MNRAS **000**, 1–11 (2020) Preprint 11 February 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 massive 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 in terms of the statistics. 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. Hon et al. (2018) developed a convolutional neural network classifier for solar oscillations in red giants. 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. (**Guy: can you add more relevant papers?**)

A machine-learning algorithm that involves a Gaussian process (GP) measures the similarity between data points (the kernel function) to predict values for unseen points from training data. **more intro about GP**.

The aim of this paper is training GP models to turn a sparse stellar grid into a continuous function and augmenting the grid.

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

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

The paper is organised as follow. We describe the computation of a representative stellar grid in Section 2. We then introduce the underline theory of GP and set up the training for GP in Section 3. Section 4 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 5. 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 6.

2 THEORETICAL STELLAR GRID

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`.

3 GAUSSIAN PROCESS MODEL

GP can be applied as probabilistic models to regression problems. Here we will use the GP model to generalise a grid of stellar models to a continuous and probabilistic function that maps inputs (i.e., initial mass, chemical composition, etc.) to observable quantities (i.e., effective temperature, surface gravity, radius, etc.). We aim to use the GP model as a non-parametric emulator, that is emulating the comparatively slow calls to models of stellar evolution. 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`.

3.1 Gaussian Process Application

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 (or 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, GP's (**Guy, please check " 's "**) may be written as

$$y(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), \mathbf{\Sigma}\right), \tag{2}$$

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 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 covariance matrix is defined by the covariance function or *kernel function* k which has hyperparameters θ and is given by,

$$\mathbf{\Sigma}_{n,m} = k(\mathbf{x}_n, \mathbf{x}_m), \tag{3}$$

where the inputs $\mathbf{x}_i$ are D-dimensional vectors and the output is a scalar covariance.

As a GP is a collection of random variables, where any finite number of which have a joint Gaussian distribution, the joint probability of our data $\mathbf{y}$ is

$$p(\mathbf{y}|\mathbf{X}, \phi, \theta) = \mathcal{N}(m(\mathbf{X}), \mathbf{\Sigma}). \tag{4}$$

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 Gaussian and given by

$$p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{y}_\star\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{X}\\ \mathbf{X}_\star\end{bmatrix}, \begin{bmatrix}\mathbf{\Sigma} & \mathbf{K}_\star\\ \mathbf{K}_\star{}^T & \mathbf{K}_{\star\star}\end{bmatrix}\right), \tag{5}$$

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

$$\mathbf{\Sigma}_{n,m} = k(\mathbf{X}_n, \mathbf{X}_m), \tag{6}$$

which is an $N \times N$ matrix.

$$\mathbf{K}_{\star\, n,m} = k(\mathbf{X}_n, \mathbf{X}_{\star\, m}), \tag{7}$$

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

$$\mathbf{K}_{\star\star\, n,m} = k(\mathbf{X}_{\star\, n}, \mathbf{X}_{\star\, m}), \tag{8}$$

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

$$\mathbf{y}_\star \sim \mathcal{N}(\hat{\mathbf{y}}_\star, \mathbf{C}), \tag{9}$$

where

$$\hat{\mathbf{y}}_\star = m(\mathbf{X}_\star) + \mathbf{K}_\star^T\, \mathbf{\Sigma}^{-1}\, (\mathbf{y} - m(\mathbf{X})), \tag{10}$$

and

$$\mathbf{C} = \mathbf{K}_{\star\star} - \mathbf{K}_\star^T\, \mathbf{\Sigma}^{-1}\, \mathbf{K}_\star. \tag{11}$$

At point we can make predictions on model properties given a grid of stellar models using equation 9. 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 GP Model Inputs and Outputs

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. Moreover, we need another input to describe the evolutionary phase. The GP model hence contents five input demissions.

Regarding the evolution index, stellar age is not ideal because its dynamical range varies track by track. The fractional age is an option. However, we find that global parameters (e.g. effective temperature) sharply change as a function of 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

$$\delta d_n = ((T_{\rm eff,n} - T_{\rm eff,n-1})^2 + (\log g_n - \log g_{n-1})^2))^f\,, \tag{12}$$

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

$$d_n = \sum_{i=0}^{i=n} \delta d_i. \tag{13}$$

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. 12 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. 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)

Thus, the GP model can be described as

$$\text{Outputs} = f(M, EEP, (\text{Fe/H})_{\rm init}, Y_{\rm init}, \alpha_{\rm MLT}). \tag{14}$$

3.3 Training Procedure

3.3.1 Data Selection

We use three types of data for training, validating, and testing GP models. Evolutionary tracks in the primary and additional grids (as described in Table 1) are 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, typical data size for GP 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 for 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

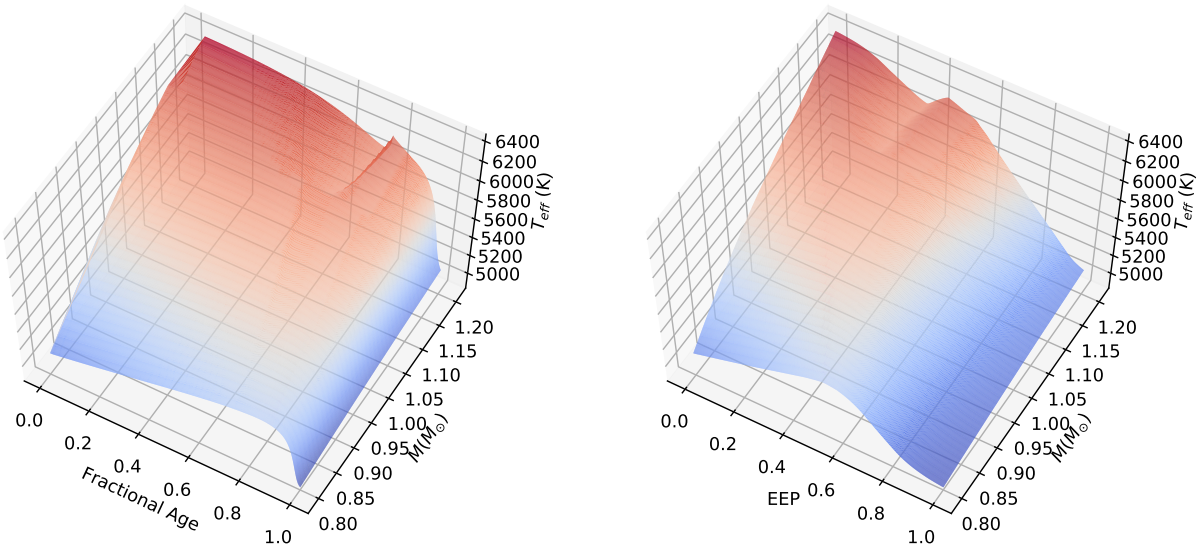

Figure 1. Surface plots of model effective temperature on the mass-fractional age (left) and mass-EEP (right) diagrams. Models in this figure are from the primary grid with fixed initial metallicity ($[\mathrm{Fe/H}]_{\mathrm{init}} = 0.0$), helium fraction ($Y_{\mathrm{init}} = 0.28$) and mixing-length 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.

sparse at the subgiant stage. We test a few methods and find that using the displacement (δd_n) defined in Eq. 12 as the weight of sampling give a relatively uniform data distribution in the parameter space.

3.3.2 *Training and Testing GP Models*

The procedure contents two steps: training and testing. Training is the process that optimises hyperparameters with training dataset and validates GP model with validating dataset. When training is completed, the final GP model is tested by testing dataset to quantify its accuracy. Details of the training method will be described in the following Section.

Here we discuss the method for validating and testing a GP model. We do not use a method which estimates a global error, such like Root Mean Square Error (RMSE). Because we find that GP predictions are not at similar accuracy level for differences evolutionary phases. We illustrate this with an example in Figure 2. As it can be seen that, we train a GP model which maps M and EEP to the effective temperature. The kernel function in the area of $M \geq 1.05\mathrm{M}_\odot$ 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 hence poorly predicted by the GP model. When there is a subregion in which the GP model performs worse than other areas, the error distribution is not Gaussian-like. We examine the error distribution as shown at the bottom of Figure 2. Errors of most data follows a Gaussian distribution but about 10% data form long tails on both sides. Thus, a global error estimator like RMSE is not suitable. For other outputs, surface gravity and radius are similar to the case of effective temperature. The surface metallicity is not significantly affected by the hook, but it has a quick raise 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. This results in a sharp raise of the surface metallicity and the GP models have relatively poor prediction at this phase. The age predictions are also relatively poor for the old low-mass stellar models. This is because the age vary in a big dynamic range (15 - 50 Gyr) in a small fraction of data points. We examine tail feature in error distribution for each output. The data outside 3 times of standard deviation is around 10% (8 - 12% for different outputs). What we need here is a simple method that reflects the general accuracy as well as the worst case. For the majority 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 2, cumulative values at 68%, 95%, and 99.7% are 1.1, 4.9, and 11.1K, which give an EI equals to 17.1K.

3.4 Preliminary Tests for Setting up the Training

For training GP models, we need set up mean function, kernel function, likelihood function, loss function, and optimiser. We do preliminary tests and compare different options for the above elements. These tests are carried out with 2-demission (2D) GP models which map two inputs M and EEP to observable outputs. The 2D GP models can be described as Outputs = $f(M, EEP)$. Training data are selected with fixed $[\mathrm{Fe/H}]_{\mathrm{init}}$ (0.0), Y_{init} (0.28), and α_{MLT} (2.1) from the primary grid. 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 sampling method mentioned in previous section and select 20,000 data points for training, 10,000 for validating, and 10,000 for testing. The GpyTorch Exact GP module is adopted. 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.

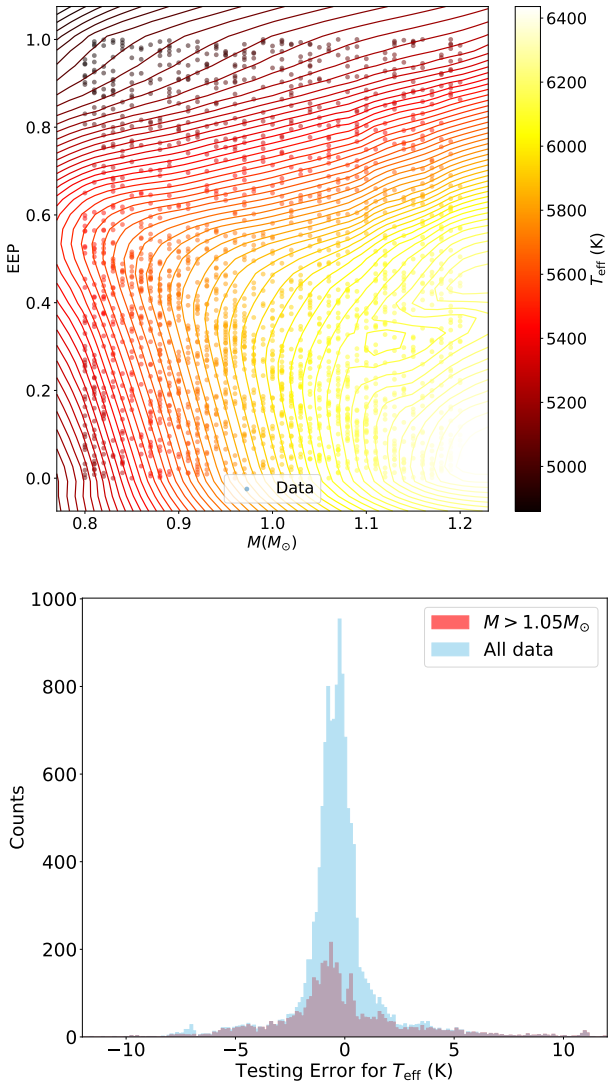

Figure 2. Top: The 2D GP model for $T_{\rm eff}$. Bottom: probability distributions of validating errors of the GP model.

3.4.1 *Mean Function*

We first investigate the mean function. As discussed above, the data distribution is generally smooth but also complex in particular 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 linear mean function leads to a long time for training and a significant increase of the complexity of kernels. For this reason, we apply a Neural Network mean function which is flexible enough to fit either smooth or curved 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?**)

3.4.2 *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

$$p(y|f(x)) = f + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{15}$$

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 can not be a proper description for the stellar grid even if it has small validating or testing errors. 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.

3.4.3 *Optimiser*

Afterwards, we run tests to choose an optimiser. We mainly compare between two 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.

3.4.4 *Early stopping*

We set up an early stopping for tow purpose: avoiding overfitting and terminating the training progress. The early stopping is controlled by the validating EI (introduced in Section 3.3.2). When an optimal solution is found, the validating errors stop decreasing and then increase when overfitting occurs. In practice, we track the validating EI every iteration and terminate training process when EI does not decrease in previous 300 iterations.

3.4.5 Kernel Function

We lastly test kernel functions. We involved four basic kernels in the tests 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 the 2D GP models. Among these kernels, a 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 all features in the training data. However, the spiky function can loss accuracy for off-grid regions when training data are sparse. 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 the function is relatively smooth across the grid. We hence adopt the Mat32 kernel.

3.4.6 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 hence need a strategy for large data sample. We work on this with 3-demission (3D) GP models, which map M, EEP, and $[\mathrm{Fe/H}]_{\mathrm{init}}$ to observable outputs. The 3D GP model can be described as Outputs $= f(M, EEP, [\mathrm{Fe/H}]_{\mathrm{init}})$. We select training data in the primary grid with fixed Y_{init} (0.28), and α_{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_{init} and α_{MLT} but random M and $[\mathrm{Fe/H}]_{\mathrm{init}}$. 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_{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 testing errors for T_{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 limit become critical for the high-demission case. The parameter space exponentially increases with the demission and hence the GP model accuracy inevitable declines. 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_{eff} decreases from 23.5 to 21.6K (1.7/5.0/14.9K at 68/95/99.7%).

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 this as illustrated in Figure 3. We inspect absolute testing errors on the $M - EEP$ diagram. No obvious edge effect is found. We also do a statistical comparison between all testing errors and those around section boundaries ($\pm 0.01 EEP$). As shown in the bottom graph, the density distributions of two samples are very similar and we hence conclude that there is no edge effect. The section scenario is hence adopted in the following study.

Here we summary our set up for the GP model.

- 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

4 TRAINING RESULTS

We train GP models using the method described in Section 3. The training data are evolutioanry tracks in both primary and additional girds as mentioned in Table 1. The role of the additional

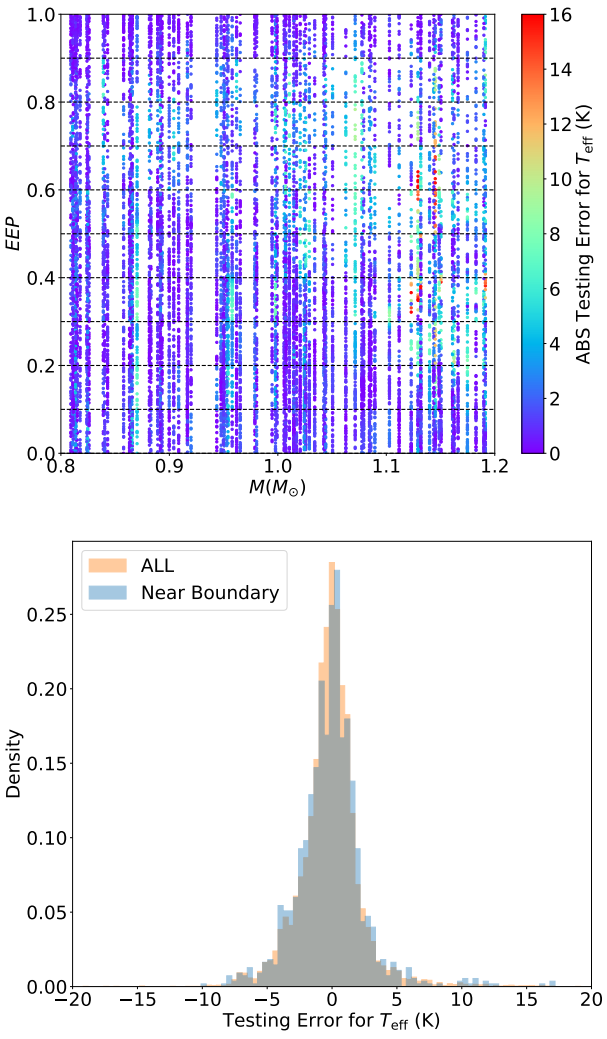

Figure 3. Top: Testing errors of 3D GP model for $T_{\rm eff}$ on the $M - EEP$ diagram. Dashes indicates section boundaries. Bottom: examination of the edge effects of the section scenario. Probability distributions of testing errors of all testing data and those near the boundary (±0.01 EEP) in the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

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 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 a GP model for each output parameter. Each training involves 20,000 training data points and 20,000 validating data points. For the testing dataset, we sample 50, 000 data points in the whole EEP range. Note that we do not use models with $\tau \geq$ 20.0 Gyr, $[\mathrm{Fe/H}]_{\rm surf} \leq$ -0.6 dex, or $T_{\rm eff} \geq 7000K$ in testing dataset.

4.1 Overview of Results

We start with training one Exact GP model for the whole EEP range as a standard. 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 EIs 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. Following analysis are all based on it.

A overview of testing errors can also be seen in Figure 4, 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 $[\mathrm{Fe/H}]_{\rm init}$. The long tails in marginal distributions are similar to what is illustrated in Figure 2: 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.

4.2 Mapping Systematical Uncertainties using another GP model

As shown in Figure 4, systematical uncertainties relate to M, EEP, and $[\mathrm{Fe/H}]_{\rm init}$ but not to $Y_{\rm init}$ or $\alpha_{\rm MLT}$. Thus, a comprehensive model for the systematical uncertainty can be described as a function of M, EEP, and $[\mathrm{Fe/H}]_{\rm init}$. In the M-EEP-$[\mathrm{Fe/H}]_{\rm init}$ 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 $[\mathrm{Fe/H}]_{\rm init}$. 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 ~ 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 5. As it can be seen that, local $\sigma T_{\rm eff}$ values are mostly below ~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 $[\mathrm{Fe/H}]_{\rm init}$) 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

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 ($10^{-3}R_\odot$)	$[{\rm Fe/H}]_{\rm surf}$ (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

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 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 $[{\rm Fe/H}]_{\rm init}$
- Model Outputs: $\sigma_{T_{\rm eff}}$, $\sigma_{\log g}$, σ_R, $\sigma_{[{\rm Fe/H}]_{\rm surf}}$, 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. Final models show good agreement with validating data. For instance, the average validating error for $\sigma_{T_{\rm eff}}$ is only 0.15K. For other output parameters, we summary the results in Table 3. Figure 5 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.

5 AUGMENTING THE MESA GRID

5.1 A GP-Trained Model Set

Now we have GP-Grid models for predicting observable outputs and GP-SYS models for estimate their local systematical uncertainties. In Figure 6, we demonstrate a GP-trained Kiel diagram comparing with the stellar grid. It shows that GP has transformed the sparse model grid into a non-sparse model set. Here we generate a GP-Trained model set to augment the stellar grid. We randomly sample 5,000,000 model data points with a flat distribution on each input demission. We then predict observable outputs with GP-Grid models and systematic uncertainties with GP-SYS models. This GP-trained model set hence includes five fundamental inputs (M, EEP, $[{\rm Fe/H}]_{\rm init}$, $Y_{\rm init}$, and $\alpha_{\rm MLT}$), five outputs, ($T_{\rm eff}$, $\log g$, R, $[{\rm Fe/H}]_{\rm surf}$, and τ), and five systematic uncertainties ($\sigma_{T_{\rm eff}}$, $\sigma_{\log g}$, σ_R, $\sigma_{[{\rm Fe/H}]_{\rm surf}}$, and σ_τ). This model set can be downloaded at `a-place-for-data`.

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
$\sigma_{[{\rm Fe/H}]_{\rm surf}}$ (10^{-3}dex)	0.09
σ_τ (10^{-2} Gyr)	0.2

5.2 Modelling fake stars with GP-trained models

With the GP-trained model set, we model 100 fake stars to examine the accuracy of our method. Fake stars are randomly selected from the off-grid models. We use four observables, i.e., $T_{\rm eff}$, $\log g$, R, and $[{\rm Fe/H}]_{\rm surf}$, as constraints. We apply typical observed uncertainty that is ±50K for $T_{\rm eff}$ (high-resolution spectroscopy), ±0.005dex for $\log g$ (seismology), 3% for R (seismology), and ±0.05dex for $[{\rm Fe/H}]_{\rm surf}$ (high-resolution spectroscopy). To avoid edge effect, fakes stars are selected in the range of $T_{\rm eff}$ = [4700K, 6800K], $\log g$ = [3.7, 4.6], $[{\rm Fe/H}]_{\rm surf}$ = [-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 fit fake stars using the Maximum Likelihood Estimate (MLE) method. Note that the error term in MLE formula contents observed and systematic uncertainties given by GP-SYS models ($\sigma^2 = \sigma^2_{obs} + \sigma^2_{sys}$). We present likelihood distributions of inferred stellar parameters for a representative fake star in Figure 7. The results based on grid modelling are also plotted as comparisons. Observed constraints for this fake star are $T_{\rm eff}$ = 4926±50K, $\log g$ = 4.536±0.005, $[{\rm Fe/H}]_{\rm surf}$ = 0.34±0.05, and R = 0.829±0.025$R_\odot$. True values of fundamental stellar parameters are M = 0.861 $M_\odot$, τ = 10.8 Gyr, $[{\rm Fe/H}]_{\rm init}$ = 0.403, $Y_{\rm init}$ = 0.281, and $\alpha_{\rm MLT}$ = 2.356.

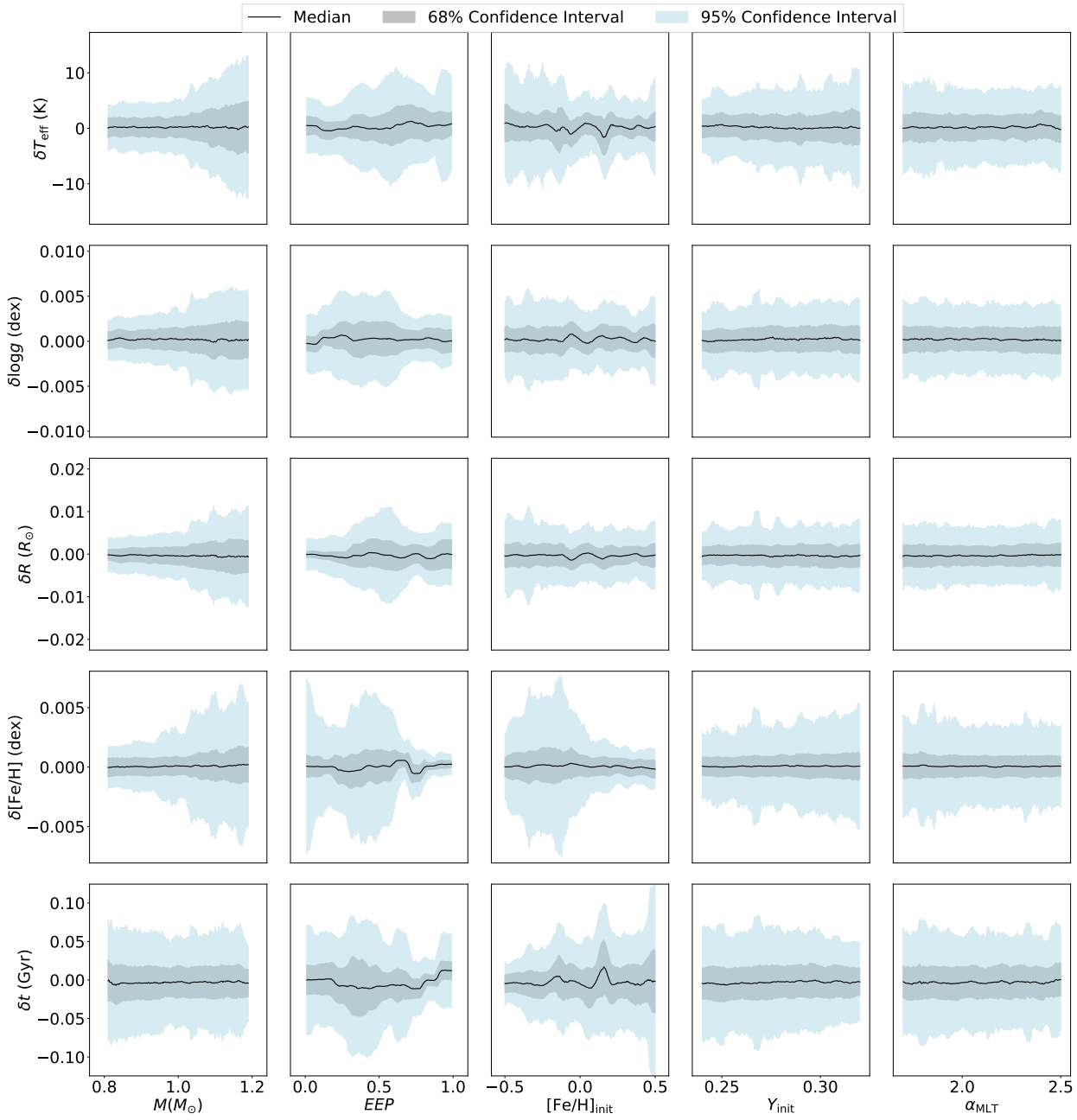

Figure 4. Roll medians and 68/95% confidential intervals of testing errors against GP model inputs. Black solid lines indicate the median value; grey and blue shadowes represent the 68% and 95% confidential interval. Testing errors of $T_{\rm eff}$, log g, and R mainly depend on M and EEP. Metallicity error strongly depends on M, EEP, and $[{\rm Fe/H}]_{\rm init}$, and age error has a significant correlation to EEP and $[{\rm Fe/H}]_{\rm init}$. However, testing errors do not obviously relate to $Y_{\rm init}$ or $\alpha_{\rm MLT}$.

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, while 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 grid. For initial metallicity, initial helium fraction, and the mixing-length parameter, GP makes it possible to statistically estimate these parameters and inferred $[{\rm Fe/H}]_{\rm init}$ and $Y_{\rm init}$ well agree with 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 overcomes the under-sampling issue of a sparse grid and improve the accuracy and precision of estimates.

We now examine the accuracy of inferred mass and age of GP-based modelling. This can be done by comparing the offset (truth - estimated value) with the estimated uncertainty. If the offset is average zero and its scatter consists with typical estimated uncertainty, it would indicate that the offset is just the random error. We make

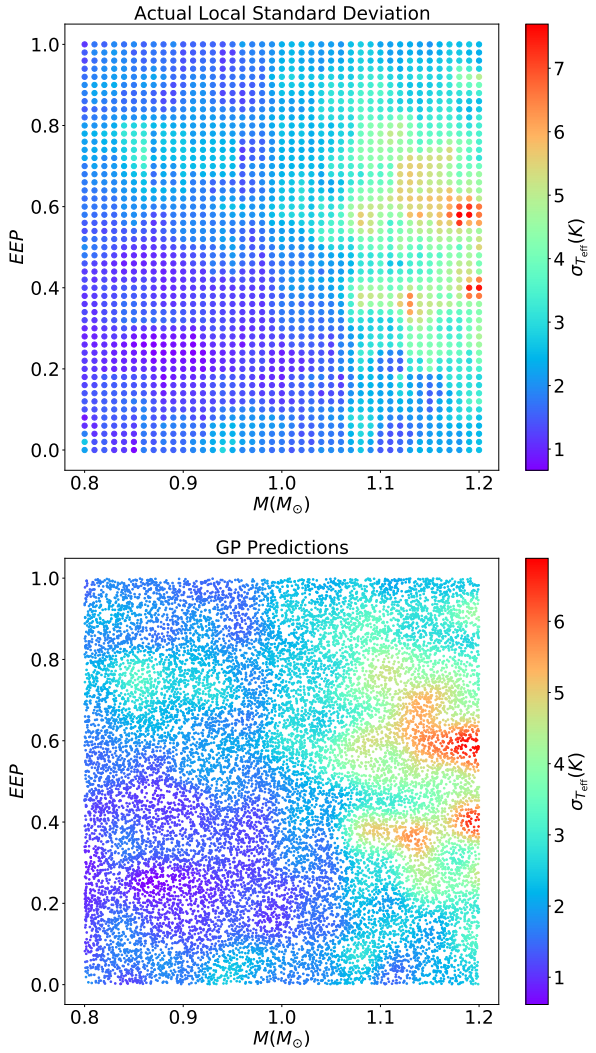

Figure 5. Local systematic uncertainty (1-σ) for $T_{\rm eff}$ on the $M - EEP$ diagram for $[{\rm Fe/H}]_{\rm init}$ = 0.0. The actual distribution is on the top and GP predictions are presented at the bottom.

marginal likelihood distributions for each fake star and measure the 16th, 50th, and 84th percentile values to estimate the mass and the age. The comparison between true and estimated values is demonstrated in Figure 8. Mass offsets are around zero and the scatter range ($0.032{\rm M}_\odot$) is smaller than the typical estimated uncertainty ($0.04{\rm M}_\odot$). Age differences also has a mean value at approximate zero. The scatter is relatively large compared with the case for mass but still reasonably consistent with the estimated uncertainty: offsets of 96 fake stars are within 1-σ (1.6 Gyr) and the rest 4 stars slightly exceeds up to 1.7Gy. The results infer that this GP-based modelling gives reasonable accurate estimates when modelling real stars.

6 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 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.

REFERENCES

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

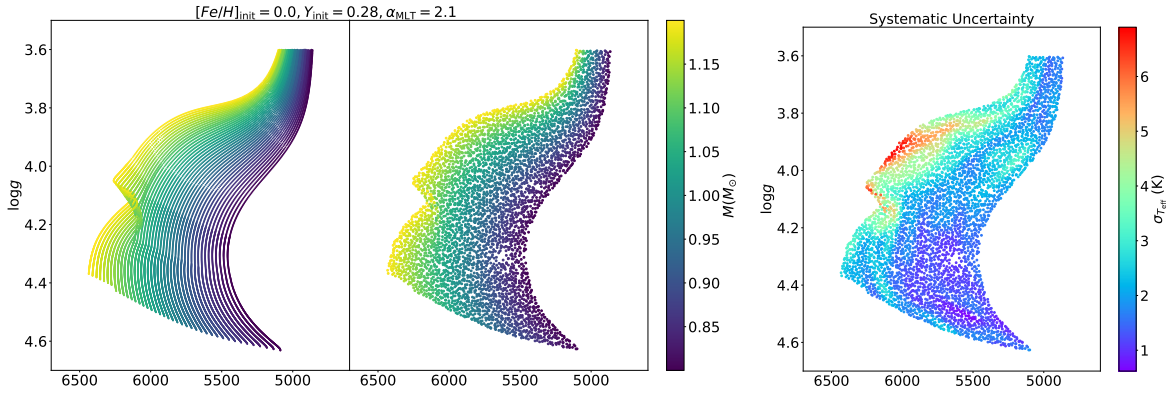

Figure 6. Left: a GP-trained Kiel diagram compared with the stellar grid. Right: GP-predicted systematical uncertainties for all GP-trained models.

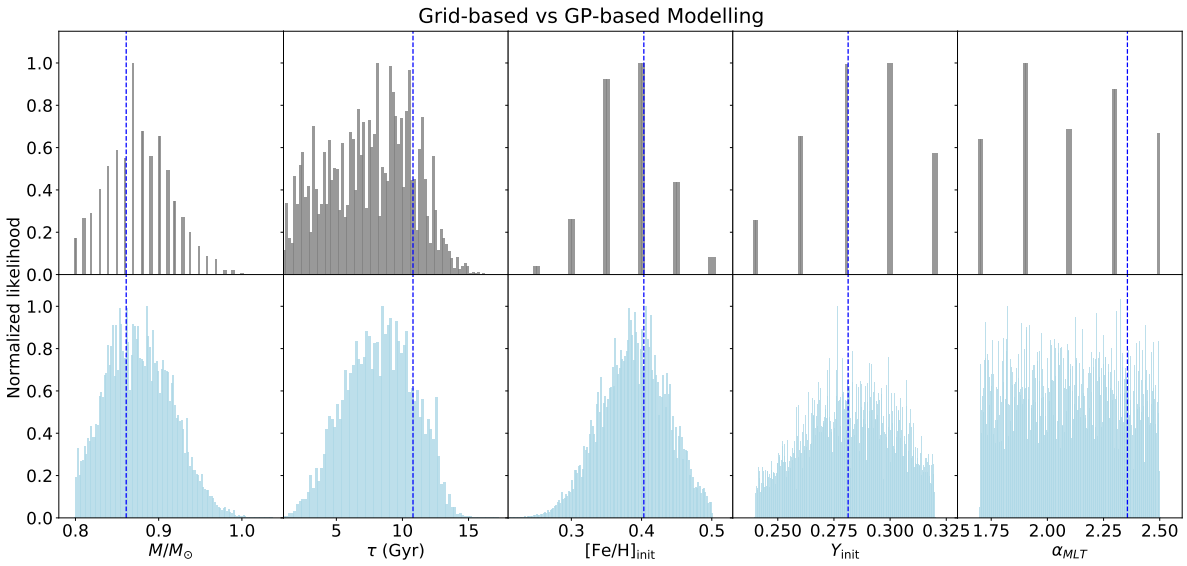

Figure 7. Probability distributions of estimated fundamental parameters from grid-based (Top) and GP-based modelling (Bottom) for a fake star. Observed constraints for this fake star are $T_{\rm eff} = 4926\pm50$K, $\log g = 4.536\pm0.005$, $[{\rm Fe/H}]_{\rm surf} = 0.34\pm0.05$, and $R = 0.829\pm0.025 R_{\odot}$. True values of fundamental parameters are $M = 0.861$ N$_{\odot}$, $\tau = 10.8$ Gyr, $[{\rm Fe/H}]_{\rm init} = 0.403$, $Y_{\rm init} = 0.281$, $\alpha_{\rm MLT} = 2.356$, which are represented by blue dashes.

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

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

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

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

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.

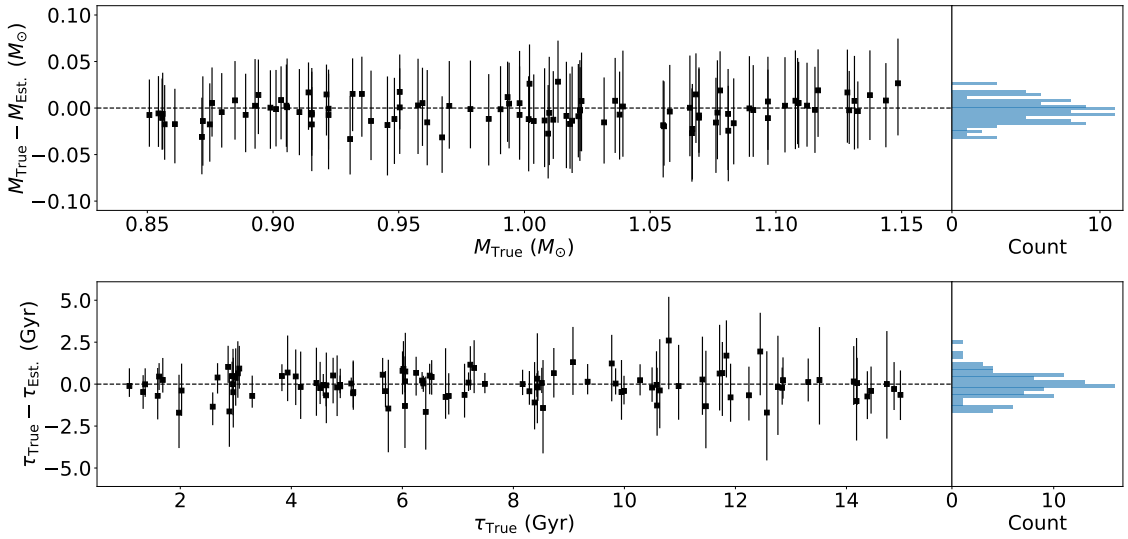

Figure 8. Differences between true and estimated masses (Top) and ages (Bottom) of 100 fake stars. Count distributions of offsets are demonstrated on the right side.