Summer Internship Programme - 2023

June 1st - July 31th, 2023

Project: Spectral Characterisation of Metal-Poor Dwarf Stars

Mentor: Dr. Arvind Singh Rajpurohit

Rushiraj Gadhvi

(CSAI, Plaksha University, Mohali)

CERTIFICATE

This is to certify that the project entitled **“Spectral Characterisation of Metal-Poor Dwarf Stars”** is the original work carried out by **Rushiraj Gadhvi**, a student of Plaksha University, Mohali, during the tenure from **1st June to 31st July,2023** at the **Physical Research Laboratory (PRL)** under my guidance and supervision up to a satisfactory level. This work is entirely original and has not been submitted to any other program, degree, or diploma.

[illegible]
29/07/2023

Dr. Arvind Singh Rajpurohit
Astronomy and Astrophysics
Physical Research Laboratory

Introduction

In the vast tapestry of the cosmos, stars serve as celestial storytellers, revealing the intriguing history of our universe. Among these cosmic actors, a class of stars known as M dwarfs captivates astronomers and researchers alike. M dwarfs, the most abundant stellar type in the Milky Way galaxy, hold a special place in our exploration of the cosmos.

Here, we are particularly interested in Metal-poor M dwarfs that are members of the stellar family characterized by their spectral class M, which represents small size, cool, and stars with low luminosity. M-class classified bodies exhibit distinct spectral characteristics, including very weak hydrogen absorption lines and prominent molecular lines, notably titanium monoxide (TiO). These spectral features contribute to their unique identification and classification within the M-class of stars.

Spectral analysis involves the study of a star's light, split into its constituent wavelengths, to unveil details about its composition, temperature, and other characteristics. However, obtaining high-quality observational data can be challenging due to various factors, such as instrumental limitations, noise, and complex atmospheric effects. Synthetic spectrum spectral analysis offers an alternative approach to overcome these limitations and extract accurate information from the observed spectra.

By comparing observed spectra with synthetic spectra generated using theoretical models, we can fine-tune various parameters to achieve a better fit. This process allows for precise determination of stellar properties, including effective temperature, metallicity, surface gravity, and abundances of different chemical elements. Accurate characterization of these parameters is crucial for understanding the evolution, formation, and potential habitability of M dwarf stars.

Data Preprocessing

Before go any further, the spectrum needs to be preprocessed, following steps must be sequentially implemented on both the observation and synthetic data:

1. **Gaussian Convolution and Telluric Range Removal**

   To eliminate undesired noise and achieve a smoother spectrum, the observed spectrum undergoes Gaussian convolution. Similarly, the synthetic spectrum is convolved with a Gaussian function to simulate the blurring effects caused by the instrument. The width of the Gaussian function is appropriately tuned according to the instrument's spectral resolution, ensuring that the data integrity is maintained without compromise. Furthermore, it is necessary to eliminate any gaps, regions with extreme noise, and telluric regions from both the synthetic and observed spectra.

2. **Continuum Fitting and Normalization**

   Continuum fitting was performed on both spectra using mathematical functions like polynomials of various degrees or splines. This step involved estimating the continuum level, which enhanced the visibility of spectral features and facilitated the identification and measurement of absorption or emission lines. Additionally, to ensure comparability, both spectra were normalized individually, scaling them to a common reference level. Normalization enabled accurate quantitative analysis by removing any differences in overall intensities between the two spectra.

3. Radial Velocity (RV) Correction

RV correction is employed to align the observation and synthetic spectra by compensating for any Doppler shifts arising from the motion of the observer, the target, or both. To achieve this alignment, cross-correlation techniques, including cross-correlation with a template, are utilized. These techniques enable the identification and alignment of the prominent features present in the spectra, ensuring that corresponding features are correctly matched. By applying RV correction, the spectra are brought into the same reference frame, allowing for accurate comparison and analysis of the spectral characteristics.

4. Data Interpolation

In order to enable a comparison between the observation and synthetic spectra, it is necessary to match their data resolutions. To achieve this, the cubic interpolation method is employed, allowing for the interpolation of data points in the synthetic spectrum to align with the wavelength points of the observed spectrum. This process ensures that both spectra have the same data point density, enabling a meaningful and accurate comparison of their spectral features and characteristics

Using MCMC (Markov Chain Monte Carlo) for Parameter Estimation

In the context of estimating parameters such as effective temperature, surface gravity, and metallicity by comparing synthetic spectra with observational spectra, the **Markov Chain Monte Carlo (MCMC)** method is commonly employed. MCMC is a powerful computational technique that allows for efficient exploration of high-dimensional parameter spaces and provides a statistical framework to estimate parameter uncertainties.

MCMC is particularly useful in this scenario because it can handle complex likelihood functions and efficiently explore the parameter space, even when it is multimodal or has non-trivial correlations between parameters. It provides a robust way to sample from the posterior distribution, which represents the probability distribution of the model parameters given the observed data.

The **emcee** library is a popular implementation of MCMC in Python that is specifically designed for Bayesian inference. It implements an ensemble sampler called the affine-invariant ensemble sampler, which is efficient in exploring the parameter space and has the advantage of being less sensitive to tuning parameters compared to other MCMC algorithms.

Here's a brief overview of how MCMC works in the emcee library:

1. **Initial Parameter Sampling**:

   We can initialize the walkers in the MCMC algorithm using the three different methodologies describe below; The choice of initialization method is crucial as it can impact the convergence rate of the MCMC chain

   - **Self-Defined Initial Parameters**: You have the freedom to manually set the initial parameter values for each walker in the ensemble. This approach allows you to provide educated guesses based on prior knowledge or reasonable assumptions about the parameter values. However, it is essential to ensure that the chosen values are within a reasonable range and not biased towards a specific region of the parameter space.

   - **Uniform Distribution in the Parameter Space**: Another option is to randomly sample initial parameter values from a uniform distribution within the parameter space. This approach ensures that the walkers start from a diverse set of initial positions, facilitating a more efficient exploration of the parameter space. It prevents the walkers from being concentrated in a specific region and helps to overcome potential biases in the analysis.

- **Maximum Likelihood Estimation (MLE)**: Alternatively, you can use the Maximum Likelihood Estimation to obtain initial parameter values. MLE involves finding the parameter values that maximize the likelihood of the observed data given the model. This approach provides a data-driven initialization but requires additional computational time to compute the maximum likelihood estimates. It is important to note that MLE might not always be suitable if the likelihood function has multiple local maxima or if there are strong parameter correlations.

2. **Likelihood Evaluation**: For each set of parameters, the synthetic spectrum is computed based on the assumed values of effective temperature, surface gravity, and metallicity. The likelihood of the observed spectrum given the synthetic spectrum is evaluated. The likelihood function quantifies the agreement between the model and the data, considering uncertainties in the observations.

3. **Markov Chain Update**: The MCMC algorithm then proceeds to update the walkers' positions in the parameter space. It generates a proposal distribution to suggest new parameter values for each walker. The proposal distribution is typically based on the current position and is designed to ensure efficient exploration of the parameter space.

4. **Acceptance/Rejection**: The proposed parameter values are accepted or rejected based on a criterion that incorporates both the likelihood of the data given the proposed parameters and the prior information about the parameters. This criterion is defined by the Metropolis-Hastings algorithm, which ensures that the MCMC samples converge to the desired posterior distribution.

5. **Sampling and Burn-in**: The MCMC algorithm iterates over many steps, updating the walkers' positions and accepting/rejecting proposals. During the initial steps, known as the burn-in phase, the walkers explore the parameter space to reach a region of high probability. These initial samples are discarded as they do not represent the desired posterior distribution.

6. **Convergence and Autocorrelation**: After the burn-in phase, the MCMC samples are considered to be drawn from the posterior distribution. Convergence diagnostics, such as examining the autocorrelation of the samples, can be employed to assess the convergence and ensure sufficient exploration of the parameter space.

7. **Parameter Estimation**: Once the MCMC samples have converged, statistical summaries such as the mean, median, and credible intervals can be computed from the samples to estimate the parameters of interest (e.g., effective

temperature, surface gravity, and metallicity). These summaries provide both point estimates and uncertainties associated with the parameters.

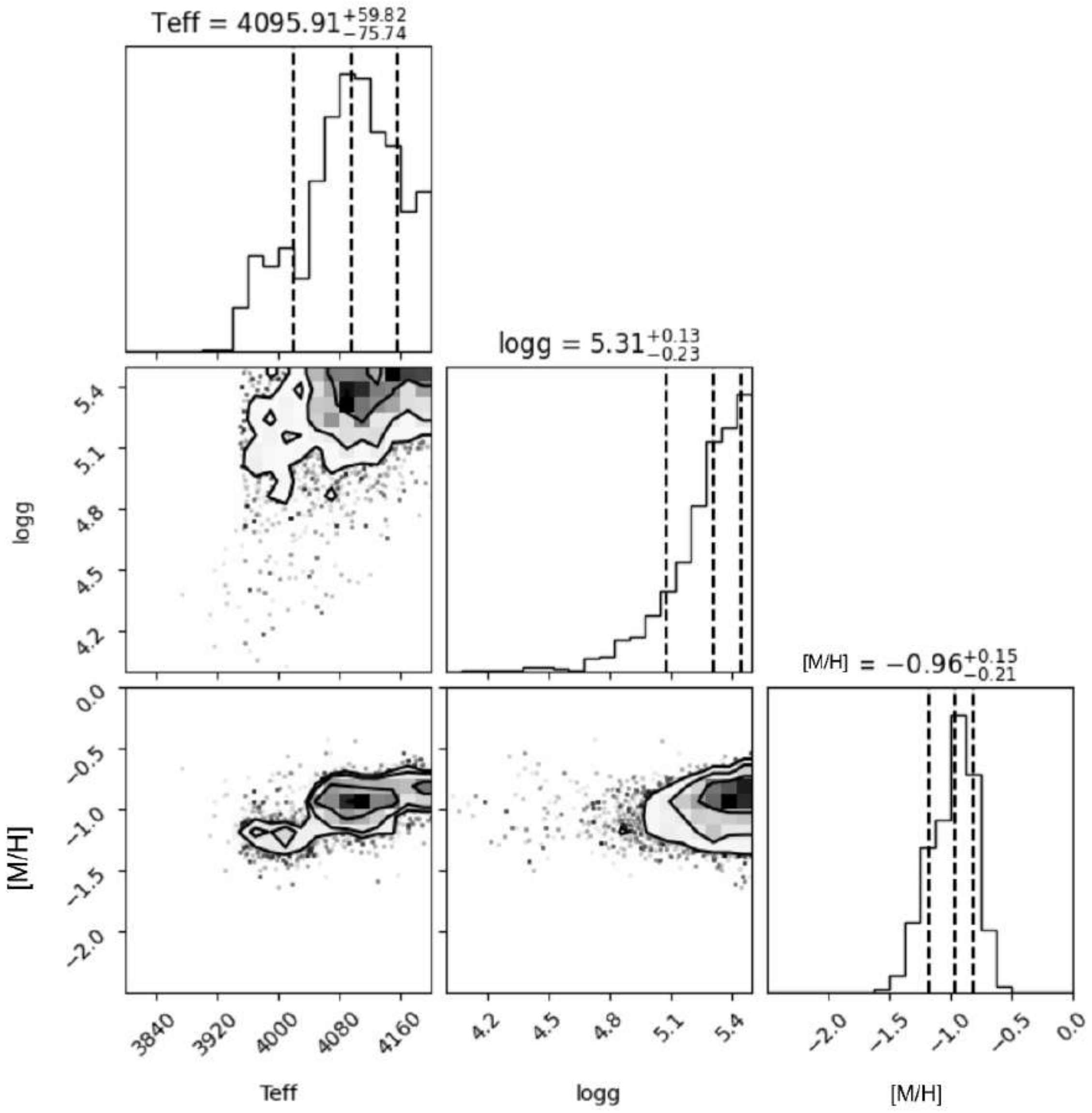
Improving Data Interpolation Efficiency

In our synthetic model spectrum, we have generated a grid that spans effective temperatures ranging from **3800 K to 4200 K** in increments of **100 K**. Additionally, we have considered a range of metallicities from **-2.5 to 0** in increments of **0.5**, and surface gravity values varying from **4.0 to 5.5** in increments of **0.5**. This grid provides a comprehensive coverage of parameter space for our model. To efficiently handle parameter estimation using MCMC, it is impractical to directly interpolate from the entire grid for each step and walker. Such an approach would be computationally intensive and time-consuming. Therefore, we adopt a more efficient strategy.

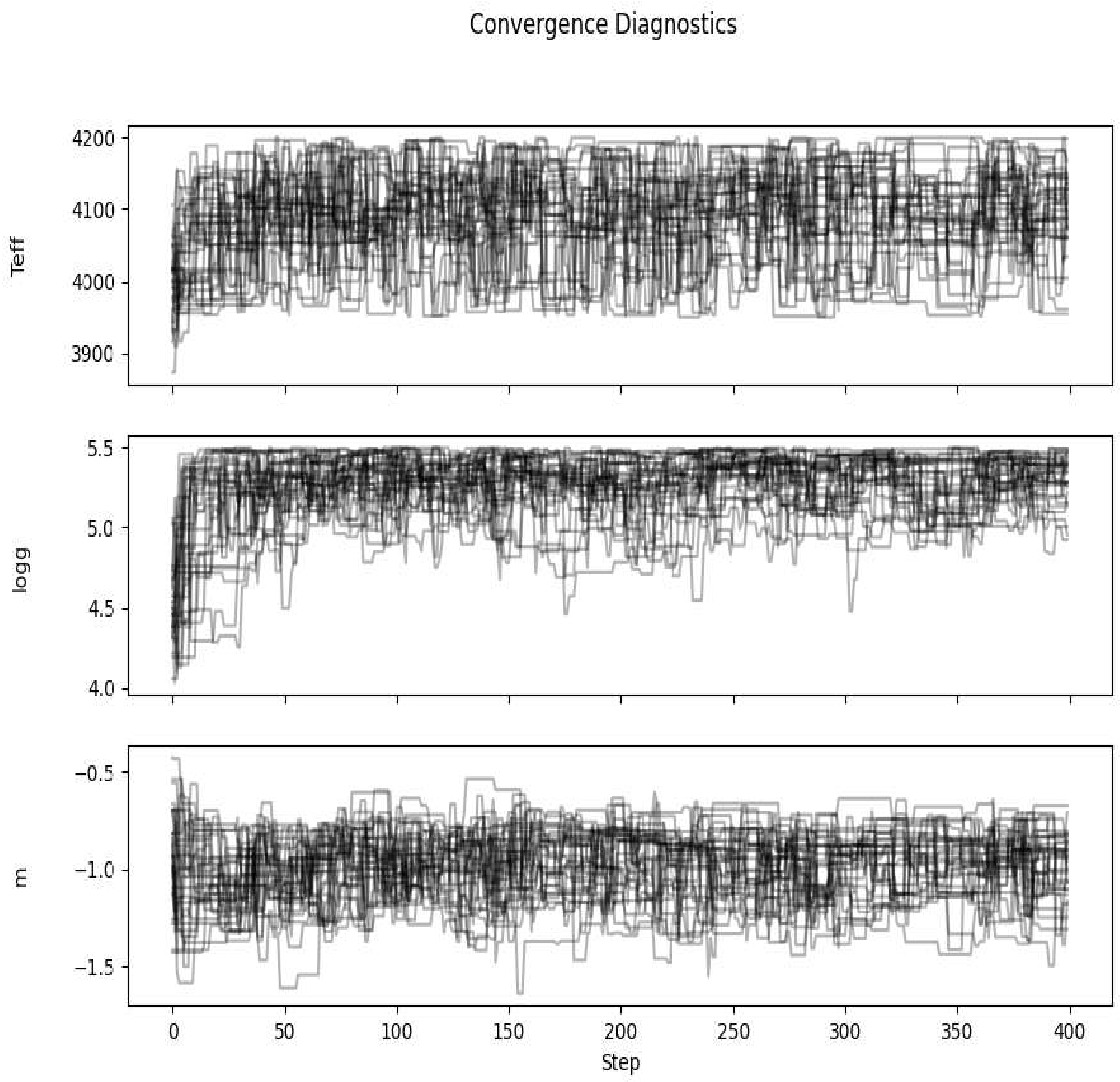
We employ a technique where we measure the Euclidean distance between the parameter values of interest (asked points) and the available models in our grid. From this distance calculation, we select the nearest 10 models. By choosing a smaller subset of models, we reduce computational complexity while still ensuring reasonable accuracy. Once we have identified the nearest 10 models, we create an interpolated grid based on this subset. This approach significantly speeds up the interpolation process while maintaining sufficient accuracy for the parameter estimation. This step is crucial in optimizing computational resources and saving valuable time during the MCMC procedure.

By adopting this technique, we strike a balance between accuracy and computational efficiency, allowing us to efficiently sample and generate values for any parameter requested by the MCMC algorithm.

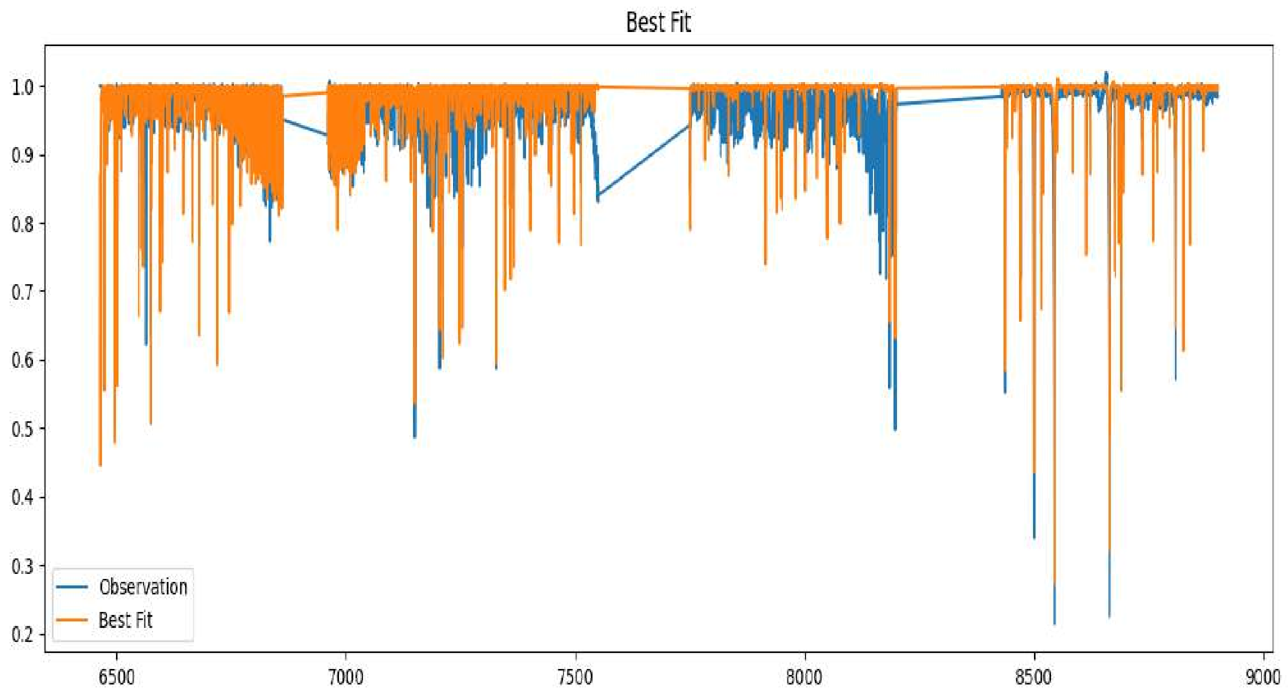
Results of MCMC runs



The corner plot shown above was created using a well-converged chain consisting of **400 steps** and **30 walkers**. The MCMC code took approximately **7 hours** to run and produce these results. The convergence of the chain indicates that a sufficient exploration of the parameter space has been achieved, allowing for reliable estimates of the parameters of interest.



Following this we generated the best fit graph, using the parameter estimation received from the MCMC run:



Storing MCMC data

By default, the entire Markov Chain Monte Carlo (MCMC) run is saved in a Hierarchical Data Formats (HDF) file named data.h5. This file contains all the necessary information to analyze the entire run and generate various statistics and graphs. This feature is particularly useful in cases where a run is interrupted due to technical issues. Instead of recomputing everything from scratch, researchers can utilize the data file to resume the process, saving significant time and effort.

Making MCMC even more faster

To expedite the MCMC process and reduce computation time, one effective approach is to leverage multiprocessing techniques. By utilizing multiple processors or cores, multiprocessing allows for parallel execution of the MCMC algorithm, significantly accelerating the overall computation.

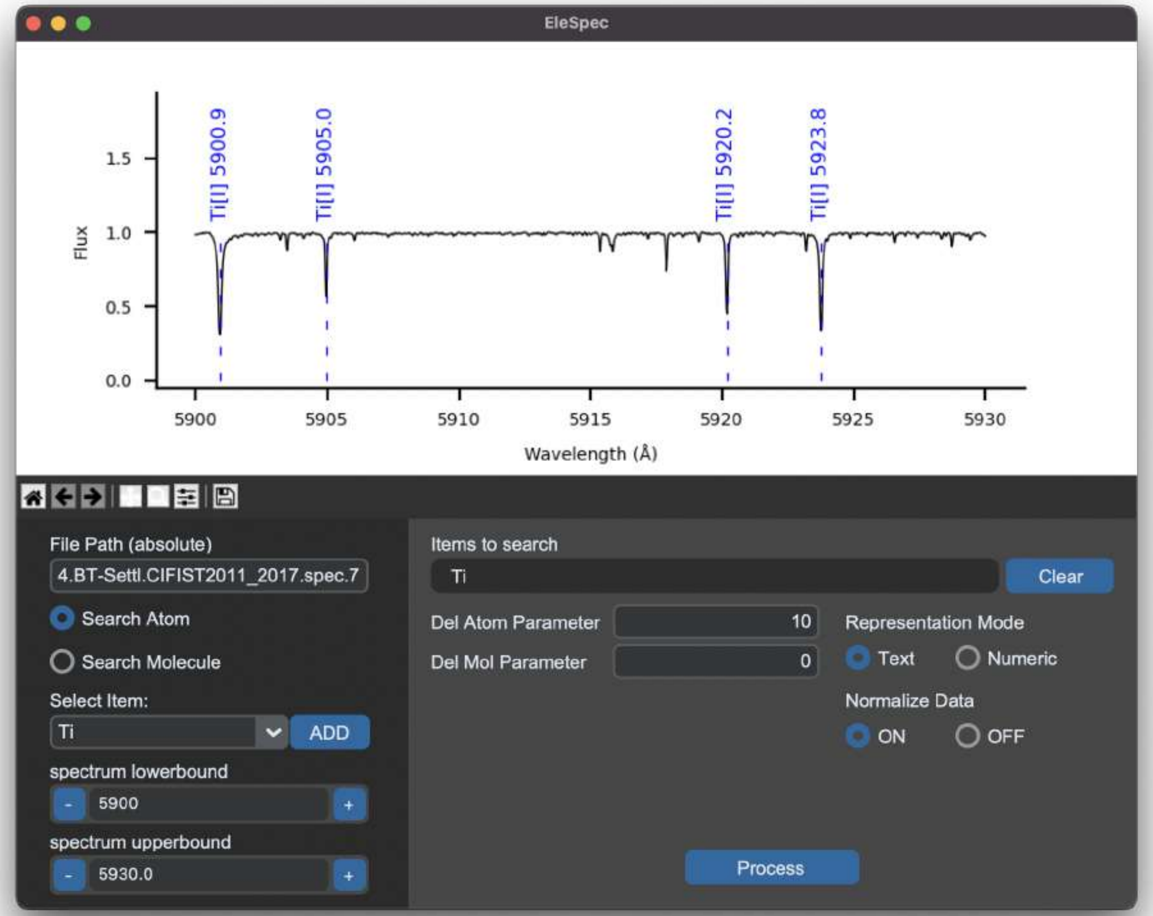
The multiprocessing strategy involves distributing the workload across multiple processes, with each process running a separate walker. By utilizing multiprocessing, multiple walkers can be run in parallel, exploring different regions of the parameter space simultaneously. Each walker operates independently, allowing for efficient exploration and convergence to the target distribution. We can observe instantly that the parallel execution of walkers drastically reduces the total execution time required for MCMC sampling when compared to a sequential implementation.

It is important to ensure that the MCMC algorithm is designed to take advantage of multiprocessing. This may involve adapting the algorithm to handle parallel computations, such as using lock-free data structures or employing appropriate communication and synchronization mechanisms. Additionally, the efficient utilization of multiprocessing may require careful consideration of memory management and load balancing to avoid resource contention or imbalances among the processes.

Building EleSpec

The primary objective of the software EleSpec is to simplify the process of visualization and exploration of synthetic spectra in an intuitive user interface, facilitating the detection and identification of atoms and molecules.

Searching for Ti in the spectra range (5900 - 5930 Å):



File Path (absolute): 4.BT-Settl.CIFIST2011_2017.spec.7

[✓ Search Atom] Search Molecule

Select Item: Ti

spectrum lowerbound: 5900

spectrum upperbound: 5930.0

Items to search: Ti

Del Atom Parameter: 10

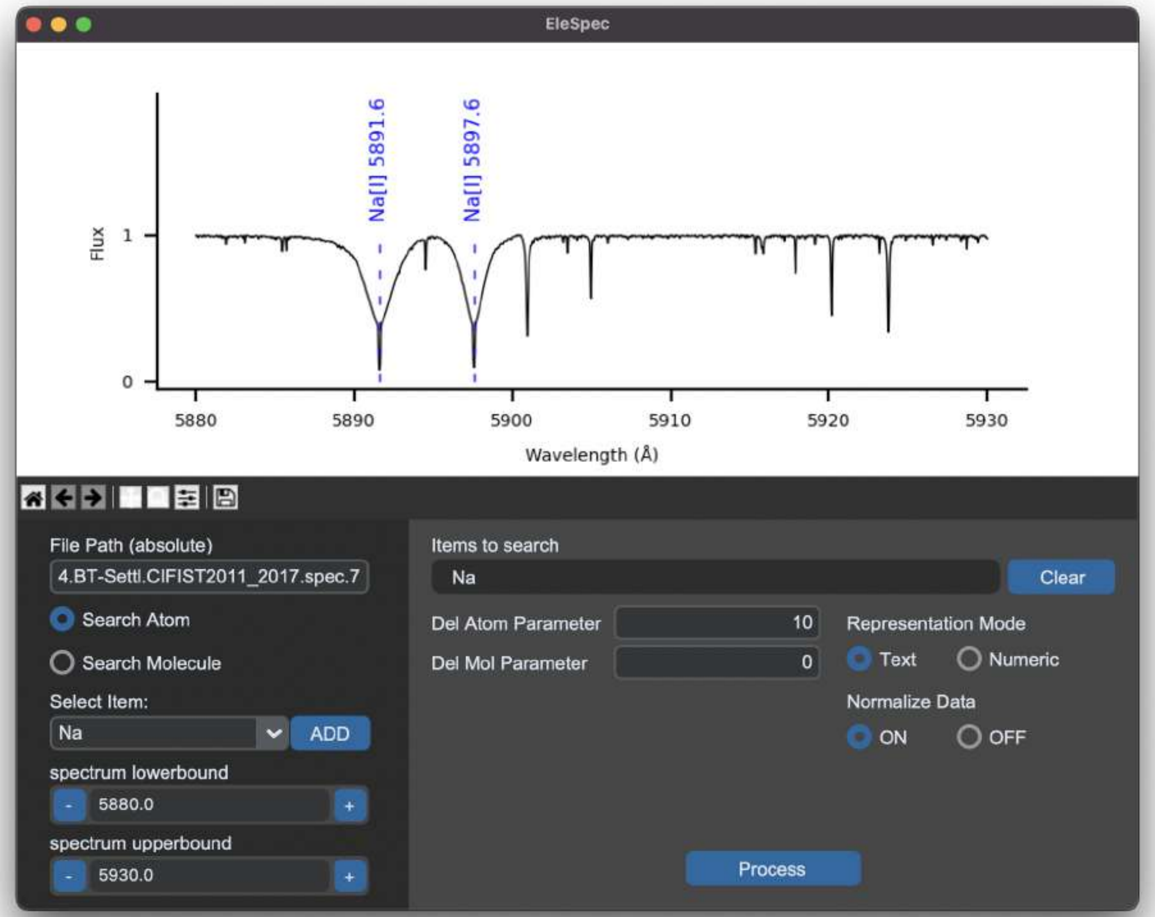
Del Mol Parameter: 0

Representation Mode: [✓ Text] Numeric

Normalize Data: [✓ ON] OFF

EleSpec offers users the liberty to input spectra in any text extension of their choice. Whether it be CSV, TXT, or other common formats, the software seamlessly integrates the data. Furthermore, EleSpec supports xz compression, allowing users to efficiently manage and analyze large datasets without sacrificing performance or storage.

Searching for Na in the spectra range (5880 - 5930 Å):



File Path (absolute): 4.BT-Settl.CIFIST2011_2017.spec.7

[✓ Search Atom] Search Molecule

Select Item: Na

spectrum lowerbound: 5880.0

spectrum upperbound: 5930.0

Items to search: Na

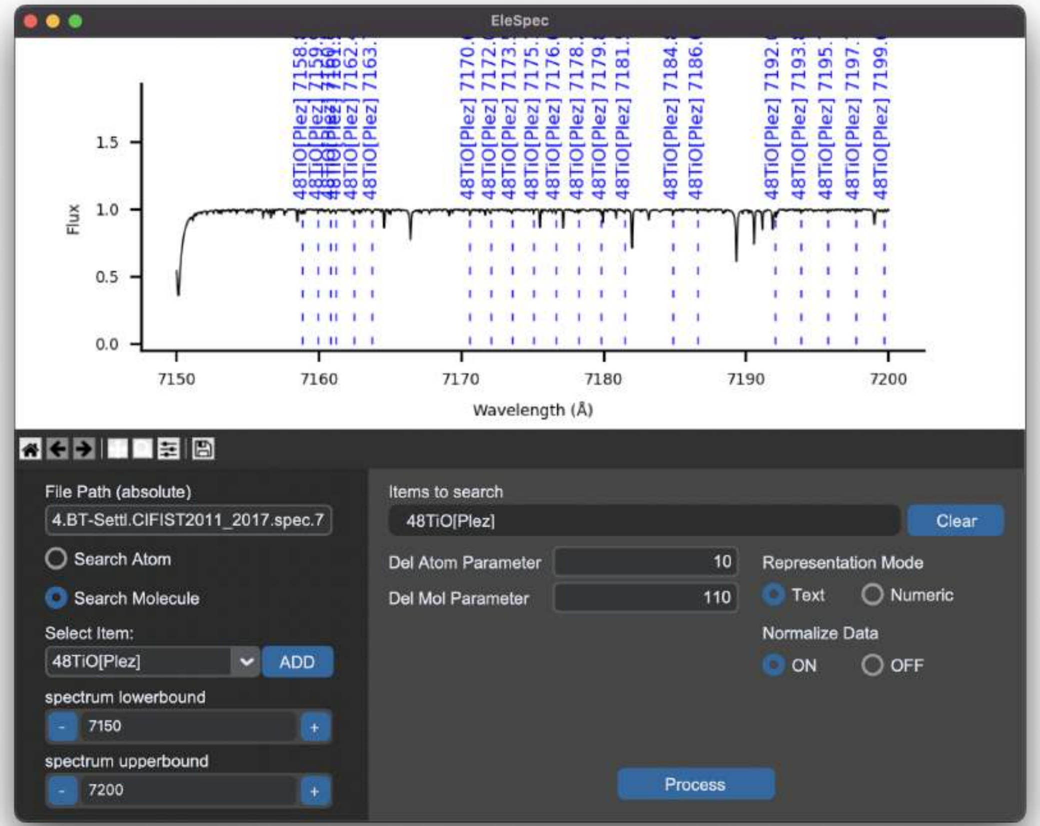
Del Atom Parameter: 10

Del Mol Parameter: 0

Representation Mode: [✓ Text] Numeric

Normalize Data: [✓ ON] OFF

Searching for 48TiO in the spectra range (7150 - 7200 Å):



File Path (absolute): 4.BT-Settl.CIFIST2011_2017.spec.7

Search Atom [✓ Search Molecule]

Select Item: 48TiO[Plez] ADD

spectrum lowerbound: 7150

spectrum upperbound: 7200

Items to search: 48TiO[Plez] Clear

Del Atom Parameter: 10

Del Mol Parameter: 110

Representation Mode: [✓ Text] Numeric

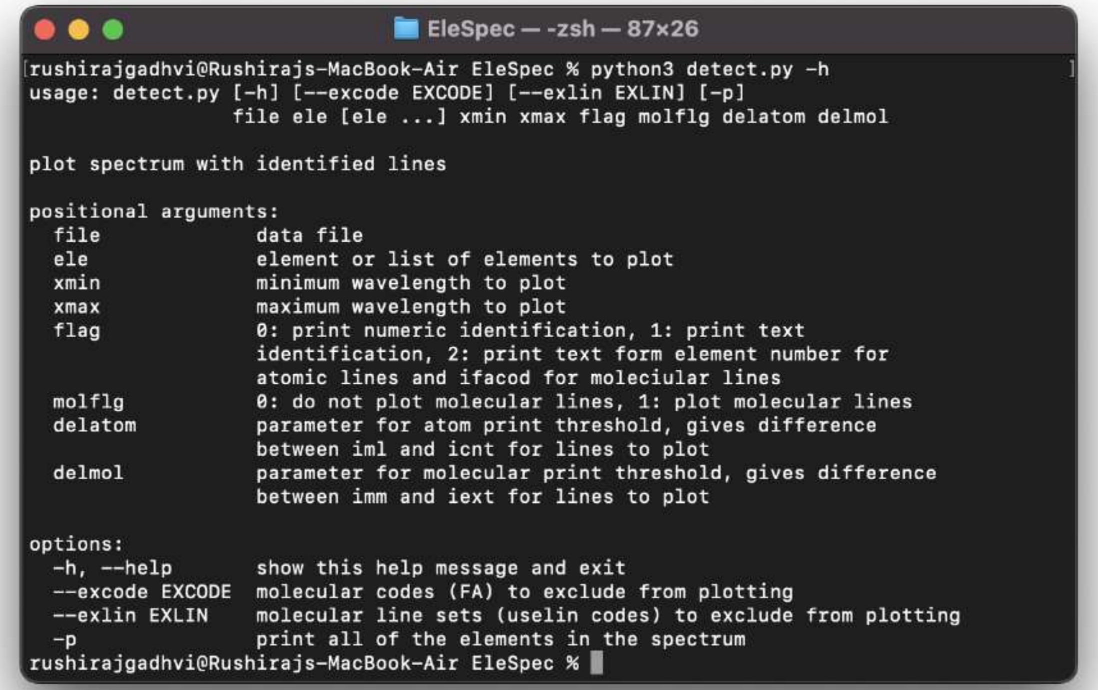
Normalize Data: [✓ ON] OFF

Process

Users have the option to normalize spectra, ensuring consistency and facilitating accurate comparisons between different spectra. Additionally, EleSpec allows users to specify subsections of the spectrum for analysis. The software supports the search for both atoms and molecules within the spectra, giving users the freedom to select multiple atoms or molecules from a drop-down menu based on their requirements. Alternatively, users can choose to search for the entire list of atoms or molecules within the spectra.

To fine-tune the obtained results, EleSpec provides parameters such as **"del-atom"** and **"del-mol."** Visual representations of the data are generated using the **matplotlib** library, offering users the flexibility to zoom in, pan, and save the graph images.

Additionally, For the users who want to use EleSpec without GUI, user can run the **detect.py** using the terminal and provide with the following arguments:



```
rushirajgadhvi@Rushirajs-MacBook-Air EleSpec % python3 detect.py -h
usage: detect.py [-h] [--excode EXCODE] [--exlin EXLIN] [-p]
                 file ele [ele ...] xmin xmax flag molflg delatom delmol

plot spectrum with identified lines

positional arguments:
  file             data file
  ele              element or list of elements to plot
  xmin             minimum wavelength to plot
  xmax             maximum wavelength to plot
  flag             0: print numeric identification, 1: print text
                   identification, 2: print text form element number for
                   atomic lines and ifacod for moleciular lines
  molflg           0: do not plot molecular lines, 1: plot molecular lines
  delatom          parameter for atom print threshold, gives difference
                   between iml and icnt for lines to plot
  delmol           parameter for molecular print threshold, gives difference
                   between imm and iext for lines to plot

options:
  -h, --help       show this help message and exit
  --excode EXCODE  molecular codes (FA) to exclude from plotting
  --exlin EXLIN    molecular line sets (uselin codes) to exclude from plotting
  -p               print all of the elements in the spectrum
rushirajgadhvi@Rushirajs-MacBook-Air EleSpec %
```

References

- EleSpec Software Github Repository
  [https://github.com/gadhvirushiraj/EleSpec.git]

- MCMC Github Repository
  [https://github.com/gadhvirushiraj/AstroMCMC.git]