
ExoTiC-LD

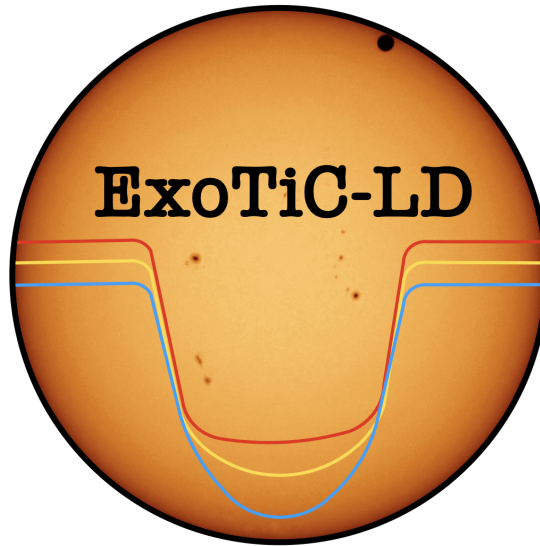
Release v3.0.0

David Grant and Hannah Wakeford

Mar 22, 2024

CONTENTS

1	Installation	3
2	Quick start	5
3	Supported stellar grids	7
3.1	Phoenix	7
3.2	Kurucz	7
3.3	Stagger	7
3.4	MPS-ATLAS-1	7
3.5	MPS-ATLAS-2	8
4	Supported instrument modes	9
4.1	Spectroscopic	9
4.2	Photometric	11
4.3	Custom	11
5	Tutorials	13
5.1	Calculate limb-darkening coefficients	13
5.2	Calculate probabilistic coefficients	16
5.3	Use a custom throughput	18
5.4	Use a custom stellar model	19
5.5	View the stellar spectrum	20
6	API	23
6.1	Primary Interface	23
6.2	Subpackages	23
7	Citation	25
8	Acknowledgements	27
	Python Module Index	29
	Index	31



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.

INSTALLATION

There is only one step (as of v3.2.0) for installing ExoTiC-LD.

- 1) Using **python>=3.8**, install the package with pip:

```
pip install exotic-ld
```

And now you are ready to go. Stellar models and instrument throughput data are selected and downloaded automatically at runtime, *or custom files can be input by the user*. These data are saved locally and so are only downloaded once, speeding up subsequent runs. Head straight to the [quick start](#) page to begin computing limb-darkening coefficients .

Backwards compatibility/optional step:

Prior to v3.2.0, the stellar models and instrument data had to be downloaded manually. If you wish to not rely on an internet connection or you already have these data, then you can proceed with step 2, which remains supported. However this is no longer necessary, you can skip this step and start computing limb-darkening coefficients right away.

- 2) Download the stellar models and instrument throughputs from [this zenodo link](#).

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.

QUICK START

After installing the code (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 store stellar and instrument data.
ld_data_path = 'exotic_ld_data'

# Stellar models grid.
ld_model = 'mps1'

# Metallicity [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)
```

Note that only the stellar and instrument data required for your calculation are automatically downloaded. These data are saved to your specified `ld_data_path`, and so on subsequent runs with the same parameters the calculation will run much faster.

The limb-darkening laws available are linear, quadratic, the Kipping reparameterisation, 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*.

SUPPORTED STELLAR GRIDS

Here we list each of the available stellar grids.

3.1 Phoenix

`ld_model = 'phoenix'`

The Phoenix grid used the [Husser \(2013\)](#) stellar models and spans a range of metallicities from -1.5 to 1.0, effective temperatures from 2300 to 15000 k, and `logg` from 0.0 to 6.0. There are 5079 models in total, each evaluated at 54500 wavelengths and 78 radial positions on the stellar disc.

3.2 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.

3.3 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.

3.4 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 & Sauval 1998 abundances with a constant chemical mixing length of 1.25.

3.5 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 $\log g$ 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).

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’

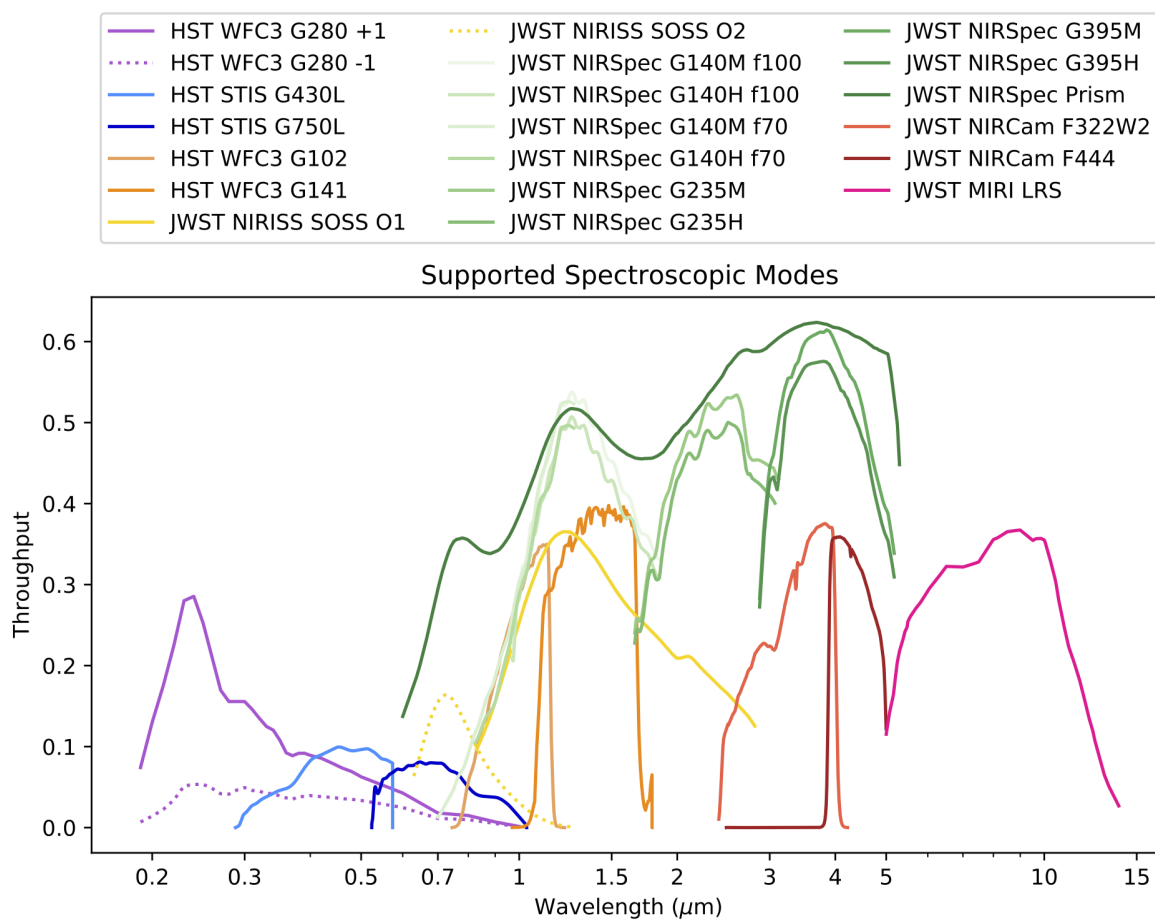
JWST NIRSpec : ‘JWST_NIRSpec_Prism’, ‘JWST_NIRSpec_G395H’, ‘JWST_NIRSpec_G395M’,
‘JWST_NIRSpec_G235H’, ‘JWST_NIRSpec_G235M’, ‘JWST_NIRSpec_G140H’,
‘JWST_NIRSpec_G140M-f100’, ‘JWST_NIRSpec_G140H-f070’,
‘JWST_NIRSpec_G140M-f070’

JWST NIRISS : ‘JWST_NIRISS_SOSSo1’, ‘JWST_NIRISS_SOSSo2’

JWST NIRCам : ‘JWST_NIRCам_F322W2’, ‘JWST_NIRCам_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.

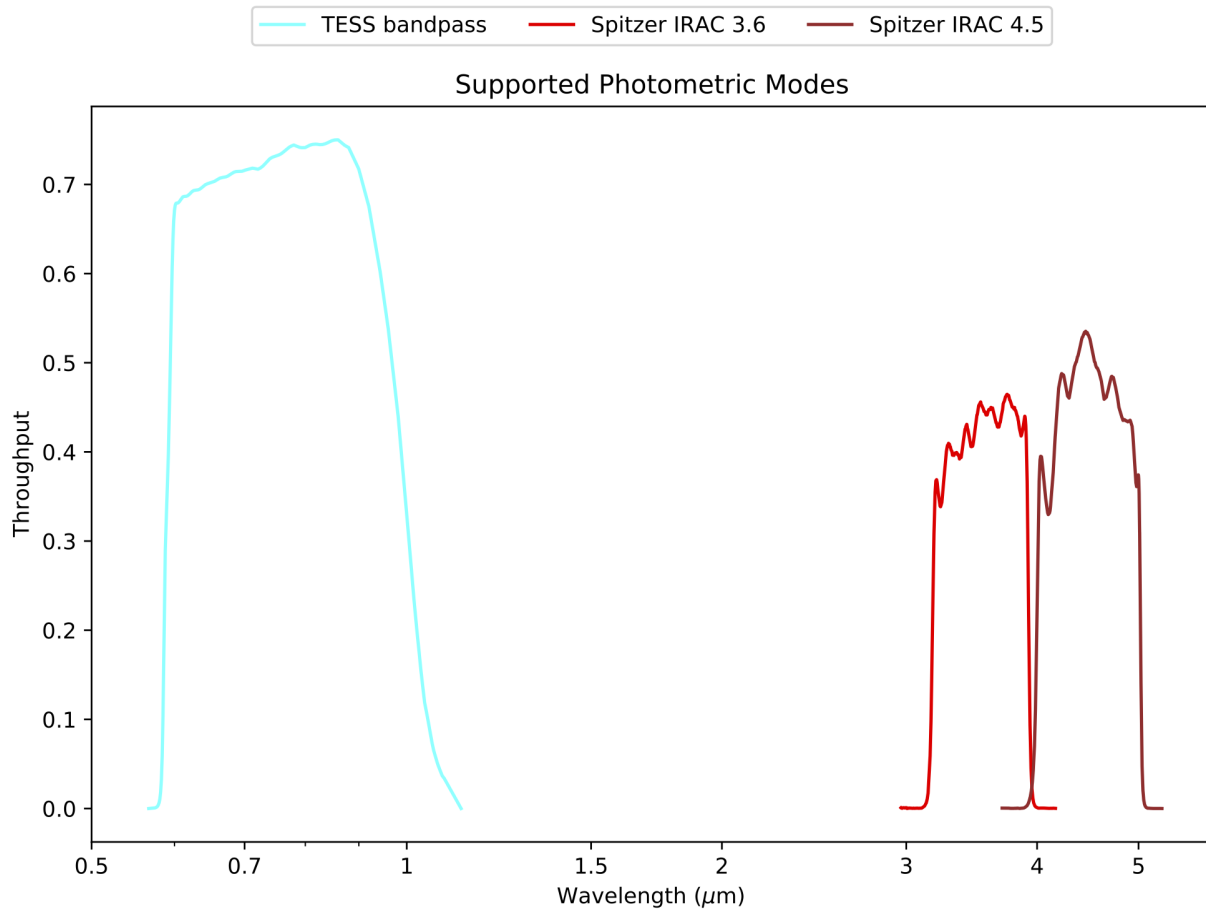


4.2 Photometric

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](#).

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.

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

from exotic_ld import StellarLimbDarkening
```

First, instantiate the `StellarLimbDarkening` class. Here you must specify the stellar parameters and the stellar models. We have also set `verbose=2` to get some extra information about what is going on under the hood, but you can set this to 1 or 0 to get fewer or no outputs to your console.

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

```
Input stellar parameters are M_H=0.01, Teff=5512, logg=4.47.
Loading stellar model from mps1 grid.
Using interpolation type = nearest.
Matched nearest with M_H=0.0, Teff=5500.0, logg=4.5.
```

```
Downloading https://www.star.bris.ac.uk/exotic-ld-data/mps1/MH0.0/teff5500/logg4.5/mps1_
↪spectra.dat: 100%|| 561k/561k [00:00<00:00, 80.3MiB/s]
```

```
Downloaded exotic_ld_data/mps1/MH0.0/teff5500/logg4.5/mps1_spectra.dat.
Stellar model loaded.
```

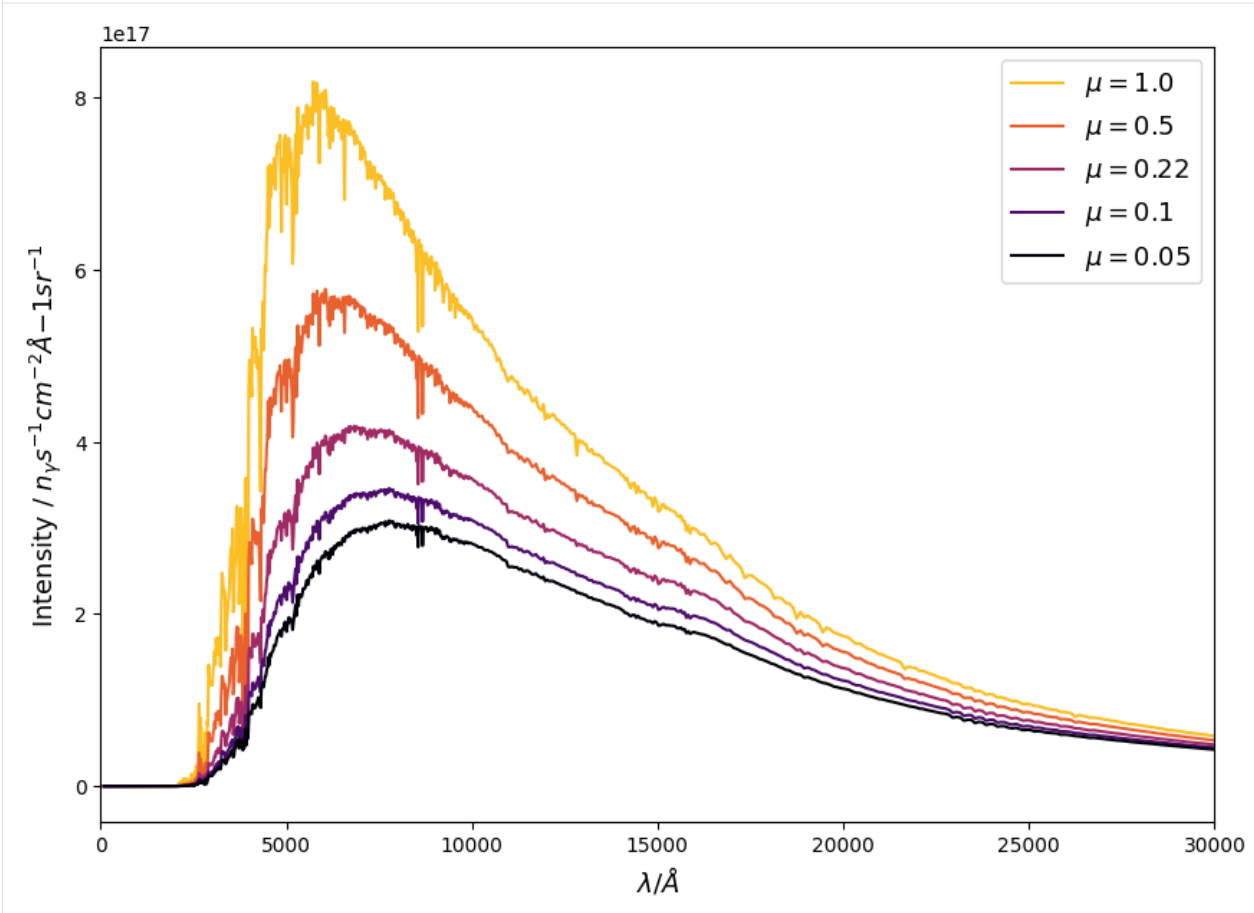
During instantiation stellar models are loaded based on your input stellar parameters. By default, the nearest matching model is located in the specified stellar grid. However, you may set `interpolate_type="trilinear"`, and the stellar models will be linearly interpolated in each of the three parameter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

      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("$\lambda / \text{\AA}$", fontsize=13)
      plt.ylabel("Intensity / $n_{\gamma} s^{-1} \text{cm}^{-2} \text{\AA}^{-1} \text{sr}^{-1}$", 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",
```

(continues on next page)

(continued from previous page)

`mu_min=0.1)`

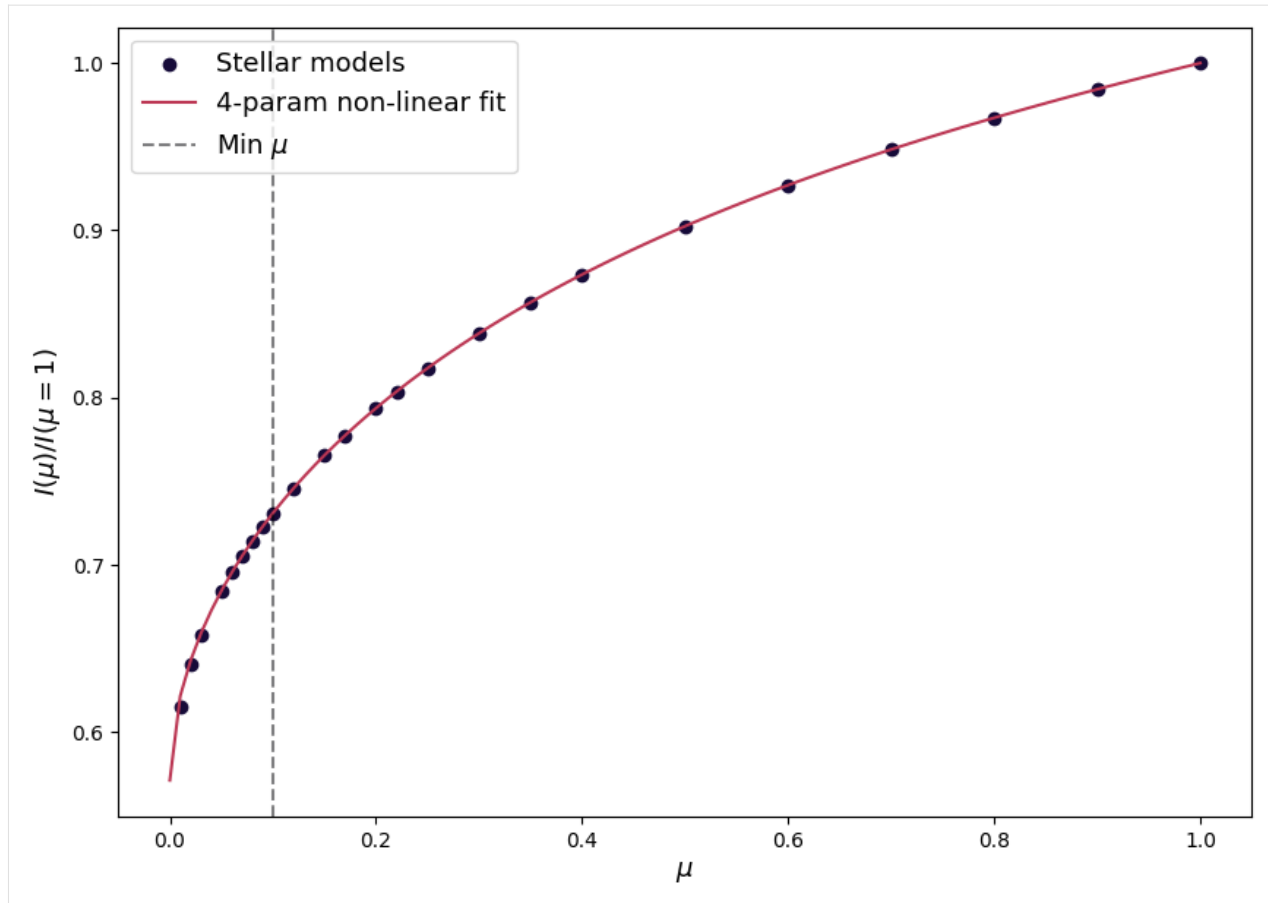
```
Downloading https://www.star.bris.ac.uk/exotic-ld-data/Sensitivity_files/JWST_NIRSpec_
↳Prism_throughput.csv: 100%|| 24.4k/24.4k [00:00<00:00, 15.6MiB/s]
```

```
Downloaded exotic_ld_data/Sensitivity_files/JWST_NIRSpec_Prism_throughput.csv.
Loading instrument mode=JWST_NIRSpec_Prism with wavelength range 60000.0-530000.0 A.
Integrating I(mu) for wavelength limits of 200000.0-300000.0 A.
Integral done for I(mu).
Fitting limb-darkening law to 16 I(mu) data points where 0.1 <= mu <= 1, with the
↳Levenberg-Marquardt algorithm.
Fit done, resulting coefficients are [ 0.49195979  0.13317744 -0.32752393  0.13150929].
```

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.mu, sld.I_mu, color=cm.inferno(0.1), label="Stellar models")
check_mu = np.linspace(0., 1., 100)
plt.plot(check_mu, nonlinear_4param_ld_law(check_mu, *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.ylabel("$I(\mu) / I(\mu=1)$", fontsize=13)
plt.xlabel("$\mu$", 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 likely to be used as priors in your data analysis.

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

from exotic_ld import StellarLimbDarkening
```

First, instantiate the `StellarLimbDarkening` class.

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

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 option returns the standard deviation of the uncertainty

distribution for each limb-darkening coefficient from the fitting algorithm.

```
[4]: 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=0.492+-0.0088
u_2=0.1332+-0.0216
u_3=-0.3275+-0.0226
u_4=0.1315+-0.0085
```

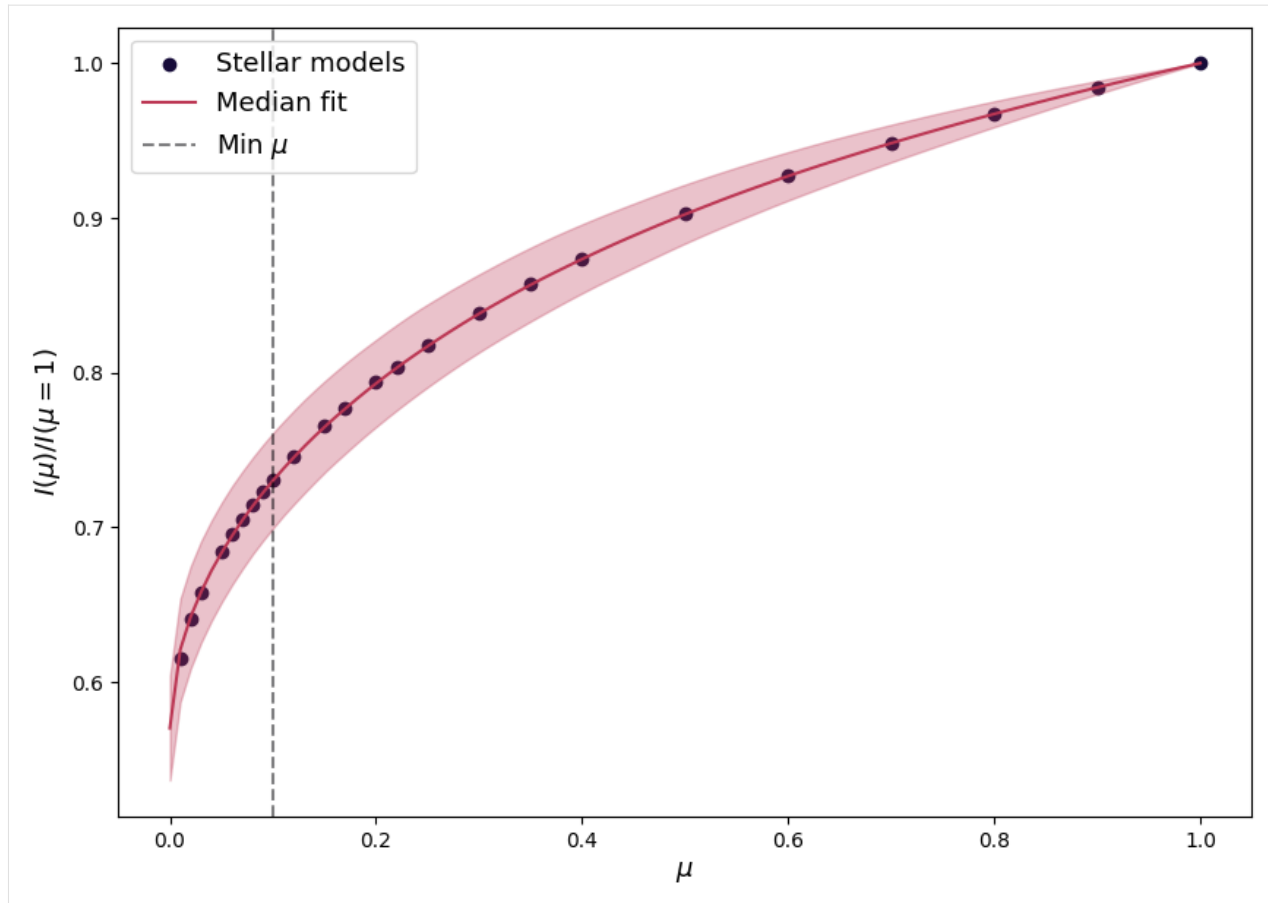
You can visualise this distribution as follows

```
[5]: from exotic_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)

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.ylabel("$I(\mu) / I(\mu=1)$", fontsize=13)
plt.xlabel("$\mu$", 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="exotic_ld_data")
cs = sld.compute_4_parameter_non_linear_ld_coeffs(wavelength_range=[13000., 17000.],
                                                    (continues on next page)
```

(continued from previous page)

```

mode="custom",
custom_wavelengths=wvs,
custom_throughput=throughput)

print(cs)
(0.6196927861080398, 0.1956224528931391, -0.4096670803299519, 0.1603758009182527)

```

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="exotic_ld_data",
                           ld_model="custom",

```

(continues on next page)

(continued from previous page)

```

        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)

(0.0005949930583144949, 0.16760039051680015, 0.0026860142027271758, 0.
↪ 0012534501563158235)

```

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

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

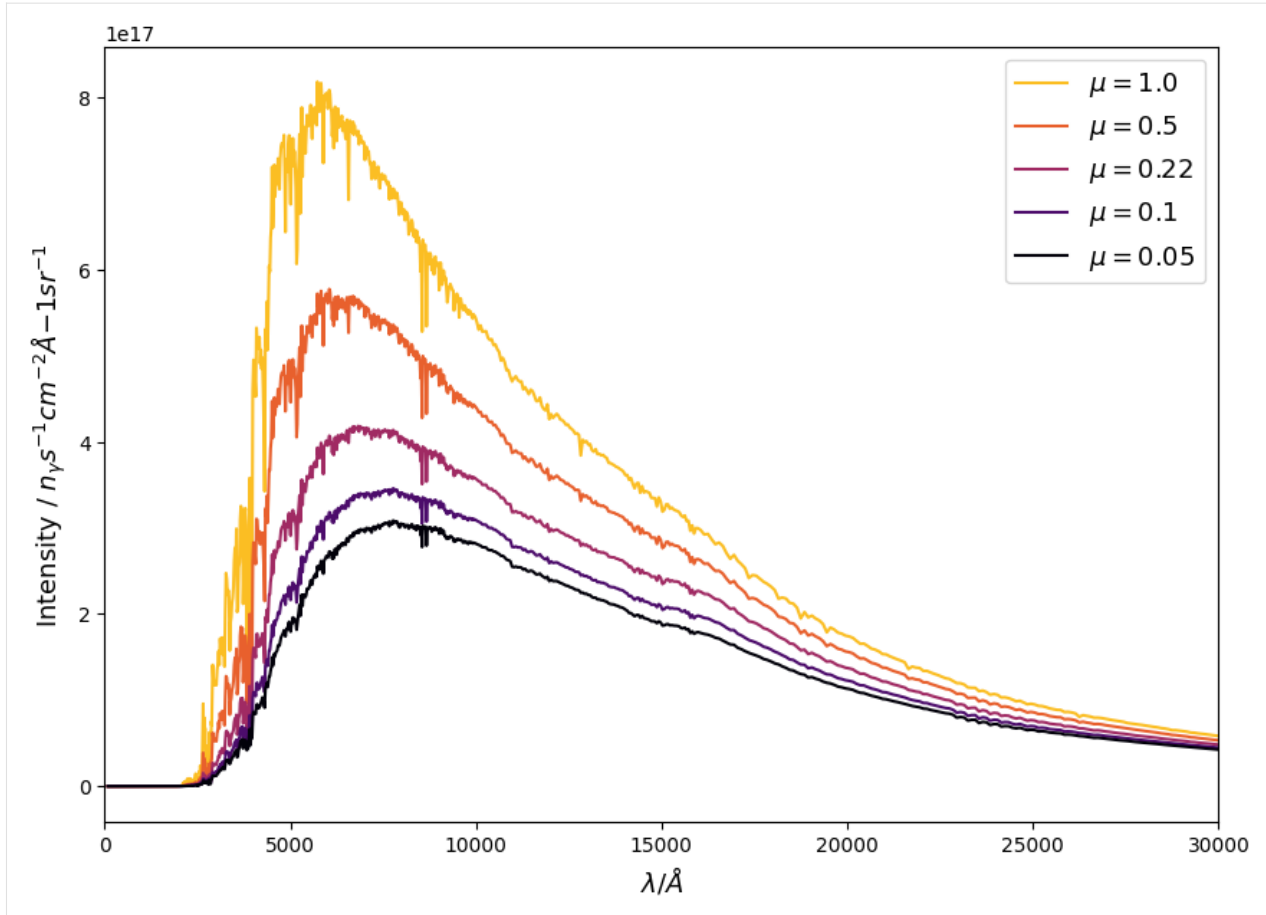
```

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

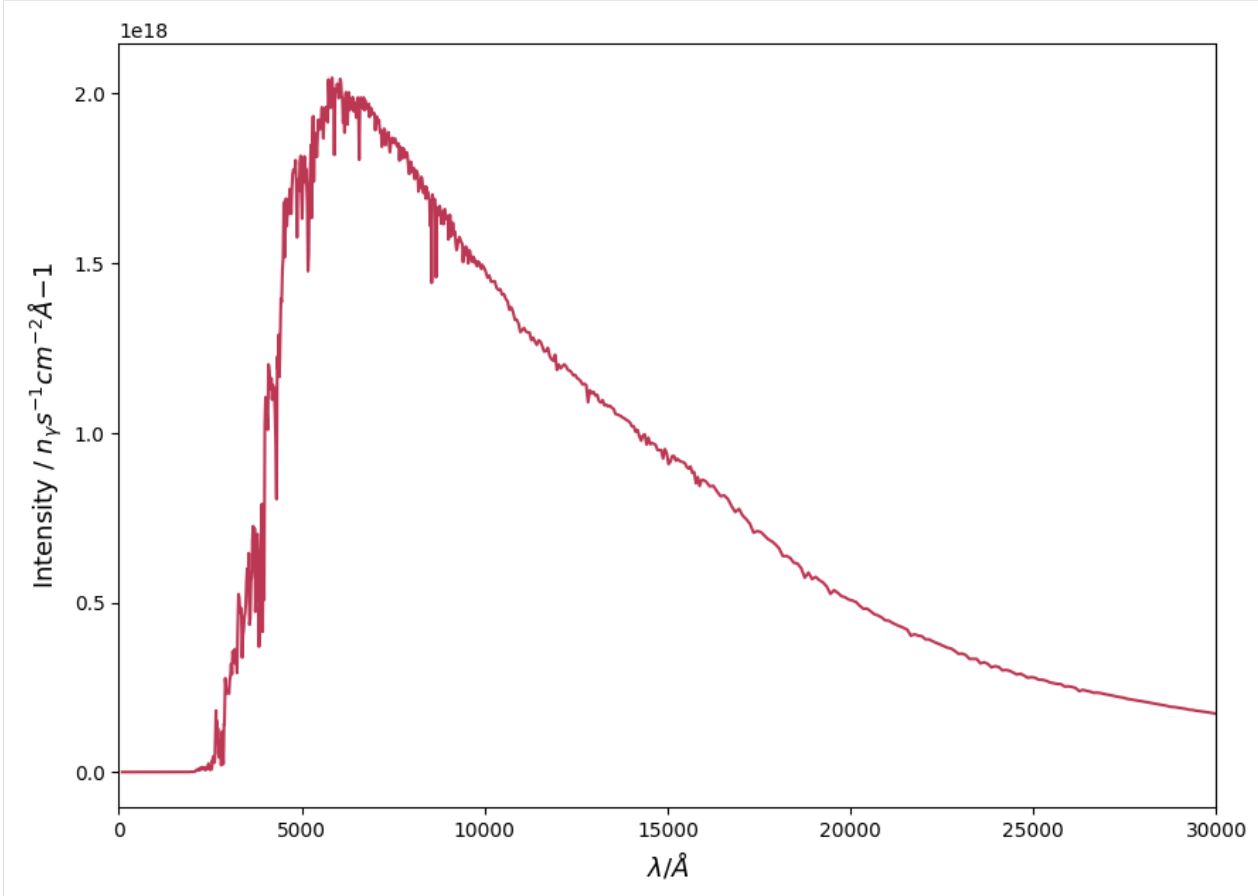
    spectrum.append((b - a) / 2. * integrand(t).dot(weights))

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()
```



6.1 Primary Interface

<code>StellarLimbDarkening([M_H, Teff,logg,...])</code>	Stellar limb darkening class.
---	-------------------------------

6.2 Subpackages

6.2.1 Limb-darkening laws

<code>linear_ld_law(mu, u1)</code>	Linear limb darkening law.
<code>quadratic_ld_law(mu, u1, u2)</code>	Quadratic limb darkening law.
<code>kippling_ld_law(mu, q1, q2)</code>	Kipping limb darkening law.
<code>squareroot_ld_law(mu, u1, u2)</code>	Square root limb darkening law.
<code>nonlinear_3param_ld_law(mu, u1, u2, u3)</code>	Non-linear 3-parameter limb darkening law.
<code>nonlinear_4param_ld_law(mu, u1, u2, u3, u4)</code>	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.kipping_ld_law

`exotic_ld.ld_laws.kipping_ld_law(mu, q1, q2)`

Kipping 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.

CITATION

If you make use of ExoTiC-LD in your research, please cite [Grant and Wakeford 2022](#):

```
@software{david_grant_2022_7437681,  
  author      = {David Grant and  
                 Hannah R. Wakeford},  
  title       = {Exo-TiC/ExoTiC-LD: ExoTiC-LD v3.0.0},  
  month       = dec,  
  year        = 2022,  
  publisher    = {Zenodo},  
  version     = {v3.0.0},  
  doi         = {10.5281/zenodo.7437681},  
  url         = {https://doi.org/10.5281/zenodo.7437681}  
}
```

Stellar models

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

```
@article{husser2013new,  
  title={A new extensive library of PHOENIX stellar atmospheres and synthetic spectra},  
  author={Husser, T-O and Wende-von Berg, Sebastian and Dreizler, Stefan and Homeier,  
↪Derek and Reiners, Ansgar and Barman, Travis and Hauschildt, Peter H},  
  journal={Astronomy & Astrophysics},  
  volume={553},  
  year={2013},  
  publisher={EDP Sciences}  
}
```

```
@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},
```

(continues on next page)

(continued from previous page)

```

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{kostogryz2023mps,
  title={MPS-ATLAS library of stellar model atmospheres and spectra},
  author={Kostogryz, N and Shapiro, AI and Witzke, V and Grant, D and Wakeford, HR and
→Stevenson, KB and Solanki, SK and Gizon, L},
  journal={Research Notes of the AAS},
  volume={7},
  number={3},
  pages={39},
  year={2023},
  publisher={The American Astronomical Society}
}

```

```

@article{kostogryz2022stellar,
  title={Stellar limb darkening. A new MPS-ATLAS library for Kepler, TESS, CHEOPS, 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={Astronomy \& Astrophysics},
  volume={666},
  pages={A60},
  year={2022},
  publisher={EDP Sciences}
}

```

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 Luginja. 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

`exotic_ld`, [23](#)

`exotic_ld.ld_laws`, [23](#)

INDEX

E

`exotic_ld`
 module, [23](#)
`exotic_ld.ld_laws`
 module, [23](#)

K

`kippling_ld_law()` (*in module `exotic_ld.ld_laws`*), [24](#)

L

`linear_ld_law()` (*in module `exotic_ld.ld_laws`*), [23](#)

M

module
 `exotic_ld`, [23](#)
 `exotic_ld.ld_laws`, [23](#)

N

`nonlinear_3param_ld_law()` (*in module `exotic_ld.ld_laws`*), [24](#)
`nonlinear_4param_ld_law()` (*in module `exotic_ld.ld_laws`*), [24](#)

Q

`quadratic_ld_law()` (*in module `exotic_ld.ld_laws`*), [23](#)

S

`squareroot_ld_law()` (*in module `exotic_ld.ld_laws`*),
[24](#)