

A New Hybrid Machine Learning Method for Stellar Parameter Inference

SUJAY SHANKAR,¹ MICHAEL A. GULLY-SANTIAGO,¹ AND CAROLINE V. MORLEY¹

¹*Department of Astronomy, The University of Texas at Austin, Austin, TX 78712, USA*

ABSTRACT

The advent of machine learning (ML) is revolutionary to numerous scientific disciplines, with a growing number of examples in astronomical spectroscopic inference, as ML is more powerful than traditional techniques. Here we introduce a hybrid ML (HML) method combining automatic differentiation, interpolation, and Bayesian optimization to infer stellar parameters of stellar spectra. We study T_{eff} , $\log(g)$, and $[\text{Fe}/\text{H}]$, but this method **could be extended** to other **parameters** such as $[\alpha/\text{Fe}]$ (**alpha element abundance**), C/O (**carbon-oxygen ratio**), and f_{sed} (**sedimentation efficiency**). We first use **blase** to semi-empirically recast spectra into sets of **Voigt profiles**. **blase** is run on 1,314 **synthetic** spectra from a **rectilinear subset** of the PHOENIX model grid (T_{eff} : [2300, 12000] K, $\log(g)$: [2, 6], $[\text{Fe}/\text{H}]$: [-0.5, 0], λ : [8038, 12849] Å). For **128,723 detected features**, we map stellar parameters to spectral line parameters using linear interpolation. **This** creates the PHOENIX generator, **enabling parallelized spectral synthesis**. Gaussian Process minimization is used to infer stellar parameters by minimizing a root-mean-square loss **function**. Testing **210 noise-free models** (T_{eff} : [3000, 11000] K, $\log(g)$: [2, 6], $[\text{Fe}/\text{H}]$: [-0.5, 0]), we **find inference errors**: T_{eff} : **93 K**, $\log(g)$: **0.24**, and $[\text{Fe}/\text{H}]$: **0.056** for $T_{\text{eff}} < 7000$ K, and T_{eff} : **347 K**, $\log(g)$: **0.26**, and $[\text{Fe}/\text{H}]$: **0.16** for $T_{\text{eff}} \geq 7000$ K. We also **upload online** an archive of **blase models** of the PHOENIX subset. This proof-of-concept **study** shows that semi-empirical HML is a viable alternative to traditional approaches **in spectroscopic inference**.

1. INTRODUCTION

Stellar spectra are exceedingly rich sources of information about the stars that produce them. Spectra encode fundamental properties such as temperature, surface gravity, and chemical composition via their numerous absorption lines. Extrinsic properties such as the stellar radial velocity or the projected equatorial rotation shift the wavelengths of lines and broaden their widths, respectively. Observing the spectrum alters it again; the resolution, bandwidth, and other properties of the instrument change the fidelity at which we observe the spectrum, and can limit our ability to extract fundamental properties precisely. Observed stellar spectra thus represent extremely complex, data, influenced by multiple parameters, that have gone through multiple transformations before reaching our detectors.

Modern astronomical spectroscopy **takes advantage of intuitive, performant spectral models such as KORG (Wheeler et al. 2023)**, however it would

further benefit from the paradigm of interpretability. The determination of fundamental properties, the creation of Extreme Precision Radial Velocity (EPRV) templates, and the application of composite spectral fitting could all leverage such an innovation. This ambitious aim may be broadly referred to as a “foundational spectroscopy model for astrophysics”, in reference to the same category of models used for large language models (LLMs) in artificial intelligence (AI). There are many challenges with creating such a foundational spectral model for astrophysics. Physical inputs into the spectral modeling process are imperfect, **and simplifying assumptions in stellar (and substellar) atmospheres are necessarily inexact due to factors such as asymmetries and unknown physics.** Computational costs **make** the training of such models **challenging; however, advances have been made with attention-based (Róžański et al. 2023) and transformer-based (Leung & Bovy 2024) models, among others.** Most prevailing solutions have had to choose either a model driven approach (**such as this work**), in which precomputed models are taken as gospel, or a data driven/empirical approach (**such as Lux (Horta et al. 2025)**), in which our knowledge of stellar physics is ig-

nored or treated phenomenologically, depending on the application. Hybrid solutions exist and have been implemented, such as those presented in Leung & Bovy 2019 and Rains et al. 2024. Although a lingering challenge is always how to ideally balance the fusion of model driven and data driven paradigms.

Spectroscopic surveys such as APOGEE (Majewski et al. 2017) use their own in-house pipelines to extract stellar parameters from spectra (ASPCAP (García Pérez et al. 2016) in the case of APOGEE), and these pipelines appear effective. The core assumption for the vast majority of pipelines is that the data to be analyzed is a list of pixels. Analysis pipelines may be closed-source or limited to the scope of the survey itself, causing a sort of siloing effect among surveys. This motivates the development of more universal, instrument-agnostic open-source frameworks that can apply broadly to a range of spectral observations with relatively little tuning.

Multiple efforts have been made in this direction, treating spectra in different ways. The standard practice is to treat the wavelength and flux as simply two arrays and use bespoke algorithms tailored to a small number of well-calibrated spectral lines to obtain fundamental stellar parameters and chemical compositions (e.g. the IGRINS YSO Survey (López-Valdivia et al. 2023) and the IRTF Spectral Library (Rayner et al. 2009)). Other whole-spectrum fitting abstractions decompose model spectra into an eigenbasis, implementing the data-model comparison stage as a tractable regression, such as starfish (Czekala et al. 2015).

Ideally, we want a system that can self-consistently learn, a genuine AI foundational spectral model. Such a system would enable the assignment of accurate stellar parameters, and could yield re-usable interpretable spectral models. **blase**, first presented in Gully-Santiago & Morley 2022, took an important, albeit limited, step in this direction, treating spectra not as a set of pixels or a set of eigenbasis coefficients but as a set of interpretable and traceable spectral lines, specifically Voigt profiles. Each of these approaches has tradeoffs, but one key scientific advantage of **blase** comes from its ability to adapt to new information, while preserving some adherence to physics-based models. This intelligent capability stems from its ability to fit a theoretically unlimited number of nonlinear spectral line parameters with automatic differentiation (autodiff). Autodiff is a technology that tracks transformations made to data using the chain rule, even being able to differentiate control flow transformations involving if-else blocks, for exam-

ple. With the gradient obtained from autodiff’s chain rule, we can optimize model parameters by just going against the direction of the gradient (because we usually want to minimize something such as a loss function, we go in the direction of greatest decrease, which is always the negative of the gradient). Autodiff has been used successfully in other astrophysical contexts such as **exojax** (Kawahara et al. 2022) and **wobble** (Bedell et al. 2019), but the recasting of spectra into sets of inherently nonlinear spectral lines positions **blase** as a unique and promising semi-empirical tool. The original **blase** paper demonstrated the ability to tune spectral lines with autodiff and ‘clone’ spectra, recasting them as ML models defined by interpretable sets of tuned spectral lines, however it was restricted to a pre-selected static synthetic model.

Here we introduce the next logical step in the sequence of expanding interpretable spectroscopic machine learning from operating on a single grid point and towards an entire 3D grid of precomputed spectra. We rebrand this augmentation as **blase3D**.

Ideally this process would be monolithic, with the end-to-end spectral inference code powered by a single autodifferentiable machine learning framework, like PyTorch (Paszke et al. 2019) or JAX (Bradbury et al. 2018). However, here we separate the problem into three pieces, only the first currently being autodifferentiable. In section 2, we scale out the **blase** method to 1,314 precomputed synthetic spectral model clones, yielding a downloadable archive of pre-trained machine learning models with 128,723 unique spectral lines. In section 3, we then fit manifolds mapping stellar parameters to uniformly-derived spectral line parameters using regular grid linear interpolation. Finally, in section 4, we show how reconstructions of the spectra using said manifolds can be used for inferring stellar fundamental properties from spectra. This final step resembles the aims of “atmospheric retrievals”, but in principle should be faster, adaptive, and reusable. An overview of this process is shown in Figure 1.

2. CLONING THE PHOENIX MODEL GRID

2.1. The PHOENIX Subset

For the purposes of this study, we chose the widely-adopted PHOENIX synthetic spectral model grid (Husser et al. 2013). Our approach can be straightforwardly applied to any other model grid in the future, including substellar atmosphere grids such as Sonora (Marley et al. 2021; Karalidi et al. 2021; Morley et al. 2024; Mukherjee et al. 2024), but for now we limit our scope to a **rectilinear** subset of the PHOENIX grid,

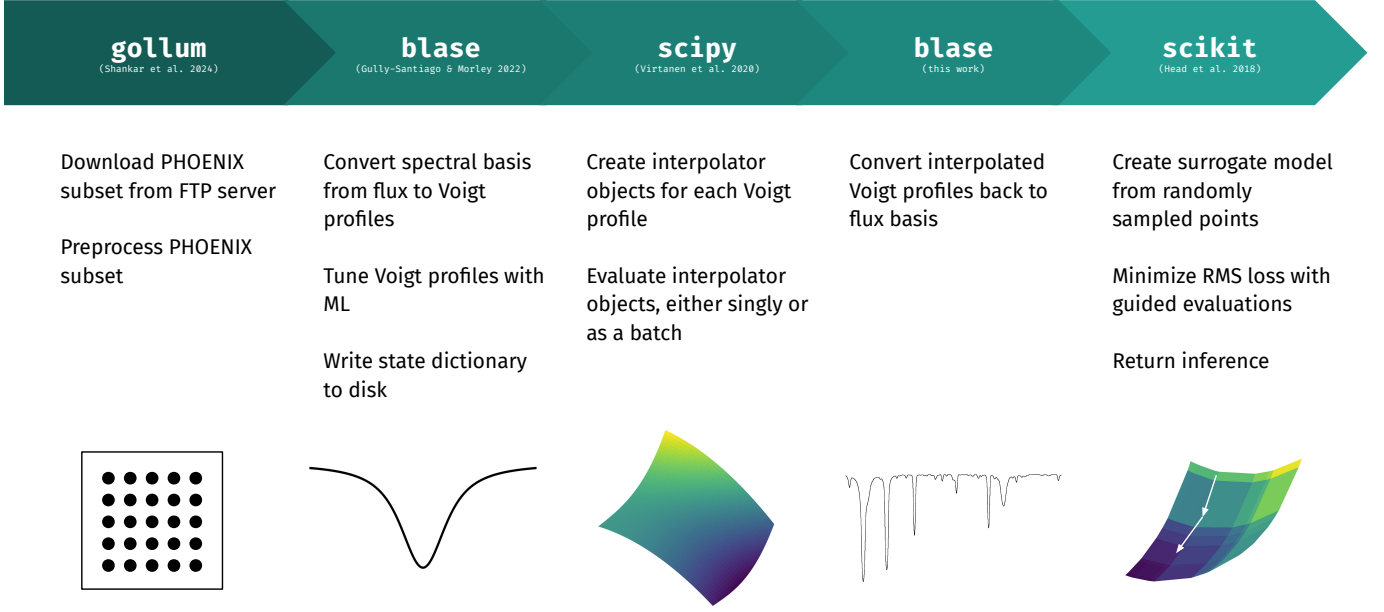


Figure 1. Overview of the process used in this study.

focusing on near solar metallicities and a broad range of effective temperature and surface gravity, with details given in Table 1. This approach is taken due to the computational cost and storage impact of interpolation algorithms. Future versions with more advanced models will be able to cover a broader scope and bypass the limitation of a rectilinear subset.

2.2. Preprocessing with gollum

First, the PHOENIX subset was programmatically retrieved with gollum (Shankar et al. 2024), which downloaded spectra from the PHOENIX FTP server. The spectra were then put through a short preprocessing pipeline similar to that from Gully-Santiago & Morley 2022.

1. *Blackbody Division:* Since the T_{eff} of each spectrum is known, the corresponding blackbody spectrum was divided out.
2. *Continuum Normalization:* The spectra were further normalized by dividing them by a 5th order polynomial continuum fit using a peak-finding filter in order to eliminate curvature that would inhibit line modeling.

Mathematically, we can express the preprocessing as follows:

$$\bar{S} = \frac{S}{BQ_5} \quad (1)$$

where \bar{S} is the preprocessed spectrum, S is the original spectrum, B is the blackbody spectrum, and Q_n is the n^{th} order polynomial continuum fit. Arithmetic operations between arrays are assumed to be elementwise in all following notation.

2.3. Line Identification with blase

The next step was to convert the PHOENIX subset into a physically interpretable intermediate representation: a table of spectral line properties rather than an array of fluxes. We used blase, which models spectral lines as Voigt profiles and tunes the profiles to mimic the original PHOENIX spectrum with back propagation. Back propagation, put simply, is the process of moving in the autodiff gradient field to update ML model parameters (like mentioned earlier, usually against the gradient because we minimize loss functions). Four parameters were optimized: the line center μ , the log-amplitude $\ln(a)$, the Gaussian width σ , and the Lorentzian width γ . The optimization used the Adam optimizer (Kingma & Ba 2017) with a learning rate of 0.05 over 100 epochs. Gully-Santiago & Morley 2022 recommends this setup as the minimum, and for a proof-of-concept implementation, we found it best to leave it as is. In addition, we limited two custom parameters: wing cut to 6,000 and prominence to 0.005. Wing cut (in pixels) is a parameter that determines the extent of the Voigt profile to evaluate, saving computational resources by not evaluating negligible line wings. Prominence (in normalized flux counts) sets a lower limit for the amplitude of detected lines, which saves resources

Parameter	Symbol	Interval	Sampling
Alpha Element Abundance	α	[0]	N/A
Iron Abundance	[Fe/H]	[-0.5, 0]	0.5
Effective Temperature	T_{eff}	[2300, 12000] K	100 K until 7000 K, then 200 K
Surface Gravity	$\log(g)$	[2, 6]	0.5
Wavelength	λ	[8038, 12849] Å	R = 500,000

Table 1. The subset of the PHOENIX grid used in this study. These limits were imposed to reduce the computational cost of the algorithms and to ensure a rectilinear parameter space in order to work with `scipy`’s `RegularGridInterpolator` (Virtanen et al. 2020). The wavelength limits in particular roughly line up with that of the Habitable Zone Planet Finder (HPF) spectrograph (Mahadevan et al. 2012). This subset is comprised of 1,314 individual spectra: 73 T_{eff} values, 9 $\log(g)$ values, and 2 [Fe/H] values.

by disregarding shallow lines, so in our case we disregard lines **with amplitude less than 0.005**. In short, our choices for wing cut and prominence decrease the computational cost of `blase`’s cloning process at the expense of decreasing its accuracy slightly. `blase` uses the pseudo-Voigt approximation, which saves on computational cost while remaining accurate to about 1% (Ida et al. 2000). The pseudo-Voigt approximation uses a weighted average of a Gaussian and Lorentzian as opposed to a convolution. `blase`’s pseudo-Voigt profile implementation uses the following:

$$\tilde{V}_\mu(\lambda) = a[\eta \mathbf{L}(\lambda - \mu'; f) + (1 - \eta) \mathbf{G}(\lambda - \mu'; f)] \quad (2a)$$

$$\eta = \sum_{n=0}^2 \mathbf{u}_n \left(\frac{2\gamma}{f} \right)^{n+1} \quad (2b)$$

$$f = 32 \sum_{n=0}^5 \mathbf{v}_n \left(\sqrt{2 \ln(2)} \sigma \right)^{5-n} (\gamma)^n \quad (2c)$$

$$\mathbf{u} = \begin{bmatrix} 1.36603 \\ -0.47719 \\ 0.11116 \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} 1 \\ 2.69269 \\ 2.42843 \\ 4.47163 \\ 0.07842 \\ 1 \end{bmatrix}$$

where \mathbf{L} and \mathbf{G} are abbreviations for Lorentzian and Gaussian profiles, respectively. Notice that we use μ' instead of μ in the formula. This is because `blase` optionally allows the line center to shift slightly during optimization, and it is this shifted center which is used in computation. The individual Voigt profiles are still indexed by μ for cross-model line identification, explained in the next section. Once optimization was complete, the list of identified lines was saved to a ‘state dictionary’: a common representation for pre-trained machine learning models that can be stored to disk for reuse later. These are stored in the `.pt` file format for each of the 1,314 PHOENIX subset grid points. The total disk space these files take up is 465 MB (382 MB when downloaded as a zip archive). For reference, the

storage space the PHOENIX subset takes up on disk is approximately 8.1 GB. This represents a data compression factor of around 20 just by recasting the spectrum with `blase`. **The state dictionaries are available on Zenodo at <https://zenodo.org/records/11246174> (Shankar 2024).**

3. INTERPOLATING MANIFOLDS

3.1. Cross-Model Line Identification

As previously mentioned, `blase` tunes the line centers of detected lines. This means that from one PHOENIX spectrum to the next, the same line could have a slightly different line center. Since the goal of this study is to interpolate the properties of each line, we needed to identify the presence of a particular line across the PHOENIX subset, associating the same line with every occurrence. We decided to do this by using the line centers μ of the detected lines pre-optimization.

Pre-optimized line centers were required to be equal in order to group lines together. The values of μ pre-optimization are recorded by `blase` to 0.01 Å (picometer) precision; if they instead used the full available floating point precision, this method would likely detect singular (un-groupable) instances of spectral lines across the grid, completely negating the premise of this study. Of course, this method can be confused by species that produce spectral lines at very close wavelengths and other confounding factors, likely resulting in some false positives and negatives. However, we deem it sufficient to approximate this way, as an in-depth treatment would be time-consuming and computationally expensive.

Now with each spectral line indexed by μ , we had four parameters to interpolate: μ' , $\ln(a)$, σ , and γ . Note that since we are dealing with the parameters of Voigt profiles, we can see in Equation 2 that even if the interpolation method is linear, a final spectral reconstruction will vary nonlinearly in flux.

Spectral lines were often only detected in some spectra from the PHOENIX subset. In Figure 2, we show that different grid points show differing counts of detected spectral lines.

The number of detected spectral lines changes as a function of stellar parameter for different reasons. Perhaps for astrophysical reasons: stellar atmospheres genuinely do not produce that line at detectable strength at the given temperature and surface gravity. Alternatively, **blase’s fitting or our line identification assumptions led to artifacts**. Whatever the cause, these missing lines have immediate practical consequences. Rectilinear interpolation schemes break in regions where a line does not appear; you can’t interpolate a quantity that simply doesn’t exist.

To solve this, we artificially populated missing grid points with log-amplitudes of -1000, which retained interpolator stability while nullifying the evaluated line. To prove the feasibility of this computational trick, we can do a simple thought experiment. Moving 1 K in T_{eff} away from a grid point on the boundary of detectability would reduce a line’s amplitude by a factor of over 20,000 (or more precisely, e^{10}). This is an extremely large reduction for an extremely small deviation from a grid point, meaning that to any realistic precision, the line immediately vanishes as soon as **blase no longer detects it**.

Examples of the appearance of missing sections in heatmaps where a line does not appear can be seen in Figure 3 and Figure 4, which show heatmaps of the 11617.66 Å C I and the 10123.66 Å He II lines (identified with NIST (Kramida et al. 2023)) varying over a solar metallicity slice of the PHOENIX subset stellar parameter space. Across the entire PHOENIX subset, **blase** detected 128,723 spectral lines with unique μ . On average, a PHOENIX spectrum has 9,167 detected spectral lines, with the minimum being 252 ($T_{\text{eff}} = 12000$ K, $\log(g) = 6$, $[\text{Fe}/\text{H}] = -0.5$) and the maximum being 34,551 ($T_{\text{eff}} = 2300$ K, $\log(g) = 3$, $[\text{Fe}/\text{H}] = 0$). Every one of these lines can be visualized as a manifold mapping a 3D stellar parameter vector to a 4D spectral line parameter vector, and every one was interpolated to map stellar parameters to spectral line properties.

3.2. Continuously Evaluable Manifolds

For each line, the inputs to the interpolator were the three input parameters T_{eff} , $\log(g)$, and $[\text{Fe}/\text{H}]$, and the output was a list of four parameters, μ' , $\ln(a)$, σ , and γ . For each line, one of these interpolator objects was

created using linear interpolation, and these interpolators were aggregated into a single list, which was then written to disk in the .pkl file storage format. These interpolators generate multiple manifolds representing the following mapping:

$$\begin{bmatrix} T_{\text{eff}} \\ \log(g) \\ [\text{Fe}/\text{H}] \end{bmatrix} \rightarrow \begin{bmatrix} \mu' \\ \ln(a) \\ \sigma \\ \gamma \end{bmatrix} \quad (3)$$

These interpolators could now be evaluated at any point lying within the domain of the PHOENIX subset, turning a discretely sampled PHOENIX subset into a continuously evaluable function, sometimes called a spectral emulator. With the given size of the PHOENIX subset, the interpolator list takes up 13.2 GB of disk space. **This is larger than the original PHOENIX subset by about 63% and than the state dictionary clone archive by a factor of over 34. This is due to the fact that we are now pickling entire function objects that must memorize the data.** Future work with more space-efficient models and clever storage strategies could reduce storage space requirements in exchange for I/O performance. This evaluation is able to reconstruct an existing PHOENIX spectrum or alternatively interpolate a new spectrum within the domain of the PHOENIX subset, so we call this the PHOENIX generator.

The spectral reconstruction process is done by iterating over the PHOENIX generator, evaluating each interpolator at the given coordinates, then reshaping the data into the same format that PyTorch uses for state dictionaries. During the iteration, if the interpolated log-amplitude of the line is less than -100, the line is excluded from the state dictionary. We do this to avoid artifacts in the manifolds due to the artificial population of log-amplitudes of -1000 where grid points were missing.

Finally, the state dictionary is fed into **blase’s SparseLinearEmulator**, which reconstructs the spectrum by constructing a forward model based on the input state dictionary. Any nan values are set to 1 (which we can do because the spectra are all normalized), and the spectral reconstruction is complete. In Figure 5, we see a comparison between a PHOENIX generator solar spectrum and two nearby native PHOENIX grid point spectra at the 10123.66 Å He II spectral line.

We can observe in Figure 6 that our He II line from a reconstructed solar spectrum is not simply a pixel interpolation between the nearest PHOENIX grid point spectra. It is interpolating the spectral line properties

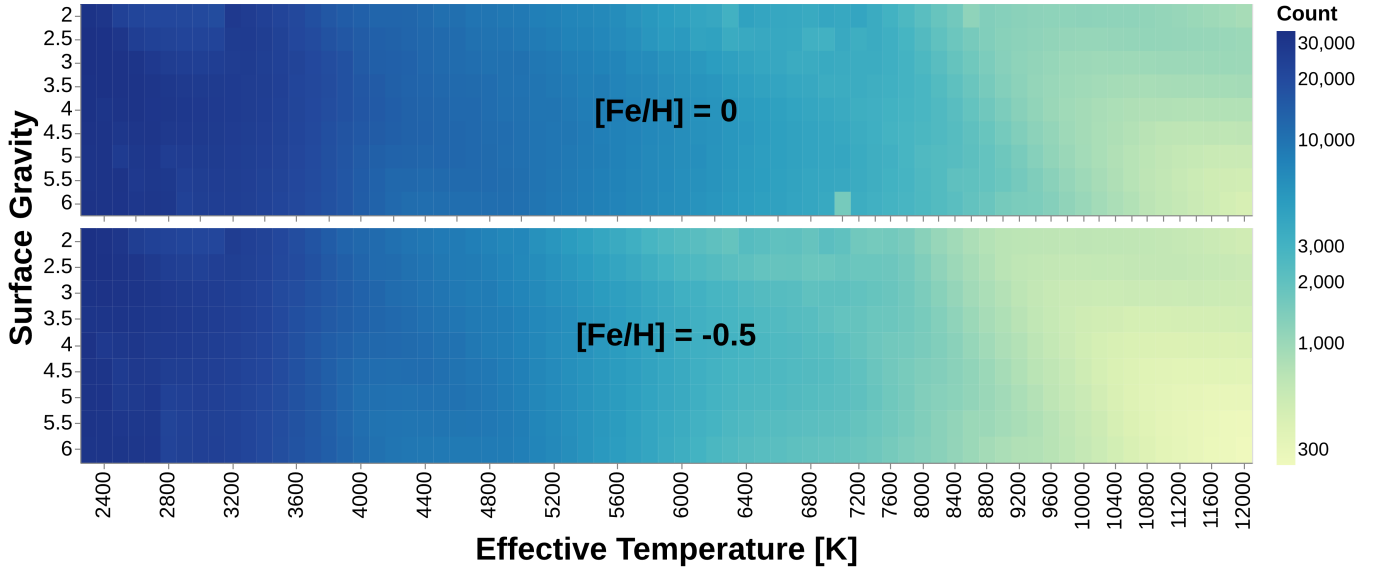


Figure 2. Number of detected spectral lines at each grid point of the PHOENIX subset. We can see that the number of detected lines decreases with increasing T_{eff} . **Remember** that from $T_{\text{eff}} = 7000$ K onward, PHOENIX’s sampling increment changes from 100 K to 200 K. **Due to some combination of computational artifacts from our line identification assumptions or innate behavior of PHOENIX grid spectra themselves, outliers such as the obvious one here at $T_{\text{eff}} = 7000$ K and $\log(g) = 6$ dex appear. Further exploration into why this happens is not in the scope of this study; for inference we are concerned with the lines themselves, not overall grid point behavior.**

390 using hundreds of thousands of manifolds, each repre-
 391 senting a nonlinear parameter in the shape of the spec-
 392 tral line.

3.3. Spectral Reconstruction Time

394 A typical use case for the PHOENIX generator may
 395 be to batch reconstruct spectra from an array of input
 396 stellar parameters. Therefore, there is some motivation
 397 to reduce the computational cost of this procedure to
 398 tractable levels. We evaluate the computational time
 399 needed to use the PHOENIX generator in two distinct
 400 ways. First, for a single input, which would be relevant
 401 in serial applications. Second, for an array of multiple
 402 inputs, which would be relevant in parallel applications.
 403 `scipy`’s `RegularGridInterpolator` API allows for the
 404 passing in of an entire array of input coordinates to be
 405 evaluated at once. However using `blase` to reconstruct
 406 the spectrum from our interpolated state dictionary is
 407 always done serially, leading to what is actually more of
 408 a pseudo-parallel evaluation, but extremely performant
 409 nonetheless.

410 Performance results are shown in Figure 7, and we can
 411 see that the PHOENIX generator’s parallel mode
 412 is much faster than serial mode. While the gen-
 413 erator takes around 15 seconds per spectrum in
 414 serial mode, this drops to 1 second per spectrum
 415 in parallel mode as soon as 30 reconstructions.
 416 The speedup factor between the two methods in-
 417 creases as more spectra are generated, reaching a

419 factor of 20 at 60 reconstructions. Assuming lin-
 420 ear time scaling with wavelength range, accord-
 421 ing to KORG benchmarks (Wheeler et al. 2023),
 422 which generate spectra over a 500 Å wavelength
 423 range, the PHOENIX generator in serial mode
 424 is approximately as fast as KORG and ~30 times
 425 faster than Turbospectrum (Gerber et al. 2023).

426 The computer used for this test has specifica-
 427 tions shown in Table 2 (note that spectral recon-
 428 struction does not currently leverage the GPU):

CPU	AMD EPYC 7513 (32c/64t, 3.65 GHz boost)
RAM	256 GB
GPU	Nvidia A100 [40GB] ($\times 2$)

Table 2. Specifications of the machine used for cloning and reconstruction testing. This machine is owned by Caroline Morley.

4. BAYESIAN INFERENCE AND TESTING

4.1. Gaussian Process Minimization

434 The goals of this study’s inference algorithm
 435 are to infer the stellar parameters T_{eff} , $\log(g)$, and
 436 $[\text{Fe}/\text{H}]$ by comparing a spectrum with our recon-
 437 structions and solving the optimization problem.
 438 We elected to use Bayesian optimization for this com-
 439 ponent, specifically the `gp_minimize` function from the

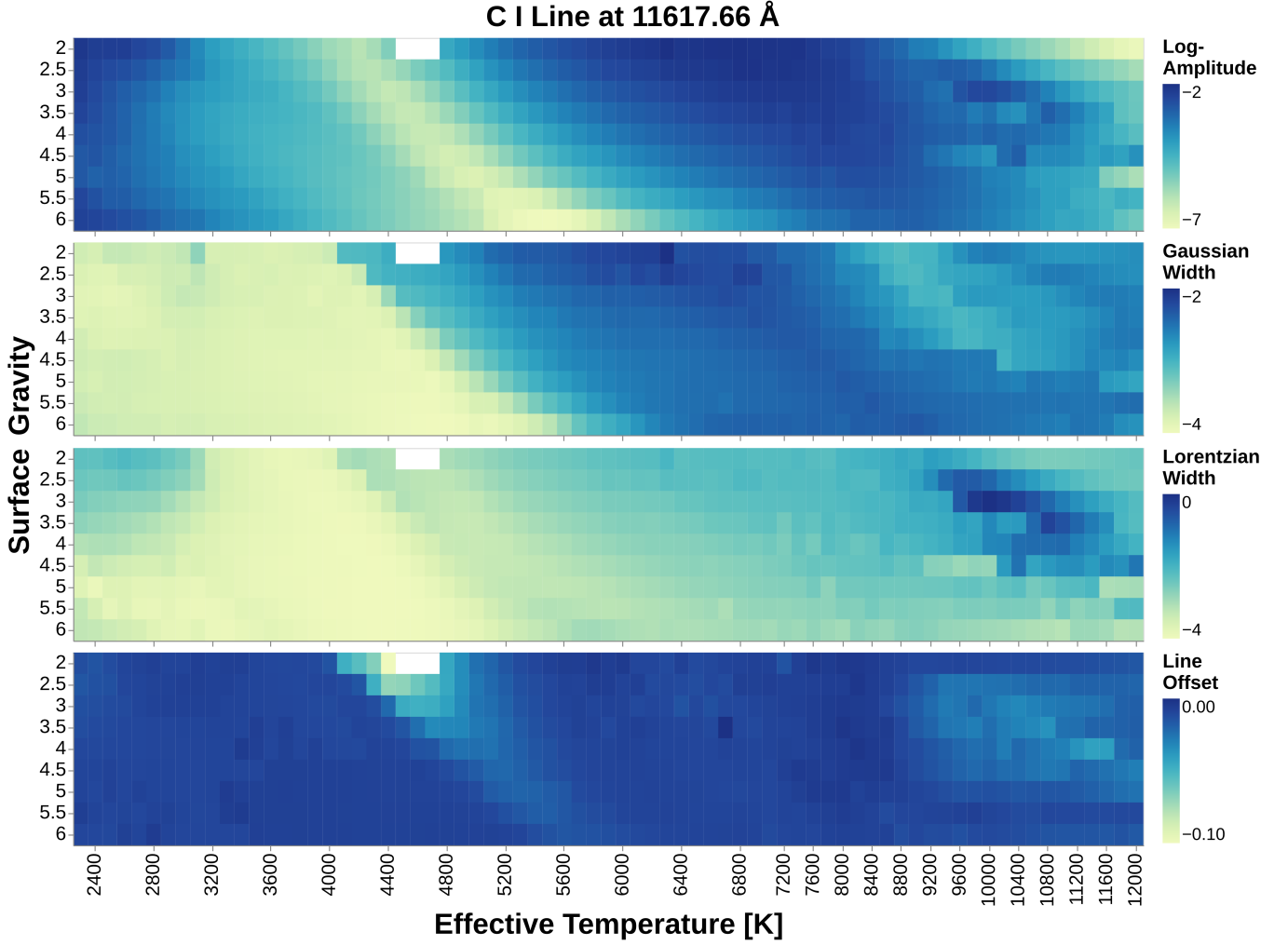


Figure 3. Heatmaps showing how $\ln(a)$, σ , γ , and $\mu' - \mu$ vary over the PHOENIX subset slice at solar metallicity of a C I spectral line, notable for being detected at nearly all subset grid points. Notice the missing chunk in the top left; this could perhaps be due to the line center shifting to the point that blaze identified the chunk as a different line entirely (notice the line center offset increasing in magnitude approaching the chunk border). Since blaze did not detect a spectral line here, we artificially populate those points with lines that have $\ln(a) = -1000$ during interpolation.

scikit-optimize library (Head et al. 2018). This algorithm uses a Gaussian Process to model the objective function, which in this case was the RMS (Root-Mean-Square) loss between the interpolated spectrum M and the true spectrum D , defined as:

$$\mathcal{L} = \langle (M - D)^2 \rangle^{1/2} \quad (4)$$

The optimizer was configured to first run 100 random evaluations to seed the surrogate model, then run 20 more evaluations now guided by the surrogate model. A random evaluation is simply a randomly generated tuple of stellar parameters within the parameter space of the PHOENIX generator, which is then evaluated with the generator to get a reconstruction and then compared with the inference target to retrieve its RMS error. The

surrogate model is the approximation to the true objective function that the optimizer constructs based on the set of random evaluations (in this case defined by a set of 100 points rather than a fully continuous, infinitely resolved function). Guided evaluations are different, they are not randomly generated tuples of stellar parameters. They use the surrogate model in order to select new test points for the minimum of the objective function. This totals to 120 evaluations, 100 to create the surrogate model, then 20 guided evaluations, which was deemed sufficient for this study. Fine-tuning these numbers is possible, but simply not warranted for a proof-of-concept method. Tweaking these numbers could lead to a more accurate inference, however it would incur additional computational cost as

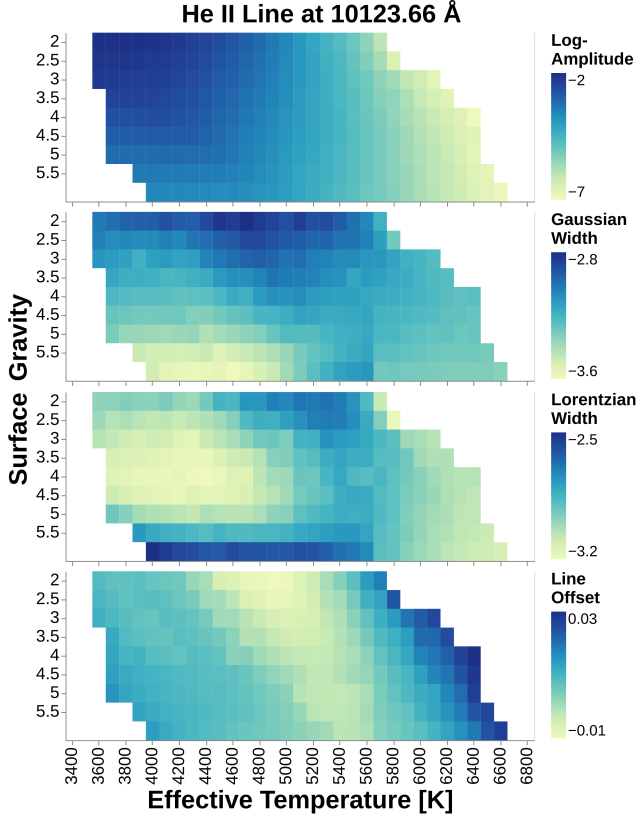


Figure 4. Heatmaps showing how $\ln(a)$, σ , γ , and $\mu' - \mu$ vary over the PHOENIX subset slice at solar metallicity of a He II spectral line. `blase` detects this line at a select chunk of grid points, leading to data only existing between T_{eff} 3600 K and 6600 K.

well. One inference takes on average 200 seconds, using the computer with specifications in Table 3. Inference algorithms that output more information and produce posterior distributions such as `starfish` (Czekala et al. 2015) or `MINESweeper` (Cargile et al. 2020) take on the order of an hour to complete an inference on supercomputers. `astroNN` (Leung & Bovy 2019) is a more direct comparison, and represents an extremely powerful inference tool that performs inference thousands of times faster thanks to its neural network architecture.

CPU	AMD Ryzen 9 7950X3D (16c/32t, 5.7 GHz boost)
RAM	32 GB
GPU	Nvidia GeForce RTX 4080 SUPER [16GB]

Table 3. Specifications of the machine used for general testing and for generating visualizations. This machine is owned by Sujay Shankar.

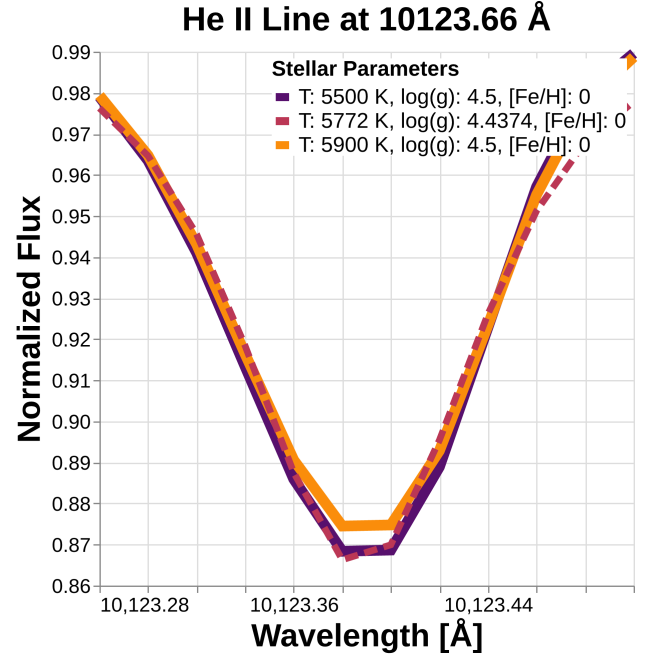


Figure 5. Plot of the 10123.66 Å He II spectral line shown at PHOENIX grid points in the neighborhood of the Sun’s stellar parameters (solid lines), as well as a reconstruction at solar parameters with the PHOENIX generator (dotted line).

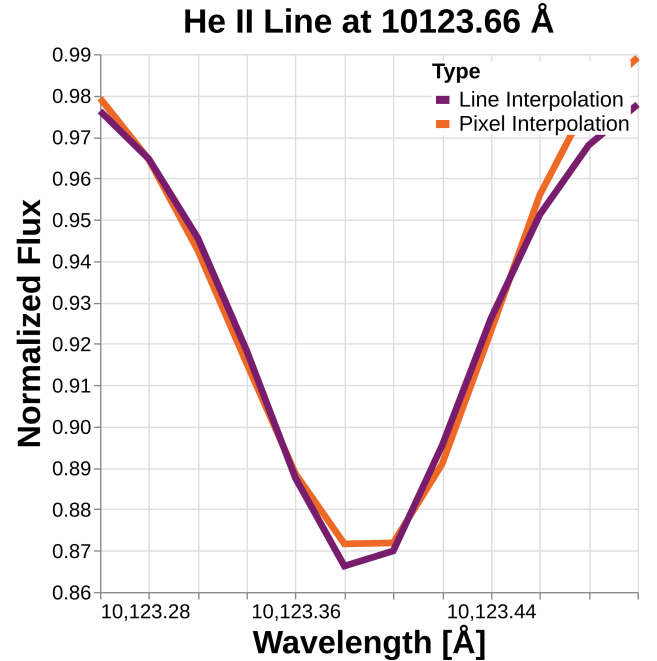


Figure 6. Comparison between the PHOENIX generator’s line interpolation and a standard pixel interpolation of the 10123.66 Å He II spectral line. Notice that line interpolation results in more symmetrical line wings due to the use of Voigt profiles.

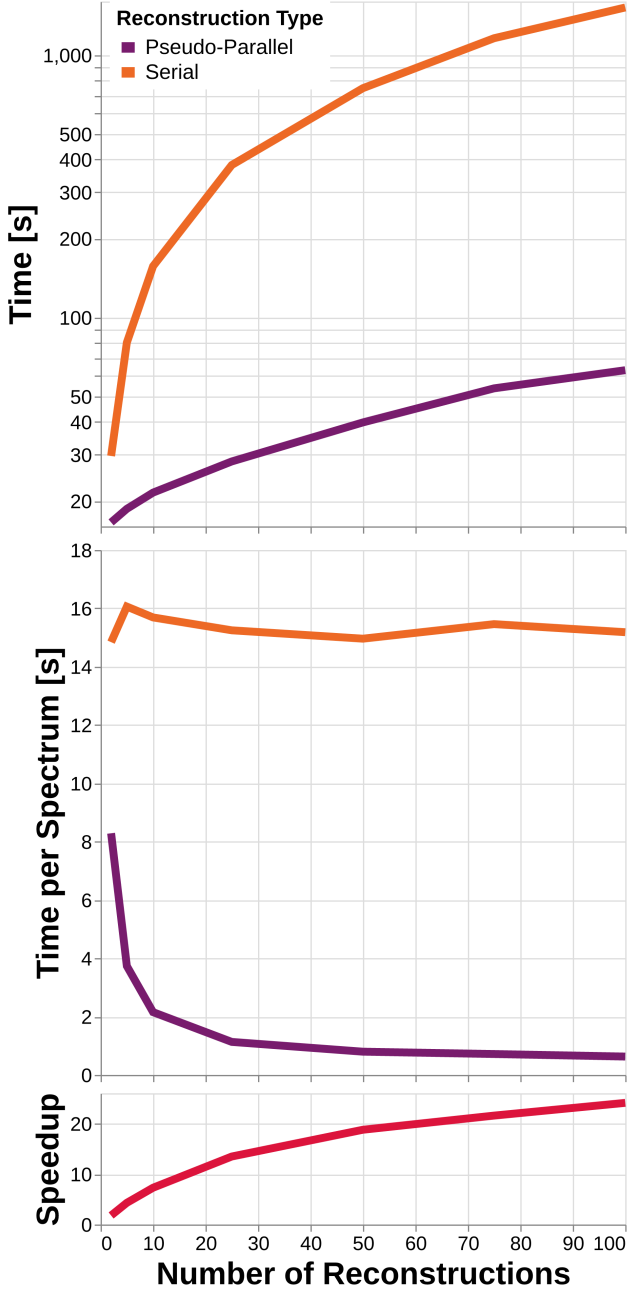


Figure 7. Plots showing the time taken to reconstruct varying numbers of spectra using the PHOENIX generator (lower is better). We can see that the time taken per spectrum for the serial implementation hovers around 15 seconds within run-to-run variance, while the pseudo-parallel implementation continually decreases in time taken per spectrum as the number of inputs increases. The speedup factor (higher is better) increases as more spectra are generated, which is also a desirable outcome.

4.2. Inference Accuracy

To test the performance of the inference algorithm, we used the PHOENIX subset itself. This is possible

for two reasons: first, the PHOENIX generator has not truly memorized the PHOENIX subset due to its Voigt profile basis and the ignoring of weaker features, and secondly, random samples within the parameter space are extremely unlikely to be PHOENIX grid points themselves, avoiding any overlap.

We know that in typical observational settings, a coarse estimate for T_{eff} tends to be fairly well-constrained from ancillary data such as photometric color index. So when testing the inference algorithm, we limited its search space to only include T_{eff} which lay within 500 K of the true value on either side. $\log(g)$ and $[\text{Fe}/\text{H}]$ were allowed to vary freely. The test sample consisted of 21 T_{eff} values: [3000, 11000] K in steps of 400 K, 5 $\log(g)$ values: [2, 6] in steps of 0.5, and 2 $[\text{Fe}/\text{H}]$ values: [-0.5, 0] in steps of 0.5, totaling 210 unique spectra in the test set. Two test passes were run, meaning that 420 inference tests were run in total.

In Figure 8, we show the residuals of the inferred stellar parameters with respect to the true values. We see that there is a distinction between the algorithm’s behavior above and below $T_{\text{eff}} = 7000$ K, so we define low and high temperature regimes accordingly. We know that as T_{eff} increases, we detect less lines with blase, as shown in Figure 2, and lines tend to spread out, possibly resulting in artifacts from our wing cut approximation. Any number of physical or computational factors could explain this behavior, although the apparent sharpness of the transition leads us to postulate that it could be due to PHOENIX grid T_{eff} sampling changing from 100 K to 200 K at this point.

For $T_{\text{eff}} < 7000$ K, the inferred T_{eff} differed from the true value by an average of 93 K, $\log(g)$ 0.24, and $[\text{Fe}/\text{H}]$ 0.056 (standard deviations of 139 K, 0.40, and 0.15, respectively). For $T_{\text{eff}} \geq 7000$ K, the inferred T_{eff} differed from the true value by an average of 347 K, $\log(g)$ 0.26, and $[\text{Fe}/\text{H}]$ 0.16 (standard deviations of 293 K, 0.32, and 0.23, respectively). We see that blase3D is more effective and generally more self-consistent when predicting stellar parameters of lower temperature spectra.

In the low-temperature regime, our median T_{eff} residual is about 10 K, about half that of MINESweeper (Cargile et al. 2020) and astroNN (Leung & Bovy 2019), but about ten times larger than ASPCAP (García Pérez et al. 2016). Median $\log(g)$ and $[\text{Fe}/\text{H}]$ residuals are 0 dex,

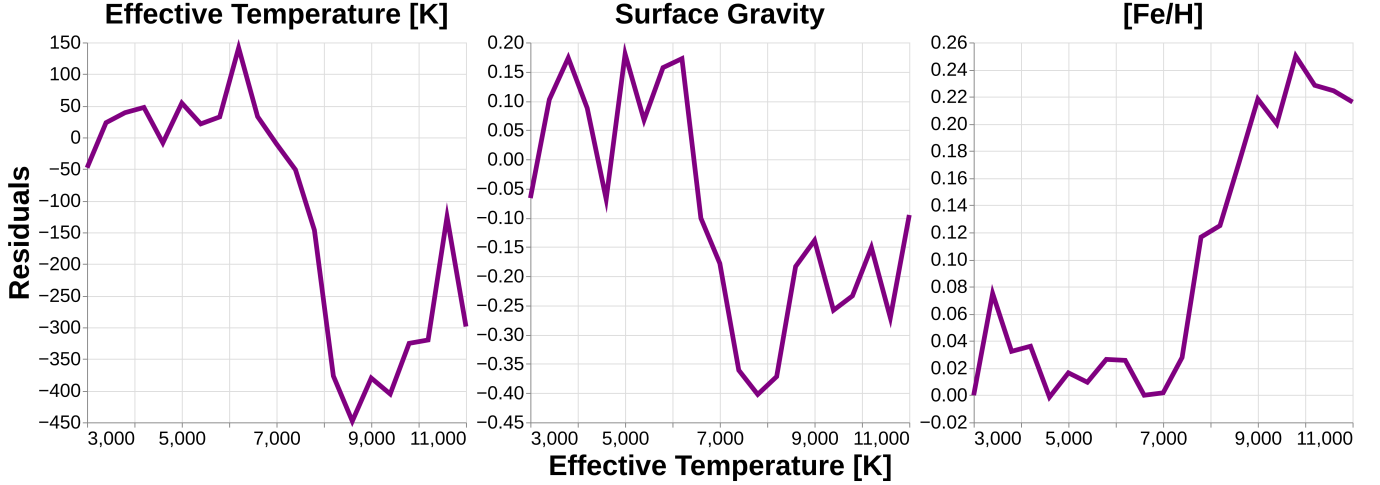


Figure 8. Residual plots for the GP minimization inference algorithm, all plotted with respect to T_{eff} (values averaged across $\log(g)$ and $[\text{Fe}/\text{H}]$). Notice the interesting change in behavior of all residuals at around $T_{\text{eff}} = 7000$ K.

which are comparable to all three aforementioned methods, however `blase3D` is overall less consistent as the standard deviation of the residuals are about an order of magnitude larger. At high temperatures, the median T_{eff} residual changes drastically to about -434 K, which is significantly worse than the other methods. $\log(g)$ becomes a few times less accurate, and both $\log(g)$ and $[\text{Fe}/\text{H}]$ have larger standard deviations, all leading to large inconsistencies and worse performance in this temperature regime. `blase3D` at this stage does not quite reach the level of contemporary methods that can deal with more realistic data and cover much larger parameter spaces, but the fact that, in its best case, it can perform to within an order of magnitude using a drastically simpler implementation, sparks promise for future improvements.

5. DISCUSSION

5.1. Scientific Applications

This study’s paradigm of spectral inference can enable scientists to adopt self-consistent model grids and analyze their spectral line behavior in an interpretable fashion, which has been fairly uncommon practice thus far. Notably, this system tracks the shifting of spectral lines as a function of stellar parameters, which traditionally has been uncommon for algorithms to assess systematically. We balance rigidity in the **Voigt profile basis** and flexibility in **line parameter tunability, resulting in an interpretable system**.

To summarize, we bring to scientists the ability to understand exactly what spectral properties we consider mathematically and how we expect those properties to

behave on an incredibly detailed level (the 4 line parameters of all spectral lines and their corresponding interpolating manifolds). We treat the **PHOENIX** precomputed model grid as the ground truth, constructing the **PHOENIX** generator that then interfaces with our inference algorithm.

5.2. Technical Considerations

To reiterate, the manifold fitting steps are not end-to-end autodifferentiable. As seen in Figure 1, these steps rely on `scipy`, which is not equipped with autodiff. Without autodifferentiability, the ability to “machine learn” is significantly reduced compared to a hypothetical monolithic JAX or PyTorch system. We see three reasons for developing a non-autodifferentiable system. First, the familiar `scipy`-based system will serve as an easy entry point for most practitioners who are unfamiliar with autodiff, and may still benefit from and modify the code without expert ML knowledge. Second, this non-autodiff version serves as an **initial** benchmark against which an inevitable autodifferentiable version of **blase** may be compared. Finally, the inventory of familiar interpolation algorithms have not yet been ported to PyTorch or JAX, since machine learning or Gaussian Process fitting schemes are generally preferred within the ML community.

The interpolation scheme presented here represents a proof of concept, showing that leveraging the mapping between synthetic spectral lines and their inputs can yield a semi-empirical basis for data-model comparisons. There are numerous design considerations that could be improved upon with future work. These include but are not limited to the following:

- *Limited PHOENIX Subset:* The PHOENIX subset used in this study did not include the full PHOENIX grid, which expands the $[\text{Fe}/\text{H}]$ range to $[-4.0, 1.0]$ dex and the $\log(g)$ range to $[0, 6]$, and also includes the alpha element abundance parameter, which we elected to fix at 0 for this study. In addition to the actual stellar parameters, we also took a subset of the PHOENIX wavelength range, with the full $[500, 55000]$ Å wavelength range also being left to future work. **With this**, users would be able to fit a greater variety of stellar spectra in many different wavelength regimes.

- *Strict Wavelength Range:* Currently, the generator only supports inference on spectra whose wavelength limits are either equal to it, or encompass that of the generator and have been truncated to match. However, when the spectrum in question has a smaller wavelength range than the generator, currently there is no functionality to truncate the generator. This would require externally indexing the generator's individual interpolators by line center position and selectively evaluating those to eliminate wasteful computation. This takes burden off the users to truncate their data to the PHOENIX generators, making use simpler.

- *Single Model Grid:* The PHOENIX grid is not the only model grid of synthetic spectra available, and it does not apply to all types of stars. Future work would extend the reach of this study's algorithm to encompass other model grids such as the Sonora series of substellar models (Marley et al. 2021; Karalidi et al. 2021; Morley et al. 2024; Mukherjee et al. 2024), ATLAS (Kurucz 2005), and coolTLUSTY (Lacy & Burrows 2023), reaching practitioners studying various types of stars and spectra. **Future blase versions will** be able to have an option for the user to input which model grid they would like to base the inference on, and to get even more advanced, perhaps even have the ability to intelligently determine which model grid to use automatically.

- *Memorization vs. Generalization:* The current design of the algorithm constructs manifolds using interpolation. This means that performance is good at points close to PHOENIX subset grid points, but is highly dependent on the type of interpolation used. As interpolators require memorization of the data, advanced interpolation becomes extremely expensive in terms of disk utilization. Future work would involve constructing

manifolds using **more generalizable ML methods such as lasso or ridge regression**, which would allow for much better generalization, high speed, and lower disk utilization at the expense of some accuracy.

- *Extrinsic Absence:* The current design of our algorithm does not account for extrinsic parameters that modify the appearance of spectra such as rotational broadening and Doppler shifting. Future work would need to develop ways to tune these extrinsic parameters alongside stellar parameters.

- *Framework Overhead:* As this algorithm is currently more proof of concept than practical, it uses convenience functions from various libraries, which naturally introduces some level of overhead and leaves performance on the table. Future work would involve writing custom functions expressly designed for **blase**, most likely a complete rewrite of the library from the ground up. This has the potential to greatly increase the speed of this algorithm, depending on how much overhead is avoided with a bespoke implementation.

- *Pseudo-Interpretability:* Our algorithm boasts interpretability by considering spectral lines as the objects of interest as opposed to the rather uninterpretable flux values of other approaches. However, this is only a step in the direction of interpretability. True interpretability would decompose a spectrum not into a set of spectral lines, but into a set of species component spectra, which requires a much more advanced understanding of different species and their behavior, as well as direct access to a radiative transfer code as opposed to an off-the-shelf model grid. This approach would also extend the inference from just stellar parameters defined by a grid to any set of parameters accounted for in the radiative transfer model, down to specific species abundances. So while we were able to identify the spectral lines used in our figures, it is not necessarily valuable to try to identify all 128,723 lines that we identify as unique with our algorithm. **blase** is *agnostic* to the identity of the line that it is optimizing. We study these lines as **blase** sees them (i.e. their four shape parameters), because for the purposes of this study, that is the only information that is useful. Having more interpretability would let scientists actually study certain species and their spectral lines.

- *The Continuum Black Box:* Continuum normalization is a process that is not yet completely un-

derstood, and is currently done as a preprocessing step with a fairly simple algorithm. Future work would dive deeper into the science of continua and develop more advanced methods that can discern continua with greater accuracy and less modeling restrictions. This would increase accuracy for end users.

- *One Voigt Fits All*: The current assumption of **blase** is that every spectral line is a Voigt profile. This assumption is largely true, but there are situations where that is simply not enough. Future studies need to account for more advanced spectral line profiles and procedures to deal with phenomena such as ro-vibrational bands. This would increase accuracy for end users.

6. CONCLUSION

In this study, we have presented a proof-of-concept algorithm that **deals with** a subset of the PHOENIX spectral model grid (T_{eff} : [2300, 12000] K, $\log(g)$: [2, 6] dex, [Fe/H]: [-0.5, 0] dex, λ : [8038, 12849] Å). We create state dictionaries for all spectra in the PHOENIX subset **using blase, recasting spectra into lists of Voigt profiles, and** lossily compress the **spectral data by** a factor of around 20. They are **available on** Zenodo at <https://zenodo.org/records/11246174> (Shankar 2024).

We create the PHOENIX generator and implement a performant spectral reconstruction algorithm, enabling anyone to create reconstructions of PHOENIX spectra with continuously valued stellar parameters.

We **then** introduce and test our **GP minimization algorithm to infer stellar parameters. Two test passes on 210 noise-free synthetic models** (T_{eff} : [3000, 11000] K, $\log(g)$: [2, 6] dex, [Fe/H]: [-0.5, 0] dex) **yield** average absolute deviations from true values of **93 K** in T_{eff} , **0.24 dex** in $\log(g)$, and **0.056 dex** in [Fe/H] **in low-temperature** ($T_{\text{eff}} < 7000$) **regimes. In high-temperature regimes** ($T_{\text{eff}} \geq 7000$), **the errors increase to 347 K** in T_{eff} , **0.26 dex** in $\log(g)$, **and 0.16 dex** in [Fe/H].

In its current state, our algorithm requires that the spectra not contain noticeable Doppler shifting, rotational broadening, **noise**, or other confounding factors. The methods discussed here represent a step down a road not traveled in spectral inference, and have the potential to become more advanced in the future by fully utilizing the strengths of physics-informed machine learning.

All figures in this paper are publicly available on GitHub at <https://github.com/Sujay-Shankar/blase3D> **under paper/paper2.**

756

This material is based on work supported by the National Aeronautics and Space Administration under grant No. 80NSSC21K0650 for the NNH20ZDA001N-ADAP:D.2 program. C.V.M. acknowledges support from the Alfred P. Sloan Foundation under grant number FG-2021-16592 and support from the National Science Foundation under grant number 1910969.

Software: **altair** (VanderPlas et al. 2018; Satyanarayan et al. 2017), **astropy** (Astropy Collaboration et al. 2013, 2018, 2022), **blasé/blase** (Gully-Santiago & Morley 2022), **CUDA** (NVIDIA et al. 2020), **gollum** (Shankar et al. 2024), **matplotlib** (Hunter 2007), **numpy** (Harris et al. 2020), **pandas** (pandas development team 2020; Wes McKinney 2010), **Python** (Van Rossum & Drake 2009), **PyTorch/torch** (Paszke et al. 2019), **scikit-optimize/skopt** (Head et al. 2018), **scipy** (Virtanen et al. 2020), **tqdm** (da Costa-Luis 2019), **vegafusion** (Kruchten et al. 2022),

REFERENCES

- 775 Astropy Collaboration, Robitaille, T. P., Tollerud, E. J.,
 776 et al. 2013, *A&A*, 558, A33,
 777 doi: [10.1051/0004-6361/201322068](https://doi.org/10.1051/0004-6361/201322068)
- 778 Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M.,
 779 et al. 2018, *AJ*, 156, 123, doi: [10.3847/1538-3881/aabc4f](https://doi.org/10.3847/1538-3881/aabc4f)
- 780 Astropy Collaboration, Price-Whelan, A. M., Lim, P. L.,
 781 et al. 2022, *ApJ*, 935, 167, doi: [10.3847/1538-4357/ac7c74](https://doi.org/10.3847/1538-4357/ac7c74)
- 782 Bedell, M., Hogg, D. W., Foreman-Mackey, D., Montet,
 783 B. T., & Luger, R. 2019, *The Astronomical Journal*, 158,
 784 164, doi: [10.3847/1538-3881/ab40a7](https://doi.org/10.3847/1538-3881/ab40a7)
- 785 Bradbury, J., Frostig, R., Hawkins, P., et al. 2018, JAX:
 786 composable transformations of Python+NumPy
 787 programs, 0.3.13. <http://github.com/google/jax>
- 788 Cargile, P. A., Conroy, C., Johnson, B. D., et al. 2020, *The*
 789 *Astrophysical Journal*, 900, 28,
 790 doi: [10.3847/1538-4357/aba43b](https://doi.org/10.3847/1538-4357/aba43b)
- 791 Czekala, I., Andrews, S. M., Mandel, K. S., Hogg, D. W., &
 792 Green, G. M. 2015, *ApJ*, 812, 128,
 793 doi: [10.1088/0004-637X/812/2/128](https://doi.org/10.1088/0004-637X/812/2/128)
- 794 da Costa-Luis, C. O. 2019, *Journal of Open Source*
 795 *Software*, 4, 1277, doi: [10.21105/joss.01277](https://doi.org/10.21105/joss.01277)
- 796 García Pérez, A. E., Allende Prieto, C., Holtzman, J. A.,
 797 et al. 2016, *AJ*, 151, 144,
 798 doi: [10.3847/0004-6256/151/6/144](https://doi.org/10.3847/0004-6256/151/6/144)
- 799 Gerber, J. M., Magg, E., Plez, B., et al. 2023, *A&A*, 669,
 800 A43, doi: [10.1051/0004-6361/202243673](https://doi.org/10.1051/0004-6361/202243673)
- 801 Gully-Santiago, M., & Morley, C. V. 2022, *ApJ*, 941, 200,
 802 doi: [10.3847/1538-4357/aca0a2](https://doi.org/10.3847/1538-4357/aca0a2)
- 803 Harris, C. R., Millman, K. J., van der Walt, S. J., et al.
 804 2020, *Nature*, 585, 357, doi: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2)
- 805 Head, T., MechCoder, Louppe, G., et al. 2018,
 806 scikit-optimize/scikit-optimize: v0.5.2, v0.5.2, Zenodo,
 807 doi: [10.5281/zenodo.1207017](https://doi.org/10.5281/zenodo.1207017)
- 808 Horta, D., Price-Whelan, A. M., Hogg, D. W., Ness, M. K.,
 809 & Casey, A. R. 2025, arXiv e-prints, arXiv:2502.01745.
 810 <https://arxiv.org/abs/2502.01745>
- 811 Hunter, J. D. 2007, *Computing in Science & Engineering*, 9,
 812 90, doi: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55)
- 813 Husser, T. O., Wende-von Berg, S., Dreizler, S., et al. 2013,
 814 *A&A*, 553, A6, doi: [10.1051/0004-6361/201219058](https://doi.org/10.1051/0004-6361/201219058)
- 815 Ida, T., Ando, M., & Toraya, H. 2000, *Journal of Applied*
 816 *Crystallography*, 33, 1311,
 817 doi: <https://doi.org/10.1107/S0021889800010219>
- 818 Karalidi, T., Marley, M., Fortney, J. J., et al. 2021, *ApJ*,
 819 923, 269, doi: [10.3847/1538-4357/ac3140](https://doi.org/10.3847/1538-4357/ac3140)
- 820 Kawahara, H., Kawashima, Y., Masuda, K., et al. 2022,
 821 ExoJAX: Spectrum modeling of exoplanets and brown
 822 dwarfs, *Astrophysics Source Code Library*, record
 823 ascl:2206.003
- 824 Kingma, D. P., & Ba, J. 2017, Adam: A Method for
 825 Stochastic Optimization.
 826 <https://arxiv.org/abs/1412.6980>
- 827 Kramida, A., Yu. Ralchenko, Reader, J., & and NIST ASD
 828 Team. 2023, NIST Atomic Spectra Database (ver. 5.11),
 829 [Online]. Available: <https://physics.nist.gov/asd>
 830 [2024, June 12]. National Institute of Standards and
 831 Technology, Gaithersburg, MD.
- 832 Kruchten, N., Mease, J., & Moritz, D. 2022, in 2022 IEEE
 833 Visualization and Visual Analytics (VIS), 11–15,
 834 doi: [10.1109/VIS54862.2022.00011](https://doi.org/10.1109/VIS54862.2022.00011)
- 835 Kurucz, R. L. 2005, *Memorie della Societa Astronomica*
 836 *Italiana Supplementi*, 8, 14
- 837 Lacy, B., & Burrows, A. 2023, *The Astrophysical Journal*,
 838 950, 8, doi: [10.3847/1538-4357/acc8cb](https://doi.org/10.3847/1538-4357/acc8cb)
- 839 Leung, H. W., & Bovy, J. 2019, *MNRAS*, 483, 3255,
 840 doi: [10.1093/mnras/sty3217](https://doi.org/10.1093/mnras/sty3217)
- 841 —. 2024, *MNRAS*, 527, 1494, doi: [10.1093/mnras/stad3015](https://doi.org/10.1093/mnras/stad3015)
- 842 López-Valdivia, R., Mace, G. N., Han, E., et al. 2023, *ApJ*,
 843 943, 49, doi: [10.3847/1538-4357/acab04](https://doi.org/10.3847/1538-4357/acab04)
- 844 Mahadevan, S., Ramsey, L., Bender, C., et al. 2012, in
 845 Society of Photo-Optical Instrumentation Engineers
 846 (SPIE) Conference Series, Vol. 8446, Ground-based and
 847 Airborne Instrumentation for Astronomy IV, ed. I. S.
 848 McLean, S. K. Ramsay, & H. Takami, 84461S,
 849 doi: [10.1117/12.926102](https://doi.org/10.1117/12.926102)
- 850 Majewski, S. R., Schiavon, R. P., Frinchaboy, P. M., et al.
 851 2017, *AJ*, 154, 94, doi: [10.3847/1538-3881/aa784d](https://doi.org/10.3847/1538-3881/aa784d)
- 852 Marley, M. S., Saumon, D., Visscher, C., et al. 2021, *ApJ*,
 853 920, 85, doi: [10.3847/1538-4357/ac141d](https://doi.org/10.3847/1538-4357/ac141d)
- 854 Morley, C. V., Mukherjee, S., Marley, M. S., et al. 2024,
 855 arXiv e-prints, arXiv:2402.00758,
 856 doi: [10.48550/arXiv.2402.00758](https://doi.org/10.48550/arXiv.2402.00758)
- 857 Mukherjee, S., Fortney, J. J., Morley, C. V., et al. 2024,
 858 arXiv e-prints, arXiv:2402.00756,
 859 doi: [10.48550/arXiv.2402.00756](https://doi.org/10.48550/arXiv.2402.00756)
- 860 NVIDIA, Vingelmann, P., & Fitzek, F. H. 2020, CUDA,
 861 release: 10.2.89.
 862 <https://developer.nvidia.com/cuda-toolkit>
- 863 pandas development team, T. 2020, pandas-dev/pandas:
 864 Pandas, latest, Zenodo, doi: [10.5281/zenodo.3509134](https://doi.org/10.5281/zenodo.3509134)
- 865 Paszke, A., Gross, S., Massa, F., et al. 2019, PyTorch: An
 866 Imperative Style, High-Performance Deep Learning
 867 Library. <https://arxiv.org/abs/1912.01703>
- 868 Rains, A. D., Nordlander, T., Monty, S., et al. 2024,
 869 *MNRAS*, 529, 3171, doi: [10.1093/mnras/stae560](https://doi.org/10.1093/mnras/stae560)
- 870 Rayner, J. T., Cushing, M. C., & Vacca, W. D. 2009, *ApJS*,
 871 185, 289, doi: [10.1088/0067-0049/185/2/289](https://doi.org/10.1088/0067-0049/185/2/289)

- 872 Róžański, T., Ting, Y.-S., & Jabłońska, M. 2023, arXiv
873 e-prints, arXiv:2306.15703,
874 doi: [10.48550/arXiv.2306.15703](https://doi.org/10.48550/arXiv.2306.15703)
- 875 Satyanarayan, A., Moritz, D., Wongsuphasawat, K., &
876 Heer, J. 2017, IEEE transactions on visualization and
877 computer graphics, 23, 341
- 878 Shankar, S. 2024, blase II: PHOENIX Subset Clone
879 Archive, 0.0.1, Zenodo, doi: [10.5281/zenodo.11246174](https://doi.org/10.5281/zenodo.11246174)
- 880 Shankar, S., Gully-Santiago, M., Morley, C., et al. 2024,
881 The Journal of Open Source Software, 9, 6601,
882 doi: [10.21105/joss.06601](https://doi.org/10.21105/joss.06601)
- 883 Van Rossum, G., & Drake, F. L. 2009, Python 3 Reference
884 Manual (Scotts Valley, CA: CreateSpace)
- 885 VanderPlas, J., Granger, B., Heer, J., et al. 2018, Journal of
886 Open Source Software, 3, 1057, doi: [10.21105/joss.01057](https://doi.org/10.21105/joss.01057)
- 887 Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020,
888 Nature Methods, 17, 261, doi: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2)
- 889 Wes McKinney. 2010, in Proceedings of the 9th Python in
890 Science Conference, ed. Stéfan van der Walt & Jarrod
891 Millman, 56 – 61, doi: [10.25080/Majora-92bf1922-00a](https://doi.org/10.25080/Majora-92bf1922-00a)
- 892 Wheeler, A. J., Abruzzo, M. W., Casey, A. R., & Ness,
893 M. K. 2023, AJ, 165, 68, doi: [10.3847/1538-3881/aaaaad](https://doi.org/10.3847/1538-3881/aaaaad)