laspec

Release 2022.804.0

Bo Zhang

May 08, 2023

TUTORIALS (WILL BE UPDATED SOON)

A toolkit for LAMOST spectra.

CHAPTER

ONE

AUTHOR

Bo Zhang, LAMOST Fellow at Beijing Normal University.

- bozhang@nao.cas.cn
- bozhang@bnu.edu.cn

CHAPTER

TWO

INSTALLATION

- for the github version (recommended):

```
pip install -U git+git://github.com/hypergravity/laspec
```

- for the PYPI version (usually behind the github version):

```
pip install -U laspec
```

2.1 How to read LAMOST MRS spectra

to be updated …

2.2 How to measure stellar Radial Velocity

to be updated …

2.3 laspec API

2.3.1 laspec.bh module

laspec.bh.**calculate_m2**(*Pday=14.5*, *ideg=90*, *m1=2.16*, *Kkms=50*)

Calculate the mass of the secondary

Parameters

- **Pday** (*float*) – Period in days
- **ideg** – inclination in degrees
- **m1** – the mass of the primary in Msun
- **Kkms** – the K of the primary in km/s

Returns

the mass of the secondary

Return type

M2

2.3.2 laspec.binning module

laspec.binning.**center2edge**(*x*)

laspec.binning.**rebin**(*wave*, *flux=None*, *flux_err=None*, *mask=None*, *wave_new=None*)

Rebin spectrum to a new wavelength grid

Parameters

- **wave** (*array*) – old wavelength
- **flux** (*array*) – old flux
- **flux_err** (*array (optional)*) – old flux error
- **mask** (*array (optional)*) – old mask, True for bad.
- **wave_new** – new wavelength. if None, use log10 wavelength.

Return type

re-binned (flux, [flux_err], [mask])

2.3.3 laspec.ccf module

class laspec.ccf.**RVM**(*pmod*, *wave_mod*, *flux_mod*, *npix_lv=0*)

Bases: object

ccf_1mod(*wave_mod*, *flux_mod*, *wave_obs*, *flux_obs*, *w_mod=None*, *w_obs=None*, *sinebell_idx=0.0*, *rv_grid=array([-600., -587.87878788, -575.75757576, -563.63636364, -551.51515152, -539.39393939, -527.27272727, -515.15151515, -503.03030303, -490.90909091, -478.78787879, -466.66666667, -454.54545455, -442.42424242, -430.3030303, -418.18181818, -406.06060606, -393.93939394, -381.81818182, -369.6969697, -357.57575758, -345.45454545, -333.33333333, -321.21212121, -309.09090909, -296.96969697, -284.84848485, -272.72727273, -260.60606061, -248.48484848, -236.36363636, -224.24242424, -212.12121212, -200., -187.87878788, -175.75757576, -163.63636364, -151.51515152, -139.39393939, -127.27272727, -115.15151515, -103.03030303, -90.90909091, -78.78787879, -66.66666667, -54.54545455, -42.42424242, -30.3030303, -18.18181818, -6.06060606, 6.06060606, 18.18181818, 30.3030303, 42.42424242, 54.54545455, 66.66666667, 78.78787879, 90.90909091, 103.03030303, 115.15151515, 127.27272727, 139.39393939, 151.51515152, 163.63636364, 175.75757576, 187.87878788, 200., 212.12121212, 224.24242424, 236.36363636, 248.48484848, 260.60606061, 272.72727273, 284.84848485, 296.96969697, 309.09090909, 321.21212121, 333.33333333, 345.45454545, 357.57575758, 369.6969697, 381.81818182, 393.93939394, 406.06060606, 418.18181818, 430.3030303, 442.42424242, 454.54545455, 466.66666667, 478.78787879, 490.90909091, 503.03030303, 515.15151515, 527.27272727, 539.39393939, 551.51515152, 563.63636364, 575.75757576, 587.87878788, 600.])*, *flux_bounds=(0, 3.0)*)

measure RV

chi2_1mod(*imod*, *wave_obs*, *flux_obs*, *rv_grid=array([-600., -587.87878788, -575.75757576, -563.63636364, -551.51515152, -539.39393939, -527.27272727, -515.15151515, -503.03030303, -490.90909091, -478.78787879, -466.66666667, -454.54545455, -442.42424242, -430.3030303, -418.18181818, -406.06060606, -393.93939394, -381.81818182, -369.6969697, -357.57575758, -345.45454545, -333.33333333, -321.21212121, -309.09090909, -296.96969697, -284.84848485, -272.72727273, -260.60606061, -248.48484848, -236.36363636, -224.24242424, -212.12121212, -200., -187.87878788, -175.75757576, -163.63636364, -151.51515152, -139.39393939, -127.27272727, -115.15151515, -103.03030303, -90.90909091, -78.78787879, -66.66666667, -54.54545455, -42.42424242, -30.3030303, -18.18181818, -6.06060606, 6.06060606, 18.18181818, 30.3030303, 42.42424242, 54.54545455, 66.66666667, 78.78787879, 90.90909091, 103.03030303, 115.15151515, 127.27272727, 139.39393939, 151.51515152, 163.63636364, 175.75757576, 187.87878788, 200., 212.12121212, 224.24242424, 236.36363636, 248.48484848, 260.60606061, 272.72727273, 284.84848485, 296.96969697, 309.09090909, 321.21212121, 333.33333333, 345.45454545, 357.57575758, 369.6969697, 381.81818182, 393.93939394, 406.06060606, 418.18181818, 430.3030303, 442.42424242, 454.54545455, 466.66666667, 478.78787879, 490.90909091, 503.03030303, 515.15151515, 527.27272727, 539.39393939, 551.51515152, 563.63636364, 575.75757576, 587.87878788, 600.])*, *pw=2*, *flux_bounds=(0, 3.0)*)

measure RV

delete_cache(*cache_name='B'*)

make_cache(*cache_name='B'*, *wave_range=(5000, 5300)*, *rv_grid=(-1000, 1000, 10)*)

make cache for fast RV measurements

Parameters

- **cache_name** – suffix of cached data
- **wave_range** – wavelength bounds
- **rv_grid** – rv_start, rv_stop, rv_step

measure(*wave_obs*, *flux_obs*, *flux_err=None*, *w_mod=None*, *w_obs=None*, *sinebell_idx=0.0*, *rv_grid=(-600, 600, 10)*, *flux_bounds=(0, 3.0)*, *nmc=100*, *method='BFGS'*, *cache_name=None*, *return_ccfgrid=False*)

measure RV

Parameters

- **wave_obs** – observed wavelength
- **flux_obs** – observed flux (normalized)
- **flux_err** – observed flux error
- **w_mod** – if cache, not used
- **w_obs** – if cache, not used
- **sinebell_idx** – sin(flux)**sinebess_idx
- **rv_grid** – if cache, use the cached rv_grid else, use this rv_grid
- **flux_bounds** – flux bounds
- **nmc** – number of MC repeats
- **method** – optimization method
- **cache_name** – cache name. if None: no acceleration; if "vector": partial acceleration.
- **return_ccfgrid** – if True, return ccfgrid

measure2(*wave_obs*, *flux_obs*, *flux_err*, *wave_mod1*, *flux_mod1*, *wave_mod2*, *flux_mod2*, *w_obs=None*, *rv1_init=0*, *eta_init=0.3*, *eta_lim=(0.1, 1.0)*, *drvmax=500*, *drvstep=5*, *method='Powell'*, *nmc=100*)

given a template, optimize (rv1, drv, eta)

measure_binary(*wave_obs*, *flux_obs*, *flux_err=None*, *w_obs=None*, *cache_name='B'*, *rv_grid=array([-600., -587.87878788, -575.75757576, -563.63636364, -551.51515152, -539.39393939, -527.27272727, -515.15151515, -503.03030303, -490.90909091, -478.78787879, -466.66666667, -454.54545455, -442.42424242, -430.3030303, -418.18181818, -406.06060606, -393.93939394, -381.81818182, -369.6969697, -357.57575758, -345.45454545, -333.33333333, -321.21212121, -309.09090909, -296.96969697, -284.84848485, -272.72727273, -260.60606061, -248.48484848, -236.36363636, -224.24242424, -212.12121212, -200., -187.87878788, -175.75757576, -163.63636364, -151.51515152, -139.39393939, -127.27272727, -115.15151515, -103.03030303, -90.90909091, -78.78787879, -66.66666667, -54.54545455, -42.42424242, -30.3030303, -18.18181818, -6.06060606, 6.06060606, 18.18181818, 30.3030303, 42.42424242, 54.54545455, 66.66666667, 78.78787879, 90.90909091, 103.03030303, 115.15151515, 127.27272727, 139.39393939, 151.51515152, 163.63636364, 175.75757576, 187.87878788, 200., 212.12121212, 224.24242424, 236.36363636, 248.48484848, 260.60606061, 272.72727273, 284.84848485, 296.96969697, 309.09090909, 321.21212121, 333.33333333, 345.45454545, 357.57575758, 369.6969697, 381.81818182, 393.93939394, 406.06060606, 418.18181818, 430.3030303, 442.42424242, 454.54545455, 466.66666667, 478.78787879, 490.90909091, 503.03030303, 515.15151515, 527.27272727, 539.39393939, 551.51515152, 563.63636364, 575.75757576, 587.87878788, 600.])*, *flux_bounds=(0, 3.0)*, *twin=True*, *eta_init=0.3*, *eta_lim=(0.01, 3.0)*, *drvmax=500*, *drvstep=5*, *method='Powell'*, *nmc=100*, *suffix=''*)

measure_binary_mrsbatch(*fp*, *lmjm*, *snr_B=None*, *snr_R=None*, *snr_threshold=5*, *raise_error=False*)

measure_pw(*wave_obs*, *flux_obs*, *rv_grid=array([-600., -587.87878788, -575.75757576, -563.63636364, -551.51515152, -539.39393939, -527.27272727, -515.15151515, -503.03030303, -490.90909091, -478.78787879, -466.66666667, -454.54545455, -442.42424242, -430.3030303, -418.18181818, -406.06060606, -393.93939394, -381.81818182, -369.6969697, -357.57575758, -345.45454545, -333.33333333, -321.21212121, -309.09090909, -296.96969697, -284.84848485, -272.72727273, -260.60606061, -248.48484848, -236.36363636, -224.24242424, -212.12121212, -200., -187.87878788, -175.75757576, -163.63636364, -151.51515152, -139.39393939, -127.27272727, -115.15151515, -103.03030303, -90.90909091, -78.78787879, -66.66666667, -54.54545455, -42.42424242, -30.3030303, -18.18181818, -6.06060606, 6.06060606, 18.18181818, 30.3030303, 42.42424242, 54.54545455, 66.66666667, 78.78787879, 90.90909091, 103.03030303, 115.15151515, 127.27272727, 139.39393939, 151.51515152, 163.63636364, 175.75757576, 187.87878788, 200., 212.12121212, 224.24242424, 236.36363636, 248.48484848, 260.60606061, 272.72727273, 284.84848485, 296.96969697, 309.09090909, 321.21212121, 333.33333333, 345.45454545, 357.57575758, 369.6969697, 381.81818182, 393.93939394, 406.06060606, 418.18181818, 430.3030303, 442.42424242, 454.54545455, 466.66666667, 478.78787879, 490.90909091, 503.03030303, 515.15151515, 527.27272727, 539.39393939, 551.51515152, 563.63636364, 575.75757576, 587.87878788, 600.])*, *method='BFGS'*, *pw=1*)

mock_binary_spectrum(*imod1*, *imod2*, *rv1*, *drv*, *eta*)

mrsbatch(*fpout*, *fp_list*, *lmjm_list*, *snr_B_list*, *snr_R_list*, *snr_threshold=5*)

reproduce_spectrum_binary(*rvr*)

reproduce the spectrum (binary)

reproduce_spectrum_single(*rvr*)

reproduce the spectrum (single)

shrink(*nmod=0.5*, *method='top'*)

laspec.ccf.**calculate_local_variance**(*flux*, *npix_lv: int = 5*) → ndarray

calculate local variance

laspec.ccf.**calculate_local_variance_multi**(*flux*, *npix_lv: int = 5*, *n_jobs: int = -1*, *verbose: int = 10*) → ndarray

calculate local variance

laspec.ccf.**respw_cost**(*rv*, *wave_obs*, *flux_obs*, *wave_mod*, *flux_mod*, *pw=1*)

laspec.ccf.**respw_rvgrid**(*wave_obs*, *flux_obs*, *wave_mod*, *flux_mod*, *pw=1*, *rv_grid=array([-500, -490, -480, -470, -460, -450, -440, -430, -420, -410, -400, -390, -380, -370, -360, -350, -340, -330, -320, -310, -300, -290, -280, -270, -260, -250, -240, -230, -220, -210, -200, -190, -180, -170, -160, -150, -140, -130, -120, -110, -100, -90, -80, -70, -60, -50, -40, -30, -20, -10, 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500])*)

laspec.ccf.**sinebell**(*n=1000*, *index=0.5*)

sine bell to left & right end of spectra

laspec.ccf.**sinebell_like**(*x*, *index=0.5*)

laspec.ccf.**test_lmfit**()

load data

laspec.ccf.**test_new_rvm**()

laspec.ccf.**test_sinebell**()

laspec.ccf.**test_sinebell2**()

load data

laspec.ccf.**test_xcorr_rvgrid**()

load data

laspec.ccf.**wcov**(*x1*, *x2*, *w=None*)

weighted covariance

laspec.ccf.**wmean**(*x*, *w=None*)

weighted mean

laspec.ccf.**wxcorr**(*x1*, *x2*, *w=None*)

weighted cross-correlation

laspec.ccf.**wxcorr_rvgrid**(*wave_obs*, *flux_obs*, *wave_mod*, *flux_mod*, *rv_grid*, *w_mod=None*, *w_obs=None*)

weighted cross-correlation method Interpolate a model spectrum with different RV and cross-correlate with the observed spectrum, return the CCF on the RV grid.

wave_obs: array
wavelength of observed spectrum (normalized)

flux_obs: array
flux of observed spectrum

wave_mod: array
wavelength of model spectrum (normalized)

flux_mod:
flux of model spectrum

mask_obs:
True for bad pixels

rv_grid:
km/s RV grid

laspec.ccf.**wxcorr_rvgrid_binary**(*wave_obs*, *flux_obs*, *wave_mod1*, *flux_mod1*, *wave_mod2*, *flux_mod2*, *flux_err=None*, *rv1_init=0*, *eta_init=0.3*, *eta_lim=(0.1, 1.2)*, *drvmax=500*, *drvstep=5*, *w_obs=None*, *method='Powell'*, *nmc=100*)

laspec.ccf.**wxcorr_spec**(*rv*, *wave_obs*, *flux_obs*, *wave_mod*, *flux_mod*, *w_mod=None*, *w_obs=None*)

weighted cross correlation of two spectra

laspec.ccf.**wxcorr_spec_binary**(*rv1*, *rv2*, *eta*, *wave_obs*, *flux_obs*, *wave_mod1*, *flux_mod1*, *wave_mod2*, *flux_mod2*, *w_obs=None*)

weighted cross correlation of two spectra .. note:: w_mod is not supported in this case

laspec.ccf.**wxcorr_spec_cost**(*rv*, *wave_obs*, *flux_obs*, *wave_mod*, *flux_mod*, *w_mod=None*, *w_obs=None*)

the negative of wxcorr_spec, used as cost function for minimiztion

laspec.ccf.**wxcorr_spec_cost_binary**(*rv1_rv2_eta*, *wave_obs*, *flux_obs*, *wave_mod1*, *flux_mod1*, *wave_mod2*, *flux_mod2*, *w_obs=None*, *eta_lim=(0.1, 1.2)*)

the negative of wxcorr_spec, used as cost function for minimiztion

laspec.ccf.**wxcorr_spec_fast**(*rv_grid*, *wave0*, *flux0*, *wave1*, *flux1*, *w_mod=None*, *w_obs=None*)

vectorized for multiple model flux, but w_mod, w_obs are not considered!

laspec.ccf.**wxcorr_spec_twin**(*rv1*, *drv*, *eta*, *wave_obs*, *flux_obs*, *wave_mod*, *flux_mod*, *w_obs=None*)

weighted cross correlation of two spectra .. note:: w_mod is not supported in this case

laspec.ccf.**xcorr_rvgrid**(*wave_obs*, *flux_obs*, *wave_mod*, *flux_mod*, *mask_obs=None*, *rv_grid=array([-500, -490, -480, -470, -460, -450, -440, -430, -420, -410, -400, -390, -380, -370, -360, -350, -340, -330, -320, -310, -300, -290, -280, -270, -260, -250, -240, -230, -220, -210, -200, -190, -180, -170, -160, -150, -140, -130, -120, -110, -100, -90, -80, -70, -60, -50, -40, -30, -20, -10, 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500])*)

a naive cross-correlation method Interpolate a model spectrum with different RV and cross-correlate with the observed spectrum, return the CCF on the RV grid.

wave_obs: array
wavelength of observed spectrum (normalized)

flux_obs: array
flux of observed spectrum

wave_mod: array
wavelength of model spectrum (normalized)

flux_mod:
flux of model spectrum

mask_obs:
True for bad pixels

rv_grid:
km/s RV grid

2.3.4 laspec.convolution module

laspec.convolution.**conv_spec**(*wave*, *flux*, *R_hi=2000.0*, *R_lo=500.0*, *over_sample_additional=3.0*, *gaussian_kernel_sigma_num=5.0*, *wave_new=None*, *wave_new_oversample=3.0*, *verbose=True*, *return_type='array'*)

to convolve high-R spectrum to low-R spectrum

Parameters

- **wave** (*array*) – wavelength
- **flux** (*array*) – flux array
- **R_hi** (*float or function*) – higher resolution
- **R_lo** (*float or function*) – lower resolution
- **over_sample_additional** (*float*) – additional over-sample rate
- **gaussian_kernel_sigma_num** (*float*) – the gaussian kernel width in terms of sigma
- **wave_new** (*None or float or array*) – if None: wave_new auto-generated using wave_new_oversample if float: this specifies the over-sample rate if voctor: this specifies the new wave_new array
- **wave_new_oversample** – if wave_new is None, use auto-generated wave_new_oversample
- **verbose** (*bool*) – if True, print the details on the screen
- **return_type** (*string*) – if 'array': return wave and flux as array if 'table': retrun spec object

Return type
wave_new, flux_new OR Table([wave, flux])

laspec.convolution.**find_R_for_wave_array**(*wave*)

find the R of wavelength array (sampling resolution array)

laspec.convolution.**find_R_max_for_wave_array**(*wave*)

find the maximum sampling resolution of a given wavelength array

laspec.convolution.**find_Rgk**(*R_hi=2000.0*, *R_lo=500.0*, *over_sample=1.0*)

find Rgk as a function of wavelength

Parameters

- **R_hi** (*float or funtion*) – higher resolution (as a function of wavelength)
- **R_lo** (*float or funtion*) – lower resolution (as a function of wavelength)
- **over_sample** (*float*) – over-sampled resolution, default is 1.

Returns
Rgk – Gaussian Kernel resolution as a function of wavelength

Return type
function

laspec.convolution.**find_delta_lambda_for_wave_array**(*wave*)

find the delta_lambda of wavelength array (delta_lambda array)

laspec.convolution.**find_delta_lambda_min_for_wave_array**(*wave*)

find the minimum delta_lambda of a given wavelength array

laspec.convolution.**fwhm2resolution**(*fwhm*, *wave=5000.0*)

laspec.convolution.**fwhm2sigma**(*fwhm*)

laspec.convolution.**generate_gaussian_kernel_array**(*over_sample_Rgk*, *sigma_num*)

generate gaussian kernel array according to over_sample_Rgk

Parameters

- **over_sample_Rgk** (*float*) – over_sample rate
- **sigma_num** (*float*) – 1 sigma of the Gaussian = sigma_num pixels

Return type

normalized gaussian array

laspec.convolution.**generate_wave_array_R**(*wave_start*, *wave_stop*, *R=2000.0*, *over_sample=1.0*, *wave_test_step=1.0*)

generate a wavelength array matching the given R

Parameters

- **wave_start** (*float*) – start from this wavelength
- **wave_stop** (*float*) – stop at this wavelength
- **R** (*float or function*) – specify a fixed R or specify R as a function of wavelength
- **over_sample** (*float*) – over-sampling rate, default is 1.
- **wave_test_step** – used to determing the wave_step_min

Returns

wave – an array matching the given R

Return type

array

Example

```
>>> def R(x): return 0.2*x
>>> wave_array_R = generate_wave_array_R(4000., 5000., R)
```

laspec.convolution.**generate_wave_array_delta_lambda**(*wave_start*, *wave_stop*, *delta_lambda=<function <lambda>>*, *over_sample=1.0*, *wave_test_step=1.0*)

generate a wavelength array matching the given delta_lambda a function of wavelength

Parameters

- **wave_start** (*float*) – where the wavelength starts
- **wave_stop** (*float*) – where the wavelength stops
- **delta_lambda** (*float or function*) – specifies the delta_lambda as a fixed number or a function of wavelength

- **over_sample** (*float*) – over-sampling
- **wave_test_step** (*float*) – tests for the smallest wave_guess step

Returns
wave_guess – the guess of wavelength array

Return type
array

Example

```
>>> def dl(x): return 0.002*x
>>> wave_array_dl = generate_wave_array_delta_lambda(4000., 5000., dl)
```

laspec.convolution.**normalized_gaussian_array**(*x*, *b=0.0*, *c=1.0*)

laspec.convolution.**read_phoenix_sun**()
read PHOENIX synthetic spectrum for the Sun

laspec.convolution.**resolution2fwhm**(*R*, *wave=5000.0*)

laspec.convolution.**sigma2fwhm**(*sigma*)

laspec.convolution.**test_conv_phoenix_sun**()
testing convolving PHOENIX synthetic spectrum for the Sun

2.3.5 laspec.core module

laspec.core.**test_wpercentile**()

laspec.core.**wmean**(*x*, *w=None*)
weighted mean

laspec.core.**wpercentile**(*x*, *w*, *q*, *eps=0.0*)
weighted percentile

laspec.core.**wstd**(*x*, *w=None*)
Weighted standard deviation

Parameters

- **x** (*array*) – data
- **w** (*array*) – weights

Returns

- *wstd*
- *Ref*
- —
- **https** (*//www.itl.nist.gov/div898/software/dataplot/refman2/ch2/weightsd.pdf*)

2.3.6 laspec.helper module

laspec.helper.**laspec_path**()

laspec.helper.**stilts_path**()

laspec.helper.**test**()

2.3.7 laspec.interpolate module

class laspec.interpolate.**Interp1q**(*x*, *y*, *method='linear'*, *fill_value=nan*, *cushion=100000.0*, *issorted=True*)

Bases: object

2.3.8 laspec.lamost module

laspec.lamost.**lamost_filepath**(*planid*, *mjd*, *spid*, *fiberid*, *dirpath=''*, *extname='.fits'*)

generate file path of a LAMOST spectrum

Parameters

- **planid** (*string*) – planid
- **mjd** (*5-digit integer*) – mjd (use lmjd rather than mjd for DR3 and after!)
- **spid** (*2-digit integer*) – spid, the number of the spectrogragh
- **fiberid** (*3-digit integer*) – fiberid
- **dirpath** (*string*) – the root directory for storing spectra.

Returns
filepath – the path of root dir of directory (prefix). if un-specified, return file name.

Return type
string

laspec.lamost.**lamost_filepath_med**(*planid*, *mjd*, *spid*, *fiberid*, *dirpath=''*, *extname='.fits'*)

generate file path of a LAMOST spectrum (medium resolution)

Parameters

- **planid** (*string*) – planid
- **mjd** (*5-digit integer*) – mjd (use lmjd rather than mjd for DR3 and after!)
- **spid** (*2-digit integer*) – spid, the number of the spectrogragh
- **fiberid** (*3-digit integer*) – fiberid
- **dirpath** (*string*) – the root directory for storing spectra.

Returns
filepath – the path of root dir of directory (prefix). if un-specified, return file name.

Return type
string

laspec.lamost.**sdss_filepath**(*plate*, *mjd*, *fiberid*, *dirpath=''*, *extname='.fits'*)

generate file path of a LAMOST spectrum

Parameters

- **plate** (*string*) – plate
- **mjd** (*5-digit integer*) – mjd (use lmjd rather than mjd for DR3 and after!)
- **fiberid** (*4-digit integer*) – fiberid
- **dirpath** (*string*) – the root directory for storing spectra.
- **extname** (*string*) – in case that users want to synthesize other data format

Returns

filepath – the path of root dir of directory (prefix). if un-specified, return file name.

Return type

string

2.3.9 laspec.light_curve module

laspec.light_curve.**read_lightcurve**(*fp='/Users/cham/projects/sb2/lightcurve/lc/lc488994.dat'*)

2.3.10 laspec.line_index module

laspec.line_index.**get_equivalent_width**(*line_indx_star*)

laspec.line_index.**integrate_spectrum**(*wave*, *flux_norm*, *flux_norm_err=None*, *mask=None*, *nmc=50*, *wave_range=(6554, 6574)*, *suffix='Ha'*)

laspec.line_index.**measure_line_index**(*wave*, *flux*, *flux_err=None*, *mask=None*, *z=None*, *line_info=None*, *num_refit=(100, None)*, *filepath=None*, *return_type='dict'*, *verbose=False*)

Measure line index / EW and have it plotted

Parameters

- **wave** (*array-like*) – wavelength vector
- **flux** (*array-like*) – flux vector
- **flux_err** (*array-like*) – flux error vector (optional)

  If un-specified, auto-generate an np.ones array
- **mask** (*array-like*) – andmask or ormask (optional)

  If un-specified, auto-generate an np.ones array (evenly weighted)
- **line_info** (*dict*) – information about spectral line, eg:

```
>>> line_info_dib5780 = {'line_center':         5780,
>>>                      'line_range':          (5775, 5785),
>>>                      'line_shoulder_left':  (5755, 5775),
>>>                      'line_shoulder_right': (5805, 5825)}
```

- **num_refit** (*non-negative integer*) – number of refitting.

  If 0, no refit will be performed

  If positive, refits will be performed after adding normal random noise

- **z** (*float*) – redshift (only specify when z is large)
- **filepath** (*string*) – path of the diagnostic figure

  if None, do nothing, else print diagnostic figure

- **return_type** (*string*) – 'dict' or 'array'

  if 'array', np.array(return dict.values())

- **verbose** (*bool*) – if True, print details

Returns
line_indx – A dictionary type result of line index. If any problem encountered, return the default result (filled with nan).

Return type
dict

laspec.line_index.**measure_line_index_loopfun**(*filepath*)

loopfun for measuring line index

Parameters
filepath (*string*) – path of the spec document

Returns
several line_indx – every line_indx is a dictionary type result of line index.

Return type
tuple

laspec.line_index.**measure_line_index_null_result**(*return_type*)

generate default value (nan/False) when measurement fails :rtype: default value (nan/False)

laspec.line_index.**measure_line_index_recover_spectrum**(*wave*, *params*, *norm=False*)

recover the fitted line profile from params

Parameters

- **wave** (*array-like*) – the wavelength to which the recovered flux correspond
- **params** (*5-element tuple*) – the 1 to 5 elements are: mod_linear_slope mod_linear_intercept mod_gauss_amplitude mod_gauss_center mod_gauss_sigma
- **norm** (*bool*) – if True, linear model (continuum) is deprecated else linear + Gaussian model is used

laspec.line_index.**plot_equivalent_width_hist**(*EW_star*)

laspec.line_index.**plot_line_indices**(*EW_star*)

laspec.line_index.**save_image_line_indice**(*filepath*, *wave*, *flux*, *ind_range*, *cont_range*, *ind_shoulder*, *line_info*)

Plot a line indice and save it as a .png document.

Parameters

- **filepath** (*string*) – path of the spec document

- **wave** (*array*) – wavelength vector
- **flux** (*array*) – flux vector
- **ind_range** (*array*) – bool indicating the middle range of a particular line
- **cont_range** (*array*) – continuum flux of the middle range derived from linear model
- **ind_shoulder** (*array*) – bool indicating the shoulder range of a particular line
- **line_info** (*dict*) – information about spectral line, eg:

```
>>> line_info_dib5780 = {'line_center':         5780,
>>>                      'line_range':          (5775, 5785),
>>>                      'line_shoulder_left':  (5755, 5775),
>>>                      'line_shoulder_right': (5805, 5825)}
```

laspec.line_index.**test_**()

laspec.line_index.**test_measure_line_index**()

laspec.line_index.**walk_dir**(*dirpath*)

enumerate all files under dirpath

Parameters
dirpath (*string*) – the directory to be walked in

Returns
filename – filepaths of all the spectra in finder dirpath

Return type
list

2.3.11 laspec.mpl module

laspec.mpl.**set_cham**(*fontsize=15*, *xminor=True*, *yminor=True*, *latex=True*)

laspec.mpl.**set_xminor**(*b=True*)

laspec.mpl.**set_yminor**(*b=True*)

2.3.12 laspec.mrs module

class laspec.mrs.**MrsEpoch**(*speclist*, *specnames=('B', 'R')*, *epoch=-1*, *norm_type=None*, ***norm_kwargs*)

Bases: object

MRS epoch spcetrum

bjdmid = 0.0

dec = 0.0

epoch = -1

property exptime

fibermask = 0.0

fibertype = ''

filename = ''

flux = array([], dtype=float64)

flux_cont = array([], dtype=float64)

flux_err = array([], dtype=float64)

flux_norm = array([], dtype=float64)

flux_norm_dbd(***kwargs*)

return fixed flux_norm

flux_norm_err = array([], dtype=float64)

ivar = array([], dtype=float64)

ivar_norm = array([], dtype=float64)

jdbeg = 0.0

jdend = 0.0

jdltt = 0.0

jdmid = 0.0

jdmid_delta = 0.0

lmjm = 0

mask = array([], dtype=int64)

norm_kwargs = {}

normalize(*llim=0.0*, *norm_type=None*, ***norm_kwargs*)

normalize each spectrum with (optional) new settings

nspec = 0

obsid = 0

plot()

plot_cont()

plot_err()

plot_norm(*shift=0*)

plot_norm_err(*shift=0*)

plot_norm_reduce(*shift=0*)

plot_reduce()

ra = 0.0

reduce(*wave_new_list=None*, *norm_type='spline'*, *niter=3*, ***rdc_kwargs*)

Parameters

- **wave_new_list** – new wavelength grid list that will be interpolated to
- **norm_type** – type of normalization
- **niter** – number of iteration in normalization

Returns
mer – reduced epoch spectrum

Return type
MrsEpoch

rv = 0.0

seeing = 0.0

snr = []

speclist = []

specnames = []

wave = array([], dtype=float64)

wave_rv(*rv=None*)
calculate RV-corrected wavelength array

Parameters
rv (*float*) – radial velocity in km/s

class laspec.mrs.**MrsFits**(*fp=None*)
Bases: HDUList

property epoch

get_all_epochs(*including_coadd=False*, *norm_type=None*, ***norm_kwargs*)

get_one_epoch(*lmjm=84420148*, *norm_type='spline'*, ***norm_kwargs*)
get one epoch spec from fits

get_one_spec(*lmjm='COADD'*, *band='B'*)

hdunames = []

isB = []

isCoadd = []

isEpoch = []

isR = []

property ls_epoch

property ls_snr

nhdu = 0

property snr

ulmjm = []

class laspec.mrs.MrsSource(*data*, *name=''*, *norm_type=None*, ***norm_kwargs*)

Bases: ndarray

array of MrsEpoch instances,

property bjdmid

property epoch

getkwd(*k*)

static glob(*fmt*, *norm_type=None*, ***norm_kwargs*)

property jdltt

property jdmid

mes = []

name = ''

property nepoch

normalize(*norm_type=None*, ***norm_kwargs*)

static read(*fps*, *norm_type=None*, ***norm_kwargs*)

property rv

shiftplot(*shift=1.0*)

property snr

class laspec.mrs.MrsSpec(*wave=None*, *flux=None*, *ivar=None*, *mask=None*, *info={}*, *norm_type='spline'*, ***norm_kwargs*)

Bases: object

MRS spectrum

bjdmid = 0.0

dec = 0.0

exptime = 0

fibermask = 0.0

fibertype = ''

filename = ''

flux = array([], dtype=float64)

flux_cont = array([], dtype=float64)

flux_err = array([], dtype=float64)

flux_norm = array([], dtype=float64)

flux_norm_err = array([], dtype=float64)

static from_hdu(*hdu=None*, *norm_type=None*, ***norm_kwargs*)
convert MRS HDU to spec

static from_lrs(*fp_lrs*, *norm_type='spline'*, ***norm_kwargs*)
read from LRS fits file

static from_mrs(*fp_mrs*, *hduname='COADD_B'*, *norm_type=None*, ***norm_kwargs*)
read from MRS fits file

indcr = array([], dtype=float64)

info = {}

interp(*new_wave*, *rv=None*)
interpolate to a new wavelength grid

interp_norm(*new_wave*, *rv=None*)
interpolate to a new wavelength grid

interp_then_norm(*new_wave*, *rv=None*)
interpolate to a new wavelength grid

isempty = True

isnormalized = False

ivar = array([], dtype=float64)

ivar_norm = array([], dtype=float64)

jdbeg = 0

jdend = 0

jdltt = 0.0

jdmid = 0

lamplist = ''

lmjm = 0

lmjmlist = []

mask = array([], dtype=bool)

meta()

name = ''

norm_kwargs = {}

norm_type = None

normalize(*llim=0.0*, *norm_type=None*, ***norm_kwargs*)
normalize spectrum with (optional) new settings

obsid = 0

plot()

plot_cont()

plot_err()

plot_norm(*shift=0*)

plot_norm_err(*shift=0*)

ra = 0.0

reduce(*wave_new=None*, *rv=0*, *npix_cushion=50*, *cr=True*, *nsigma=(4, 8)*, *maxiter=5*, *norm_type='spline'*, *niter=2*, *flux_bounds=(0, 3)*)

Parameters

- **wave_new** – if specified, spectrum is interpolated to wave_new
- **rv** – if specified, radial velocity is corrected
- **npix_cushion** (*int*) – if speficied, cut the two ends
- **cr** – if True, remove cosmic rays using the *debad* function
- **nsigma** – sigma levels used in removing cosmic rays
- **maxiter** – max number of iterations used in removing cosmic rays
- **norm_type** – "spline" | None
- **niter** – number iterations in normalization

Return type
wave_new, flux_norm, flux_norm_err

rv = 0.0

seeing = 0.0

snr = 0

wave = array([], dtype=float64)

wave_rv(*rv=None*)

calculate RV-corrected wavelength array

Parameters
rv (*float*) – radial velocity in km/s

laspec.mrs.**datetime2jd**(*datetime='2018-10-24T05:07:06.0'*, *format='isot'*, *tz_correction=8*)

laspec.mrs.**debad**(*wave*, *fluxnorm*, *nsigma=(4, 8)*, *mfarg=21*, *gfarg=(51, 9)*, *maskconv=7*, *maxiter=3*)

Parameters

- **wave** – wavelength
- **fluxnorm** – normalized flux
- **nsigma** – lower & upper sigma levels
- **mfarg** – median filter width / pixel

- **gfarg** – Gaussian filter length & width / pixel
- **maskconv** – mask convolution –> cushion
- **maxiter** – max iteration

Return type
fluxnorm

laspec.mrs.**eval_ltt**(*ra=180.0*, *dec=40.0*, *jd=2456326.4583333*, *site=None*)
evaluate the jd

laspec.mrs.**get_kwd_safe**(*hdr*, *key*, *fallback=0.0*)

laspec.mrs.**test_meta**()

2.3.13 laspec.normalization module

class laspec.normalization.**PolySmooth**(*x*, *y*, *deg=4*, *pw=2.0*)
Bases: object

laspec.normalization.**cost_poly**(*p*, *x*, *y*, *pw=2.0*)

laspec.normalization.**normalize_spectra_block**(*wave*, *flux_block*, *norm_range*, *dwave*, *p=(1e-06, 1e-06)*, *q=0.5*, *ivar_block=None*, *eps=1e-10*, *rsv_frac=3.0*, *n_jobs=1*, *verbose=10*)
normalize multiple spectra using the same configuration This is specially designed for TheKeenan

Parameters

- **wave** (*ndarray (n_pix,)*) – wavelegnth array
- **flux_block** (*ndarray (n_obs, n_pix)*) – flux array
- **norm_range** (*tuple*) – a tuple consisting (wave_start, wave_stop)
- **dwave** (*float*) – binning width
- **p** (*tuple of 2 ps*) – smoothing parameter between 0 and 1: 0 -> LS-straight line 1 -> cubic spline interpolant
- **q** (*float in range of [0, 100]*) – percentile, between 0 and 1
- **ivar_block** (*ndarray (n_pix,) | None*) – ivar array, default is None
- **eps** (*float*) – the ivar threshold
- **rsv_frac** (*float*) – the fraction of pixels reserved in terms of std. default is 3.
- **n_jobs** (*int*) – number of processes launched by joblib
- **verbose** (*int / bool*) – verbose level

Returns

- **flux_norm_block** (*ndarray*) – normalized flux
- **flux_cont_block** (*ndarray*) – continuum flux

laspec.normalization.**normalize_spectrum**(*wave*, *flux*, *norm_range*, *dwave*, *p=(1e-06, 1e-06)*, *q=0.5*, *ivar=None*, *eps=1e-10*, *rsv_frac=1.0*)
A double smooth normalization of a spectrum

Converted from Chao Liu's normSpectrum.m Updated by Bo Zhang

Parameters

- **wave** (*ndarray (n_pix,)*) – wavelegnth array
- **flux** (*ndarray (n_pix,)*) – flux array
- **norm_range** (*tuple*) – a tuple consisting (wave_start, wave_stop)
- **dwave** (*float*) – binning width
- **p** (*tuple of 2 ps*) – smoothing parameter between 0 and 1: 0 -> LS-straight line 1 -> cubic spline interpolant
- **q** (*float in range of [0, 100]*) – percentile, between 0 and 1
- **ivar** (*ndarray (n_pix,) | None*) – ivar array, default is None
- **eps** (*float*) – the ivar threshold
- **rsv_frac** (*float*) – the fraction of pixels reserved in terms of std. default is 3.

Returns

- **flux_norm** (*ndarray*) – normalized flux
- **flux_cont** (*ndarray*) – continuum flux

Example

```
>>> flux_norm, flux_cont = normalize_spectrum(
>>>     wave, flux, (4000., 8000.), 100., p=(1E-8, 1E-7), q=0.5,
>>>     rsv_frac=2.0)
```

laspec.normalization.**normalize_spectrum_general**(*wave*, *flux*, *norm_type='spline'*, *deg=4*, *lu=(-1, 4)*, *q=0.5*, *binwidth=100.0*, *niter=2*, *pw=1.0*, *p=1e-06*)

poly / spline normalization spline –> normalize_spectrum_spline: dict(p=1e-6, q=0.5, lu=(-2, 3), binwidth=100., niter=2) poly –> normalize_spectrum_poly: (deg=4, lu=(-2, 3), q=0.5, binwidth=100., niter=2, pw=1.)

Parameters

- **wave** (*array*) – wavelength
- **flux** (*array*) – flux
- **norm_type** (*str*) – “spline” / “poly”
- **deg** (*int*) – poly deg
- **lu** (*tuple*) – defaults to (-1, 4), the data below 1 sigma and above 4 sigma will be excluded
- **q** (*float*) – percentile, default is 0.5,
- **binwidth** – bin width, detault to 100.
- **niter** – number of iterations, detaults to 3
- **pw** – power of residuals, defaults to 1, only used when norm_type==”poly”
- **p** – spline smoothness, defaults to 1e-6

laspec.normalization.**normalize_spectrum_null**(*wave*)

laspec.normalization.**normalize_spectrum_poly**(*wave*, *flux*, *deg=10*, *pw=1.0*, *lu=(-1, 4)*, *q=0.5*, *binwidth=100.0*, *niter=2*)

normalize spectrum using polynomial

laspec.normalization.**normalize_spectrum_spline**(*wave*, *flux*, *p=1e-06*, *q=0.5*, *lu=(-1, 3)*, *binwidth=30*, *niter=2*)

A double smooth normalization of a spectrum

Converted from Chao Liu's normSpectrum.m Updated by Bo Zhang

Parameters

- **wave** (*ndarray (n_pix,)*) – wavelegnth array
- **flux** (*ndarray (n_pix,)*) – flux array
- **p** (*float*) – smoothing parameter between 0 and 1: 0 -> LS-straight line 1 -> cubic spline interpolant
- **q** (*float in range of [0, 1]*) – percentile, between 0 and 1
- **lu** (*float tuple*) – the lower & upper exclusion limits
- **binwidth** (*float*) – width of each bin
- **niter** (*int*) – number of iterations

Returns

- **flux_norm** (*ndarray*) – normalized flux
- **flux_cont** (*ndarray*) – continuum flux

Example

```
>>> fnorm, fcont=normalize_spectrum_spline(
>>>     wave, flux, p=1e-6, q=0.6, binwidth=200, lu=(-1,5), niter=niter)
```

2.3.14 laspec.optimize module

class laspec.optimize.**RandomWalkMinimizer**(*fun*, *x0*, *dx*, *maxiter=20*, *args=[]*, *kwargs={}*, *optind=None*, *verbose=False*, *random='normal'*)

Bases: object

Random Walk Minimizer

static minimize(*fun*, *x0*, *dx*, *maxiter=10*, *args=None*, *kwargs={}*, *optind=None*, *verbose=False*, *info=''*, *random='normal'*)

a single

Parameters

- **fun** – objective function
- **x0** (*array like*) – initial guess of x
- **dx** – a list of different scales. e.g. []
- **maxiter** – number of max iterations
- **args** – arguments

- **kwargs** – keyword arguments
- **optind** – a subset ind of parameters, e.g., [5, 7]
- **verbose** (*bool*) – if True, print verbose
- **random** (*[uniform, normal]*) – type of random number generator
- **info** – additional info appended in msg

run(*fun=None*, *x0=None*, *dx=None*, *maxiter=None*, *args=None*, *kwargs=None*, *optind=None*, *verbose=None*, *random=None*)

laspec.optimize.**test1**()

laspec.optimize.**test2**()

laspec.optimize.**test3**()

2.3.15 laspec.qconv module

laspec.qconv.**Gaussian_kernel**(*dRV_sampling=0.1*, *dRV_Gk=2.3548200450309493*, *n_sigma_Gk=5.0*)

Gaussian kernel

Parameters

- **dRV_sampling** – the sampling rate of the input spectrum (km/s)
- **dRV_Gk** – the sampling rate of the Gaussian kernel (km/s)
- **n_sigma_Gk** – the length of the Gaussian kernel, in terms of sigma

Return type
Gaussian kernel

laspec.qconv.**Rotation_kernel**(*dRV_sampling=10*, *vsini=100*, *epsilon=0.6*, *osr_kernel=3*)

Rotation kernel

Parameters

- **dRV_sampling** – the sampling rate of the input spectrum (km/s)
- **vsini** – v*sin(i) (km/s)
- **epsilon** – the limb-darkening coefficient (0, 1)
- **osr_kernel** – the over-sampling rate of the kernel

Return type
rotation kernel

laspec.qconv.**conv_spec_Gaussian**(*wave*, *flux*, *dRV_Gk=None*, *R_hi=300000.0*, *R_lo=2000.0*, *n_sigma_Gk=5.0*, *interp=True*, *osr_ext=3.0*, *wave_new=None*)

to convolve instrumental broadening (high-R spectrum to low-R spectrum)

Parameters

- **wave** (*array*) – wavelength
- **flux** (*array*) – flux array
- **dRV_Gk** (*float*) – the FWHM of the Gaussian kernel (km/s) if None, determined by R_hi and R_lo

- **R_hi** (*float*) – higher resolution
- **R_lo** (*float*) – lower resolution
- **n_sigma_Gk** (*float*) – the gaussian kernel width in terms of sigma
- **interp** (*bool*) – if True, interpolate to log10 wavelength
- **osr_ext** – the extra oversampling rate if interp is True.
- **wave_new** – if not None, return convolved spectrum at wave_new if None, return log10 spectrum

Return type
wave_new, flux_new

laspec.qconv.**conv_spec_Rotation**(*wave*, *flux*, *vsini=100.0*, *epsilon=0.6*, *interp=True*, *osr_ext=3.0*, *wave_new=None*)

to convolve instrumental broadening (high-R spectrum to low-R spectrum)

Parameters

- **wave** (*array*) – wavelength
- **flux** (*array*) – flux array
- **vsini** (*float*) – the projected stellar rotational velocity (km/s)
- **epsilon** (*float*) – 0 to 1, the limb-darkening coefficient, default 0.6.
- **interp** (*bool*) – if True, interpolate to log10 wavelength
- **osr_ext** – the extra oversampling rate if interp is True.
- **wave_new** – if not None, return convolved spectrum at wave_new if None, return log10 spectrum

Return type
wave_new, flux_new OR Table([wave, flux])

laspec.qconv.**read_phoenix_sun**()

read PHOENIX synthetic spectrum for the Sun

laspec.qconv.**test_convolution**()

testing convolving PHOENIX synthetic spectrum for the Sun

2.3.16 laspec.read_spectrum module

class laspec.read_spectrum.**MedSpec**(**args*, ***kwargs*)

Bases: OrderedDict

for Median Resolution Spectrum

meta = None

static read(*fp*)

laspec.read_spectrum.**read_lamostms**(*fp*)

laspec.read_spectrum.**read_spectrum**(*filepath*, *filesource='auto'*)

read SDSS/LAMOST spectrum

Parameters

- **filepath** (*string*) – input file path
- **filesource** (*string*) – {'sdss_dr12' / 'lamost_dr2' / 'lamost_dr3'}

Returns

specdata – spectra as a table

Return type

astropy.table.Table

laspec.read_spectrum.**read_spectrum_elodie_r42000**(*fp*)

read spectrum from ELODIE library (R42000)

laspec.read_spectrum.**read_spectrum_phoenix_r10000**(*fp*)

read spectrum from PHOENIX R10000

laspec.read_spectrum.**reconstruct_wcs_coord_from_fits_header**(*hdr*, *dim=1*)

reconstruct wcs coordinates (e.g., wavelength array)

2.3.17 laspec.snstat module

laspec.snstat.**eval_xi_1**(*n*)

convert to MAD

laspec.snstat.**eval_xi_2**(*n*)

convert to MAD

laspec.snstat.**eval_zeta_q**(*n*)

convert to std

laspec.snstat.**eval_zeta_std**(*n*)

convert to std

2.3.18 laspec.spec module

class laspec.spec.**Spec**(**args*, ***kwargs*)

Bases: Table

extract_chunk_wave_interval(*wave_intervals=None*)

return spec chunk in a given wavelength interval

norm_spec_median()

norm_spec_pixel(*norm_wave*)

laspec.spec.**break_spectra_into_chunks**(*spec_list*, *ranges=None*, *amp=None*)

laspec.spec.**break_spectrum_into_chunks**(*spec*, *ranges=None*, *amp=None*)

laspec.spec.**norm_spec_chunk_median**(*spec_chunks*)

laspec.spec.**norm_spec_median**(*spec*)

laspec.spec.**norm_spec_pixel**(*spec*, *norm_wave*)

laspec.spec.**spec_quick_init**(*wave*, *flux*)

Parameters

- **wave** (*numpy.ndarray*) – wavelength array
- **flux** (*numpy.ndarray*) – flux array

Returns
spec_ – Spec, at least contains 'wave' and 'flux' columns.

Return type
bopy.spec.Spec

laspec.spec.**wave2ranges**(*wave*, *wave_intervals=None*)
convert wavelength intervals to (pixel) ranges

2.3.19 laspec.stilts module

python wrapper of stilts

2.3.20 laspec.wavelength module

This module implements the conversion of wavelengths between vacuum and air Reference: Donald Morton (2000, ApJ. Suppl., 130, 403) VALD3 link: http://www.astro.uu.se/valdwiki/Air-to-vacuum%20conversion

laspec.wavelength.**air2vac**(*wave_air*)

Parameters
wave_air – wavelength (A) in air

Returns
wavelength (A) in vacuum

Return type
wave_vac

laspec.wavelength.**mdwave**(*wave*)
median delta wavelength

laspec.wavelength.**vac2air**(*wave_vac*)

Parameters
wave_vac – wavelength (A) in vacuum

Returns
wavelength (A) in air

Return type
wave_air

laspec.wavelength.**wave_log10**(*wave*, *osr_ext=1.0*, *dwave=None*)
generate log10 wavelength array given wave array

Parameters

- **wave** – old wavelength array
- **osr_ext** – extra over-sampling rate
- **dwave** – delta wavelength. if not specified, use median(dwave).

Example

```
>>> import numpy as np
>>> wave = np.arange(3000, 5000)
>>> wave_new = wave_log10(wave, osr_ext=3)
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

FOUR

HISTORY

last modified : 2021.01.24

I finally know how the *sphinx* and *readthedoc* work…

PYTHON MODULE INDEX

l

INDEX

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X