PAPER

Improving performance of SEOBNRv3 by ~300×

IOP Publishing Classical and Quantum Gravity

Class. Quantum Grav. **35** (2018) 155003 (14pp) https://doi.org/10.1088/1361-6382/aacb8c

Improving performance of SEOBNRv3 by ∼300×

Tyler D Knowles[1], Caleb Devine[1], David A Buch[1], Serdar A Bilgili[2], Thomas R Adams[1], Zachariah B Etienne[1,3] and Sean T McWilliams[2,3]

[1] Department of Mathematics, West Virginia University, Morgantown, WV 26506, United States of America
[2] Department of Physics and Astronomy, West Virginia University, Morgantown, WV 26506, United States of America
[3] Center for Gravitational Waves and Cosmology, West Virginia University, Chestnut Ridge Research Building, Morgantown, WV 26505, United States of America

E-mail: tk0014@mix.wvu.edu

Received 16 March 2018, revised 31 May 2018
Accepted for publication 11 June 2018
Published 28 June 2018

Abstract

When a gravitational wave is detected by Advanced LIGO/Virgo, sophisticated parameter estimation (PE) pipelines spring into action. These pipelines leverage approximants to generate large numbers of theoretical gravitational waveform predictions to characterize the detected signal. One of the most accurate and physically comprehensive classes of approximants in wide use is the 'spinning effective one body-numerical relativity' (SEOBNR) family. Waveform generation with these approximants can be computationally expensive, which has limited their usefulness in multiple data analysis contexts. In prior work we improved the performance of the aligned-spin approximant SEOBNR version 2 (v2) by nearly 300×. In this work we focus on optimizing the full eight-dimensional, precessing approximant SEOBNR version 3 (v3). While several v2 optimizations were implemented during its development, v3 is far too slow for use in state-of-the-art source characterization efforts for long-inspiral detections. Completion of a PE run after such a detection could take centuries to complete using v3. Here we develop and implement a host of optimizations for v3, calling the optimized approximant v3_Opt. Our optimized approximant is about 340× faster than v3, and generates waveforms that are numerically indistinguishable.

Keywords: SEOBNR, EOBNR, effective-one-body, parameter estimation, numerical relativity, optimization

(Some figures may appear in colour only in the online journal)

1. Introduction

With its first detections of gravitational waves [1–6], the advanced Laser Interferometer Gravitational-Wave Observatory (Advanced LIGO) has provided a fundamentally new means of observing the Universe. At the heart of each of these detections was a merger of compact binaries. In such binaries, each compact object possesses four intrinsic parameters: mass, and the three components of the spin vector. Inferring all eight intrinsic parameters[4] from a gravitational wave observation, which analysis is part of the more general *parameter estimation* (PE), remains a challenging and computationally expensive enterprise.

The LIGO/Virgo Scientific Collaboration (LVC) performs PE in a Bayesian framework, implemented within the LALINFERENCE software package that is part of the larger open-source software framework `LALSuite` [7]. In such a framework, we sample the posterior distribution by repeatedly calculating the likelihood that a particular waveform matches the data and applying Bayes' theorem. Evaluating the likelihood requires the rapid, sequential generation of as many as $\sim 10^8$ theoretical gravitational wave predictions [8]. Generating so many predictions via a full solution of the general relativistic field equations (using the tools of numerical relativity) would be far too computationally expensive. Thus theoretical models adopted for PE generally employ approximate solutions called *approximants*. State-of-the-art approximants adopt post-Newtonian techniques for evaluating the gravitational waveform throughout most of the inspiral and ringdown, and inject information from numerical relativity calculations for the late inspiral and merger.

One such gravitational wave approximant is the spinning effective one body-numerical relativity (SEOBNR) algorithm. This algorithm marries an effective-one body inspiral gravitational waveform approximation—with unknown higher-order terms fit to numerical relativity-generated gravitational wave predictions—to a black hole ringdown model [9]. In particular, SEOBNR starts with the effective one body (EOB) approach to non-spinning binary modeling [10] by mapping the dynamics of the two-body system to the dynamics of an effective particle moving in a deformed Schwarzschild metric. This work was then extended to include the effects of spinning, precessing binaries [11]. Implemented numerically, this spinning EOB procedure adopts a precessing source frame in which precession-induced variations in amplitude and phase are minimized during inspiral, and a source frame aligned with the spin of the final body for matching the inspiral to the merger-ringdown [9].

The other widely adopted approximant within the LVC for PE is the Phenom series of phenomenological waveform models. These waveform models are based on the combination of accurate post-Newtonian inspiral models with late-inspiral and merger phenomenological fits to suites of numerical relativity simulations [12]. More recently, Phenom models have been built to include the effects of precession [13]. In particular, precession effects are included by using post-Newtonian methods to compute precession angles and then 'twisting' the underlying non-precessing model [13–15]. Phenom models are simulated completely in the frequency domain, and therefore simplify some aspects of analysis. The only Phenom model designed to generate gravitational waveform predictions across all eight dimensions of parameter space is PhenomP [13], which was extensively used in the first six detection papers. We remark that Phenom is limited to a relatively small number of numerical relativity simulations against

[4] *Intrinsic* parameters are fundamental to the underlying physics of the system. In contrast, *extrinsic* parameters are related to the observer (e.g. polarization, sky location, and distance) and are not considered in this paper. Some authors refer to seven intrinsic parameters in the full-dimensional space, which include each spin component and the mass ratio of the system. This is because the total mass of the system is simply a scaling factor; we choose to refer to eight parameters since the total mass sets the time and frequency scales and therefore must be considered in PE.

which it has been calibrated, and it is difficult to determine the degree of systematic uncertainty in the model without appealing to another model for comparison.

Evaluating the systematic uncertainties of the Phenom model requires construction of an independent gravitational waveform model with independent systematics, and the SEOBNR family of models is a good candidate for this task. The only SEOBNR model capable of generating theoretical gravitational waveform predictions in all eight intrinsic dimensions of parameter space is the third version of the model, v3; the first and second versions were restricted to aligned-spin cases. In particular, v3 was built to accommodate arbitrary mass ratios, spin magnitudes, and spin orientations and has been calibrated and validated against a variety of numerical relativity simulations [16]. Thus v3 is vital for precessing compact binary merger PE.

Unfortunately, v3 is too currently too slow for PE. A single waveform generation across the LIGO band for, say, a black hole-neutron star system using v3 can take as long as an hour on a modern desktop computer. If LIGO observed a black hole-neutron star system merge, a sequential-gravitational-wave-generation PE would take thousands of years. Attempts to overcome the computational challenge of generating such time-consuming gravitational waveforms include the construction of reduced order model (ROM) approximants. ROMs make use of multidimensional interpolations between sampled points in another underlying approximant. For example, a ROM based on the aligned-spin SEOBNR version 2 (v2) approximant [17, 18] is constructed by first generating an extensive collection of waveform predictions using v2 that adequately samples the 4D parameter space reliably covered by v2. Then to obtain the gravitational waveform at any desired point in parameter space, the ROM simply interpolates within the four dimensions of sampled parameter space. A ROM version of v2 can generate waveforms up to $\sim 3000\times$ faster than v2 directly [17], which explains in part why ROMs enjoy such widespread use within the LVC for data analysis applications.

While ROMs have been constructed with favorable performance characteristics in aligned-spin situations, the cost of generating a ROM grows exponentially with the dimension of the ROM (though see [19] for ideas on combating this using a reduced basis approach). No strategy yet exists that can perform the 8-dimensional (8D) interpolations faster than the 8D approximant; until such a strategy is invented, the most promising way to improve the performance of theoretical waveform generation in the full 8D parameter space will be to optimize the approximant directly. As a proof-of-principle, we demonstrated that such an approach is capable of improving the performance of the aligned-spin v2 approximant by a typical factor of $\sim 280\times$ [20]. We call our optimized v2 approximant v2_opt. The precessing (8D) v3 approximant was in development as we independently prepared v2_opt, and thus originally contained all the same inefficiencies as v2. This suggests that if the full suite of optimizations we implemented in v2 were incorporated into v3, v3-based PE timescales might drop by two orders of magnitude at least.

This paper documents our incorporation of applicable v2 optimizations into v3, as well as our implementation of innovative new optimization ideas, which together act to speed up v3 by $\sim 340\times$. Optimization strategies are summarized in section 2. Section 3 presents code validation tests that demonstrate roundoff-level agreement between v3 and our latest optimized version of v3, designated v3_Opt, along with benchmarks providing an overview of performance gains across parameter space in v3_Opt. For convenience, table 1 defines all SEOBNR approximants referenced in this paper.

2. SEOBNRv3_opt: optimizations migrated from v2_opt

Optimizations to v3 were performed in two phases. In the first phase, described in section 2.1, we migrated to v3 all applicable optimizations developed during the preparation of v2_opt.

Table 1. Approximant naming conventions. These conventions apply throughout this paper.

Base approximant	Approx. name	Description
SEOBNRv2 (spin-aligned)	v2	Initial SEOBNRv2 implementation[a]; see [21]
	v2_opt	Optimized v2[a]; see [20]
SEOBNRv3 (precessing)	v3_preopt	Initial SEOBNRv3 implementation[b]; see [9]
	v3	Partially optimized v3_preopt with bug fixes[c]
	v3_pert	v3 with machine-ϵ mass perturbation[c]
	v3_opt	v3 optimized similarly to v2_opt[c]
	v3_Opt	v3_opt with new optimization strategies[d]
	v3_Opt_rk4	v3_Opt implementing RK4 rather than RK8[d]

[a] As of publication, the most recent updates to v2/v2_opt are found on commit ID `2cce415` in the `LALSuite master` branch.
[b] To generate a waveform with v3_preopt, download `LALSuite` from the archived repository page `https://git.ligo.org/lscsoft/lalsuite-archive/tree/14414694698a2f18c9135445003cade805ad2096` and use approximant tag SEOBNRv3.
[c] As of publication, the most recent updates to v3 and v3_opt are found on commit ID `19e95b4` in the `LALSuite master` branch.
[d] Approximants v3_opt and v3_opt_rk4 were updated to run v3_Opt and v3_Opt_rk4, respectively, on commit ID `1391f77` in the `LALSuite master` branch.

Sections 2.2 and 2.3 detail the second phase of optimization, outlining new strategies incorporated into v3_Opt.

2.1. Migrated optimizations

Here we summarize the optimizations to v2 which were migrated to v3 and thus implemented in v3_opt.

- *Switching compilers*. Switching from the `GNU Compiler Collection` (`gcc`) [22] `C` compiler to the `Intel Compiler Suite` (`icc`) [23] `C` compiler improves performance by roughly a factor of 2×. It is well-known that the `icc` compiler often produces more efficient executables than the `gcc` compiler[5].
- *Minimize transcendental function evaluations*. The EOB Hamiltonian equations of motion were hand-optimized by minimizing calls to some expensive transcendental functions such as `exp()`, `log()`, and `pow()`.
- *Replacing finite difference with exact derivatives*. When solving the EOB Hamiltonian equations of motion, v3 computes partial derivatives of the Hamiltonian using finite difference approximations. We replaced these with exact, Mathematica-generated expressions for the derivatives, using Mathematica's code generation facilities—which includes common subexpression elimination (CSE)—to generate the `C` code [24]. Although this alone acts to significantly speed up v3, in this work we further optimize these Mathematica-generated derivatives.
- *Increasing the order of the ODE solver*. v3 solves the EOB Hamiltonian equations of motion via a Runge–Kutta fourth order (RK4) ODE solver. After implementing exact derivatives, we noticed that the number of RK4 steps needed dropped significantly—

[5] We used the following compiler flags when compiling with `icc`: `-xHost`, `-O2`, and `-fno-strict-aliasing`.

presumably due to the effective removal of high numerical noise intrinsic to finite-difference derivatives. We then found that adopting a Runge–Kutta eighth order (RK8) ODE solver resulted in 2× larger timesteps, so an even larger speed-up was observed.

- *Reducing orbital angular velocity calculations*. The orbital angular velocity ω was calculated for each (ℓ, m) mode (as defined in [9]) inside the ODE solver. As ω exhibits no dependence on ℓ or m, this expensive recalculation was unnecessary and needs only be performed once.

For more details about these optimizations see our v2 optimization paper [20].

2.2. Guided automatic differentiation: a more efficient way of generating symbolic derivatives of the Hamiltonian

After migrating the v2 optimizations described in section 2.1 to v3, profiling analyses indicated that approximately 75% of v3_opt's total runtime was spent computing the v3 Hamiltonian [9] and its partial derivatives with respect to the twelve degrees of freedom (consisting of three spatial degrees $\{x, y, z\}$, three momentum degrees $\{p_x, p_y, p_z\}$, and three spin degrees for each of the two binary components $i \in \{1, 2\}$: $\{s_i^x, s_i^y, s_i^z\}$).

In v3, the ODE solver computes these partial derivatives by direct evaluations of the Hamiltonian itself via finite difference techniques [21]. In v3_opt, these numerical derivatives were replaced with Mathematica-generated exact derivatives. Although these exact derivatives unlock significant performance gains, the Mathematica-generated `C` code was neither particularly human-readable (comprising thousands of lines of code output by Mathematica's CSE routines) nor particularly well-optimized (common patterns were still visible and recomputed in the `C` code). Attempts to gain performance through consolidation of all derivatives—as was possible in our optimizations of v2—proved beyond Mathematica's capabilities when differentiating the v3 Hamiltonian on our high-performance workstations. Therefore, `C` codes for all twelve exact derivatives needed to be output separately, resulting in a significant number of unnecessary re-computations.

We present here our new strategy for computing partial derivatives of the Hamiltonian, called *guided automatic differentiation* (GAD), which results in a significant reduction in computational cost while ensuring the resulting code is highly human-readable. GAD is based on forward accumulation automatic differentiation, with the advantage of the subexpressions being chosen by hand to minimize the overall number of floating point operations.

The following describes the process of computing a partial derivative of the v3 Hamiltonian H with respect to an *arbitrary* independent variable x_1 using GAD. We may write H in the following form, where I is a set of input quantities:

$$\begin{aligned} v_1 &= f_1(I) \\ v_2 &= f_2(v_1, I) \\ v_3 &= f_3(v_1, v_2, I) \\ &\vdots \\ H &= f_N(v_1, v_2, v_3, ..., I). \end{aligned}$$

Here f_ℓ is the ℓth function of the set of input quantities I and previously computed subexpressions $\{v_0, v_1, \ldots, v_{\ell-1}\}$. Although $N \approx 200$ for v3, *for the sake of example* we suppose $N = 3$, $I = \{x_1, x_2\}$, and

Table 2. Step-by-step GAD code evolution.

Step 1: Convert C to Mathematica	Step 2: Parameterize subexpressions
`v1 = Sqrt[x1] + a*x1`	`v1 = Sqrt[x1[x]] + a*x1[x]`
`v2 = Sqrt[x2] + a*x2`	`v2 = Sqrt[x2] + a*x2`
`v3 = (v1 + v2)/(v1*v2)`	`v3 = (v1[x] + v2[x])/(v1[x]*v2[x])`
`H = v3*v3`	`H = v3[x]*v3[x]`

Step 3: Utilize Mathematica to compute derivatives

```
v1' = x1'[x]/(2*Sqrt[x1[x]]) + a*x1'[x]
v2' = 0
v3' = (v1[x]*v2[x]*(v1'[x] + v2'[x]) - ((v1[x] + v2[x])*(v1'[x]*v2[x]
      + v1[x]*v2'[x]))/(v1[x]*v1[x]*v2[x]*v2[x])
H' = 2*v3'[x]*v3[x]
```

Step 4: Convert Mathematica to C; prime notation becomes a protected `prm` suffix

```
v1prm = x1prm/(2*sqrt(xi)) + a*x1prm
v2prm = 0
v3prm = (v1*v2*(v1prm + v2prm) - ((v1 + v2)*(v1prm*v2 + v1*v2prm))/(v1*v1*v2*v2)
Hprm = 2*v3prm*v3
```

Step 5: Replace `x1prm` with 1 and remove terms equaling 0

```
v1prm = 1./(2*sqrt(x1)) + a
v3prm = (v1*v2*v1prm - (v1 + v2)*v1prm*v2)/(v1*v1*v2*v2)
Hprm = 2*v3prm*v3
```

$$
\begin{aligned}
v_1 &= \sqrt{x_1} + a x_1 \\
v_2 &= \sqrt{x_2} + a x_2 \\
v_3 &= (v_1 + v_2)/(v_1 v_2) \\
H &= v_3^2.
\end{aligned}
$$

We demonstrate GAD by taking a partial derivative of H with respect to the independent input variable x_1. Table 2 displays the evolution of this example code under the GAD scheme, which proceeds as follows:

1. We begin with a list of variables and subexpression computations for the Hamiltonian, and translate this C code into the Mathematica language.
2. We parameterize the terms of each subexpression according to their dependence on x_1.
3. Mathematica computes derivatives of each subexpression.
4. We convert the Mathematica output into C code.
5. We replace each occurrence of x_1' with 1 and remove terms equal to 0.

The resulting C code is short, optimized, and human readable. Furthermore, any terms that are common to all derivative expressions are computed and saved before computing the partial derivatives, further reducing the computational cost.

Since each v_ℓ is merely an intermediate of H, there is significant freedom in our choice of the set of subexpressions $\mathcal{V} \equiv \{v_1, v_2, \ldots, v_{N-1}\}$. Our choices do, however, have a direct effect on the number of calculations necessary to compute $\partial_{x_1} H$, which we measure in floating

Table 3. Relative FLOPs count of the mathematical operations.

$a+b$	$a-b$	$a*b$	$a=b$	a/b	$\texttt{sqrt}(a)$	$\texttt{log}(a)$	$\texttt{pow}(a,b)$
1	1	1	1	3	3	24	24

Table 4. Number of FLOPs using ED versus GAD methods.

Derivative scheme	Space derivative (FLOPs)	Momentum derivative (FLOPs)	Spin derivative (FLOPs)	Total (FLOPs)
ED	3×5073	3×2319	6×4333	48174
GAD	3×1418	3×527	6×1264	13419

Table 5. Benchmark comparison of ED to GAD strategies. In each scenario, we adopt a 10 Hz start frequency.

	v3_opt (s)	v3_opt (s)
Parameters	ED	GAD
Neutron star binary	36.75	20.49
$1.4M_\odot + 1.4M_\odot$, $s_1^y = 0.05$		$\times$(**1.79**)
Black Hole + Neutron Star	8.07	4.69
$10M_\odot + 1.4M_\odot$, $s_1^y = 0.4$		$\times$(**1.72**)
Black hole binary (GW150914-like)	0.64	0.38
$36M_\odot + 29M_\odot$		$\times$(**1.68**)
$s_1^y = 0.05,\ s_1^z = 0.5,\ s_2^y = -0.01,\ s_2^z = -0.2$		

point operations (FLOPs[6].) Our goal in GAD, therefore, is to choose $\mathcal{V}$ to minimize the number of FLOPs needed to compute $\partial_{x_1} H$.

In general, the largest contributor to FLOPs is the product rule. If there are M different subexpressions multiplied together in a given expression, computing the derivative will require $\mathcal{O}(M^2)$ FLOPs. If we therefore choose $\mathcal{V}$ such that each v_ℓ contains no more than two previous subexpressions multiplied together, we should minimize the overall cost. We expect a significant reduction in FLOPs to correspond to a significant reduction in the time to generate a waveform.

We estimated the number of FLOPs based on benchmarks provided in [25] for CPUs corresponding to the CPU family in our workstations (Intel Core i7-6700) and generated table 3. We emphasize that the values listed in table 3 are truly rough estimates, used only to provide us general direction as we seek an optimal $\mathcal{V}$.

Table 4 compares the number of Hamiltonian derivative FLOPs under GAD to the number in the exact derivatives (EDs) generated by Mathematica's CSE code generation algorithm. In principle, the difference in FLOPs between ED and GAD schemes may be used to predict the waveform generation speedup factor. A direct comparison from table 4 indicates a 3.6$\times$ reduction in FLOPs when using GAD. For a double neutron star coalescence, Hamiltonian derivative computations constitute about 80% of waveform generation time. This suggests a speedup factor of 2.3$\times$. Waveform generation times for three scenarios comparing ED and GAD implemented in v3_opt are shown in table 5, and demonstrates a speedup factor of about 1.7$\times$. We emphasize again that counting FLOPs using the relative values of table 3

[6] Not to be confused with 'FLOPs per second' (FLOPS).

only provides a rough estimate of the reduction in FLOPs, and the compiler itself rearranges arithmetic expressions to minimize FLOPs as well so the gap between our estimated and observed speed-ups is not surprising.

2.3. Dense output: a more efficient way of interpolating sparsely-sampled data

An RK4 ODE solver with adaptive timestep control solves the EOB Hamiltonian equations of motion in v3; thus solutions are unevenly sampled in time. Subsequent analyses require mapping these data into the frequency domain via the fast Fourier transform (FFT), which expects evenly-sampled data. Rather than restricting the integration timestep, v3 uses cubic splines to interpolate the Hamiltonian solutions after RK4 runs to completion. During optimization of v2, the GSL cubic spline interpolation routine was optimized and gave significant performance gains. During optimization of v3, it was discovered that third-order Hermite interpolation made v3_Opt more faithful to v3 (see section 3). Hermite interpolation requires only two function values and the derivatives at those values, which are available at each step of RK8. Thus we may interpolate the sparsely-sampled data to the desired evenly-sampled data 'on the fly' during integration. Such an integration routine is called a *dense output* method [26]. In particular, suppose the RK8 integrator computes the solution $y(t_0)$ and $y(t_1)$ at times t_0 and t_1 with timestep h and derivative values y'_0 y'_1. Then for any $0 \leqslant \theta \leqslant 1$, we have

$$y(t_0+\theta h) = (1-\theta)y(t_0) + \theta y(t_1) + \theta(\theta-1)\left[(1-2\theta)y(t_1) - y(t_0) + (\theta-1)hy'(t_0) + \theta h y'(t_1)\right].$$

As this cubic Hermite interpolation routine uses both the solution data and derivative values at each point, it therefore requires only the output of the RK8 integration and no further data storage or function evaluations.

3. Results

In section 3.1 we establish that v3_Opt produces waveforms which agree with v3 at the level of roundoff error. Section 3.2 then describes the process of measuring speedup and demonstrates the speedup factor achieved.

3.1. Determining faithfulness

Given two waveforms $h_1(t)$ and $h_2(t)$ (in the time domain), we determine if $h_1(t)$ is *faithful* to $h_2(t)$ using the LVC's open-source PyCBC software [27–29]. This computation depends on the following definitions, which we write in the same form as [30]. The *noise-weighted overlap* between h_1 and h_2 is defined as

$$(h_1|h_2) \equiv 4\mathrm{Re}\int_{f_l}^{f_h} \frac{\tilde{h}_1(f)\tilde{h}_2^*(f)}{S_n(f)}\mathrm{d}f$$

with $\tilde{h}_i(f)$ denoting the Fourier transform of the waveform $h_i(t)$, h_i^* denoting the complex conjugate of h_i, f_l and f_h denoting the endpoints of the range of frequencies of interest, and $S_n(f)$ denoting the one-sided power spectral density (PSD) of the LIGO detector noise. We chose $f_l = 20$ Hz and $S_n(f)$ to be Advanced LIGO's design zero-detuned high-power noise PSD [31]. For each waveform, f_h is the Nyquist critical frequency [26]. We then define the *faithfulness* between h_1 and h_2 to be the overlap between the normalized waveforms maximized over relative time and phase shifts:

Table 6. Ranges of values for random input parameters in our faithfulness tests.

Mass of object 1 (solar masses)	$m_1 \in [1, 100]$
Mass of object 2 (solar masses)	$m_2 \in [1, 100]$
Spin magnitude of object 1 (dimensionless)	$\lvert a_1\rvert \in [0, 0.99]$
Spin magnitude of object 2 (dimensionless)	$\lvert a_2\rvert \in [0, 0.99]$
Binary total mass (solar masses)	$m_{\text{total}} \in [4, 100]$
Starting orbital frequency (Hz)	$f = 19$

$$\langle h_1|h_2\rangle \equiv \max_{\phi_c, t_c} \frac{(h_1(\phi_c, t_c)|h_2)}{\sqrt{(h_1|h_1)(h_2|h_2)}}.$$

Here t_c and ϕ_c denote the coalescence time and phase, respectively. Note that normalization forces $\langle h_1|h_2\rangle \in [0, 1]$, with $\langle h_1|h_2\rangle = 1$ indicating complete overlap (and therefore a perfectly faithful waveform) while $\langle h_1|h_2\rangle = 0$ indicates no overlap (an *unfaithful* waveform[7]). For each faithfulness test conducted, we generate a waveform with two different approximants and the same set of input parameters.

We ran 100 000 faithfulness tests for each set of waveform approximants we wished to compare. The input parameters for each test are randomly chosen by PyCBC with bounds as outlined in table 6; these bounds are chosen to capture the relevant parameter space for v3. Note that each of the spin parameters s_i^x, s_i^y, s_i^z are chosen randomly in $(-1, 1)$ with the constraint

$$\sqrt{(s_i^x)^2 + (s_i^y)^2 + (s_i^z)^2} \leqslant 0.99, \quad i \in \{1, 2\}.$$

The specific faithfulness runs we conducted were organized as follows. The approximant v3_pert is identical to v3 except m_1 is replaced with $m_1\left(1 + 10^{-16}\right)$; such a perturbation should result in waveforms that are nearly identical and provides a measure of how sensitive v3 is to roundoff error. Thus faithfulness tests comparing v3 and v3_pert provide a 'control' against which we compare the faithfulness of v3_Opt to v3. As another point of comparison, we also test v3 (which is RK4-based) against the RK4-based v3_Opt_rk4. For each approximant comparison we compare the effect of increasingly stricter ODE solver tolerance. By default, v3 sets the ODE solver's absolute and relative error tolerances to $\varepsilon \equiv 1 \times 10^{-8}$; we compare faithfulness at tolerances of ε, $\varepsilon \times 10^{-1}$, $\varepsilon \times 10^{-2}$, $\varepsilon \times 10^{-3}$, and $2\varepsilon \times 10^{-4}$. Finally, we also consider the effect of compiler choice on faithfulness and so conduct faithfulness runs using both `gcc` and `icc`. Table 7 summarizes the faithfulness tests conducted and their results; the rightmost column displays the counting error $\sqrt{n}$ for the number of waveforms n with $\langle\cdot|\cdot\rangle < 0.999$.

We comment on the values in table 7. For a couple of parameters for which $\langle\cdot|\cdot\rangle < 0.8$ when comparing v3 to v3_Opt compiled with `gcc`, one author back-traced a significant difference between v3 and v3_Opt to the ODE stopping condition or the time of maximum amplitude being clearly wrong in v3 but not v3_Opt. In particular, there are some algorithms within v3 that are fundamentally non-robust, and v3_Opt inherits most of these functions. The RK8 integration of v3_Opt should be just as accurate as the RK4 integration of v3_Opt_rk4 when the tolerances are equal, but the output from RK8 should be much sparser (by more than a factor of 2) than RK4. Since we observe worse faithfulness with v3_Opt than v3_Opt_rk4, we conclude that most of the truncation error stems from the interpolation of the sparsely-sampled ODE solution to a uniform timestep.

[7] Another common measure in faithfulness tests is *mismatch*, defined as $1 - \langle h_1|h_2\rangle$.

Table 7. Summary of `PyCBC` faithfulness results. Here $\varepsilon = 1 \times 10^{-8}$ and each row reports the results of a run of 100 000 faithfulness tests. `icc` refers to Intel compiler version 15.5.223, while `gcc` refers to GNU compiler version 4.9.

Comparison	Compiler	ODE tolerance	Number of waveforms with faithfulness					Counting
			<0.8	<0.9	<0.95	<0.99	<0.999	Error
v3 versus v3_pert	`gcc`	ε	1	5	13	104	399	± 20
(per 10^5 for 10^6 tests)	`icc`	ε	1.0	4.2	11.5	109.0	398.2	± 6.3
v3 versus v3_Opt	`gcc`	ε	5	28	136	1184	5466	± 74
	`icc`	ε	5	28	135	1174	5509	± 74
	`icc`	$\varepsilon \times 10^{-1}$	2	16	44	327	1510	± 39
	`icc`	$\varepsilon \times 10^{-2}$	0	2	12	143	727	± 27
	`icc`	$\varepsilon \times 10^{-3}$	1	3	8	80	511	± 23
	`icc`	$2\varepsilon \times 10^{-4}$	1	1	2	60	457	± 21
v3 versus v3_Opt_rk4	`gcc`	ε	1	9	35	427	1529	± 39
	`icc`	ε	0	9	35	420	1510	± 39
	`icc`	$\varepsilon \times 10^{-1}$	1	7	24	223	926	± 30
	`icc`	$\varepsilon \times 10^{-2}$	0	0	8	114	585	± 24
	`icc`	$\varepsilon \times 10^{-3}$	1	3	8	77	483	± 22
	`icc`	$2\varepsilon \times 10^{-4}$	1	2	3	52	423	± 21

Most importantly, notice that as we make the ODE solver's tolerance ε stricter (resulting in smaller errors and more finely sampled output data from the ODE solver), the faithfulness between v3 and v3_Opt improves to the level of agreement between v3 and v3_pert. Thus we conclude that v3_opt generates roundoff-level agreement in the limit of $\varepsilon \to 0$ with errors dominated by interpolation otherwise.

3.2. Performance benchmarks

In order to capture the full effect of our optimizations to v3, we compared waveform generation times of v3_Opt with waveform generation times of v3_preopt. In particular, v3_preopt lacks by-hand optimizations of the EOB Hamiltonian implemented in the development of v2_opt; thus unnecessary computations of transcendental functions `pow()`, `log()`, and `exp()` remain therein. All reported benchmarks were completed on a single core of a modern desktop computer with an Intel Core i7-7700 CPU and 64 GB RAM.

To highlight cases of interest, table 8 summarizes benchmarks of v3_Opt and v3_Opt_rk4 in comparison to v3_preopt for a handful of scenarios of interest to LIGO. The speedup factors are also included, with speedup simply defined to be the ratio of time to generate a waveform with v3_preopt to the time to generate the same waveform with v3_Opt or v3_Opt_rk4.

To demonstrate that the advertised speedup factors of table 8 apply across the parameter space of binaries of interest to the LVC, we completed four benchmark surveys. The first two concern binary black hole systems, one with varying masses and the other with varying spins. The third survey considers mixed binaries (one black hole and one neutron star), and the fourth binary neutron stars. The parameters tested in each run are included in table 9. The results of these surveys are plotted in figure 1 and summarized in table 8.

Table 8. Benchmarks and speedups of v3_Opt and v3_Opt_rk4 compared to v3.

	v3_preopt	v3_Opt_rk4	v3_Opt	v3_Opt
Physical scenario	`gcc`, (s)	`gcc`, (s)	`gcc`, (s)	`icc`, (s)
DNS, $s_2^y = 0.05$	8618.60	98.51	42.85	21.22
$1.3M_\odot + 1.3M_\odot$		×(**87.49**)	×(**201.1**)	×(**406.2**)
BHNS, $s_{\rm NS}^y = 0.05$	2760.77	20.75	8.84	4.37
$10M_\odot + 1.3M_\odot$		×(**133.0**)	×(**312**)	×(**632**)
BHB, $s_2^y = 0.05$	127.71	1.70	0.90	0.46
$16M_\odot + 16M_\odot$		×(**75.1**)	×(**140**)	×(**280**)
BHB, $s_1^y = s_2^y = 0.9$	168.13	1.75	0.91	0.46
$16M_\odot + 16M_\odot$		×(**96.1**)	×(**180**)	×(**370**)
BHB, $s_1^y = s_2^z = 0.9$	235.53	3.48	1.55	0.76
$10M_\odot + 10M_\odot$		×(**67.7**)	×(**152**)	×(**310**)
BHB, GW150914-like	31.48	0.75	0.51	0.27
$36M_\odot + 29M_\odot$		×(**42**)	×(**60**)	×(**120**)
$s_1^y = 0.05$, $s_1^z = 0.5$				
$s_2^y = -0.01$, $s_2^z = -0.2$				

Table 9. Surveyed parameters: each survey tested 400 parameter combinations, with 20 evenly-spaced values taken in each range indicated. Here $\mathrm{BHB_M}$ indicates the black hole binary mass survey, $\mathrm{BHB_S}$ the black hole binary spin survey, BHNS the black hole neutron star survey, and DNS the double neutron star survey. We define $q \equiv \frac{m_1}{m_2}$, the ratio of the mass of object 1 to the mass of object 2. The dimensionless Kerr spins of each object are denoted a_1 and a_2, respectively. Each waveform generation started with a frequency of 10 Hz used a sample rate of 16 384 Hz.

Ranges	m_1 ($M_\odot$)	q (dimensionless)	a_1 (dimensionless)	a_2 (dimensionless)
$\mathrm{BHB_M}$	$[16.7, 100.3]$	$[1, 10]$	0.0500001	0
$\mathrm{BHB_S}$	10	1	$[-0.95, 0.95]$	$[-0.95, 0.95]$
BHNS	$[7, 100]$	$\frac{M}{1.4}$	$[-0.95, 0.95]$	0
DNS	$[1.2, 2.3]$	$\frac{M}{m\in[1.2,2.3]}$	0.0500001	0

We would like to measure an average speedup based on the four surveys. As in [20], we define an overall speedup factor as a waveform cycle-weighted average

$$\mathcal{S} = \frac{\sum_i \mathcal{S}_i N_i}{\sum_i N_i}$$

where $\mathcal{S}_i$ is the speedup factor for generating the ith waveform and N_i is the number of wave-cycles in the ith waveform. We found $\mathcal{S} \sim 340$. This reduces the time necessary for a black hole binary PE run from ∼100 years (with v3_preopt) to ∼8 months (with v3_Opt). We expect lower mass PE runs will be possible on similar timescales with additional optimizations.

4. Conclusions and future work

Anticipating the potential detection by Advanced LIGO of significantly precessing compact binaries, we have optimized v3 to make costly precessing-waveform-approximant-based data

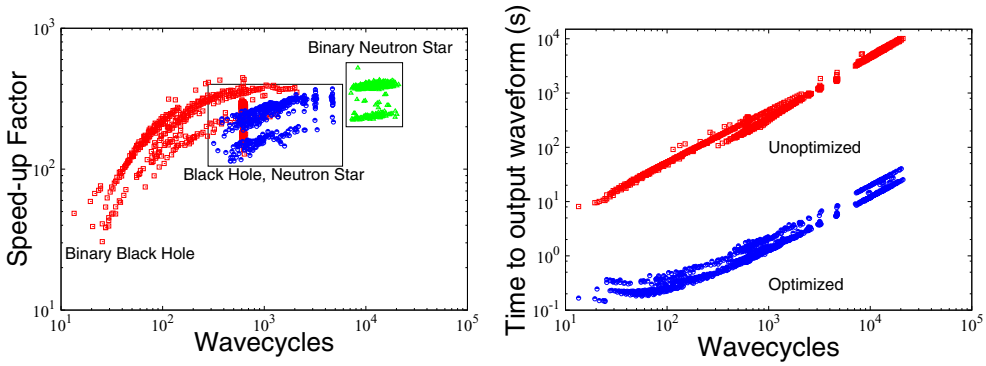

Figure 1. Performance benchmarks: Left panel: plots speedup factor versus number of wavecycles in the binary inspiral. Measuring the number of wavecycles allows us to compactly display the results of the benchmark tests without explicit reference to mass or spin. Right panel: plots the number of wavecycles versus the time taken to output the waveform. Note that the speedup factor in the left panel is simply the ratio of the curves in the right panel.

analysis applications like PE possible in a reasonable amount of time. If an efficient 8D ROM is found, such optimizations will make the construction of this ROM faster. After migrating v2/v4 optimizations to v3, we further optimized partial derivatives of the Hamiltonian using a GAD scheme. This resulted in waveforms that are faithful to v3, as evidenced by faithfulness increasing to 1 as ODE tolerance decreases. We achieved an average overall speedup of $\sim$340$\times$, ranging from $\sim$120$\times$ for GW150914-like black hole binaries to $\sim$630$\times$ for black hole-neutron star binaries. We expect that further optimizations are possible, achieving an additional speedup factor of at least $\sim$3$\times$. Future work will focus on transforming Cartesian coordinates to spherical coordinates to lower sampling rates even more during ODE solving and integration.

Acknowledgments

We thank O Birnholtz, N Johnson-McDaniel, R Sturani, A Taracchini, and C Haster for helpful comments and discussion during a review of the v3_opt code. A Taracchini is especially thanked for introducing us to faithfulness testing via `PyCBC`, as is S Teukolsky for suggesting dense output methods. We also thank R Haas for his work optimizing v3 during its development, and I Ruchlin for numerous helpful discussions. This work was supported primarily by NSF LIGO Research Support Grant PHY-1607405. Early work on this project was supported by NSF EPSCoR Grant 1458952 and NASA Grant 13-ATP13-0077. We are grateful for the computational resources provided by the Leonard E Parker Center for Gravitation, Cosmology and Astrophysics at University of Wisconsin-Milwaukee (NSF Grant 0923409), LIGO Livingston Observatory, LIGO Hanford Observatory, and the LIGO-Caltech Computing Cluster.

ORCID iDs

Tyler D Knowles https://orcid.org/0000-0002-3997-115X

References

[1] Abbott B P *et al* 2016 Observation of gravitational waves from a binary black hole merger *Phys. Rev. Lett.* **116** 061102
[2] Abbott B P *et al* 2016 GW151226: observation of gravitational waves from a 22-solar-mass binary black hole coalescence *Phys. Rev. Lett.* **116** 241103
[3] Abbott B P *et al* 2017 GW170104: observation of a 50-solar-mass binary black hole coalescence at redshift 0.2 *Phys. Rev. Lett.* **118** 221101
[4] Abbott B P *et al* (The LIGO Scientific Collaboration, The Virgo Collaboration) 2017 GW170608: observation of a 19-solar-mass binary black hole coalescence
[5] Abbott B P *et al* 2017 GW170814: a three-detector observation of gravitational waves from a binary black hole coalescence *Phys. Rev. Lett.* **119** 141101
[6] Abbott B P *et al* 2017 GW170817: observation of gravitational waves from a binary neutron star inspiral *Phys. Rev. Lett.* **119** 161101
[7] The LIGO Scientific Collaboration 2016 LALSuite: LSC algorithm library suite (https://lsc-group.phys.uwm.edu/daswg/projects/lalsuite.html)
[8] Veitch J *et al* 2015 Parameter estimation for compact binaries with ground-based gravitational-wave observations using the LALInference software library *Phys. Rev.* D **91** 042003
[9] Pan Y, Buonanno A, Taracchini A, Kidder L E, Mroué A, Pfeiffer H P, Scheel M A and Szilágyi B 2014 Inspiral-merger-ringdown waveforms of spinning, precessing black-hole binaries in the effective-one-body formalism *Phys. Rev.* D **89** 084006
[10] Buonanno A and Damour T 1999 Effective one-body approach to general relativistic two-body dynamics *Phys. Rev.* D **59** 084006
[11] Buonanno A, Chen Y and Damour T 2006 Transition from inspiral to plunge in precessing binaries of spinning black holes *Phys. Rev.* D **74** 104005
[12] Santamaría L *et al* 2010 Matching post-Newtonian and numerical relativity waveforms: systematic errors and a new phenomenological model for nonprecessing black hole binaries *Phys. Rev.* D **82** 064016
[13] Hannam M, Schmidt P, Bohé A, Haegel L, Husa S, Ohme F, Pratten G and Pürrer M 2014 Simple model of complete precessing black-hole-binary gravitational waveforms *Phys. Rev. Lett.* **113** 151101
[14] Husa S, Khan S, Hannam M, Pürrer M, Ohme F, Jiménez Forteza X and Bohé A 2016 Frequency-domain gravitational waves from nonprecessing black-hole binaries. I. New numerical waveforms and anatomy of the signal *Phys. Rev.* D **93** 044006
[15] Khan S, Husa S, Hannam M, Ohme F, Pürrer M, Jiménez Forteza X and Bohé A 2016 Frequency-domain gravitational waves from nonprecessing black-hole binaries. II. A phenomenological model for the advanced detector era *Phys. Rev.* D **93** 044007
[16] Babak S, Taracchini A and Buonanno A 2017 Validating the effective-one-body model of spinning, precessing binary black holes against numerical relativity *Phys. Rev.* D **95** 024010
[17] Pürrer M 2016 Frequency domain reduced order model of aligned-spin effective-one-body waveforms with generic mass ratios and spins *Phys. Rev.* D **93** 064041
[18] Field S E, Galley C R, Hesthaven J S, Kaye J and Tiglio M 2014 Fast prediction and evaluation of gravitational waveforms using surrogate models *Phys. Rev.* X **4** 031006
[19] Field S E, Galley C R and Ochsner E 2012 Towards beating the curse of dimensionality for gravitational waves using reduced basis *Phys. Rev.* D **86** 084046
[20] Devine C, Etienne Z B and McWilliams S T 2016 Optimizing spinning time-domain gravitational waveforms for Advanced LIGO data analysis *Class. Quantum Grav.* **33** 125025
[21] Taracchini A *et al* 2014 Effective-one-body model for black-hole binaries with generic mass ratios and spins *Phys. Rev.* D **89** 061502
[22] Stallman R M and GCC DeveloperCommunity 2015 *Using The Gnu Compiler Collection* (Boston)
[23] Intel 2014 User and reference guide for the Intel C++ compiler 15.0 (https://software.intel.com/en-us/compiler_15.0_ug_c)
[24] Wolfram Research, Inc. 2016 Mathematica 10.4 (Champaign, IL) (https://wolfram.com)
[25] Limare N 2014 Floating-point math speed vs precision (http://nicolas.limare.net/pro/notes/2014/12/16_math_speed/)
[26] Press W H, Teukolsky S A, Vetterling W T and Flannery B P 2007 *Numerical Recipies Art of Scientific Computing* 3rd edn (Cambridge: Cambridge University Press)

[27] Nitz A *et al* 2017 ligo-cbc/pycbc: O2 Production Release 11 (https://doi.org/10.5281/zenodo.556097)
[28] Dal Canton T *et al* 2014 Implementing a search for aligned-spin neutron star-black hole systems with advanced ground based gravitational wave detectors *Phys. Rev.* D **90** 082004
[29] Usman S A *et al* 2016 The PyCBC search for gravitational waves from compact binary coalescence *Class. Quantum Grav.* **33** 215004
[30] Bohé A *et al* 2017 Improved effective-one-body model of spinning, nonprecessing binary black holes for the era of gravitational-wave astrophysics with advanced detectors *Phys. Rev.* D **95** 044028
[31] LSC 2010 Advanced LIGO anticipated sensitivity curves (https://dcc.ligo.org/LIGO-T0900288/public)