Supplemental material: High-accuracy mass, spin, and recoil predictions of generic black-hole merger remnants

Vijay Varma,[1,*] Davide Gerosa,[1,†] François Hébert,[1,‡] Leo C. Stein,[1,2,§] and Hao Zhang[1,3,¶]

[1] *TAPIR 350-17, California Institute of Technology, 1200 E California Boulevard, Pasadena, CA 91125, USA*
[2] *Department of Physics and Astronomy, The University of Mississippi, University, MS 38677, USA*
[3] *Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, PA 19104, USA*

Gaussian process regression– We construct fits in this work using Gaussian process regression (GPR) [S1, S2] as implemented in *scikit-learn* [S3].

The starting point is a training set of n observations, $\mathcal{TS} = \{(x^i, f(x^i))|i = 1, \ldots, n\}$, where x^i denotes an input vector of dimension D and $f(x^i)$ is the corresponding output. In our case, x is mass ratio and spins of the merging binary, and $f(x)$ is the remnant property we are fitting. Our goal is to use $\mathcal{TS}$ to make predictions for the underlying $f(x)$ at any point x_* that is not in $\mathcal{TS}$.

A Gaussian process (GP) can be thought of as a probability distribution of functions. More formally, a GP is a collection of random variables, any finite number of which have a joint Gaussian distribution [S1]. A GP is completely specified by its mean function $m(x)$ and covariance function $k(x, x')$, i.e. $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$. Consider a prediction set of n_* test inputs and their corresponding outputs (which are unknown): $\mathcal{PS} = \{(x^i_*, f(x^i_*))|i = 1, \ldots, n_*\}$. By the definition of a GP, outputs of $\mathcal{TS}$ and $\mathcal{PS}$ (respectively $\mathbf{f} = \{f(x^i)\}$, $\mathbf{f}_* = \{f(x^i_*)\}$) are related by a joint Gaussian distribution

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_{xx} & K_{xx_*} \\ K_{x_*x} & K_{x_*x_*} \end{bmatrix}\right), \tag{S1}$$

where K_{xx_*} denotes the $n \times n_*$ matrix of the covariance $k(x, x_*)$ evaluated at all pairs of training and prediction points, and similarly for the other K matrices.

Eq. (S1) provides the Bayesian prior distribution for $\mathbf{f}_*$. The posterior distribution is obtained by restricting this joint prior to contain only those functions which agree with the observed data points, i.e. [S1]

$$p(\mathbf{f}_*|\mathcal{TS}) = \mathcal{N}\left(K_{x_*x}K^{-1}_{xx}\mathbf{f},\, K_{x_*x_*} - K_{x_*x}K^{-1}_{xx}K_{xx_*}\right). \tag{S2}$$

The mean of this posterior provides an estimator for $f(x)$ at x_*, while its width is the prediction error.

Finally, one needs to specify the covariance (or kernel) function $k(x, x')$. In this *Letter* we implement the following kernel

$$k(x, x') = \sigma_k^2 \exp\left[-\frac{1}{2}\sum_{j=1}^{D}\left(\frac{x^j - x'^j}{\sigma_j}\right)^2\right] + \sigma_n^2\,\delta_{x,x'}\,, \tag{S3}$$

where $\delta_{x,x'}$ is the Kronecker delta. In words, we use a product between a squared exponential kernel and a constant kernel, to which we add a white kernel term to account for additional noise in the training data [S1, S3].

GPR fit construction involves determining the $D+2$ hyperparameters (σ_k, σ_n and σ_j) which maximize the marginal likelihood of the training data under the GP prior [S1]. Local maxima are avoided by repeating the optimization with 10 different initial guesses, obtained by sampling uniformly in log in the hyperparameter space described below.

Before constructing the GPR fit, we pre-process the training data as follows. We first subtract a linear fit and the mean of the resulting values. Datapoints are then normalized by dividing by the standard deviation of the resulting values. The inverse of these transformations is applied at the time of the fit evaluation.

For each dimension of x, we define Δx^j to be the range of the values of x^j in $\mathcal{TS}$ and consider $\sigma_j \in [0.01 \times \Delta x^j, 10 \times \Delta x^j]$. Larger length scales are unlikely to be relevant and smaller length scales are unlikely to be resolvable. The remaining hyperparameters are sampled in $\sigma_k^2 \in [10^{-2}, 10^2]$ and $\sigma_n^2 \in [10^{-7}, 10^{-2}]$. These choices are meant to be conservative and are based on prior exploration of the typical magnitude and noise level in our pre-processed training data.

Input parameter space– Fits for *surfinBH3dq8* are parameterized using $x = [\log(q), \hat{\chi}, \chi_a]$, where $\hat{\chi}$ is the spin parameter entering the GW phase at leading order [S4–S7] in the PN expansion,

$$\chi_{\rm eff} = \frac{q\,\chi_{1z} + \chi_{2z}}{1+q}\,, \qquad \eta = \frac{q}{(1+q)^2}\,, \tag{S4}$$

$$\hat{\chi} = \frac{\chi_{\rm eff} - 38\eta(\chi_{1z} + \chi_{2z})/113}{1 - 76\eta/113}\,, \tag{S5}$$

and χ_a is the "anti-symmetric spin",

$$\chi_a = \tfrac{1}{2}(\chi_{1z} - \chi_{2z})\,. \tag{S6}$$

For *surfinBH7dq2* we use $x = [\log(q), \chi_{1x}, \chi_{1y}, \hat{\chi}, \chi_{2x}, \chi_{2y}, \chi_a]$. Subscripts x, y

* vvarma@caltech.edu
† Einstein Fellow; dgerosa@caltech.edu
‡ fhebert@caltech.edu
§ lcstein@olemiss.edu
¶ zhangphy@sas.upenn.edu

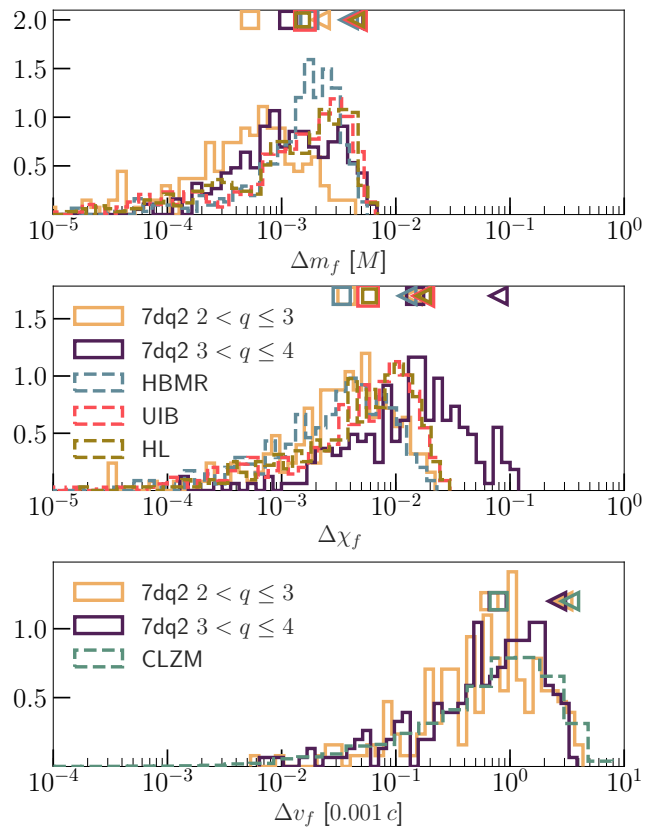

FIG. S1. Errors in *surfinBH7dq2* when extrapolating to higher mass ratios, and the spins are specified at an orbital frequency $f_0 = 10\,\mathrm{Hz}$, for a total mass $M = 70M_\odot$.

and z refer to components specified in the coorbital frame at $t = -100M$. We empirically found these parameterizations to perform more accurately than the more intuitive choice $x = [q, \chi_{1x}, \chi_{1y}, \chi_{1z}, \chi_{2x}, \chi_{2y}, \chi_{2z}]$.

In the main text we describe how we evolve spins given at earlier times to $t = -100M$, using PN and NRSur7qd2. Is it worth noting that the NR spins used to train NRSur7qd2 had some additional smoothening filters applied to them (see Eq. 6 in [S8]). This introduces additional systematics when evolving spins from times $t < -100M$. We verified that the resulting errors on our fits are subdominant.

Extrapolation erorrs– The right panel of Fig. 4 shows the errors in remnant quantities when extrapolating *surfinBH7dq2* to mass ratios beyond its training range ($q \leq 2$). These errors are computed using the spins at $t = -100M$. If the spins are given at earlier times, we expect larger extrapolation errors as this also involves extrapolation of the NRSur7dq2 waveform model (which was also trained in the $q \leq 2$ space). Figure S1 shows the extrapolation errors when the spins are specified at at orbital frequency $f_0 = 10\,\mathrm{Hz}$ for a total mass $M = 70M_\odot$, computed by comparing against the same NR simulations as in Fig. 4. Errors are comparable to or lower than those of existing fits for $q \leq 3$. For $3 < q \leq 4$, our errors for the remnant spin magnitude can become larger, but the remnant mass and kick magnitude remains as accurate as in other fits.

Figure S2 shows errors in *surfinBH3dq8* when extrapolated beyond its training space to higher mass ratios and/or spin magnitudes (this figure complements the results shown in Fig. 4 of the main text for *surfinBH7dq2*). Here we used some of the simulations of [S9–S13] with $q > 8$ and/or $\chi_1, \chi_2 > 0.8$. Accuracy in the remnant mass degrades noticeably only at high (~ 0.9) co-aligned spins. Errors in final spin become larger at both high spins and extreme mass ratios. For counter-aligned spins, our errors are always comparable to those found within the training region. Errors in kick magnitude and direction appear to be insensitive to extrapolation.

GPR error prediction– As stressed above and in the main body of our *Letter*, GPR naturally associates errors to the estimated quantities. In this Section we test the efficacy of this prediction by comparing the GPR errors against the out-of-sample errors. The GPR errors shown here are evaluated using the same cross-validation data sets used to generate the out-of-sample errors. Therefore, both error estimates are evaluated at points in parameter space where models were not trained.

Error comparisons for *surfinBH3dq8* and *surfinBH7dq2* are reported in Figs. S3 and S4, respectively. While GPR predictions miss some of the features captured by the "k-fold" cross validations, overall it provides faithful estimates of the fit errors.

Public python implementation– Our fits are made publicly available through the easy-to-use Python package, *surfinBH* [S14]. Our code is compatible with both `Python 2` and `Python 3`. The latest release can be installed from the Python Package Index using

```
pip install surfinBH
```

Python packages *numpy* [S15], *scipy* [S16], *h5py* [S17], *scikit-learn* [S3], *lalsuite* [S18], and *NRSur7dq2* [S8] are specified as dependencies and are automatically installed if missing. *surfinBH* is hosted on GitHub at github.com/vijayvarma392/surfinBH, from which development versions can be installed. Continuous integration is provided by *Travis* [S19]

The *surfinBH* module can be imported in Python using

```
import surfinBH
```

Documentation is provided for each submodule of surfinBH and can be accessed via Python's `help()` function. The fit class has to be initialized using, e.g.

```
fit = surfinBH.LoadFits("surfinBH7dq2")
```

Given mass ratio and component spins, the fits and 1σ GPR error estimates of the remnant mass, spin vector and kick vector can be evaluated as follows:

```
q = 1.2
chiA = [0.5, 0.05, 0.3]
```

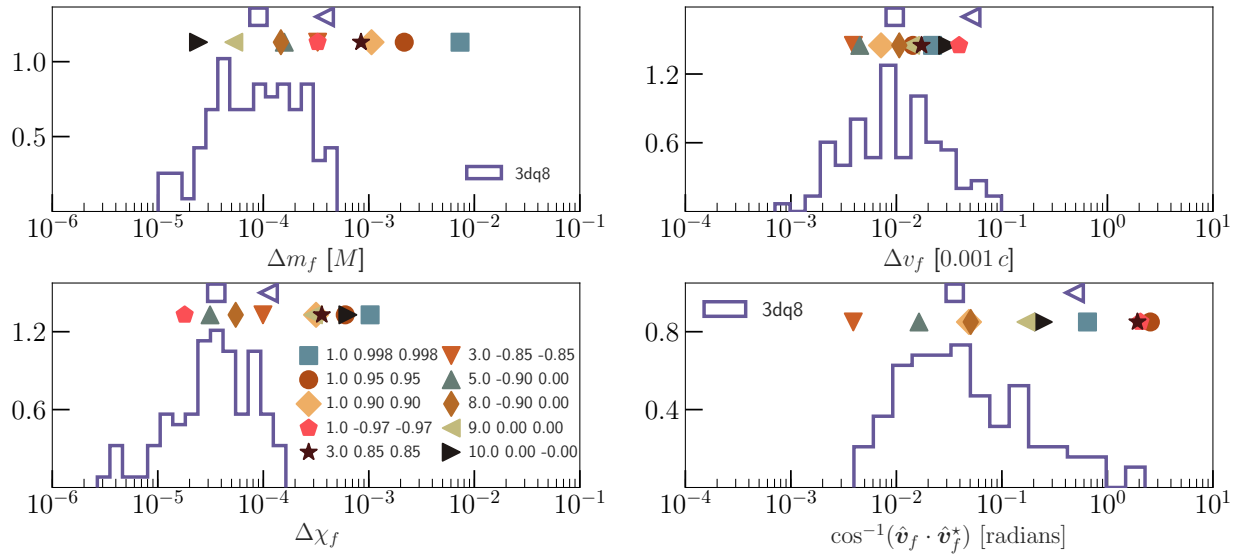

FIG. S2. Errors in predicting the remnant mass, spin, kick magnitude and kick direction for nonprecessing BBH when *surfinBH3dq8* is extrapolated outside of the training region (i.e. $q > 8$ and $\chi_1, \chi_2 > 0.8$). Each solid symbol marks the error of the extrapolated model against a single nonprecessing NR simulation. The legend in the bottom-left panel displays the mass ratio and spin components of the two BHs along the orbital angular momentum direction. Histograms of errors within the training region (from Fig. 2) are reproduced here for comparison. The hollow square (triangle) markers indicate the median (95^{th} percentile) values for those errors.

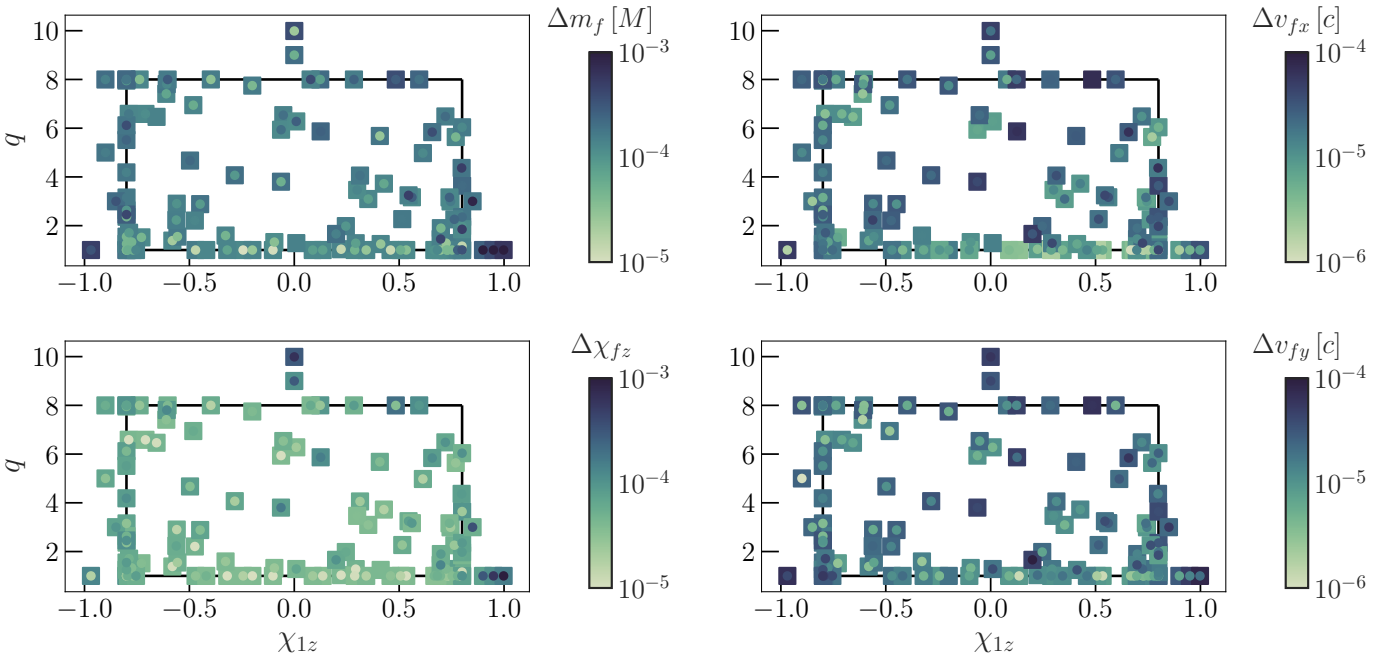

FIG. S3. Prediction errors for remnant mass, spin and kick for the model *surfinBH3dq8* against NR simulations. Two error estimates, as reported on the color scale, are compared: out-of-sample errors marked with circles, and 1σ GPR errors marked with squares. We include cases where *surfinBH3dq8* needs to be extrapolated to higher mass ratios and/or spin magnitudes. The bounds of the training parameter space are indicated as a black rectangle.

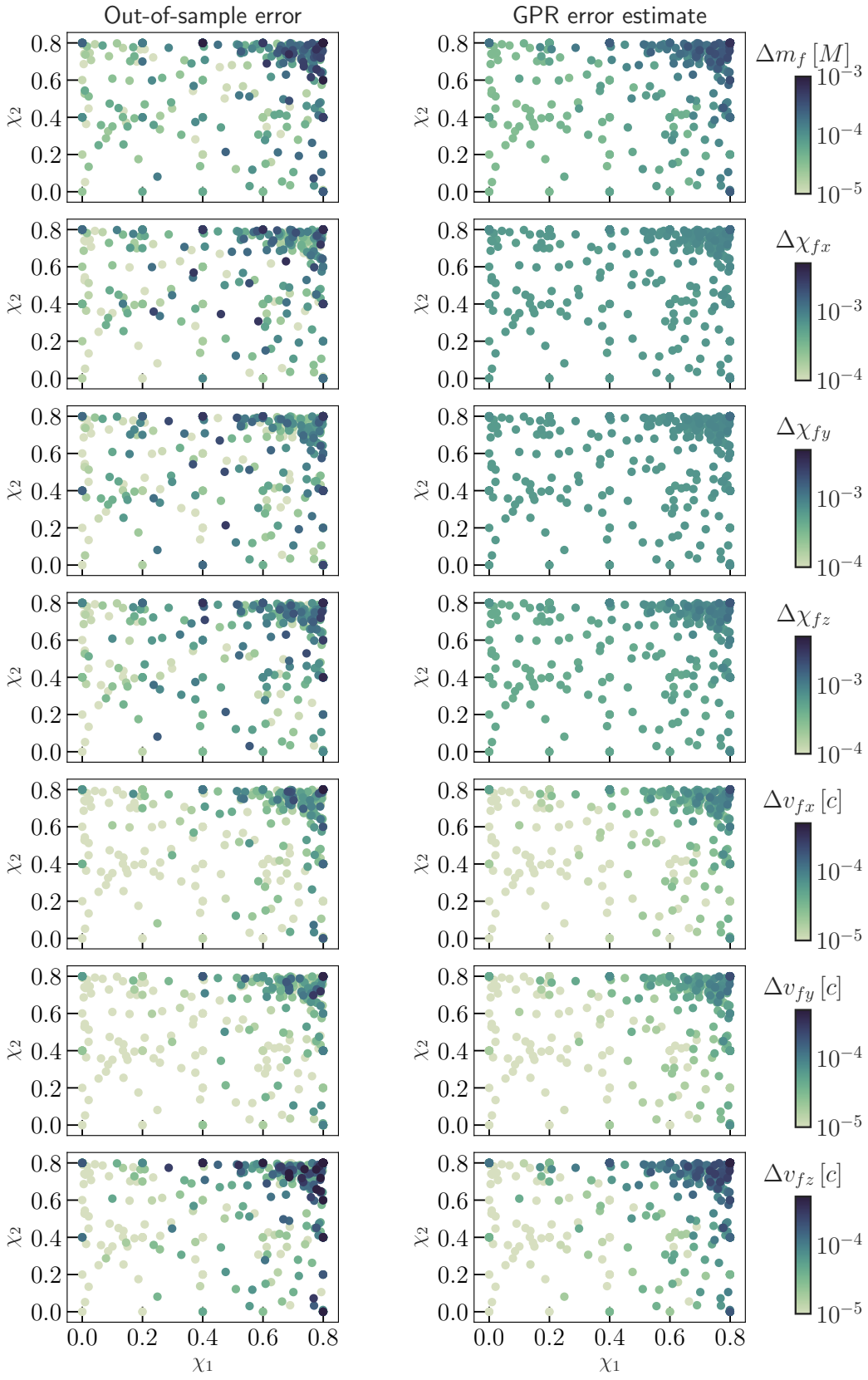

FIG. S4. Comparison between out-of-sample (left) and 1σ GPR (right) errors for *surfinBH7dq2*. The axes show the magnitudes of the component spins, and the color scale indicates the parameter error being plotted.

```
chiB = [-0.5, -0.05, 0.1]
mf, mf_err = fit.mf(q, chiA, chiB)
chif, chif_err = fit.chif(q, chiA, chiB)
vf, vf_err = fit.vf(q, chiA, chiB)
```

Both the input spins as well as the remnant spin and kick vectors are assumed to be specified in the coorbital frame at $t=-100M$. Performance of *surfinBH* was tested on a 3.1 GHz Intel Core i5 processor by averaging over 10^3 evaluations at randomly chosen points in parameter space. For *surfinBH7dq2*, evaluation cost of final mass (spin) [kick] is 2.5 ms (7 ms) [7 ms]. For *surfinBH3dq8*, evaluation cost of final mass (spin) [kick] is 0.4 ms (0.4 ms) [0.6 ms].

We also allow specifying an orbital frequency (in units of rad/M), e.g.:

```
omega0 = 5e-3
mf, mf_err = fit.mf(q, chiA, chiB,
                    omega0 = omega0)
chif, chif_err = fit.chif(q, chiA, chiB,
                          omega0 = omega0)
vf, vf_err = fit.vf(q, chiA, chiB,
                    omega0 = omega0)
```

In this case, the component spins, as well as the final remnant spin/kick vectors are specified in the coorbital frame at this orbital frequency. The evaluation costs are larger when specifying an initial orbital frequency since this involves two additional stages of spin evolution. Execution times depend on the initial frequency, the specific PN approximant used and the time step size in the integration routine. For instance, with `omega0 = 5e-3`, SpinTaylorT4, and a step size of $0.1M$ the evaluation cost is ~ 0.5s for each of the remnant quantities.

Additional resources are provided in the package installation page [S14]. This includes example `jupyter` notebooks for both models presented in this *Letter*.

[S1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning, by C.E. Rasmussen and C.K.I. Williams. ISBN-13 978-0-262-18253-9* (2006).

[S2] D. J. C. Mackay, *Information Theory, Inference and Learning Algorithms, by David J. C. MacKay, pp. 640. ISBN 0521642981. Cambridge, UK: Cambridge University Press, October 2003.* (2003) p. 640.

[S3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2012), 1201.0490.

[S4] S. Khan, S. Husa, M. Hannam, F. Ohme, M. Pürrer, X. J. Forteza, and A. Bohé, PRD **93**, 044007 (2016), arXiv:1508.07253 [gr-qc].

[S5] P. Ajith, PRD **84**, 084037 (2011), arXiv:1107.1267 [gr-qc].

[S6] C. Cutler and É. E. Flanagan, PRD **49**, 2658 (1994), gr-qc/9402014.

[S7] E. Poisson and C. M. Will, PRD **52**, 848 (1995), gr-qc/9502040.

[S8] J. Blackman, S. E. Field, M. A. Scheel, C. R. Galley, C. D. Ott, M. Boyle, L. E. Kidder, H. P. Pfeiffer, and B. Szilágyi, PRD **96**, 024058 (2017), arXiv:1705.07089 [gr-qc].

[S9] A. H. Mroué, M. A. Scheel, B. Szilágyi, H. P. Pfeiffer, M. Boyle, D. A. Hemberger, L. E. Kidder, G. Lovelace, S. Ossokine, N. W. Taylor, A. Zenginoğlu, L. T. Buchman, T. Chu, E. Foley, M. Giesler, R. Owen, and S. A. Teukolsky, PRL **111**, 241104 (2013), arXiv:1304.6077 [gr-qc].

[S10] P. Kumar, K. Barkett, S. Bhagwat, N. Afshari, D. A. Brown, G. Lovelace, M. A. Scheel, and B. Szilágyi, PRD **92**, 102001 (2015), arXiv:1507.00103 [gr-qc].

[S11] J. Blackman, S. E. Field, C. R. Galley, B. Szilágyi, M. A. Scheel, M. Tiglio, and D. A. Hemberger, PRL **115**, 121102 (2015), arXiv:1502.07758 [gr-qc].

[S12] T. Chu, H. Fong, P. Kumar, H. P. Pfeiffer, M. Boyle, D. A. Hemberger, L. E. Kidder, M. A. Scheel, and B. Szilagyi, CQG **33**, 165001 (2016), arXiv:1512.06800 [gr-qc].

[S13] M. Boyle *et al.*, (2018), in preparation.

[S14] V. Varma *et al.*, pypi.org/project/surfinBH, doi.org/10.5281/zenodo.1418525.

[S15] S. van der Walt, S. Colbert, and G. Varoquaux, Computing in Science Engineering **13**, 22 (2011).

[S16] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," `http://www.scipy.org/` (2001–).

[S17] A. Collette, *Python and HDF5* (O'Reilly, 2013).

[S18] LIGO Scientific Collaboration and Virgo Collaboration, git.ligo.org/lscsoft/lalsuite.

[S19] Travis Continuous Integration, travis-ci.org.