one-body framework, but it has only been calibrated to a few non-precessing NR simulations and requires significant computational resources to generate waveforms [15]. Data-driven models exist as well, such as NR surrogates, which fit to existing NR simulations to facilitate interpolation of waveforms at new parameter points [16, 17]. As computational resources for generating NR waveforms increase, data-driven models can more readily be used for PE, in addition to calibration and testing of approximate models (e.g. [18]).

Here we propose a method to obtain reduced-order-model (ROM) waveforms and "interpolation" uncertainties using only a training set of simulations at a small number of points in parameter space. The impetus for using ROMs, which are concise representations of waveforms, is that our method interpolates between waveforms "observed" in simulations and hence benefits computationally from reduced waveform dimensionality. That is, if a time-series waveform with N points can instead be represented by a list of $m < N$ features, only m, rather than N, evaluations are required to produce a gravitational waveform with source parameters $\vec{\lambda}$. Here we employ an interpolation technique known as Gaussian process regression (GPR). The advantage of a Gaussian process (GP) method is that it is *statistical* and fast. We describe our basic method in §II, which largely replicates the ROM work of [19], but instead of using spline interpolation to provide point predictions we use GPR to provide a statistical interpolation with uncertainties. The primary application of our method would be to use NR simulations to train a flexible, NR-driven GPR waveform model with quantified uncertainties. Although the end goal is to use NR simulations for training, in this paper we only present a proof of concept. Rather than NR waveforms, we use approximant waveforms from IMRPhenomD [20] to train and cross-validate, since they can be generated quickly and for a wide range of parameter values. §III details the results of cross-validating GPR models trained on a small number of IMRPhenomD waveforms with the IMRPhenomD waveforms themselves. We find that the GPR model, trained on just a small subsample of IMRPhenomD waveforms, is able to reproduce IMRPhenomD waveforms to excellent accuracy. Future work will implement an NR training set in the GPR model.

One major advantage of the use of GPR methods is that they naturally provide estimates of the errors in the resulting waveforms. Current waveform approximants used in LIGO PE are implicitly assumed to be perfect, although multiple approximants are used to assess systematic errors. A more refined analysis, leading to improved PE results, would incorporate errors in the waveform approximants, especially as a function of location in parameter space. The GPR methods naturally provide a statistically consistent estimate of these errors. [21] and [22] proposed an improvement to PE in which GPR is used to infer the systematic error on an approximant over the parameter space. They train their GP on the systematic errors between approximants and simulations for parameters where simulations exist. The systematic errors inferred from GPR are then propagated to the likelihood function used in PE. A proof of concept of this systematic error interpolation method has only been done in 1-d, and it requires an existing approximant off of which to interpolate the errors. In contrast, our method requires no other approximants. Additionally, we consider both 1- and 2-d in this work, though extending to more dimensions is straightforward in principle.

A further advantage of GPR methods and the associated waveform error estimates is that these can be used to optimize the placement of new simulations which can be added to the training set. In §IV we describe a method for estimating where in parameter space the GPR model has high error, based only on the GPR uncertainties. This estimate is a natural metric for "greedily" deciding where new simulations are needed to minimize GPR model errors. In effect, our method will tell you the "optimal" place in parameter space to run the next simulation. An iterative procedure of GPR building and NR simulation leads to an efficient training set and GPR interpolation based on fewer training points than a naive regular grid. We outline that iterative procedure here:

1. Use existing (NR) simulations to build a GPR model with uncertainties.

2. Use GPR uncertainties to estimate where in parameter space errors in the GPR model are highest.

3. Generate new simulations at parameter values with high estimated GPR error and rebuild a GPR model with the augmented training set that includes the new simulations.

4. Repeat.

In §V we discuss alternate choices that could be made with respect to GPR modeling of GWs, as the ideas presented herein serve more as a framework for GPR GW models than as an immutable, specific method. We also consider the evaluation time of GPR models, which vary in speed depending on the number of training waveforms and ROM coefficients. We conclude in §VI.

II. METHOD

A. Outline for building GPR models

To minimize the computational expense of waveform interpolation at source parameters $\vec{\lambda}$, we represent the waveforms with a ROM. We utilize the method described in Pürrer [19] (hereafter P14), but other choices for building ROMs could also be made (e.g. [23]). The steps we take to create a GPR waveform interpolation model based on simulations are:

1. Simulate $n_{\rm train}$ frequency-domain waveforms at $\vec{\lambda} = \{\vec{\lambda}_j\}_{j=1}^{n_{\rm train}}$ to cover the parameter space of interest for use as the training set.

2. Project waveforms into a reduced-order basis so that a waveform with parameters $\vec{\lambda}$ is described by a list of coefficients $\{c_i(\vec{\lambda})\}_{i=1}^{m}$, where $m = \dim(\text{basis})$.

3. Regularize each c_i function by subtracting a linear fit of the training values $c_i(\{\vec{\lambda}_j\})$ and normalizing. Define $\tilde{c}_i \equiv \text{Regularize}(c_i)$.

4. Assume each $\tilde{c}_i(\vec{\lambda})$ is a realization of a Gaussian process, and use $\tilde{c}_i(\{\vec{\lambda}_j\})$ as the training set to regress $\tilde{c}_i(\vec{\lambda}) \approx \tilde{c}_{i,\rm GP}(\vec{\lambda})$.

5. Transform $\{\tilde{c}_i(\vec{\lambda})\}$ and their uncertainties to the domain of interest (e.g. frequency domain for LIGO PE) by applying the reverse of the regularizations in step 3 and then projecting the coefficients out of the ROM basis back to the time or frequency domain. The specific operations to go from the ROM to the time or frequency domain depend on which ROM is used.

We discuss all of these steps in detail below.

B. Waveform Generation and Representation

We generate fixed-chirp-mass, frequency-domain IMRPhenomD waveforms (with a starting frequency of 20 Hz) at various source parameter values, and interpolate the amplitudes and phases separately onto sparse frequency grids as in §5.1 of P14[24]. For simplicity, only the h_+ components of the waveforms are considered here, but the methods presented herein can be identically applied to $h_\times$, or h_+ can be used to calculate $h_\times$ as in equations 6.14 and 6.15 in P14. For the sparse frequency grid, we choose an amplitude grid spacing at frequency f of $\Delta_A = 0.1f$ and a phase grid spacing of $\Delta_\Phi = 0.3f^{4/3}$; P14 found that these grid spacings keep a constant spline interpolation error at all frequencies. The interpolated amplitudes and phases are then packed into the columns of matrices $\mathcal{T}_A$, $\mathcal{T}_\Phi$, and a SVD is performed on each matrix (see §6 of P14):

$$\begin{aligned} \mathcal{T}_A &= V_A \Sigma_A U_A^\top, \\ \mathcal{T}_\Phi &= V_\Phi \Sigma_\Phi U_\Phi^\top. \end{aligned} \tag{1}$$

P14 truncates the V matrices to reduce the dimensionality of the ROM, but for simplicity we instead directly compute projection coefficients $c(\tau)$ for each input amplitude or phase τ (amplitude and phase subscripts dropped for ease of notation):

$$c(\tau) = V^\top \tau. \tag{2}$$

Each element of c is a function of τ and hence a function of the input parameters $\vec{\lambda}$. The following sections describe how the elements of c are interpolated across the parameter space to extract new waveforms with uncertainties.

C. Gaussian Process Regression

GPR is a tool for emulating (i.e., statistically inferring) the behavior of functions of continuous variables. It is commonly used to predict the output of simulations which are too expensive to run for many parameter values. In the case of GWs, we wish to infer new strain waveforms from existing NR simulations. The basic assumption in GPR is that any finite subset of values of a process $f(\vec{x})$ have a joint Gaussian distribution and thus can be described by a mean function $\vec{\mu}$ and covariance function $\mathbf{k}(\vec{x}, \vec{x}')$:

$$f(\vec{x}) \sim \mathcal{GP}(\vec{\mu}, \mathbf{k}(\vec{x}, \vec{x}')). \tag{3}$$

Given a list of known values of a function $\mathbf{f} = \{f_1(\vec{x}_1), f_2(\vec{x}_2), ...\}$ at $n_{\rm train}$ training points $X = \{\vec{x}_1, \vec{x}_2, ...\}$, we can calculate the probability distribution for $\mathbf{f}_* = \{f_{1*}(\vec{x}_{1*}), f_{2*}(\vec{x}_{2*}), ...\}$ at new points $X_* = \{\vec{x}_{1*}, \vec{x}_{2*}, ...\}$, using the definition of a Gaussian process: the prediction and the known values have a joint Gaussian distribution. If the mean of the GP prior in Equation 3 is zero[25] then:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right), \tag{4}$$

where $\mathcal{N}(\mathbf{m}, \mathbf{K})$ is a multivariate normal distribution with mean $\mathbf{m}$ and covariance matrix $\mathbf{K}$. $K(X,X)$, $K(X,X_*)$, $K(X_*,X)$, and $K(X_*,X_*)$ are the matrices of covariances between pairs of training and prediction points, the elements of which are calculated using the covariance function $\mathbf{k}(\vec{x}, \vec{x}')$. The conditional probability distribution of $\mathbf{f}_*$ given $\mathbf{f}$ is itself Gaussian (see e.g. [26]):

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{f}) = \mathcal{N}\Big(& K(X_*,X)K(X,X)^{-1}\mathbf{f}, \\ & K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*) \Big). \end{aligned} \tag{5}$$

With covariances and a training set, $\mathbf{f}$, one can calculate the conditional mean and covariance of $\mathbf{f}_*$. In GPR, the entries of the covariance matrices are computed using a covariance function, or kernel, which is specified by the user and depends on the application. The kernels are symmetric and hence only depend on the distance between points $r = |x - x'|$. Here we primarily consider the squared-exponential kernel, but we discuss the choice of covariance function in §V. A squared-exponential covariance constrains the process to be infinitely mean-square

differentiable[27] and takes the form of a Gaussian:

$$\mathbf{k}_{SE}(\vec{x}_1, \vec{x}_2) = \mathbf{k}_{SE}(r = |\vec{x}_1 - \vec{x}_2|) = \sigma^2 \exp\left(-\frac{1}{2} r^2/l^2\right). \tag{6}$$

where σ and l are *hyperparameters* of the process and parameterize the signal variance and length scale, respectively. The hyperparameters of this kernel can be fixed a priori, but are typically chosen to maximize the *hyperlikelihood*, the likelihood of the training data under the GP prior (Equation 3). Here we instead maximize the *hyperposterior* which incorporates prior distributions on the hyperparameters:

$$\text{hyperposterior} \propto \text{hyperlikelihood} \times \text{hyperprior}. \tag{7}$$

The priors on the hyperparameters, or *hyperpriors*, are discussed further in §III.

In addition to selection of hyperparameters, one must choose small values called "nuggets"[28] to add to the diagonal of the training points covariance matrix $K(X, X)$ to ensure numerical stability when computing the conditional covariance. Computation of the conditional covariance can result in a non-positive-semi-definite matrix due to precision errors when the prior covariance matrix has large values off-diagonal. These nuggets are described further in §III.

D. Implementation of GPR-based models

We now return our focus to the training set projection coefficients $c_i(\vec{\lambda}_j)$ at points $\vec{\lambda}_j$ in the input parameter space. To reconstruct waveforms, P14 interpolates the training projection coefficients over the parameter space using a tensor spline interpolation. We instead use GPR in order to account for uncertainties in the interpolation. Rather than directly interpolate c_i, we interpolate *regularized* coefficients $\tilde{c}_i$. The training values $c_i(\{\vec{\lambda}_j\}_{j=1}^{n_{\rm train}})$ are regularized by first subtracting a linear fit over the parameter space, then normalizing the residual variance to 1 and removing the mean:

$$\begin{aligned} \Delta c_i(\{\vec{\lambda}_j\}) &\equiv c_i(\{\vec{\lambda}_j\}) - \text{linfit}\left(c_i(\{\vec{\lambda}_j\})\right) \\ \tilde{c}_i(\{\vec{\lambda}_j\}) &\equiv \frac{\Delta c_i(\{\vec{\lambda}_j\}) - \text{mean}\left(\Delta c_i(\{\vec{\lambda}_j\})\right)}{\text{std}\left(\Delta c_i(\{\vec{\lambda}_j\})\right)}. \end{aligned} \tag{8}$$

We then assume that each $\tilde{c}_i$ is a Gaussian process, and use $\tilde{c}_i(\{\vec{\lambda}_j\})$ as a training set to regress $\tilde{c}_i(\{\vec{\lambda}\})$. Thus for any point in parameter space $\vec{\lambda}$ we can predict $\tilde{c}_i(\vec{\lambda})$ and marginal uncertainties $\delta\tilde{c}_i(\vec{\lambda})$ which can then be transformed back to amplitude/phase by applying the reverse of the transformations used to generate $\tilde{c}_i(\{\vec{\lambda}_j\})$. Assuming c_i is uncorrelated with c_j unless $i = j$, the mean amplitude $A(F_k, \vec{\lambda})$ in the k-th frequency bin and covariance matrix $\Sigma_{kl}(\vec{\lambda})$ are given by[29]:

$$\begin{aligned} A(F_k, \vec{\lambda}) &= \sum_i V^A_{ki} c^A_{i,\rm GP}(\vec{\lambda}), \\ \Sigma^A_{kl}(\vec{\lambda}) &= \sum_i V^A_{ki} \delta c^A_{i,\rm GP}(\vec{\lambda}) V^{A\top}_{il}. \end{aligned} \tag{9}$$

The phase means and covariances, $\Phi(F_k, \vec{\lambda})$ and $\Sigma^\Phi_{kl}(\vec{\lambda})$, can be reconstructed similarly with the corresponding phase projection coefficients.

To assess the accuracy of our GPR model, we compare reconstructed frequency waveform means from GPR $h_{\rm GPR}(F_k, \vec{\lambda})$ with the waveforms from IMRPhenomD $h(F_k, \vec{\lambda})$ using the *mismatch* function. The mismatch between two frequency domain waveforms h_1 and h_2 is defined as:

$$\begin{aligned} &\text{mismatch}(h_1, h_2) = \\ &1 - \frac{4}{||h_1||||h_2||} \\ &\times \Re\left(\int_{f_{\rm min}}^{f_{\rm max}} \frac{h_1(f) h_2^*(f)}{S_f} \mathrm{d}f\right), \end{aligned} \tag{10}$$

where

$$||h|| = 4\Re\left(\int_{f_{\rm min}}^{f_{\rm max}} \frac{h(f)h(f)^*}{S_f} \mathrm{d}f\right). \tag{11}$$

$f_{\rm min}$ and $f_{\rm max}$ are the minimum and maximum frequencies at which the two waveforms are compared, respectively, and S_f is the noise power spectral density of a GW detector. In this paper, we use the aLIGO O1 noise curve `2015-10-01_H1_O1_Sensitivity_strain_asd.txt`[30].

III. RESULTS

We now implement GPR-based models for three sets of parameter spaces using the `GaussianProcessRegressor` module from `scikit-learn`[31]:

1. constant zero spin
   mass ratio $q \in [1, 6]$
2. equal-and-aligned spin $\chi_1 = \chi_2 \in [-1, 1]$
   constant $q = 1$
3. equal-and-aligned spin $\chi_1 = \chi_2 \in [-0.5, 0.5]$
   $q \in [1, 3]$

All parameter spaces use waveforms with a constant chirp mass $M_c = 20 M_\odot$, distance $D = 1$ Mpc, inclination angle $i = 0$, and starting frequency of 20 Hz.

A. 1-d GPR in Mass Ratio

Beginning with Parameter Space 1, we generate $n_{\rm train} = 15$ equally spaced IMRPhenomD waveforms

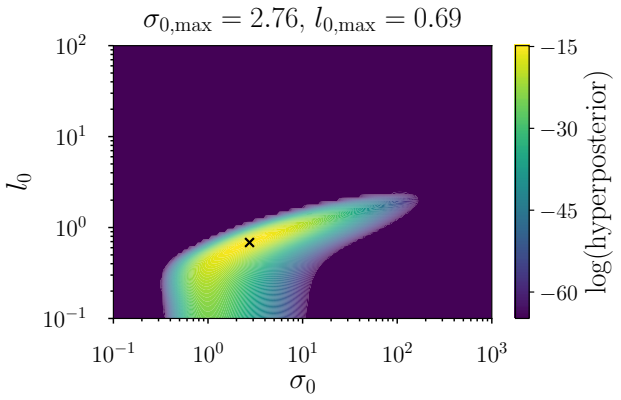

FIG. 1. Hyperposterior for the first regularized amplitude coefficient $\tilde{c}_0^A$ as a function of kernel covariance scale σ_0 and length scale l_0. The hyperparameter values with largest hyperposterior, $\sigma_{0,\max}$ and $l_{0,\max}$, are marked with an X on the plot and printed at the top.

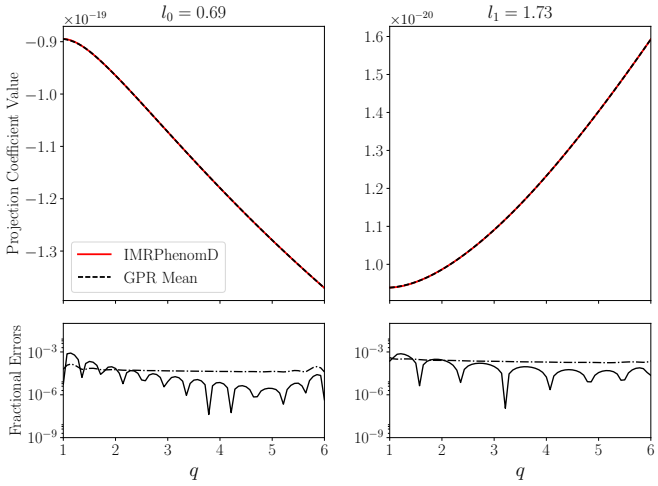

FIG. 2. First two amplitude coefficients for the IMRPhenomD waveforms and GPR-based waveforms with their residuals. *Top panels*: IMRPhenomD coefficient values (red) and GPR mean coefficient values (black, dashed) for the first two amplitude coefficients. The optimized length scales are shown above the top panels. Although the remaining coefficients are not shown here, they have similar morphologies. *Bottom panels*: the fractional residuals $(|c_i^A - c_{i,\mathrm{GP}}^A|)/c_i^A$ (solid) and the GPR fractional 1σ uncertainties $\delta c_{i,\mathrm{GP}}^A/c_i^A$ (dashed-dotted).

from $q = 1$ to $q = 6$ as a proxy for an NR training set and compute their amplitude and phase projection coefficients as described in §II B. The i-th coefficient is then de-trended by removing a linear fit of the 20 training coefficient values $c_i(\{\vec{\lambda}_j\})$. The de-trended $c_i(\{\vec{\lambda}_j\})$ are then normalized by their standard deviation and their mean is removed. We refer to the de-trended, de-meaned, normalized coefficients as $\tilde{c}_i$ and treat each one as a GP:

$$\tilde{c}_i \sim \mathcal{GP}(\vec{0}, \mathbf{k}_i(q, q')). \tag{12}$$

The $\tilde{c}_i$ can be trivially transformed back to c_i by reapplying the mean, standard deviation, and linear fit. We take the covariance function for the i-th coefficient, $\mathbf{k}_i(q, q')$, to be a squared exponential with hyperparameters σ_i and l_i.

To ensure numerical stability in the GPR conditional covariance matrix calculation, we add a nugget to the input covariances for each training value. We assume a constant relative error on the training waveform amplitude of 10^{-4} at each frequency and transform these errors to errors in the amplitude coefficients, which are used as the kernel nugget. For the training phases, the error at each frequency is kept below the nominal LIGO phase measurement uncertainty of ~ 0.1 radians by assuming a constant error of 0.01 radians at each phase value on the sparse frequency grid. These errors are projected to the coefficient errors analogously to the amplitude error case. The nugget levels here are chosen for numerical stability and adequate accuracy, but also roughly correspond to the resolution errors found in NR simulation studies [6]. In principle, the NR errors as a function of source parameters could be incorporated into the nugget values to fully account for the resolutions of different simulations.

To optimize the kernel hyperparameters for each coefficient, we use the `scipy.optimize` implementation [32] of the Broyden-Fletcher-Goldfarb-Shanno algorithm `fmin_l_bfgs_b` [33] to maximize the hyperposterior. We apply log-normal hyperpriors on σ_i and l_i:

$$\begin{aligned} \log_{10}(\sigma_i) &\sim \mathcal{N}(0, 0.5) \\ \log_{10}(l_i) &\sim \mathcal{N}\left(\log_{10}\left(\frac{1}{2}\mathrm{width}(\{q_j\}_1^{n_{\mathrm{train}}})\right), 1\right). \end{aligned} \tag{13}$$

Since the regularized coefficient functions being interpolated have been normalized by their standard deviation, we expect that $\sigma_i \sim 1$, motivating the hyperprior on σ_i above. We have less information about the length scales a priori, but we know that they should not be much shorter than the distance between the closest training points, nor should they be much larger than the width of the parameter space. As such, the normal distribution on $\log_{10}(l_i)$ is chosen to have a standard deviation of 1 (i.e. 1 order of magnitude) and peak at half the width of the parameter space spanned by the training set. Figure 1 shows the hyperposterior surface for the first amplitude coefficient as a function of the hyperparameters using the squared-exponential kernel. The hyperposteriors for other coefficients are similar in morphology to those shown here.

With optimized hyperparameters, the GP is used to interpolate each regularized coefficient on a grid five times finer than the training grid. With the GPR model of the $\tilde{c}_i$'s, we can calculate each c_i and transform them back to amplitudes and phases on the sparse frequency grid. Figures 2 and 3 show the values of the first two amplitude and phase c_i's, respectively, as a function of q from IMRPhenomD and from the GPR-based model. Also shown are the fit residuals, the GPR 1σ uncertainties, and the optimized length scale hyperparameter for

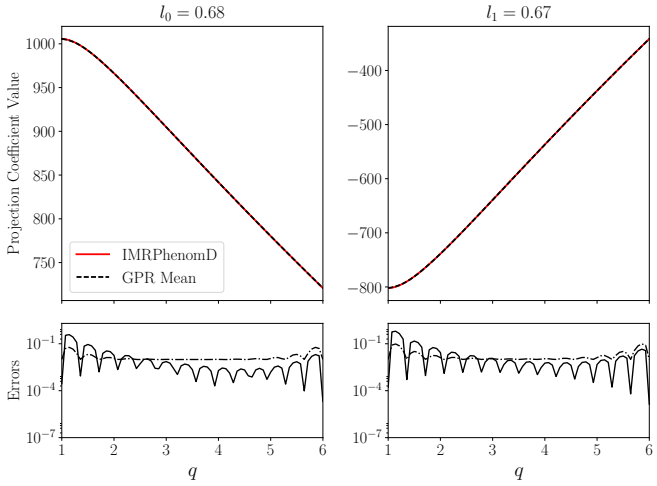

FIG. 3. First two phase coefficients for the IMRPhenomD waveforms and GPR-based waveforms with their residuals. *Top panels*: IMRPhenomD coefficient values (red) and GPR mean coefficient values (black, dashed) for the first two phase coefficients. The optimized length scales are shown above the top panels. Although the remaining coefficients are not shown here, they have similar morphologies. *Bottom panels*: the residuals $|c_i^\Phi - c_{i,\mathrm{GP}}^\Phi|$ (solid) and the GPR 1σ uncertainties $\delta c_{i,\mathrm{GP}}^\Phi$ (dashed-dotted).

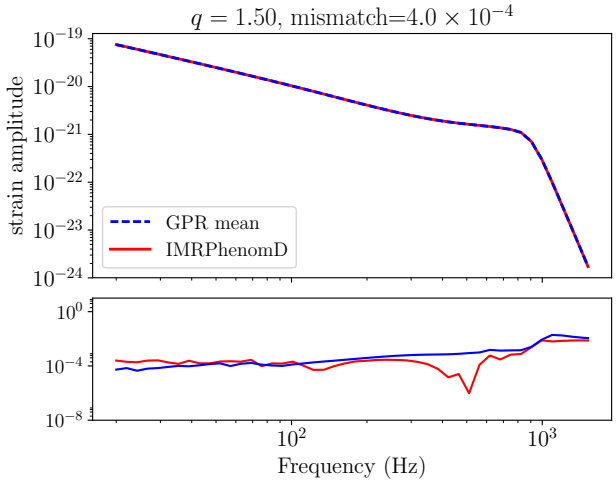

FIG. 4. Example reconstructed GPR amplitude function. *Top:* The interpolated GPR mean amplitude vs. frequency is shown in blue and the IMRPhenomD amplitude is overlaid in red. *Bottom:* The IMR-GPR-mean residual amplitude is shown in red, and the GPR 1σ uncertainty is shown in blue as a function of frequency. Both errors are normalized to the IMRPhenomD amplitude.

each coefficient function's GP. Although the GPR errors do not perfectly match the residuals across the parameter space, they are indicative of the maximum error level and of the fact that the errors are largest on the edges of the space. The interpolated coefficients and their uncertainties are then propagated back to amplitudes and

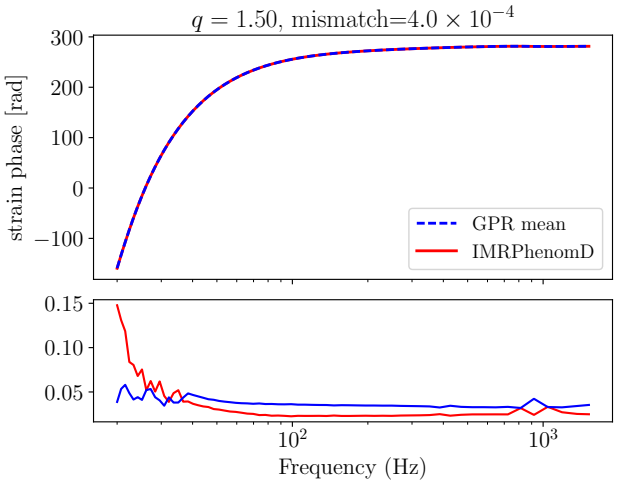

FIG. 5. Example reconstructed GPR phase function. *Top:* The interpolated GPR mean phase vs. frequency is shown in blue and the IMRPhenomD phase is overlaid in red. *Bottom:* The absolute value of the IMR-GPR residual phase is shown in red, and the GPR 1σ uncertainty is shown in blue as a function of frequency.

phases on the sparse frequency grid. One such example of a GPR-interpolated waveform is shown in Figures 4 and 5, which show the amplitude and phase functions, respectively. Notably, the fractional amplitude error between the GPR model and IMRPhenomD waveform is largest at high frequencies, because small errors in the amplitude coefficients combine when projected back to the sparse frequency grid domain. Nonetheless, the GPR mean agrees well with the IMRPhenomD model, and the GPR uncertainties give a reasonable indication of the true error levels at different frequency bins. Since PE is done in the frequency domain, these errors do not need to be propagated to the time domain, although frequency-domain waveforms can be sampled to create a distribution of time-domain waveforms if required. Figure 6 shows the mismatch between the GPR mean waveform and the IMR waveform at various values of q. The mismatch increases near the boundaries of the space where there are fewer nearby training points (training points shown in dashed black vertical lines). Indeed, the mismatches and residuals suggest that more training points (i.e. simulations) are needed towards the edge of the space of interest. Increasing the number of training points typically lowers the mismatches, although it depends on the nugget and hyperparameters used in the GP. The relative placement of the simulations is also of interest, which we discuss in §IV.

B. 1-d GPR in equal-and-aligned spin

Following an analogous method to that presented in §III A, we generate 12 IMRPhenomD waveforms span-

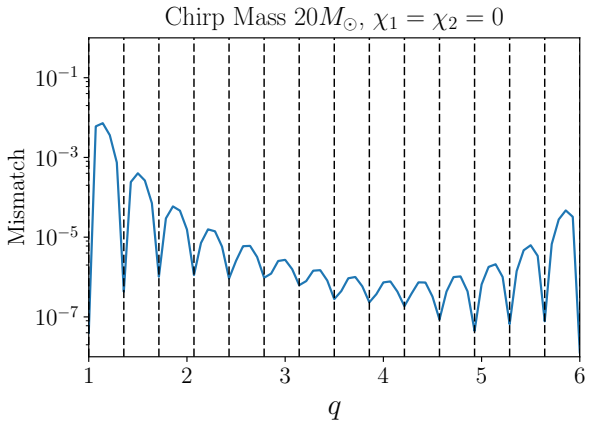

FIG. 6. Mismatch between the IMRPhenomD waveform and the GPR mean waveform for different mass ratios assuming a constant chirp mass and zero spin. The dashed, vertical lines show the IMRPhenomD training waveform locations used to build the GPR model.

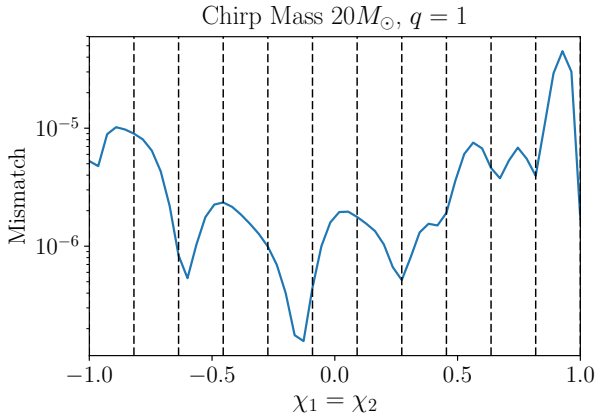

FIG. 7. Mismatch between the IMRPhenomD waveform and the GPR mean waveform for different equal-and-aligned spin values assuming a constant chirp mass and equal masses ($q = 1$). The dashed, vertical lines show the IMRPhenomD training waveform locations used to build the GPR model.

ning equally-spaced equal-and-aligned spin[34] values from $\chi = -1$ to $\chi = 1$. The chirp mass of $M_c = 20M_\odot$ and mass ratio $q = 1$ are held fixed. A GPR-based waveform model is built using these waveforms as a training set, again using a squared-exponential covariance function to model each $\tilde{c}_i$. The hyperpriors on the length scale and covariance scale are chosen in the same way as in §III A. The GPR model is evaluated on a grid five times finer than the training grid and is then compared to the waveforms predicted in IMRPhenomD via the mismatch function. Figure 7 shows the mismatch between the GPR mean waveform and IMRPhenomD for the 1-d space of equal-and-aligned spin χ. Similarly to the case of mass-ratio space, the mismatch between the GPR model and IMRPhenomD is largest at the boundaries of the training set. Nevertheless, we are able to achieve low mismatches with just a few training points in this case.

C. 2-d GPR in equal aligned spin and mass ratio

We again build a training set using IMRPhenomD waveforms as a proxy for NR simulations in order to train a GPR-based model, except here we vary two source parameters: The mass ratio and the value of the equal-and-aligned spins. The training waveforms are generated on a regular grid with 15 points in q and 8 points in χ from $q \in [1, 3]$ and $\chi \in [-0.5, 0.5]$. The kernel used here is simply an overall covariance σ times the product of two squared-exponential kernels, one for the q dimension and one for the χ dimension. As such, there are now two length scale parameters, l_q and l_χ, as well as the overall covariance σ which need to be fit for each coefficient function. Hyperpriors for the hyperparameters are chosen as in previous sections, and the hyperparameters are optimized via the hyperposterior. For computational savings, smaller windows in q and χ are considered here than in the previous sections. Extending to the full parameter space would simply require more training points[35]. Another option would be to decompose the extended domain into smaller overlapping patches and build a GPR model for each patch (e.g. Figure 2 of [36]). Although a regular, equal-spacing grid leads to reasonable mismatches in this case, there is no a priori reason to use such a grid, and in practice the existing simulations will have non-regular placements throughout the parameter space. Figure 8 shows the "true" values, interpolations, and residuals of the first amplitude SVD coefficient c_0^A as a function of q and χ. The top left panel is a color map of the coefficient values from IMRPhenomD. The top middle and right panels show $c^A_{0,\mathrm{GP}}$ and $c^A_{0,\mathrm{spline}}$, which are the interpolations of c_0^A with GPR and B-splines, respectively. The bottom panels show the fractional residuals of the interpolants and the estimated error from GPR. The residuals between IMRPhenomD and the GPR model (bottom left panel) for this amplitude coefficient are below the 0.1% level and are comparable to the predicted GPR uncertainties, providing evidence for the accuracy and precision of the GPR model. Comparing the bottom left and bottom right panels of Figure 8, the GPR mean is roughly as accurate at the B-spline interpolation for most coefficients, but the spline does not give any information about interpolation errors. The bottom middle panel shows the GPR-estimated fractional errors which give an estimate of the maximum true GPR-IMR residual.

Figure 9 is analogous to the mismatch plots shown in §III A and §III B, but now in two dimensions. The black circles denote the IMRPhenomD training points, and the color map shows the mismatch between the IMRPhenomD waveform and the GPR mean waveform at each point. The mismatches calculated here are all at or below 4.3×10^{-5}.

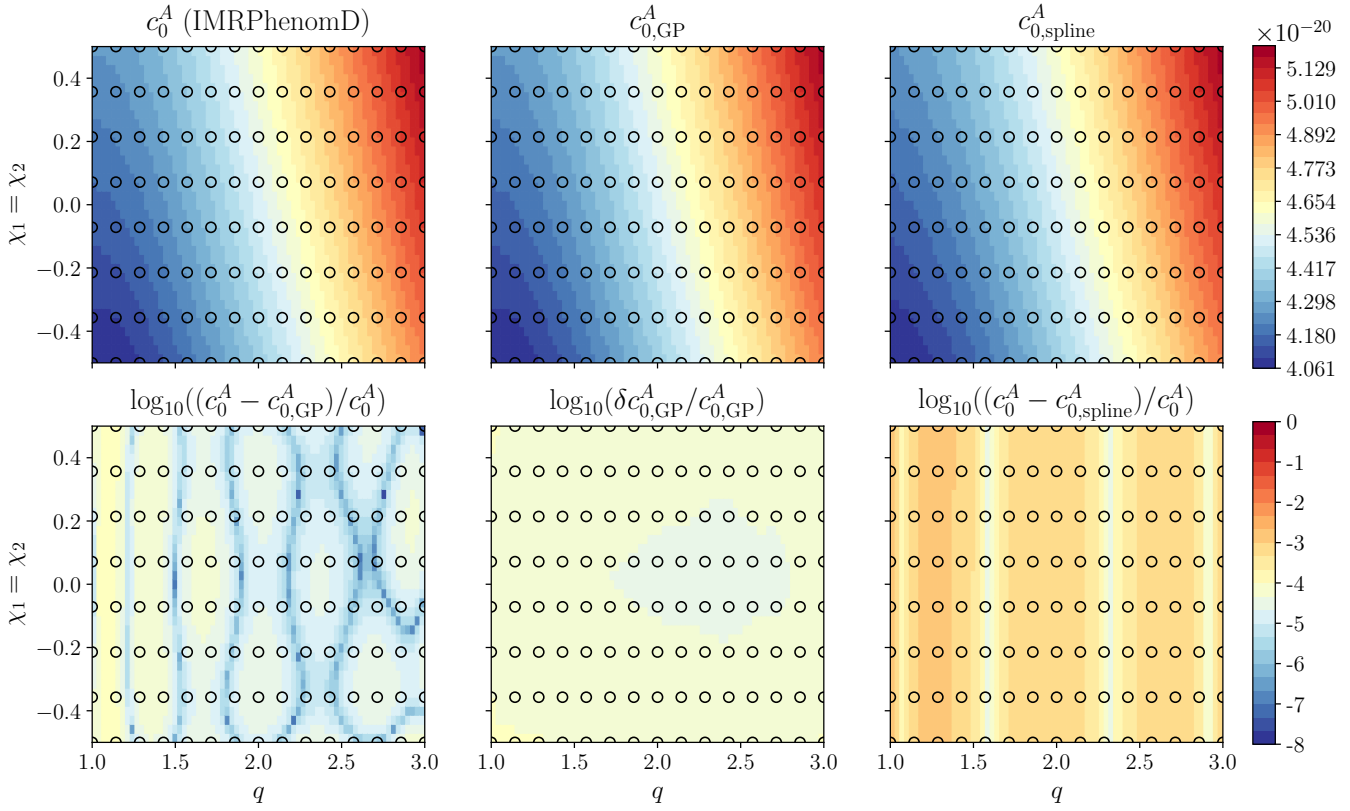

FIG. 8. First amplitude coefficient c_0^A as a function of q and equal-and-aligned spin $\chi_1 = \chi_2$. Black circles show the training point locations. *Top left*: c_0^A from the accurate model IMRPhenomD from which the training points are generated. *Top middle*: The GPR mean interpolation of c_0^A. *Top right*: The B-spline interpolation of c_0^A. *Bottom left*: The log of the fractional residual of c_0^A between IMRPhenomD and the GPR mean. *Bottom middle*: the log of the fractional 1σ uncertainty on c_0^A from the GPR. *Bottom right*: The log of the fractional residual of c_0^A between IMRPhenomD and the B-spline.

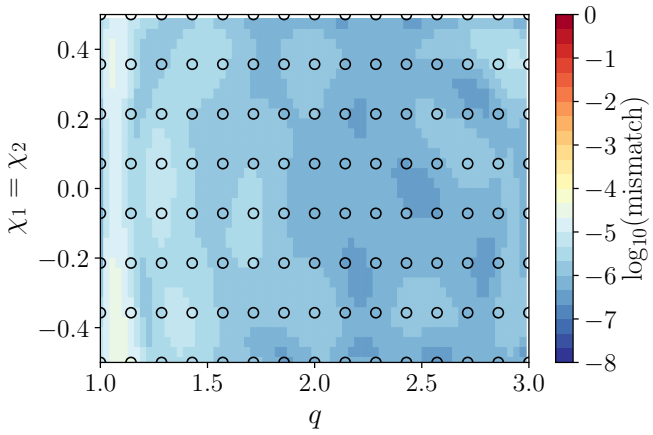

FIG. 9. Mismatch between IMRPhenomD waveforms and GPR mean waveforms with a regularly-gridded training set. The black circles show the locations of training waveforms from IMRPhenomD used to train the GPR. There are $15 \times 8 = 120$ training points on this grid, and the maximum mismatch in the region is 4.3×10^{-3}.

IV. WHERE SHOULD WE RUN THE NEXT NUMERICAL RELATIVITY SIMULATION?

Two natural questions arise from considering a GP model: 1) what is the error level of the GPR model for different GW source parameters, and 2) given a pre-existing set of training waveforms, where is the "optimal" placement of an additional training waveform? Thus far, we have evaluated the accuracy of our GPR models by computing the mismatch between the GPR model and the IMRPhenomD, but in practice such comparisons will not be possible, since the "true waveform" (i.e. NR simulation results) will be unknown everywhere other than at the existing simulation points. As such, the mismatch between the GPR mean and the true waveform cannot be used to determine the GPR error level nor can it be used as a parameter of the optimization function which selects new NR simulation parameters. Instead, we propose using the GPR posterior uncertainties to guide the overall error estimation and training set optimization. Here we present a simple metric for estimating GPR waveform errors and for choosing where in parameter space to add a new training waveform. The basic idea is to estimate the mismatch between a GPR and NR waveform with the same param-

eters based on the spread of GPR samples. Specifically, we estimate the GPR-IMR mismatch by computing the largest mismatch between M GPR samples and the GPR mean. New accurate waveforms can be added where this estimated mismatch is large, analogously to the greedy algorithm 8.1 in [26]. We summarize this training point placement strategy in Algorithm 1.

Algorithm 1 greedy training point placement

$\{\vec{\lambda}_j\} \leftarrow n_{\rm train}$ initial parameter values, $j \in [1, n_{\rm train}]$
$\{\vec{\lambda}^*_k\} \leftarrow$ fine interpolation grid, $k \in [1, n_{\rm interp}]$
loop
  Calculate regularized coefficients $\tilde{c}_i(\{\vec{\lambda}_j\})$
  $\tilde{c}_i(\vec{\lambda}^*_k) \sim \mathrm{GPR}(\tilde{c}_i(\{\vec{\lambda}_j\}))$
  for $k \in [1, n_{\rm interp}]$ **do**
    $m \leftarrow 0$, $O_k \leftarrow 0$
    for $m \in [1, M]$ **do**
      $O \leftarrow \texttt{mismatch}(h^{\rm mean}_{\rm GPR}(\vec{\lambda}^*_k), h^{\rm sample}_{\rm GPR}(\vec{\lambda}^*_k))$
      $O_k \leftarrow \max(O \cup O_k)$
  $\vec{\lambda}_{n_{\rm train}+1} \leftarrow \vec{\lambda}^*[\mathrm{argmax}_k(O_k)]$
  $\{\vec{\lambda}_j\} \leftarrow \{\vec{\lambda}_j\} \cup \{\vec{\lambda}_{n_{\rm train}+1}\}$
  $n_{\rm train} \leftarrow n_{\rm train} + 1$

First, a few training waveforms, preferably on the boundaries of the parameter space P, are used to seed a GPR model of the waveforms in P. GPR waveforms are interpolated on a fine grid in P, and at each grid point the maximum mismatch between the GPR waveform mean and M GPR waveform samples is recorded (hereafter called O_k for the k-th interpolation point). O_k at each interpolation grid point is used as a proxy for the true mismatch between the GPR mean and NR in order to determine where to generate a new simulation. By adding a new simulation to the training set at the point with largest O_k, the greedy algorithm attempts to minimize error in locations in parameter space with the largest estimated error.

As a proof of concept, we apply a computationally simplified variant of the greedy algorithm to GPR interpolations in the same q–χ space as in §III C. Rather than determining O_k at every point on a dense interpolation grid, we instead partition the space into 100 equally-sized, rectangular domains and determine O_k at a random point in each domain. These 100 O_k values are then used to determine training point placement. Additionally, at each iteration we add a training waveform at the ten points with highest O_k of the 100 computed, rather than just adding one at a time. In the example we show here, we seed the GPR model with 12 initial waveforms: one on each corner of the parameter space and two equally-spaced training waveforms on each edge. We perform 11 iterations (i.e. 122 total training points) of the greedy algorithm, and compute the mismatch between the IMR model and the GPR mean just as in §III C. Figure 10 shows these mismatch values over the parameter space. Comparing Figure 10 to Figure 9, which have 122 and 120 training points, respectively, we see that the

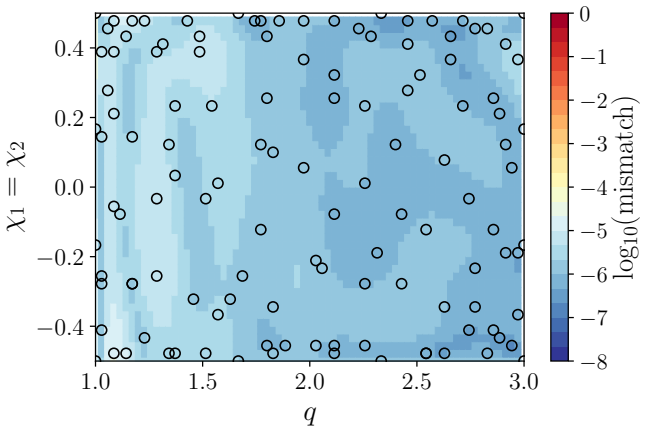

FIG. 10. Mismatch between IMRPhenomD waveforms and GPR mean waveforms with an iteratively-built training set. The training set shown here was seeded with 12 initial IMRPhenomD waveforms and 10 points were added at each iteration based on 100 samples of O_k across the space. The black circles show the locations of training waveforms from IMRPhenomD used to train the GPR. In this example, 10 iterations were performed, yielding $10 \times 10 + 12 = 122$ training points and a maximum mismatch in the region of 3.4×10^{-5}.

iterative method results in lower mismatches than the regular grid across the parameter space. Additionally, the maximum mismatch over the parameter space in the iterative case is 9.3×10^{-5}, which is an order of magnitude lower than the maximum mismatch over the regular grid of 1.4×10^{-3}.

To address the question of whether we can estimate the true mismatches using O_k, we compare the maximum of the 100 O_k values and the maximum GPR mean-IMR mismatch over the parameter space at each iteration in Figure 11. Using a greedy grid, the maximum O_k value (blue) tracks the maximum GPR mean-IMR mismatch (orange) to within an order of magnitude, though the maximum O_k would likely be higher if more than 100 O_k samples were taken. On the last iteration, we calculate O_k on the fine interpolation grid used in §III C rather than just sampling 100 O_k values. This yields the O_k map shown in Figure 12. By construction, O_k is relatively constant over the parameter space. Additionally, the maximum O_k on this finer grid is 9.3×10^{-5}, which bounds the maximum GPR-IMR mismatch of 3.4×10^{-5}, further suggesting that the maximum O_k value can be used to estimate the maximum true error level of the GPR mean. As such, O_k can indicate when a sufficient number of training waveforms have been used.

It is worth noting that the maximum mismatch of 4.3×10^{-5} on the 120-point regular grid from §III C is comparable to the maximum mismatch of 3.4×10^{-5} using the 122-point greedy grid. However, this fact does not indicate that regular training grids are as effective as greedy grids: On the regular grid, the number of training points in the q-direction (15) and in the χ-direction

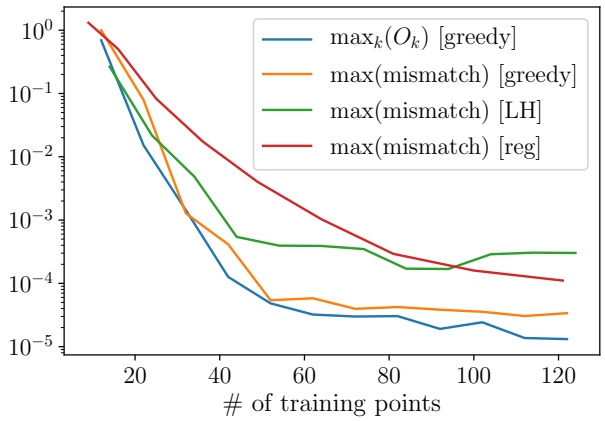

FIG. 11. The maximum O_k value using the greedy algorithm and maximum mismatch between the GPR mean and IMRPhenomD for different training point schemes and numbers of training points. For the greedy training point placement, twelve training waveforms on the boundaries of the space from IMRPhenomD seed the GPR model at the first iteration. At each subsequent iteration, the ten points with the highest O_k (out of 100 points tested) decide the locations for new training waveforms. The maximum O_k with greedy placement is shown in blue, and maximum mismatches for greedy, Latin-hypercube, and square grids are shown in orange, green, and red, respectively.

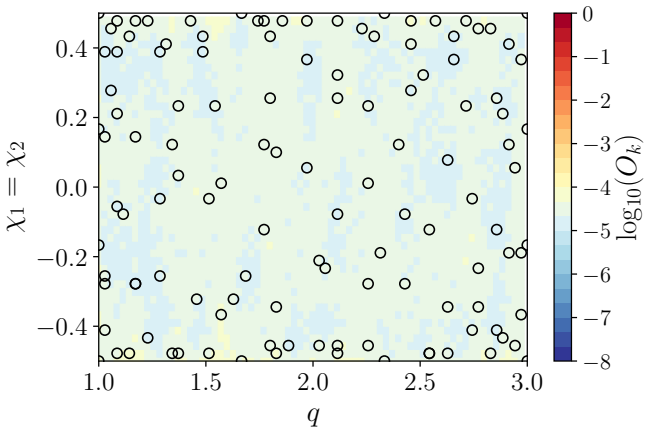

FIG. 12. Maximum GPR sample-mean mismatch O_k over 20 samples calculated at each point on the fine interpolation grid based on the same iteratively-built training set shown in Figure 10. The black circles show the locations of training waveforms from IMRPhenomD used to train the GPR. The maximum O_k in the region is 9.3×10^{-5}.

(8) were tuned to achieve low mismatches. In practice though, such tuning will not be possible since (a) the true GPR-mean error will not be known at points without simulations, and (b) building different regular grids for tuning would use significant simulation resources.

To compare the greedy algorithm to other training point placement schemes, Figure 11 also shows the maximum mismatch over the q-χ space between GPR and IMR as a function of the number of training points for a Latin hypercube (LH) grid (green) and a regular, square grid (red). For the LH case, a training point is placed at each corner of the space, and then the space is LH sampled with multiples of 10 additional training points. In other words, for each trial, training points are put on the corners of the space, and a new hypercube with $10 \times n$ partitions per axis ($n \in [1, 12]$) is constructed and randomly populated with training points under the constraint that there is exactly one training point in each row and column of the hypercube. For the square grid case, an equally-spaced $n \times n$ training grid spanning the space of interest is created for $n \in [3, 11]$. Examining Figure 11, the greedy algorithm is able to achieve mismatches considerably below the Latin hypercube or square grid mismatches for the same numbers of training points, suggesting that the greedy algorithm is the best simulation placement strategy when the training grid cannot be tuned with trial and error.

From a theoretical standpoint, it is not surprising that the greedy grid tends to be more accurate than these other "pseudo-uniform" sampling techniques. To see this, consider the conditional covariance $K_{\rm cond} = K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)$ from Equation 5. Note that as the elements of $K(X_*, X)$ get small, the second term diminishes, making the whole expression approach the prior covariance $K(X_*, X_*)$. Assuming the training set is relatively uniform over the input space, points on the edges of the space have fewer nearby training points and hence result in smaller elements of $K(X_*, X)$. In the center of the space, there are many nearby training points, so $K(X_*, X)$ tends to have larger elements, which decreases the elements of the conditional covariance. In effect, the GPR model has higher uncertainties near the boundaries when the training points are uniformly spread out. Neither LH sampling nor square grids take into account that the GPR uncertainties are highest near the edges. On the other hand, the greedy algorithm accounts for the GPR uncertainties and preferentially puts training points near the boundary, as can be seen in Figures 10 and 12.

One counter-argument to using O_k to measure the error is that there could be sharp features in the coefficient functions which are not sufficiently sampled by the training points and hence are poorly interpolated. This could indeed be true in some cases, but it is an issue that applies to any interpolation scheme. A benefit of using GPR is that if sharp features exist, some of which are sampled by the training points, the hyperparameter optimization will select shorter length scales and larger covariances and hence increase the overall coefficient uncertainty across the space. Additionally, the GPR conditional distributions are Gaussian, meaning that large excursions from the mean are not ruled out — they are just less likely.

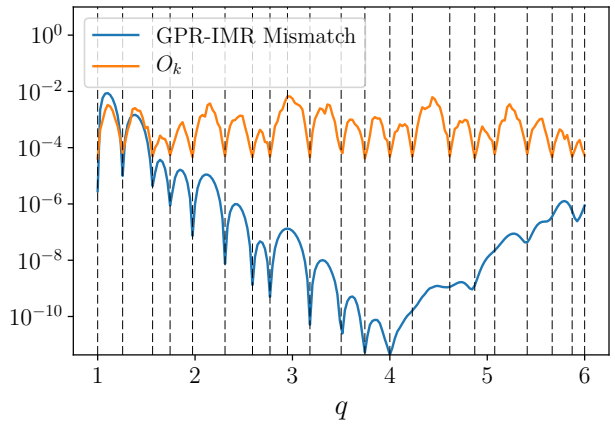

FIG. 13. Mismatch between the GPR mean using a Matern 5/2 kernel and IMRPhenomD (blue), and the mismatch estimated using O_k based on 20 GPR samples at each interpolation point (orange).

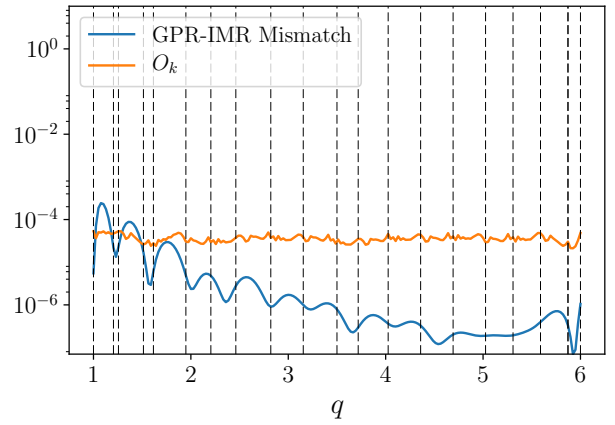

FIG. 14. Mismatch between the GPR mean using a squared-exponential kernel and IMRPhenomD (blue), and the mismatch estimated using O_k based on 20 GPR samples at each interpolation point (orange).

V. DISCUSSION

In §III we presented three examples of GPR-based ROM models trained on a subset of simulated waveforms. These example models produced accurate mean waveforms and quantified uncertainties across the parameter spaces of interest. §IV showed that further improvements in speed and accuracy can be made to GPR-based models through use of the greedy algorithm. Although we have made specific choices in our implementation, it is to be emphasized that our method is completely general. For example, different ROM or surrogate models could be used rather than the SVD-based ROM we employ, hyperparameters could instead be treated as nuisance parameters and marginalized over, the GP kernel could be changed, more sophisticated coefficient regularizations could be applied, and the greedy algorithm could be modified to incorporate other constraints. We discuss a few of these possibilities here. First, let us consider using the greedy algorithm with O_k as the metric for placing new simulations. It is worth emphasizing that greedy training point placement with O_k does not strictly result in the smallest possible training set for a desired error level. Rather, the greedy algorithm attempts to flatten the error across the parameter space by adding training points where the error is estimated to be highest. A principal limitation to just using O_k to guide simulations is that it does not encode other constraints or priorities. Two possible modifications to our greedy strategy include weighting O_k by the expected simulation cost at certain parameter values and weighting by a prior on the source population parameters. Future work will investigate these possibilities. Next we discuss the kernel functions. The kernel functional form encodes our assumptions about the smoothness and fluctuations of a GP. Another common kernel choice, other than the squared-exponential kernel adopted above, is the Matern 5/2 covariance:

$$\mathbf{k}_{\nu=5/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right), \quad (14)$$

The squared-exponential covariance constrains the GP to be infinitely mean-square differentiable, imposing a strong smoothness condition on the interpolations. The Matern 5/2 kernel is less restrictive in that it only demands the process be twice mean-square differentiable. To illustrate the effects of different kernel choices, we compare with the squared-exponential covariance. For both the squared-exponential and the Matern 5/2 kernels, we apply Algorithm 1 to build a training set to interpolate waveforms for $q \in [1, 6]$ as in §III A. In each case, we begin with three training waveforms to seed the algorithm: one waveform on each end of the space and one in the center. At each iteration, O_k is calculated on the fine interpolation grid from §III A and a new point is placed at the point of highest O_k.

Figure 13 shows the real GPR-IMR mismatch and the mismatch estimated from O_k using the Matern 5/2 kernel, and Figure 14 shows the same for a squared-exponential kernel. In both cases, the maximum O_k over the space bounds the real mismatch, except near $q = 1$ where the coefficient functions vary faster than elsewhere in the space (see Figures 2 and 3). The squared-exponential kernel is able to keep the error relatively constant over the space. Additionally, the maximum mismatch using the squared exponential is about two orders of magnitude lower than when using the Matern 5/2 kernel. On the other hand, use of the Matern kernel results in significantly lower error on the interior of the space. Further study of kernel effects will be required as new portions of parameter space are explored with the GPR model. In particular, higher dimensional GPR models may require more complex kernels, which can be constructed by summing or multiplying pre-existing kernel

functions. Additionally, kernels with compact support should be considered, as they can allow faster GP evaluation and enhanced computational stability.

We now shift our focus to the execution time of generating GPR waveforms. The examples shown here were designed to run in less than one day on one computing node with 16 cores, but future work would make use of more cores, allowing, for example, a larger ROM basis, or interpolation and O_k calculation on a finer grid. To illustrate the scaling of required time and resources with the number of GP training points, we perform a GPR on one coefficient in the q-χ space for different numbers of training values and evaluation points. The numbers of training and evaluation points determine the sizes of the matrices that must be multiplied in Equation 5 and hence the execution time. It is worth noting that $K(X,X)^{-1}$ or $K(X,X)^{-1}\mathbf{f}$ with optimized hyperparamters can be precalculated, allowing the most computationally intensive step in the GPR-building to be done just once ahead of time. We assume here that the optimization and matrix inversion steps have already been done and simply look at the evaluation time of a GP.

The timing results are shown in Figure 15, which plots the conditional GP mean and covariance evaluation time per coefficient per interpolated point at $n_{\rm interp}$ points as a function of the number of training points. That is, we evaluate the GP mean and covariance at $n_{\rm interp}$ points and divide the total time by $n_{\rm interp}$ to show the time per interpolated coefficient value. In principle, the total evaluation time should scale directly with the number of interpolated points, but Figure 15 indicates that interpolating more points at once gives an overall speedup. This is due to the overhead in constructing the training-test covariance matrices in the `scikit-learn` implementation of the GPR. This overhead is further evidenced by the fact that the $n_{\rm interp} = 100$ and $n_{\rm interp} = 1000$ curves in Figure 15 converge. Future work will consider alternate GPR implementations to mitigate such overhead, since typically only single waveform evaluations are required.

Note that to build a full GPR waveform, a GP must be evaluated for each of the ~ 100 phase and amplitude coefficients. In the case that the overhead cannot be bypassed ($n_{\rm interp} = 1$, blue curve), it would take ~ 1 minute per waveform evaluation, assuming 100 coefficients and 1000 training points. If the overhead can be entirely removed ($n_{\rm interp} = 1000$, red curve), each waveform evaluation would instead take ~ 200ms. This can be compared to the evaluation time for the spline-based ROM in P14 of $\sim$ 1 ms depending on the system's total mass (see Figure 1 of P14). Further speedups to our model could be achieved by lowering the number of ROM coefficients, decreasing the number of training points, or evaluating the coefficients in parallel. To decrease the number of training points, domain decomposition could be used as suggested in §III C. If domains of ~ 100 training points were used, the GP evaluation times would fall by over an order of magnitude (evaluation time goes as $n^2_{\rm train}$).

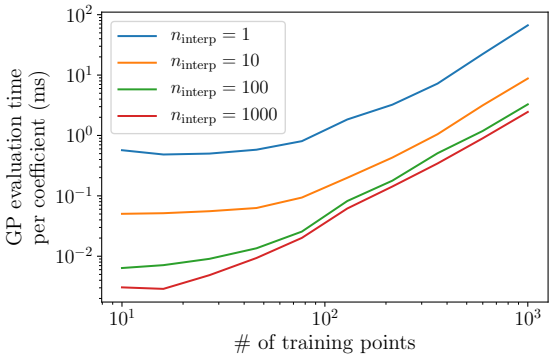

FIG. 15. Time to evaluate (mean and variance) one coefficient per interpolation point in two dimensions as a function of the number of training points. Times shows are from GP evaluations on a 2.6GHz Intel E5-2670 CPU.

Another possibility is to apply the subset of regressors method, which effectively considers a only a subset of rows of $K(X,X)^{-1}\mathbf{f}$ in Equation 5. This scheme reduces the size of the matrices multiplied when calculating the GP conditional mean and covariance (see §8.3.1 in [26]).

As mentioned before, the most expensive step of building a GPR model is in training, where the hyperparameters must be optimized and $K(X,X)$ must be inverted. The matrix inversion computation time scales as $O(n^3_{\rm train})$, since $K(X,X)$ is an $n_{\rm train} \times n_{\rm train}$ matrix (see e.g. [26]). Given the timescale for generating an NR waveform, we would not expect more than $O(10,000)$ NR waveforms in the near future, so the matrix inversion step should not be prohibitively expensive, especially since the kernel for each coefficient can be handled in parallel, and a $O(1000)$ training set can be optimized on a personal computer in minutes and only needs to be done once. Also, as mentioned earlier, smaller domains with fewer training waveforms can be handled with separate GPR models or a subset of regressors can be used, mitigating the need for the inversion of very large covariance matrices.

VI. CONCLUSIONS

We have demonstrated that Gaussian process regression can be used to construct reduced-order-model waveforms with uncertainties using only a few existing simulations, and that these uncertainties can guide the choice of future simulations. The overall motivation for such GPR models is that GPR uncertainties can be propagated to the parameter estimation of compact binary coalescences in order to remove bias in estimates due to systematic waveform errors. Figures 4 and 5 show example amplitude and phase functions with uncertainties from the GPR model.

This work has also shown that GPR can model waveforms accurately over a parameter space of interest. Fig-

ures 9 and 10 show that with a sufficient number of training simulations, the error level of the GP model can be reduced to levels adequate for parameter estimation with LIGO data, especially if Algorithm 1 is used to construct the training set. Such greedy algorithms will be a particularly useful tool for efficiently choosing the parameters of new simulations in the nominal 7-d parameter space of interest to LIGO. Since the methods presented here are general, they could in principle be applied to other scenarios such as eccentric-orbit Laser Interferometer Space Antenna sources or neutron-star binaries with tidal deformability parameters.

Another finding of this work is that the error level of the GPR model can be estimated from the GPR itself rather than through cross-validation. Figure 11, which compares the maximum true GPR error to the maximum estimated error, demonstrates that the GPR uncertainties can alone be used to estimate the maximum error level of the GPR model. This allows one to know when a GPR model has reached a desired error level and does not require further simulations in the parameter region of interest.

Finally, we describe future directions for this work. In the immediate future, GPR models will be applied to three- or higher-dimensional parameter spaces to test the robustness of these models as the complexity grows. In particular, we will see if the ROM coefficient functions can generally be treated as being uncorrelated. Additionally, a wider range of kernel functions will be explored than what has been presented here. In the longer term, GPR training sets will be built directly from NR simulations, rather than utilizing a stand-in approximant. Apart from doing PE studies with an NR-based GPR model, the model could also be used to validate or study families of approximants. In sum, GPR models present an exciting frontier for NR-simulation-driven models of GW waveforms.

VII. ACKNOWLEDGEMENTS

DEH acknowledges valuable discussions with Salman Habib, Katrin Heitmann, David Higdon, and Michael Stein. ZD would like to thank Richard Chen for consultation on GPR. ZD is supported by NSF Graduate Research Fellowship grant DGE-1144082. ZD, BF, and DEH were partially supported by NSF CAREER grant PHY-1151836 and NSF grant PHYS-1708081. They were also supported in part by the Kavli Institute for Cosmological Physics at the University of Chicago through NSF grant PHY-1125897 and an endowment from the Kavli Foundation. This work was completed in part with resources provided by the University of Chicago Research Computing Center.

[1] B. P. Abbott, R. Abbott, T. D. Abbott, *et al.*, Physical Review Letters **116**, 061102 (2016), arXiv:1602.03837 [gr-qc].

[2] B. P. Abbott, R. Abbott, T. D. Abbott, *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Phys. Rev. Lett. **116**, 241103 (2016).

[3] B. P. Abbott, R. Abbott, T. D. Abbott, *et al.* (LIGO Scientific and Virgo Collaboration), Phys. Rev. Lett. **118**, 221101 (2017).

[4] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al., Physical Review Letters **116**, 241102 (2016), arXiv:1602.03840 [gr-qc].

[5] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al., Astrophysical Journal **818**, L22 (2016), arXiv:1602.03846 [astro-ph.HE].

[6] G. Lovelace, C. O. Lousto, J. Healy, M. A. Scheel, A. Garcia, R. O'Shaughnessy, M. Boyle, M. Campanelli, D. A. Hemberger, L. E. Kidder, H. P. Pfeiffer, B. Szilágyi, S. A. Teukolsky, and Y. Zlochower, Classical and Quantum Gravity **33**, 244002 (2016), arXiv:1607.05377 [gr-qc].

[7] B. P. Abbott *et al.* (Virgo, LIGO Scientific), Phys. Rev. **D94**, 064035 (2016), arXiv:1606.01262 [gr-qc].

[8] S. Husa, S. Khan, M. Hannam, M. Pürrer, F. Ohme, X. Jiménez Forteza, and A. Bohé, Phys. Rev. **D93**, 044006 (2016), arXiv:1508.07250 [gr-qc].

[9] S. Khan, S. Husa, M. Hannam, F. Ohme, M. Pürrer, X. Jiménez Forteza, and A. Bohé, Phys. Rev. **D93**, 044007 (2016), arXiv:1508.07253 [gr-qc].

[10] A. Taracchini *et al.*, Phys. Rev. **D89**, 061502 (2014), arXiv:1311.2544 [gr-qc].

[11] S. Babak, A. Taracchini, and A. Buonanno, Phys. Rev. **D95**, 024010 (2017), arXiv:1607.05661 [gr-qc].

[12] M. Hannam, P. Schmidt, A. Bohé, L. Haegel, S. Husa, F. Ohme, G. Pratten, and M. Pürrer, Physical Review Letters **113**, 151101 (2014), arXiv:1308.3271 [gr-qc].

[13] A. Bohé, L. Shao, A. Taracchini, A. Buonanno, S. Babak, I. W. Harry, I. Hinder, S. Ossokine, M. Pürrer, V. Raymond, T. Chu, H. Fong, P. Kumar, H. P. Pfeiffer, M. Boyle, D. A. Hemberger, L. E. Kidder, G. Lovelace, M. A. Scheel, and B. Szilágyi, Phys. Rev. D **95**, 044028 (2017), arXiv:1611.03703 [gr-qc].

[14] B. P. Abbott *et al.* (Virgo, LIGO Scientific), Class. Quant. Grav. **34**, 104002 (2017), arXiv:1611.07531 [gr-qc].

[15] Y. Pan, A. Buonanno, A. Taracchini, L. E. Kidder, A. H. Mroué, H. P. Pfeiffer, M. A. Scheel, and B. Szilágyi, Physical Review D **89**, 084006 (2014), arXiv:1307.6232 [gr-qc].

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

[17] J. Blackman, S. E. Field, M. A. Scheel, C. R. Galley, D. A. Hemberger, P. Schmidt, and R. Smith, ArXiv e-prints 1701.00550 (2017), arXiv:1701.00550 [gr-qc].

[18] R. O'Shaughnessy, J. Blackman, and S. E. Field, ArXiv e-prints (2017), arXiv:1701.01137 [gr-qc].

[19] M. Pürrer, Classical and Quantum Gravity **31**, 195010 (2014), arXiv:1402.4146 [gr-qc].

[20] S. Khan, S. Husa, M. Hannam, F. Ohme, M. Pürrer, X. J. Forteza, and A. Bohé, Phys. Rev. D **93**, 044007 (2016).

[21] C. J. Moore and J. R. Gair, Physical Review Letters **113**, 251101 (2014), arXiv:1412.3657 [gr-qc].

[22] C. J. Moore, C. P. L. Berry, A. J. K. Chua, and J. R. Gair, Phys. Rev. D **93**, 064001 (2016), arXiv:1509.04066 [gr-qc].

[23] S. E. Field, C. R. Galley, J. S. Hesthaven, J. Kaye, and M. Tiglio, Physical Review X **4**, 031006 (2014), arXiv:1308.3565 [gr-qc].

[24] We interpolate fixed-chirp-mass waveforms, whereas P14 interpolated fixed-total-mass waveforms. This results in interpolations which are over the same frequency range rather than interpolations over the same geometric frequency range.

[25] We assume the prior mean is zero, as we only interpolate functions which are de-meaned, i.e. which have their mean subtracted off.

[26] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (The MIT Press, 2005).

[27] See §4.1.1 in Rasmussen and Williams 26 for the definition of mean-square differentiability.

[28] I. Andrianakis and P. G. Challenor, Computational Statistics & Data Analysis **56**, 4215 (2012).

[29] The correlations between c_i's are outside the scope of this paper, and will be considered in future work.

[30] This noise curve can be found at https://dcc.ligo.org/LIGO-G1501223/public.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).

[32] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," (2001–).

[33] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, ACM Trans. Math. Softw. **23**, 550 (1997).

[34] The spins of the inspiraling objects are assumed to be equal and aligned with the orbital angular momentum.

[35] The computational time to build a GPR given hyperparameters and N training waveforms goes as $\sim O(N^3)$. Additional cost is incurred by the hyperposterior maximization.

[36] M. Pürrer, Phys. Rev. D **93**, 064041 (2016), arXiv:1512.02248 [gr-qc].