Stellar spectrum interpolator

Abstract

The program described uses a set of precomputed spectra on a 4-dimensional grid of stellar parameter values: surface temperature, surface gravity, metallicity and carbonicity. Using the interpolation by IDW (inverse distance weighting), it computes a spectrum for the specified parameter values. This manual describes the method and explains how to run the program.

1 Introduction

The purpose of the program is to generate a spectrum, given specified values of stellar surface temperature T_{eff}, surface gravity $\log g$, metallicity [Fe/H] and carbonicity [C/Fe]. It is done by means of interpolating the computer-generated spectral library grid on the entire 4-dimensional parameter space (T_{eff}, $\log g$, [Fe/H], [C/Fe]).

2 Data

The spectral library used is called *CEMP_grid* and contains spectra for most combinations of the following parameter values – a total of $\sim$ 88000 spectra:

Parameter	Values
$T_{eff}(K)$	3750; 4750 – 8000 with 250 increment
$\log g$	0.0 – 5.0 with 0.5 increment
[Fe/H]	-4.50 – 1.00 with 0.25 increment
[C/Fe]	-1.50 – 4.50 with 0.25 increment

Table 1: Parameter values in the CEMP_grid library. T_eff is in Kelvin, and other parameters are unitless.

For each of those values, a spectrum consisting of the flux values at wavelengths $3000, 3001, ..., 10000$ Åis available.

The data is located on the internal server of the galactic archaeology group at University of Notre Dame, *nuit*, at emc3/grid_cemp/R2000_clean.hdf5. The normalized version of the spectra is at emc3/grid_cemp/R2000_clean_normed.hdf5.

3 Method

For interpolation, the IDW (Inverse Distance Weighting) method is used. First, we normalize the parameter values in our spectral library to be on the same scale:

$$x \to \frac{x - \overline{x}}{\sigma(x)}, \tag{1}$$

where x is the initial parameter value, $\overline{x}$ is the average parameter value, and $\sigma(x)$ is the standard deviation. After that, each parameter point in our spectral library can be written as a 4-dimensional vector

$$\mathbf{x}^i \;=\; \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \;=\; \begin{pmatrix} \hat{T_{eff}} \\ \hat{\log g} \\ [\text{Fe}\hat{/}\text{H}] \\ [\text{C}\hat{/}\text{Fe}] \end{pmatrix}, \tag{2}$$

where ˆ denotes a normalized value.

Now we have the set of grid points $\{\mathbf{x}^i\}_{i=1,\ldots,N}$, where N is the total number of points. At each of these points, a function $s(\mathbf{x^i})$ is defined, which returns a set of flux values of the spectrum. We want to interpolate this function on the entire space, so for any arbitrary $\mathbf{x}$ we could find $s(\mathbf{x})$. This is done by assigning weights to each of the available spectra and weight-averaging over them as follows:

$$s(\mathbf{x}) = \begin{cases} \dfrac{\sum_{i=1}^{N} w_i(\mathbf{x}) s(\mathbf{x}^i)}{\sum_{i=1}^{N} w_i(\mathbf{x})}, & \text{if } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \text{ for all } i, \\ s(\mathbf{x}^i), & \text{if } d(\mathbf{x}, \mathbf{x}_i) = 0 \text{ for some } i, \end{cases} \tag{3}$$

where summation over s_i means summation over each flux value at a given wavelength, and $\omega_i(\mathbf{x})$ is the weight of the i-th spectrum.

In IDW, the weights are calculated as the inverse of the 4-dimensional distance between the points $\mathbf{x}$ and $\mathbf{x}^i$ to some power:

$$\omega_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}^i)^p}, \qquad d(\mathbf{x}, \mathbf{x}^i) = \sqrt{\sum_{j=1}^{4} (x_j - x_j^i)^2} \tag{4}$$

The exponent p is called *Shepard's parameter* and is a variable parameter in our algorithm ($p = 2$ by default). The larger this parameter is, the more the weights are biased towards the closest points.

4 HowTo

4.1 Setup

1. Download the program files from Spectra-IDW. Unpack everything in the same directory.

2. Download the normalized grid from *nuit*:

```
scp [username]@nuit.phys.nd.edu:/emc3/grid_cemp/R2000_clean_normed.hdf5
    R200_clean_normed.hdf5
```

Put this file into the program directory.

3. Open file *params.py* with a text editor, for example:

```
gedit params.py
```

Set the Shepard parameter to any value you prefer (the values 2 and 3 have been tested to work fine). The program is ready to run now.

4.2 Generate a single spectrum

Run the interpolation routine:

```
python Run_single.py
```

The program will prompt you to enter the 4 stellar parameter values to generate the interpolated spectrum for. Note: the process can take a few minutes, or even run out of memory, on weaker systems, as it has to read in several GB of data into memory.

When the process is finished, the *.hdf5 file will appear in the output directory, its name listing the parameter values.

4.3 Generating multiple spectra

1. Open file *param_vals.csv* with a text editor or a table processor like OpenOffice Calc or Microsoft Excel.

```
gedit param_vals.csv
```

Make a list of the parameter values to generate spectra for.

2. Run the interpolation routine:

```
python Run.py
```

Note: the process can take a long time, or even run out of memory, on weaker systems, as it has to read in several GB of data into memory. It is recommended to run this process in the background.

The resulting *.hdf5 spectral files will appear in the output directory, their names listing the parameter values.

4.4 Plotting a spectrum

In the output directory, you can run the plotting routine to look at the generated spectrum:

```
python plot_spectrum.py *filename*
```

The code in *plot_spectrum.py* also shows how to obtain the wavelength and flux values from the file in Python.