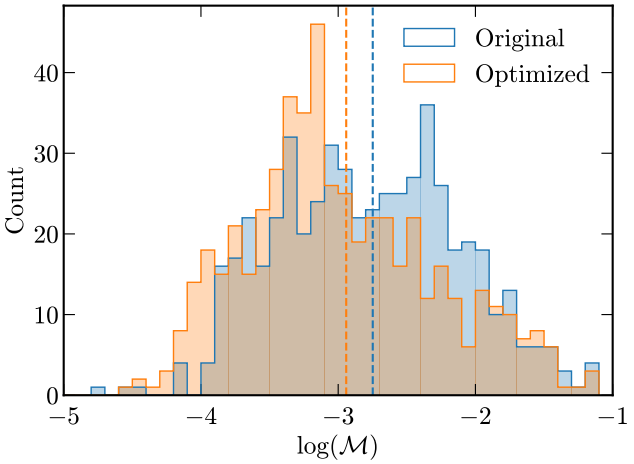

Figure 3. Distribution of 536 log mismatches (see Eq. 4) for the original (blue) and optimized (orange) IMRPhenomD. Overall, the optimized distribution shifts to lower mismatches and therefore a better waveform model. More quantitatively, the mean mismatch (shown as vertical dashed lines) is reduced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

Forecasting the sensitivity of future experiments is an essential task in GW science. Due to its theoretical simplicity and evaluation speed, the Fisher matrix formalism is commonly deployed to estimate how well a binary system's parameters can 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

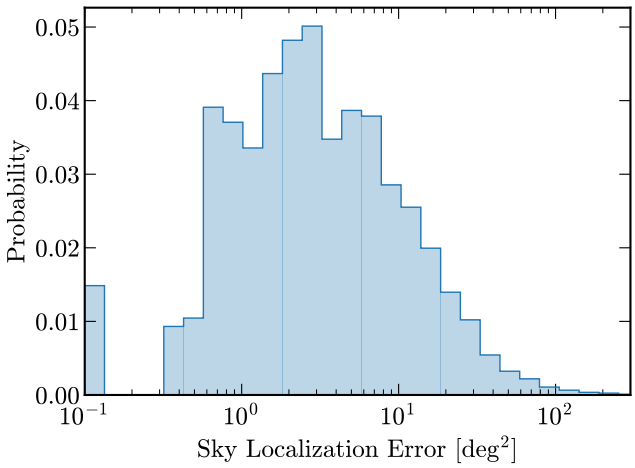

Figure 4. Distribution of Fisher information sky localization error for a population of nearby binaries. Automatic differentiation (AD) was used to compute the derivatives in Eq. 7. As emphasized in the text, after JIT compilation, each error calculation was around 100 times faster than a similar forecasting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

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 simplicitly 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?]

$$\mathcal{I}^k_{ij} = (\partial_i h_0 | \partial_j h_0)\,, \tag{7}$$

where k indicates the detector, $\partial_i = \partial/\partial\theta_i$, and h_0 is the strain measured by the detector which is given by,

$$h_0(\theta) = F_+(\lambda)h_+(\phi) + F_\times(\lambda)h_\times(\phi)\,. \tag{8}$$

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

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 for our measurement error estimate.

To illustrate the computational speed of computing Fisher matrices with AD, we consider a population of binaries and compute the sky localtization error following Eq. (28) in (Iacovelli et al. 2022a,b). 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 procress) 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. As discussed above, performance can be further improved by utilising 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 parameter estimation (PE). To do this, one typically uses a Markov Chain Monte Carlo (MCMC) or nested sampler (Skilling 2004; Feroz et al. 2009; Speagle 2020). 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 faciliate follow-up electromagnetic observations. Nevertheless, these do not present the whole picture; fast, general parameter estimation therefore remains a key aim of GW data analysis.

A primary issue with both MCMC and nested sampling is that neither utlizes 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 demonstrate the utility of a derivative based sampler and their efficiency on a small test problem. In particular, we will show that the autocorrelation time of a Hamiltonian Monte Carlo (HMC) sampler is significantly lower than a traditional MCMC algorithm.

For our basic example we perform an injection recovery test on a seven-dimensional parameter space with the same three detector setup considered in Sec. 3.2. More specifically, we generate Gaussian noise consistent with the measured PSDs for each detector and then inject a BBH signal with parameters: $M_c = 23.82\,\mathrm{M}_\odot$, $\eta = 0.248$, $\chi_1 = 0.3$, $\chi_1 = -0.4$, $D = 1.6\,\mathrm{Gpc}$, $t_c = 0.0$, and $\phi_c = 0.0$.[5] Using a standard Gaussian likelihood, we then run the HMC sampler implemented in `flowMC` for 2×10^5 steps and the gradient-free Gaussian random walk (GRW) sampler for 15×10^5 steps for comparison (each with four randomly-initialized independent chains). [AC: briefly summarize and give refs for flowMC and GRW. Is GRW Metropolis-Hastings with a

[5] The remaining parameters are set to $\pi/3$.

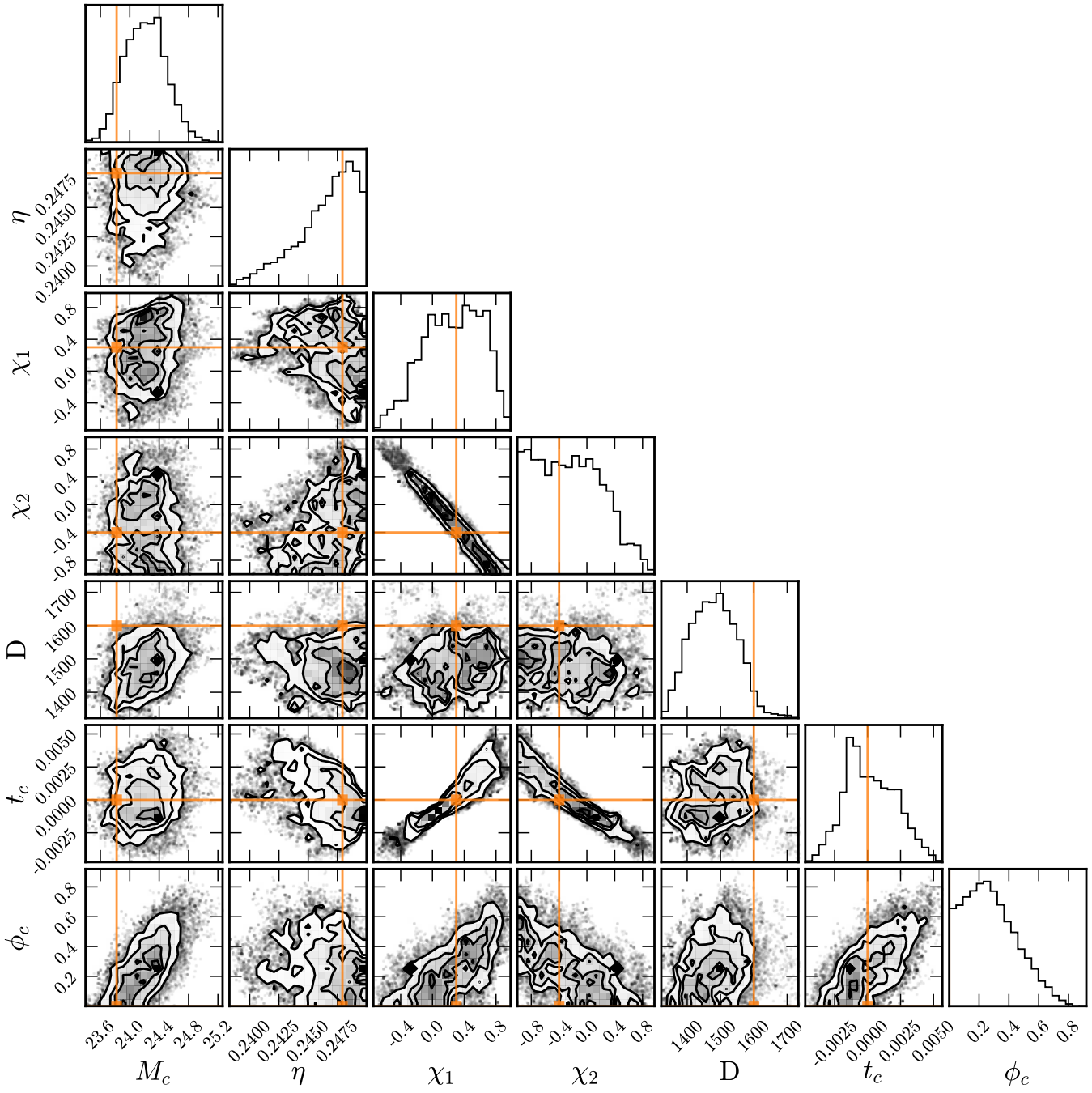

Figure 5. Corner plot for the posteriors from an HMC (see text for details) on simulated noise with injected signal. Orange lines indicate the true values of the injection. Although not fully converged, it is clear that we find posteriors consistent with the injected parameters. [AC: could apply a mild 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?]

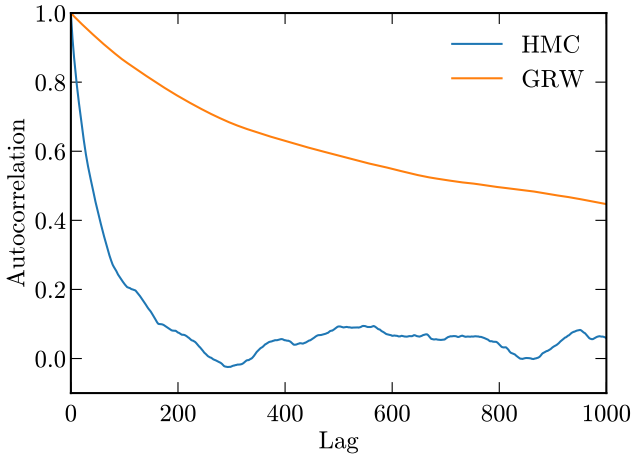

Figure 6. Autocorrelation as a function of lag for HMC and GRW on the same simulated data as discussed in Fig. 5. The smaller autocorrelation of the HMC leads to a larger number of independent samples and therefore a faster converging Monte Carlo.

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). 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.

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, indicating its superior performance.

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 to future PE efforts in GW astronomy, especially for low-latency pipelines.

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 free Google Colab GPU we found that waveform evaluations are nearly 5 orders 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. 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 tool 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, 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`, which requires fixed-length arrays for its JIT compilation. Since EOB waveforms are often regarded as the state-of-the-art, 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. Ideally, future waveforms should be implemented either in pure python or using `JAX` directly. [AC: if they're pure python, some translation will be required due to jax's restrictions.] This would ensure that the community can easily utilize differentiability and hardware acceleration in the future.

5. ACKNOWLEDGMENTS

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. C. acknowledges funding from the Schmidt Futures Foundation.

REFERENCES

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

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

Betancourt, M. 2017, arXiv e-prints, arXiv:1701.02434. https://arxiv.org/abs/1701.02434

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

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

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

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

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

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-libr pdf

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

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

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

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.