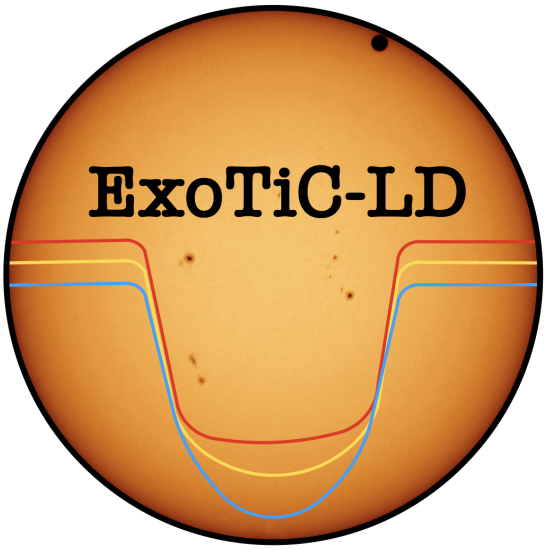
ExoTiC-LD

Release v3.0.0

David Grant and Hannah Wakeford

Apr 28, 2023

CONTENTS



ExoTiC-LD is a python package for calculating stellar limb-darkening coefficients for specific instruments, stars, and wavelength ranges.

Stellar intensity (I) is modelled, as a function of radial position on the stellar disc (μ), from pre-computed grids of models spanning a range of metallicity, effective temperature, and surface gravity. These intensities are combined with an instrument's throughput and integrated over a specified wavelength range, resulting in a one-dimensional profile, $I(\mu)$. Limb-darkening coefficients are then calculated by fitting one of the various functional forms, as outlined in Claret (2000) and Sing (2010), to the modelled $I(\mu)$ profile.

CHAPTER

ONE

INSTALLATION

There are two steps for installing ExoTiC-LD.

1) install the package with pip:

```
pip install exotic-ld
```

2) Download the stellar models and instrument throughputs from this zenodo link. Make sure that you download the version that matches the exotic-ld package version.

   The downloaded and unzipped directory structure should have the following layout:

```
exotic_ld_data/
├── Sensitivity_files/
├── kurucz/
├── stagger/
├── mps1/
├── mps2/
```

   Some of the stellar grids are very large in size, and so you can choose to download only the stellar grids that you require. The instrument throughputs directory, "Sensitivity_files", must always be included.

   You can place the downloaded directory anywhere on your machine, you'll just need to pass the path, "path/to/exotic_ld_data", as an input when running the code.

CHAPTER

TWO

QUICK START

After installing the code and downloading the accompanying data (see *installation*) you are ready to calculate limb-darkening coefficients. Below we demonstrate a minimal example.

First, we define the stellar parameters and which stellar models to use in the computation.

```
# Path to the downloaded data.
ld_data_path = 'path/to/exotic_ld_data'

# Stellar models grid.
ld_model = 'mps1'

# Metallicty [dex].
M_H = 0.01

# Effective temperature [K].
Teff = 5512

# Surface gravity [dex].
logg = 4.47
```

Next, import the StellarLimbDarkening class and set the stellar parameters.

```
from exotic_ld import StellarLimbDarkening


sld = StellarLimbDarkening(M_H, Teff, logg, ld_model, ld_data_path)
```

Now you can compute the stellar limb-darkening coefficients for the limb-darkening law of your choice. You simply have to specify the instrument mode and the wavelength range you require.

```
# Start and end of wavelength interval [angstroms].
wavelength_range = [20000., 30000.]

# Instrument mode.
mode = 'JWST_NIRSpec_prism'

u1, u2 = sld.compute_quadratic_ld_coeffs(wavelength_range, mode)
```

The limb-darkening laws available are linear, quadratic, square root, 3-parameter and 4-parameter non-linear. The available stellar grids are listed in *supported stellar grids*, and the available instrument modes are listed in *supported instruments*.

CHAPTER

THREE

SUPPORTED STELLAR GRIDS

Here we list each of the available stellar grids.

3.1 Kurucz

ld_model = 'kurucz' or '1D'

The Kurucz (1993) grid (CD-ROM No. 13) span a range of metallicities from -5.0 to 1.0, effective temperatures from 3500 to 6500 k, and logg from 4.0 to 5.0. There are 741 models in total, each evaluated at 1221 wavelengths and 17 radial positions on the stellar disc.

3.2 Stagger

ld_model = 'stagger' or '3D'

The Stagger grid uses the Magic (2015) 3D stellar models and spans a range of metallicities from -3.0 to 0.0, effective temperatures from 4000 to 7000 k, and logg from 1.5 to 5.0. Note that this grid has non-uniform coverage, see figure 1 of Magic (2013). There are 99 models in total, each evaluated at 105767 wavelengths and 10 radial positions on the stellar disc.

3.3 MPS-ATLAS-1

ld_model = 'mps1'

The MPS-ATLAS (set 1) grid uses the Kostogryz (2022) stellar models and spans a range of metallicities from -5.0 to 1.5, effective temperatures from 3500 to 9000 k, and logg from 3.0 to 5.0. This grid has extensive parameter space coverage, showcasing 34160 models in total, each evaluated at 1221 wavelengths and 24 radial positions on the stellar disc. Set 1 uses the Grevesse & Saul 1998 abundances with a constant chemical mixing length of 1.25.

3.4 MPS-ATLAS-2

ld_model = 'mps2'

The MPS-ATLAS (set 2) grid uses the Kostogryz (2022) stellar models and spans a range of metallicities from -5.0 to 1.5, effective temperatures from 3500 to 9000 k, and logg from 3.0 to 5.0. This grid has extensive parameter space coverage, showcasing 34160 models in total, each evaluated at 1221 wavelengths and 24 radial positions on the stellar disc. Set 2 uses the the Asplund (2009) abundances with a chemical mixing length dependent on stellar parameters from Viani (2018).

CHAPTER

FOUR

SUPPORTED INSTRUMENT MODES

Here we list each of the available instrument modes.

4.1 Spectroscopic

Hubble STIS gratings : 'HST_STIS_G430L', 'HST_STIS_G750L'

Hubble WFC3 grisms : 'HST_WFC3_G280p1', 'HST_WFC3_G280n1', 'HST_WFC3_G102', 'HST_WFC3_G141'

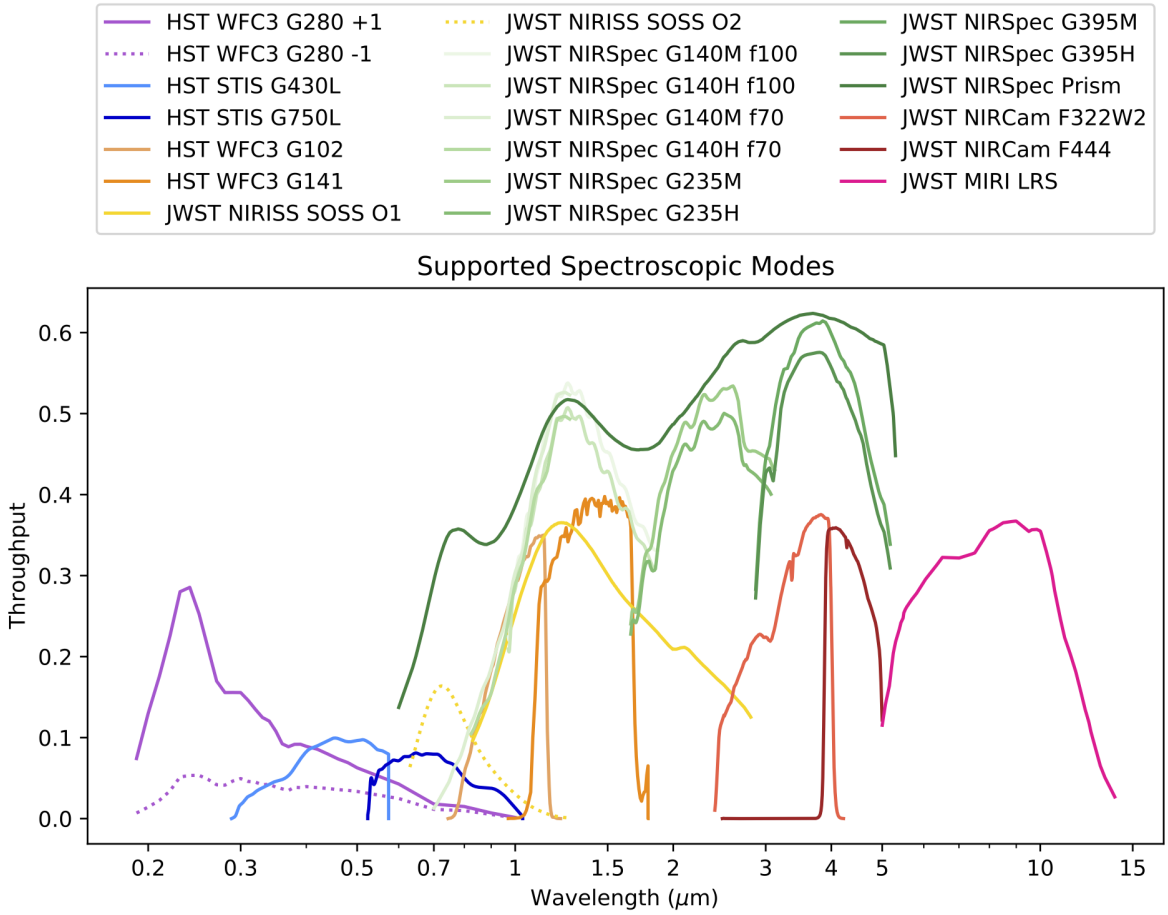
JWST NIRSpec : 'JWST_NIRSpec_Prism', 'JWST_NIRSpec_G395H', 'JWST_NIRSpec_G395M', 'JWST_NIRSpec_G235H', 'JWST_NIRSpec_G235M', 'JWST_NIRSpec_G140H', 'JWST_NIRSpec_G140M-f100', 'JWST_NIRSpec_G140H-f070', 'JWST_NIRSpec_G140M-f070'

JWST NIRISS : 'JWST_NIRISS_SOSSo1', 'JWST_NIRISS_SOSSo2'

JWST NIRCam : 'JWST_NIRCam_F322W2', 'JWST_NIRCam_F444'

JWST MIRI : 'JWST_MIRI_LRS'

Note for the WFC3 G280 grism the p1 and n1 signify the positive 1st order spectrum and negative 1st order spectrum for the UVIS grism.

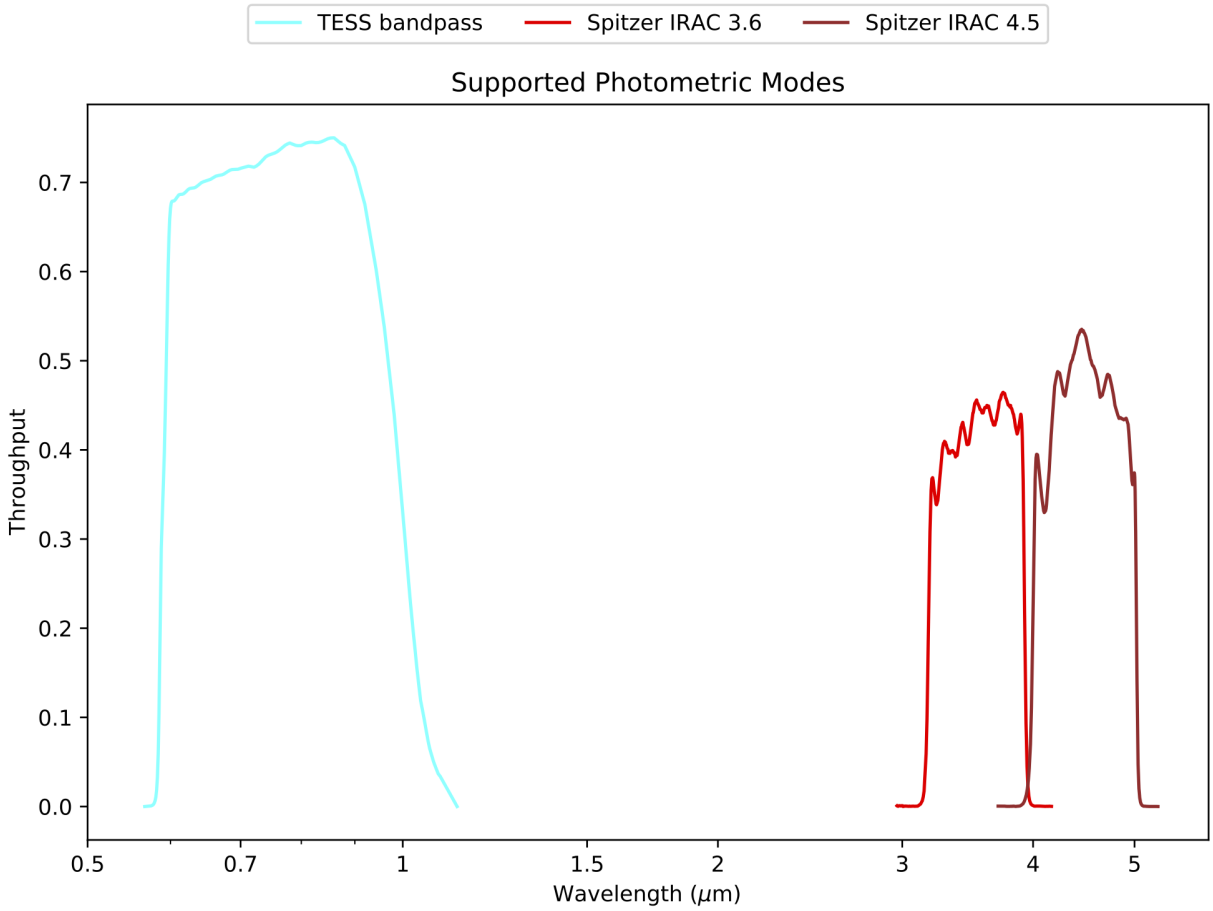

HST WFC3 G280 +1
HST WFC3 G280 -1
HST STIS G430L
HST STIS G750L
HST WFC3 G102
HST WFC3 G141
JWST NIRISS SOSS O1
JWST NIRISS SOSS O2
JWST NIRSpec G140M f100
JWST NIRSpec G140H f100
JWST NIRSpec G140M f70
JWST NIRSpec G140H f70
JWST NIRSpec G235M
JWST NIRSpec G235H
JWST NIRSpec G395M
JWST NIRSpec G395H
JWST NIRSpec Prism
JWST NIRCam F322W2
JWST NIRCam F444
JWST MIRI LRS
Supported Spectroscopic Modes
Throughput
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.2
0.3
0.5
0.7
1
1.5
2
3
4
5
10
15
Wavelength (μm)

4.2 Photometric

Spitzer IRAC : 'Spitzer_IRAC_Ch1', 'Spitzer_IRAC_Ch2'

TESS : 'TESS'

Note for Spitzer Ch1 is the 3.6 micron channel and Ch2 is the 4.5 microns channel.



4.3 Custom

Custom profile : 'custom'

See the custom throughput *tutorial*.

CHAPTER

FIVE

TUTORIALS

5.1 Calculate limb-darkening coefficients

In this tutorial we walk through various options for calculating stellar limb-darkening coefficients. Let us start by importing the required packages and setting the path to the downloaded stellar data. In this tutorial we get the path from an enviroment variable, but you can simply set the path as a string.

[1]:
```
import os
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt

from exotic_ld import StellarLimbDarkening


# "path/to/exotic_ld_data".
ld_data_path = os.environ["exotic_ld_data"]
```

First, instantiate the StellarLimbDarkening class. Here you must specify the stellar parameters and the stellar models. We have also set verbose=True to get some extra information about what is going on under the hood.

[2]:
```
sld = StellarLimbDarkening(M_H=0.01, Teff=5512, logg=4.47,
                           ld_model="mps1",
                           ld_data_path=ld_data_path,
                           interpolate_type="nearest",
                           verbose=True)
```

```
Input stellar parameters are M_H=0.01, Teff=5512, logg=4.47.
Loading stellar model from mps1 grid.
Matched nearest with M_H=0.0, Teff=5500, logg=4.5.
Stellar model loaded.
```

During instantiation stellar models are loaded based on your input stellar parameters. By default, the nearest matching model is located in the specified stellar grid. However, you may set interpolate_type="trilinear", and the stellar models will be linearly interpolated in each of the three parameter dimensions.

You can inspect the loaded stellar model using the following attributes.

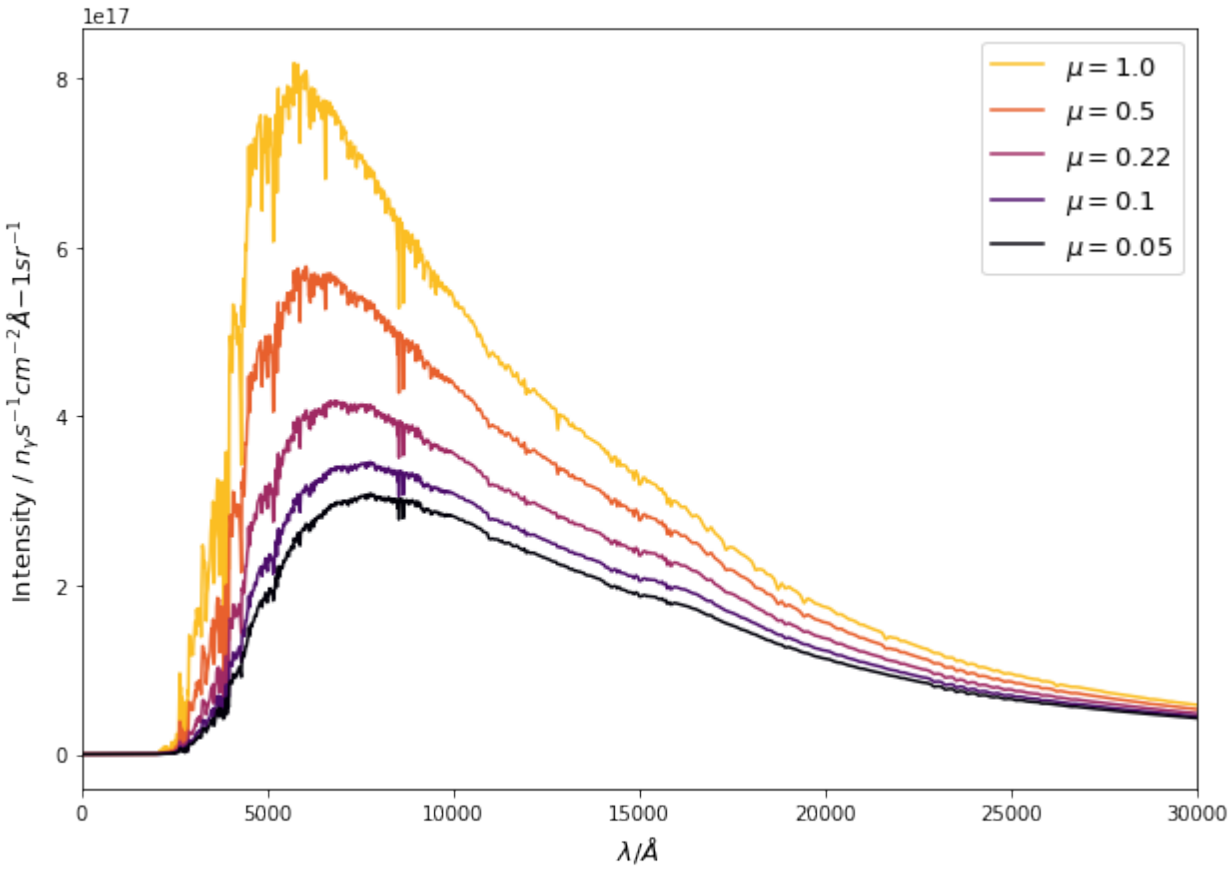
[3]:
```
print(sld.stellar_wavelengths.shape)
print(sld.mus.shape)
print(sld.stellar_intensities.shape)
```

(continues on next page)

(continued from previous page)

```
plt.figure(figsize=(10, 7))
for mu_idx in np.arange(0, sld.mus.shape[0], 5):
    plt.plot(sld.stellar_wavelengths, sld.stellar_intensities[:, mu_idx],
             color=cm.inferno(0.85 - mu_idx/sld.mus.shape[0]), label="$\mu={}$".
↪format(sld.mus[mu_idx]))
plt.xlabel("$\lambda / \AA$", fontsize=13)
plt.ylabel("Intensity / $n_{\gamma} s^{-1} cm^{-2} \AA{-1} sr^{-1}$", fontsize=13)
plt.xlim(0, 3e4)
plt.legend(loc="upper right", fontsize=13)
plt.show()
```

```
(1221,)
(24,)
(1221, 24)
```



With the stellar model loaded, you can compute the stellar limb-darkening coefficients for the limb-darkening law of your choice. You simply have to specify the instrument mode and the wavelength range you require. You can also limit the range of μ values that are included in the fit.

[4]:
```
us = sld.compute_4_parameter_non_linear_ld_coeffs(wavelength_range=[20000., 30000.],
                                                  mode="JWST_NIRSpec_prism",
                                                  mu_min=0.1)
```

```
Loading instrument mode=JWST_NIRSpec_prism with wavelength range 6000.0-53000.0 A.
Integrating I(mu) for wavelength limits of 20000.0-30000.0 A.
```

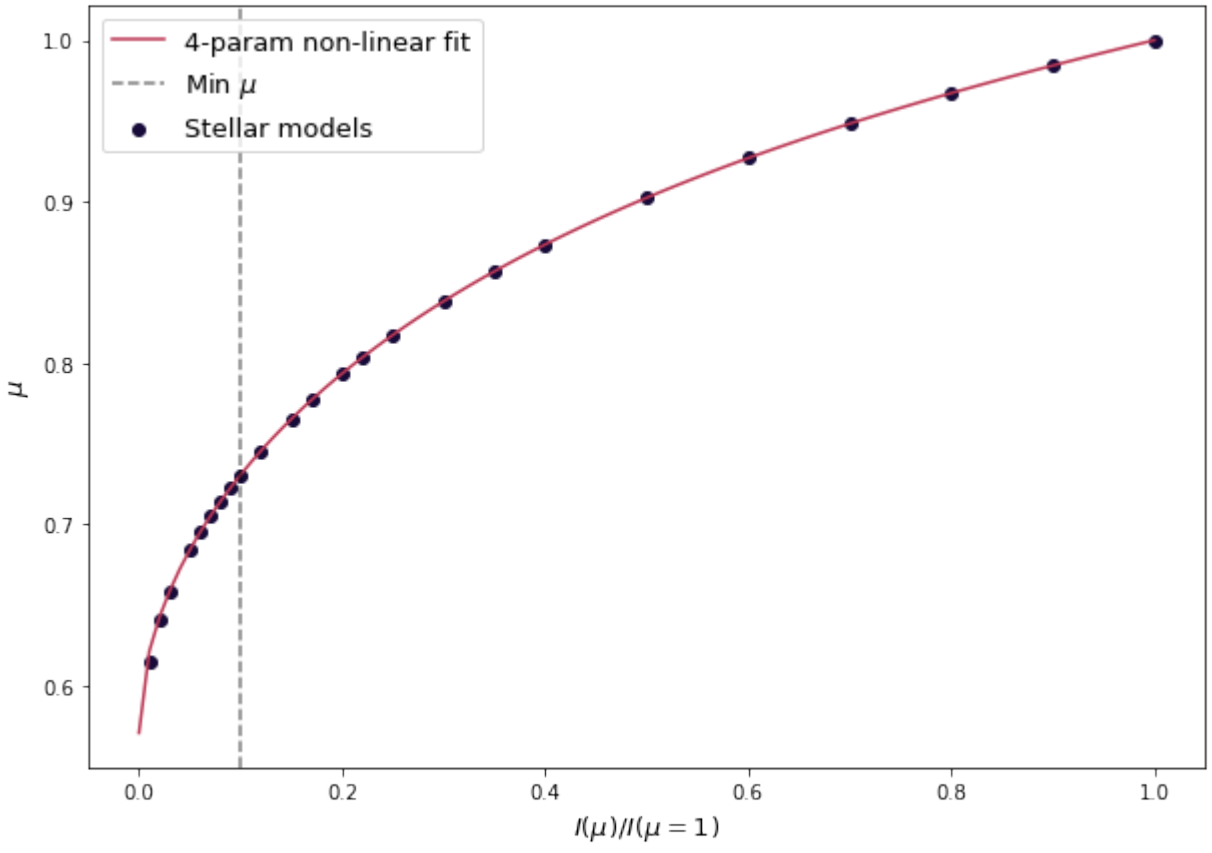
(continues on next page)

(continued from previous page)

```
Integral done for I(mu).
Fitting limb-darkening law to 16 I(mu) data points where 0.1 <= mu <= 1.
Fit done, resulting coefficients are [ 0.49168688  0.13344898 -0.32763496  0.13151495].
```

Finally, you can inspect the fitted limb-darkening coefficients. To do this, you can find the integrated intensity profile in the attribute sld.I_mu, and the limb-darkening laws are all defined in the submodule, exotic_ld.ld_laws.

[5]:
```
from exotic_ld.ld_laws import nonlinear_4param_ld_law


plt.figure(figsize=(10, 7))
plt.scatter(sld.mus, sld.I_mu, color=cm.inferno(0.1), label="Stellar models")
check_mus = np.linspace(0., 1., 100)
plt.plot(check_mus, nonlinear_4param_ld_law(check_mus, *us),
         color=cm.inferno(0.5), label="4-param non-linear fit")
plt.axvline(0.1, ls="--", color=cm.inferno(0.), alpha=0.5, label="Min $\mu$")
plt.xlabel("$I(\mu) / I(\mu=1)$", fontsize=13)
plt.ylabel("$\mu$", fontsize=13)
plt.legend(loc="upper left", fontsize=13)
plt.show()
```



The returned tuple of limb-darkening coefficients, us, is now ready to be used in your analysis.

5.2 Calculate probabilistic coefficients

In this tutorial we take a look at how to generate distributions of stellar limb-darkening coefficients, most likley to be used as priors in your data analysis.

[1]:
```
import os
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt

from exotic_ld import StellarLimbDarkening


# "path/to/exotic_ld_data".
ld_data_path = os.environ["exotic_ld_data"]
```

First, instantiate the StellarLimbDarkening class.

[2]:
```
sld = StellarLimbDarkening(M_H=0.01, Teff=5512, logg=4.47,
                           ld_model="mps1",
                           ld_data_path=ld_data_path)
```

Next, compute the stellar limb-darkening coefficients for a specified limb-darkening law, instrument mode, and wavelength range. Note that we have set return_sigmas=True. This options returns the standard deviation of the uncertainty distribution for each limb-darkening coefficient from the fitting algorithm.

[3]:
```
us, us_sigmas = sld.compute_4_parameter_non_linear_ld_coeffs(wavelength_range=[20000.,␣
↪30000.],
                                                             mode="JWST_NIRSpec_prism",
                                                             return_sigmas=True)

for u_idx, u, u_sigma in zip(range(1, 5), us, us_sigmas):
    print("u_{}={}+-{}".format(u_idx, round(u, 4), round(u_sigma, 4)))
```

```
u_1=0.4885+-0.0094
u_2=0.1276+-0.0231
u_3=-0.3138+-0.0241
u_4=0.1255+-0.009
```

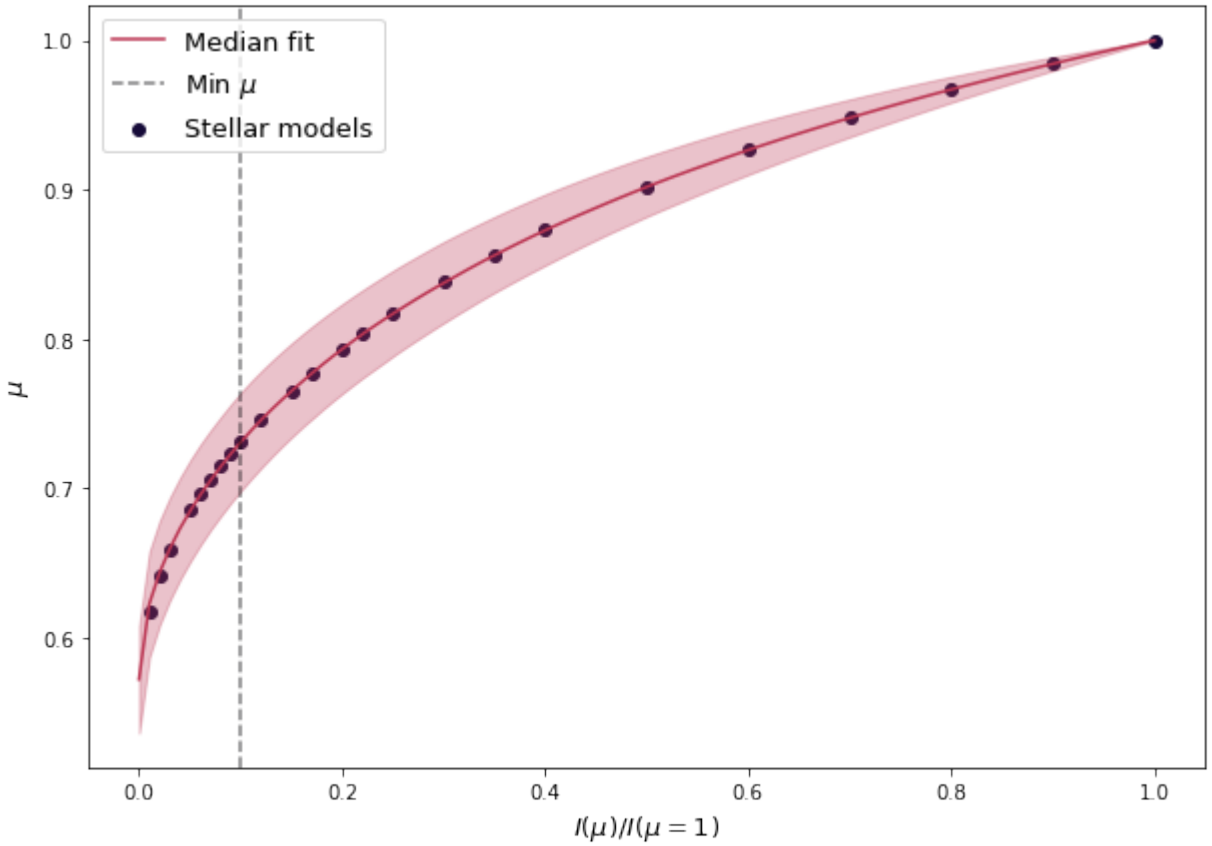
You can visualise this distribution as follows

[4]:
```
from exotic_ld.ld_laws import nonlinear_4param_ld_law


# Sample ld-coeff distributions.
n_mc_samples = 1000
check_mus = np.linspace(0., 1., 100)
us_dist = np.random.normal(loc=us, scale=us_sigmas, size=(n_mc_samples, len(us)))
I_mu = nonlinear_4param_ld_law(check_mus[..., np.newaxis],
                               us_dist[:, 0], us_dist[:, 1],
                               us_dist[:, 2], us_dist[:, 3])

# Get 16th, 50th, 84th percentiles.
I_mu_16, I_mu_50, I_mu_84 = np.percentile(I_mu, [16., 50., 84.], axis=1)
```

(continues on next page)

(continued from previous page)

```
plt.figure(figsize=(10, 7))
plt.scatter(sld.mus, sld.I_mu, color=cm.inferno(0.1), label="Stellar models")
plt.plot(check_mus, I_mu_50, color=cm.inferno(0.5), label="Median fit")
plt.fill_between(check_mus, I_mu_16, I_mu_84, color=cm.inferno(0.5), alpha=0.3)
plt.axvline(0.1, ls="--", color=cm.inferno(0.), alpha=0.5, label="Min $\mu$")
plt.xlabel("$I(\mu) / I(\mu=1)$", fontsize=13)
plt.ylabel("$\mu$", fontsize=13)
plt.legend(loc="upper left", fontsize=13)
plt.show()
```



5.3 Use a custom throughput

In this tutorial we take a look at how to use a custom throughput if you cannot find the mode you are looking for in supported instruments.

Let us mock up some throughput data.

[1]:
```
import numpy as np


wvs = np.linspace(10000., 20000., 100)
throughput = np.exp(-0.5 * ((wvs - 15000.) / 5000.)**2)
```

With this data in hand, you can run the code as follows.

[2]:
```
import os
from exotic_ld import StellarLimbDarkening


sld = StellarLimbDarkening(M_H=0.01, Teff=5512, logg=4.47,
                           ld_model="mps1",
                           ld_data_path=os.environ["exotic_ld_data"])
cs = sld.compute_4_parameter_non_linear_ld_coeffs(wavelength_range=[13000., 17000.],
                                                  mode="custom",
                                                  custom_wavelengths=wvs,
                                                  custom_throughput=throughput)
print(cs)
```

```
(0.6017116782763823, 0.21480303681178103, -0.41422249107058384, 0.16119402196903784)
```

5.4 Use a custom stellar model

In this tutorial we take a look at how to use a custom stellar model.

Let us mock up some stellar data.

[1]:
```
import numpy as np



def generate_synthetic_stellar_models(n_wvs=1000, n_mus=20):
    # Generate I(lambda, mu).
    wvs = np.linspace(0.01e-6, 50e-6, n_wvs)
    mus = np.linspace(1., 0.01, n_mus)
    temps = np.linspace(5000., 4500., n_mus)
    stellar_intensity = []
    for mu, temp in zip(mus, temps):
        stellar_intensity.append(plancks_law(wvs, temp))
    return wvs * 1.e10, mus, np.array(stellar_intensity).T

def plancks_law(wav, temp):
    a = 2.0 * 6.62607004e-34 * 2.99792458e8**2
    b = 6.62607004e-34 * 2.99792458e8 / (wav * 1.38064852e-23 * temp)
    intensity = a / (wav**5 * (np.exp(b) - 1.0))
    return intensity


s_wvs, s_mus, stellar_intensity = generate_synthetic_stellar_models()

print(s_wvs.shape)
print(s_mus.shape)
print(stellar_intensity.shape)
```

```
(1000,)
(20,)
(1000, 20)
```

With this data in hand, you can run the code as follows.

[2]:
```
import os
from exotic_ld import StellarLimbDarkening


sld = StellarLimbDarkening(ld_data_path=os.environ["exotic_ld_data"],
                           ld_model="custom",
                           custom_wavelengths=s_wvs,
                           custom_mus=s_mus,
                           custom_stellar_model=stellar_intensity)
cs = sld.compute_4_parameter_non_linear_ld_coeffs(wavelength_range=[20000., 30000.],
                                                  mode="JWST_NIRSpec_prism")
print(cs)
```

```
(0.0005946623383321141, 0.1675823469013117, 0.00268452272138109, 0.0012526197251732931)
```

5.5 View the stellar spectrum

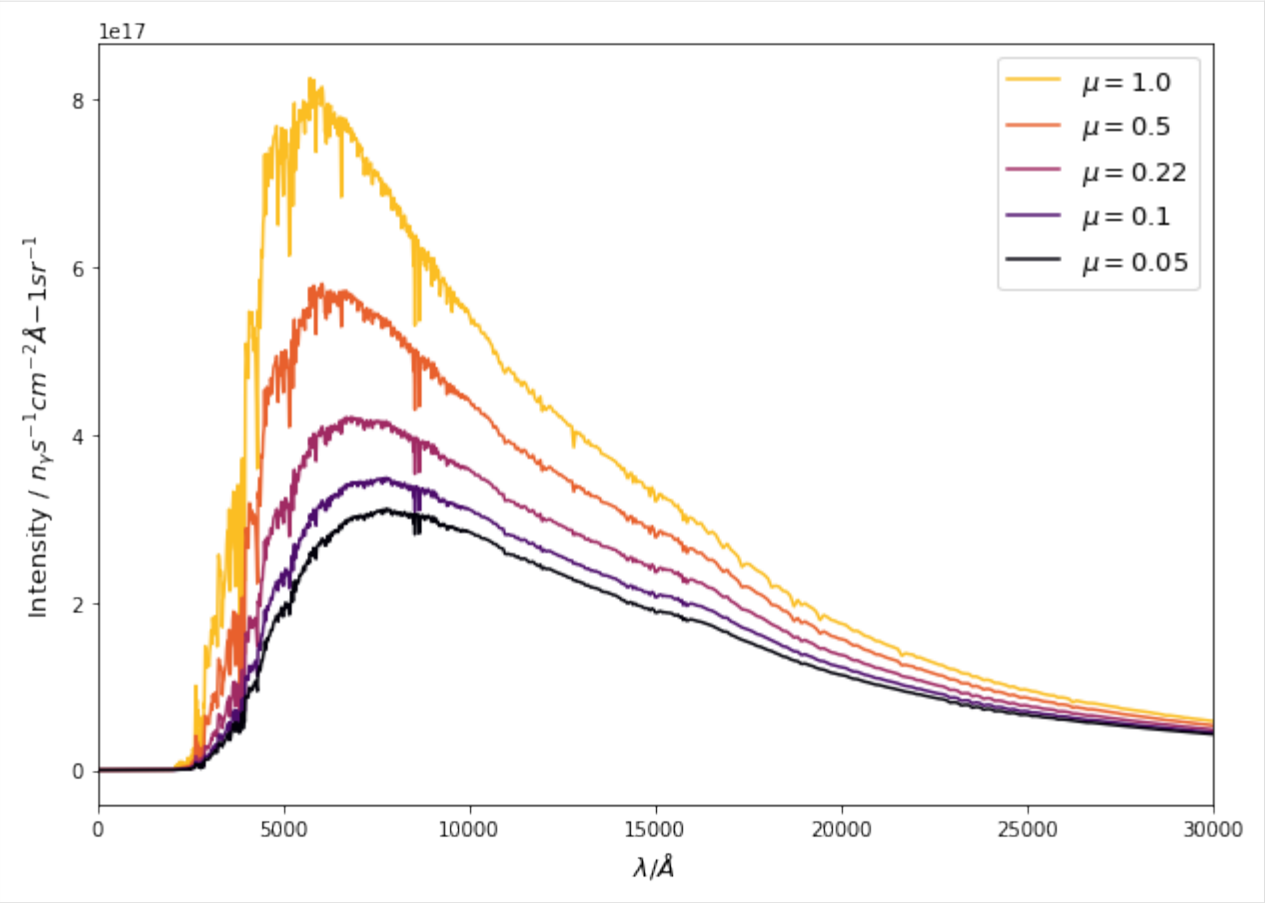
In this tutorial we take a look at how to inspect the stellar spectrum used in the limb-darkening calculation.

Let us instantiate the StellarLimbDarkening class for some specified parameters.

[1]:
```
import os
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt

from exotic_ld import StellarLimbDarkening


# "path/to/exotic_ld_data".
ld_data_path = os.environ["exotic_ld_data"]

sld = StellarLimbDarkening(M_H=0.01, Teff=5512, logg=4.47,
                           ld_model="mps1",
                           ld_data_path=ld_data_path,
                           interpolate_type="trilinear")
```

You can easily inspect the stellar intensity as a function of radial position on the stellar disc.

[2]:
```
plt.figure(figsize=(10, 7))
for mu_idx in np.arange(0, sld.mus.shape[0], 5):
    plt.plot(sld.stellar_wavelengths, sld.stellar_intensities[:, mu_idx],
             color=cm.inferno(0.85 - mu_idx/sld.mus.shape[0]), label="$\mu={}$".
↪format(sld.mus[mu_idx]))
plt.xlabel("$\lambda / \AA$", fontsize=13)
plt.ylabel("Intensity / $n_{\gamma} s^{-1} cm^{-2} \AA{-1} sr^{-1}$", fontsize=13)
plt.xlim(0, 3e4)
plt.legend(loc="upper right", fontsize=13)
plt.show()
```



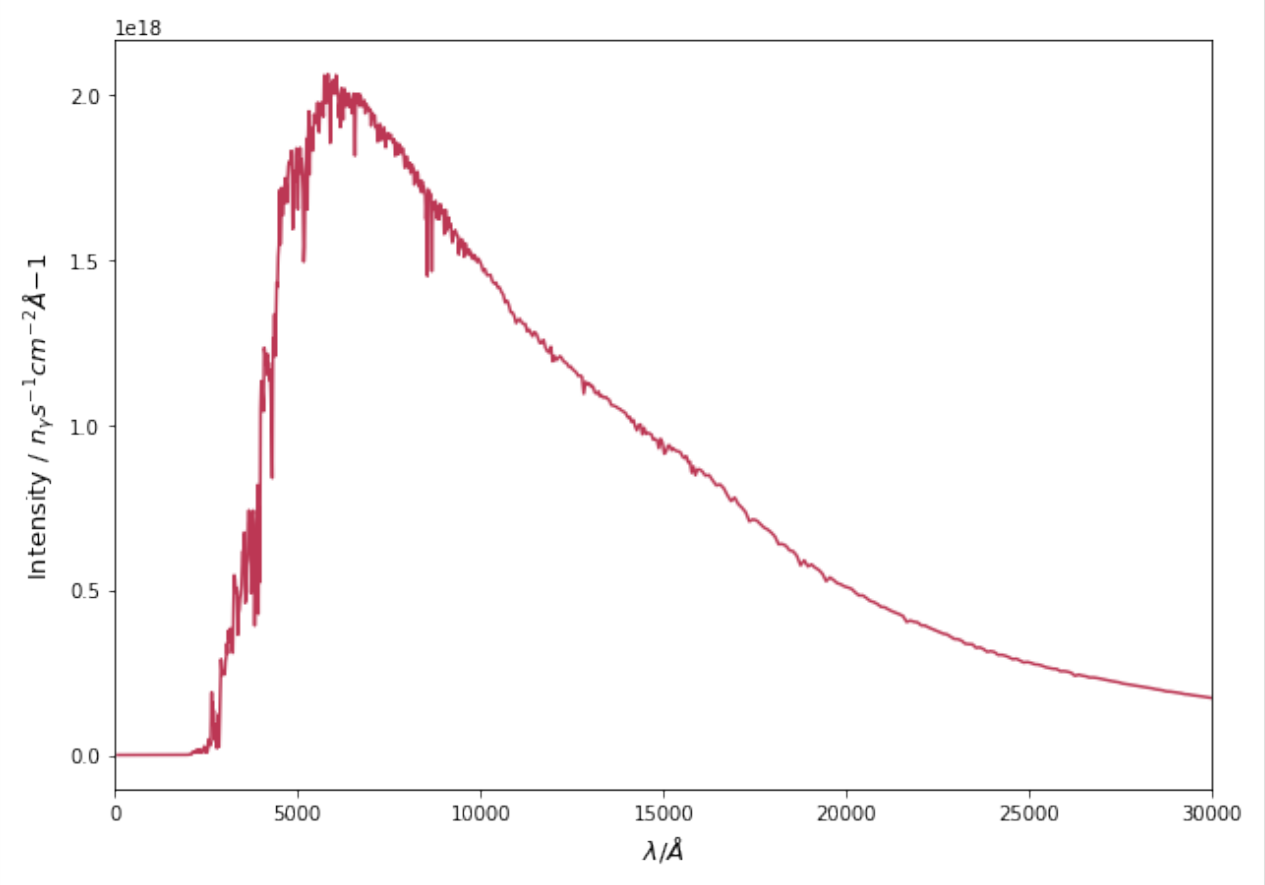
Or, you can inspect the total spectrum by integrating across the stellar disc.

[3]:
```
from scipy.interpolate import interp1d
from scipy.special import roots_legendre


rs = (1 - sld.mus**2)**0.5
roots, weights = roots_legendre(500)
a, b = (0., 1.)
t = (b - a) / 2 * roots + (a + b) / 2

spectrum = []
for wv_idx in range(sld.stellar_wavelengths.shape[0]):

    i_interp_func = interp1d(
        rs, sld.stellar_intensities[wv_idx, :], kind='linear',
        bounds_error=False, fill_value=0.)

    def integrand(_r):
        return i_interp_func(_r) * _r * 2. * np.pi

    spectrum.append((b - a) / 2. * integrand(t).dot(weights))

plt.figure(figsize=(10, 7))
```

(continues on next page)

(continued from previous page)

```
plt.plot(sld.stellar_wavelengths, spectrum, color=cm.inferno(0.5))
plt.xlabel("$\lambda / \AA$", fontsize=13)
plt.ylabel("Intensity / $n_{\gamma} s^{-1} cm^{-2} \AA{-1}$", fontsize=13)
plt.xlim(0, 3e4)
plt.show()
```



CHAPTER

SIX

API

6.1 Primary Interface

`StellarLimbDarkening`([M_H, Teff, logg, ...])	Stellar limb darkening class.

6.2 Subpackages

6.2.1 Limb-darkening laws

linear_ld_law(mu, u1)	Linear limb darkening law.
quadratic_ld_law(mu, u1, u2)	Quadratic limb darkening law.
squareroot_ld_law(mu, u1, u2)	Square root limb darkening law.
nonlinear_3param_ld_law(mu, u1, u2, u3)	Non-linear 3-parameter limb darkening law.
nonlinear_4param_ld_law(mu, u1, u2, u3, u4)	Non-linear 4-parameter limb darkening law.

exotic_ld.ld_laws.linear_ld_law

exotic_ld.ld_laws.**linear_ld_law**(*mu*, *u1*)

Linear limb darkening law.

exotic_ld.ld_laws.quadratic_ld_law

exotic_ld.ld_laws.**quadratic_ld_law**(*mu*, *u1*, *u2*)

Quadratic limb darkening law.

exotic_ld.ld_laws.squareroot_ld_law

exotic_ld.ld_laws.**squareroot_ld_law**(*mu*, *u1*, *u2*)

Square root limb darkening law.

exotic_ld.ld_laws.nonlinear_3param_ld_law

exotic_ld.ld_laws.**nonlinear_3param_ld_law**(*mu*, *u1*, *u2*, *u3*)

Non-linear 3-parameter limb darkening law.

exotic_ld.ld_laws.nonlinear_4param_ld_law

exotic_ld.ld_laws.**nonlinear_4param_ld_law**(*mu*, *u1*, *u2*, *u3*, *u4*)

Non-linear 4-parameter limb darkening law.

CHAPTER

SEVEN

CITATION

If you make use of ExoTiC-LD in your research, please cite Wakeford and Grant 2022:

```
@software{hannah_wakeford_2022_6809899,
  author       = {Hannah Wakeford and
                  David Grant},
  title        = {Exo-TiC/ExoTiC-LD: ExoTiC-LD v2.1 Zenodo Release},
  month        = jul,
  year         = 2022,
  publisher    = {Zenodo},
  version      = {v2.1.0},
  doi          = {10.5281/zenodo.6809899},
  url          = {https://doi.org/10.5281/zenodo.6809899}
}
```

Stellar models

Depending on which stellar models you calculate your limb-darkening coefficients, you may also consider citing one of the following:

```
@article{kurucz1993atlas9,
  title={ATLAS9 Stellar Atmosphere Programs and 2km/s grid},
  author={Kurucz, R-L\_},
  journal={Kurucz CD-Rom},
  volume={13},
  year={1993},
  publisher={Smithsonian Astrophysical Observatory}
}
```

```
@article{magic2015stagger,
  title={The Stagger-grid: A grid of 3D stellar atmosphere models-IV. Limb darkening␣
↪coefficients},
  author={Magic, Zazralt and Chiavassa, Andrea and Collet, Remo and Asplund, Martin},
  journal={Astronomy \& Astrophysics},
  volume={573},
  pages={A90},
  year={2015},
  publisher={EDP Sciences}
}
```

```
@article{kostogryz2022stellar,
  title={Stellar limb darkening. I. A new MPS-ATLAS library for Kepler, TESS, CHEOPS,␣
```

(continues on next page)

(continued from previous page)

```
↪and PLATO passbands},
  author={Kostogryz, NM and Witzke, V and Shapiro, AI and Solanki, SK and Maxted, PFL␣
↪and Kurucz, RL and Gizon, L},
  journal={arXiv preprint arXiv:2206.06641},
  year={2022}
}
```

CHAPTER

EIGHT

ACKNOWLEDGEMENTS

The present version of ExoTiC-LD is built by David Grant and Hannah Wakeford.

The original IDL code was translated into python by Matthew Hill with improvements by Iva Laginja. The git history associated with the original implementation can be found in the ExoTiC-ISM package from which this is a spin-off repository. We also thank Natasha Batalha for providing the JWST throughput information from their PandExo package and to Lili Alderson for reviewing and testing.

If you make use of ExoTiC-LD in your research, see the *citation page* for info on how to cite this package and the underlying stellar models.

You can find other software from the Exoplanet Timeseries Characterisation (ExoTiC) ecosystem over on GitHub.

PYTHON MODULE INDEX

e

INDEX