A Method for the spectroscopic inference of fundamental stellar parameters

May 21, 2014

Ian Czekala[1] et al.

iczekala@cfa.harvard.edu

1. Introduction

- Where our technique fits in the ecosystem of stellar methods.
- Applicable to any kind of spectrum (long-slit, echelle, infrared, flux-calibrated or not)
- Examples of fields where accurate and unbiased stellar parameters are crucial. Exoplanets. T Tauri stars.

2. Method

A single spectrum provides a wealth of information that can be used to infer the fundamental properties of a star: effective temperature $T_{\rm eff}$, surface gravity $\log g$, and metallicity [Fe/H]; together $\boldsymbol{\theta}_{\star,\rm grid}$. We wish to extract the maximal amount information from a given spectrum while also recognizing the limitations imposed by our stellar model and exposing any degeneracies between the parameters. For example, if we model a large range of the spectrum at once (e.g., the full optical range at high resolution), then we must explicitly account for any covariances and biases introduced by improper calibration or inaccuracies in the model spectrum. We present a generative model which uses a synthetic stellar library as a backend and apply post-processing techniques to mimic the observed spectrum as a function of several stellar parameters and calibration parameters in order to determine the best fit. By forward modelling the data spectrum in a Bayesian framework, our modular approach can easily incorporate additional calibration artifacts and stellar complexity (e.g., veiling or Zeeman splitting) while still being true to the limitations of the dataset. This data driven approach will enable us to learn how the underlying synthetic models may be improved and refined in an iterative manner.

The roadmap for our method is as follows. First, for a given set of fundamental stellar parameters $\boldsymbol{\theta}_{\star,\rm grid}$, we generate a synthetic spectrum by interpolating from a grid of synthetic spectra (§2.1). Then, we post-process the interpolated spectrum for realism by using additional parameters to describe Doppler shift, projected rotational velocity, interstellar extinction, and residual flux calibration errors (§2.2). We perform a pixel-by-pixel comparison of the generated spectrum with the data spectrum using a prescribed likelihood function (§2.3) and a parametric treatment of the covariance between pixel residuals (§2.4). We

[1]Harvard-Smithsonian Center for Astrophysics, 60 Garden Street MS 10, Cambridge, MA 02138

use a Markov chain Monte Carlo (MCMC) Gibbs sampler to numerically explore the posterior probability density and determine constraints on the stellar and nuisance parameters (§2.5). The benefit of our method is that it provides robust estimates of the stellar and nuisance parameters with well-quantified uncertainties and parameter degeneracies. Our parameterization of the covariance between pixel residuals also yields a means to identify and quantify the key discrepancies between the data and model, and provides a path forward to improve the stellar models in an iterative fashion (§2.6).
TODO: I really want to overload the word "covariance" and use it to describe both the covariance between $T_{\rm eff}$ and $\log g$ as well as the covariance structure of the pixel residuals themselves. However I think this might be confusing so I am using "degeneracies" for the former and reserving "covariance" for the later.

2.1. Synthetic Model Spectra

Many methods exist to generate a synthetic model of a stellar spectrum. Generally, when a high quality spectrum is needed for a specific set of stellar parameters, one runs a stellar atmosphere and radiative transfer code to synthesize the spectrum. Unfortunately, it is computationally expensive to accurately generate a spectrum over a wide range of wavelengths, and so these codes are generally employed to generate libraries of stellar spectra, which exist over a range of $\boldsymbol{\theta}_{\star,\rm grid}$ at specified points on a regular grid, typically covering the main sequence of stellar types. Several high-quality libraries of synthetic spectra now exist (Husser et al. 2013, Kurucz, and more are being developed for GAIA).

In a forward-modelling approach, which may make many calls to generate a spectra of varying $\boldsymbol{\theta}_{\star,\rm grid}$, it is currently computationally impractical to iterate the structure of the stellar atmosphere and then run the stellar synthesis for each call of $\boldsymbol{\theta}_{\star,\rm grid}$. Some methods compromise by taking a halfway approach, whereby a precomputed library of stellar atmospheres is interpolated for a given $\boldsymbol{\theta}_{\star,\rm grid}$, and then a simple radiative transfer code is run to synthesize a spectrum (e.g. SME; Valenti & Piskunov 1996). This technique has the advantage that interpolating atmospheres and then running the spectral synthesis for each $\boldsymbol{\theta}_{\star,\rm grid}$ might be more accurate than directly interpolating synthetic spectra. However, if a large region of wavelength is required at high resolution, then the synthesis step will be slow. A different approach might be instead to eschew forward modelling (and thus repeated synthesis or grid interpolation) of the spectra entirely and evaluate the posterior probability density only at the grid points of the spectral library. Then, these discrete samples of the posterior density can be interpolated to find the posterior surface and draw confidence intervals on the parameters (e.g. SPC; Buchhave et al. 2012). However, the difficulty with this approach is that the parameter uncertainty might be smaller than the grid spacing, and then interpolation will not properly capture degeneracies between parameters.

Because we desire a generative model of the spectra without the added complexity or computational overhead of running a stellar synthesis, we opt to use a
TODO: Implement a more accurate but still fast interpolator for the grid. Likely this will be a combination of using splines to pre-interpolate a finer grid (20 K, 0.1 dex spacing in gravity and metallicity and then doing linear interpolation off of this new grid during the actual MCMC run. It will be described here. However,

the techniques we develop in this paper to account for and parameterize the correlated, systematic pixel residuals are generally applicable to spectra fit by any of these methods.

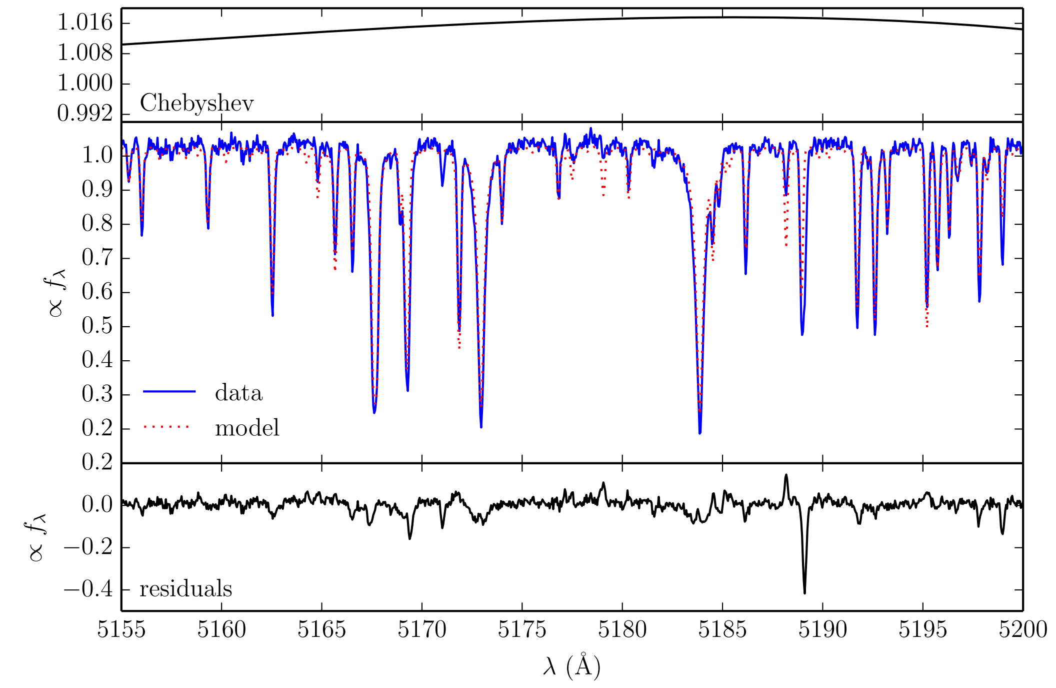

Fig. 1.— **Top**: Chebyshev polynomial which multiplies the model spectrum to account for inaccuracies in the flux-calibration. **Middle**: The data spectrum and model spectrum, after it has been interpolated, post-processed, and multiplied by the Chebyshev polynomial. **Bottom**: Residuals from the model fit. Note the large residual at λ5189Å due to a missing opacity source from Ca II.

2.2. Post-Processing

Libraries of synthetic spectra are generally computed at high resolution ($R \geq$100,000), sampled with many pixels per resolution element, and do not include any rotational broadening or account for any instrumental effects. In order to transform a raw synthetic spectrum interpolated from the grid into one that matches the spectrum of a real star, we must post-process the spectrum to account for these secondary effects. In addition to these fundamental parameters, a star also has several observed properties that are a function of its kinematics, geometry, and location in our galaxy: projected rotational velocity $v \sin i$, line of sight velocity v_z, total extinction A_V, and solid angle Ω. We call these secondary parameters $\boldsymbol{\theta}_{\star,\mathrm{post}}$, because we can model their effects on the synthetic spectra in a "post-processing" step by convolution with an appropriate kernel or multiplication with a smooth function. Together, we call $\boldsymbol{\theta}_\star = \{\boldsymbol{\theta}_{\star,\mathrm{grid}}, \boldsymbol{\theta}_{\star,\mathrm{post}}\}$.

The projected rotational velocity of the star, parameterized by $v \sin i$, broadens spectral line profiles and is mathematically described by convolution with a specific kernel (Gray 2008, Equation 18.14). UV, optical, and infrared spectra are acquired using a spectrograph which disperses light onto a CCD, defining a specific resolution and sampling rate for the spectrum. Because the resolution of the spectrograph and pixel sampling are different from the synthetic spectral library, we must convolve the raw spectrum with the line spread function (LSF) of the instrument and resample to the exact pixels of the CCD. In wavelength space, the $v \sin i$ and LSF operations are represented by the convolution of the synthetic spectrum with these two kernels. Using the convolution theorem, we can rewrite these convolutions as multiplications in Fourier space. In order to use the fast Fourier transform (FFT) with discrete samples of the synthetic spectrum, we must first resample it onto a uniform grid. In the case of a spectrum, the natural uniform grid is one that is equally sampled in velocity space, such there is an equal velocity shift Δv between pixels. This results in a wavelength grid that is linearly sampled in the logarithm of wavelength. Next, we multiply the Fourier transform of the synthetic spectrum by the Fourier transforms of the rotational velocity and line spread function kernels. Lastly, we do an inverse FFT to transform the modified spectrum back to wavelength space, where it is sampled at the wavelength locations of the pixels in the detector. When resampling the spectrum, we are careful to ensure band-limited interpolation using splines in order to prevent introducing non-physical high frequency structure into the spectrum.

We scale the raw synthetic flux (often reported at the stellar surface) to an observed flux by multiplying by the solid angle of the star, $\Omega = R_\star^2/d^2$, where $R_\star$ is the stellar radius and d is the distance to the star from earth. Then, we correct for a radial velocity difference v_z between the star and the earth by Doppler shifting the model spectrum. Finally, we correct for interstellar extinction by dividing by a dust extinction model A_λ, which is parameterized by extinction in the V photometric band, A_V.
TODO: We can add in convolution with photometric filter profiles to use photometry and provide a better constraint on the luminosity of the star
TODO: We might try implementing the convolution in wavelength space, rather than the FFT. This would allow the LSF to change with wavelength.

Calibration polynomials In contrast to a collection of photometric points in a spectral energy distribution, or any ensemble of data for which each data point is acquired by an individual measurement, adjacent wavelength points in a spectrum are measured simultaneously, with the same detector and–more importantly–the same calibration. Typically, spectra are flux calibrated by dividing by a "sensitivity function," which is determined by comparing an observation of a spectrophotometric standard to a previously calibrated model spectrum. Therefore, any error in the sensitivity function, which is typically a smooth function with wavelength, will create a region of the data spectrum which is systematically offset by a small amount from what the true flux should be. The flux calibration error is then the residual difference in the empirically determined sensitivity function from the what the "true" sensitivity function would be if we had a perfectly calibrated instrument and spectrophotometric standard. Therefore, it is necessary to introduce additional degrees of freedom into our model in order to account for the calibration error and allow the model spectrum to properly match the data. By adding the residual calibration as a parameter in our model, we can explore the distribution of possible calibrations (§2.5) and then marginalize the calibration parameters out of the final posterior distribution of the stellar parameters. This means that we can derive estimates of stellar parameters that properly account for the intrinsic uncertainty in the flux calibration.

A traditional way to deal with errors in the flux calibration is to avoid flux-calibration entirely and simply normalize both the data and the spectrum to the continuum level before comparison. While this procedure may work well for hotter stars with well-defined continuua, normalizing spectra of cooler stars with large molecular features may introduce artificial features into the spectrum when the pseudo-continuum is incorrectly placed. Instead, we choose to parameterize the residual flux calibration error by a low-order Chebyshev polynomial which is normalized to unity. The coefficients of the polynomial are parameters in the model that will be determined by the fitting algorithm, $\boldsymbol{\theta}_{\rm Cheb} = \{c_0, c_1, \ldots, c_N\}$. If we have multiple observations of spectrophotometric standards, then we can characterize the typical flux-calibration error of the spectrograph and put priors on the coefficients that will prevent overfitting of the polynomials. For an example of the Chebyshev polynomial determined for a typical spectrum, see Figure 1. Using priors on the Chebyshev coefficients ensures that real spectral features, such as deep molecular bands in later type stars, are not accidentally treated as calibration artifacts. The overall normalization of the polynomial c_0 is degenerate with the solid angle of the star Ω. Therefore we apply an additional constraint and enforce that the mean of the Chebyshev polynomial be equal to 1. If there is only one order in the spectrum, such as with a long-slit spectrum or a single order of an echelle spectrum, then this means setting $c_0 = 1$. If we are fitting more than one order of an echelle spectrum then we adopt an independent Chebyshev polynomial for each order. This means that we chose the c_0 of a specific order to anchor to 1, while the c_0 for other orders are allowed to take on perturbations about 1.

This polynomial formalism can in principle be used to model raw (not flux calibrated) spectra by using the Chebyshev polynomial to solve directly for the sensitivity function itself, rather than a residual calibration error as is currently done. When used in this manner, there will be greater uncertainty about the properties of the spectrograph, which will translate into wider priors on the Chebyshev coefficients. In this case, there is more of a chance that the polynomial might destroy broad-scale information present in the spectrum by fitting it out, something that would be avoided with more accurate priors provided by a flux calibration.

2.3. Likelihood Calculation

To evaluate which parameters of the model spectrum fit the data spectrum best, we use a likelihood function to compare individual pixels between the two spectra. A likelihood function provides the probability of the data spectrum $\vec{D}$, given the model spectrum $\vec{M}(\boldsymbol{\theta})$, which is parameterized by the stellar and calibration parameters $\boldsymbol{\theta}$. The data spectrum and model spectrum are both composed of N pixels, and so $\vec{D}$ and $\vec{M}$ represent the collection of these pixels in vector form, $\vec{D} = \{D_i\}_{i=1}^{N}$ and $\vec{M} = \{M_i\}_{i=1}^{N}$, respectively.

The vector of residuals is the difference between the data spectrum and the model spectrum

$$R(\boldsymbol{\theta}) = \vec{D} - \vec{M}(\boldsymbol{\theta}) \tag{1}$$

In the language of probability calculus, the likelihood function is written as $p(\vec{D}|\vec{M}(\boldsymbol{\theta}))$, or $p(\vec{D}|\boldsymbol{\theta})$ for short. We choose a standard multidimensional Gaussian likelihood function which allows for covariance between the pixel residuals

$$p(\vec{D}|\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^N \det(C)}} \exp\left(-\frac{1}{2} R^T C^{-1} R\right) \tag{2}$$

which penalizes models that yield larger residuals. C is the covariance matrix which describes the covariance between the pixel residuals. The structure of the covariance matrix is discussed in detail in the next section (§2.4). Because the likelihood function generally exhibits a large dynamic range, it is numerically preferred to use the log-likelihood function

$$\ln[p(\vec{D}|\boldsymbol{\theta})] = -\frac{1}{2} R^T C^{-1} R - \frac{1}{2} \ln \det C - \frac{N}{2} \ln 2\pi \tag{3}$$

2.4. Parameterizing the covariance structure

The covariance matrix C describes the measurement noise in each pixel of the spectrum as well as the covariance between pixel residuals. It is fundamental to the likelihood calculation described in §2.3. For most problems, such as fitting a spectral energy distribution to photometric measurements, each data point is acquired in an independent measurement from the other data points, which means that the noise for each data point, σ_i, is also independent from the other data points. This results in a diagonal covariance matrix

$$C = \begin{bmatrix} \sigma_0^2 & 0 & \cdots & 0 \\ 0 & \sigma_1^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{bmatrix} \tag{4}$$

with $\sigma_{ij} = \delta_{ij}\sigma_{ij}$, where δ_{ij} is the Kronecker delta function. When the covariance matrix is diagonal, the likelihood function (Equation 3) reduces to the familiar χ^2 form of a sum over the square of the residuals, weighted by the inverse variance of each data point

$$\ln[p(\vec{D}|\boldsymbol{\theta})] \propto -\frac{1}{2}\chi^2 = -\frac{1}{2}\sum_i^N \frac{R_i^2}{\sigma_i^2} \tag{5}$$

However, the pixel residuals from a spectroscopic fit will often be correlated and will invalidate Equation 5. These correlations arise because the CCD pixels of a spectrograph do not sample independent wavelengths of the spectrum. Instead, spectrographs are designed so that the LSF is properly sampled by at least a few pixels, meaning that adjacent pixels do not sample completely independent parts of the spectrum. Because of this, when a spectral feature differs between the model and the data, it creates a residual

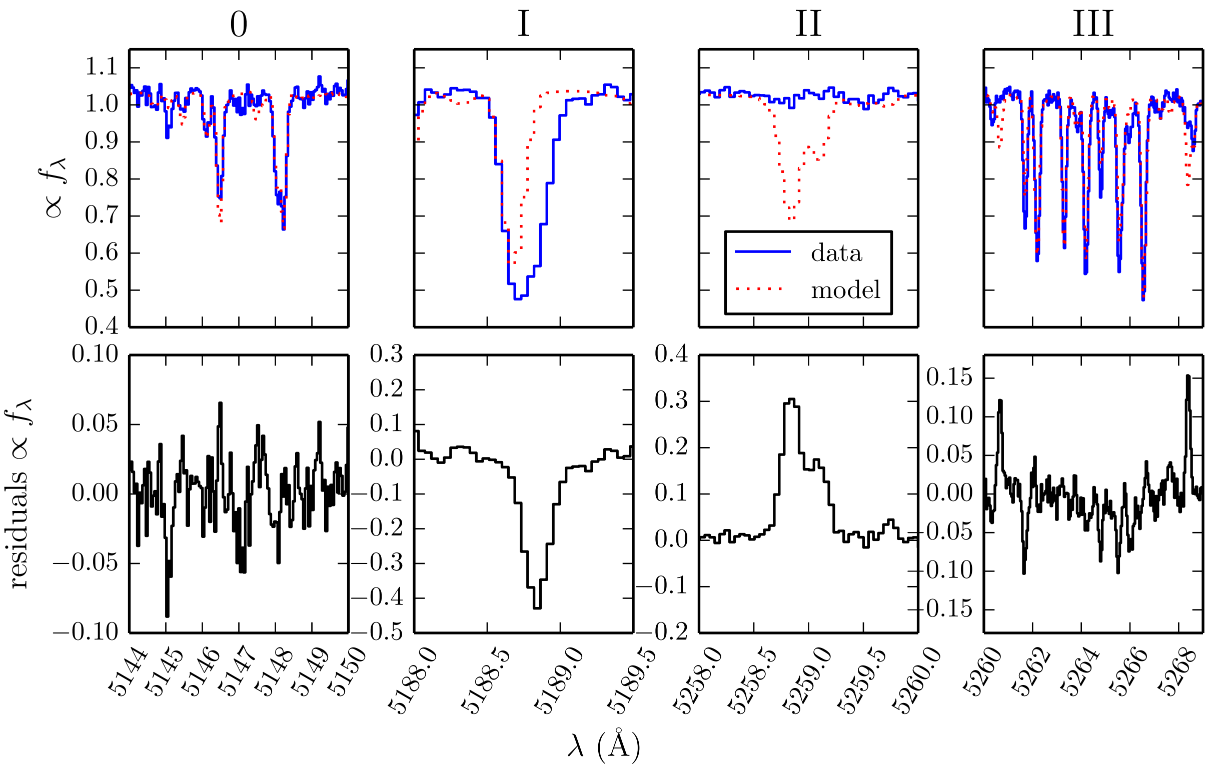

Fig. 2.— A collection of spectral lines which have imperfect model fits. From left to right: **Class 0** The majority of spectral lines ($\gtrsim 60\%$) will have minor differences in strength between data and model spectrum, which produce low-amplitude correlations in the residuals on the length scale of the width of a typical spectral line. **Class I**: Sometimes ($\lesssim 5\%$ of all lines), a missing opacity source in the model (in this case a line-blended Ca II) leaves a large, highly correlated patch of negative residuals. **Class II**: Sometimes ($\lesssim 5\%$ of all lines), an extraneous line in the model leaves a large, highly correlated patch of positive residuals. **Class III**: If the line strengths are substantially discrepant ($\lesssim 10\%$ of all lines), there will be many correlated residuals of moderate amplitude. The difficulty with class III lines is that for any specific line, there might exist a $\boldsymbol{\theta}_\star$ that will fit the line, but there does not exist a $\boldsymbol{\theta}_\star$ that will properly fit *all* the lines.

which spans several adjacent data points. For a collection of correlated residuals produced by a typical spectroscopic fit, see Figure 2.

There is an important distinction between astrophysical or instrumental "noise" and the residuals from a spectroscopic fit. Noise introduced to a spectrum by astrophysical or instrumental effects is generally uncorrelated with wavelength. This is because the arrival of each photon to the detector is a discrete event, and while photons are scattered by the LSF, the particular scattering direction of each photon is independent from any other photon. We can verify this by measuring the autocorrelation of residuals in a region of well-fit stellar continuum, and show that it is flat. If the model spectrum is *perfect*, then the residuals from a spectroscopic fit will be uncorrelated and equivalent to the astrophysical and instrumental noise introduced

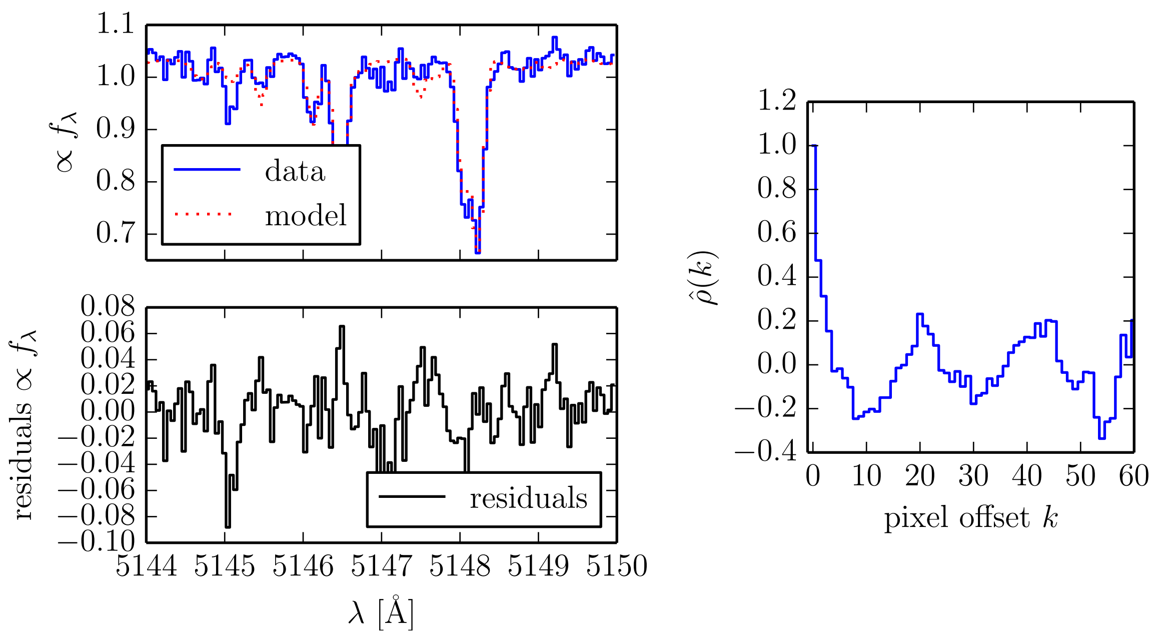

Fig. 3.— **Left** The same residuals from Figure 2, panel 0, but enlarged to show the mildly covariant structure produced by slight mismatch between the data and model spectra. **Right** The autocorrelation (Equation 6) of the residual sequence shown at left. Notice that there is significant correlation for offsets of $\lesssim 4$ pixels.

into the measurement. However, when we compare a data spectrum and a model spectrum that have many absorption lines, even small differences will produce correlated *residuals*. We can verify this correlation by estimating the autocorrelation of the residuals as a function of pixel offset k

$$\hat{\rho}(k) = \frac{1}{(n-k)\,\sigma^2} \sum_{i=1}^{n-k} (R_i - \mu_R)\,(R_{i+k} - \mu_R) \tag{6}$$

over a region of spectrum of length n pixels. Figure 3 shows a general residual structure induced by slight model mismatch and the autocorrelation of this series. There is significant correlation for offsets of $\lesssim 4$ pixels, which is also the typical width of a spectral line for this spectrum.

Although the correlation in the residuals is artificially induced by an imperfect model fit rather than originating from a physical source of correlated noise, we can treat these two effects as mathematically similar. By using a non-trivial covariance matrix with off-diagonal terms we can describe the small-scale covariance between pixel residuals shown in Figures 2 and 3. A non-diagonal covariance also requires that we use a likelihood function which uses a matrix product instead of a sum over independent pixels (Equation 3).

We seek to understand and parameterize the structure of covariance in the spectroscopic residuals, because ignoring this complexity will bias the estimates of the model parameters $\boldsymbol{\theta}$. As a simple analogy, consider the fit of a straight line to a dataset. If the noise in the dataset is correlated, then nearby data points might be offset from the linear trend in the same direction by a similar amount. If the covariant structure of the noise is ignored, a simple χ^2 will treat these correlated offsets as due to the linear trend, which will result in a biased determination of the slope and intercept of the line, typically with uncertainties that are too small. We face a mathematically similar problem with correlated pixel residuals from a spectroscopic fit.

We parameterize the covariance structure using covariance kernels, taken from the field of Gaussian processes. Covariance kernels describe the covariance between two pixels λ_i and λ_j. These kernels are analogous to the cosmological two-point correlation function, where the distance between two galaxies is used instead of the wavelength of light.

Global and mildly covariant structure To account for the mild covariance structure shown in Figure 3, which is generally low amplitude and has a correlation length of only a few pixels but is present throughout the majority of the spectrum, we use a *stationary* covariance kernel. Stationary covariance kernels, also called *radial basis functions*, satisfy the property that the amount of correlation only depends on the distance between the two pixels r. For spectra, we map r to the velocity difference between two pixels

$$r(\lambda_i, \lambda_j) = \Delta v = \frac{c}{2}\left|\frac{\lambda_i - \lambda_j}{\lambda_i + \lambda_j}\right| \tag{7}$$

Then the covariance kernel describes the expected covariance between pixel residuals

$$k(\lambda_i, \lambda_j) = \langle R_i \, R_j \rangle \tag{8}$$

In theory, we could chose from a plethora of covariance kernels to parameterize this mildly covariant structure, as long as they satisfy the property that the degree of covariance smoothly declines with increasing radial distance. In practice, there are a few covariance kernels which are well-respected within the machine-learning community and frequently employed to parameterize covariance of this nature (Rasmussen & Williams 2005). We choose to use the Matérn $\nu = 3/2$ kernel because it is one of the most commonly used kernels for this purpose and has an extensive history of use (cite?).

$$k_{\nu=3/2}(r \,|\, a_{\rm g}, l) = a_{\rm g}\left(1 + \frac{\sqrt{3}r}{l}\right)\exp\left(-\frac{\sqrt{3}r}{l}\right) \tag{9}$$

This kernel is an exponential tapered by a polynomial, which makes for a smooth transition from small scale covariance to negligible covariance at longer scales. In order to ensure sparsity of the covariance matrix for computational efficiency, we taper the Matérn kernel for compact support using a Hann window

$$w(r|\, r_0) = \begin{cases} 0.5 + 0.5\cos(\pi r/r_0) & r \le r_0 \\ 0 & r > r_0 \end{cases} \tag{10}$$

for which we generally fix r_0 to be some fixed multiple of the length scale, say $r_0 = 4l$. Then, if we define the global parameters $\boldsymbol{\theta}_{\rm global} = \{a_g, l\}$, we have a final covariance kernel of

$$k_{\rm global}(r \,|\, \boldsymbol{\theta}_{\rm global}) = w(r|\, r_0 = 4l)\; k_{\nu=3/2}(r|\, a_{\rm g}, l) \tag{11}$$

TODO: This might be a good place to actually include a simple plot of $k_{\rm global}$ as a function of r, as long as we aren't providing too many figures. Maybe it could be combined with Figure 4.

We add this covariance kernel to the independent pixel noise $\delta_{ij}\sigma_{ij}$ already present along the matrix diagonal, and introduce an additional parameter to allow a constant rescaling of the independent noise. Then, each element of the covariance matrix is completely described by

$$k(\lambda_i, \lambda_j | \boldsymbol{\theta}_{\rm global}, b) = k_{\rm global}(r|\boldsymbol{\theta}_{\rm global}) + b\,\delta_{ij}\sigma_{ij} \qquad (12)$$

Therefore, the *hyperparameters* of this covariance kernel directly determine the structure of the covariance matrix. By varying the hyperparameters, we can create a covariance matrix which accurately mimics the correlated structure of the pixel residuals. Most importantly, when the hyperparameters are correctly chosen to properly model this correlated structure, the likelihood function will be maximized (Equation 2) while also encapsulating the true uncertainty in the spectroscopic fit.

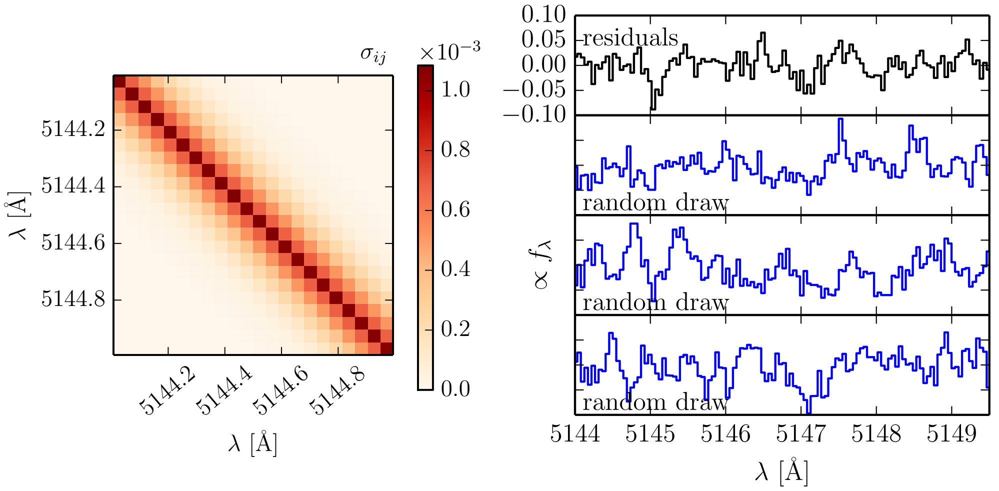

Fig. 4.— **Left** Inset zoomed to show a small region of a typical covariance matrix, generated using the kernel in Equation 11 and common values for the hyperparameters. There is a small degree of covariance in elements within a few pixels off the diagonal, which quickly tapers off so that the majority of the matrix remains sparse ($\sigma_{ij} = 0$). This matrix is used to model a spectrum which is correlated on a ∼few pixel scale. **Right** To demonstrate that this matrix properly models the correlated structure of the residuals, we compare the residuals to random residuals generated from a multivariate normal distribution with this covariance matrix. The top panel shows the same residuals shown in Figure 3, and below are plotted three sets of simulated residuals. The amplitude and correlation length of the simulated residuals closely approximates the structure of the actual pixel residuals.

The left panel of Figure 4 shows what the covariance matrix might look like for reasonable choices of the kernel hyperparameters. To verify that this covariance matrix indeed mimics the correlated structure seen in the residuals, we can draw a random vector of residuals $R_{\rm rand}$ from our noise model and compare it to the actual pixel residuals. Because we are using a multidimensional Gaussian as our likelihood function (Equation 2), we have assumed that the underlying residual structure in a spectroscopic fit is described by a multidimensional Gaussian. This is directly analogous to the implicit assumption of independent Gaussian noise when using a standard χ^2 likelihood (Equation 5). If one wished to compare the residuals that resulted from a χ^2 fit to the noise in the data set, then one could simulate $R_{\rm rand}$ by drawing N random samples from a univariate normal distribution with zero mean and variance σ_i^2. We can make the same test with our multidimensional Gaussian likelihood function, however in this case the mean is a vector $\vec{\mu} = 0$ with length N and the variance is described by the covariance matrix C. The residual vector is then a random draw from this multivariate normal distribution

$$R_{\rm rand} \sim \mathcal{N}(\vec{\mu} = 0, C) \tag{13}$$

Functions to generate random samples from a multidimensional Gaussian are implemented in most numerical programming languages. As we see in the right panel of Figure 4, the structure and amplitude of the fake residuals closely mimics that of the actual data residuals, suggesting that the covariance kernel faithfully reproduces the covariance induced by the data-model mismatch.

Specific line covariance Even after the mildly covariant structure has been accounted for by the global covariance kernel (Equation 11), there still exist a significant number large amplitude pixel residuals that are *highly correlated*. These patches of large amplitude residuals are caused by incorrect synthetic lines like those in the class I, II, and III examples in Figure 2. To parameterize the covariance in a given patch of these residuals, we introduce an additional *non-stationary* covariance kernel. In this context, non-stationary means that the degree of covariance explicitly depends on the values of λ_i and λ_j and not simply the distance between them, enabling us to individually target the large residuals induced by the mismatch of a *specific* spectral line.

Because the shape of a spectral line can be generally approximated by a Gaussian function, and the difference of two Gaussians with the same mean is also another Gaussian, we use Equation 8 to design a new covariance kernel about the Gaussian function. Consider the residuals from a single spectral line, labeled by the subscript 0. If we expect the shape of the residuals to be a Gaussian function about the line center with parameters $\boldsymbol{\theta}_{{\rm line}_1} = \{a_1, \lambda_{\mu_1}, \sigma_1\}$

$$R(\lambda|\boldsymbol{\theta}_{{\rm line}_1}) = \frac{a_1}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{r^2(\lambda, \lambda_{\mu_1})}{2\sigma_1^2}\right) \tag{14}$$

then the covariance of any two pixels along this Gaussian residual is

$$k_{\rm G}(\lambda_i, \lambda_j|\boldsymbol{\theta}_{{\rm line}_1}) = \langle R(\lambda_i)\, R(\lambda_j)\rangle \tag{15}$$

$$= \frac{a_1^2}{2\pi\sigma_1} \exp\left(-\frac{r^2(\lambda_i, \lambda_{\mu_1}) + r^2(\lambda_j, \lambda_{\mu_1})}{2\sigma_1^2}\right) \tag{16}$$

We again taper the kernel with a Hann window (Equation 10) for compact support, and so the covariance kernel for the highly correlated residuals induced by a poor line fit becomes

$$k_{\rm line}(\lambda_i, \lambda_j|\boldsymbol{\theta}_{{\rm line}_1}) = w(r, r_0)\, k_{\rm G}(\lambda_i, \lambda_j|\boldsymbol{\theta}_{{\rm line}_1}) \tag{17}$$

To generate the covariance matrix, we now sum the effects of all the kernels, including the global covariance kernel, the specific line kernel, and the individual pixel noises

$$k(\lambda_i, \lambda_j | \boldsymbol{\theta}_{\mathrm{line}_1}, \boldsymbol{\theta}_{\mathrm{global}}, b) = k_{\mathrm{line}}(\lambda_i, \lambda_j | \boldsymbol{\theta}_{\mathrm{line}_1}) + k_{\mathrm{global}}(r | \boldsymbol{\theta}_{\mathrm{global}}) + b\,\delta_{ij}\sigma_{ij} \tag{18}$$

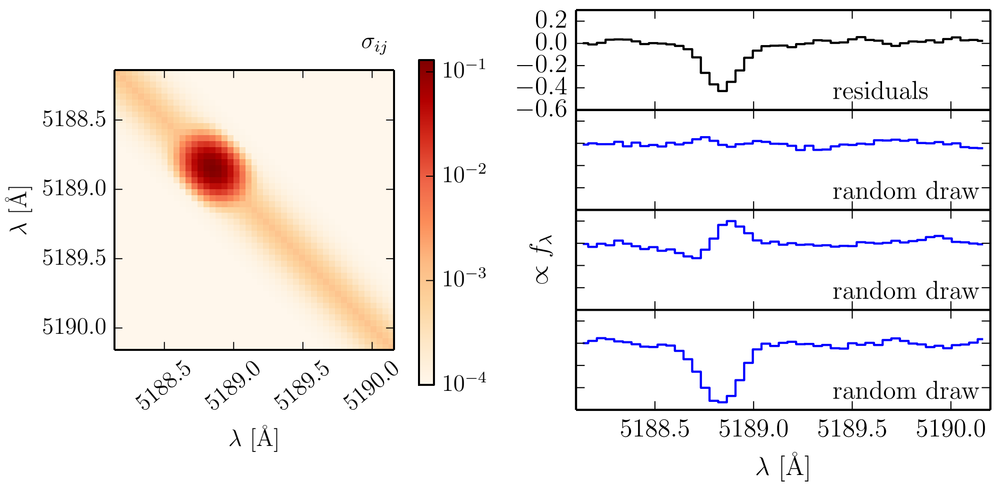

Fig. 5.— **Left** A typical covariance matrix including the Gaussian line kernel (Equation 18). The same global covariance shown in Figure 4 is still present with the same hyperparameters, however now there is an additional patch of high covariance corresponding to the large, Gaussian-shaped residuals. These larger elements in the covariance matrix effectively down-weight the contribution of a poorly modeled synthetic spectral line. **Right** The same spectroscopic residuals shown in Figure 2, class I, shown with three random draws from the covariance matrix. The random draws can take on a range of amplitudes–positive, negative, or even flat–because they are simply random draws that are described by the covariance matrix. The wide range of possible residual amplitudes match the structure and amplitude of the pixel residuals.

In Figure 5, we see that the non-stationary Gaussian line kernel localizes the increased variance to a narrow range of the spectrum, corresponding to the residual from a specific poorly modeled spectral line. This increased covariance has the effect of down-weighting the influence of a specific spectral line on the likelihood function, the same way we might down-weight an outlier while fitting a straight line to a collection of data points.

When iteratively fitting a spectrum (see §2.5), we continue to add line covariance kernels until we have covered all of the high amplitude residuals. Typically, we will add line kernels until all residuals greater than three times the amplitude of the global covariance kernel are covered. In a single order of an echelle spectrum, there may be N regions of high covariance that are parameterized by several line kernels (Equation 17), which we group into an aggregate parameter $\boldsymbol{\theta}_{\mathrm{lines}} = \{\boldsymbol{\theta}_{\mathrm{line}_1}, \boldsymbol{\theta}_{\mathrm{line}_2}, \ldots, \boldsymbol{\theta}_{\mathrm{line}_N}\}$. Along with the global covariance parameters and Chebyshev parameters for this order, we call the collection of nuisance hyperparameters for a specific order (e.g., order 1) $\boldsymbol{\theta}_{\mathrm{order}_1} = \{\boldsymbol{\theta}_{\mathrm{Cheb}}, \boldsymbol{\theta}_{\mathrm{global}}, \boldsymbol{\theta}_{\mathrm{lines}}\}$. Taken together, the aggregated nuisance parameters for all of the N orders are stored in $\boldsymbol{\theta}_{\mathrm{orders}} = \{\boldsymbol{\theta}_{\mathrm{order}_1}, \boldsymbol{\theta}_{\mathrm{order}_2}, \ldots, \boldsymbol{\theta}_{\mathrm{order}_N}\}$.

TODO: We can (should?) add an additional level to the hierarchy of hyperparameters and add parameters to describe the *population characteristics* of poorly modeled spectral lines (mostly the typical width, amplitude of these lines). This will tell us about the frequency and distribution of spectral modelling errors. We explicitly fit for the hyperparemeters of the covariance kernels at the same time we fit for the stellar parameters. While including these extra parameters does increase the dimensionality and complexity of our model, they do provide several advantages. *A priori* we do not know which regions of the spectrum are improperly modeled. The covariance hyperparameters provide a framework to identify these regions iteratively and in a self-consistent manner.

The traditional practice method of dealing with spectral mismatch is to simply mask out the regions of the spectrum which do not agree to within a certain tolerance. Rather than arbitrarily excluding regions of the spectrum from the fit, these regions should instead be incorporated into in the fit with the appropriate weight. A model that includes covariance is far more likely than forcing the synthetic spectrum to fit perfectly, and far more flexible than arbitrarily masking regions which do not fit. The fitting procedure allows these weights to be determined self-consistently, such that lines which are slightly wrong can still bring information to bear on the stellar parameters. In fact, it may be the case that the lines that are off by a small amount (class 0 and III lines in Figure 2) actually provide the *most information* about the stellar parameters, precisely because these lines are the most sensitive to stellar structure and consequentially are the most difficult to model correctly.

Another powerful benefit of including the covariance hyperparameters is the result of their ability to quantify and account for model-data mismatch. When a data spectrum is fit with a high-quality spectral library, the pixel residuals are less likely to be correlated, and the covariance structure will have a smaller amplitude and correlation length. This means that the inference on the stellar parameters will be more precise, in the same way that high quality data allows a more precise result. However, if the systematic mismatch between the data and model is large, then the hyperparameters will be larger and the precision of the stellar parameters will naturally inflate to respect the quality of model-data fit.

The benefits of using covariance hyperparameters extend beyond the use case of fitting a single stellar spectrum. If we fit many stars with the same set of synthetic models, we can use the structure of the covariance matrix to improve the models themselves (see §2.6).

M dwarf residual

TODO: This third and final covariance kernel example might push this section length over the top. However, I think this application is the most exciting and could be the "killer app" that would drive people to adopt our framework given the computational overhead.
TODO: I think to properly design and test this kernel, I will need to actually explore fitting an M dwarf spectrum. For now I have just been looking at the figure by Andrew Mann. However, I have high hopes!

A powerful application of covariance kernels is that data-model mismatch can be accounted for even if it takes on a messy, not-easily-parameterized form like the Gaussian line example. We can not just inflate the noise in a specific region, but we also taper the kernel for more complicated structure. For example, consider a typical residual pattern from the spectroscopic fit of an M dwarf. There is a complicated pattern of high and low residuals, or there might even be a region of large offset, such that features are correlated on a large scale but not so well on the near scale. Therefore we need a kernel which can transition between these two. We multiply the existing kernel by the squared exponential kernel

$$k_{\rm exp}(r|\,h) = \exp\left(\frac{-r^2}{2h^2}\right) \tag{19}$$

where h is a bandwidth. If h is small, then there will be high-frequency structure. If h is large, then only low-frequency structure will remain.

$$k(\lambda_i, \lambda_j | h, \boldsymbol{\theta}_{{\rm line}_0}) = k_{\rm exp}(r|\,h)\; k_{\rm line}(\lambda_i, \lambda_j | \boldsymbol{\theta}_{{\rm line}_0}) \tag{20}$$

TODO: Show how the kernel can accounts for complicated M dwarf structure where there is large, messy mismatch. Produce figures analogous to Figures 4 and 5. Also add an example of M dwarf mismatch to Figure 2.

2.5. Exploring the posterior

We use a Markov chain Monte Carlo (MCMC) algorithm coupled to a Gibbs sampler to explore the posterior distribution (Equation 3). The MCMC routine efficiently explores a high dimensional probability space using a stochastic, iterative approach. The Gibbs sampler provides a way to easily sample a large number of parameters in a simple, organized manner by sampling only a subset of parameters at any given time, but then rotating among all of the subsets of parameters. For more on MCMC and Gibbs samplers see Gelman et al. (2013, ch 11) and the references therein.

In addition to the stellar parameters $\boldsymbol{\theta}_\star$, we have introduced several nuisance parameters for calibration and residual modeling. These nuisance parameters have a logical hierarchical structure. At the lowest level of the hierarchy, a single order of an echelle spectrum has N regions of high covariance $\boldsymbol{\theta}_{\rm lines} = \{\boldsymbol{\theta}_{{\rm line}_1}, \boldsymbol{\theta}_{{\rm line}_2}, \ldots, \boldsymbol{\theta}_{{\rm line}_N}\}$. If we wish to evaluate the probability of the parameters of a specific line (e.g., line 1) conditional on the current values of all other lines, then we denote this by $p(\boldsymbol{\theta}_{\rm line=1} | \boldsymbol{\theta}_{\rm lines \neq 1})$. The collection of nuisance parameters for a specific order (e.g., order 1) is the aggregate parameter $\boldsymbol{\theta}_{{\rm order}_1} = \{\boldsymbol{\theta}_{\rm Cheb}, \boldsymbol{\theta}_{\rm global}, \boldsymbol{\theta}_{\rm lines}\}$. Taken together, the aggregated nuisance parameters for all of the N orders are stored in $\boldsymbol{\theta}_{\rm orders} = \{\boldsymbol{\theta}_{{\rm order}_1}, \boldsymbol{\theta}_{{\rm order}_2}, \ldots, \boldsymbol{\theta}_{{\rm order}_N}\}$. Because the nuisance parameters for a single echelle order are independent from the nuisance parameters for any other echelle order, we have $p(\boldsymbol{\theta}_{\rm order=1} | \boldsymbol{\theta}_{\rm orders \neq 1}) = p(\boldsymbol{\theta}_{\rm order=1})$.

The Gibbs sampler iteratively explores this hierarchy of parameters. At each level of the hierarchy, corresponding to a different subset of nuisance parameters, we use the Metropolis-Hastings algorithm to propose a new subset of parameters and then either accept or reject them. This process of proposal and acceptance/rejection is called “sampling.” The Gibbs sampler rotates between sampling in different subsets

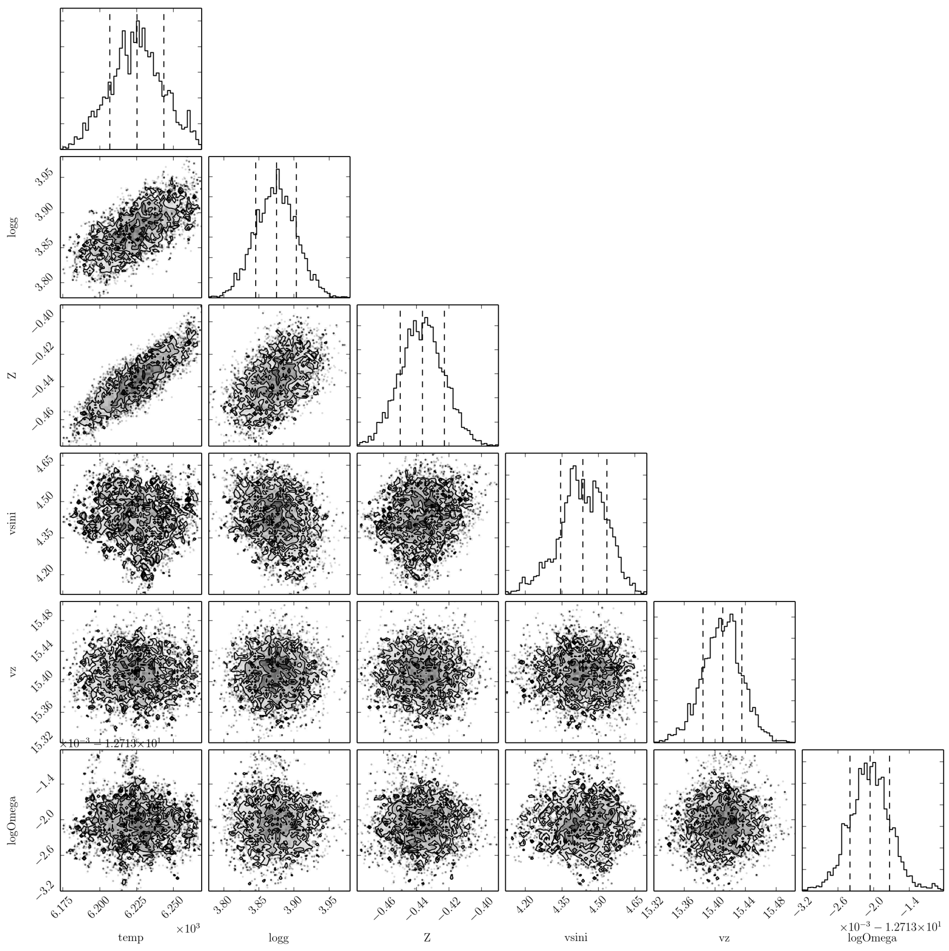

Fig. 6.— The posterior probability function of the stellar parameters for WASP-14, an F star, as explored by the MCMC Gibbs sampler. These stellar parameters are marginalized over the Chebyshev and noise parameters. Made with `triangle.py`[2]
TODO: In the updated figure I won't show v_z or Ω so the font is easier to see. Many more samples so posteriors look smooth.

https://github.com/dfm/triangle.py

of parameters, eventually sampling all of the nuisance parameters at a certain cadence. To initialize the MCMC algorithm, we make a reasonable guess for the starting parameters. We use existing spectral types in the literature for the stellar parameters, flat Chebyshev polynomials (no flux-calibration correction), and no covariance structure in the residuals (the global covariance parameters are set to zero and no line kernels are instantiated). We use superscripts to denote iterations of the MCMC algorithm. i denotes the parameters from the current iteration and $i-1$ denotes the previous iteration. The Gibbs sampler rotates among subsets of the parameters as follows figs/global_posteriors.pdf

figs/line_posteriors.pdf

Fig. 7.— **Left**: The posterior probability function of the global covariance parameters, as explored by the MCMC Gibbs sampler, marginalized over the stellar and line noise parameters. **Right**: The posterior probability function of the line parameters, marginalized over the stellar and global noise parameters.

1. Sample in the stellar parameters. For each proposal of the Metropolis-Hastings algorithm, generate a model spectrum following the steps in §2.2. When deciding whether or not to accept the parameters, the Gibbs sampler evaluates $p(\boldsymbol{\theta}_\star^i | \boldsymbol{\theta}_{\rm orders}^{i-1})$.
2. For each order of the echelle spectrum
   (a) Sample in the Chebyshev polynomial parameters. Adjust the spectrum as detailed in §2.2.Gibbs sampler evaluates $p(\boldsymbol{\theta}_{\rm Cheb}^i | \boldsymbol{\theta}_\star^i, \boldsymbol{\theta}_{\rm global}^{i-1}, \boldsymbol{\theta}_{\rm lines}^{i-1})$.
   (b) Sample in the global covariance parameters. Adjust the covariance matrix C as described in §2.4. Gibbs sampler evaluates $p(\boldsymbol{\theta}_{\rm global}^i | \boldsymbol{\theta}_\star^i, \boldsymbol{\theta}_{\rm Cheb}^i, \boldsymbol{\theta}_{\rm lines}^{i-1})$.
   (c) Check to see whether the algorithm should instantiate/delete new/old line kernels
   (d) For each line kernel k
      i. Sample in the parameters for each line, conditional on the parameters for all the other lines. Adjust the covariance matrix C as described in §2.4. Gibbs sampler evaluates $p(\boldsymbol{\theta}_{{\rm line}_k}^i | \boldsymbol{\theta}_{{\rm lines} \neq k}^{i-1})$.

At each stage of the Gibbs sampler, the likelihood function (Equation 3) is evaluated for a proposal of a particular subset of parameters, conditional on the current values of all the other parameters. This hierarchical parameter structure and the kernel parameterization of the covariance matrix influences how the algorithm may be efficiently implemented in code. For a typical optical spectrum with $\gtrsim$1,000 pixels, the most computationally intensive step of the likelihood evaluation is usually the matrix product $R^T C^{-1} R$. Since we have designed the covariance matrix to be sparse, we can use optimized sparse matrix algorithms which are much faster and memory efficient than dense matrix operations. Because we are not interested in the matrix inverse C^{-1} by itself, but rather the product with the residual vectors, we can use efficient routines for solving linear systems to bypass the computationally difficult step of matrix inversion. Additionally, because the covariance matrix is positive semi-definite, we can use the Cholesky factorization of the matrix to optimize the evaluation of the matrix product. Once the covariance matrix is factorized, any subsequent evaluation of the matrix product for different residual vectors R is extremely rapid. This makes the $\boldsymbol{\theta}_\star$ and $\boldsymbol{\theta}_\text{Cheb}$ steps of the Gibbs sampler extremely fast. When we sample in the nuisance parameters which affect the covariance matrix, we must redo the Cholesky factorization of C for each update. However, because we designed the kernels to deliver a sparse matrix these operations are efficient enough to be used with an MCMC algorithm. We use the high-performance `SuiteSparse/CHOLMOD`[3] library to implement the sparse matrix and Cholesky factorization operations (Chen et al. 2008; Davis & Hager 2009) and the Metropolis-Hastings sampler included in the `Python` MCMC package `emcee` (Foreman-Mackey et al. 2012).

We run the MCMC Gibbs sampler for many iterations until the estimate of the posterior distribution has converged. To check that the chain is not stuck in a local maximum of the posterior, we redo the MCMC run many times with different starting parameters, to ensure that the algorithm converges to the same global maximum. A major advantage of using the MCMC algorithm to explore the multidimensional probability space is that it provides numerical samples in each dimension. Therefore, marginalizing out a parameter (i.e., numerically integrating over a dimension in probability space) is as simple as combining all of the samples in this dimension. This enables us to present a posterior of the stellar parameters $\boldsymbol{\theta}_\star$ (Figure 6) which has been marginalized over all of the nuisance hyperparameters. This posterior is the final estimate of the stellar parameters which *incorporates* any inherent uncertainty due to model mismatch (via the covariance hyperparameters) and flux-calibration (via the Chebyshev polynomials).

The benefit of including nuisance parameters and then marginalizing over them is that we can self-consistently model the uncertainty inherent to any spectrum while naturally capturing any degeneracy between the model parameters. If stellar parameters are estimated using a method which ignores these nuisance parameters (e.g., by-eye fitting) and then the astronomer arbitrarily inflates the parameter uncertainties to reflect intuition, the degeneracy between parameters is artificially destroyed. This leads to estimates of stellar parameters with more uncertainty than necessary.
TODO: Since this point is better carried with an example or test using the global covariance structure and not using it (see Test 1), we might want to defer this discussion to the testing section.

[3]`http://www.cise.ufl.edu/research/sparse/cholmod/`

2.6. Applications

By cataloguing the covariance structure of the residuals, especially those generated from strong spectral line mismatch, we collect valuable information about the quality of the synthetic spectra. After fitting many stars, the accumulated knowledge of the data-model mismatch can be used when fitting a new star. The previous structure of the covariance matrix allows us to set priors on what the covariance in certain regions of the spectrum should be, which will speed convergence for this new star. Additionally, after fitting several stars, the average value of the covariance matrix will inform us about the quality of specific spectral lines in the synthetic models.

Linking the covariance matrices of stars could be done serially, where the aggregate covariance matrix of all previous fits is used as a prior for the current iteration. Or, the covariance matrices could instead be linked hierarchically, in that the parameters describing the depth and width of a line residual for a specific star a_1 and σ_1, are modeled as coming from of a population of possible depths and widths for a given spectral type. Each stellar spectrum will have a slightly different realization of a spectral line, which will have some scatter about the average residual height. Linking the covariance matrix between spectra of similar stars allows us to grow more confident in our assessment that certain synthetic spectral lines are indeed outliers and should be appropriately down-weighted. In turn, as we become more certain of the weights, the stellar posteriors will become narrower and make our estimates of the stellar parameters more precise. This ability of the model to mutually inform sets of parameters is one of the major advantages of hierarchical Bayesian analysis (Kruschke 2010).

Once determined, this average covariance matrix could be delivered to the communities that created the synthetic libraries, which would enable them to rapidly pinpoint and correct defects in the synthetic models.

TODO: more delicate phrasing? Alternatively, we could correct the models ourselves by using the chain of logic and mathematical post-processing that we used to created the synthetic model spectrum to reverse-engineer what the behavior of the raw synthetic spectrum *should* be, at the raw $R \gtrsim 100{,}000$ resolution. This fundamental application of machine learning would enable us to create our own library of semi-empirical stellar models. Rather than simply assembling an empirical spectral library using only real stellar spectra, this combined approach is more powerful because the stellar atmospheres provide an actual anchor point of fundamental stellar parameters tied down by the laws of stellar physics.

3. Tests

Here are some ideas of which tests we might want to show.

Test 1 : Fitting with and without the global covariance kernel to show how the width of the posteriors nicely inflates to reasonable uncertainties.

Test 2 : I think we should fit both an F/G star (WASP-14) and a K/M star (TBD) with both the Kurucz and PHOENIX spectra. This will show a few things

- Spectral parameters can vary by a large margin depending on which spectral library you use (200 K or more).
- Both spectral libraries have stars that they perform better and worse on.
- This will be reflected in the increased level of global noise, and number of "bad" regions that have been instantiated.

REFERENCES

Buchhave, L. A., Latham, D. W., Johansen, A., et al. 2012, Nature, 486, 375

Chen, Y., Davis, T. A., Hager, W. W., & Rajamanickam, S. 2008, ACM Trans. Math. Softw., 35, 22:1

Davis, T. A., & Hager, W. W. 2009, ACM Trans. Math. Softw., 35, 27:1

Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2012, ArXiv e-prints, arXiv:1202.3665 [astro-ph.IM]

Gelman, A., Carlin, J., Stern, H., et al. 2013, Bayesian Data Analysis, Third Edition, Chapman & Hall/CRC Texts in Statistical Science (Taylor & Francis)

Gray, D. F. 2008, The Observation and Analysis of Stellar Photospheres

Husser, T.-O., Wende-von Berg, S., Dreizler, S., et al. 2013, A&A, 553, A6

Kruschke, J. 2010, Doing Bayesian Data Analysis: A Tutorial Introduction with R (Academic Press)

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

Valenti, J. A., & Piskunov, N. 1996, A&AS, 118, 595

This version of the paper was generated from a git repository available at `http://github.com/iancze/StellarSpectra/` with git hash `6cc06b1 (2014-05-13)`.

This preprint was prepared with the AAS LATEX macros v5.0.

Symbol	Description
i	index specifying a pixel
λ_i	wavelength corresponding to a given pixel i
$\boldsymbol{\theta}_{\star,\mathrm{grid}}$	fundamental stellar parameters, $T_{\mathrm{eff}}, \log(g), [\mathrm{Fe}/\mathrm{H}], [\alpha/\mathrm{Fe}]$
	that parameterize a synthetic spectrum from the grid
$\boldsymbol{\theta}_{\star,\mathrm{post}}$	stellar parameters $v \sin i$, v_z, A_V, and R^2/d^2 that
	are applied during “post processing” of the synthetic spectrum
$\boldsymbol{\theta}_{\star}$	$\{\boldsymbol{\theta}_{\star,\mathrm{grid}}, \boldsymbol{\theta}_{\star,\mathrm{post}}\}$
$f_{\lambda,\mathrm{inst}}(\lambda)$	data spectrum
$f_{\lambda,\mathrm{synth}}(\lambda)$	synthetic spectrum
$\boldsymbol{\theta}_{\mathrm{Cheb}}$	the set of Chebyshev polynomial coefficients $\{c_0, c_1, \ldots, c_N\}$
$\boldsymbol{\theta}_{\mathrm{line}_1}$	
$\boldsymbol{\theta}_{\mathrm{lines}}$	
$\boldsymbol{\theta}_{\mathrm{order}_1}$	
$\boldsymbol{\theta}_{\mathrm{orders}}$	
$\boldsymbol{\theta}$	the parameters $\{\boldsymbol{\theta}_{\star,\mathrm{grid}}, \boldsymbol{\theta}_{\star,\mathrm{post}}, \boldsymbol{\theta}_{\mathrm{N}}\}$ that completely describe a model spectrum
D_i	data flux for a given pixel, $D(\lambda_i)$
$\vec{D}$	data vector comprised of all D_i, $i = \{1, \ldots, N\}$
M_i	model flux for a given pixel, $M(\lambda_i \vert \boldsymbol{\theta})$
$\vec{M}$	model vector comprised of all M_i, $i = \{1, \ldots, N\}$
σ_i	Poisson noise for a given pixel i
R_i	residuals $D_i - M_i$
R	residual vector $\vec{D} - \vec{M}$
C	covariance matrix
σ_{ij}	element in the covariance matrix
$r(\lambda_i, \lambda_j)$	radial distance in wavelength space corresponding to Δv
k_{global}	global covariance kernel
k_{line}	regional covariance kernel

Table 1: Nomenclature used in this document.