qusp Documentation

Release 0.0

Daniel Margala

February 04, 2017

Contents

A package for working with **qu**asar **sp**ectra.

Here is an example recipe for calculating the median signal to noise in the lya forest for quasar targets:

```
# rest frame lya forest limits
rest_wave_min = qusp.wavelength.Wavelength(1040)
rest_wave_max = qusp.wavelength.Wavelength(1190)
for target, spectrum in qusp.target.get_combined_spectra(targets):
    # determine the observed frame forest window
    obs_wave_min = rest_wave_min.observed(target['z'])
    obs_wave_max = rest_wave_max.observed(target['z'])
    # get the median sn in the forest window
    forest_sn = spectrum.median_signal_to_noise(obs_wave_min, obs_wave_max)
```

The target module provides support for reading target lists:

```
# parse target list with ra, dec, and z attributes, example line:
#   1234-56789-109 87.6 54.3 2.1
targets = qusp.target.load_target_list('quasars.txt',
    fields=[('ra', float, 1), ('dec', float, 2), ('z', float, 3)])
```

The `filter.py` utility makes it east to create target lists from a catalog file:

```
$ examples/filter.py -i spAll-v5_7_0.fits \
    --select "(['OBJTYPE'] == 'QSO') & (['Z'] > 2.1)" \
    --annotate 'ra:dec:z' --save quasars.txt --verbose
```

CHAPTER 1

User Documentation

Package API and how to use the example programs included in this package.

1.1 Quick Start

Download package:

```
git clone https://github.com/dmargala/qusp.git
```

Make sure python knows where about the package:

```
export PYTHONPATH=/path/to/qusp:$PYTHONPATH
```

Set environment variables to find BOSS data:

```
export BOSS_ROOT=/data/boss
export BOSS_VERSION=v5_7_0
```

1.2 Package API

1.2.1 qusp.target module

Provides support for working with BOSS targets.

In qusp, a target is identified by a unique plate-mjd-fiber. They are implemented as dictionaries and must have at least 'plate', 'mjd', and 'fiber' keys specified. The Target model is designed to be flexible, in that other attributes can be added to targets as needed.

Examples

Construct a target from a string identifier:

```
target = qusp.target.Target.from_string('plate-mjd-fiber')
```

Construct a target from a dictionary:

```
target = qusp.target.Target({'target':'plate-mjd-fiber'})
```

Read a target list along with **ra**, **dec**, and **z** columns:

```
targets = qusp.target.load_target_list(filename,
    fields=[('ra', float, 1), ('dec', float, 2), ('z', float, 3)])
```

Save a target list along with **z** and **sn** fields:

```
qusp.target.save_target_list(filename, targets, fields=['z', 'sn'])
```

Iterate over combined spectra for a list targets:

```
for target, spectrum in qusp.target.get_combined_spectra(targets):
    ...
```

Iterate over plates for a list of targets:

```
for target, spplate in qusp.target.get_target_plates(targets):
    spectrum = qusp.read_combined_spectrum(spplate, target)
    ...
```

---

class qusp.target.**Target**(**args*, ***kwargs*)

Bases: dict

Represents a BOSS target.

Parameters

- **args** – Variable length argument list.
- **kwargs** – Arbitrary keyword arguments.

Raises AssertionError – if 'target' key is not specified

classmethod from_plate_mjd_fiber(*plate*, *mjd*, *fiber*)

Returns a Target object constructed from plate, mjd, fiber.

Parameters

- **plate** (*int*) – target's plate id
- **mjd** (*int*) – mjd observation
- **fiber** (*int*) – target's fiber id

Returns *Target* object

classmethod from_string(*target_string*)

Returns a Target object constructed from a target string identifier.

Parameters **target_string** (*str*) – a target string identifier.

Returns *Target* object

to_string()

Returns the standard plate-mjd-fiber string represntation of the target.

Returns plate-mjd-fiber string represntation of target

qusp.target.**add_args**(*parser*)

Adds arguments to the provided command-line parser.

Parameters **parser** (*argparse.ArgumentParser*) – an argument parser

qusp.target.**get_combined_spectra**(*targets*, *paths=None*, *sort=True*, *verbose=False*, *tpcorr=None*)

A generator that yields (target, spectrum) tuples for the provided list of targets. With sort=True, the targets will be sorted by plate-mjd-fiber to reduce the number of io operations.

Parameters

- **targets** (*Target*) – list of *Target* objects to iterate through.
- **boss_path** (*str, optional*) – path to boss data directory. Default is to look this up using env var.
- **sort** (*bool, optional*) – Whether or not to sort the provided targets by plate-mjd-fiber. Defaults to True.
- **verbose** (*bool, optional*) – Whether or not to print verbose output. Defaults to False.

Yields The next tuple `(target, spectrum)`, where `target` is a *Target* and `spectrum` is a *qusp.spectrum.Spectrum* that corresponds to the target's coadded spectrum.

qusp.target.**get_combined_spectrum**(*target*, *paths=None*)
Returns the coadded spectrum of the specified target.

Parameters

- **target** (*Target*) – a target
- **paths** (*qusp.paths.Paths*, optional) – paths object that knows where the location of the boss data dir.

Returns Coadded spectrum of the specified target.

qusp.target.**get_corrected_spectrum**(*target*, *tpcorr*, *paths=None*)
Returns the coadded spectrum of the specified target.

Parameters

- **target** (*Target*) – a target
- **tpcorr** (*hdf5 File object*) – hdf5 file with throughput corrections
- **paths** (*qusp.paths.Paths*, optional) – paths object that knows where the location of the boss data dir.

Returns Coadded spectrum of the specified target.

qusp.target.**get_lite_spectra**(*targets*)

qusp.target.**get_lite_spectrum**(*target*, *paths=None*)

qusp.target.**get_target_plates**(*targets*, *boss_path=None*, *sort=True*, *verbose=False*)
A generator that yields (target,spplate) tuples for the provided list of targets. With sort=True, the targets will be sorted by plate-mjd-fiber to reduce the number of io operations.

Parameters

- **targets** (*Target*) – list of *Target* objects to iterate through.
- **boss_path** (*str, optional*) – path to boss data directory. Default is to look this up using env var.
- **sort** (*bool, optional*) – Whether or not to sort the provided targets by plate-mjd-fiber. Defaults to True.
- **verbose** (*bool, optional*) – Whether or not to print verbose output. Defaults to False.

Yields The next tuple `(target, spplate)`, where `target` is a *Target* and `spplate` is the corresponding FITS file containing its coadded spectrum from the list of *targets*.

qusp.target.**load_target_list**(*filename*, *fields=None*, *verbose=False*)
Loads a target data from a text file.

The first column must be plate-mjd-fiber target identifier. Use the fields argument to specify additional columns to read. Must specify a (name, type, column index) tuple for each field.

Parameters

- **filename** (*str*) – The filename to load.
- **fields** (*list, optional*) – A list of columns to read, see example. Defaults to None.
- **verbose** (*bool, optional*) – Whether or not to print verbose output. Defaults to False.

Returns list of *Target* objects.

qusp.target.**load_target_list_from_args**(*args*, *fields=None*)
Loads a target list from a text file specified using the default target arguments.

Parameters

- **args** (*argparse.Namespace*) – argparse argument namespace
- **fields** (*list, optional*) – A list of columns to read, see example. Defaults to None.

Returns list of *Target* objects.

qusp.target.**save_target_list**(*filename*, *targets*, *fields=None*, *verbose=False*)
Writes a list of targets to the provided file.

By default, only the target plate-mjd-fiber is written to file. Use the fields argument to specify additional target fields to save.

Parameters

- **filename** (*str*) – The filename of the output text file to create.
- **targets** (*Target*) – A list of *Target* objects to save.
- **fields** (*list, optional*) – A list of *Target* keys to annotate output list. Defaults to None.
- **verbose** (*bool, optional*) – Whether or not to print verbose output. Defaults to False.

1.2.2 qusp.spectrum module

Provides classes to represent functions of wavelength.

Examples

Construct a *Spectrum* object from `wave`, `flux`, and `ivar` arrays:

```
>>> spectrum = qusp.spectrum.Spectrum(wave, flux, ivar)
```

Get the mean flux between `wave_min` and `wave_max`:

```
>>> spectrum.mean_flux(wave_min, wave_max)
```

Get the median signal to noise between `wave_min` and `wave_max`:

```
>>> spectrum.median_signal_to_noise(wave_min, wave_max)
```

Load the combined spectrum of a `qsub.target.Target` object, `target`:

```
filename = 'spPlate-%s-%s.fits' % (target['plate'], target['mjd'])
spplate = fits.open(os.path.join(paths.boss_path, str(target['plate']), filename))
combined = qusp.read_combined_spectrum(spplate, target)
```

---

class `qusp.spectrum.`**BOSSSpectrum**(*wavelength*, *flux*, *ivar*, *wavelengths_units=<Mock id='140258957330640'>*, *flux_units=None*, *extrapolated_value=None*)

Bases: `object`

Represents a BOSS co-added spectrum.

Parameters

- **wavelength** (*numpy.ndarray*) – wavelength pixel centers.
- **flux** (*numpy.ndarray*) – flux values.
- **ivar** (*numpy.ndarray*) – flux inverse variance values.

create_corrected(*correction*)

find_pixel(*wavelength*, *clip=False*)

Returns the corresponding pixel index of the specified wavelength.

Parameters

- **wavelength** (*float*) – value
- **clip** (*bool*) – if wavelength is out of range, return 0, or npixels-1

Returns pixel index

Return type pixelIndex (int)

mean_flux(*min_wavelength*, *max_wavelength*, *ivar_weighting=True*)

Returns the mean flux between the specified wavelengths. Use `ivar_weighting=False` to turn ignore weights.

Parameters

- **min_wavelength** (*float*) – minimum wavelength for mean flux calculation range.
- **max_wavelength** (*float*) – maximum wavelength for mean flux calculation range.
- **ivar_weighting** (*bool, optional*) – Whether or not to weight calculation using inverse variance.

Returns the mean flux between `min_wavelength` and `max_wavelength`.

median_signal_to_noise(*min_wavelength*, *max_wavelength*)

Returns the median signal to noise ratio between the specified wavelengths.

Parameters

- **min_wavelength** (*float*) – minimum wavelength for median flux calculation range.
- **max_wavelength** (*float*) – maximum wavelength for median flux calculation range.

Returns the median flux between `min_wavelength` and `max_wavelength`.

Return type median (float)

trim_range(*wave_min*, *wave_max*, *clip=True*)
Returns a *BOSSSpectrum* object trimmed to the specified range.

Parameters

- **wave_min** (*float*) – wavelength range lower bound
- **wave_max** (*float*) – wavelength range upper bound

Returns A *BOSSSpectrum*

Raises ValueError – if no pixels in specified range

class qusp.spectrum.**SpectralFluxDensity**(*wavelength*, *flux*, *wavelengths_units=<Mock id='140258957136528'>*, *flux_units=None*, *extrapolated_value=None*)
Bases: *qusp.spectrum.WavelengthFunction*

Represents a spectral flux density as a function of wavelength.

Initializes a spectral flux density using tabulated flux values at specified wavelengths. See the documentation of WavelengthFunction for details. The default flux unit is 1e-17 erg/(s*cm^2*Ang), which is available as SpectralFluxDensity.fiducialFluxUnit, but other units can be specified.

Parameters

- **wavelength** (*numpy.ndarray*) – tabulated wavelength values
- **flux** (*numpy.ndarray*) – tabulated flux values
- **wavelengths_units** (*astropy.units.Quantity*) – wavelength value units
- **flux_units** (*astropy.units.Quantity*) – flux value units
- **extrapolated_value** (*float, optional*) –

create_redshifted(*new_z*, *old_z=0.0*, *preserve_wavelengths=False*)
Returns a new SpectralFluxDensity whose wavelengths and fluxes have been rescaled for the transformation from old_z to new_z. If preserve_wavelengths is True and an extrapolated_value has been set, then the rescaled spectrum will be resampled to the original wavelengths. Otherwise, the new spectrum will be tabulated on the redshifted grid.

Parameters

- **new_z** (*float*) – redshift to rescale to
- **old_z** (*float, optional*) – original redshift to rescale from
- **preserve_wavelengths** (*bool*) – preserve wavelength grid, otherwise create redshifted grid

Returns rescaled spectrum (*qusp.spectrum.SpectralFluxDensity*)

Raises RuntimeError – if preserve_wavelengths is True and no extrapolated_value has been set

create_rescaled(*sdss_band*, *ab_magnitude*)

Parameters

- **sdss_band** (*string*) – SDSS band (identified by a character 'u','g','r','i','z')
- **ab_magnitude** (*float*) – AB magnitude to match

Returns rescaled spectrum (*qusp.spectrum.SpectralFluxDensity*)

Raises RuntimeError – in case our spectrum does not fully cover the band.

fiducialFluxUnit
1e-17 erg/(s*cm^2*Ang)

get_ab_magnitudes()
Returns a dictionary of AB magnitudes calculated in each SDSS filter. Magnitude values of None are returned when our spectrum has no extrapolated_value set and a filter extends beyond our tabulated wavelengths.

get_filtered_rates(*filter_curves*, *wavelength_step=1.0*)
Returns the counting rates in photons/(s*cm^2) when our spectral flux density is filtered by the specified curves. The curves should be specified as a dictionary of WavelengthFunctions and the results will also be a dictionary of floats using the same keys. Rates of None are returned when our spectrum has no extrapolated_value set and a filter extends beyond our tabulated wavelengths.

Parameters **filter_curves** (*dict*) – dictionary of WavelengthFunctions

sdss_filter_curves = None

sdss_filter_rates = None

class qusp.spectrum.**Spectrum**(*wavelength*, *flux*, *ivar*)
Bases: object

Represents a BOSS co-added spectrum.

Parameters

- **wavelength** (*numpy.ndarray*) – wavelength pixel centers.
- **flux** (*numpy.ndarray*) – flux values.
- **ivar** (*numpy.ndarray*) – flux inverse variance values.

find_pixel(*wavelength*)
Returns the corresponding pixel index of the specified wavelength.

Parameters **wavelength** (*float*) – value

Returns pixel index

Return type pixelIndex (int)

mean_flux(*min_wavelength*, *max_wavelength*, *ivar_weighting=True*)
Returns the mean flux between the specified wavelengths. Use `ivar_weighting=False` to turn ignore weights.

Parameters

- **min_wavelength** (*float*) – minimum wavelength for mean flux calculation range.
- **max_wavelength** (*float*) – maximum wavelength for mean flux calculation range.
- **ivar_weighting** (*bool, optional*) – Whether or not to weight calculation using inverse variance.

Returns the mean flux between `min_wavelength` and `max_wavelength`.

median_signal_to_noise(*min_wavelength*, *max_wavelength*)
Returns the median signal to noise ratio between the specified wavelengths.

Parameters

- **min_wavelength** (*float*) – minimum wavelength for median flux calculation range.
- **max_wavelength** (*float*) – maximum wavelength for median flux calculation range.

Returns the median flux between `min_wavelength` and `max_wavelength`.

Return type median (float)

class qusp.spectrum.**WavelengthFunction**(*wavelength*, *values*, *wavelengths_units=<Mock id='140258957136912'>*, *value_units=None*, *extrapolated_value=None*)

Bases: object

Represents an arbitrary function of wavelength.

Initializes a function of wavelength using tabulated values at specified wavelengths. The default wavelength units are angstroms but other units can be specified. Optional value units can also be specified but are not required. The input wavelengths must be increasing. The input wavelength and value arrays must have the same size. If either is already a numpy array, no internal copy is made (except when conversion to Angstroms is needed) so these are lightweight objects but be aware of possible side effects.

Parameters

- **wavelength** (*np.ndarray*) – tabulated wavelength values
- **values** (*np.ndarray*) – tabulated values
- **wavelengths_units** (*astropy.units.Quantity*) – wavelength value units
- **value_units** (*astropy.units.Quantity, optional*) – value units
- **extrapolated_value** (*float, optional*) –

get_model()

Returns a model for interpolating within our tabulated wavelength function values. If an extrapolated_value was provided to the constructor, the model will use this for any wavelengths outside the tabulated grid.

Returns model (scipy.interpolate.interp1d)

get_resampled_values(*wavelength*, *wavelengths_units=<Mock id='140258957136016'>*)

Returns a numpy array of values resampled at the specified wavelengths (which do not need to be sorted). The default wavelength units are angstroms but other units can be specified.

Parameters

- **wavelength** (*numpy.ndarray*) – wavelengths to resample at
- **wavelengths_units** (*astropy.Units.Quantity*) – wavelength value units

Returns resampled values (numpy.ndarray)

Raises RuntimeError – if resampling would require an extrapolation but no extrapolated_value was provided to the constructor.

classmethod load_from_text_file(*filename*, *wavelength_column=0*, *values_column=1*, *wavelengths_units=<Mock id='140258957136336'>*, *extrapolated_value=None*)

Returns a new WavelengthFunction (or subclass of WavelengthFunction) from the specified text file. Any comment lines beginning with '#' are ignored. Uses the specified columns for the wavelength and values. Additional columns are allowed and silently ignored. The default wavelength units are Angstroms but other units can be specified. Refer to the WavelengthFunction constructor for details on extrapolated_value.

Parameters **filename** (*string*) –

save_to_text_file(*filename*)

Writes a text file containing two columns: wavelength and values.

Parameters **filename** (*string*) –

qusp.spectrum.**load_sdss_filter_curves**(*which_column=1*)
Loads SDSS filter curves from a standard location within this module. The default which_column=1 corresponds to curves of the quantum efficiency on the sky looking through 1.3 airmasses at APO for a point source. Values of 2-4 are also possible but probably not what you want. Consult the filter data file headers for details.

qusp.spectrum.**read_combined_spectrum**(*spplate*, *fiber*)
Returns the combined spectrum of the specified fiber from the provided spPlate.

Parameters

- **spplate** (*astropy.io.fits.HDUList*) – spPlate file
- **fiber** (*qusp.target.Target*) – boss target's fiberid, or a *qusp.target.Target* object.

Returns a *Spectrum* object of `fiber` of `spplate`.

Return type spectrum (*Spectrum*)

qusp.spectrum.**read_lite_spectrum**(*spec*)
Returns the combined spectrum of the specified fiber from the provided spPlate.

Parameters **fiber** (*qusp.target.Target*) – boss target's fiberid, or a *qusp.target.Target* object.

Returns a *Spectrum* object of `fiber` of `spplate`.

Return type spectrum (*Spectrum*)

1.2.3 qusp.wavelength module

Provides support for working with BOSS wavelengths

Examples

Add sky lines to quasar spectrum plot:

```
>>> qusp.wavelength.draw_lines(qusp.wavelength.load_wavelengths('sky'))
```

Get a combined spectrum's fiducial pixel offset:

```
>>> offset = qusp.wavelength.get_fiducial_pixel_index_offset(np.log10(combined.wave[0]))
```

Construct a fiducial pixel wavelength array:

```
>>> wave = qusp.wavelength.get_fiducial_wavelength(np.arange(4800))
```

---

class qusp.wavelength.**LabeledWavelength**(*value*, *label*)
Bases: *qusp.wavelength.Wavelength*

A LabeledWavelength is a Wavelength with a label attribute

class qusp.wavelength.**Wavelength**(*value*)
Bases: `float`

A Wavelength is a float.

Parameters **value** (*float*) – wavelength value

observed(*redshift*)

Parameters **redshift** (*float*) – source redshift

Returns the shifted observed wavelength.

rest(*redshift*)

Parameters **redshift** (*float*) – source redshift

Returns the shifted rest wavelength.

qusp.wavelength.**draw_lines**(*waves*, *offset=0*, *delta=0.1*, ***kwargs*)
Draws vertical lines on the current plot.

qusp.wavelength.**get_fiducial_pixel_index_offset**(*loglam*, *coeff1=0.0001*, *log10lam0=<Mock name='mock()' id='140258956851920'>*)
Returns the pixel index offset from the start of the BOSS co-add fiducial wavelength grid.

Parameters

- **coeff0** (*float*) – central wavelength (log10) of first pixel
- **coeff1** (*float, optional*) – log10 dispersion per pixel

Returns pixel index offset from the start of the fiducial wavelength grid.

qusp.wavelength.**get_fiducial_wavelength**(*pixel_index*, *coeff1=0.0001*, *log10lam0=<Mock name='mock()' id='140258956851600'>*)
Returns the wavelength at the center of the specified index of the BOSS co-add fiducial wavelength grid.

Parameters **pixel_index** (*int*) – index of the BOSS co-add fiducial wavelength grid.

Returns central wavelength of the specified index on the fiducial wavelength grid

Return type wavelength (float)

qusp.wavelength.**load_wavelengths**(*filename*, *ignore_labels=False*)
Loads a list of wavelengths from the specified file

Parameters **filename** (*str*) – wavelength data filename

Returns wavelengths (list)

1.2.4 qusp.continuum module

Provides classes that represent quasar continuum objects.

class qusp.continuum.**Continuum**
Bases: object

Abstract base class for quasar continuum objects.

get_continuum(*target*, *combined*)
Returns a SpectralFluxDensity object that represent's the specified target's unabsorbed continuum.

Parameters

- **target** (*qusp.target.Target*) – the target
- **combined** (*qusp.spectrum.BOSSSpectrum*) – the target's combined spectrum

class qusp.continuum.**LinearFitContinuum**(*specfits*)
Bases: *qusp.continuum.Continuum*

An interface to linearized continuum fit results file.

Parameters **specfits** (*str*) – name of linearized continuum fit results file.

get_continuum(*target*, *combined*)
Returns a SpectralFluxDensity object that represent's the specified target's unabsorbed continuum.

Parameters

- **target** (*qusp.target.Target*) – the target
- **combined** (*qusp.spectrum.BOSSSpectrum*) – the target's combined spectrum

Raises ValueError – if target is not found in fit results.

class qusp.continuum.**MeanFluxContinuum**(*wave_min=None*, *wave_max=None*)
Bases: *qusp.continuum.Continuum*

A simple continuum estimate calculated using the mean flux for a quasar.

Parameters

- **wave_min** (*float*) – Optional rest frame wavelength for lower bound of mean flux calculation.
- **wave_max** (*float*) – Optional rest frame wavelength for upper bound of mean flux calculation.

get_continuum(*target*, *combined*)
Returns a SpectralFluxDensity object that represent's the specified target's unabsorbed continuum.

Parameters

- **target** (*qusp.target.Target*) – the target
- **combined** (*qusp.spectrum.BOSSSpectrum*) – the target's combined spectrum

Raises ValueError – if mean flux <= 0

1.2.5 qusp.paths module

Provides a class to manage paths to boss data directories.

Examples

Via standard constructor:

```
>>> paths = qusp.paths.Paths(boss_root='/Users/daniel/data/boss', boss_version='v5_7_0')
```

Via environment variables:

```
BOSS_ROOT='/Users/daniel/data/boss'
BOSS_VERSION='v5_7_0'
```

```
>>> paths = qusp.paths.Paths()
```

Via command line arguments:

```
./example --boss-root /Users/daniel/data/boss --boss-version v5_7_0
```

where the example program looks something like:

```
def main():
    parser = argparse.ArgumentParser()
    qusp.Paths.add_args(parser)
    args = parser.parse_args()
    paths = qusp.Paths(**qusp.Paths.from_args(args))
```

---

class qusp.paths.**Paths**(*boss_root=None*, *boss_version=None*)

Bases: object

Manages a path to a boss data directory.

Parameters

- **boss_root** (*str, optional*) – The root boss directory path. Defaults to None, in which case, the environment variable *BOSS_ROOT* is expected to specify the path to the root boss directory.
- **boss_version** (*str, optional*) – The boss version directory name. Defaults to None, in which case, the environment variable *BOSS_VERSION* is expected to specify the boss version directory name.

Raises *RuntimeError if either 'boss_root' or 'boss_version' are not* – specifed by parameters or environment variables.

static add_args(*parser*)

Adds arguments to the provided command-line parser.

Parameters **parser** (*argparse.ArgumentParser*) – an argument parser

static from_args(*args*)

Returns a dictionary of constructor parameter values based on the parsed args provided.

Parameters **args** (*argparse.Namespace*) – argparse argument namespace

Returns a dictionary of *Paths* constructor parameter values

get_spec_filename(*target*, *lite=False*)

get_spplate_filename(*target*)

Returns full path to the specified target's spPlate fits file.

Parameters **target** (*qusp.target.Target*) –

Returns spplate_filename (str)

1.2.6 qusp.model module

Provides support for modeling a universal quasar continuum.

Test laptop dev using:

```
time ./examples/fitspec.py --boss-root ~/data/boss -i test.txt -o fitspec-test -n 100 --verbose --sk
./examples/plotfitspec.py --boss-root ~/data/boss -i fitspec-test.hdf5 -o output/fitspec-test --force
```

Test on darkmatter using:

```
time ./examples/fitspec.py --boss-root /data/boss -i sn-sorted.txt -o fitspec-test -n 1000 --verbose
./examples/plotfitspec.py --boss-root /data/boss -i fitspec-test.hdf5 -o output/fitspec-test --force-
```

Profile on darkmatter:

```
python -m cProfile -o profile-test.out ./examples/fitspec.py --boss-root /data/boss -i sn-sorted.txt
```

List top function calls by time:

```
import pstats
p = pstats.Stats('profile-test.out')
p.sort_stats('time').print_stats(10)
```

Generate call tree:

```
gprof2dot -f pstats profile-test.out | dot -Tpng -o profile-test.png
```

---

class qusp.model.**ContinuumModel**(*transmission_min*, *transmission_max*, *continuum_min*, *continuum_max*, *continuum_nparams*, *tiltwave*, *absorption_min*, *absorption_max*, *absorption_modelexp*, *absorption_scale*, *fix_transmission=False*, *continuum=None*, *verbose=False*)

Bases: object

Represents a linearized quasar continuum model.

Initializes a linearized quasar continuum model using the specified parameter limits and values.

Parameters

- **transmission_min** (*float*) – minimum observed frame wavelength bin center of transmission model.
- **transmission_max** (*float*) – maximum observed frame wavelength bin center of transmission model.
- **continuum_min** (*float*) – minimum rest frame wavelength bin center of continuum model.
- **continuum_max** (*float*) – maximum rest frame wavelength bin center of continuum model.
- **continuum_nparams** (*int*) – number of rest frame bins of continuum model.
- **tiltwave** (*float*) – pivot wavelength of rest frame spectral tilt.
- **absorption_min** (*float*) – minimum rest frame wavelength bin center of absorption model.
- **absorption_max** (*float*) – maximum rest frame wavelength bin center of absorption model.
- **absorption_modelexp** (*float*) – exponent of (1+z) factor of absorption model.
- **absorption_scale** (*float*) – internal scaling of absorption model coefficients.
- **verbose** (*bool, optional*) – whether or not to print verbose output.

static add_args(*parser*)

Add arguments to the provided command-line parser.

Parameters **parser** (*argparse.ArgumentParser*) – an argument parser

add_continuum_constraint(*yvalue*, *wavemin*, *wavemax*, *weight*)
Adds a constraint equation on the geometric mean of the continuum model in between the specified rest frame wavelengths.

Parameters

- **yvalue** (*float*) – y value of constraint equation.
- **wavemin** (*float*) – minimum rest frame wavelength bin center to constrain.
- **wavemax** (*float*) – maxmimum rest frame wavelength bin center to constrain.
- **weight** (*float*) – weight to apply to constraint equation.

add_observation(*target*, *flux*, *wave*, *ivar*, *unweighted=True*)
Adds an observation to be fit. Returns the number of pixels added. The provided target argument must have an attribute named 'z' with the target's redshift.

Note: Weighted fit not yet implemented.

Parameters

- **target** (*qusp.target.Target*) – a Target object
- **flux** (*numpy.array*) – flux array
- **wave** (*numpy.array*) – wavelength array
- **ivar** (*numpy.array*) – ivar array
- **unweighted** (*bool, optional*) – ignore pixel variances. Defaults to True.

Returns number of pixels added to model from this observation

Return type npixels (int)

add_tilt_constraint(*weight*)
Adds a constraint equation on the mean of the non-fixed spectral tilt params.

Parameters **weight** (*float*) – weight to apply to constraint equation.

add_transmission_constraint(*yvalue*, *wavemin*, *wavemax*, *weight*)
Adds a constraint equation for each of the transmission model params between the specified observed frame wavelengths.

Parameters

- **yvalue** (*float*) – y value of constraint equation.
- **wavemin** (*float*) – minimum observed frame wavelength bin center to constrain.
- **wavemax** (*float*) – maxmimum observed frame wavelength bin center to constrain.
- **weight** (*float*) – weight to apply to constraint equation.

finalize()
Does final assembly of the sparse matrix representing the model.

static from_args(*args*)
Returns a dictionary of constructor parameter values based on the parsed args provided.

Parameters **args** (*argparse.Namespace*) – argparse argument namespace

Returns a dictionary of *ContinuumModel* constructor parameter values

get_chisq(*soln*)
Calculates the chi-squared between the specified solution and the model y values.

Parameters **soln** (*numpy.array*) – model parameter solution array

Returns value

Return type chisq (float)

get_model()
Returns the assembled model matrix and corresponding y values

Returns

A tuple of (model, yvalues), where *model* is a scipy.sparse.csc_matrix and *yvalues* is a numpy.array.

get_obs_chisq(*soln*, *obs_index*)
Returns the chi-squared value of the specified observation index, *obs_index*, using the specified soln.

Parameters

- **soln** (*numpy.array*) – model parameter solution array
- **obs_index** (*int*) – observation index

Returns value

Return type chisq (float)

get_results(*soln*)
Converts the provided solution to model params. Transforms log params to linear and inserts fixed model params.

Returns a dictionary of model params

Return type results (dict)

save(*filename*, *soln*, *args*, *save_model=True*, *save_chisq=True*)
Saves soln to the specified filename as an hdf5 file. Parsed results and fit meta data are also saved. Use the saveModel arg to indicate whether or not to save the raw data of the sparse matrix model.

Parameters

- **filename** (*str*) – filename of the hdf5 output to create
- **soln** (*numpy.array*) – model parameter solution array
- **args** (*argparse.Namespace*) – argparse argument namespace
- **save_model** (*bool, optional*) – whether or not to save the model matrix and y values. Defaults to True.
- **save_chisq** (*bool, optional*) – whether or not to save per observation chisq values. Defaults to True.

Returns the output hdf5 file created

Return type outfile (h5py.File)

1.3 Example Programs

1.3.1 filter

A program to select entries from FITS catalogs and create target lists.

Filter targets from spAll:

```
$ examples/filter.py -i spAll-v5_7_0.fits \
    --select "(['OBJTYPE'] == 'QSO') & (['Z'] > 2.1)" \
    --annotate 'ra:dec:z' --verbose --save quasars.txt
```

1.3.2 fitspec

A progam for performing simultaneous least squares fits of BOSS quasar spectra to a model that includes:

- universal quasar continuum function
- redshift dependent $Ly\alpha$ absorption model
- observed frame transmission function
- individual quasar parameters (amplitude, spectral tilt)

CHAPTER 2

Developer Documentation

Info on how to build this documentation and other miscellany

2.1 Developement

Contents

2.1.1 Documentation

Build

Requires sphinx extensions to build:

```
$ pip install sphinx_bootstrap_theme
$ pip install sphinxcontrib-napoleon
$ pip install sphinxcontrib-programoutput
```

Build instructions:

```
$ cd docs
$ make html
$ open _build/html/index.html
```

Initial setup

```
$ mkdir docs
$ cd docs
$ sphinx-quickstart
```

Edit config.py and specify sys.path to top level:

```
sys.path.insert(0, os.path.abspath('..'))
```

```
$ sphinx-apidoc -o src ../qusp --separate
```

2.1.2 Profiling

Run program w/ profiler:

```
$ python -m cProfile -o profile.out <program> <args>
```

View stats in interactive session:

```
import pstats
p = pstats.Stats('profile.out')
p.sort_stats('time').print_stats(10)
```

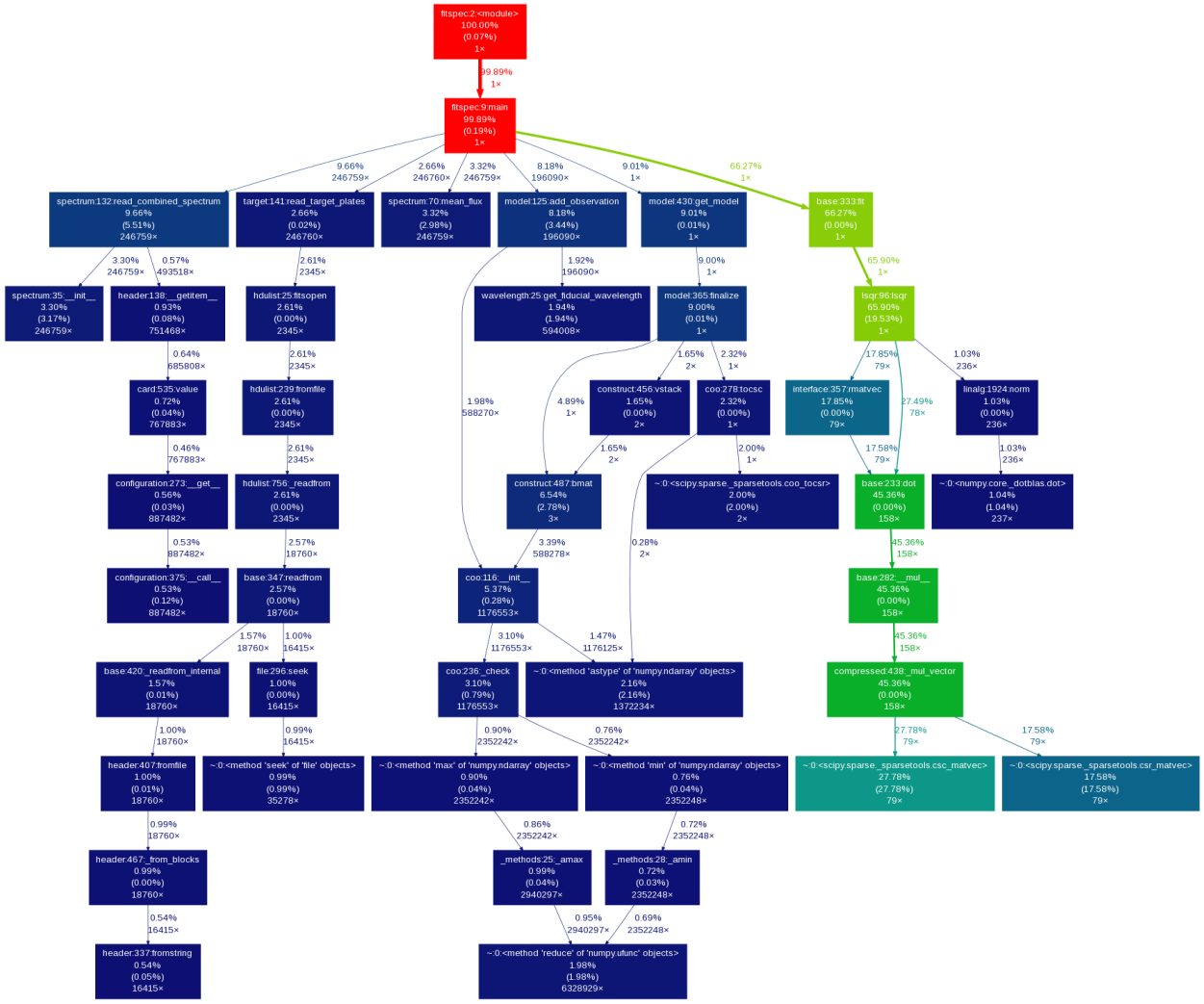
Create a call tree diagram:

```
$ gprof2dot -f pstats profile.out | dot -Tpng -o profile.png
```

Might need to install `gprof2dot`:

```
$ pip install gprof2dot
```

Here is an example call tree diagram:



2.1.3 Code Style Guide

Run pylint to help keep everything nice and pretty.

```
$ pylint qusp/*.py --reports=no --extension-pkg-whitelist=numpy
```

The `pylintrc` file in the toplevel directory specifies configuration options. For example, we ignore `bad-continuation` and extend the character limit per line.

Here is a quick find `,([^\s])` and replace `, \1` regex pattern for adding spaces after commas.

2.1.4 Data transfers

Copy spPlates from darkmatter to hpc:

```
$ rsync -avz --prune-empty-dirs --include '*/' --include 'spPlate*.fits' --exclude '*' -e ssh dmargal
```

Copy spAll from darkmatter to hpc:

```
$ scp dmargala@darkmatter.ps.uci.edu:/data/boss/spAll-v5_7_0.fits /share/dm/all/data/boss/
```

Create target list from lists of plates:

```
for plate in $(cat ~/blue-plates.txt); \
do \
examples/filter.py -i /share/dm/all/data/boss/spAll-v5_7_0.fits \
    --select "(['plate'] == $plate) & (['objtype'] == 'QSO') & (['zwarning'] == 0) & (['z'] > .5)" \
    --save systematics/$plate.txt --annotate 'ra:dec:z' --verbose; \
done
```

2.1.5 Environment Setup

HPC

Only tested using interactive session so far...

```
qrsh -q dm
```

Use a local user install of anaconda for python.

```
module purge
export PATH=/data/users/dmargala/anaconda/bin:$PATH
export PYTHONPATH=/data/users/dmargala/source/qusp

export BOSS_ROOT=/share/dm/all/data/boss
export BOSS_VERSION=v5_7_0
```

Darkmatter

```
export PYTHONPATH=/home/dmargala/source/qusp

export BOSS_ROOT=/data/boss
export BOSS_VERSION=v5_7_0
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

q

A

B

C

D

F

G

L

M

O

P

Q

R

S

T

W