MNRAS **000**, 1–13 (2020) Preprint 20 October 2021 Compiled using MNRAS LATEX style file v3.0

Modelling stars with Gaussian Process Regression – I: Augmenting Stellar Model Grid

Tanda Li,[1]★ Guy R. Davies,[1]† Alex Lyttle,[1] Lindsey Carboneau,[1] and A. N. Others[1]
[1] *School of Physics and Astronomy, University of Birmingham, Birmingham, B15 2TT, United Kingdom*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

Grid-based modelling is widely used for estimating stellar parameters. However, stellar model grid is sparse because of the computational cost. This paper demonstrates an application of Gaussian Process (GP) Regression that turns a sparse model grid to a continuous function. We train GP models to map five fundamental inputs (mass, equivalent evolutionary phase, initial metallicity, initial helium fraction, and the mixing-length parameter) to observable outputs (effective temperature, surface gravity, radius, surface metallicity, and stellar age). We preliminarily test different approaches with a small subset of data and set up the training with the most promising methods. To overcome the limitation of training data size in the GP framework, we section the whole grid and train each section separately. An off-grid stellar model dataset is then used to test GP predictions. We find no obvious systematic offsets for all five outputs. The median testing error is ~2K for effective temperature, ~ 0.001dex for surface gravity, ~ 0.002$R_\odot$ for radius, ~ 0.001dex for surface metallicity, and ~0.02Gyr for stellar age. However, we find that local systematic uncertainty is not uniform across the parameter space and it mainly varies with mass, equivalent evolutionary phase, and initial metallicity. We hence train another GP model to describe the systematic uncertainties. We lastly use 100 fake stars to validate the accuracy of GP for modelling stars. GP-determined masses and ages are well consistent with true values within one standard deviation. We also note that GP models give sensible statistical sampling which overcomes the under-sampling issue in the grid-based modelling.

Key words: Star: Modelling – Machine Learning – keywords

1 INTRODUCTION

Theoretical stellar model has been developed for decades to simulate star structure and evolution. However, star modelling is mostly based on sparse stellar grids (e.g. Choi et al. 2016) because model computations are time-consuming. Moreover, stellar model contents adjusted input parameters (e.g. the mixing-length parameter). Varying one of these adjusted parameter adds on an input demission and hence exponentially increases the computational cost. A comprehensive and fine stellar model grid is hence expensive.

A sparse grid is not ideal for the statistics analysis. Classical method like interpolation has been applied overcome this disadvantage. For instance, Dotter (2016) developed a method to transform stellar evolution tracks onto a uniform basis and then interpolate to construct stellar isochrones. More recently, Rendle et al. (2019) uses Bayesian statistics and a Markov Chain Monte Carlo approach to find a representative set of interpolated models from a grid. The interpolation of both works achieve good accuracy for 3-demission girds (inputs are mass, age, and metallicity). However, this approach becomes less reliable in higher demissions and it hence limits the flexibility for varying input physics. The algorithm is another approach that has been used in stellar codes (e.g. Paxton et al. 2013). It offers an automated likelihood minimisation to search for optimal solutions. This method is statistically sound and works fairly well for modelling individual stars. However, it becomes much less efficient while modelling a large sample, because the algorithm needs to iteratively compute stellar tracks many time for each single star.

Machine learning is being applied to the field of stellar research in many ways to efficiently characterise stars. Verma et al. (2016) applied artificial neural networks (Bain 1873; James 1890), i.e., a series of algorithms that endeavours to recognise underlying relationships in a set of data, and successfully determined the evolutionary parameters of the sun and sun-like stars based on spectroscopic and seismic measurements. Based on a similar artificial neural network interference, Hendriks & Aerts (2019) developed a method to achieve optimal initiation of CPU-intensive computations in a 6-dimensional parameter space. The method provides the optimal starting point for further and more detailed forward asteroseismic modelling for the core-hydrogen-burning stage of

★ E-mail: t.li.2@bham.ac.uk
† E-mail: G.R.Davies@bham.ac.uk

intermediate-mass and high-mass stars. Another example is that Mombarg et al. (2021) trained neural networks to predict theoretical pulsation periods of high-order gravity modes, as well as the luminosity, effective temperature, and surface gravity for a given mass, age, overshooting parameter, diffusive envelope mixing, metallicity, and near-core rotation frequency. They compared neural-network predictions with observations to constrain a sample of 37 γ Doradus stars. Using different machine learning tools, Bellinger et al. (2016) trained a random forest regressors(Ho 1995), which is an ensemble learning method for regression and operates by constructing a multitude of decision trees, to rapidly estimating fundamental parameters of main-sequence solar-like stars from classical and asteroseismic observations. Hon et al. (2018) developed a convolutional neural network classifier that analyses visual features in asteroseismic frequency spectra to distinguish between red giant branch stars and helium-core burning stars. Wu et al. (2019) determined masses and ages for massive RGB stars from their spectra with a machine-learning method based on kernel principal component analysis, which is a nonlinear form of principal component analysis using integral operator kernel functions and can efficiently compute principal components in high dimensional feature spaces related to input space by some nonlinear map (Schölkopf et al. 1997). Hon et al. (2020) applied the mixture density network (Bishop 1994), which learns a transformation from a set of input variables to a set of output variable, to determine stars' fundamental parameters like mass and age based on observed mode frequencies, spectroscopic and global seismic parameters. It can be noted that the discriminative machine learning model is mostly used in above studies. The discriminative model treats observables as given facts to directly infer star properties. The method is efficient and easy for computation, however, the downside is not allowing any priors for the star properties in the Bayesian framework. The generative model treats the data set in an opposite direction, says that, using the star fundamental parameters as given facts to predict observables. Predicted observables are then compared with observations to give posterior distributions for fundamental parameters. This approach offers flexibility to prior fundamental parameters. For instance, Lyttle et al. (2021) determined the initial helium fraction and the mixing-length parameters for a sample of Kepler dwarfs and subgiants based on the artificial neural network. The application of the generative model allow them giving either weakly or strongly informative priors whlie constraining the two fundamental parameters.

Because constructing a dense model grid is computationally expensive, we aim to apply the machine learning tool to transform a sparse model grid onto a continuous function, says that, to augment a model grid. We use a different machine learning algorithm that involves a Gaussian process (GP) to map between the stellar fundamental parameters and the observables. The Gaussian process measures the similarity between data points (i.e., the kernel function) to predict values for unseen points from training data. Because the generative model is advanced in terms of the priority, and we treat fundamental parameters as given facts.

We organise the rest of the paper as follow. Section 2 contents descriptions about the computation of a representative stellar model grid. We then introduce the underline theory of GP and set up the training for GP in Section 3. Section 5 demonstrates the results of GP predictions and we analyse the systematic uncertainties. We subsequently augment the stellar grid, present a set of continuously-sampled models, and model 100 fake stars with these GP-trained models for testing the accuracy of our method in Section 6. Finally we discuss advantages and limitations of this approach, highlight areas where improvements can be found in the near future, and summary conclusions in Section 7.

Table 1. Computation of Stellar model grid.

Primary Grid		
Input Parameter	Range	Increment
M ($M_\odot$)	0.80 – 1.20	0.01
[Fe/H] (dex)	-0.5 – 0.2/0.2 – 0.5	0.1/0.05
$Y_{\rm init}$	0.24 – 0.32	0.02
$\alpha_{\rm MLT}$	1.7 – 2.5	0.2
Additional Grid		
Input Parameter	Range	Increment
M ($M_\odot$)	1.055 – 1.195	0.01
[Fe/H] (dex)	0.25 – 0.45	0.1
$Y_{\rm init}$	0.25 – 0.31	0.02
$\alpha_{\rm MLT}$	1.8– 2.4	0.2
Off-grid Models		
Input Parameters		N
Random M, $[{\rm Fe/H}]_{\rm init}$ = 0.0, $Y_{\rm init}$ = 0.28, $\alpha_{\rm MLT}$ = 2.1		44
Random M and $[{\rm Fe/H}]_{\rm init}$, $Y_{\rm init}$ = 0.28, $\alpha_{\rm MLT}$ = 2.1		174
Random M, $[{\rm Fe/H}]_{\rm init}$, $Y_{\rm init}$, and $\alpha_{\rm MLT}$		4880

2 THEORETICAL STELLAR GRID

2.1 Grid computation

We compute a stellar model grid as the training dataset. We aim to cover stars with approximate solar mass on the main-sequence and the subgiant phases. The mass range is set up as 0.8 – 1.2$M_\odot$. The computation of evolutionary tracks starts at the Hayashi line and terminates at the base of red-giant branch (RGB) where $\log g = 3.6$ dex. Note that we only use models after the zero-age-main-sequence (ZAMS). We define ZAMS as the point where core-hydrogen burning contributes over 99.9% of the total luminosity. The stellar gird considers four independent fundamental inputs which are stellar mass (M), initial helium fraction ($Y_{\rm init}$), initial metallicity ($[{\rm Fe/H}]_{\rm init}$), and the mixing-length parameter ($\alpha_{\rm MLT}$). We calculated three model grids. First, a primary grid covers the whole input range. Uniform grid step is applied for M, $Y_{\rm init}$, $\alpha_{\rm MLT}$, and we use tow different grid steps for $[{\rm Fe/H}]_{\rm init}$ below and above 0.2 dex. Second, an additional grid is computed for $M > 1.05 M_\odot$. Grid points of this grid are in between of the primary grid to increase the resolution for tracks with the 'hook'. Third, we compute off-grid models with random fundamental input values as an independent dataset for validating and testing GP models. Details of the computation are listed in Table 1.

We use the stellar code Modules for Experiments in Stellar Astrophysics (MESA, version 12115) to construct stellar grids. MESA is an open-source stellar evolution package which is undergoing active development. Descriptions of input physics and numerical methods can be found in Paxton et al. (2011, 2013, 2015). We adopted the solar chemical mixture $[(Z/X)_\odot = 0.0181]$ provided by Asplund et al. (2009). The initial helium fraction ($Y_{\rm init}$) and initial metallicity ($[{\rm Fe/H}]_{\rm init}$) are independent inputs. The initial chemical composition is calculated with

$$\log(Z_{\rm init}/X_{\rm init}) = \log(Z/X)_\odot + [{\rm Fe/H}]_{\rm init}. \quad (1)$$

We use the MESA $\rho - T$ tables based on the 2005 update of OPAL EOS tables (Rogers & Nayfonov 2002) and OPAL opacity supplemented by low-temperature opacity (Ferguson et al. 2005). The grey Eddington $T - \tau$ relation is used to determine boundary conditions for modelling the atmosphere. The mixing-length theory is implemented and the convection is adjusted by the mixing-length parameter ($\alpha_{\rm MLT}$). We also apply the MESA predictive mixing scheme (Paxton et al. 2018, 2019), which improves model structures at the convective boundary. Atomic diffusion of helium and heavy elements was also taken into account. MESA calculates particle diffusion and gravitational settling by solving Burger's equations using the method and diffusion coefficients of Thoul et al. (1994). We consider eight elements (^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{14}N, ^{16}O, ^{20}Ne, and ^{24}Mg) for diffusion calculations, and have the charge calculated by the MESA ionization module, which estimates the typical ionic charge as a function of T, ρ, and free electrons per nucleon from Paquette et al. (1986). The MESA inlist used for the computation is available on `https://github.com/litanda/mesa_inlist`.

2.2 Equivalent Evolutionary Phase

For training GP models, we need to convert stellar age, whose dynamical range varies track by track, into a uniform range. The fractional age is an option. However, we find that global parameters (e.g. effective temperature) sharply change with the fractional age around the 'hook' and the turn-off point (left panel in Figure 1). It requires a complex and spiky kernel function to fit the curvatures in this area and hence difficult for GP to learn. Dotter (2016) has introduced a quantity Equivalent Evolutionary Phase (*EEP*), which numbers evolutionary stages and transform stellar tracks onto a uniform basis. We follow this idea but define *EEP* in a different way. On each evolutionary track, we compute the displacement between model n and model $n-1$ on the $T_{\rm eff} - \log g$ diagram as

$$\delta d_n = ((T_{\rm eff,n} - T_{\rm eff,n-1})^2 + (\log g_n - \log g_{n-1})^2))^f \,, \quad (2)$$

and the total displacement of model n from the ZAMS (model 0) can be calculated with

$$d_n = \sum_{i=0}^{i=n} \delta d_i . \quad (3)$$

We then normalise d_n to the 0 –1 range and define it as *EEP*. On the same evolutionary track, *EEP* equals to 0 at the ZAMS and 1 on the RGB where log = 3.6 *dex*. The factor f in Eq. 2 is adjustable for modulating *EEP* to avoid obvious gap in the data space, and we find that $f = 0.18$ gives the best data distribution. In Figure 1, we demonstrate how the effective temperature changes with fractional age and *EEP*. It can be seen that the usage of *EEP* significantly smoothes the sharp features at the 'hook' and the turn-off point.

2.3 Data Selection

We need three types of data for training, validating, and testing GP models. Evolutionary tracks in the primary and additional grids are the training data. Off-grid tracks are divided 50-to-50 as validating and testing data . There is a limitation of the data size in the GP framework, because the computational and memory complexity exponentially increase with the number of data points. In practice, the typical data size is on an order of 10^4. Given that the grid contents $\sim 10,000,000$ stellar models, only a small subset can be used. The sampling method is hence critical. A flat sampling is not appropriate, because the evolving step is not uniform at different evolutionary stages due to the MESA step-control strategy. For instance, stellar models are dense at the main-sequence and lower RGB but quite sparse at the subgiant stage. We test a few methods and find that using the displacement (δd_n) defined in Eq. 2 as the weight of sampling give a relatively uniform data distribution.

3 GAUSSIAN PROCESS MODEL

GP can be applied as probabilistic models to regression problems. Here we use the GP model to generalise a stellar model grid to a continuous and probabilistic function that maps inputs to observable quantities. As mentioned in Section 2, our model grid has four independent fundamental inputs, i.e., mass, initial metallicity, initial helium fraction, and mixing-length parameter. Adding the *EEP* which describes the evolutionary phase, the GP model contents five inputs. We intend to train GP models that predict effective temperature, surface gravity, radius, surface metallicity, and stellar age. We summary GP model inputs and outputs as below.

- GP model inputs and their dynamic ranges:
  Mass (M = 0.8 – 1.2M$_\odot$)
  Equivalent Evolutionary Phase (EEP = 0 – 1)
  Initial metallicity ([Fe/H]$_{\rm init}$ = -0.5 – 0.5 dex)
  Initial helium fraction ($Y_{\rm init}$ = 0.24 – 0.32)
  Mixing-length parameter ($\alpha_{\rm MLT}$ = 1.7 – 2.5)
- GPR model outputs:
  Effective temperature ($T_{\rm eff}$)
  Surface gravity ($\log g$)
  Radius (R)
  Surface metallicity ([Fe/H]$_{\rm surf}$)
  Stellar age (τ)

We aim to use the GP model as a non-parametric emulator, that is emulating the comparatively slow calls to models of stellar evolution. This emulator can be described as

$$\text{Outputs} = f(M, EEP, (\text{Fe/H})_{\rm init}, Y_{\rm init}, \alpha_{\rm MLT}). \quad (4)$$

3.1 Gaussian Process Application

In our application to grids of stellar models, a GP has a number of desirable properties. While a GP is a stochastic process, the distribution of a GP can be considered as a distribution of functions with a continuous domain. In fact, the marginal likelihood considered in function space is equal to the likelihood of the data given some function values, multiplied by the prior on those function values marginalised over all function values Williams & Rasmussen (1996). That is to say that, the GP allows for the analytical evaluation of a fit over many different functions (perhaps an infinite number) weighted by some concept of a prior and the agreement with the data. In addition, while the marginal likelihood will be assessed on discrete data, predictions can be made using linear algebra for new data in the continuous domain, but crucially again marginalised over these many different functional forms. It is possible to see how this might be useful for generalising (or emulating or augmenting) a discrete grid of stellar models in order to obtain predictions in the continuous domain.

In this section we will look at the required mathematics to be able to implement a GP for our application to grids of stellar models. We start with a series of definitions before dealing with the marginal likelihood and the posterior predictive distributions.

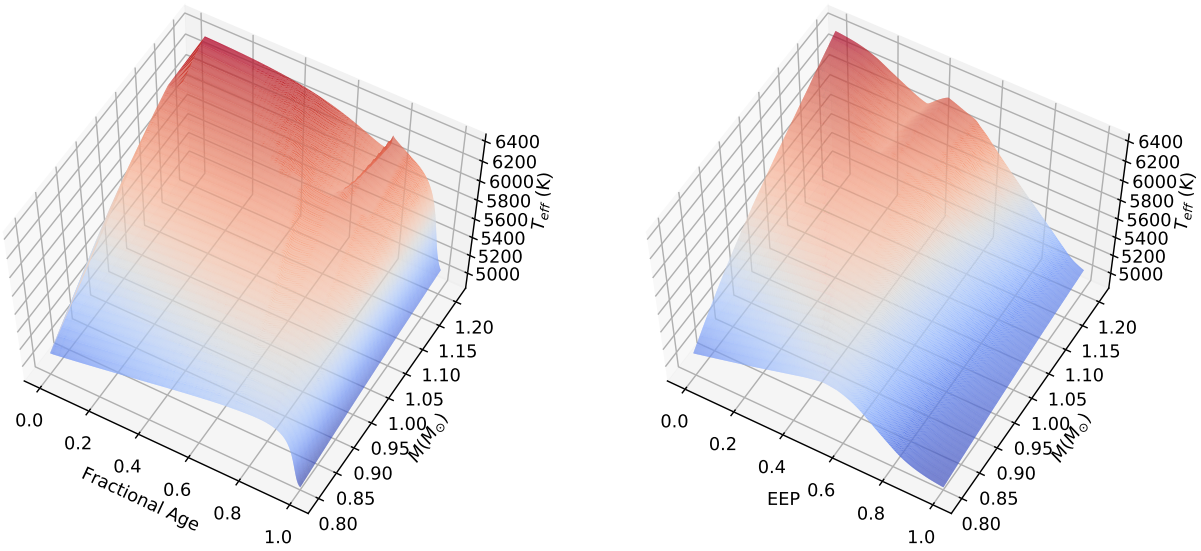

Figure 1. Surface plots of model effective temperature on the mass-fractional age (left) and mass-EEP (right) diagrams. Models in this figure are from the primary grid with fixed initial metallicity ($[\mathrm{Fe/H}]_{\mathrm{init}} = 0.0$), helium fraction ($Y_{\mathrm{init}} = 0.28$) and mixing-length parameter ($\alpha_{\mathrm{MLT}} = 2.1$). It can be seen that the effective temperature changes much smoother on the mass-EEP diagram at the hook and turn-off points.

We start with a grid of stellar models containing N models with a label we want to learn, for example model effective temperature, which we will denote with the general symbol $\mathbf{y}$, and a set of input labels $\mathbf{X}$ (e.g., mass, age, and metallicity). We can use a GP to make predictions of the effective temperature (labelled y) for additional input values given by $\mathbf{X}_\star$. The vector $\mathbf{y}$ is arranged $\mathbf{y} = (y_i, ..., y_N)^T$ where the subscript label references the stellar model. The input labels are arranged into a $N \times D$ matrix where D is the number of input dimensions (e.g., $D = 3$ for mass, age, and metallicity) so that $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$ where $\mathbf{x_i} = (x_{1,i}, ..., x_{D,i})^T$. The matrix of additional inputs $\mathbf{X}_\star$ has the same form as $\mathbf{X}$ but size $N_\star \times D$.

Williams & Rasmussen (1996), from which our description below is based, define a GP as a collection of random variables, where any finite number of which have a joint Gaussian distribution. In general terms, a GP may be written so that our labels are random variables drawn from our GP distribution,

$$\mathbf{y}(\mathbf{X}) \sim \mathcal{GP}\left(m(\mathbf{X}), \mathbf{\Sigma}\right), \tag{5}$$

where $m(\mathbf{X})$ is some mean function, and $\mathbf{\Sigma}$ is some covariance matrix. The mean function controls the deterministic part of the regression and the covariance function controls the stochastic part. The mean function defined here could be any deterministic function and we will label the additional parameters, or hyperparameters, ϕ. Each element of the more familiar covariance matrix is defined by the covariance function or *kernel function* $\mathbf{K}$ which has hyperparameters θ and is given by,

$$\mathbf{\Sigma} = \mathbf{K}(\mathbf{X}, \mathbf{X}, \theta), \tag{6}$$

or

$$\mathbf{\Sigma}_{n,m} = k(\mathbf{X}_n, \mathbf{X}_m, \theta), \tag{7}$$

where the inputs $\mathbf{X}_n$ and $\mathbf{X}_n$ are D-dimensional vectors and the output is a scalar covariance. In addition to the covariance defined by the kernel function, we include additional white noise in the covariance matrix by adding an identity matrix $\mathcal{I}$ multiplied by a scalar value σ_w^2, so that,

$$\mathbf{\Sigma} = \mathbf{K}(\mathbf{X}, \mathbf{X}, \theta) + \sigma_w^2 \mathcal{I}, \tag{8}$$

where σ_w^2 is another hyperparameter to learnt in training.

3.1.1 *The likelihood*

Conceptually we value the GP because of it's ability to marginalise over many functions $\mathbf{f}$ and return a marginal likelihood,

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X})\, \mathrm{d}\mathbf{f}. \tag{9}$$

This integral could be evaluated. However, by noting that a GP is a collection of random variables, where any finite number of which have a joint Gaussian distribution, the marginal probability of our data $\mathbf{y}$ is also the joint likelihood of a multivariate normal distribution,

$$p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(m(\mathbf{X}), \mathbf{\Sigma}), \tag{10}$$

which can be straightforward to evaluate. Thus the marginal likelihood is,

$$p(\mathbf{y}|\mathbf{X}) = (2\pi)^{k/2} \det(\mathbf{\Sigma})^{-0.5} \exp\left(\frac{-1}{2}(\mathbf{X} - m(\mathbf{X}))^T\, \mathbf{\Sigma}^{-1}\, (\mathbf{X} - m(\mathbf{X}))\right), \tag{11}$$

which can be evaluated without integrating over all possible function space. While this marginal likelihood expression is clearly more computationally feasible that the integral over functional space it not without it's limitations. Because it is necessary to calculate the determinant and the inverse of the covariance matrix, typically applied algorithms, make this a $O(N^3)$ or $O(N^2 \log N)$ operation. This naturally limits the size of the data set for which the likelihood, and optimisations of the likelihood, can be applied.

3.1.2 *Making predictions*

If we want to obtain predictive distributions for the output $\mathbf{y}_\star$ given the inputs $\mathbf{X}_\star$ the joint probability distribution of $\mathbf{y}$ and $\mathbf{y}_\star$ is Gaus-

sian and given by

$$p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{y}_\star\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}m(\mathbf{X})\\ m(\mathbf{X}_\star)\end{bmatrix}, \begin{bmatrix}\mathbf{\Sigma} & \mathbf{K}_\star\\ \mathbf{K}_\star{}^T & \mathbf{K}_{\star\star}\end{bmatrix}\right), \tag{12}$$

where the covariance matrices $\mathbf{\Sigma}$ and $\mathbf{K}$ are computed using the kernel function so that

Guy- can we name Σ and K in some way? I feel that a name helps readers to remember what the parameter is.

$$\mathbf{\Sigma}_{n,m} = k(\mathbf{X}_n, \mathbf{X}_m), \tag{13}$$

which is an $N \times N$ matrix.

$$\mathbf{K}_{\star\, n,m} = k(\mathbf{X}_n, \mathbf{X}_{\star\, m}), \tag{14}$$

which is an $N \times N_\star$ matrix, and finally

$$\mathbf{K}_{\star\star\, n,m} = k(\mathbf{X}_{\star\, n}, \mathbf{X}_{\star\, m}), \tag{15}$$

which is an $N_\star \times N_\star$ matrix. The predictions of $\mathbf{y}_\star$ are again a Gaussian distribution so that,

$$\mathbf{y}_\star \sim \mathcal{N}(\hat{\mathbf{y}}_\star, \mathbf{C}), \tag{16}$$

where

$$\hat{\mathbf{y}}_\star = m(\mathbf{X}_\star) + \mathbf{K}_\star^T\, \mathbf{\Sigma}^{-1}\, (\mathbf{y} - m(\mathbf{X})), \tag{17}$$

and

$$\mathbf{C} = \mathbf{K}_{\star\star} - \mathbf{K}_\star^T\, \mathbf{\Sigma}^{-1}\, \mathbf{K}_\star. \tag{18}$$

At point we can make predictions on model properties given a grid of stellar models using equation 16. But these predictions will be poor unless we select sensible values for the form and hyperparameters of the mean function and covariance function. In the following section we detail a number of kernel functions that will be tested against the data. We will then discuss the method for determining the values of the hyperparameters to be used.

3.2 Setting up the GP Model

We adopt a tool package named GPyTorch, which is a GP framework developed by Gardner et al. (2018). It is a Gaussian process library based on an open source machine-learning framework PyTorch (https://pytorch.org). The package provides significant GPU acceleration, state-of-the-art implementations of the latest algorithmic advances for scalability and flexibility, and easy integration with deep learning frameworks. Source codes and detailed introductions could be found on https://gpytorch.ai.

For training GP models, we need set up mean function, kernel function, likelihood function, loss function, and optimiser. To find out the best option for each of above elements, we do a number of preliminary tests. These tests are carried out with 2-demission (2D) GP models which map M and EEP to the five observable outputs (Outputs = $f(M, EEP)$). Training data are selected from the primary grid with fixed $[\mathrm{Fe/H}]_{\mathrm{init}}$ (0.0), Y_{init} (0.28), and α_{MLT} (2.1). There are 41 evolutionary tracks which content 24,257 data points. We also compute 44 evolutionary tracks with the same $[\mathrm{Fe/H}]_{\mathrm{init}}$, Y_{init}, and α_{MLT} but random M for the purpose of validating and testing. We use the method mentioned in Section 2.3 and sample 20,000 data points for training, 10,000 for validating, and 10,000 for testing. We start with the Simple GP Regression example (https://docs.gpytorch.ai/en/stable/examples/01_Exact_GPs/Simple_GP_Regression.html) and develop our own method step by step as follow.

3.2.1 *Mean Function*

We first investigate the mean function. As discussed above, the data distribution is generally smooth but complex in sub-areas. Although a GP model does not significantly affected by the choice of mean function because of the flexibility of kernels, we find that using a constant or a linear mean function leads to a long time for training. For this reason, we apply a Neural Network mean function which is flexible enough to manage both simple and complex features. We adopt an architecture including 6 hidden layers and 128 nodes per layer. All layers apply the linear transformation to the incoming data. We apply the element-wise function (Elu) because it give relatively smooth mean functions. (**Can Guy or Alex add some references for NN and Elu here? The referee may ask why the 6x128 architecture is sufficient, so should we mention Alex's paper?**)

3.2.2 *Likelihood and Loss Function*

We then work on the likelihood function and the loss function. Our training object is a theoretical model grid. There is hence no observed uncertainty for each data point, but a tiny random uncertainty exists due to the approximations in the MESA numerical method. The noise model can be assumed as a Gaussian function with a very small deviation. A likelihood specifies the mapping from latent function values $f(X)$ to observed labels y. We adopt the the standard likelihood for regression which assumes a standard homoskedastic noise model whose conditional distribution is

$$p(y|f(x)) = f + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{19}$$

where σ is a noise parameter. We use a small and fixed noise parameter and run a few tests. However, the strict noise parameter makes GP models hard to converge. When this noise parameter is set as free, it reduces to a small value anyway in the training progress because it is data-driven. For this reason, we decide not put strict constraint for or prioritise this noise parameter. In practice, we only set up a loose upper limit ($\sigma < 0.1$) to speed up the training. One thing should be noted that a GP model with a large noise parameter is not a proper description for the stellar grid. Because of this, we only adopt GP models with $\sigma \lesssim 10^{-4}$. The loss function is simply the exact marginal likelihood in the logarithmic scale.

3.2.3 *Optimiser*

Afterwards, we study the optimiser. We mainly compare between two optimiser named SGD and Adam. Here SGD refers to Stochastic Gradient Descent, and Adam is a combination of the advantages of two other extensions of stochastic gradient descent, specifically, Adaptive Gradient Algorithm and Root Mean Square Propagation. The SGD optimiser in the GPyTroch package involves the formula given by Sutskever et al. (2013). The formula makes it possible to train using stochastic gradient descent with momentum thanks to a well-designed random initialisation and a particular type of slowly increasing schedule for the momentum parameter. The application of momentum in SGD could improve its efficiency and make it less likely to stuck in local minimums. On the other hand, the Adam optimiser includes the 'AMSGrad' variant developed by Reddi et al. (2018) to improve its weakness in the convergence to an optimal solution. With these new developments, the two optimisers give very similar results. We finally choose Adam because it works relatively efficiently and stable. We adaptive learning rate in the training process. Our training starts with a learning rate of 0.01 and decreases

by a factor of 2 when the loss value does not reduce in previous 100 iterations.

3.2.4 *Early Stopping*

We set up an Early Stopping for avoiding overfitting and also determining when to terminate a training progress. When an optimal solution is found, the validating errors stop decreasing and then increase when overfitting occurs. We hence track down the validating error every iteration and terminate training process when the validating error does not decrease in previous 300 iterations.

3.2.5 *Kernel Function*

With the above set up, we test different kernel functions. We involve four basic kernels as listed below:

- RBF: Radial Basis Function kernel (also known as squared exponential kernel)
- RQ: Rational Quadratic Kernel (equivalent to adding together many RBF kernels with different lengthscales)
- Mat12: Matern 1/2 kernel (equivalent to the Exponential Kernel)
- Mat32: Matern 3/2 kernel

We apply each basic kernel and a couple of combined kernels (RBF + Mat21, RQ + Mat21, Mat32 + Mat21, RBF + Mat32, RQ + Mat32) to train and validate GP model for each output. Among these kernels, the combined kernel RBF+Mat21 gives the best fit to the training data, however, it does not give the best predictions for validating and testing data. On the other hand, the Mat32 kernel fits training data reasonably well and it predicts the best for validating and testing data. Comparing between the two kernels, the RBF+Mat12 Kernel is a combination of a smooth and a spiky kernel, which has sufficient flexibility to fit features of data. However, the spiky function make it less accurate for off-grid regions. The smoothness of Mat32 kernel is somewhere between a spiky (Mat21) and a smooth kernel (RBF). It does not exactly fit some sharp features but it maps the off-grid region with better continuity. We hence adopt the Mat32 kernel for the training.

3.2.6 *Training Procedure*

We lastly introduce the procedure of training a GP model. The full process includes training, validating, and testing. In the training process, we literately optimise hyperparameters of a GP model to learn the underline function which maps inputs to outputs from on-grid evolutionary tracks (training dataset). In each iteration, the GP model is validated by comparing true and GP predicted values of off-grid tracks (validating dataset). Although the validating dataset is not directly involved in training hyperparameters, it still constructs the GP model to some extend because the optimal solution is the one that best fits the validating dataset. For this reason, the validating dataset does not give a completely independent validation for a GP model. We hence have a testing process at the end for the purpose of estimating the systematic uncertainty of GP. We reserve a half of off-grid tracks as testing dataset and qualify a GP model using the testing error (true value - GP prediction).

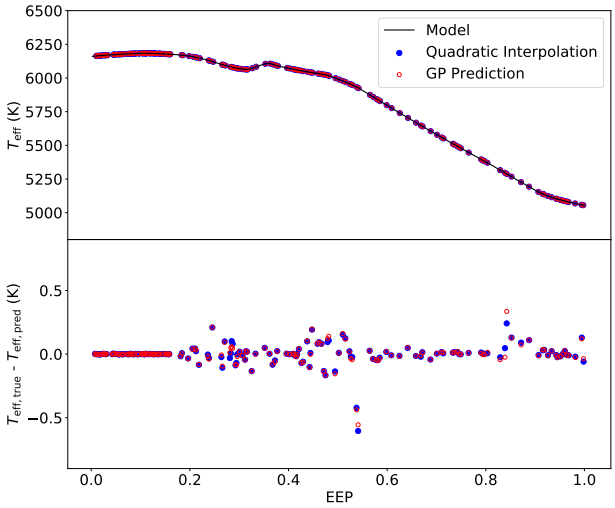

Figure 2. GP application on 1D problem. Models on this track are split into training and testing data by 70-to-30, Top: the evolution of effective temperature for a 1.1$M_\odot$ track. The grey line is the evolutionary track computed with MESA; blue and red circles indicate predictions for the testing data from the quadratic interpolator and the GP model. Bottom: residuals of predictions in the top graph.

4 PRELIMINARY STUDIES ON LOW-DEMISSION PROBLEMS

In this section, we present GP applications on 1-, 2-, and 3-demissions (1D, 2D, 3D) problems. We also investigate two key questions before training the 5-demissions (5D) stellar grid: testing method and strategy for large sample.

4.1 1D Problem

We first demonstrate an example of GP application on an 1D problem. We train a GP model to learn the evolution of effective temperature for a 1.1$M_\odot$ track. We split the data points on this track into training and testing data by 70-to-30. We train a GP model which maps *EEP* to effective temperatures. We then compare true and GP-predicted effective temperatures of testing data. As it can be seen in Figure 2, the GP model gives very accurate predictions: residuals are within ±0.5 K. As a comparison, we fit the training data with the quadratic function and use it to predict testing data. We find that the two methods give very similar results. It indicate that GP can be an alternative of classical interpolators on the 1D problem.

4.2 2D Problem: Investigating testing method

We present a 2D example here. We use the same training, validating, and testing datasets mentioned in Section 3.2 and train GP models which map M and *EEP* to observables. We illustrate the GP model for effective temperature in Figure 3. As it shown that, GP transforms the sparse data into a continuous function and hence is able to predict values for unseen points. We also notice that the kernel function in the area of $M \geq 1.05M_\odot$ and $EEP \leq 0.7$ is more complex than that for other regions because of the appearance of the 'hook'. This particular area are relatively difficult to learn and poorly predicted by the GP model.

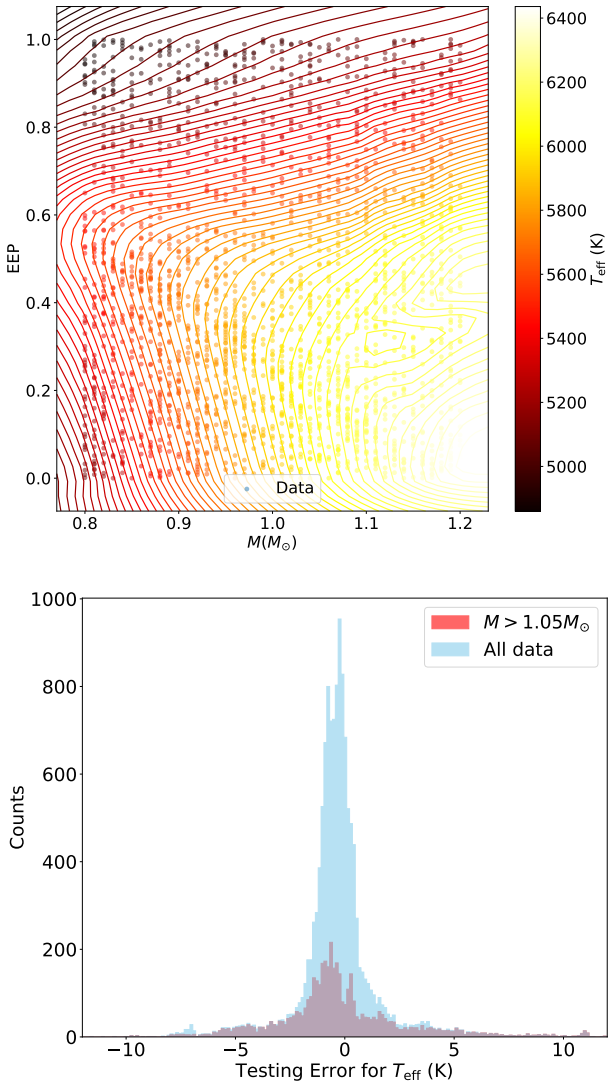

Figure 3. Top: The 2D GP model for $T_{\rm eff}$. Bottom: probability distributions of validating errors of the GP model.

When there is a subregion in which the GP model performs worse than other areas, the probability distribution of testing error is not quite a Gaussian function. We examine the error distribution as shown at the bottom of Figure 3. Errors of most data follows a Gaussian distribution but about 10% data form long tails on both sides. We inspect testing errors for other output parameters and find similar results. The cases of surface gravity and radius are similar to the effective temperature. The surface metallicity is not affected by the hook, but the GP predictions are relatively poor at the early subgiant phase in high-mass tracks. This is because high-mass tracks maintain shallow convective envelope and hence have strong diffusion effect during the main-sequence stage. At the early subgiant phase, the quick expansion of the surface convective envelope brings back the settling heavy elements to the surface, leading to a sharp raise of the surface metallicity. We also find that the accuracy of age prediction drops down for very old low-mass stellar models. This is because age values vary in a relatively big dynamic range (15 - 50 Gyr) in a small fraction of data points. Poor age resolution causes poor GP predictions. This is to say, the tail feature is common for all five outputs. This leads to a question of how to quantity the testing error.

For this case, a global error such like Root Mean Square Error (RMSE) is not ideal to quantify the error. Because it does not well reflect the tail feature. What we need here is a method that reflects the general accuracy level as well as the worst case. We examine the error distribution of each output and count data points in the tails (outside the 3 times of full width at half maximum). The amount of these data are around 10% (8 - 12% for different outputs). For the majority (90%) of data points, which from a Gaussian function, the 68% confidential interval is able to reflect their accuracies. For the worst 10% of the data, we could use the 95% and 99.7% confidential intervals to describe the median value and the length of the tail. Thus, we define an Error Index (EI), which is the sum of 68%, 95%, and 99.7% cumulative values of absolute errors, to qualify GP models. For the case in Figure 3, cumulative values at 68%, 95%, and 99.7% are 1.1, 4.9, and 11.1K, which give an EI equals to 17.1K.

4.3 3D Problem: Strategy for Large Data Sample

As a further step, we apply GP on a 3D grid and investigate the strategy for large data sample. The model grid we aim to train contents about 10,000,000 data points, which is much more than the upper limit of data size (20,000) of an Exact GP model. We work on this with 3-demission (3D) GP models, which map M, *EEP*, and $[\mathrm{Fe/H}]_{\rm init}$ to observable outputs. We select training data from the primary grid with fixed $Y_{\rm init}$ (0.28), and $\alpha_{\rm MLT}$ (2.1). The training dataset contents ~300,000 data points. For validating and testing purposes, we compute another 174 evolutionary tracks with the same input $Y_{\rm init}$ and $\alpha_{\rm MLT}$ but random M and $[\mathrm{Fe/H}]_{\rm init}$. Before proceeding with large sample, we train an Exact GP model using 20,000 training data points as a standard reference.

We first consider the Stochastic Variational GP (SVGP) approach based on the GPyTorch ApproximateGP module. SVGP is an approximate scheme rely on the use of a series of inducing points which can be selected in the parameter space. It trains using minibatches on the training dataset and build up kernels on the inducing points. Underline principles and detailed descriptions of this approach can be found in Hensman et al. (2014). The advantage of SVGP is the large capacity of sample size. However, the kernel complexities is still limited by the amount of inducing points. In our tests, we find a practical issue with the SVGP approach. Because a large training sample takes off the memory, we can only use 10,000 inducing points. This is to say, the kernel complexity of the SVGP model is even simpler than the Exact GP model which uses 20,000 data points. As a result, the SVGP model does not show any improvements. For instance, the 68th, 95th, and 99.7th testing errors for $T_{\rm eff}$ are 2.0, 5.8, and 15.7 K (EI = 23.5K) for the Exact GP model and 2.2, 6.8, and 15.1K (error index = 24.1 K) for the SVGP one. Because the evolutionary feature are complex across multiple demissions, what we need here is increasing the kernel complexity. This requires more data points that actually construct the kernel function. For this particular case, the SVGP downgrades the complexity and hence does not improve the results.

We then investigate another approach designed for large dataset named Structured Kernel Interpolation (SKI GP). SKI GP was introduced by Wilson & Nickisch (2015). It produces kernel approximations for fast computations through kernel interpolation and is a great way to scale a GP up to very large datasets (100,000+ data points). We run a few tests to train a 3D SKI GP model with 100, 000 training data. Compare with the Exact GP and SVGP, its test-

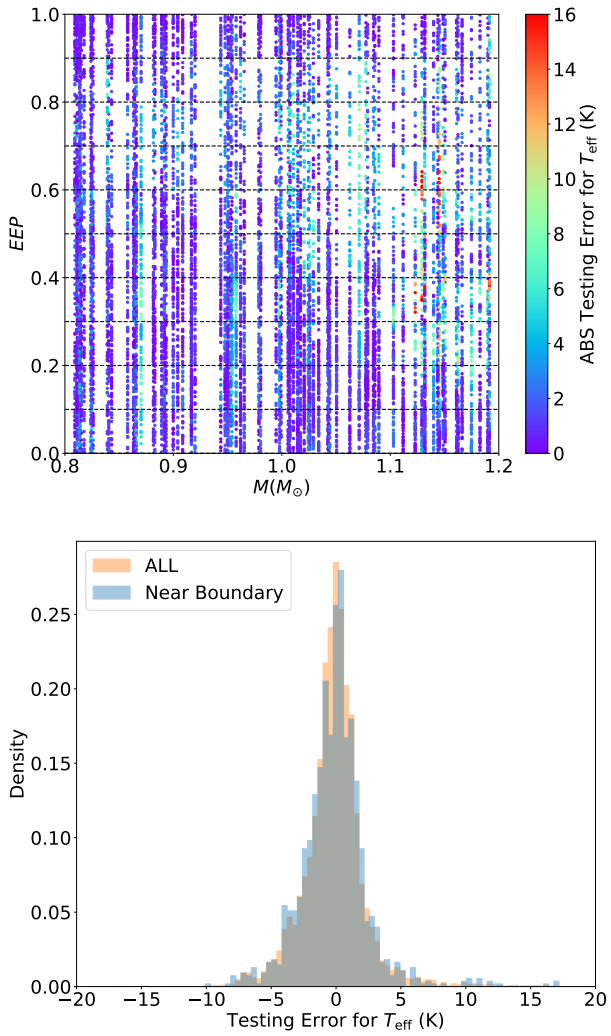

Figure 4. Top: Testing errors of 3D GP model for $T_{\rm eff}$ on the $M - EEP$ diagram. Dashes indicates section boundaries. Bottom: examination of the edge effects of the section scenario. Probability distributions of testing errors of all testing data and those near the boundary (±0.01 EEP) in the upper graph are compared. As it can be seen, testing errors do not raise around the boundary.

ing errors for $T_{\rm eff}$ are slightly improved to 2.0, 6.1, and 14.8K (EI = 22.9K). However, the further test on the 5-demission data is not ideal: a SKI GP model using 100, 000 training data performs much worse than an Exact GP model with only 20,000 training data. The poor behaviour consists with what has been discussed in Wilson & Nickisch (2015). The method poorly scale to data with high dimensions, since the cost of creating the grid grows exponentially in the amount of data. We attempt to make some additional approximations with the GpyTorch AdditiveStructureKernel module. It makes the base kernel to act as one-dimension kernels on each data dimension and the final kernel matrix will be a sum of these 1D kernel matrices. However, the testing errors are not significantly improved.

As mentioned above, the GPU memory captivity limits the actual number of data that induce the kernel function. This limitation becomes critical for the high-demission case. To improve, more training data need to be involved. A simple way is breaking the grid into sections and train GP models for each section separately. Here we divide the training dataset into 10 equal segments by *EEP*. We train one Exact GP model with 20,000 training data for each section. With this section scenario, the testing EIs for five output parameters are averagely improved by around 10%. For instance, the testing EI for $T_{\rm eff}$ decreases from 23.5 to 21.6K.

As expected, the section scenario improves the performance of GP model, but there is a major concern about the edge effect at the boundary between sections. If the GP model works significantly poorly at the section boundaries, it will be difficult to map the systematic errors across the whole parameter space. We examine potential edge effects as illustrated in Figure 4. We inspect absolute testing errors on the $M - EEP$ diagram. No obvious edge effect is found. We also do a statistical comparison between error values of all data points and those around section boundaries ($\pm 0.01 EEP$). As shown in the bottom graph, the density distributions of two samples are very similar to each other, indicating no obvious edge effect. We adopt this section scenario to train the stellar grid in the following study.

5 AUGMENTING THE STELLAR GRID

We GP models for the 5D stellar grid in this section. The training data are selected from evolutionary tracks in both primary and additional girds as mentioned in Table 1. The role of additional grid is increasing the grid resolution for evolutionary tracks with the 'hook'. The total training data includes ~15,000 evolutionary tracks (~10,000,000 stellar models). Off-grid tracks are split by 50-to-50 for validating (in the training progress) and testing (after the training progress) GP models. The section scenario is applied. For each section, we train one Exact GP model for each output parameter with 20,000 training and 20,000 validating data. For the testing purpose, we sample 50,000 data points in the whole*EEP* range. Note that we do not use models with $\tau \geq 20.0$ Gyr, $[Fe/H]_{\rm surf} \leq$ -0.6 dex, or $T_{\rm eff} \geq 7000K$ in testing dataset.

Here we summary our set up for training GP models for the stellar grid.

- Model Type: Exact GP with the section scenario
- Kernel: Mat32
- Mean Function: Neural Network with 6 linear layers x 128 nodes per layer and element-wise function (Elu)
- Likelihood Function: Gaussian Likelihood Function
- Loss Function: Exact marginal likelihood
- Optimiser: Adam including AMSGRAD variant
- Early Stoping: controlled by the validating *EI*

5.1 Overview of Results

At the beginning, we train an Exact GP model for the whole *EEP* range as a standard reference. Testing errors for this model are listed in Table 2. Compared with 2D and 3D cases, testing errors remarkably rise with increasing demission. For example, *EI* for $T_{\rm eff}$ goes up to 46K, which is much higher than that for the 2D (16K) or the 3D model (24K).

We then train GP models with the section scenario. We gradually increase the number of sections and track down the changes in testing errors. We find that testing *EI* is not significantly improved when the grid are divided by more than 10 sections. We list the testing errors for different cases in Table 2. As it shown that, testing *EI*s for 10-, 20-, and 100-sections cases are very close. It turns out that

the 10-sections case is the most efficient strategy and we take this case as the best result. We refer this model set as GP-Grid Model here after. Following analysis are all based on it.

A overview of testing errors can also be seen in Figure 5, where we demonstrate rolling medians and rolling standard deviations of testing errors as a function of input parameters. Median values are approximate along zero in most plots, indicating good agreement between GP predictions and true values. The 68% confidential intervals are generally small and do not significantly vary. However, the 95% confidential intervals vary in large dynamic ranges and clearly depend on M, EEP, and $[Fe/H]_{\rm init}$. The long tails in marginal distributions are similar to what is illustrated in Figure 3: GP predictions are relatively poor in some particular regions. Because the systematic uncertainty are not uniform through the parameter space, marginal error distributions do not well describe the systematic uncertainties. We hence investigate systematic uncertainty in a local scale.

5.2 Mapping Systematical Uncertainties using another GP model

As shown in Figure 5, systematical uncertainties relate to M,EEP, and $[Fe/H]_{\rm init}$ but not to $Y_{\rm init}$ or $\alpha_{\rm MLT}$. Thus, a comprehensive model can be described as a function of M,EEP, and $[Fe/H]_{\rm init}$. In the M-EEP-$[Fe/H]_{\rm init}$ space, we divide this 3D space into equal-size segments and examine local error distributions. The choice of segment size matters. It needs to be small enough for only presenting the local feature, but it can not be very small so that there are not enough data points for proper statistical analysis. To find an appropriate size, we apply a statistic test. The purpose of the test is to check wether the local distribution has a tail feature. The condition we apply is either the ratio between 68% and 95% confidential intervals less than 2.5, or the ratio between 68% and 99.7% confidential intervals less than 4. After several attempts, we decide to divide the input range into 40 equally spaced segments for M, 50 for EEP, and 20 for $[Fe/H]_{\rm init}$. Hence there are 43,911 (41 x 21 x 51) gird points. We compute a rolling standard deviation for each grid point by using the data in a 3-segments (1.5 previous and 1.5 after) range across each demission. The statistic test shows that $\sim$ 90% points meeting the above condition.

We present local systematic uncertainties for $T_{\rm eff}$ ($\sigma T_{\rm eff}$) on the $M-EEP$ diagram in Figure 6. As it can be seen that, local $\sigma T_{\rm eff}$ values are mostly below $\sim$4K in the low-mass regions and raise up when mass is greater than 1.05$M_\odot$. More important, there are substructures randomly appear in the parameter space. We visually inspect local systematic uncertainties for all output parameters and find similar features. Because of the existence of substructures, it is not convinced to describe the systematic uncertainty with any simple mathematical expressions. We hence use another GP model (GP-SYS model here after) to map three fundamental inputs (M, EEP, and $[Fe/H]_{\rm init}$) to local systematic uncertainty of each output parameter.

Here we discuss how we set up the GP-SYS model. There are 43,911 data points in total. We randomly select 32,000 data points as the training dataset and use the rest 11,311 data points as the validating dataset. Because the training data size exceeds the 20,000 limit for Exact GP, we use other approach for the large data sample. As discussed in Section 3, the SVGP framework can be applied for large data sample when the kernel function is not very complex. It hence suitable for training the systematic uncertainty. We use 10,000 inducing points which are randomly selected in the 3D space. The training data is split into 10 batches for the SVGP training. We use a constant mean function and the RBF kernel because the data distribution is smooth. The variational evidence lower bound (ELBO) is adopted as the loss function. This approach is designed for when there is too much data for the exact inference. We set up Early Stopping by tracking the RMSE value of validating data. The training progress terminates when the RMSE value stops decreasing for 30 iterations. The likelihood function and the optimiser are same as those for training GP-Grid models. Our set up for the GP-SYS model is listed as follow.

- Model Type: SVGP with 10,000 inducing number.
- Model Inputs: M, EEP, and $[Fe/H]_{\rm init}$
- Model Outputs: $\sigma_{T_{\rm eff}}$, $\sigma_{\log g}$, σ_R, $\sigma_{[Fe/H]_{\rm surf}}$, and σ_τ.
- Training dataset: 3200 x 10 data points
- Validating dataset: 11,311 data points
- Kernel: RBF (for all outputs)
- Mean Function: Constant Mean Function
- Likelihood Function: Gaussian Likelihood Function
- Loss Function: The variational evidence lower bound (ELBO)
- Optimiser: Adam including AMSGRAD variant
- Early Stoping: when validating RMSE stops decreasing for 30 iterations

We train GP-SYS models with the above method. Figure 6 includes a comparison between the actual and the GP-trained $\sigma_{T_{\rm eff}}$. It shows that the GP-SYS model well reproduces the $\sigma_{T_{\rm eff}}$ distributions. However, it should be noted that the GP-SYS model is only a smooth approximation because the training data resolution is not sufficient enough. We examine validating errors for all five outputs and summary their average values in Table 3. For instance, the average validating error for $\sigma_{T_{\rm eff}}$ is only 0.15K. Given that the residuals are much smaller than typical observed uncertainties, they can be ignored.

6 MODELLING STARS WITH GP-TRAINED MODELS

Now we are able to predict observable outputs with the GP-Grid model and estimate the systematical uncertainty for each prediction with the GP-SYS model. We demonstrate a GP-trained Kiel diagram comparing with the stellar grid in Figure 7. GP has transformed the sparse model grid into a non-sparse model set.

6.1 A GP-Trained Model Set

Here we augment the stellar grid. We randomly sample 5,000,000 model data points with uniform distribution for all inputs. This GP-trained model set hence includes five fundamental inputs (M, EEP, $[Fe/H]_{\rm init}$, $Y_{\rm init}$, and $\alpha_{\rm MLT}$), five outputs, ($T_{\rm eff}$, $\log g$, R, $[Fe/H]_{\rm surf}$, and τ), and five systematic uncertainties ($\sigma_{T_{\rm eff}}$, $\sigma_{\log g}$, σ_R, $\sigma_{[Fe/H]_{\rm surf}}$, and σ_τ). This model set can be downloaded at `a-place-for-data`.

6.2 Modelling Fake Stars with GP-trained Models

With the GP-trained model set, we model 1,000 fake stars to examine the accuracy of our method. Fake stars are randomly selected on off-grid tracks. To avoid the edge effect, fakes stars are selected in the range of $T_{\rm eff}$ = [4700K, 6800K], $\log g$ = [3.7, 4.6], $[Fe/H]_{\rm surf}$ = [-0.35,0.35], M = [0.85,1.15], EEP = [0.05,0.95], $Y_{\rm init}$ = [0.25,0.31], and $\alpha_{\rm MLT}$ = [1.8,2.4]. We use four observables, i.e., $T_{\rm eff}$, $\log g$, R, and $[Fe/H]_{\rm surf}$, as constraints. We apply typical observed uncertainty that is ±50K for $T_{\rm eff}$ (high-resolution spectroscopy),

Table 2. Training and validating errors for GPR Models

Model Type	Inputs	$N_{\rm Training}$	Sampling rate	Testing Errors (at 68/95/99.7%)				
				$T_{\rm eff}$ (K)	$\log g$ (10^{-3}dex)	R ($10^{-3}R_\odot$)	$[Fe/H]_{\rm surf}$ (10^{-3}dex)	τ (10^{-2}Gyr)
Exact GP	2D	20,000 x 1	96%	1/5/11	1/3/8	2/6/14	0.5/2/12	1/3/9
Exact GP	3D	20,000 x 1	5%	2/6/16	1/4/10	3/7/17	2/6/22	2/7/22
Exact GP (10 sections)	3D	20,000 x 10	50%	2/5/15	1/4/11	2/7/17	1/3/20	2/6/19
Exact GP	5D	20,000 x 1	0.2%	3/9/34	2/5/18	4/11/36	2/7/30	3/9/27
Exact GP (3 sections)	5D	20,000 x 3	0.6%	3/8/27	2/5/18	3/7/26	1/4/24	3/7/22
Exact GP (5 sections)	5D	20,000 x 5	1%	2/7/25	1/4/15	3/7/24	1/4/21	2/6/22
Exact GP (10 sections)	5D	20,000 x 10	2%	2/7/27	1/4/14	2/7/26	1/4/20	2/6/21
Exact GP (20 sections)	5D	20,000 x 20	4%	2/7/26	1/4/14	2/7/27	1/3/18	2/6/22
Exact GP (100 sections)	5D	20,000 x 100	20%	2/7/25	1/4/14	2/7/26	1/3/17	2/6/18

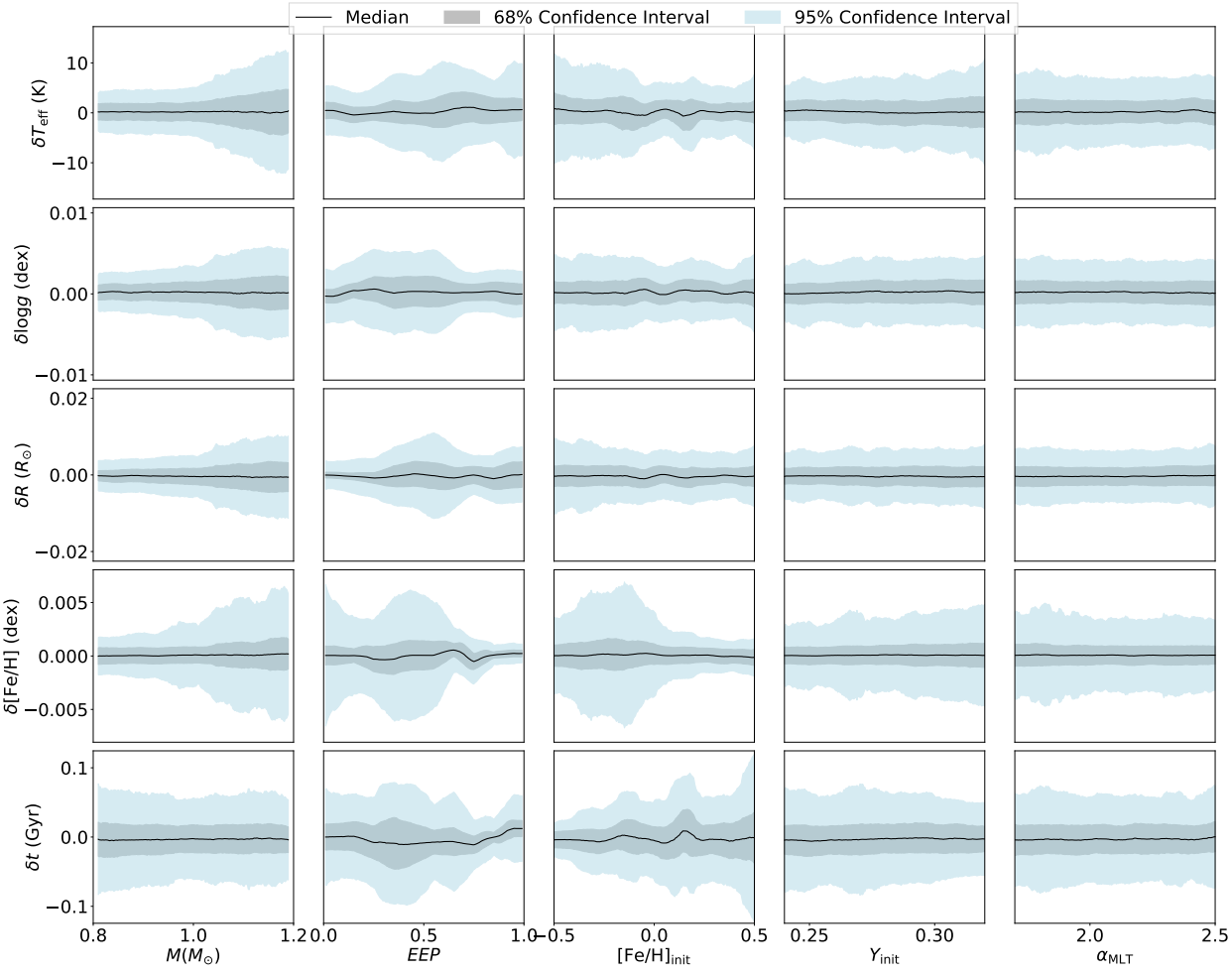

Figure 5. Roll medians and 68/95% confidential intervals of testing errors against GP model inputs. Black solid lines indicate the median value; grey and blue shadowes represent the 68% and 95% confidential interval. Testing errors of $T_{\rm eff}$, $\log g$, and R mainly depend on M andEEP. Metallicity error strongly depends on M ,EEP, and $[Fe/H]_{\rm init}$, and age error has a significant correlation toEEP and $[Fe/H]_{\rm init}$. However, testing errors do not obviously relate to $Y_{\rm init}$ or $\alpha_{\rm MLT}$.

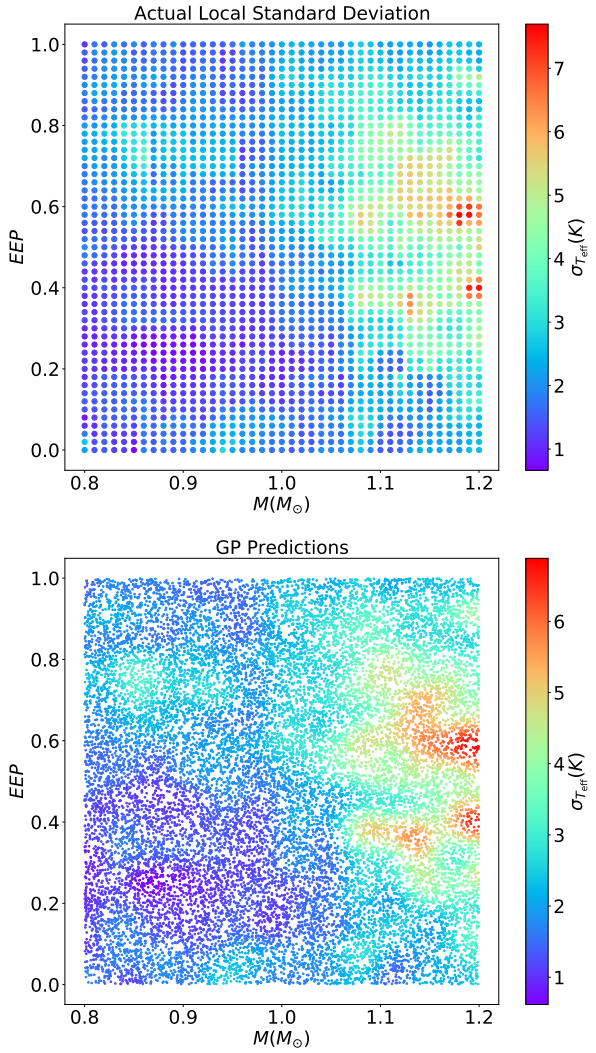

Figure 6. Local systematic uncertainty (1-σ) for $T_{\rm eff}$ on the $M-EEP$ diagram for $[Fe/H]_{\rm init}$ = 0.0. The actual distribution is on the top and GP predictions are presented at the bottom.

Table 3. Residual of GP models for systematic uncertainty

GP output	Average Validating Errors (ABS)
$\sigma_{T_{\rm eff}}$ (K)	0.15
$\sigma_{\log g}$ (10^{-3}dex)	0.08
σ_{R} ($10^{-3}R_\odot$)	0.2
$\sigma_{[Fe/H]_{\rm surf}}$ (10^{-3}dex)	0.09
σ_{τ} (10^{-2} Gyr)	0.2

$\pm$0.005dex for $\log g$ (seismology), 3% for R (seismology), and $\pm$0.05dex for $[Fe/H]_{\rm surf}$ (high-resolution spectroscopy). Observed value for each constraint is calculated with true value plus a random noise which follows a Gaussian distribution. For example, we pick a model with $T_{\rm eff}$ = 5000K and generate a random noise of -25K from a Gaussian function whose standard deviation is 50K. The fake observed value is 4975$\pm$50 K.

We fit fake stars using the Maximum Likelihood Estimate (MLE) method with the original stellar grid and the GP-trained models. Note that the error term in MLE formula contents observed and systematic uncertainties given by GP-SYS models ($\sigma^2 = \sigma^2_{obs} + \sigma^2_{sys}$) when fitting to GP-trained models. We make marginal likelihood distribution and use the 16th, 50th, 84th percentiles to estimate a fundamental parameter and its uncertainty. We present likelihood distributions of inferred stellar parameters for a representative fake star in Figure 8. Observed constraints for this fake star are $T_{\rm eff}$ = 4926$\pm$50K, $\log g$ = 4.536$\pm$0.005, $[Fe/H]_{\rm surf}$ = 0.34$\pm$0.05, and R = 0.829$\pm$0.025$R_\odot$. True values of fundamental stellar parameters are M = 0.861M$_\odot$, τ = 10.8Gyr, $[Fe/H]_{\rm init}$ = 0.403, $Y_{\rm init}$ = 0.281, and $\alpha_{\rm MLT}$ = 2.356. Compared with the stellar grid, GP-trained model set has a completed statistical sampling and hence gives more sensible posteriors. The improvement for the age is obvious. It is under-sampled in the model grid and hence the inferred age does not actually converge. The Gird-based modelling infers an age of $7.7^{+3.2}_{-4.2}$Gyr, and GP-based modelling gives $8.3^{+2.6}_{-2.8}$Gyr. Compared with the true value (10.8 Gyr), GP determines a more accurate and precise result than the original grid. For initial metallicity, initial helium fraction, and the mixing-length parameter, GP makes it possible to statistically estimate these parameters. In this example, GP determines $[Fe/H]_{\rm init}$ and $Y_{\rm init}$ values very close to true values. The mixing-length parameter is not constrained because no correlated observable is given. This comparison clearly shows the advantage of GP-based modelling. It clearly improve the sampling of a sparse grid and hence improve the accuracy and precision of estimates.

We now examine the accuracy of inferred mass and age of GP-based modelling. We calculate offsets between true and estimated values and divide them by estimated uncertainties for 1,000 fake stars. If offsets form a normal distribution, it would indicate that the fitting is only affected by random noises. We demonstrate this examination in Figure 9. Mass offsets nicely follow a normal distribution for both the grid and GP-trained models. Age differences fits a normal distribution for GP-trained models, but slightly shift by ~-0.2 for the grid. This trend consists with what is illustrate in Figure 8: the centre of probability distribution of age shifts to lower end because of the under-sampling issue. The results infer that GP-based modelling gives reasonable accurate estimates, and moreover, it fixes the systematic offsets in the original stellar grid.

7 DISCUSSION AND CONCLUSIONS

In this work, we apply a machine-learning algorithm that involves a Gaussian process for the purpose of augmenting a stellar grid. We train GP models for the stellar grid and obtain continuous functions that map fundamental inputs to observable outputs on a good accuracy level. We also train another set of GP models which describe the local systematic uncertainty of each output parameter. A GP-trained model set is then generated and we use it to model fake stars. The GP-based modelling is obviously advanced in modelling individual stars compared with the gird-based modelling because it provides continuous sampling. We lastly test GP-determined masses and ages by comparing them with true values and find good consistence.

Advantages

GP is a very efficient tool to augment a stellar grid instead of computing massive evolutionary tracks. For the case in this study, we train 10 GP-Grid models and 1 GP-SYS model for each output parameter and hence 55 GP models in total. The training process takes approximate one hour per GP model on a NVidia Tesla V100

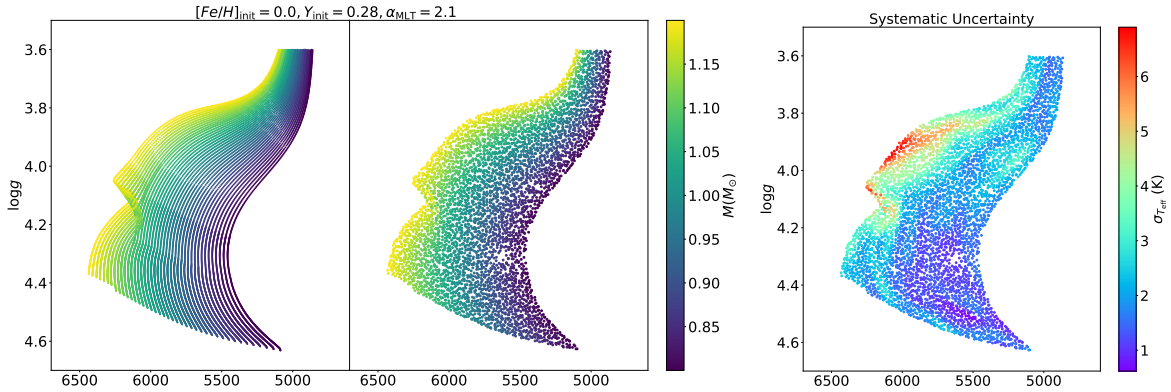

Figure 7. Left: a GP-trained Kiel diagram compared with the stellar grid. Right: GP-predicted systematical uncertainties for all GP-trained models.

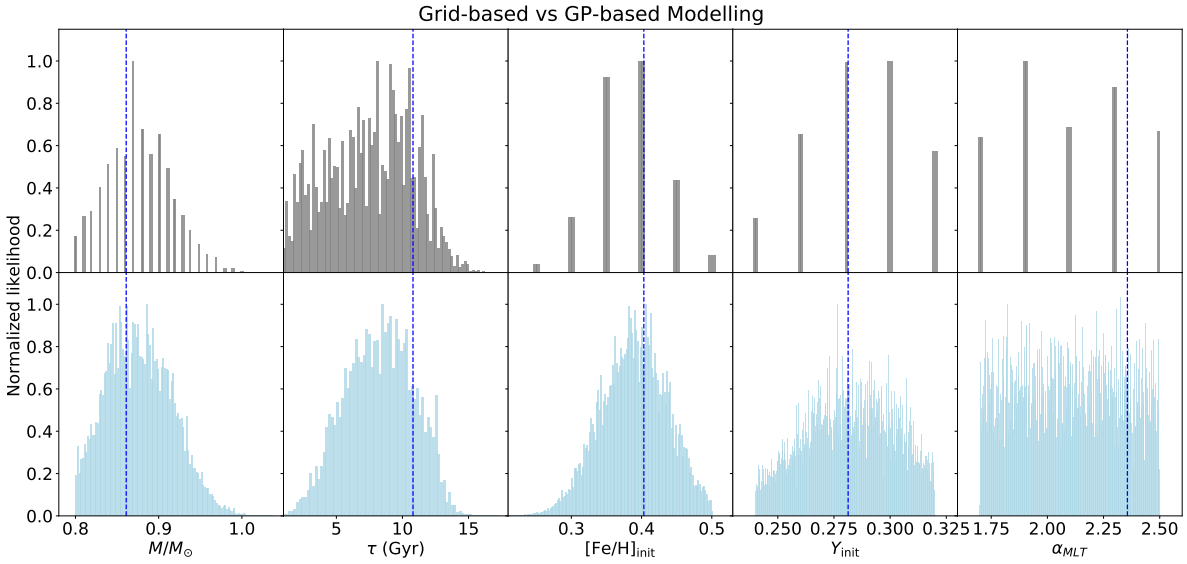

Figure 8. Probability distributions of estimated fundamental parameters from grid-based (Top) and GP-based modelling (Bottom) for a fake star. Observed constraints for this fake star are $T_{\rm eff}$ = 4926±50K, $\log g$ = 4.536±0.005, $[Fe/H]_{\rm surf}$ = 0.34±0.05, and R = 0.829±0.025$R_\odot$. True values of fundamental parameters are M = 0.861 $\rm N_\odot$, τ = 10.8 Gyr, $[Fe/H]_{\rm init}$ = 0.403, $Y_{\rm init}$ = 0.281, $\alpha_{\rm MLT}$ = 2.356, which are represented by blue dashes.

graphics processing unit (GPU). We also show that GP is able to manage the augmentation in high-demissions data (5 demissions in this work), which is very difficult for the interpolation approach. Moreover, the section scenario overcomes the limitation of training data size (20,000 for this case) and offers flexibility to apply GP on stellar grids with different size. When modelling individual stars, GP significantly improves the sampling especially for initial metallicity, helium fraction, and the mixing-length parameter, which are always well spaced in a stellar grid.

Any more points?

Limitations

GP is efficient for training global parameters but not for training the stellar structure. It hence not able to replace a stellar model. To obtain comprehensive stellar model, an optimal path would be using GP to constrain the range of fundamental parameters and then computing models with stellar code in that range.

add more limitations

Future works

This work presents an application of GP on augmenting the whole grid. However, this method is not efficient when modelling a single star. Our future work is developing a fast tool that only trains a small set of models in the grid around a star. This trained-GP model can be easily sticked with an Markov chain Monte Carlo simulator for a delicate bayesian analysis.

more future works

ACKNOWLEDGEMENTS

Development of GPyTorch is supported by funding from the Bill and Melinda Gates Foundation, the National Science Foundation, and SAP.

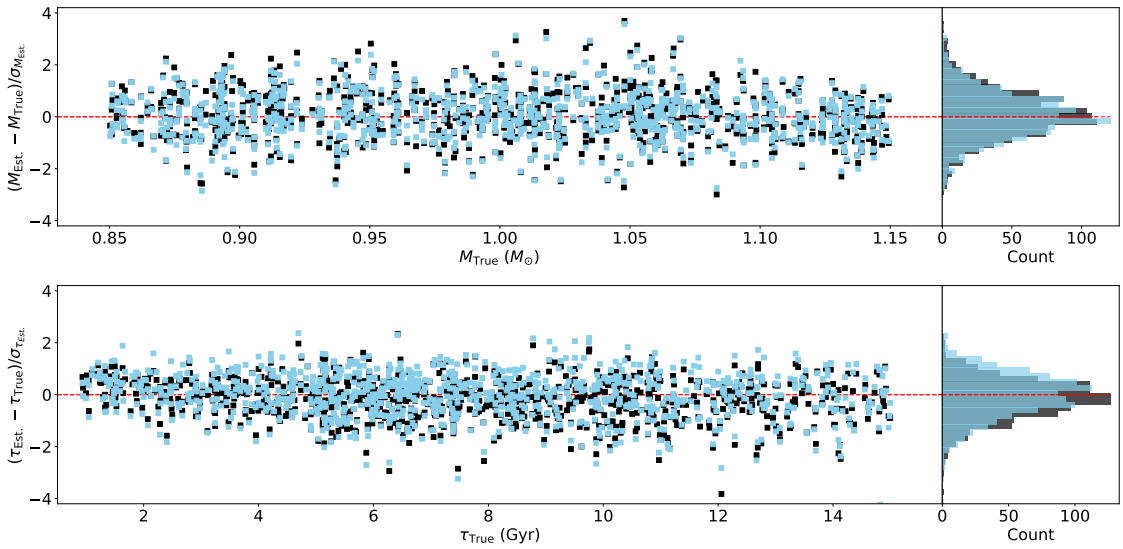

Figure 9. Differences between true and estimated values over estimated uncertainty of 1000 fake stars. Count distributions of offsets are demonstrated on the right side.

REFERENCES

Asplund M., Grevesse N., Sauval A. J., Scott P., 2009, Annual Review of Astronomy and Astrophysics, 47, 481

Bain A., 1873, Mind and body: The theories of their relation. Vol. 4, D. Appleton

Bellinger E. P., Angelou G. C., Hekker S., Basu S., Ball W. H., Guggenberger E., 2016, ApJ, 830, 31

Bishop C. M., 1994

Choi J., Dotter A., Conroy C., Cantiello M., Paxton B., Johnson B. D., 2016, ApJ, 823, 102

Dotter A., 2016, ApJS, 222, 8

Ferguson J. W., Alexander D. R., Allard F., Barman T., Bodnarik J. G., Hauschildt P. H., Heffner-Wong A., Tamanai A., 2005, ApJ, 623, 585

Gardner J. R., Pleiss G., Bindel D., Weinberger K. Q., Wilson A. G., 2018, in Advances in Neural Information Processing Systems.

Hendriks L., Aerts C., 2019, PASP, 131, 108001

Hensman J., Matthews A., Ghahramani Z., 2014, arXiv preprint arXiv:1411.2005

Ho T. K., 1995, in Proceedings of 3rd international conference on document analysis and recognition. pp 278–282

Hon M., Stello D., Yu J., 2018, MNRAS, 476, 3233

Hon M., Bellinger E. P., Hekker S., Stello D., Kuszlewicz J. S., 2020, MNRAS, 499, 2445

James W., 1890, The principles of psychology. New York, NY: Holt & Co

Lyttle A. J., et al., 2021, MNRAS,

Mombarg J. S. G., Van Reeth T., Aerts C., 2021, arXiv e-prints, p. arXiv:2103.13394

Paquette C., Pelletier C., Fontaine G., Michaud G., 1986, ApJS, 61, 177

Paxton B., Bildsten L., Dotter A., Herwig F., Lesaffre P., Timmes F., 2011, The Astrophysical Journal Supplement Series, 192, 3

Paxton B., et al., 2013, The Astrophysical Journal Supplement Series, 208, 4

Paxton B., et al., 2015, The Astrophysical Journal Supplement Series, 220, 15

Paxton B., et al., 2018, ApJS, 234, 34

Paxton B., et al., 2019, ApJS, 243, 10

Reddi S., Kale S., Kumar S., 2018, in International Conference on Learning Representations.

Rendle B. M., et al., 2019, MNRAS, 484, 771

Rogers F. J., Nayfonov A., 2002, ApJ, 576, 1064

Schölkopf B., Smola A., Müller K.-R., 1997, in International conference on artificial neural networks. pp 583–588

Sutskever I., Martens J., Dahl G., Hinton G., 2013, in International conference on machine learning. pp 1139–1147

Thoul A. A., Bahcall J. N., Loeb A., 1994, ApJ, 421, 828

Verma K., Hanasoge S., Bhattacharya J., Antia H. M., Krishnamurthi G., 2016, MNRAS, 461, 4206

Williams C. K., Rasmussen C. E., 1996

Wilson A., Nickisch H., 2015, in International Conference on Machine Learning. pp 1775–1784

Wu Y., et al., 2019, MNRAS, 484, 5315

This paper has been typeset from a TEX/LATEX file prepared by the author.