hipparchus Documentation

Release 0.0.dev26

Brett M. Morris

Jan 09, 2020

CONTENTS

This is the documentation for hipparchus, a lightweight Python package for cross correlation of high resolution spectroscopy.

Part I

hipparchus Documentation

This is the documentation for hipparchus.

CHAPTER

ONE

REFERENCE/API

1.1 hipparchus Package

1.1.1 Functions

cross_corr(spectrum, template[, start_lam, ...])	Cross-correlation of the spectrum and template.
test(**kwargs)	Run the tests for the package.

cross_corr

hipparchus.**cross_corr**(*spectrum*, *template*, *start_lam=-2*, *end_lam=2*, *n_steps=1000*, *sigma=None*, *spread_factor=10*)

Cross-correlation of the spectrum and template.

Parameters

spectrum
[Spectrum] Spectrum object (may be an order of an echelle spectrum).

template
[Template] Template object to correlate against the spectrum.

start_lam
[float] Start wavelength relative to mean wavelength

end_lam
[float] End wavelength relative to mean wavelength

n_steps
[int] Number of steps to compute the CCF over between `start_lam` and `end_lam`

sigma
[float] Gaussian smoothing filter width (standard deviation)

spread_factor
[float] Gaussian smoothing filter width spread scaling factor

Returns

ccf
[CCF] Cross-correlation object.

test

hipparchus.**test**(***kwargs*)

Run the tests for the package.

This method builds arguments for and then calls `pytest.main`.

Parameters

package
[str, optional] The name of a specific package to test, e.g. 'io.fits' or 'utils'. Accepts comma separated string to specify multiple packages. If nothing is specified all default tests are run.

args
[str, optional] Additional arguments to be passed to `pytest.main` in the `args` keyword argument.

docs_path
[str, optional] The path to the documentation .rst files.

open_files
[bool, optional] Fail when any tests leave files open. Off by default, because this adds extra run time to the test suite. Requires the `psutil` package.

parallel
[int or 'auto', optional] When provided, run the tests in parallel on the specified number of CPUs. If parallel is `'auto'`, it will use the all the cores on the machine. Requires the `pytest-xdist` plugin.

pastebin
[('failed', 'all', None), optional] Convenience option for turning on py.test pastebin output. Set to 'failed' to upload info for failed tests, or 'all' to upload info for all tests.

pdb
[bool, optional] Turn on PDB post-mortem analysis for failing tests. Same as specifying `--pdb` in `args`.

pep8
[bool, optional] Turn on PEP8 checking via the pytest-pep8 plugin and disable normal tests. Same as specifying `--pep8 -k pep8` in `args`.

plugins
[list, optional] Plugins to be passed to `pytest.main` in the `plugins` keyword argument.

remote_data
[{'none', 'astropy', 'any'}, optional] Controls whether to run tests marked with @pytest.mark.remote_data. This can be set to run no tests with remote data (`none`), only ones that use data from http://data.astropy.org (`astropy`), or all tests that use remote data (`any`). The default is `none`.

repeat
[`int`, optional] If set, specifies how many times each test should be run. This is useful for diagnosing sporadic failures.

skip_docs
[`bool`, optional] When `True`, skips running the doctests in the .rst files.

test_path
[str, optional] Specify location to test by path. May be a single file or directory. Must be specified absolutely or relative to the calling directory.

verbose
[bool, optional] Convenience option to turn on verbose output from py.test. Passing True is the same as specifying `-v` in `args`.

1.1.2 Classes

CCF(velocities, ccf[, header])	Storage object for cross-correlation functions
EchelleSpectrum(orders[, header])	Echelle spectrum object, which stores each order as a Spectrum object.
LooseVersion([vstring])	Version numbering for anarchists and software realists.
Spectrum(wavelength, flux[, header])	Simple spectrum object.
Template(wavelength, emission)	Spectral template object.
UnsupportedPythonError	

CCF

class hipparchus.**CCF**(*velocities*, *ccf*, *header=None*)
Bases: object

Storage object for cross-correlation functions

Attributes Summary

rv	Approximate radial velocity of the star
signal_to_noise	Approximate signal-to-noise of the strongest absorption feature

Methods Summary

plot(self[, ax])	Plot the CCF.

Attributes Documentation

rv
Approximate radial velocity of the star

signal_to_noise
Approximate signal-to-noise of the strongest absorption feature

Methods Documentation

plot(*self*, *ax=None*, ***kwargs*)
Plot the CCF.

Parameters

ax
[Axes] Matplotlib axis object

kwargs
[dict] Keyword arguments to pass to the matplotlib `plot` function

Returns

ax
[Axes] Matplotlib axis object

EchelleSpectrum

class hipparchus.**EchelleSpectrum**(*orders*, *header=None*)
Bases: object

Echelle spectrum object, which stores each order as a `Spectrum` object.

Methods Summary

continuum_normalize(self[, bins, order])	Normalize the continuum in each echelle order to unity.
from_e2ds(path[, harps])	Read HARPS(-N) spectrum from an E2DS FITS file.
nearest_order(self, wavelength)	Return the order with the central wavelength nearest to `wavelength`.
plot(self[, ax])	Plot the echelle spectrum

Methods Documentation

continuum_normalize(*self*, *bins=100*, *order=10*)
Normalize the continuum in each echelle order to unity.

Parameters

bins
[int] Number of bins used to compute maxes for continuum tracing

order
[int] Polynomial order fit to the binned-maxes

classmethod from_e2ds(*path*, *harps=True*)
Read HARPS(-N) spectrum from an E2DS FITS file.

Parameters

path
[str] Path to FITS file

harps
[bool (optional)] True for HARPS, False for HARPS-N

Returns

sp
[EchelleSpectrum] Echelle spectrum object

nearest_order(*self*, *wavelength*)
Return the order with the central wavelength nearest to wavelength.

Parameters

wavelength
[float] Reference wavelength

Returns

spectrum
[Spectrum]

plot(*self*, *ax=None*, ***kwargs*)
Plot the echelle spectrum

Parameters

ax
[Axes (optional)] Axis object

kwargs
[dict] Keyword arguments to pass to the plot command

Returns

ax
[axis]

Spectrum

class hipparchus.**Spectrum**(*wavelength*, *flux*, *header=None*)
Bases: object

Simple spectrum object.

Parameters

wavelength
[ndarray] Wavelengths in Angstroms

flux
[ndarray] Fluxes

Methods Summary

plot(self[, ax])	Plot the spectrum

Methods Documentation

plot(*self*, *ax=None*, ***kwargs*)
Plot the spectrum

Parameters

ax
[Axes (optional)] Axis object

kwargs
[dict] Keyword arguments to pass to the plot command

Returns

ax
[axis]

Template

class hipparchus.**Template**(*wavelength*, *emission*)
Bases: object

Spectral template object.

Parameters

wavelength
[ndarray] Wavelengths in Angstroms

emission
[ndarray] Emission from the spectral template.

Methods Summary

from_npy(path[, norm])	Load spectral template from npy pickle.
plot(self[, ax])	Plot the transmittance

Methods Documentation

classmethod from_npy(*path*, *norm=True*)
Load spectral template from npy pickle.

Parameters

path
[str] Path to emission file

norm
[bool (optional)] If `True`, normalize the template such that the sum of the template over all wavelengths is equal to unity; else skip normalization.

plot(*self*, *ax=None*, ***kwargs*)
Plot the transmittance

Parameters

ax
[`Axes` (optional)] Axis object

kwargs
[dict] Keyword arguments to pass to the `plot` command

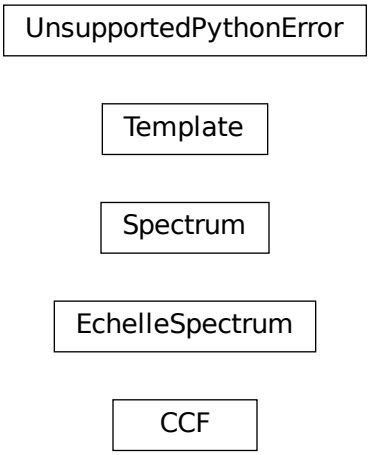
Returns

ax
[axis]

UnsupportedPythonError

exception hipparchus.**UnsupportedPythonError**

1.1.3 Class Inheritance Diagram



Part II

Installation

To install hipparchus from the source code, simply type:

```
git clone https://github.com/bmorris3/hipparchus.git
cd hipparchus
python setup.py install
```

Part III

Tutorials

In this tutorial, we'll walk through an example where we download a HARPS E2DS echelle spectrum of LkCa 4, and Proxima Centauri. We'll also download two spectral templates – one for TiO and another for water. We'll then take the cross-correlation of the TiO template with the stellar spectra to show that there is significant TiO absorption in the atmosphere of Proxima, an M5V star, and more interestingly, that there is also TiO absorption in the atmosphere of LkCa 4, a "K7" star. Lastly, we'll hunt for cool starspots on the stellar surface by searching for cooler water absorption in the optical high resolution spectrum via the cross-correlation.

CHAPTER

TWO

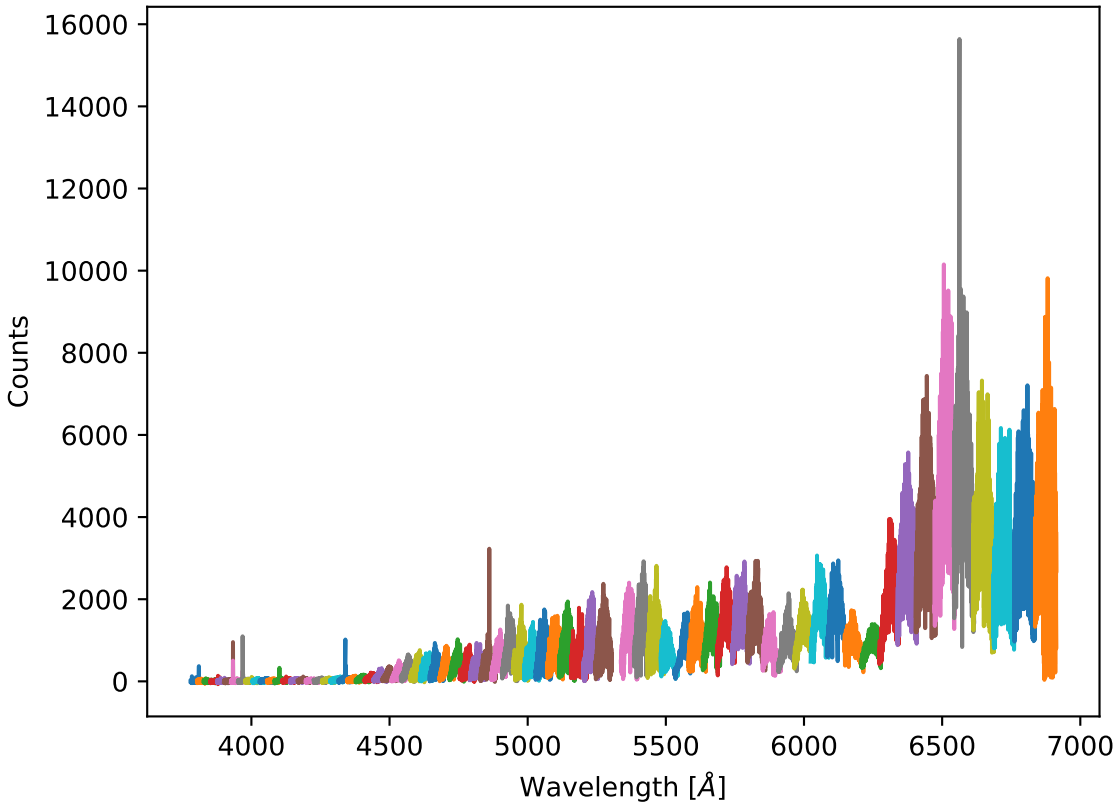
LOADING A SPECTRUM

First, we need a spectrum to work with, so we'll download a HARPS E2DS spectrum of LkCa 4 and Proxima Centauri that's currently hosted on Google Drive using astropy's `download_file` function, like so:

```
from astropy.utils.data import download_file

lkca4_spectrum_url = 'https://drive.google.com/uc?export=download&id=1x3nIg1P5tYFQqJrwEpQU11XQOs3ImH3v'
lkca4_spectrum_path = download_file(lkca4_spectrum_url)

proxima_spectrum_url = 'https://drive.google.com/uc?export=download&id=1I7E1x1XRjcxXNQXiuaajb_Jz7Wn2N_Eo
↪'
proxima_spectrum_path = download_file(proxima_spectrum_url)
```

(you could alternatively use any E2DS spectrum downloaded from the ESO Archive). We now have a local copy of the spectrum of Proxima Cen. We can load this file into an `EchelleSpectrum` object like so:

```
from hipparchus import EchelleSpectrum

lkca4_spectrum = EchelleSpectrum.from_e2ds(lkca4_spectrum_path)
proxima_spectrum = EchelleSpectrum.from_e2ds(proxima_spectrum_path)

proxima_spectrum.plot()
```


16000
14000
12000
10000
8000
6000
4000
2000
0
Counts
4000
4500
5000
5500
6000
6500
7000
Wavelength [Å]

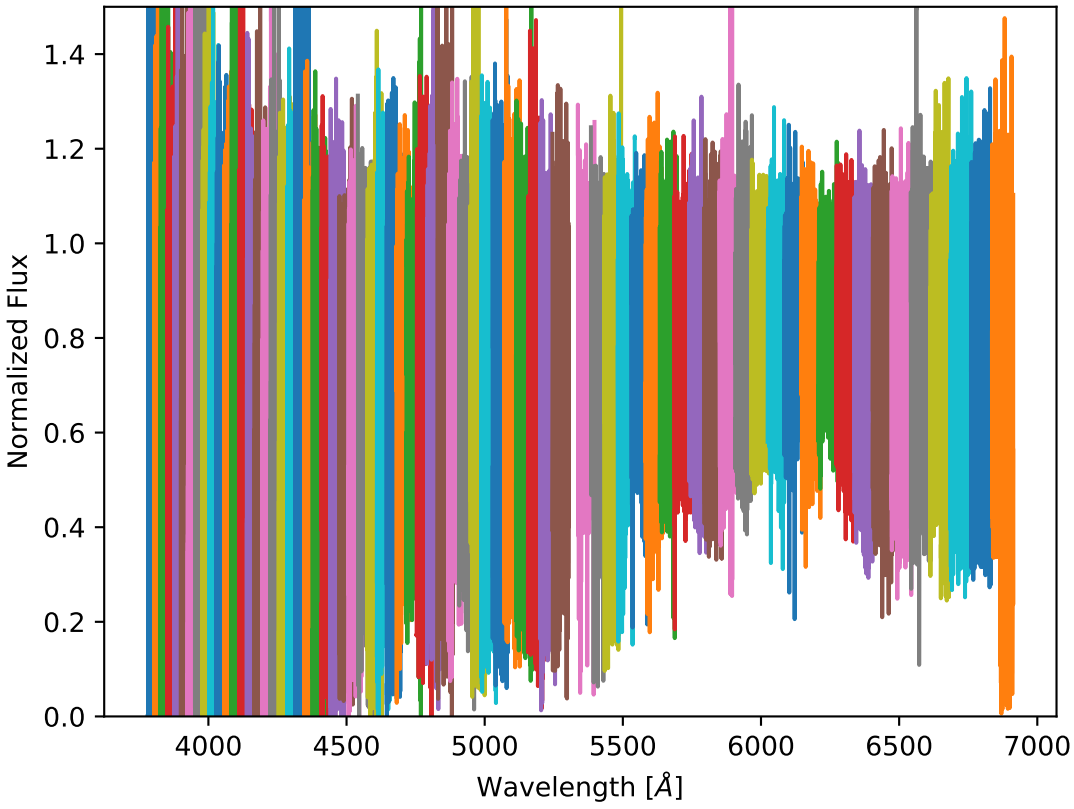
CHAPTER

THREE

CONTINUUM NORMALIZE

Next we might want to continuum normalize each echelle order. We can do that simply with the `continuum_normalize` command:

```
lkca4_spectrum.continuum_normalize()

proxima_spectrum.continuum_normalize()

proxima_spectrum.plot()
```



You can see that the continuum flux is now mostly normalized near unity.

Our spectrum is now ready for cross-correlation business!

CHAPTER

FOUR

LOADING A SPECTRAL TEMPLATE

Next we need to load spectral templates which we will cross-correlate with the observed spectrum of Proxima Centauri. We'll be loading emission spectra generated by Daniel Kitzmann which represent: TiO at 3000 K (roughly the effective temperature of Proxima Cen), and H2O at 2500 K (roughly the temperature of hypothesized starspots on Proxima Cen).

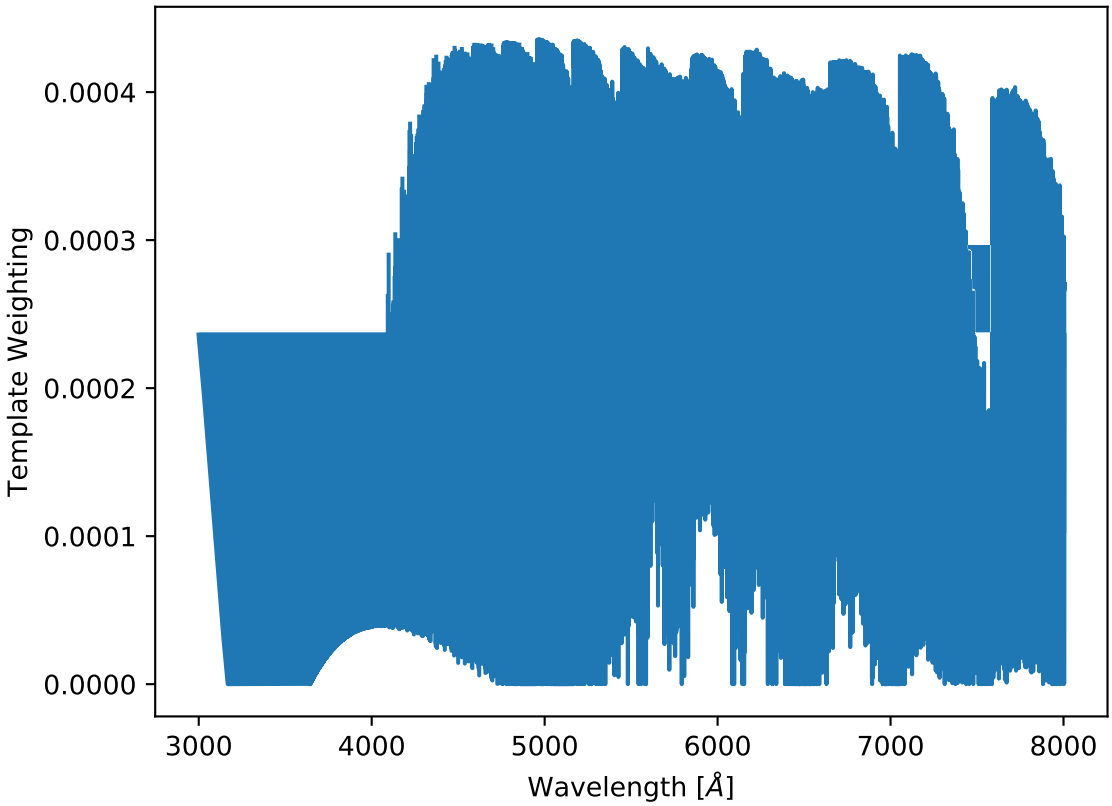
First we download the spectral templates:

```
from astropy.utils.data import download_file

template_2500_h2o_url = 'https://drive.google.com/uc?export=download&id=1RIXBl3L3J_R9PQ-k_0BqAtO-
↪9zYn2mag'
template_3000_tio_url = 'https://drive.google.com/uc?export=download&
↪id=1eGUBfk7Q9zaXgJQJtVFB6pit7cmoGCpn'

template_2500_h2o_path = download_file(template_2500_h2o_url)
template_3000_tio_path = download_file(template_3000_tio_url)
```

We now have a local copy of the spectral templates for TiO and water. Let's load those templates into the `Template` object:

```
from hipparchus import Template

template_2500_h2o = Template.from_npy(template_2500_h2o_path)
template_3000_tio = Template.from_npy(template_3000_tio_path)

template_3000_tio.plot()
```

By default the template spectrum is normalized such that it is near zero in the continuum, and greater than zero in absorption lines. This allows one to easily compute the cross-correlation via the weighted mean.


0.0004
0.0003
0.0002
0.0001
0.0000
Template Weighting
3000
4000
5000
6000
7000
8000
Wavelength [Å]

CHAPTER

FIVE

CROSS CORRELATION

So we have spectra, and a template. Now let's take the cross-correlation of the two! We define the cross-correlation function (CCF) for an observed spectrum x, given a template spectrum evaluated at a specific velocity $T(v)$,

$$\mathrm{CCF} = \sum_i x_i T_i(v),$$

where we have normalized the template such that it is positive in molecular absorption features and near-zero in the continuum, and

$$\sum_i T_i(v) = 1.$$

This definition of the CCF is straightforward to interpret: the CCF is a mean of the flux in each echelle order weighted by the values of the spectral template. When the velocity v is incorrect and/or the template does not match the observed spectrum, the weighted-mean flux is near unity (continuum). When the velocity is correct and the template matches absorption features in the observed spectrum, the absorption features in the spectrum "align" with the inverse absorption features in the template, and the weighted-mean flux is less than one. In this way, the CCF yields a "mean absorption line" due to the molecule specified by the template at the velocity of the star.

Let's compute the CCF between the template and the observed Proxima Cen spectrum in the echelle order nearest to the wavelength 6800 Angstroms using the `cross_corr` function:

```
from hipparchus import cross_cor

ccf = cross_corr(proxima_spectrum.nearest_order(6800), template_3000_tio)
ccf.plot()
```

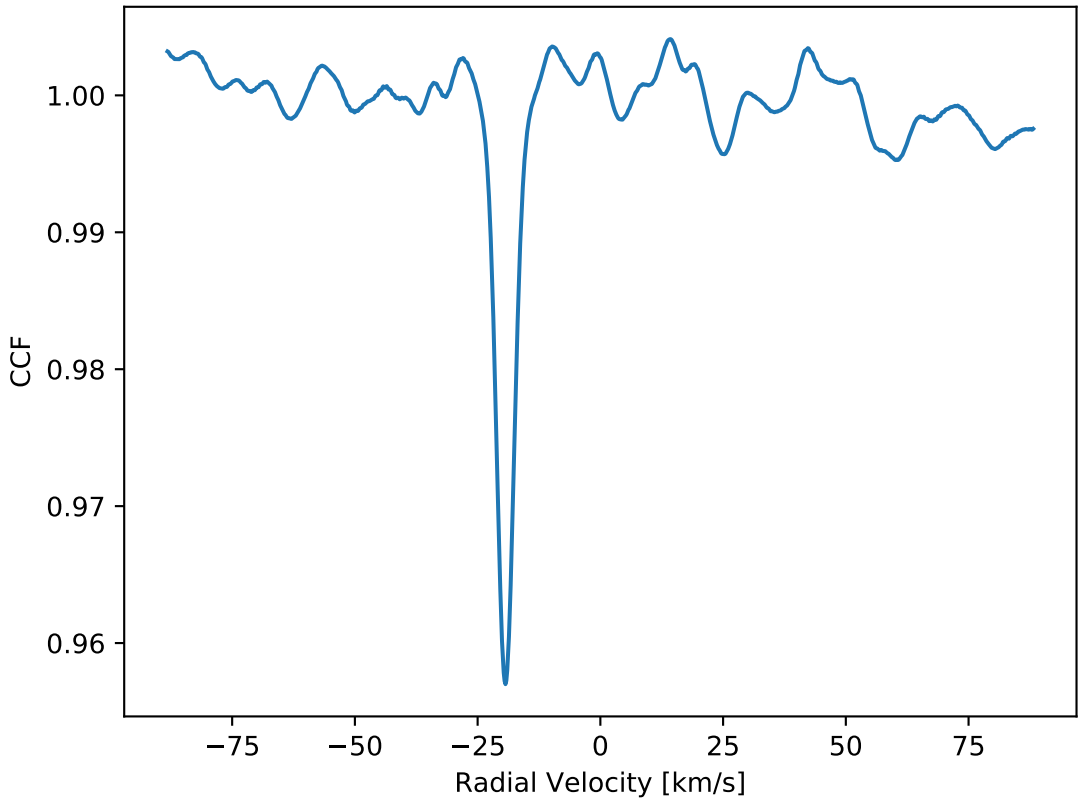
We can see a significant "mean absorption line" in the cross-correlation function of the TiO emission spectrum near the known radial velocity of Proxima Cen at -22 km/s. This is an unsurprisingly significant detection of TiO in the atmosphere of the cool star Proxima Centauri.

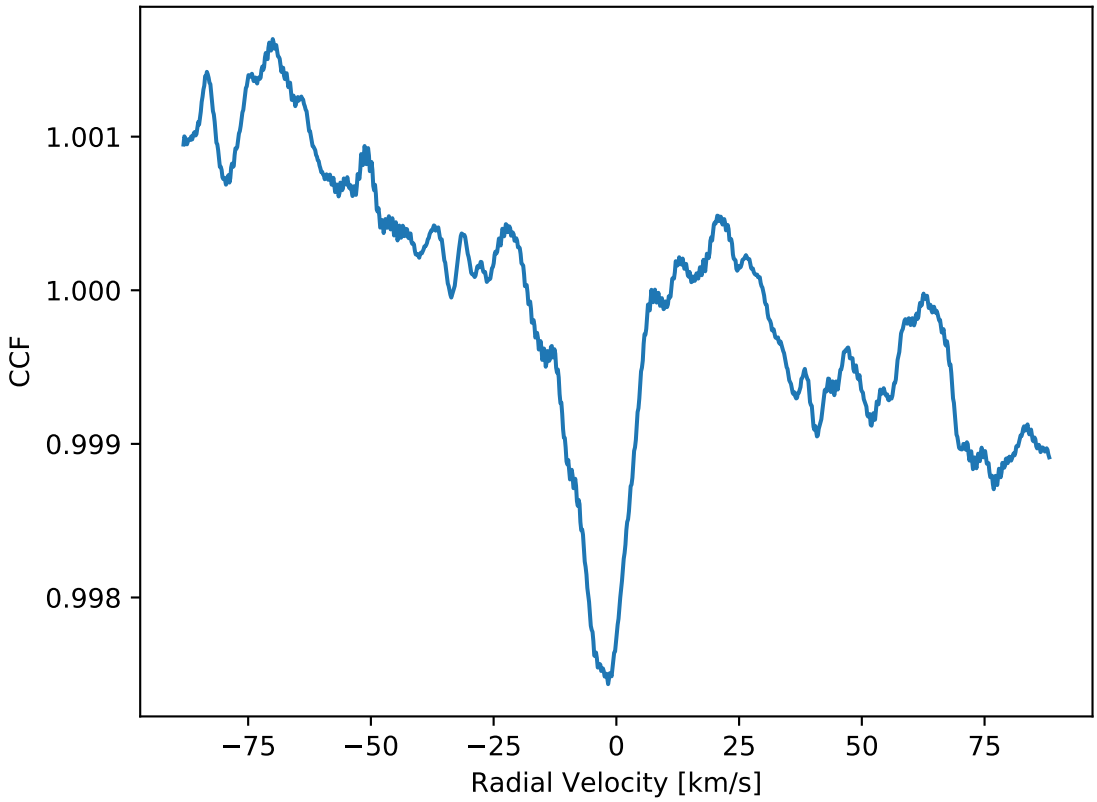
The `cross_corr` function returns a `CCF` object, which stores the resulting cross-correlation function as a function of velocity and its metadata.

Let's now take the CCF of the LkCa 4 spectrum and the TiO template:

```
from hipparchus import cross_cor

ccf = cross_corr(lkca4_spectrum.nearest_order(6800), template_3000_tio)
ccf.plot()
```

Despite LkCa 4 being classified as a K dwarf, this CCF shows a significant absorption signal due to TiO. This absorption can be understood in the context of the really neat paper by Gully-Santiago et al 2017, who showed that this star is roughly 80% covered in cool, dark regions where it is cool enough for TiO to form, and 20% covered by hot, bright regions.


1.00
0.99
0.98
0.97
0.96
CCF
−75
−50
−25
0
25
50
75
Radial Velocity [km/s]


1.001
1.000
0.999
0.998
CCF
−75
−50
−25
0
25
50
75
Radial Velocity [km/s]

CHAPTER

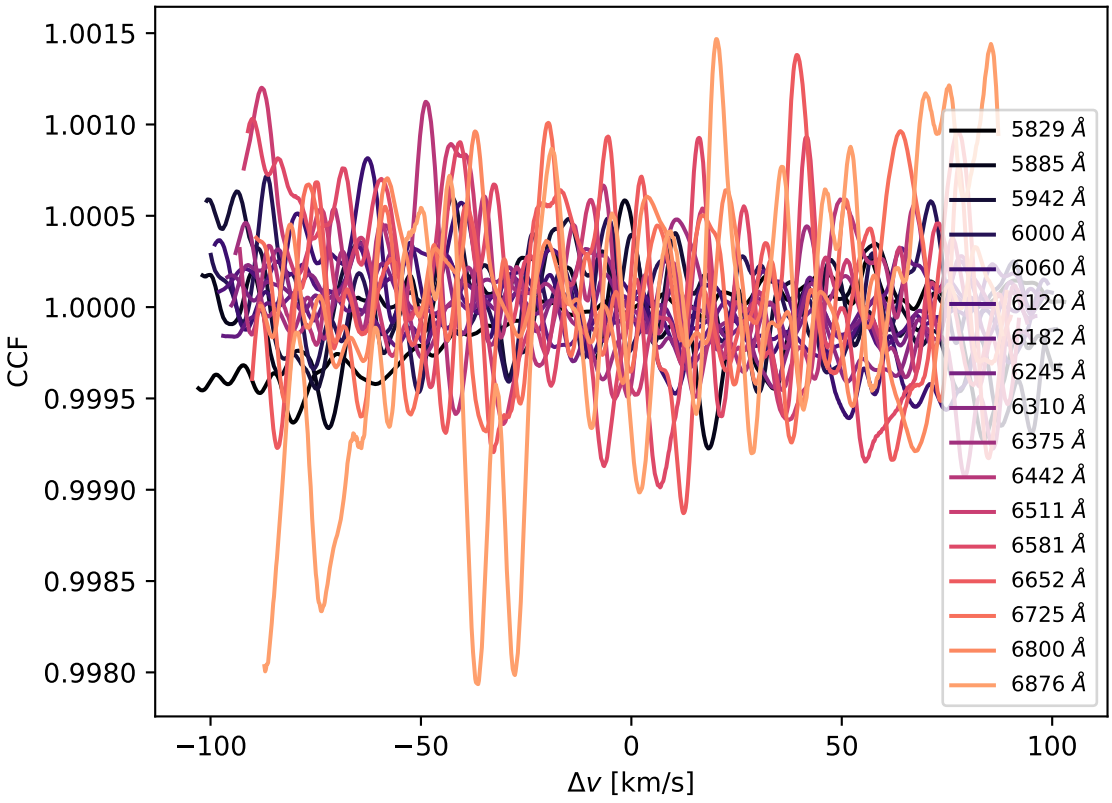
SIX

HUNTING FOR STARSPOTS

Let's now hunt for starspots on Proxima Centauri. Starspots are cool regions on stellar surfaces where strong magnetic fields inhibit convection. Since starspots are cooler than the rest of the stellar photosphere, they may have molecular absorption features that don't occur elsewhere on the star. Proxima Centauri, for example, is a 3000 K star. At 3000 K, we don't expect to see significant absorption due to water molecules, but water begins to show absorption features in the optical at and below 2500 K at wavelengths greater than 5800 Angstroms. So let's cross-correlate each order of the Proxima Centauri echelle spectrum with the high resolution template for water at 2500 K to see if there is significant starspot coverage on Proxima Centauri with temperatures $\Delta T \sim 500$ K:

```
counter = -1
for order in proxima_spectrum.orders:
    if order.wavelength.mean() > 5800:
        counter += 1
        ccf = cross_corr(order, template_2500_h2o)
        ccf.plot(label='{0:.0f} $\AA$'.format(order.wavelength.mean()),
                 color=plt.cm.magma(counter/20))
plt.legend(loc='lower right', fontsize=8)
plt.xlabel('$\Delta v$ [km/s]')
plt.ylabel('CCF')
```

The CCF of each spectral order is shown with a different color curve. You can see that there are no spectral orders which show significant absorption at the radial velocity of Proxima Centauri (-22 km/s). This suggests there is not significant absorption due to water vapor in the atmosphere of Proxima Cen. This could suggest there is insignificant coverage by cool spots, the spots are hotter than 2500 K, or some combination of both scenarios.


1.0015
1.0010
1.0005
1.0000
0.9995
0.9990
0.9985
0.9980
CCF
−100
−50
0
50
100
Δv [km/s]
5829 Å
5885 Å
5942 Å
6000 Å
6060 Å
6120 Å
6182 Å
6245 Å
6310 Å
6375 Å
6442 Å
6511 Å
6581 Å
6652 Å
6725 Å
6800 Å
6876 Å

PYTHON MODULE INDEX

h

INDEX