

Framework for optimal placement and emulation of stellar evolution models

Trevor Gravely,^{1,2} Marko Ristić^{2,3,*} and Brandon Barker^{2,4}

¹*Department of Physics and Astronomy, Vanderbilt University, Nashville, Tennessee 37212, USA*

²*Center for Theoretical Astrophysics, Los Alamos National Laboratory,
Los Alamos, New Mexico 87545, USA*

³*Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA*

⁴*Computational Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA*



(Received 20 May 2026; accepted 11 September 2026; published 23 October 2026)

Generating dense grids of detailed stellar evolution simulations is computationally prohibitive, as resolving progressively complex nucleosynthesis chains and dynamical timescales incurs a rapidly increasing computational cost that scales with dimensionality and resolution. Existing simulation grids are sparse, heterogeneous, and often cover disparate regions of parameter space, limiting (direct) systematic comparison and broader generalization to downstream science analyses. We present active simulation tuning and regression for astrophysical libraries (ASTRAL), a publicly available framework for optimal placement and emulation of stellar evolution models. The ASTRAL framework builds surrogate models that provide a continuous representation of the parameter space which is populated by sparse simulation grids. Furthermore, ASTRAL interpolates across the simulation parameter space (e.g. zero-age-main-sequence mass) to predict stellar properties of stars with associated uncertainty estimates for systems not explicitly simulated. We show that ASTRAL significantly outperforms uniform placement in constructing emulation-ready model libraries that capture both global and local variability in the output features. We showcase the performance of our framework in the context of a simple, one-dimensional MESA model. Finally, we examine how our methodology can inform downstream applications by analyzing our emulator's predictive confidence when applied to supernova simulations. We emphasize that, while developed for stellar evolution, the ASTRAL framework is directly extensible to other simulation-driven domains where model evaluations are expensive and parameter spaces are high dimensional.

DOI: [10.1103/spxh-q8cq](https://doi.org/10.1103/spxh-q8cq)

I. INTRODUCTION

Stellar evolution modeling underpins our understanding of stars across cosmic time, shaping phenomena ranging from star formation to the final presupernova structure that gives rise to many of the transient events observed in the Universe [1–6]. Uncertainties in stellar evolution modeling propagate directly into predictions of supernova outcomes, compact object populations, and other downstream observables, highlighting the need for deliberate uncertainty mitigation. One approach to systematic uncertainty reduction favors understanding populations of massive stars as opposed to focusing on highly specific, idealized case studies. Understanding the distribution of phenomena dictated by populations of massive stars requires a

collection of stellar evolution models that are equally as diverse as those that may be realized in nature. In practice, such a collection requires sampling the full diversity of stellar evolution outcomes across parameter space using dense, expansive, multidimensional grids of massive stars evolved out to late times, spanning zero-age-main-sequence (ZAMS) mass, metallicity, rotation rates, and more complex properties like binarity [7–9]. Even in simplified one-dimensional models, stellar evolution simulations, e.g., KEPLER and MESA [10–16] are computationally expensive and exhibit numerical instability, thus inhibiting long term evolution and complete grid construction. As a star evolves beyond core helium burning, the nuclear reactions responsible for core burning become increasingly stiff, requiring more demanding numerical methods with increasingly small time steps. The increasing number and complexity of nuclear reactions and the added complications introduced by neutrino losses act as significant bottlenecks in continuing the evolution of the star to late times, see discussion in Ref. [17]. These instabilities and computational costs inhibit the construction of dense, detailed stellar evolution grids which are suitable for

*Contact author: mristic@lanl.gov

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

astrophysical inference (although prior efforts have tackled the issue, e.g. [18]), necessitating alternative techniques for broad parameter space representation and grid construction.

Standard fitting formulas have historically been applied to understanding stellar and binary systems [19,20], with recent studies also leveraging now-established machine learning approaches, in particular Gaussian processes [21,22]. Gaussian processes (GPs) provide a powerful framework for emulating expensive simulations and quantifying predictive uncertainty [23] and have been implemented across many astrophysical applications including gravitational waveform surrogate modeling [24], kilonova and supernova light curve emulation [25–27], and galaxy profile fitting [28] (see Ref. [29] for a more exhaustive list of examples). Previous studies in the literature have explored scalable stellar evolution parameter forecasting using both deep learning [30] and Gaussian processes [31]. In a similar vein, population synthesis studies apply machine learning methods to interpolate and emulate stellar profiles across binary evolution model grids [32,33].

Recent work has further leveraged GP predictive uncertainties (and other machine learning tools, e.g. [34]) to guide simulation placement, enabling adaptive sampling strategies that prioritize regions of high uncertainty to achieve global uncertainty reduction [26,35]. When combined, these strategies establish a powerful emulation setup: simulation optimization ensures that each placed simulation provides maximal information, while the GP-derived placement metric all but ensures convergence and uncertainty minimization for the emulation step. This synergy is particularly appealing in contexts like complex stellar evolution models, where computational expense and the need for a low-cost representation of the parameter space beget emulation.

In this work, we present ASTRAL, a new simulation placement and emulation framework that leverages Gaussian process regression to interpolate and predict stellar evolution model outputs. ASTRAL uses GP regression to interpolate an existing grid of stellar evolution models, and produces predictions of key stellar evolution output parameters as a function of standard stellar inputs, in this case focused solely on the zero-age main sequence mass. Additionally, we show that ASTRAL is capable of optimizing the placement of computationally expensive simulations while significantly outperforming standard placement strategies.

This paper is organized as follows. In Sec. II we describe our methodology, specifically our simulation assumptions, a brief overview of the GP regression technique, and our application of GPs to uncertainty-informed simulation placement. In Sec. III we present our results which focus on GP interpolation of stellar evolution outputs, simulation grid building through optimized parameter placement, and the propagation of uncertainty minimization between stellar evolution and supernova simulation outputs. We

present an overarching discussion of our results in Sec. IV and end with our conclusions in Sec. V.

II. METHODOLOGY

At a high level, our methodology proceeds as follows. We establish a reference for interpolation and regression using the preexisting dataset from Ref. [36] and a standard PYTHON library by generating scalar GP emulators of the stellar evolution output parameters as a function of zero-age main sequence mass (M_{ZAMS}). Expanding on this reference, we present a metric which guides the reduction of GP prediction variance through an optimized simulation placement approach. We evaluate the effectiveness of this approach compared to standard (uniform) simulation placement using a simple MESA stellar evolution model. Finally, we revisit the initial stellar evolution dataset [36] in the context of the follow-up supernova simulations informed by that grid as presented in Ref. [37]. We evaluate our simulation placement metric’s effectiveness on reducing GP prediction variance of supernova simulation outputs when trained on upstream stellar evolution simulations.

A. Stellar evolution and supernova simulations

We use the dataset from Ref. [36] generated using the KEPLER code [38], the physics of which are discussed in e.g., Ref. [39]. We will refer to this dataset as the “KEPLER grid” throughout the rest of this text. In this study, we examine only the models for which $9 \leq M_{\text{ZAMS}} \leq 30$ to focus on stellar evolution models with retained hydrogen envelopes (for the supernova connection described at the end of this section). This amounts to 187 stellar models. All stars are nonrotating with initial solar composition and evolve from the zero-age main sequence until iron-core collapse. Each simulation yields numerous scalar output parameters as a function of M_{ZAMS} , several of which we consider: the presupernova mass, radius, temperature, and luminosity; the central entropy, electron fraction, and density; compactness parameters ξ_M [40] defined via

$$\xi_M = \frac{M/M_\odot}{R(M)/1000 \text{ km}}, \quad (1)$$

and the mass of the He/CO/Si/Fe cores and their corresponding radii. Additional details about the KEPLER stellar evolution simulation data we use can be found in Sec. 2.1 of Ref. [36].

The KEPLER models are used as inputs for subsequent simulations in spherical symmetry through collapse and potential explosion using the neutrino-driven, turbulence aided method STIR [41] in the FLASH code [42–44]. STIR includes multigroup two moment “M1” neutrino radiation transport, a dense matter equation of state [45], a general relativistic effective potential, and a Reynolds averaged Navier-Stokes treatment of turbulence and convection in

spherical symmetry. These explosions are followed through shock breakout producing electromagnetic observables [37] using SNEC [46]. Thus, we obtain a consistent, corresponding grid of supernova outputs from the upstream KEPLER grid. We will refer to the supernova simulation library as the “KEPLER-FLASH grid” throughout the text.

We demonstrate the capability to guide stellar evolutionary modeling *in situ* by running stellar evolution simulations using MESA, version r24.08.1 [11–16]. We use a basic stellar evolution model to reduce the total build time for the proof-of-concept grid—specifically, we use the `12M_pre_ms_to_core_collapse` test within the MESA test suite, which evolves a solar metallicity model from the premain sequence to the onset of core collapse. We note that although we opted to use MESA due to its role as an open-source software the results presented here apply to any stellar evolutionary modeling framework and beyond.

B. Gaussian processes

Below, we present a high-level overview of GP regression. We defer readers to Ref. [23] for an exhaustive general review of GP regression and Ref. [29] for a more field-specific review of GP regression using astronomical data.

Let $\{y(x): x \in X\}$ be a stochastic process, where each $y(x)$ is a random variable and $x \in X$ is an independent variable for some domain X . A stochastic process is a Gaussian process if the distribution of any finite set of random variables in the process is jointly Gaussian

$$p(y(x_1), \dots, y(x_n)) = \frac{1}{\sqrt{2\pi|\mathbf{K}|}} \times \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{m})^T \mathbf{K}^{-1}(\mathbf{y} - \mathbf{m})\right), \quad (2)$$

where each $x_i \in X$ and $\mathbf{y} \equiv (y(x_1), \dots, y(x_n))$. Here, $\mathbf{m} \in \mathbb{R}^n$, $\mathbf{K} \in \mathbb{R}^{n \times n}$ are the mean and covariance of the GP respectively—specifying a particular GP amounts to identifying a particular choice of mappings from $(x_1, \dots, x_n) \mapsto \mathbf{m}, \mathbf{K}$.

In particular, in GP regression one assumes that the covariance matrix takes the form

$$(\mathbf{K})_{ij} = k(x_i, x_j; \boldsymbol{\phi}) + \delta_{ij}\sigma_n^2. \quad (3)$$

Here $\boldsymbol{\phi}$ is a set of model parameters that parameterize the functional form of k , the kernel function, and σ_n is a parameter characterizing the noise scale given a particular application. Equation (3) contains two contributions: first, the intrinsic observational noise, which is quantified via the term $\delta_{ij}\sigma_n^2$, only contributing along the diagonal of $\mathbf{K}$; and second, the correlations between inputs, which are quantified via the kernel function k . The choice of k is free and

often dictated by what is best suited to the problem at hand. In this work, we assume a radial basis function (RBF) kernel

$$k(x_i, x_j; \boldsymbol{\phi}) = \alpha^2 \exp\left(-\frac{|x_i - x_j|^2}{2\ell^2}\right), \quad (4)$$

where the kernel parameters $\boldsymbol{\phi} = (\alpha, \ell)$ control the overall scale of fluctuations and the correlation length, respectively. For the GP presented in this work, we find no preferential performance between the RBF and Matern 3/2 and Matern 5/2 kernels. However, GP behavior may be more sensitive to kernel choices for more complex problem setups, especially those with coupled input dimensions, and thus we encourage readers to treat the choice of kernel as a sensitive hyperparameter until proven otherwise.

Equation (2) also contains a mean $\mathbf{m} \in \mathbb{R}^n$ which one assumes to be given by

$$(\mathbf{m})_i = m(x_i, \boldsymbol{\theta}), \quad (5)$$

for some mean function m and some model parameters $\boldsymbol{\theta}$. Without loss of generality, the mean can be absorbed into the kernel by demeaning the data on which one regresses. We correspondingly set $m(x, \boldsymbol{\theta}) \equiv 0$, leaving $\{\boldsymbol{\phi}, \sigma_n\}$ as the only parameters in the model.

To fit the model to some set of data $\mathbf{y} \in \mathbb{R}^n$ corresponding to inputs $\mathbf{x} = (x_1, \dots, x_n)$, we maximize the likelihood

$$\mathcal{L}(\boldsymbol{\phi}) = p(\mathbf{y}|\boldsymbol{\phi}, \sigma_n), \quad (6)$$

where $p(\mathbf{y}|\boldsymbol{\phi}, \sigma_n)$ is given by Eq. (2) with inputs $\mathbf{x}$.

Once the GP model parameters have been fit to the data, the GP can be used to predict means and variances of the underlying stochastic process at n_* unseen locations. In particular, if $\mathbf{y}$ are existing observations corresponding to inputs $\mathbf{x}$, the predictions $\mathbf{y}_* \in \mathbb{R}^{n_*}$ at unseen inputs $\mathbf{x}_* \in X^{n_*}$ are distributed as a joint Gaussian with mean $\mathbf{f}_* \in \mathbb{R}^{n_*}$ and covariance $\mathbf{C}_* \in \mathbb{R}^{n_* \times n_*}$ given by

$$\mathbf{f}_* = \mathbf{m}_* + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{y} - \mathbf{m}), \quad (7)$$

$$\mathbf{C}_* = \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* + \sigma_n^2 \mathbf{Id}, \quad (8)$$

where $\mathbf{m}_* \equiv m(\mathbf{x}_*, \boldsymbol{\theta}) \in \mathbb{R}^{n_*}$ and

$$(\mathbf{K}_*)_{ij} = k(x_i, x_{*j}; \boldsymbol{\phi}) \in \mathbb{R}^{n \times n_*}, \quad (9)$$

$$(\mathbf{K}_{**})_{ij} = k(x_{*i}, x_{*j}; \boldsymbol{\phi}) \in \mathbb{R}^{n_* \times n_*}. \quad (10)$$

We keep $\mathbf{m}$ in Eq. (7) for completeness, but in all models considered in this work we take $m(x, \boldsymbol{\theta}) \equiv 0$ for all $x \in X$. Also note that with the joint distribution in hand, one can marginalize over observations to obtain the variance in a single predicted observation.

C. Gaussian processes applied to KEPLER

We begin by training a separate scalar GP on each of the 22 KEPLER stellar evolution outputs described in Sec. II A. While we do expect correlation between the KEPLER outputs, our goal of identifying an optimal placement metric benefits from separate predictions; we are interested in choosing a parameter that both varies smoothly as a function of M_{ZAMS} and is physically well motivated. We normalize (z -score) both the input masses M_{ZAMS} and the KEPLER output parameters (separately) for GP training. We implement Gaussian process regression using `scikit-learn` [47], specifically the `GaussianProcessRegressor` module. Our kernel matches the functional form outlined in Eq. (4) and assumes all the default settings. We optimize hyperparameters using the default `scikit-learn` Gaussian process optimizer, `fmin_l_bfgs_b`, a limited-memory BFGS algorithm.

D. Gaussian processes applied to MESA

We extend the use case of our GPs to inform simulation placement in M_{ZAMS} during the construction of uncertainty-informed grids of stellar evolution models simulated with MESA. We choose M_{Fe} , the iron core mass, as the output parameter which guides our uncertainty-informed simulation placements as it is a key parameter in determining the final remnant masses and explosion mechanisms relevant to downstream supernova simulations, displays nonmonotonicity and stochasticity, and is a core output parameter in the KEPLER grid, providing cohesion with Sec. II C. While we employ only a single placement parameter for simplicity, an uncertainty metric u can combine any number of output parameters by taking the form

$$u(M_{\text{ZAMS}}) = \sum_{i=1}^N \alpha_i \sigma_i(M_{\text{ZAMS}}), \quad (11)$$

where $\sum_i \alpha_i = 1$ for N output parameters, each with GP predictive uncertainty σ_i . The weights α_i control the contribution of each output parameter to the simulation placement process, thus allowing domain-specific tailoring for downstream applications that might be more sensitive to particular parameters. In this work, u and σ_i only depend on M_{ZAMS} , but will generally be functions of the full range of input parameters varied in the stellar evolution models.

We define the total uncertainty U of a given GP as the average thickness of the 95% confidence band over the dynamic range of the training set

$$U = \frac{1}{x_{\text{max}} - x_{\text{min}}} \int_{x_{\text{min}}}^{x_{\text{max}}} (y_u(x) - y_\ell(x)) dx, \quad (12)$$

where $y_{u,\ell}(x)$ are the 95% GP confidence bounds and $x_{\text{min}/\text{max}}$ are the smallest and largest M_{ZAMS} samples in our

grid. The baseline grid to which we compare against is a uniform grid across a fixed dynamic range of $9M_\odot \leq M_{\text{ZAMS}} \leq 30M_\odot$ with 65 equally spaced simulations. The grid building procedure is as follows:

- (1) Construct a uniform initial training grid of 17 simulations for $9M_\odot \leq M_{\text{ZAMS}} \leq 30M_\odot$.
- (2) Train the Gaussian process on the current grid iteration.
- (3) Determine M_{ZAMS}^* , the M_{ZAMS} with maximal predictive uncertainty for the output parameter(s) of interest (here M_{Fe}).
- (4) Place a new MESA simulation at M_{ZAMS}^* .
- (5) Repeat Step 2 until the grid reaches the fixed size of the baseline grid (65 simulations).

We note that the choice of 17 initial simulations is arbitrary; different choices of the initial number of simulations will yield varying results. Generally, however, more simulations in the initial grid will contribute to quicker convergence toward a desired GP variance (though performance will vary depending on the complexity of the input parameter space).

E. Gaussian processes applied to KEPLER-FLASH

In Sec. II D, we describe how GP variance influences the placement and generation of MESA simulations, assuming that additional simulations can be readily generated by the user. We slightly adapt our simulation placement metric to investigate how uncertainty-informed fixed-grid sampling of KEPLER outputs affects downstream supernova models. We note that not all KEPLER stellar evolution models result in supernovae in the STIR model; black hole formation is also possible. Therefore, we first down-select the KEPLER grid based on which models lead to explosion. We are then left with two separate simulation grids: 136 down-selected stellar evolution models (the KEPLER grid) and the 136 supernova models derived from these stellar evolution models (the KEPLER-FLASH grid). We emphasize that the KEPLER-FLASH grid of stellar-evolution-into-supernova models is *distinct from the KEPLER grid of only stellar evolution models discussed in Sec. II A*. Our goal is to examine how training GPs on the KEPLER grid affects the GP prediction variance for the KEPLER-FLASH grid parameters.

We carry out the following modified procedure:

- (1) Initialize a pseudo-uniform grid of simulations by sequentially selecting every third M_{ZAMS} model from the KEPLER grid.
- (2) Train the Gaussian process on the current grid.
- (3) From the remaining stellar evolution models in the KEPLER grid, determine M_{ZAMS}^* , the M_{ZAMS} with maximal predictive uncertainty for the output parameter(s) of interest (here M_{Fe}).
- (4) Add this model to the constructed grid.
- (5) Repeat step 2 until 35 simulations have been added to the grid.

Our initial simulation grid inherits pseudo-uniformity from the KEPLER grid, which is piecewise uniform in M_{ZAMS} . We ultimately obtain a constructed supernova grid of 45 (initial grid) + 35 (uncertainty-placed models) = 80 models out of the total 136 from the KEPLER-FLASH grid. For comparison, we also construct a baseline grid of 80 models by randomly selecting 35 models from the KEPLER-FLASH grid and adding them to the initial grid. In both cases, we are using only $\sim 60\%$ of the simulations in the full KEPLER-FLASH grid.

III. RESULTS

Here we present our results for the three separate analyses described in Sec. II. As a reminder, we first examine the GP fits to stellar evolution output parameters as a function of M_{ZAMS} as described in Sec. II C. We then analyze GP uncertainty across a library of simple MESA stellar evolution models, comparing grids constructed by optimizing over GP uncertainty versus uniform sampling, as described in Sec. II D. Finally, we compare the performance of GP predictions of supernova simulation outputs using a subset of models chosen through uncertainty minimization of stellar evolution outputs as described in Sec. II E.

A. GP fits to KEPLER simulation outputs

Throughout this section, we focus specifically on the results for the iron core mass M_{Fe} , determined by the ^{56}Fe mass fraction, for the sake of brevity. Analogous results for the other output parameters considered in our analysis can be found in Appendix A. Figure 1 displays the results of GP regression on the KEPLER grid iron core mass as a function of M_{ZAMS} . The mean squared error (MSE) is given by

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2, \quad (13)$$

with $\hat{y}_i$ the GP predictive mean for the i th sample, y_i the true value of the i th sample, and N the total number of samples. In this instance, we train on the full KEPLER dataset of $N = 187$ simulations. The MSE for the fit in Fig. 1 is 6.9×10^{-4} , indicating strong predictive performance on the training set. To assess the accuracy of the predictive uncertainties (light blue band), we conduct an uncertainty-calibration test. For a confidence level of α , we first compute the normalized errors

$$z_i = \frac{y_i - \hat{y}_i}{\sigma_i}, \quad (14)$$

where σ_i is the predictive uncertainty, and then compute the coverage $\mathcal{C}(\alpha)$,

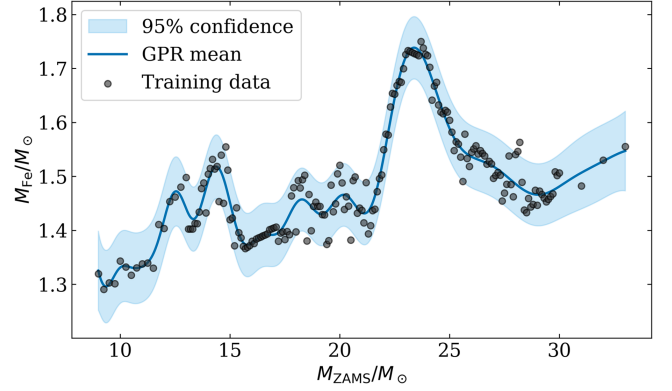


FIG. 1. Gaussian process regression fit to the KEPLER grid for the iron core mass M_{Fe} as a function of zero-age main sequence mass M_{ZAMS} . The solid line shows the GP predictive mean, while the shaded region denotes the 95% confidence interval. Black points indicate the 187 simulations used as training data. The GP accurately captures the underlying trends without introducing spurious high-frequency structure, and the majority of the training data lie within the confidence band, indicating well-calibrated predictive uncertainties.

$$\mathcal{C}(\alpha) = \frac{\text{number of samples with } |z_i| \leq \Phi^{-1}(\frac{\alpha}{2} + \frac{1}{2})}{N}, \quad (15)$$

where Φ is the CDF of the standard normal distribution. This metric assesses whether or not a particular uncertainty level α is accurately captured by the predictive uncertainties provided by the GP—in other words, $\mathcal{C}(\alpha)$ measures the fraction of observations that lie within the α percentile. A GP's predictive uncertainties are therefore accurate if $\mathcal{C}(\alpha) \approx \alpha$. The results of computing $\mathcal{C}(\alpha)$ for the 0.68, 0.95, and 0.99 percentiles are shown in Table I, which shows that indeed $\mathcal{C}(\alpha) \approx \alpha$. We note that there is some discrepancy for $\mathcal{C}(\alpha = 0.68)$, where the GP uncertainty includes more samples than expected. This

TABLE I. *Left:* Confidence levels α with associated coverages $\mathcal{C}(\alpha)$ as defined in Eq. (15). Our coverage agrees with the confidence levels except for the $\alpha = 0.68$ case, where our emulators overpredict the uncertainty at the 1- σ level. *Right:* K -means fold indices K with their associated R^2 values as defined in Eq. (16). The average R^2 value is shown at the bottom, with a value of ~ 0.90 suggesting strong generalization across different folds.

α	$\mathcal{C}(\alpha)$	K	R^2
0.68000	0.77895	0	0.9159
0.95000	0.95789	1	0.9081
0.99000	0.99474	2	0.9651
		3	0.8618
		4	0.8490
		5	0.9106
		Mean	0.9017

greater-than-expected coverage provides a conservative overestimate of the uncertainty for lower α and is thus favorable.

To assess the generalization of our model to samples outside of training data, we conduct K -fold cross-validation. In particular, we divide our training set into $K = 6$ random subsets, hold out one subset, train the GP and then generate GP predictions on the held out subset, iterating over the K folds. We compute the R^2 value for each fold using

$$R^2 = 1 - \frac{\sum_{i \in \mathcal{F}_K} (y_i - \hat{y}_i)^2}{\sum_{i \in \mathcal{F}_K} (y_i - \bar{y})^2}, \quad (16)$$

for $\mathcal{F}_K$ a set containing the indices of the K th fold, where $\bar{y}$ is the average test output over the K th fold. With this test, we measure the effectiveness of our model against the baseline of simply predicting the average output. Our results are shown in Table I, with $R^2 \approx 1$ indicating effective predictions on the out-of-sample data. We find

an average $R^2 = 0.90174$ across the K folds, indicating strong generalization and predictive performance against the baseline.

B. Optimized vs uniform placement of MESA simulations

We evaluate the efficacy of our GP-guided simulation placement method, illustrating in Fig. 2 the results of the uncertainty-informed grid-building procedure. The black circles and red stars show the initial and iteratively placed simulations, respectively, while the solid lines and light shaded regions represent the GP's mean prediction and 95% credible interval as defined in Eq. (12). The green lines and shaded regions represent the predictions by the GP trained on a uniformly placed grid, with the blue equivalents representing the GP trained on the uncertainty optimized grid.

Immediately, we notice two key differences between the GP predictions. First, the mean predictions from the two methods deviate from similarity near the $M_{\text{ZAMS}} = [9, 30]$

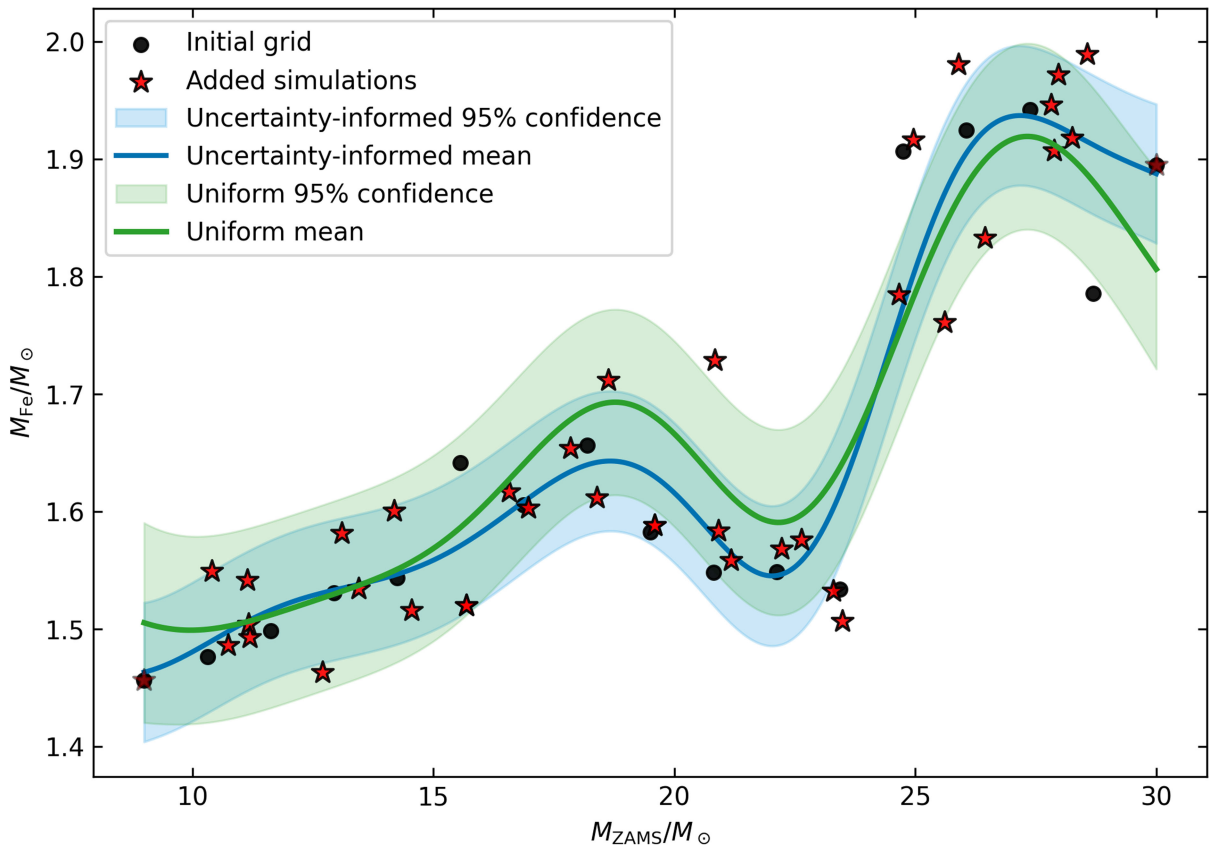


FIG. 2. Comparison of Gaussian process regression fits obtained from uncertainty-informed and uniformly spaced simulation grids for the iron core mass M_{Fe} as a function of M_{ZAMS} . Black points denote the initial grid of 17 simulations, while red stars indicate the 48 simulations added through the uncertainty-informed placement procedure (many overlap at $M_{\text{ZAMS}} = [9, 30]$, see Fig. 3). We do not plot the uniform grid simulations here. The blue curve and shaded region show the predictive mean and 95% confidence interval for the uncertainty-informed grid, while the green equivalents correspond to the uniformly sampled grid. The uncertainty-informed approach yields systematically reduced predictive uncertainty, particularly near the domain boundaries, demonstrating the advantage of targeted simulation placement. Statistical quantities related to each approach are described in Sec. III B.

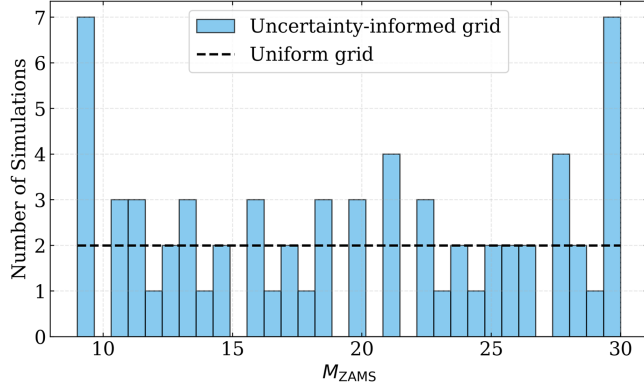


FIG. 3. Distribution of simulation placements in M_{ZAMS} for the final uncertainty-informed grid. The histogram shows the number of simulations per bin, while the dashed horizontal line indicates the expected count for a uniformly spaced grid. The uncertainty-informed placement exhibits a characteristic U-shaped distribution, with increased sampling near the domain boundaries where predictive uncertainty is largest, reflecting the GP’s tendency to prioritize poorly constrained regions of parameter space.

limits and for $15 \leq M_{\text{ZAMS}} \leq 24$. This behavior reinforces the important, though perhaps unsurprising, result that emulator predictions depend not only on simulation number, but on the specific simulation grid. Second, we observe that the uncertainty-informed 95% confidence band is always thinner than the uniform grid’s band. Correspondingly, we obtain the average uncertainty values of $U_{\text{blue}} = 0.119$ and $U_{\text{green}} = 0.159$, respectively. The uncertainty-informed confidence band is also much thinner at the domain boundaries (left edge thickness of 0.118, right edge thickness of 0.118) than the uniform placement confidence band (left edge thickness of 0.170, right edge thickness of 0.170). Note that while the left- and right-boundary values are equal due to rounding, the uncertainty bands are not uniformly wide across M_{ZAMS} .

We examine the domain edge behavior in Fig. 3 which shows the distribution of simulation placements in M_{ZAMS} . The distribution of placed simulations reveals a characteristic “U”-shaped pattern with appreciably more simulations near the domain edges. This behavior arises from the GP’s intrinsically higher predictive uncertainty at the boundaries of the training domain, where data are sparse and lack neighboring samples. Consequently, the uncertainty-informed procedure naturally places additional simulations in these poorly constrained regions near the edges to effectively bound the N -dimensional input space hypercube (in this instance, the edges of our 1D M_{ZAMS} space). See also Fig. 13 of Ref. [48] for additional examples of this phenomenon in a higher dimensional context.

In Fig. 4, we quantitatively track the confidence of the GP, as defined in Eq. (12), as we build the uncertainty-informed simulation grid. In particular, we plot the evolution of the relative uncertainty which we define as the ratio of the uncertainty of current grid iteration to the uncertainty

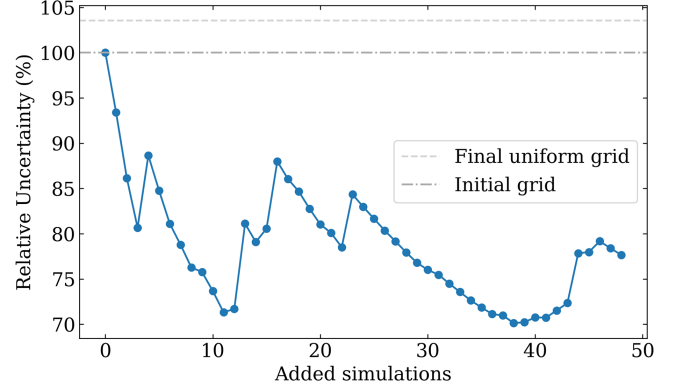


FIG. 4. Evolution of the relative uncertainty U/U_{initial} as a function of the number of added simulations during the uncertainty-informed grid construction. The solid curve shows the uncertainty reduction achieved by iteratively placing simulations at locations of maximal GP predictive variance. The horizontal dash-dotted (bottom) and dashed (top) lines indicate the uncertainty levels of the initial grid and the final uniformly spaced grid, respectively. The uncertainty-informed procedure yields a substantial overall reduction in predictive uncertainty, especially compared to the uniform placement’s increase to over 100% relative uncertainty. However, the GP uncertainty decrease is not strictly monotonic, reflecting the GP’s response to newly resolved structure in the data.

of the initial grid. The relative uncertainties of the GP trained on the initial and final uniform grids are shown as dashed and dot-dashed horizontal lines, respectively. Figure 4 reveals the unintuitive and remarkably important result that *merely adding more simulations into the training grid does not generically lead to a reduction in GP prediction uncertainty*. As the dotted line shows, the uncertainty of the GP trained on the uniform grid is *larger* than that of the GP trained on the initial, smaller grid by about 4%. Meanwhile, the uncertainty of the GP trained on the uncertainty-informed grid is lower by $\sim 20\% - 30\%$. We find that the strategic placement of simulations from our uncertainty-informed algorithm leads to a relative uncertainty curve that overall decreases and results in significant improvement over both the initial and uniform grids, keeping in mind that both final grids contain 65 models each.

C. GP fits to KEPLER-FLASH simulation outputs

In this section, we examine how our optimized simulation placement methodology can be used to both generate maximally informative grids and minimize GP prediction variance for downstream science applications. Specifically, we investigate how constructing an optimized grid of stellar evolution simulations can reduce the GP prediction uncertainty of the bolometric plateau luminosity at 50 days (L_{50}) for supernovae resulting from these stellar evolution simulations. As a reminder to the reader, in this setup we do not run new simulations, but instead select a subset

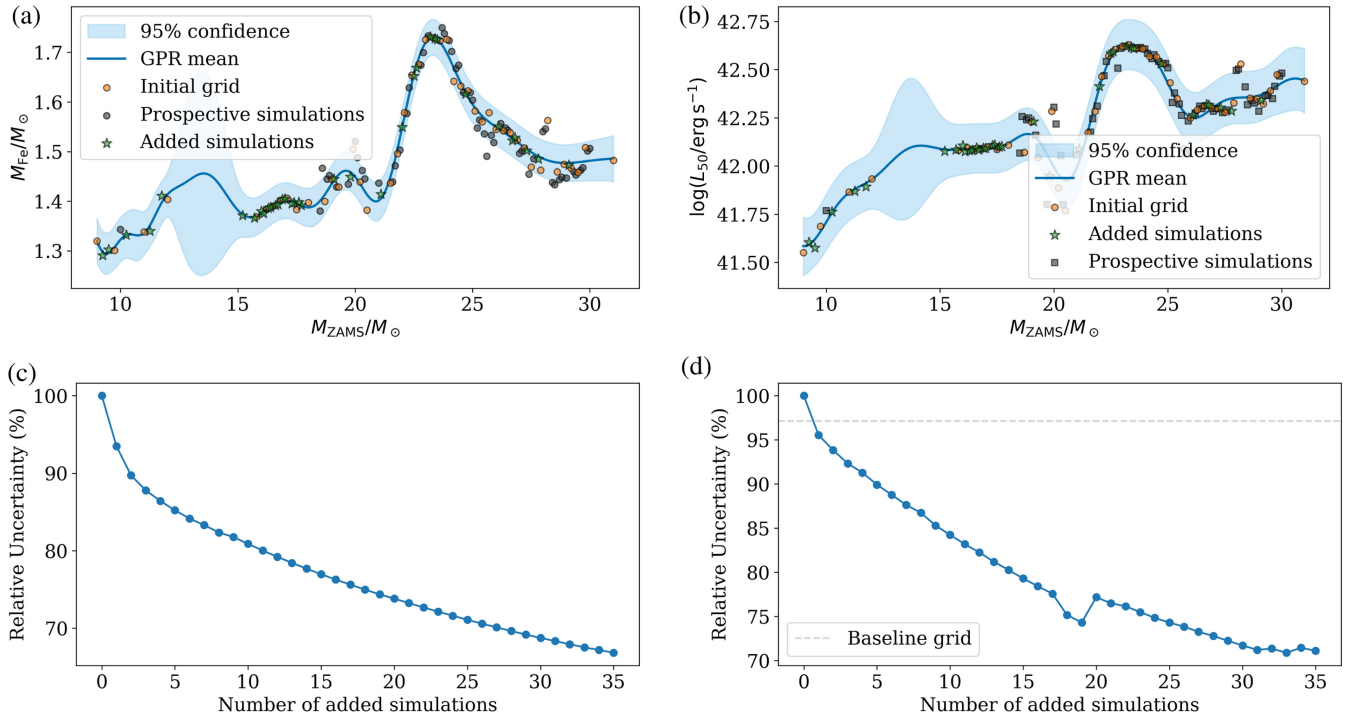


FIG. 5. Gaussian process regression results and uncertainty evolution for the KEPLER-FLASH simulation grid. *Top left*: GP fit to the iron core mass M_{Fe} as a function of M_{ZAMS} , showing the predictive mean and 95% confidence interval, along with the initial grid, added simulations, and prospective simulations. *Top right*: Corresponding GP fit for the supernova observable L_{50} . *Bottom left*: Evolution of the relative uncertainty for the M_{Fe} GP during uncertainty-informed grid construction. *Bottom right*: Evolution of the relative uncertainty for the L_{50} GP evaluated on the same grid, with the dashed line indicating the baseline (randomly selected) grid. Despite optimizing placement using only M_{Fe} , the resulting grid yields comparable uncertainty reduction for L_{50} , demonstrating the effectiveness of upstream, uncertainty-informed sampling for downstream observables. The dashed gray line represents the relative uncertainty of the baseline grid which consists of 22 initial + 35 randomly selected KEPLER-FLASH simulations.

of simulations from the prebuilt KEPLER grid. As described in Sec. II E, we greedily select a new location from the KEPLER grid that minimizes the resulting uncertainty in a GP fit to M_{Fe} (not L_{50} !). The top panels of Fig. 5 show the GP fits to the final grid obtained via our uncertainty-informed grid construction procedure for both M_{Fe} and L_{50} as functions of M_{ZAMS} . We obtain roughly the same final relative uncertainties for both parameters despite carrying out the simulation placement exclusively based on M_{Fe} (we refer the reader to Fig. 10 of [37] and elaborate why this behavior is expected in Sec. IV). We also note the large “bubbling” behavior that occurs between $\sim 13M_{\odot}$ and $\sim 16M_{\odot}$ —this region of KEPLER simulations leads to black hole formation without supernovae and thus induces a gap in the KEPLER-FLASH grid. Thus, the grid construction procedure is forbidden from selecting simulations in this region and the uncertainty in this region propagates consistently such that subsequent inferences are informed of this physical constraint.

In the bottom panels of Fig. 5, we also explore the evolution of the relative uncertainty as we optimize over the KEPLER grid. The bottom right panel highlights the relative uncertainty of the baseline grid, consisting of 22 initial + 35 randomly selected KEPLER-FLASH simulations,

represented by the dashed gray line. The baseline grid improves the L_{50} GP uncertainty by only a factor of a few percent compared to our optimized grid’s improvement of $\sim 30\%$. The significant improvement in uncertainty reduction carried by our method, when compared to the baseline, strongly supports optimized simulation placement for both grid construction and downstream science uncertainty reduction. Achieving this reduction with a subset ($\frac{57}{136} \sim 42\%$) of the total KEPLER-FLASH simulation grid is especially appealing for cases where downstream science applications are dependent on computationally expensive simulation outputs.

IV. DISCUSSION

In Fig. 3, we find that our uncertainty-informed placement procedure exhibits a characteristic U-shaped sampling pattern that suppresses the intrinsically larger uncertainties near domain boundaries. This pattern is typically not captured by common sampling methods such as latin hypercube sampling (LHS), though LHS modifications (see, e.g., Ref. [49]) offer promising semi-naïve alternatives. However, unlike these approaches, we observe that simply adding more simulations based on a sparsity metric does not guarantee improved GP fidelity, demonstrating

that informed simulation placement is critical to constructing a well-sampled model grid. While in this study we validate our uncertainty-informed placement procedure using simplified MESA models, we expect the same qualitative conclusions to hold for more realistic stellar evolution models.

Similarly, in this work, we perform simulation placement in only one input dimension (M_{ZAMS}); however, more general stellar evolution models depend on a broader set of initial conditions. ASTRAL is sufficiently flexible to allow multidimensional input spaces (e.g., with metallicity, rotation), where sparsity is more severe and active placement is expected to yield even greater efficiency gains. We anticipate novel behavior in these regimes, such as entropy imbalance, where placements are preferentially influenced by a single input dimension. This setup requires balancing the trade-off between pinning down the now higher-dimensional boundary (e.g., in two dimensions one expects the U-shape to become a “bowl,” i.e., a rotated U) and exploring the interior of the parameter space. We also guide our placement optimization using a single scalar output as opposed to a multi-objective criterion. In practice, multi-objective selection will be required for more complex models where multiple outputs contribute to downstream sensitivity. Such multiparameter weighting strategies will require additional stellar evolution physics ansatz. This ansatz can, in part, be developed through iterative simulation grid placement for which ASTRAL is ideally suited. While we do not report the results here, we have done preliminary testing of multi-objective GP fits using the GPyTorch library which supports a “multitask” vector prediction. Our analysis, while certainly not exhaustive, suggested that a multi-objective GP fit was either ill-suited to the KEPLER dataset or required more detailed hyperparameter tuning than our uniform approach assumed.

The agreement between M_{Fe} and L_{50} observed in Fig. 5 may seem initially surprising, but it follows directly from the connection between progenitor core structure (i.e. M_{Fe}) and the energy of the resultant explosion. The fundamental properties of fully evolved massive stars correlate quite strongly with a number of stellar evolutionary and explosion properties; see, e.g., Ref. [37]. For example, they find that the correlation between M_{Fe} and E_{expl} , the supernova explosion energy, is 0.93, thus explaining the feature similarity in the top panels of Fig. 5. This one-to-one relation need not hold generally (as the other correlations in Fig. 10 of Ref. [37] suggest), and carrying out the simulation selection on different input and output parameters will generically lead to different relationships between the resulting uncertainty curves. Nevertheless, we expect uncertainty-informed placement to outperform uniform placement in the context of optimization in generic applications. We also expect that the increased relative uncertainty observed in the uniform placement is strongly dependent on both the complexity of features present in

the output parameter and the stochasticity and strength of noise. Future studies, particularly those focused on more complex parameter space exploration, would benefit from an initial pass at understanding these features, including with the help of ASTRAL.

V. CONCLUSION

The ASTRAL framework successfully implements Gaussian process emulation of stellar evolution models, obtaining strong performance with reliable predictive means and well-calibrated uncertainties across a variety of stellar evolution outputs. Furthermore, ASTRAL proficiently establishes a general framework of uncertainty-informed simulation placement, the performance of which we demonstrate by (1) running simple MESA models for optimized grid construction and (2) preferentially selecting a subset of stellar-evolution-informed supernova models from an fixed simulation library (the KEPLER-FLASH grid).

We also demonstrate ASTRAL’s ability to improve GP prediction uncertainty in downstream science which we demonstrate using supernova light-curve predictions. Neutrino-radiation-hydrodynamics core collapse supernova simulations represent a significantly larger computational expense to optimize as compared to stellar evolution simulations. By creating a grid of stellar models that optimally span presupernova core structures we may simultaneously optimize for downstream science while minimizing compute time. ASTRAL represents a dual optimization of both stellar evolutionary and core-collapse supernova objectives, offering greater and more efficient yields in both computational time and science output. We acknowledge the use of the following software: NUMPY [50], MATPLOTLIB [51], SCIPY [52], scikit-learn [47].

ACKNOWLEDGMENTS

We thank Floor Broekgaarden and Mathieu Renzo for insightful discussions during the preparation of the manuscript. We acknowledge the use of Anthropic’s Claude Opus 5 model for generating the analysis and figures in Appendix B only. T.G. acknowledges funding support from NSF Career PHY-2340457, NSF AST-2307888, and Simons Foundation Grant No. SFI-MPS-BH-00012593-09. Los Alamos National Laboratory (LANL) is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001). M.R. acknowledges support from the Information Science & Technology Institute (ISTI) at LANL as an ISTI Postdoctoral Fellow and acknowledges PECASE award funds. B.L.B. acknowledges support from the Advanced Simulation and Computing Program of LANL as a Metropolis Postdoc Fellow. The authors acknowledge ChatGPT for its contribution to defining the ASTRAL

backronym. This article is cleared for unlimited release: LA-UR-26-22778.

DATA AVAILABILITY

The simulation placement optimization code used in this study is publicly available on GitHub [53].

APPENDIX A: GP REGRESSION FOR STELLAR PARAMETER PREDICTION: ADDITIONAL RESULTS

Table II shows the results for GP regression on all of the stellar output parameters. In general, the average R^2 value across the K folds is ≈ 1 , except for T_{PreSN} , R_{Si} , and M_{Si} —these stellar output parameters exhibit sharp, noisy features, which lead to weaker mean predictions. However, coverages remain accurate, indicating that the GP can still quantify its predictive uncertainty, albeit generally, in such cases.

TABLE II. K -fold cross-validation and coverage for all output parameters. $[M, R, T, L]_{\text{PreSN}}$ are the mass, radius, temperature, and luminosity of the star before the onset of core collapse; $s_c, Y_{e,c}, \rho_c$ are the central entropy, central electron fraction, and central density; ξ_M are the compactness parameters defined in Sec. II A; $M_{\text{He,CO,Si,Fe}}$ and $M_{\text{He,CO,Si,Fe}}$ are the masses and radii of the helium, carbon-oxygen, silicon, and iron cores. The T_{PreSN} profile is strongly nonlinear as a function of M_{ZAMS} , but can be naively interpreted as having negative average (avg) R^2 without consideration of more complex features.

Output Parameter	Avg R^2	$\mathcal{C}(0.68)$	$\mathcal{C}(0.95)$	$\mathcal{C}(0.99)$
M_{PreSN}	0.988	0.784	0.953	0.984
R_{PreSN}	0.999	0.832	0.979	0.995
T_{PreSN}	-0.293	0.847	0.974	0.984
L_{PreSN}	0.967	0.916	0.984	1.000
s_c	0.910	0.816	0.942	0.989
$Y_{e,c}$	0.858	0.853	0.958	0.968
ρ_c	0.876	0.821	0.947	0.989
$\xi_{1.5}$	0.821	0.774	0.947	0.974
$\xi_{1.75}$	0.807	0.758	0.958	0.984
$\xi_{2.0}$	0.893	0.816	0.926	0.995
$\xi_{2.25}$	0.921	0.800	0.937	0.974
$\xi_{2.5}$	0.925	0.795	0.937	0.974
$\xi_{2.75}$	0.934	0.800	0.942	0.974
$\xi_{3.0}$	0.944	0.805	0.942	0.974
M_{He}	0.999	0.911	0.984	1.000
R_{He}	0.787	0.853	0.968	0.995
M_{CO}	0.999	0.942	0.984	0.995
R_{CO}	0.910	0.884	0.937	0.958
M_{Si}	0.688	0.758	0.958	0.974
R_{Si}	0.105	0.853	0.942	0.963
M_{Fe}	0.902	0.779	0.958	0.995
R_{Fe}	0.792	0.811	0.947	0.979

APPENDIX B: SCALING BEHAVIOR AT HIGHER DIMENSIONS AND EXTENSION TO BETTER SUITED GP REGRESSION TOOLS

More realistic theoretical stellar evolution models will encompass a higher-dimensional parameter space, including, e.g., M_{ZAMS} , helium abundance Y , metallicity Z , mass-loss rate $\dot{M}$, initial rotational velocity v_{rot} , binarity parameters, and more. The standard case of GP regression presented in the main text scales nominally with training set size N as $\mathcal{O}(N^3)$. The scaling behavior stems from the computational cost of calculating the Cholesky decomposition for the lower-triangular factorization of the covariance matrix $\mathbf{K}$. This limitation makes the `scikit-learn` variation of GP regression intractable as dimensionality d , and conversely the training set size N , increase in the case of more realistic model grid exploration. However, alternate methodologies exist and allow for some general predictions about expected scaling behaviors within the constraints presented below.

To examine these approximate scaling behaviors in the limit of increasing d and large N , we entertain two deviations from our main text’s methodology. First, we generate the analytic forms of M_{Fe} as a function of up to $d = 5$ independent variables presented in Fig. 6 which allow us to evaluate the GP’s performance at large N without necessitating $N = 10,000$ MESA simulations for comparison. The mathematical form of the equations is entirely *ad hoc* to represent the qualitative features (which should not be interpreted as having any physicality) in M_{ZAMS} space and was formulated with the help of Anthropic’s Claude Opus 5 model. The input dimensions are not physically meaningful nor representative of stellar evolution modeling parameters; they simply exist as a tool to test GP scaling with a nontrivial function. The structure of the analytic solutions assumes that each dimension’s contribution is additive, while coupling two of the parameters to resemble the interdependence of certain quantities such as, for example, M_{ZAMS} and $\dot{M}$, while also avoiding a trivial result in which the GP is able to cleanly separate the contributions from each input dimension.

Second, although `scikit-learn` is entirely well suited to the proof-of-concept problem presented in the main text, it is limited in its ability to scale to higher d and larger N given its use of the exact matrix inversion through Cholesky decomposition. Our scaling study is enabled through a more advanced GP regression library called `gpytorch` [54]. The `gpytorch` framework allows much more flexibility in both the construction of exact and approximate GPs, as well as the training setup flexibility enabling the choice of optimizers, learning rate schedulers, and other training hyperparameters as they appear in the standard `PYTORCH` framework [55].

The scaling results as a function of d and N are presented in Fig. 7. For each run, we train an exact `gpytorch` GP for a fixed 200 epochs using the Adam optimizer using

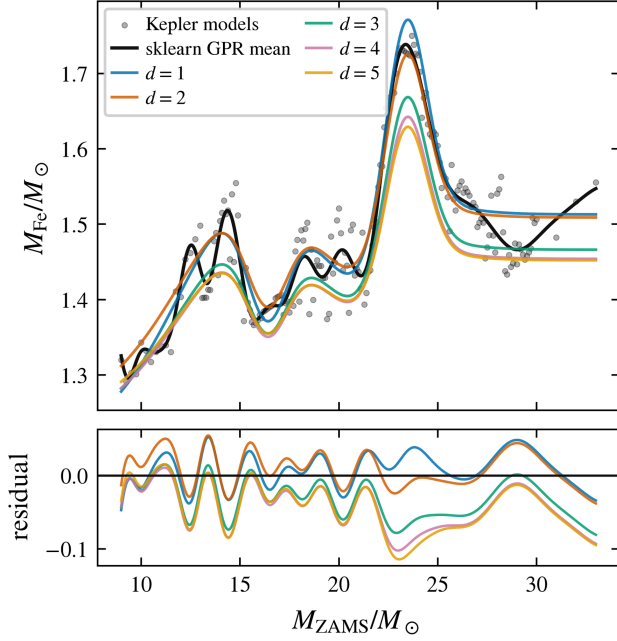


FIG. 6. Analytic approximations of the mean GP prediction from Fig. 1 for dimensionality d up to $d = 5$. The analytic forms are used to approximate the qualitative features of the iron core mass function space while offering an exact comparison for GP scaling tests in lieu of using thousands of MESA simulations.

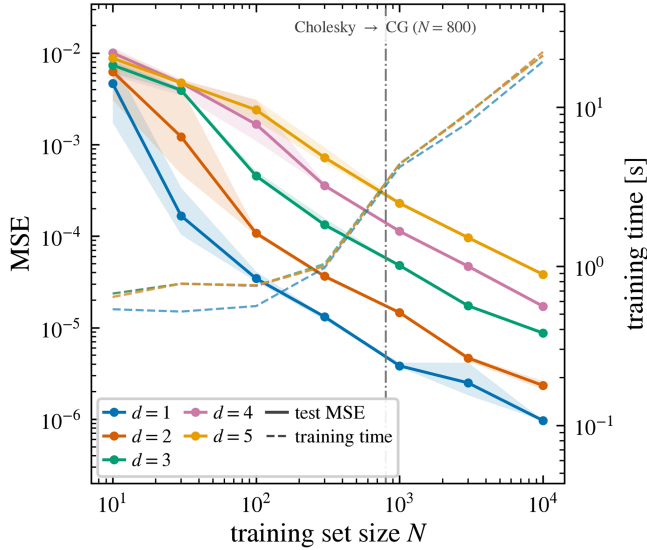


FIG. 7. GP scaling behavior for a gpytorch exact GP compared to the analytic forms shown in Fig. 6 for dimensionality d up to $d = 5$ and training set size N up to $N = 10,000$. The GP's MSE exhibits dependence on d which is expected given that N points fill up less volume in the d -dimensional hypercube as d increases, yielding reduced parameter space coverage. Training time is insensitive to d and dependent solely on N , with Cholesky decomposition ($N \leq 800$) scaling as $\mathcal{O}(N^3)$ and conjugate gradients (CG) scaling as $\mathcal{O}(N^2)$. The change in scaling behavior between these two methods is evident by the training time slope decrease at $N = 1000$.

32 bit floating point representation. The 200 epochs are suitable for getting an idea of scaling behavior, but are not guaranteed, and in fact are quite unlikely, to be representative of fully converged GP solutions. Unlike in the main text, we train on an Apple Macbook Pro M5 Max chip, leveraging the metal performance shaders (MPS) to achieve speed-ups over standard CPU training. For the $d = 1, N = 10,000$ case, our training setup hits the 32 bit noise floor and misrepresents the scaling behavior. Thus, for the $d = 1, N = 10,000$ point *and that point only*, we train our GP using the 64 bit floating point datatype on CPUs so as not to misrepresent numerical noise limits as actual scaling behavior.

The trends in Fig. 7 indicate two main features. First, the GP MSE compared to the analytic forms increases as a function of d ; this result is expected given that N data points cover a smaller volume of the d -dimensional hypercube as d increases. Second, the training time appears effectively insensitive to d ; this comes from gpytorch leveraging iterative conjugate gradient methods to arrive at a solution, reducing training time to $\mathcal{O}(N^2)$. This effect is most readily visible before and after $N = 800$, where Cholesky decomposition is used for $N \leq 800$ and conjugate gradients take over for $N > 800$ and the power-law slope of the training time curve relaxes from ~ 3 to ~ 2 . We conclude that, for any reasonable amount of MESA simulations (e.g. up to 10,000 at the upper limit of

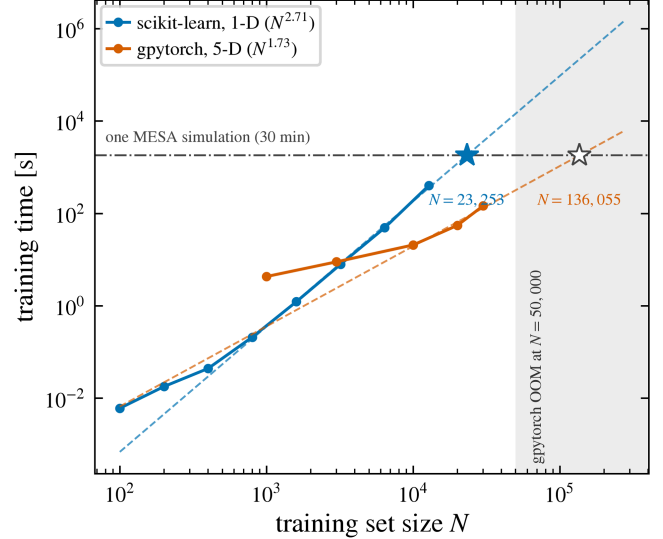


FIG. 8. Calculation of the GP training time assuming a training set of size N . At large enough N , the GP training time may exceed the time cost of generating a new MESA simulation, reducing emulator efficacy. In both the sklearn 1D case and the gpytorch 5D case, we find that GP training remains subdominant to simulation generation cost in the regime of realistic training set sizes. We note that the longer training time for the gpytorch GP at $N < 10,000$ comes from the additional overhead of CPU-GPU data transfer which is inefficient for small N compared to solely CPU calculation.

feasibility), MSE and training time scale to reasonable and efficient values, again with the caveat that our GPs were not trained to convergence. Training setups tailored to individual studies will naturally incur different actual values, but overall we expect the trends to generalize.

Finally, as we consider additional training setups, we must also consider that GP training time may eventually take longer than actual simulation generation. In Fig. 8, we explore the 1D `sklearn` and 5D `gpytorch` GPs and calculate the number of simulations N at which GP training time will outlast the generation of a single MESA simulation, which we arbitrarily assume takes 30 min. We find that, for the 1D case, the crossover happens at $N \sim 20,000$, which is substantially higher than the expected number of simulations required for convergence. We also approximately recover the $\mathcal{O}(N^3)$ training time expected from standard GP regression techniques. In the 5D case, we

are only able to train GPs for $N \leq 30,000$ before running out-of-memory (OOM). Our calculation thus requires extrapolation based on the training time slope (again nearly recovering the expected $\mathcal{O}(N^2)$) of the last few N and suggests that a GP will take longer than a single MESA simulation for $N > 100,000$. This value is again out of the realm of feasibility for simulation generation. In short, with the suggested `gpytorch` setups, we expect the GP training time to remain subdominant to the time required to generate a new simulation. While we reemphasize that our GPs are not trained to convergence, we also note that a sufficiently detailed MESA simulation may take far longer to complete depending on, e.g., nuclear network complexity. Thus, our assumptions underestimate computational costs in both cases, and we expect roughly similar scaling trends to hold in more realistic applications. For any scale of relevance the dominant computational cost remains MESA.

-
- [1] R. Kippenhahn, A. Weigert, and A. Weiss, *Stellar Structure and Evolution* (Springer-Verlag, Berlin Heidelberg, 2012).
 - [2] A. Weiss, W. Hillebrandt, H.-C. Thomas, and H. Ritter, *Cox and Giuli's principles of stellar structure* (Cambridge Scientific Publishers Ltd, United Kingdom, 2004).
 - [3] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, V. B. Adya, C. Affeldt, M. Agathos, K. Agatsuma, N. Aggarwal *et al.*, Observation of gravitational waves from a binary black hole merger, *Phys. Rev. Lett.* **116**, 061102 (2016).
 - [4] B. P. Abbott, R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, V. B. Adya *et al.*, GW170817: Observation of gravitational waves from a binary neutron star inspiral, *Phys. Rev. Lett.* **119**, 161101 (2017).
 - [5] C. Cahillane and G. Mansell, Review of the Advanced LIGO gravitational wave observatories leading to observing run four, *Galaxies* **10**, 36 (2022).
 - [6] R. Abbott, H. Abe, F. Acernese, K. Ackley, S. Adhicary, N. Adhikari, R. X. Adhikari, V. K. Adkins, V. B. Adya, C. Affeldt, D. Agarwal, M. Agathos, O. D. Aguiar, L. Aiello, A. Ain *et al.*, Open data from the third observing run of LIGO, Virgo, KAGRA, and GEO, *Astrophys. J. Suppl. Ser.* **267**, 29 (2023).
 - [7] H. Sana, S. E. de Mink, A. de Koter, N. Langer, C. J. Evans, M. Gieles, E. Gosset, R. G. Izzard, J.-B. Le Bouquin, and F. R. N. Schneider, Binary interaction dominates the evolution of massive stars, *Science* **337**, 444 (2012).
 - [8] E. Laplace, S. Justham, M. Renzo, Y. Götzberg, R. Farmer, D. Vartanyan, and S. E. de Mink, Different to the core: The pre-supernova structures of massive single and binary-stripped stars, *Astron. Astrophys.* **656**, A58 (2021).
 - [9] E. Laplace, F. R. N. Schneider, and P. Podsiadlowski, It's written in the massive stars: The role of stellar physics in the formation of black holes, *Astron. Astrophys.* **695**, A71 (2025).
 - [10] T. A. Weaver, G. B. Zimmerman, and S. E. Woosley, Pre-supernova evolution of massive stars., *Astrophys. J.* **225**, 1021 (1978).
 - [11] B. Paxton, L. Bildsten, A. Dotter, F. Herwig, P. Lesaffre, and F. Timmes, Modules for experiments in stellar astrophysics (MESA), *Astrophys. J. Suppl. Ser.* **192**, 3 (2011).
 - [12] B. Paxton, M. Cantiello, P. Arras, L. Bildsten, E. F. Brown, A. Dotter, C. Mankovich, M. H. Montgomery, D. Stello, F. X. Timmes, and R. Townsend, Modules for experiments in stellar astrophysics (MESA): Planets, oscillations, rotation, and massive stars, *Astrophys. J. Suppl. Ser.* **208**, 4 (2013).
 - [13] B. Paxton, P. Marchant, J. Schwab, E. B. Bauer, L. Bildsten, M. Cantiello, L. Dessart, R. Farmer, H. Hu, N. Langer, R. H. D. Townsend, D. M. Townsley, and F. X. Timmes, Modules for experiments in stellar astrophysics (MESA): Binaries, pulsations, and explosions, *Astrophys. J. Suppl. Ser.* **220**, 15 (2015).
 - [14] B. Paxton, J. Schwab, E. B. Bauer, L. Bildsten, S. Blinnikov, P. Duffell, R. Farmer, J. A. Goldberg, P. Marchant, E. Sorokina, A. Thoul, R. H. D. Townsend, and F. X. Timmes, Modules for experiments in stellar astrophysics (MESA): Convective boundaries, element diffusion, and massive star explosions, *Astrophys. J. Suppl. Ser.* **234**, 34 (2018).
 - [15] B. Paxton, R. Smolec, J. Schwab, A. Gaudschy, L. Bildsten, M. Cantiello, A. Dotter, R. Farmer, J. A. Goldberg, A. S. Jermyn, S. M. Kanbur, P. Marchant, A. Thoul, R. H. D. Townsend, W. M. Wolf *et al.*, Modules for experiments in stellar astrophysics (MESA): Pulsating variable stars, rotation, convective boundaries, and energy conservation, *Astrophys. J. Suppl. Ser.* **243**, 10 (2019).

- [16] A. S. Jermyn, E. B. Bauer, J. Schwab, R. Farmer, W. H. Ball, E. P. Bellinger, A. Dotter, M. Joyce, P. Marchant, J. S. G. Mombarg, W. M. Wolf, T. L. Sunny Wong, G. C. Cinquegrana, E. Farrell, R. Smolec, A. Thoul *et al.*, Modules for Experiments in Stellar Astrophysics (MESA): Time-dependent convection, energy conservation, automatic differentiation, and infrastructure, *Astrophys. J. Suppl. Ser.* **265**, 15 (2023).
- [17] A. Grichener, M. Renzo, W. E. Kerzendorf, R. Farmer, S. E. de Mink, E. P. Bellinger, C.-k. Chan, N. Chen, E. Farag, and S. Justham, Nuclear neural networks: Emulating late burning stages in core-collapse supernova progenitors, *Astrophys. J. Suppl. Ser.* **279**, 49 (2025).
- [18] F. R. N. Schneider, N. Langer, A. de Koter, I. Brott, R. G. Izzard, and H. H. B. Lau, Bonnsai: A Bayesian tool for comparing stars with stellar evolution models, *Astron. Astrophys.* **570**, A66 (2014).
- [19] J. R. Hurley, O. R. Pols, and C. A. Tout, Comprehensive analytic formulae for stellar evolution as a function of mass and metallicity, *Mon. Not. R. Astron. Soc.* **315**, 543 (2000).
- [20] A. Tanikawa, T. Yoshida, T. Kinugawa, K. Takahashi, and H. Umeda, Fitting formulae for evolution tracks of massive stars under extreme metal-poor environments for population synthesis calculations and star cluster simulations, *Mon. Not. R. Astron. Soc.* **495**, 4170 (2020).
- [21] J. W. Barrett, I. Mandel, C. J. Neijssel, S. Stevenson, and A. Vigna-Gómez, Exploring the parameter space of compact binary population synthesis, in *Astroinformatics*, IAU Symposium Vol. 325, edited by M. Brescia, S. G. Djorgovski, E. D. Feigelson, G. Longo, and S. Cavuoti (2017), pp. 46–50, [arXiv:1704.03781](https://arxiv.org/abs/1704.03781).
- [22] S. R. Taylor and D. Gerosa, Mining gravitational-wave catalogs to understand binary stellar evolution: A new hierarchical Bayesian framework, *Phys. Rev. D* **98**, 083017 (2018).
- [23] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (The MIT Press, Cambridge, MA, 2006).
- [24] Z. Doctor, B. Farr, D. E. Holz, and M. Pürrer, Statistical gravitational waveform models: What to simulate next?, *Phys. Rev. D* **96**, 123011 (2017).
- [25] M. W. Coughlin, T. Dietrich, Z. Doctor, D. Kasen, S. Coughlin, A. Jerkstrand, G. Leloudas, O. McBrien, B. D. Metzger, R. O’shaughnessy, and S. J. Smartt, Constraints on the neutron star equation of state from AT2017gfo using radiative transfer simulations, *Mon. Not. R. Astron. Soc.* **480**, 3871 (2018).
- [26] M. Ristić, E. Champion, R. O’shaughnessy, R. Wollaeger, O. Korobkin, E. A. Chase, C. L. Fryer, A. L. Hungerford, and C. J. Fontes, Interpolating detailed simulations of kilonovae: Adaptive learning and parameter inference applications, *Phys. Rev. Res.* **4**, 013046 (2022).
- [27] N. Sarin, T. J. Moriya, A. Singh, A. Gangopadhyay, K.-R. Hinds, S. Schulze, C. M. B. Omand, and K. K. Das, Surrogate models for light curves and photosphere properties of type II supernovae, *Mon. Not. R. Astron. Soc.* **544**, 2653 (2025).
- [28] C. Liu and A. A. Miller, HostSub_GP: Precise galaxy background subtraction in transient spectroscopy with Gaussian processes, [arXiv:2508.15278](https://arxiv.org/abs/2508.15278).
- [29] S. Aigrain and D. Foreman-Mackey, Gaussian Process regression for astronomical time-series, [arXiv:2209.08940](https://arxiv.org/abs/2209.08940).
- [30] K. Maltsev, F. R. N. Schneider, F. K. Röpkke, A. I. Jordan, G. A. Qadir, W. E. Kerzendorf, K. Riedmiller, and P. van der Smagt, Scalable stellar evolution forecasting. Deep learning emulation versus hierarchical nearest-neighbor interpolation, *Astron. Astrophys.* **681**, A86 (2024).
- [31] T. Li, G. R. Davies, A. J. Lyttle, W. H. Ball, L. M. Carboneau, and R. A. García, Modelling stars with Gaussian process regression: Augmenting stellar model grid, *Mon. Not. R. Astron. Soc.* **511**, 5597 (2022).
- [32] J. J. Andrews, S. S. Bavera, M. Briel, A. Chattaraj, A. Dotter, T. Fragos, M. Gallegos-Garcia, S. Gossage, V. Kalogera, E. Kasdagli, A. Katsaggelos, C. Kimball, K. Kovlakas, M. U. Kruckow, C. Liotine *et al.*, POSYDON version 2: Population synthesis with detailed binary-evolution simulations across a cosmological range of metallicities, *Astrophys. J. Suppl. Ser.* **281**, 3 (2025).
- [33] E. Teng, U. Demir, Z. Doctor, P. M. Srivastava, S. Lalvani, V. Kalogera, A. Katsaggelos, J. J. Andrews, S. S. Bavera, M. M. Briel, S. Gossage, K. Kovlakas, M. U. Kruckow, K. A. Rocha, M. Sun *et al.*, Emulators for stellar profiles in binary population modeling, *Astron. Comput.* **51**, 100935 (2025), <https://ui.adsabs.harvard.edu/abs/2025A%26C...5100935T/exportcitation>.
- [34] K. A. Rocha, J. J. Andrews, C. P. L. Berry, Z. Doctor, A. K. Katsaggelos, J. G. Serra Pérez, P. Marchant, V. Kalogera, S. Coughlin, S. S. Bavera, A. Dotter, T. Fragos, K. Kovlakas, D. Misra, Z. Xing, and E. Zapartas, Active learning for computationally efficient distribution of binary evolution simulations, *Astrophys. J.* **938**, 64 (2022).
- [35] M. Ristić, B. L. Barker, S. Cupp, A. Gross, N. Lloyd-Ronning, O. Korobkin, J. M. Miller, and M. R. Mumpower, Kilonovae and long-duration gamma-ray bursts, [arXiv:2509.03003](https://arxiv.org/abs/2509.03003).
- [36] T. Sukhbold, T. Ertl, S. E. Woosley, J. M. Brown, and H.-T. Janka, Core-collapse supernovae from 9 to 120 solar masses based on neutrino-powered explosions, *Astrophys. J.* **821**, 38 (2016).
- [37] B. L. Barker, C. E. Harris, M. L. Warren, E. P. O’Connor, and S. M. Couch, Connecting the light curves of type IIP supernovae to the properties of their progenitors, *Astrophys. J.* **934**, 67 (2022).
- [38] T. A. Weaver, G. B. Zimmerman, and S. E. Woosley, Pre-supernova evolution of massive stars., *Astrophys. J.* **225**, 1021 (1978).
- [39] S. E. Woosley, A. Heger, and T. A. Weaver, The evolution and explosion of massive stars, *Rev. Mod. Phys.* **74**, 1015 (2002).
- [40] E. O’Connor and C. D. Ott, Black hole formation in failing core-collapse supernovae, *Astrophys. J.* **730**, 70 (2011).
- [41] S. M. Couch, M. L. Warren, and E. P. O’Connor, Simulating Turbulence-aided Neutrino-driven core-collapse supernova explosions in one dimension, *Astrophys. J.* **890**, 127 (2020).
- [42] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo, FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes, *Astrophys. J. Suppl. Ser.* **131**, 273 (2000).

- [43] A. Dubey, K. Antypas, M. K. Ganapathy, L. B. Reid, K. Riley, D. Sheeler, A. Siegel, and K. Weide, Extensible component-based architecture for flash a massively parallel, multiphysics simulation code, *Parallel Comput.* **35**, 512 (2009).
- [44] A. Dubey, K. Weide, J. O’Neal, A. Dhruv, S. Couch, J. A. Harris, T. Klosterman, R. Jain, J. Rudi, B. Messer, M. Pajkos, J. Carlson, R. Chu, M. Wahib, S. Chawdhary *et al.*, Flash-x: A multiphysics simulation software instrument, *SoftwareX* **19**, 101168 (2022).
- [45] A. W. Steiner, M. Hempel, and T. Fischer, Core-collapse supernova equations of state based on neutron star observations, *Astrophys. J.* **774**, 17 (2013).
- [46] V. Morozova, A. L. Piro, M. Renzo, C. D. Ott, D. Clausen, S. M. Couch, J. Ellis, and L. F. Roberts, Light curves of core-collapse supernovae with substantial mass loss using the new open-source SuperNova explosion code (SNEC), *Astrophys. J.* **814**, 63 (2015).
- [47] 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 Édouard Duchesnay, Scikit-learn: Machine learning in PYTHON, *arXiv:1201.0490*.
- [48] M. Ristić, Smarter simulation placement of kilonova light curve models for computationally inexpensive surrogate model creation, Master’s thesis, Rochester Institute of Technology, 2020.
- [49] M. Boschini, D. Gerosa, A. Crespi, and M. Falcone, LHS in LHS: A new expansion strategy for latin hypercube sampling in simulation design, *SoftwareX* **31**, 102294 (2025).
- [50] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett *et al.*, Array programming with NUMPY, *Nature (London)* **585**, 357 (2020).
- [51] J. D. Hunter, MATPLOTLIB: A 2D graphics environment, *Comput. Sci. Eng.* **9**, 90 (2007).
- [52] E. Jones, T. Oliphant, P. Peterson *et al.*, SCIPY: Open source scientific tools for PYTHON (2001) [Online; accessed {today}].
- [53] M. Ristić, B. Barker, and T. Gravelly, Astral github repository, Github repository, <https://github.com/astrobarker/astral-gp> (2025).
- [54] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration, in *Advances in Neural Information Processing Systems (NeurIPS 2018)*, Montreal, Canada (2018), <https://ui.adsabs.harvard.edu/abs/2018arXiv180911165G/exportcitation>.
- [55] A. Paszke *et al.*, PYTORCH: An imperative style, high-performance deep learning library, *arXiv:1912.01703*.