simlightcurve Documentation

Release 0+untagged.23.g801b8e8

Tim Staley

May 11, 2017

Contents

Version 0+untagged.23.g801b8e8

Contents:

CHAPTER 1

Example notebooks:

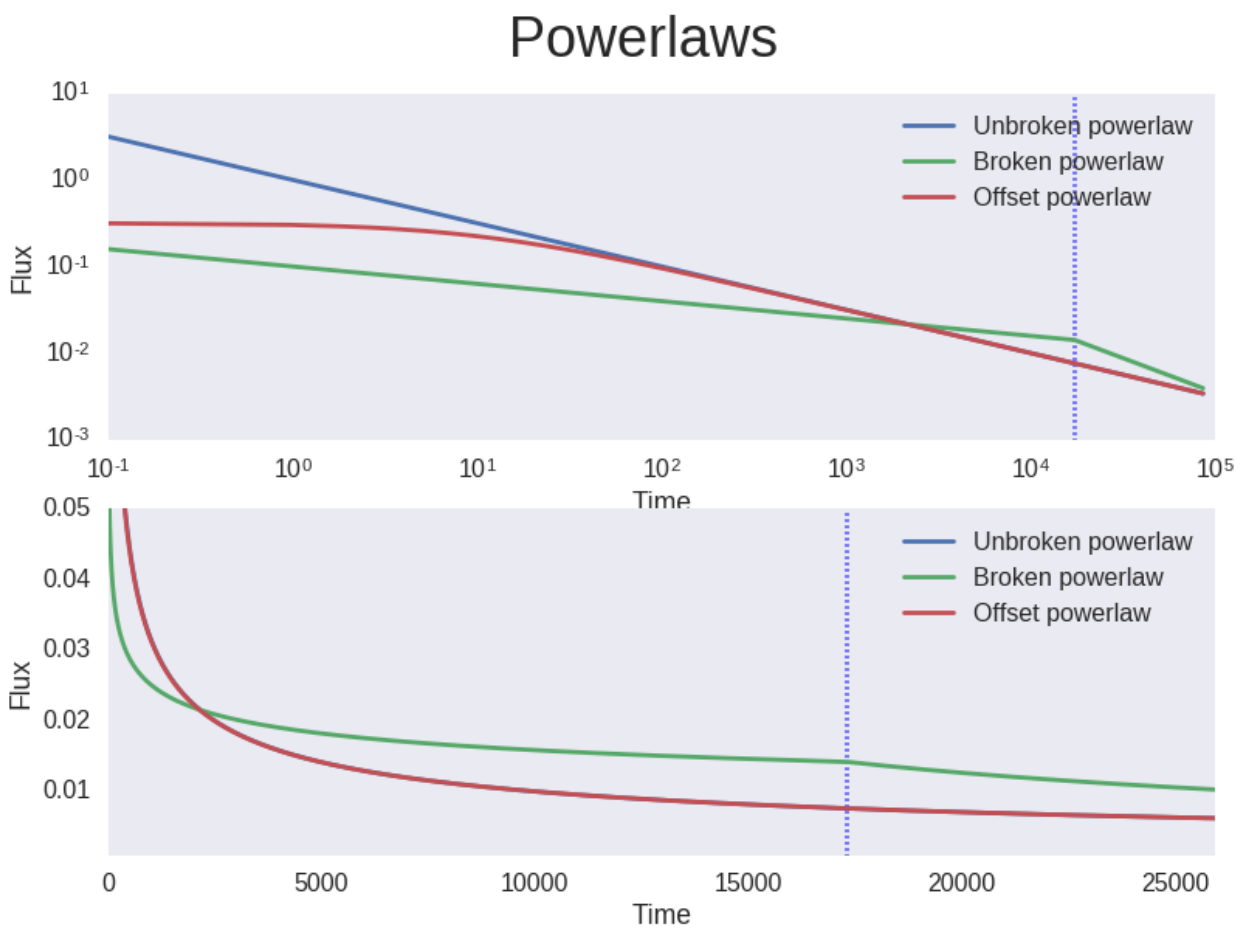
Powerlaw curves

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import seaborn

In [2]: %matplotlib inline
        seaborn.set_context('poster')
        seaborn.set_style("dark")

In [3]: from simlightcurve import curves



        hr = 60*60
        decay_tau=1.*24*hr
        rise_tau=decay_tau*0.3
        t_min = 0.1
        break_one_t_offset = 0.2*decay_tau

        unbroken_pl = curves.Powerlaw(init_amp=1,
                                      alpha_one=-0.5,
                                      t_offset_min=t_min,
                                      t0=None)

        offset_pl = curves.Powerlaw(init_amp=1,
                                    alpha_one=-0.5,
                                    t_offset_min=t_min+1.0,
                                    t0=-10
                                    )

        broken_pl = curves.SingleBreakPowerlaw(init_amp=.1,
                                    alpha_one=-0.2,
                                    break_one_t_offset=break_one_t_offset,
                                    alpha_two=-0.8,
```

```
                                    t_offset_min=t_min,
                                    t0=None
                                    )


        tsteps = np.linspace(t_min, decay_tau, 1e5, dtype=np.float)
```

In [4]:
```
fig, axes = plt.subplots(2,1)
fig.suptitle('Powerlaws', fontsize=36)
ax=axes[0]
# ax.axvline(0, ls='--')
ax.axvline(break_one_t_offset, ls=':')
ax.set_xlabel('Time')
ax.set_ylabel('Flux')
ax.plot(tsteps, unbroken_pl(tsteps), label='Unbroken powerlaw')
ax.plot(tsteps, broken_pl(tsteps), label='Broken powerlaw')
ax.plot(tsteps, offset_pl(tsteps), label='Offset powerlaw')
ax.set_yscale('log')
ax.set_xscale('log')
# ax.set_ylim(0.001,.1)
# ax.set_xlim(t_min, 0.8*decay_tau)
ax.legend()

ax=axes[1]
# ax.axvline(0, ls='--')
ax.axvline(break_one_t_offset, ls=':')
ax.set_xlabel('Time')
ax.set_ylabel('Flux')
ax.plot(tsteps, unbroken_pl(tsteps), label='Unbroken powerlaw')
ax.plot(tsteps, broken_pl(tsteps), label='Broken powerlaw')
ax.plot(tsteps, offset_pl(tsteps), label='Offset powerlaw')
# ax.set_yscale('log')
# ax.set_xscale('log')
ax.set_ylim(0.001,.05)
ax.legend()
ax.set_xlim(t_min, 0.3*decay_tau)

# plt.gcf().tight_layout()
```

Out[4]:
```
(0.1, 25920.0)
```



Supernovae

Optical

Make use of the quadratically-modulated sigmoidal rise / exponential decay, cf Karpenka 2012 and references therein:

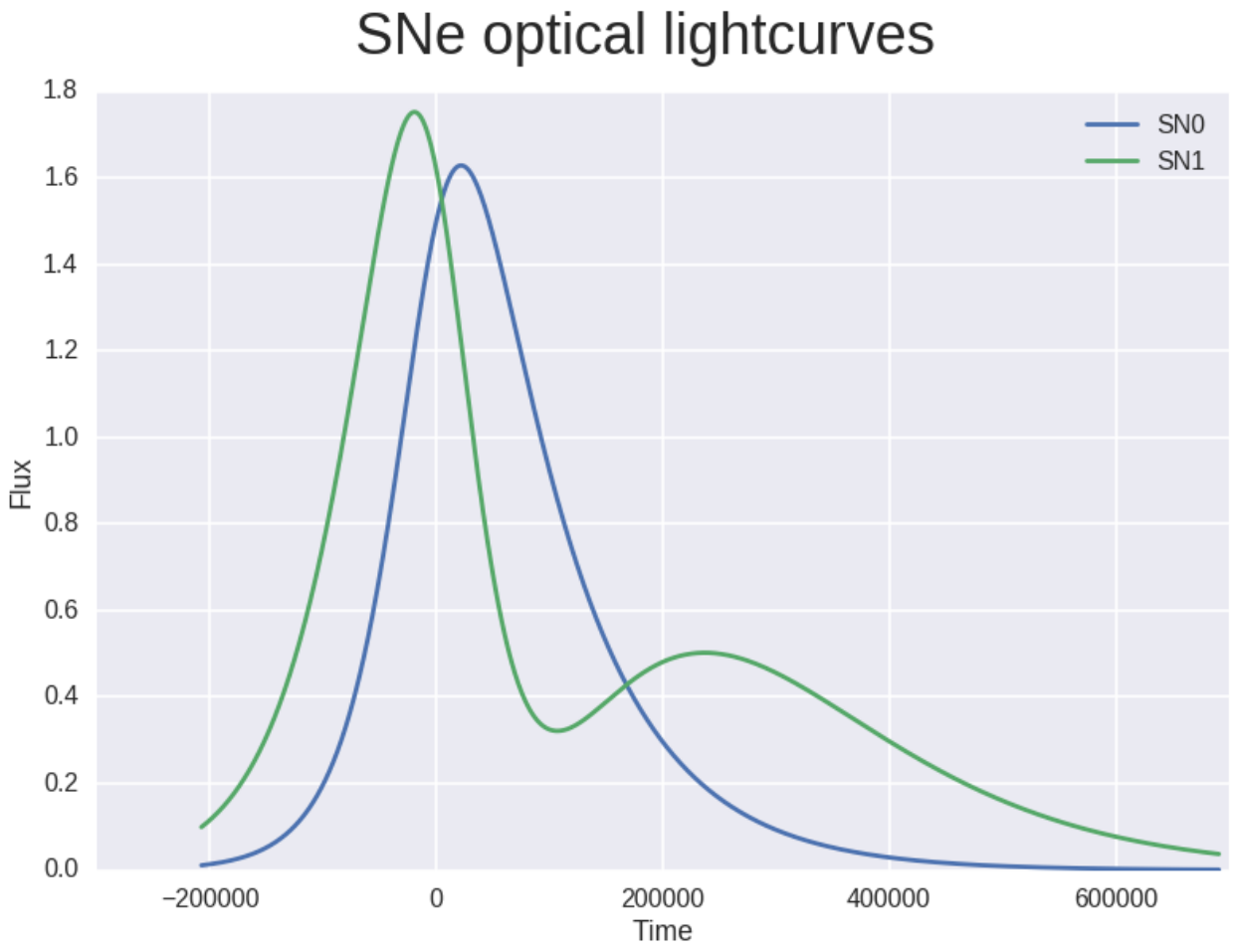
Class: *simlightcurve.curves.modsigmoidexp.ModSigmoidExp*

```
In [1]: from __future__ import absolute_import

        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn

In [2]: %matplotlib inline
        seaborn.set_context('poster')
        seaborn.set_style("darkgrid")

In [3]: from simlightcurve.curves import ModSigmoidExp as Hump


        hr = 60*60
        decay_tau=1.*24*hr
        rise_tau=decay_tau*0.3
        t1_offset = decay_tau
```

```
        sn0 = Hump(a=3, b=0,
                   t1_minus_t0=t1_offset,
                   rise_tau=rise_tau, decay_tau = decay_tau,
                   t0=None
                   )
        sn1 = Hump( a=1, b=3e-10,
                   t1_minus_t0=t1_offset,
                   rise_tau=rise_tau, decay_tau = decay_tau,
                   # t0 = 0.7*decay_tau
                   t0=None
                   )

        tsteps = np.arange(-8*rise_tau, 8*decay_tau, 30)
```

In [4]:
```
fig, axes = plt.subplots(1,1)
fig.suptitle('SNe optical lightcurves', fontsize=36)
ax=axes
# ax.axvline(0, ls='--')
# ax.axvline(t1_offset, ls='--')
ax.set_xlabel('Time')
ax.set_ylabel('Flux')
ax.plot(tsteps, sn0(tsteps), label='SN0')
ax.plot(tsteps, sn1(tsteps), label='SN1')
ax.legend()
```

Out[4]:
```
<matplotlib.legend.Legend at 0x7f87c30d9908>
```



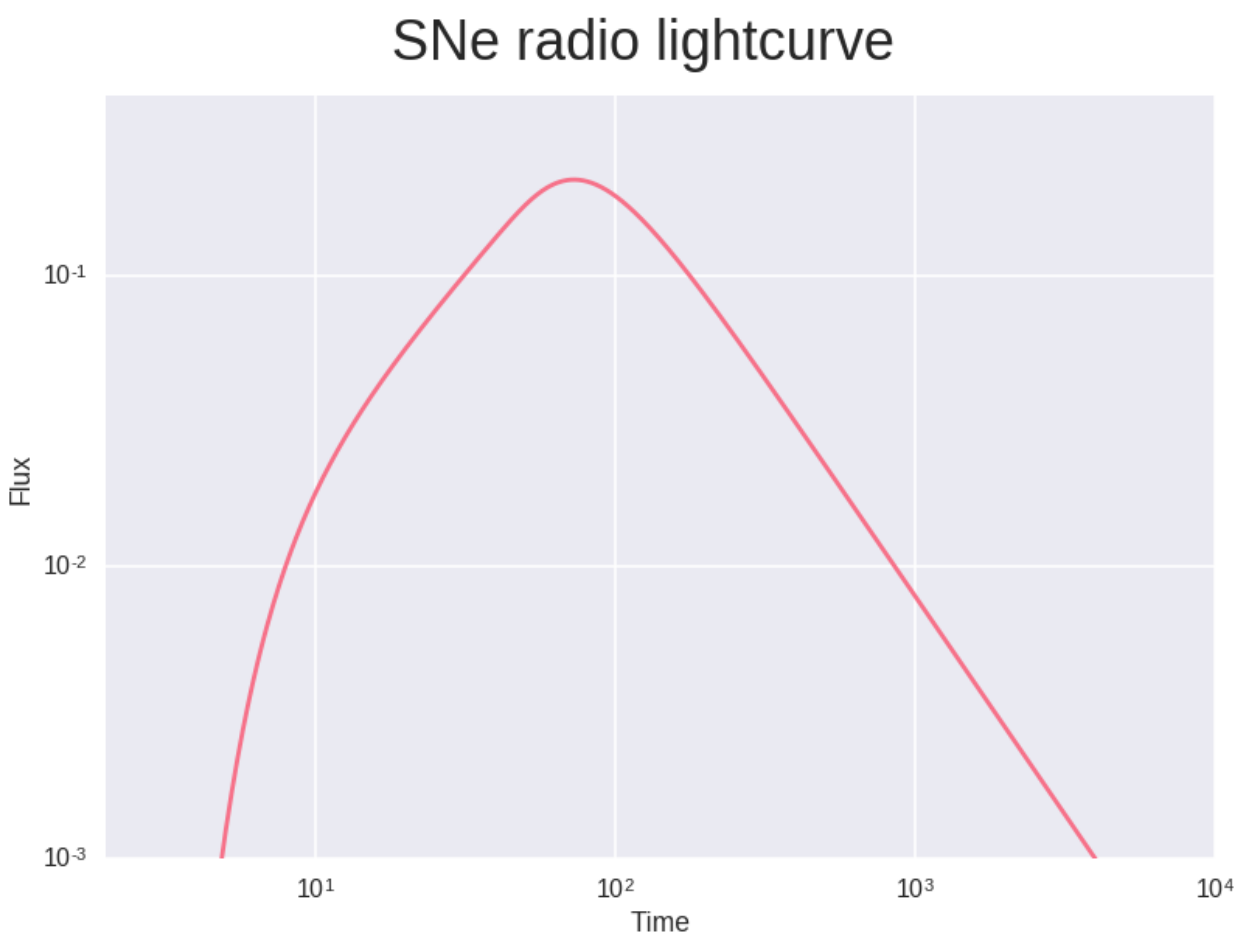
Radio

Make use of the 'minishell' model, a product of factors including exponential decay and power law, following VAST memo #3, (Ryder 2010)

In [5]:
```
from simlightcurve.curves import Minishell

hr=3600.0
timespan = 10000.
tsteps = np.linspace(-1,timespan,24*hr)

afterglow = Minishell(k1=2.5e2, k2=1.38e2, k3=1.47e5,
                      beta=-1.5, delta1=-2.56, delta2=-2.69,
                      t0=None)
```

In [6]:
```
seaborn.set_palette('husl')
fig, axes = plt.subplots(1,1)
fig.suptitle('SNe radio lightcurve', fontsize=36)
lc = afterglow(tsteps)
axes.plot(tsteps, lc)
axes.set_xlabel('Time')
axes.set_ylabel('Flux')
axes.set_xscale('log')
axes.set_yscale('log')
axes.set_ylim(0.001,np.max(lc)+0.2)
axes.set_xlim(2,timespan)
```

Out[6]:
```
(2, 10000.0)
```



General rise and decay curves

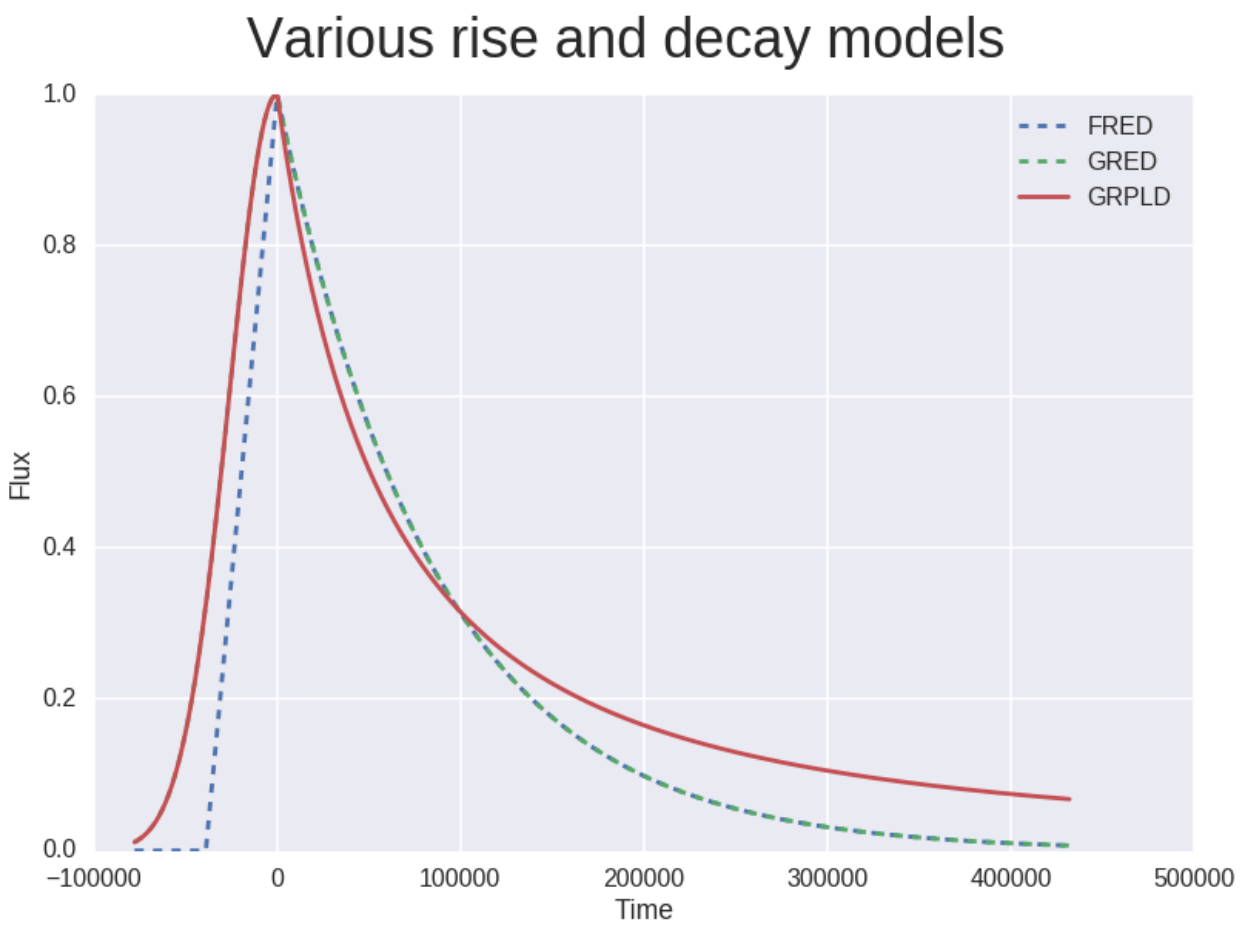
Various curves named after their rise and decay functions.

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import seaborn

In [2]: %matplotlib inline
        seaborn.set_context('poster')
        seaborn.set_style("darkgrid")

In [3]: from simlightcurve import curves

        hr = 60*60
        decay_tau=1.*24*hr
        rise_tau=decay_tau*0.3
        t1_offset = decay_tau

        fred = curves.LinearExp(
            amplitude=1.0,
            rise_time=rise_tau*1.5,
            decay_tau=decay_tau,
            t0=None
        )
```

```
        gred = curves.GaussExp(
                    amplitude=1.0,
                    rise_tau=rise_tau,
                    decay_tau=decay_tau,
                    t0=None
        )

        grpld = curves.GaussPowerlaw(
            amplitude = 1.0,
            rise_tau=rise_tau,
            decay_alpha=-1.5,
            decay_offset=decay_tau,
            t0=None
        )

        tsteps = np.arange(-rise_tau*3, decay_tau*5, 30)
```

In [4]:
```
fig, axes = plt.subplots(1,1)
fig.suptitle('Various rise and decay models', fontsize=36)
ax=axes
# ax.axvline(0, ls='--')
# ax.axvline(t1_offset, ls='--')
ax.set_xlabel('Time')
ax.set_ylabel('Flux')
ax.plot(tsteps, fred(tsteps), label='FRED', ls='--')
ax.plot(tsteps, gred(tsteps), label='GRED', ls='--')
ax.plot(tsteps, grpld(tsteps), label='GRPLD')
# ax.set_xscale('log')
# ax.set_yscale('log')
ax.set_ylim(1e-5,1.001)
ax.legend()
```

Out[4]:
```
<matplotlib.legend.Legend at 0x7f0258d2ee48>
```



In [5]:

Van Der Laan

From A Model for Variable Extragalactic Radio Sources, (Van Der Laan 1969).

See class documentation for more details: *simlightcurve.curves.vanderlaan.VanDerLaan*

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn
```

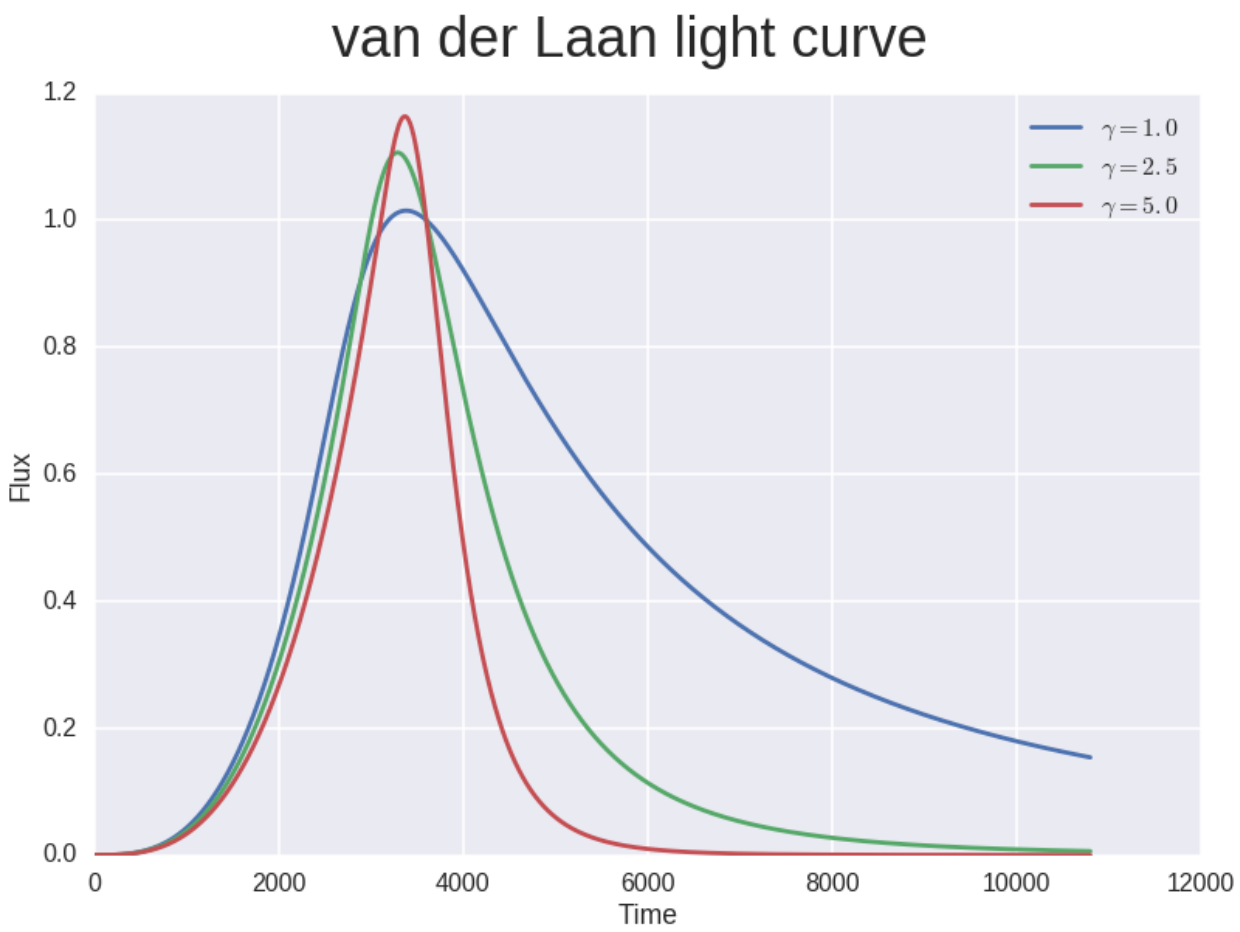
In [2]:
```
%matplotlib inline
seaborn.set_context('poster')
seaborn.set_style("darkgrid")
```

In [3]:
```
from simlightcurve import curves

energy_index_1 = 1.
energy_index_2 = 2.5
energy_index_3 = 5.
maximum_flux = 1.
maximum_time = 60. * 60.

vdl1 = curves.VanDerLaan(amplitude=maximum_flux,
                         energy_index=energy_index_1,
```

```
                                t0=maximum_time
                                )

        vdl2 = curves.VanDerLaan(amplitude=maximum_flux,
                                energy_index=energy_index_2,
                                t0=maximum_time
                                )

        vdl3 = curves.VanDerLaan(amplitude=maximum_flux,
                                energy_index=energy_index_3,
                                t0=maximum_time
                                )

        time_steps = np.arange(0., 3. * maximum_time, 1.)
```

In [4]:
```
fig, axes = plt.subplots(1,1)
fig.suptitle('van der Laan light curve', fontsize=36)
ax=axes
ax.set_xlabel('Time')
ax.set_ylabel('Flux')
for model in (vdl1, vdl2, vdl3):
    ax.plot(time_steps, model(time_steps),
            label='$\gamma={}$'.format(model.energy_index.value),
#             ls='--',
           )
plt.legend()
```

Out[4]:
```
<matplotlib.legend.Legend at 0x7fd0fc8405c0>
```


van der Laan light curve
Flux
Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
2000
4000
6000
8000
10000
12000
$\gamma = 1.0$
$\gamma = 2.5$
$\gamma = 5.0$

CHAPTER 2

API Reference

simlightcurve base modules

simlightcurve.solvers

simlightcurve.solvers.**find_peak**(*curve*, *t_init=0.0*)

Use scipy.optimize.fmin to locate a (possible local) maxima

Returns peak_t_offset, peak_flux

Return type tuple

simlightcurve.solvers.**find_rise_t**(*curve*, *threshold*, *t_min*, *t_max*)

simlightcurve.curves

simlightcurve.curves.minishell

class simlightcurve.curves.minishell.**Minishell**(*k1*, *k2*, *k3*, *beta*, *delta1*, *delta2*, *t0*, ***kwargs*)

Supernova radio-lightcurve model (Type-II).

CF K. Weiler et al, 2002: http://www.annualreviews.org/doi/abs/10.1146/annurev.astro.40.060401.093744

and VAST memo #3, Ryder 2010: http://www.physics.usyd.edu.au/sifa/vast/uploads/Main/vast_memo3.pdf

See Weiler et al for some typical parameter values.

static evaluate(*t*, *k1*, *k2*, *k3*, *beta*, *delta1*, *delta2*, *t0*)

Wraps _curve function to only process values at t > 0

simlightcurve.curves.modsigmoidexp

class simlightcurve.curves.modsigmoidexp.**ModSigmoidExp**(*a*, *b*, *t1_minus_t0*, *rise_tau*, *decay_tau*, *t0*, ***kwargs*)

Sigmoidal rise / exponential decay modulated by a quadratic polynomial.

Typically applied as a supernova optical-lightcurve model, applicable to all SNe types.

Following Karpenka et al 2012; Eq 1. (http://adsabs.harvard.edu/abs/2013MNRAS.429.1278K)

simlightcurve.curves.misc

class simlightcurve.curves.misc.**NegativeQuadratic**(*amplitude*, *t0*, ***kwargs*)

Very simple example, used for testing purposes.

simlightcurve.curves.powerlaw

class simlightcurve.curves.powerlaw.**Powerlaw**(*init_amp*, *alpha_one*, *t_offset_min*, *t0*, ***kwargs*)

Represents a simple power-law curve

The curve is defined as

amplitude * (t_offset)**alpha

Be wary of using an init_alpha<0, since this results in an asymptote at t=0.

NB The curve will always begin at the origin, because maths. (Cannot raise a negative number to a fractional power unless you deal with complex numbers. Also 0.**Y == 0.)

class simlightcurve.curves.powerlaw.**SingleBreakPowerlaw**(*init_amp*, *alpha_one*, *break_one_t_offset*, *alpha_two*, *t_offset_min*, *t0*, ***kwargs*)

Represents an power-law curve with a single index-break

The curve is defined as

init_amplitude * (t_offset)**alpha_one

until the location of the first index-break, then matched_amplitude * (t_offset)**alpha_two

where matched_amplitude is calculated to ensure the curves meet at the power-break location.

We wary of using an init_alpha<0, since this results in an asymptote at t=0.

NB The curve will always begin at the origin, because maths. (Cannot raise a negative number to a fractional power unless you deal with complex numbers. Also 0.**Y == 0.)

simlightcurve.curves.composite

class simlightcurve.curves.composite.gaussexp.**GaussExp**(*amplitude*, *rise_tau*, *decay_tau*, *t0*, ***kwargs*)

amplitude

decay_tau

static evaluate(*t*, *amplitude*, *rise_tau*, *decay_tau*, *t0*)

inputs = ('t',)

outputs = ('flux',)

param_names = ('amplitude', 'rise_tau', 'decay_tau', 't0')

rise_tau

t0

class simlightcurve.curves.composite.gausspowerlaw.**GaussPowerlaw**(*amplitude*, *rise_tau*, *decay_alpha*, *decay_offset*, *t0*, ***kwargs*)

amplitude

decay_alpha

decay_offset

static evaluate(*t*, *amplitude*, *rise_tau*, *decay_alpha*, *decay_offset*, *t0*)

inputs = ('t',)

outputs = ('flux',)

param_names = ('amplitude', 'rise_tau', 'decay_alpha', 'decay_offset', 't0')

rise_tau

t0

class simlightcurve.curves.composite.linearexp.**LinearExp**(*amplitude*, *rise_time*, *decay_tau*, *t0*, ***kwargs*)

amplitude

decay_tau

static evaluate(*t*, *amplitude*, *rise_time*, *decay_tau*, *t0*)

inputs = ('t',)

outputs = ('flux',)

param_names = ('amplitude', 'rise_time', 'decay_tau', 't0')

rise_time

t0

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

R

S

T