(MM, A) LIMEPY

(Multi-Mass, Anisotropic) Lowered Isothermal Model Explorer in PYthon

Mark Gieles & Alice Zocchi
ArXiv:1508.02120

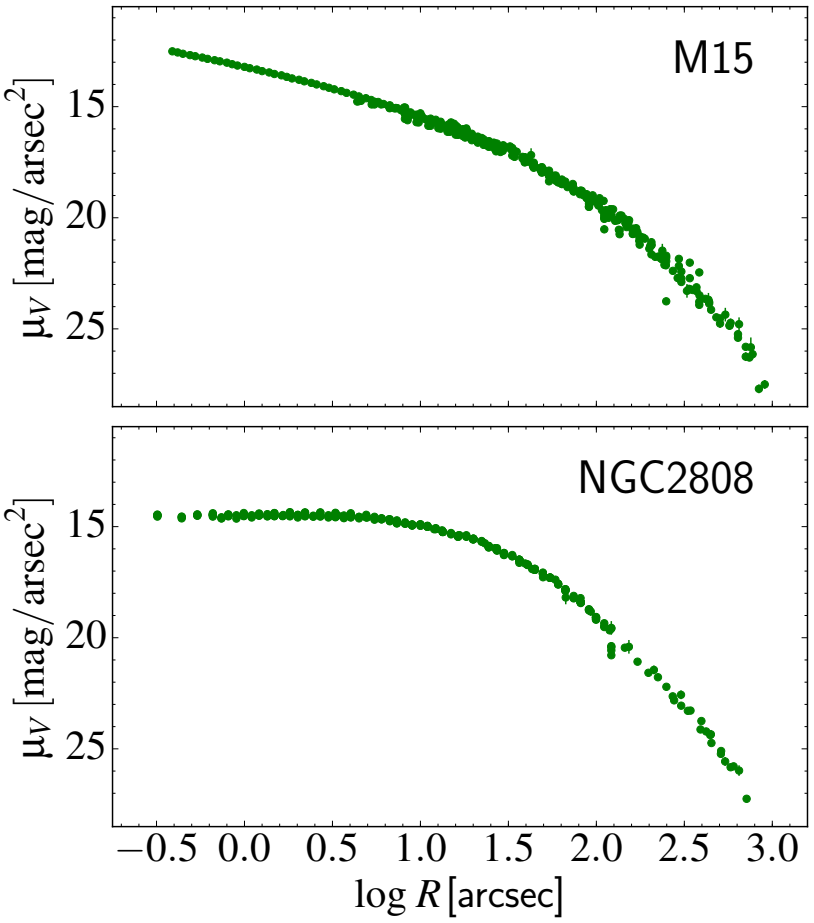
https://github.com/mgieles/limepy

GAIA CHALLENGE
UNIVERSITY OF SURREY

Motivation
Model
Applications
Installing
Running
Plotting

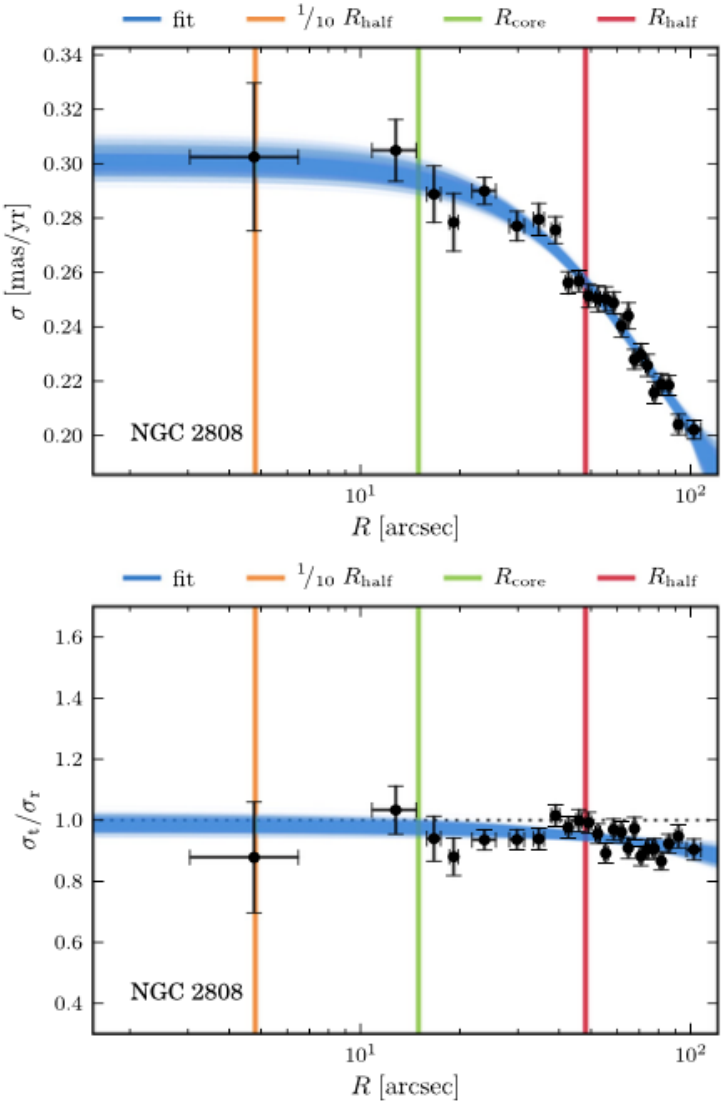
Motivation

Surface brightness



Trager, King & Djorgovski (1995)

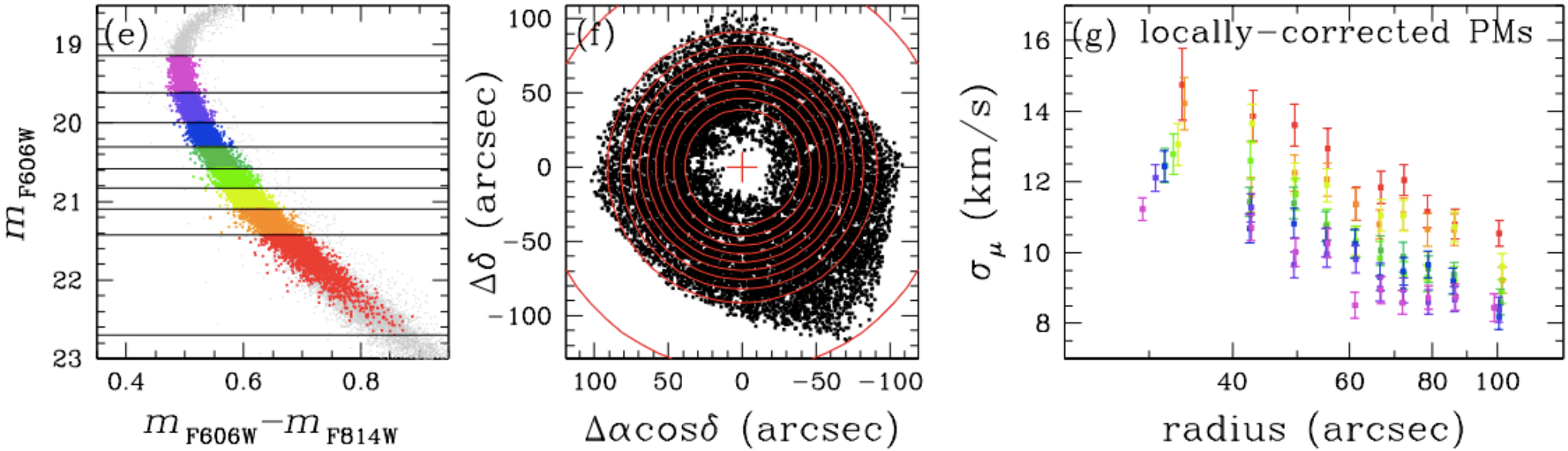
HST proper motions



Watkins+ (2015)

Motivation

Luminosity/mass dependent kinematics



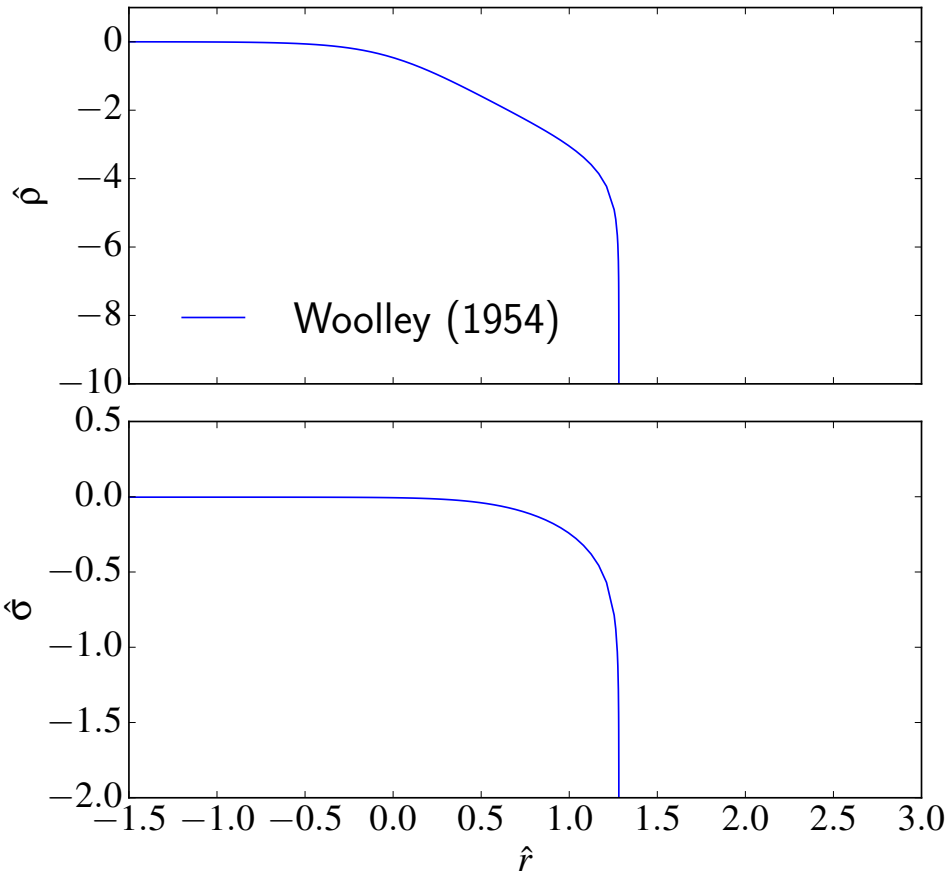
M15: Bellini+ (2014)

Lowered isothermal models

Simple truncation

Woolley (1954)

$$f(\hat{E}) = A \exp(\hat{E})$$



$$\hat{E} = \frac{\phi(r_\mathrm{t}) - E}{s^2}$$

$E =$ specific energy

$\phi(r_\mathrm{t}) =$ specific potential at r_t

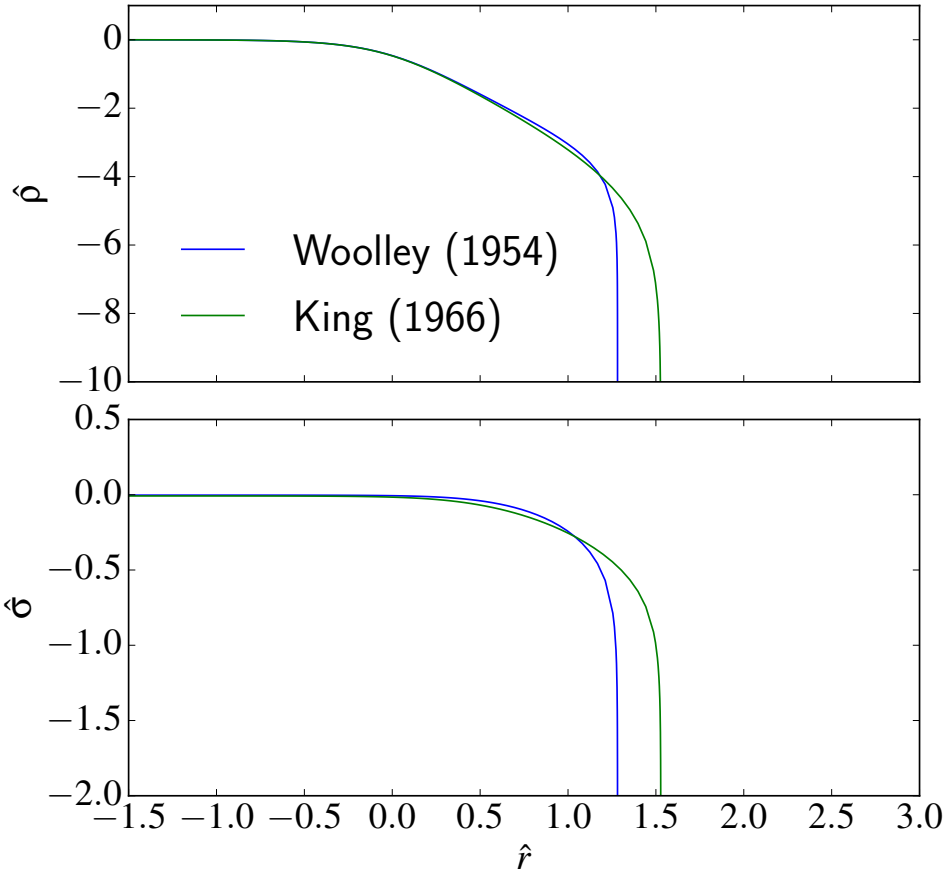
$s =$ velocity scale

Lowered isothermal models

Simple truncation + continous DF

Michie (1963); King (1966)

$$f(\hat{E}) = A\left[\exp(\hat{E}) - 1\right]$$



$$\hat{E} = \frac{\phi(r_\mathrm{t}) - E}{s^2}$$

$E =$ specific energy

$\phi(r_\mathrm{t}) =$ specific potential at r_t

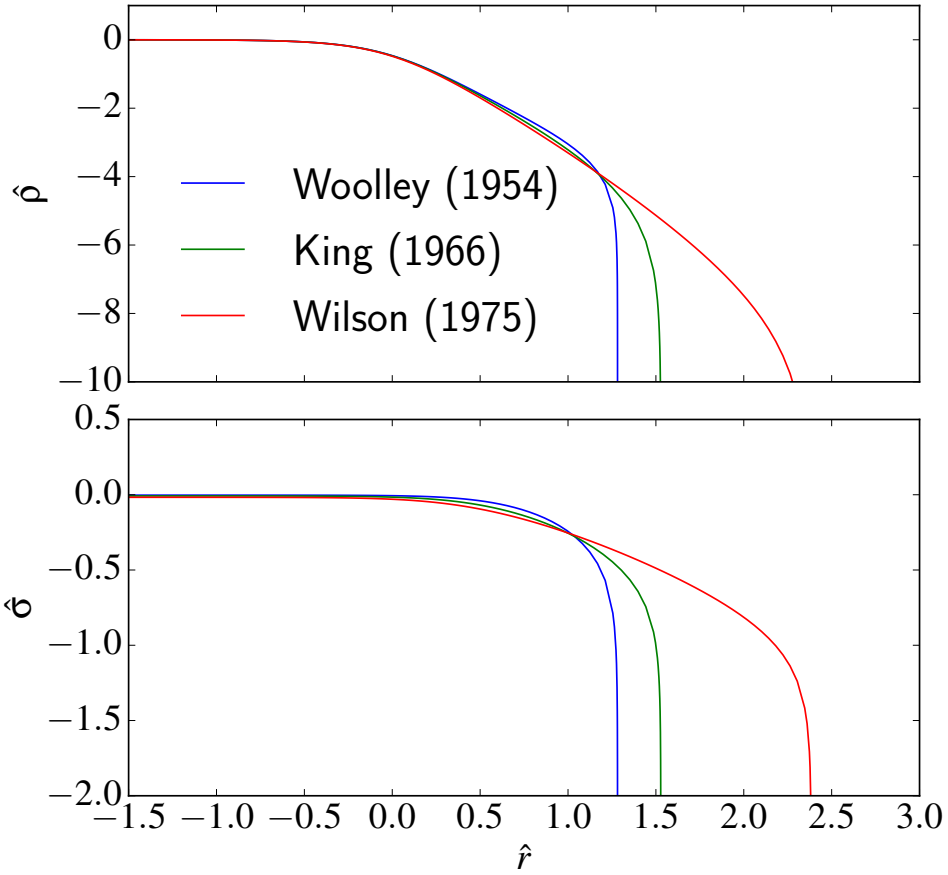
$s =$ velocity scale

Lowered isothermal models

Simple truncation + continous DF & DF'

Wilson (1975)

$$f(\hat{E}) = A\left[\exp(\hat{E}) - 1 - \hat{E}\right]$$



$$\hat{E} = \frac{\phi(r_\mathrm{t}) - E}{s^2}$$

$E =$ specific energy

$\phi(r_\mathrm{t}) =$ specific potential at r_t

$s =$ velocity scale

Lowered isothermal models

$$f(n,\hat{E}) = \begin{cases} A\exp(\hat{E}), & n = 0 \\ A\left[\exp(\hat{E}) - \sum_{i=0}^{n-1} \frac{\hat{E}^i}{i!}\right], & n > 0 \end{cases}$$

Davoust (1977)

$$f(g,\hat{E}) = AE_\gamma(g,\hat{E})$$

$$E_\gamma(g,\hat{E}) = \begin{cases} \exp(\hat{E}), & g = 0 \\ \exp(\hat{E})\frac{\gamma(g,\hat{E})}{\Gamma(g)}, & g > 0 \end{cases}$$

Gomez-Leyton & Valezquez (2014)

LIMEPY

$$f = A \exp(-\hat{J}^2) E_\gamma(g, \hat{E})$$

3 parameters: $\hat{\phi}_0, g, r_\mathrm{a}$

2 scales: A, s

$$\hat{E} = \frac{\phi(r_\mathrm{t}) - E}{s^2}$$

$$\hat{J}^2 = \frac{r^2 v_\mathrm{t}^2}{2 s^2 r_\mathrm{a}^2}$$

Multi-mass LIMEPY

Include mass dependence in *f* in a self-consistent way

$$\nabla^2\phi = 4\pi G \sum_j \int f_j d^3v$$

Da Costa & Freeman 1976; Gunn & Griffin 1979

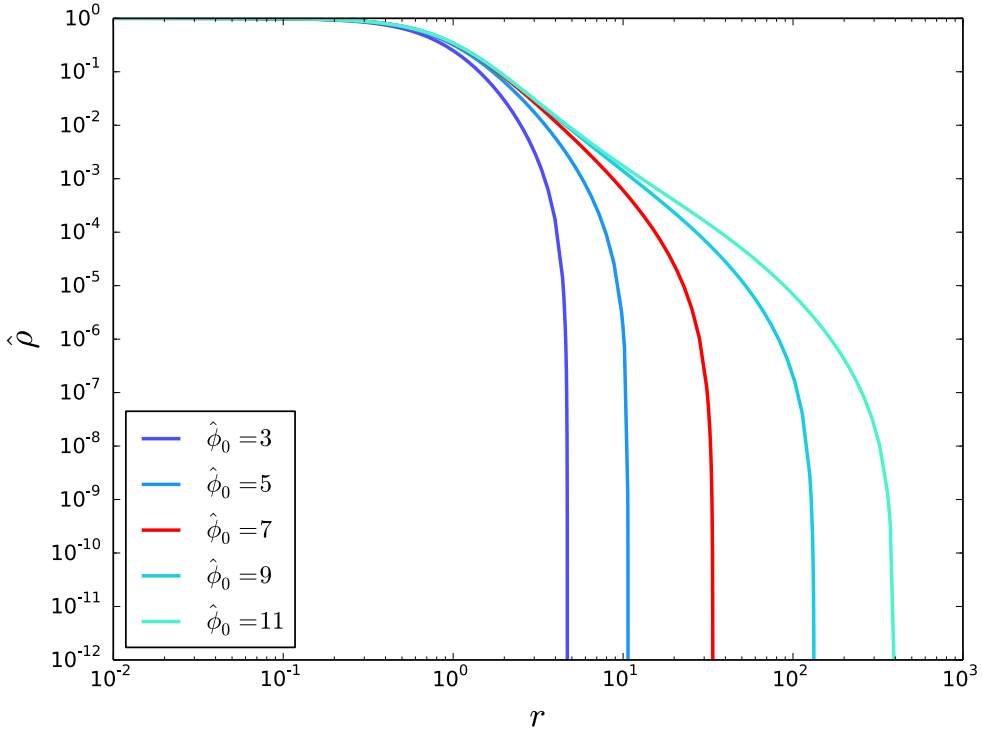
$$s_j = s\left(\frac{m_j}{\bar{m}}\right)^{-\delta} \qquad r_{\mathrm{a},j} = r_{\mathrm{a}}\left(\frac{m_j}{\bar{m}}\right)^{\eta}$$

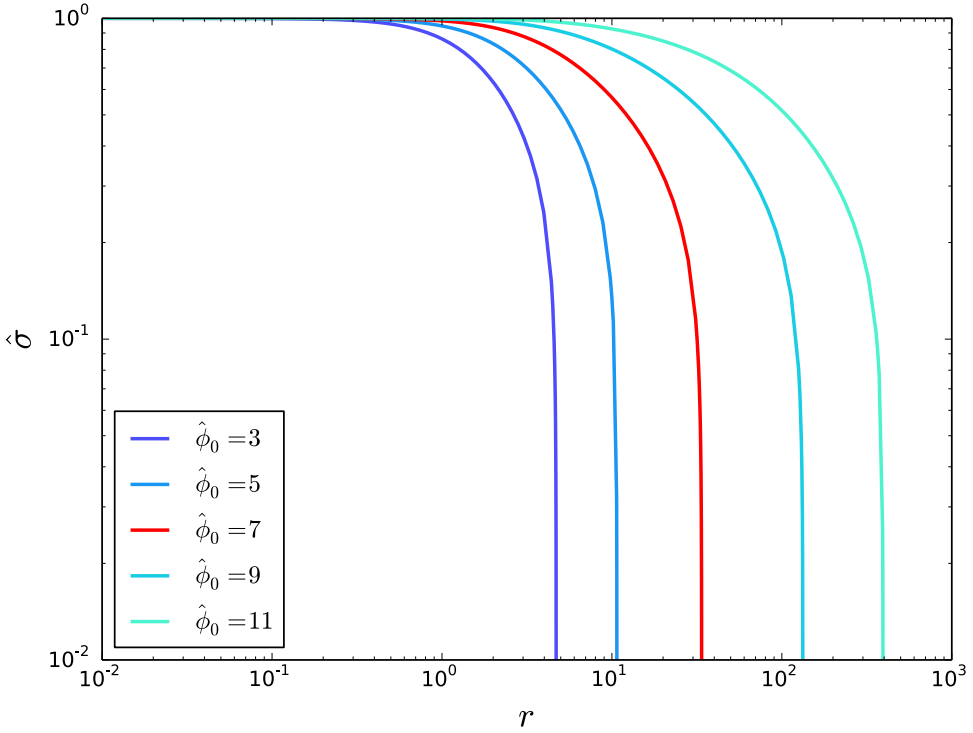
2N$_{\mathrm{bin}}$ + 2 additional parameter: m_j, M_j, δ, η

LIMEPY

Some properties of the models: the role of $\hat{\phi}_0$

$$g = 1,\ r_a \to \infty$$

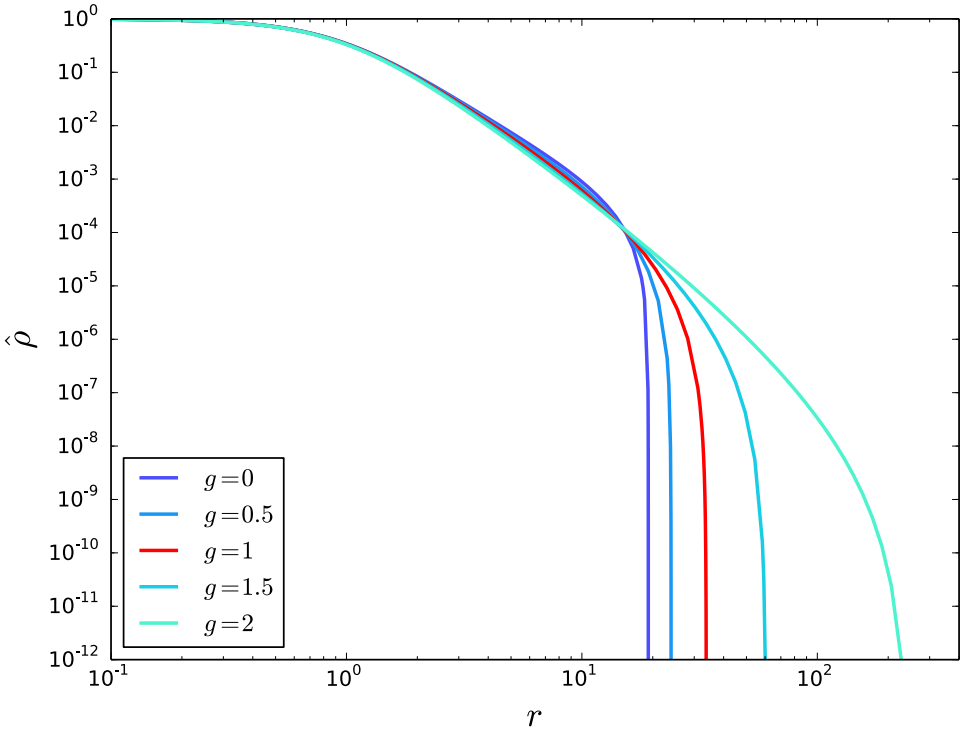


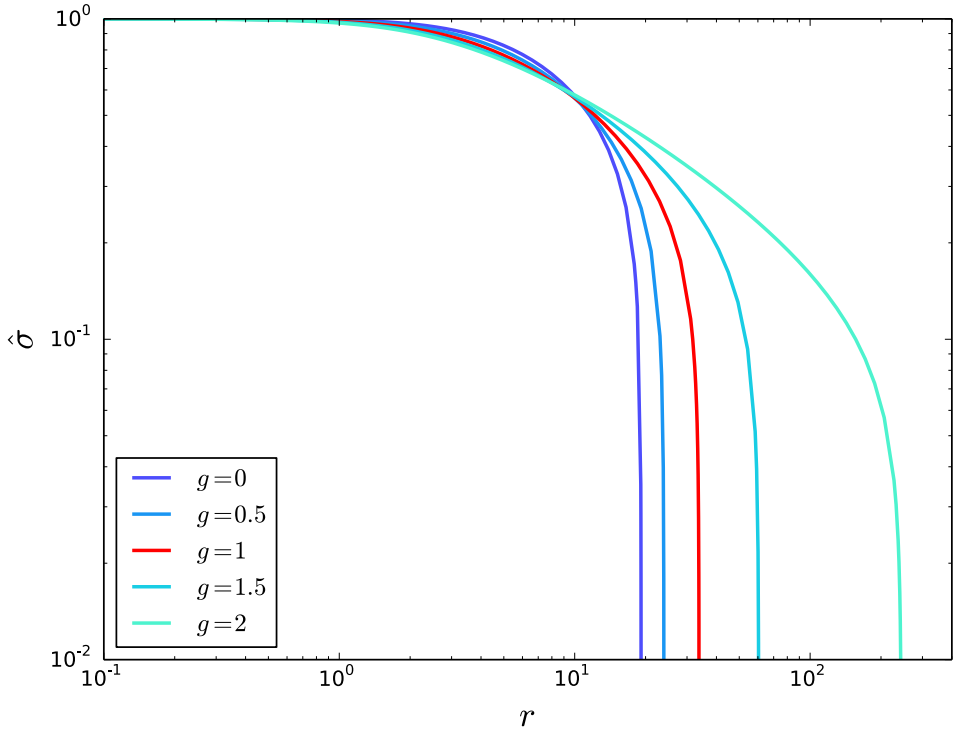


LIMEPY

Some properties of the models: the role of g

$$\hat{\phi}_0 = 7,\ r_a \to \infty$$

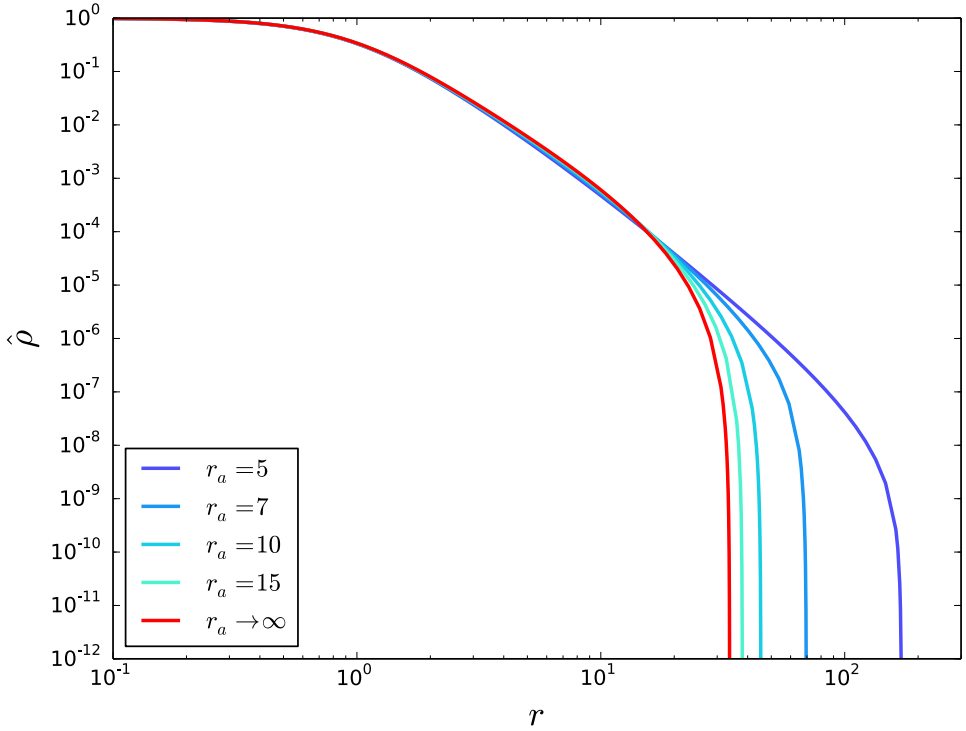


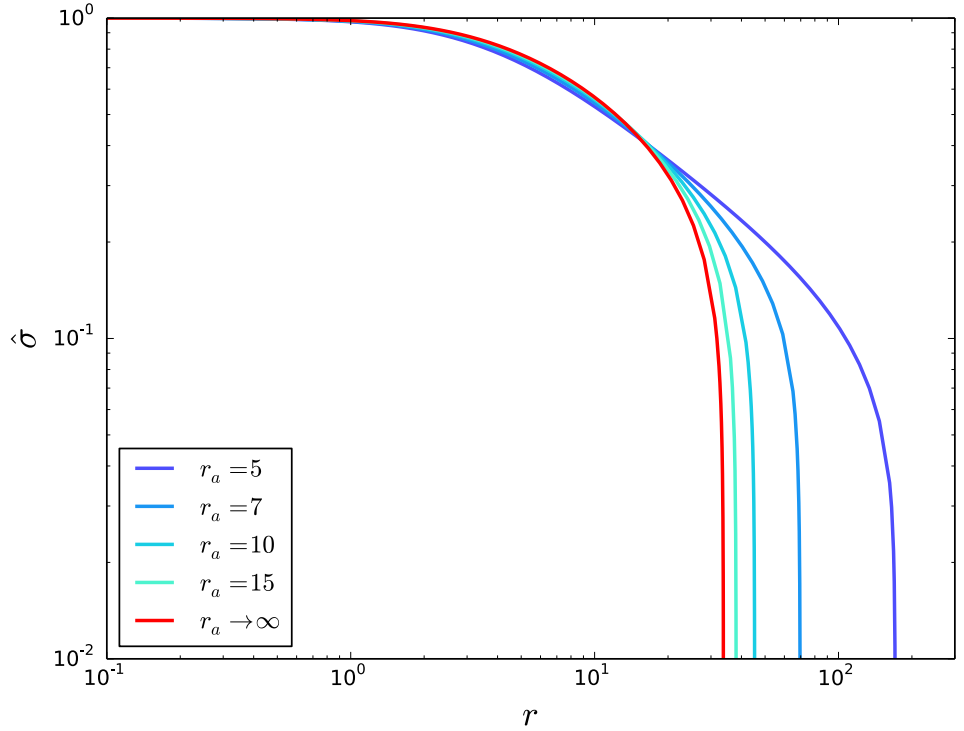


LIMEPY

Some properties of the models: the role of r_a

$$\hat{\phi}_0 = 7,\ g = 1$$

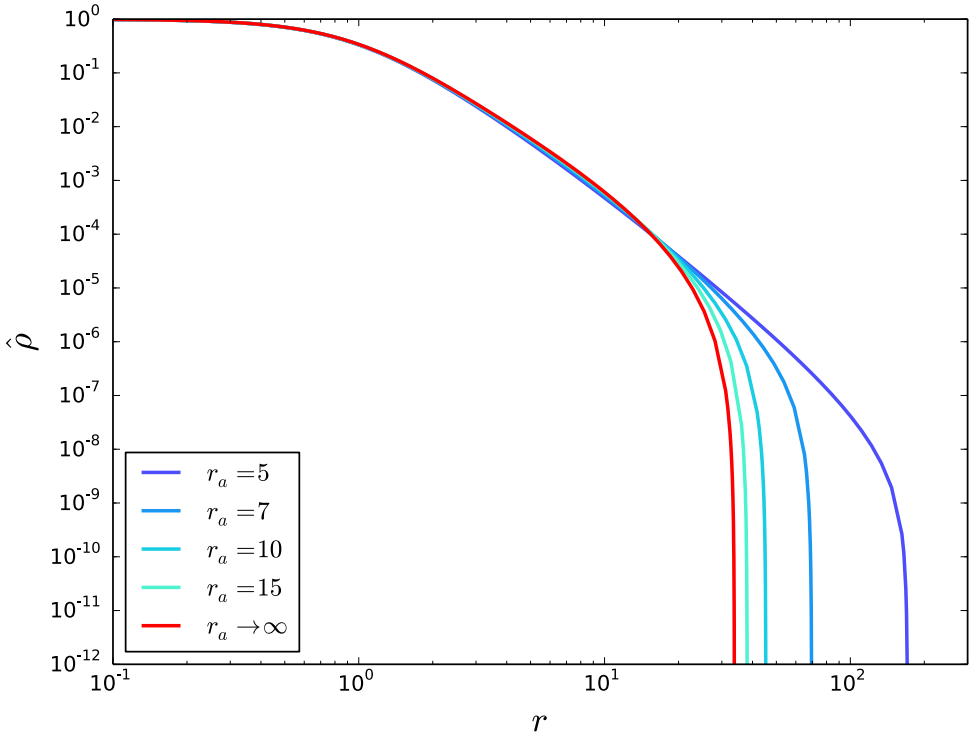


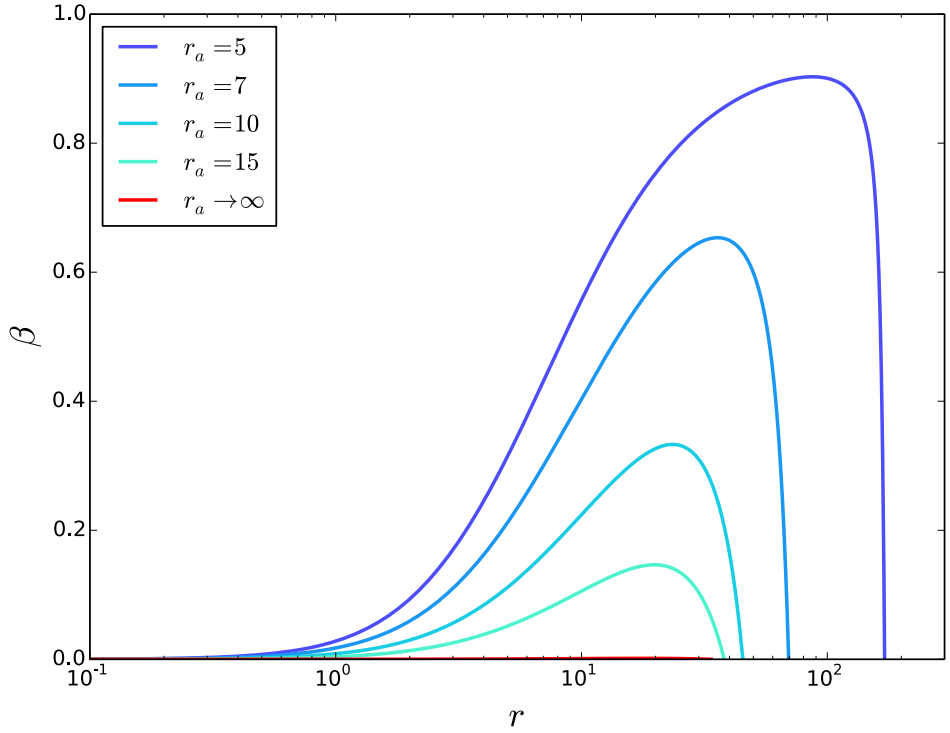


LIMEPY

Some properties of the models: the role of r_a

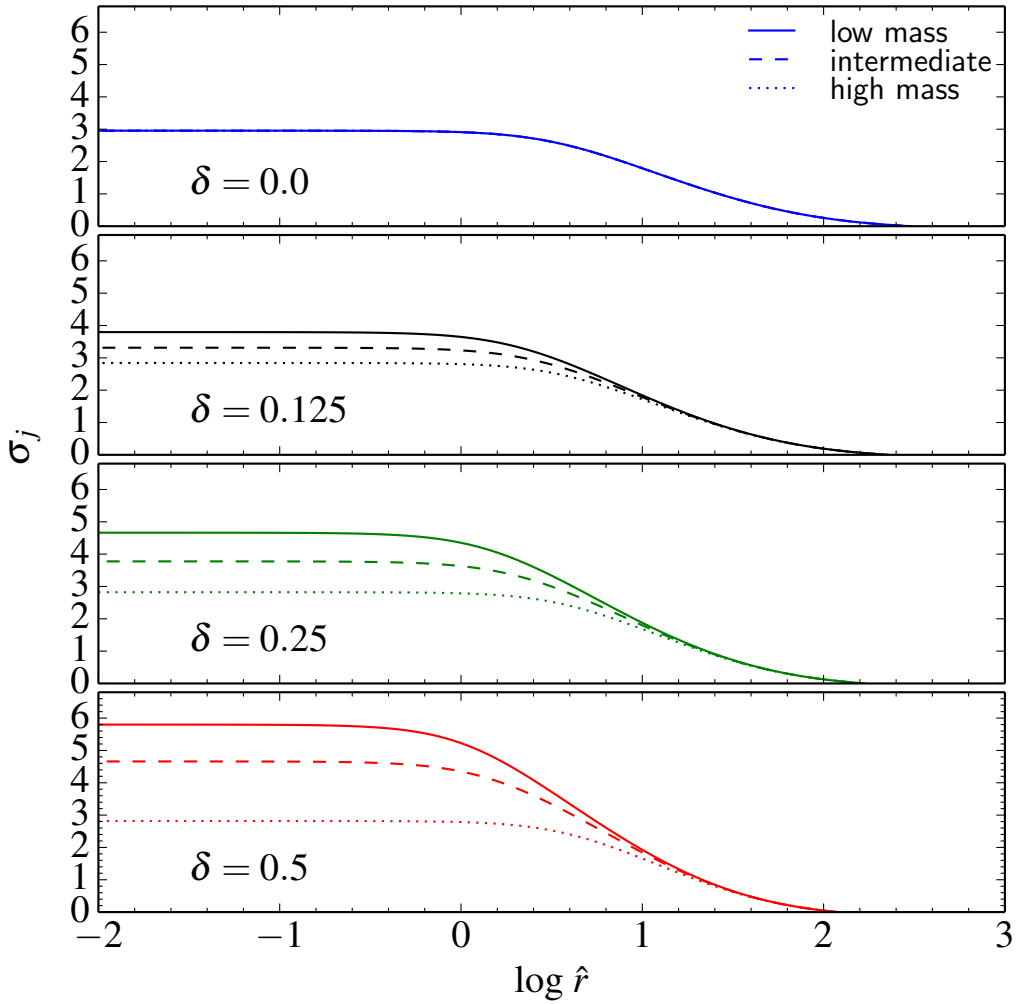
$$\hat{\phi}_0 = 7,\ g = 1$$





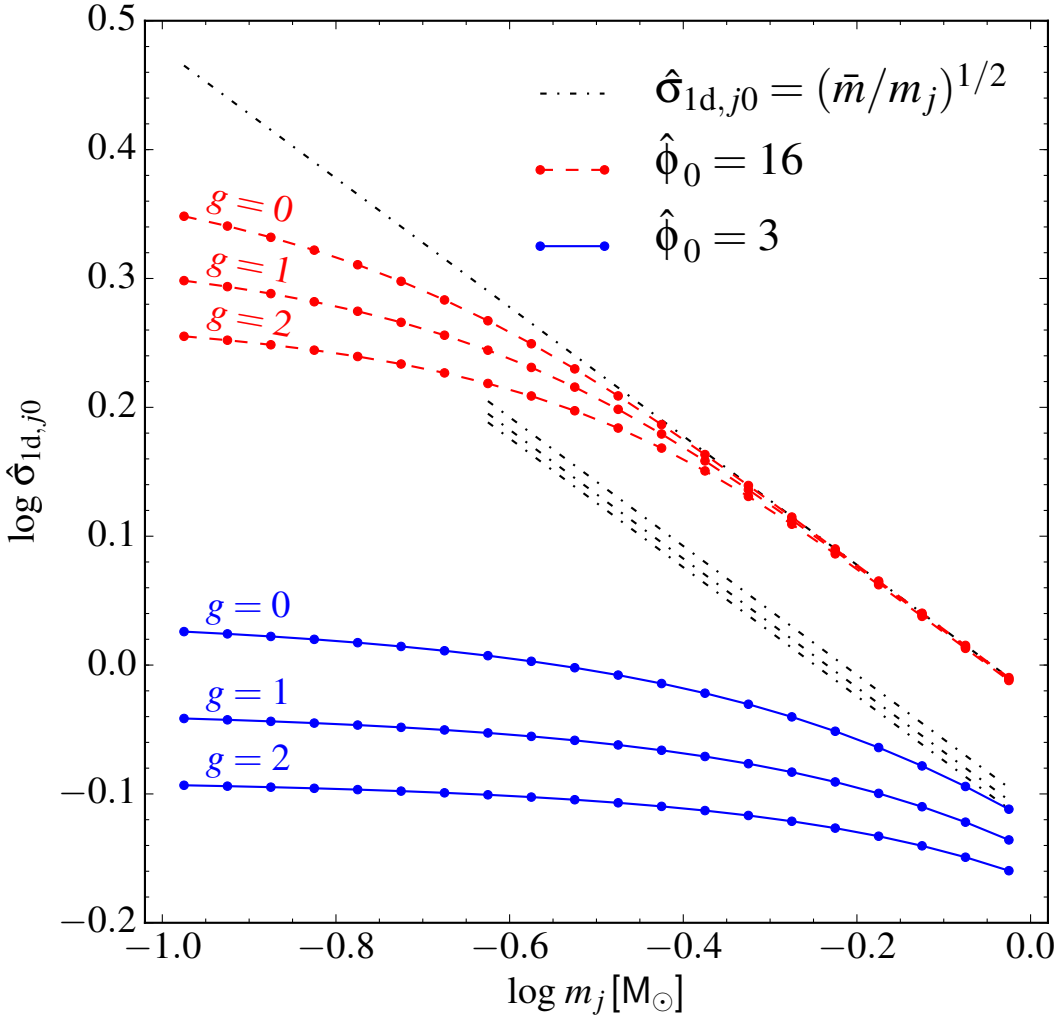
LIMEPY: Multi-mass

Some properties of the models: the role of δ



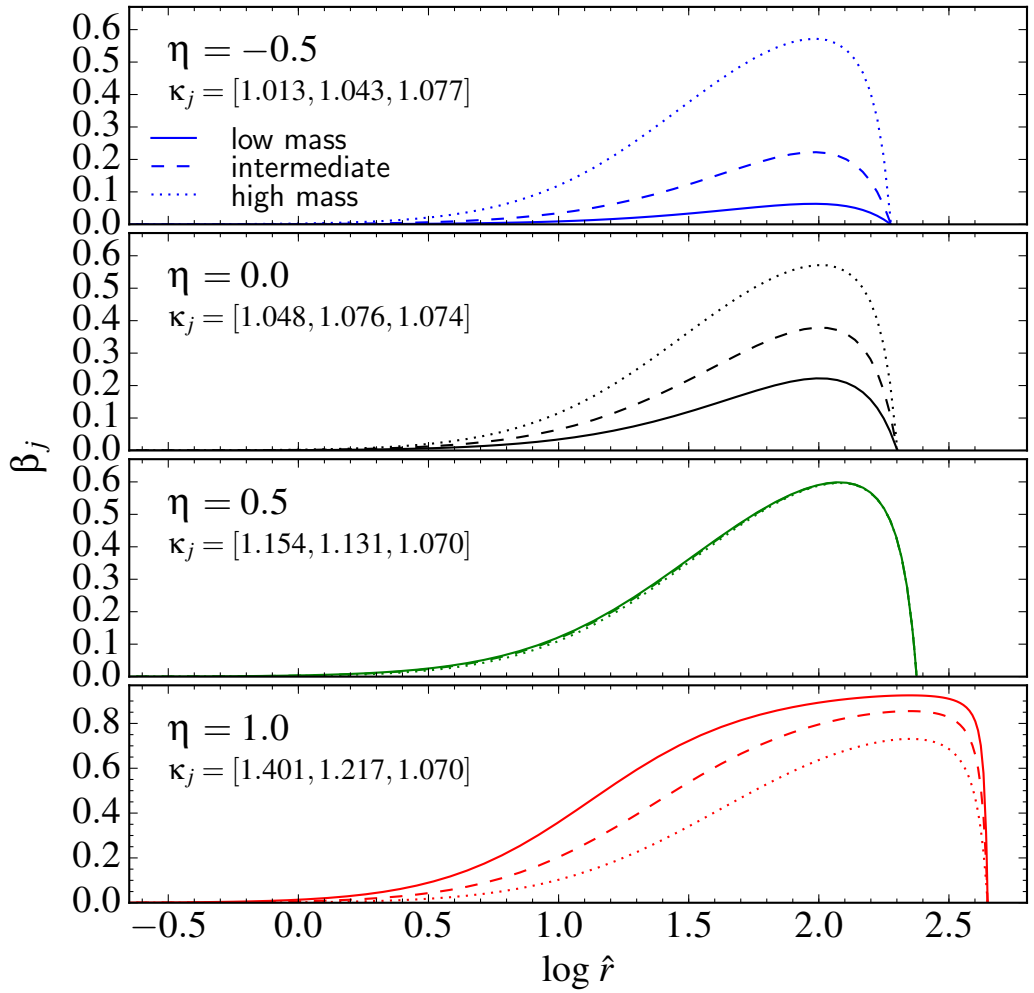
LIMEPY: Multi-mass

Some properties of the models: the role of δ

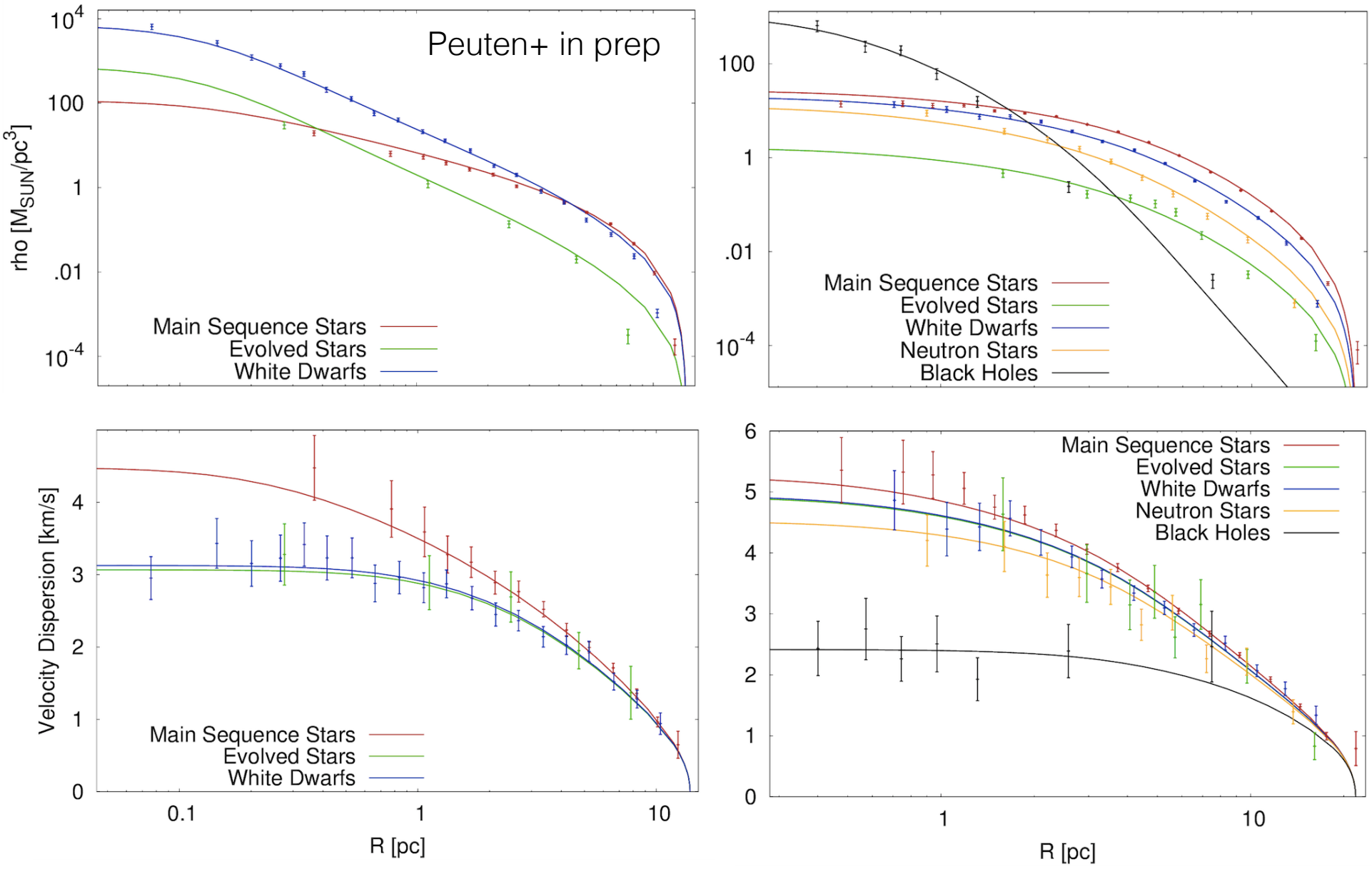


LIMEPY: Multi-mass

Some properties of the models: the role of η



Application: Challenge 2

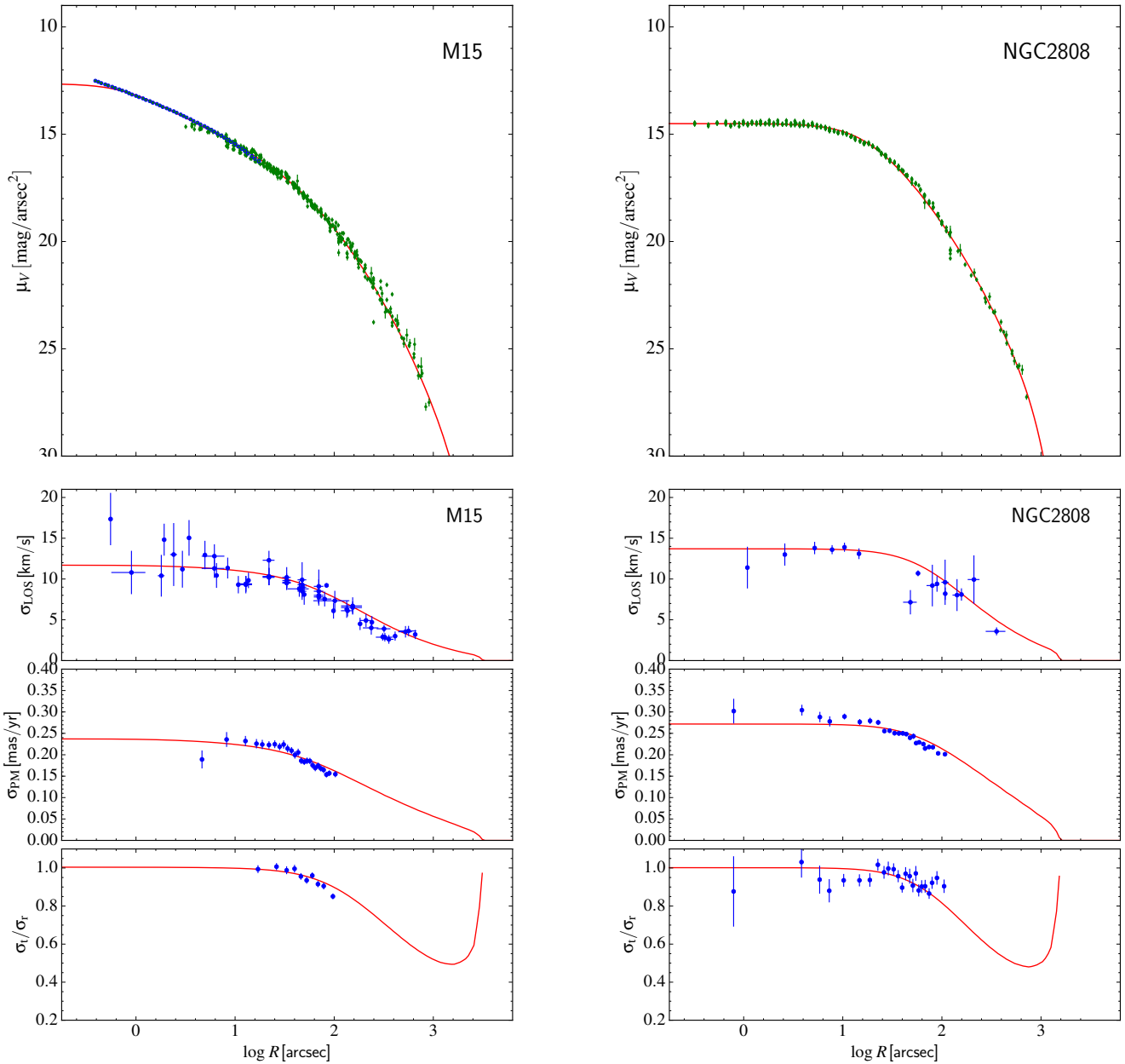


Application: M15 + NGC2808

Surface brightness:

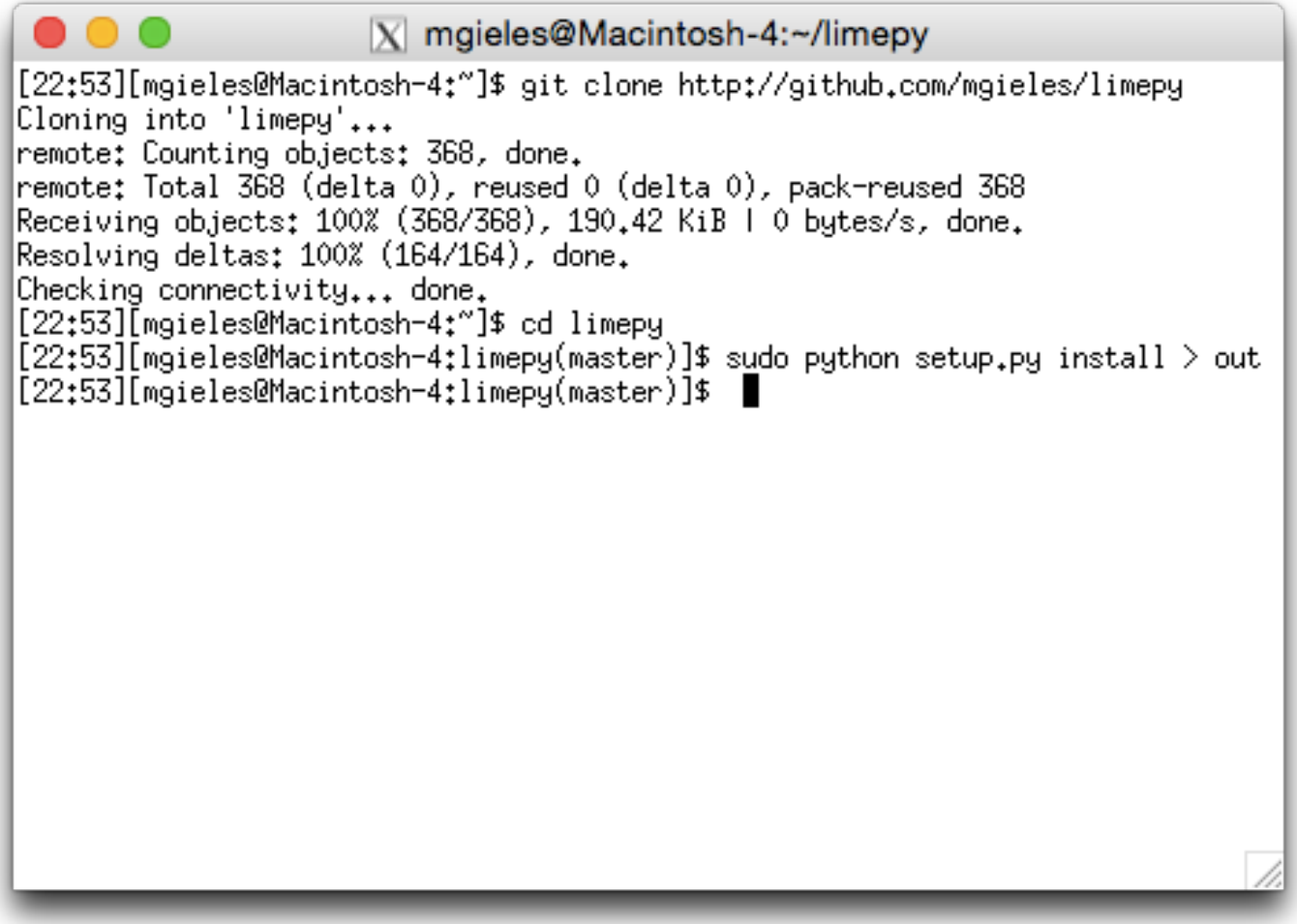
Radial velocities:

Proper motions



Gieles+, in prep

Install


mgieles@Macintosh-4:~/limepy
[22:53][mgieles@Macintosh-4:~]$ git clone http://github.com/mgieles/limepy
Cloning into 'limepy'...
remote: Counting objects: 368, done.
remote: Total 368 (delta 0), reused 0 (delta 0), pack-reused 368
Receiving objects: 100% (368/368), 190.42 KiB | 0 bytes/s, done.
Resolving deltas: 100% (164/164), done.
Checking connectivity... done.
[22:53][mgieles@Macintosh-4:~]$ cd limepy
[22:53][mgieles@Macintosh-4:limepy(master)]$ sudo python setup.py install > out
[22:53][mgieles@Macintosh-4:limepy(master)]$

>>> help(limepy)

mgieles@ub246072:~

```
Help on class limepy in module limepy.limepy:

class limepy
 |  Methods defined here:
 |
 |  __init__(self, phi0, g, **kwargs)
 |      (MM, A) LIMEPY
 |
 |      (Multi-Mass, Anisotropic) Lowered Isothermal Model Explorer in Python
 |
 |      This code solves the models presented in Gieles & Zocchi 2015 (GZ15),
 |      and calculates radial profiles for some useful quantities. The models
 |      are defined by the distribution function (DF) of equation (1) in GZ15.
 |
 |      Model parameters:
 |      =================
 |
 |      phi0 : scalar, required
 |           Central dimensionless potential
 |      g : scalar, required
 |        Order of truncation (0<= g < 3.5; 0=Woolley, 1=King, 2=Wilson)
 |
 |      ra : scalar, required for anisotropic models
 |        Anisotropy radius; default=1e8
 |      mj : list, required for multi-mass system
 |        Mean mass of each component; default=None
 |      Mj : list, required for multi-mass system
 |         Total mass of each component; default=None
 |      delta : scalar, optional
 |            Index in s_j = s x mu_j^-delta; default=0.5
 |            See equation (24) in GZ15
 |      eta : scalar, optional
 |            Index in ra_j = ra x mu_j^eta; default=0
 |            See equation (25) in GZ15
 |
 |      Input for scaling:
 |      ==================
 |
 |      scale : bool, optional
 |         Scale model to desired G=GS, M=MS, R=RS; default=False
 |      MS : scalar, optional
 |         Final scaled mass; default=10^5 [Msun]
 |      RS : scalar, optional
 |         Final scaled radius, half-mass or virial (see below); default=3 [pc]
 |      GS : scalar, optional
 |         Final scaled gravitationsl const; default=0.004302 [(km/s)^2 pc/Msun]
 |      scale_radius : str, optional
 |                   Radius to scale ['rv' or 'rh']; default='rh'
 |
 |      Options:
 |      ========
 |
 |      project : bool, optional
 |             Compute model properties in projection; default=False
 |      potonly : bool, optional
 |             Fast solution by solving potential only; default=False
 |      max_step : scalar, optional
 |              Maximum step size for ode output; default=1e10
 |      verbose : bool, optional
 |             Print diagnostics; default=False
 |      ode_atol : absolute tolerance parameter for ode solver; default=1e-7
 |      ode_rtol : relative tolerance parameter for ode solver; default=1e-7
:
```

mgieles@ub246072:~

```
 |      Output variables:
 |      =================
 |
 |      All models:
 |      -----------
 |       rhat, phihat, rhohat : radius, potential and density in model units
 |       r, phi, rho : as above, in scaled units (if scale=True)
 |       v2, v2r, v2t : total, radial and tangential mean-square velocity
 |       beta : anisotropy profile (equation 32, GZ15)
 |       mc : enclosed mass profile
 |       r0, rh, rv, rt : radii (King, half-mass, virial, truncation)
 |       K, Kr, Kt : kinetic energy; total, radial, tangential
 |       U, Q : potential energy, virial ratio
 |       A : constant in DF (equation 1, GZ15)
 |       volume : phase-space volume occupied by model
 |       nstep : number of integration steps (depends on ode_rtol & ode_atol)
 |       converged : bool flag to indicate whether model was solved
 |
 |      Projected models:
 |      -----------------
 |       Sigma : surface (mass) density
 |       v2z : line-of-sight mean-square velocity
 |       v2R, v2T : radial and tangential component of mean-square velocity
 |                  on plane of the sky
 |
 |      Multi-mass models:
 |      ------------------
 |       Properties of each component:
 |       phi0j : dimensionless central potential
 |       rhohatj : dimensionless density
 |       rhoj : density profile
 |       v2j : mean-square velocity profile
 |       v2rj, v2tj : radial and tangential component of mean-square velocity
 |                    profile
 |       r0j, raj : radii (King, anisotropy)
 |       Kj : kinetic energy
 |       Krj, Ktj : radial/tangential component of kinetic energy
 |
 |      Projected multi-mass models:
 |      ----------------------------
 |       Properties of each component:
 |       Sigmaj : surface (mass) density
 |       v2zj : line-of-sight mean-square velocity profile
 |       v2Rj, v2Tj : radial and tangential component on the plane of the sky
 |                    of the mean-square velocity profile
 |
 |      Examples:
 |      =========
 |
 |      Construct a Woolley model with phi0 = 7 and print r_t/r_0 and r_v/r_h
 |
 |      >>> k = limepy(7, 0)
 |      >>> print k.rt/k.r0, k.rv/k.rh
 |      >>> 19.1293426415 1.17783663028
 |
 |      Construct a Michie-King model and print ratio of anisotropy radius over
 |      half-mass radius and the Polyachenko & Shukhman (1981) anisotropy
 |      parameter
 |
 |      >>> a = limepy(7, 1, ra=5)
 |      >>> print a.ra/a.rh, 2*a.Kr/a.Kt
 |      >>> 1.03377960149 1.36280949941
:
```

mgieles@ub246072:~

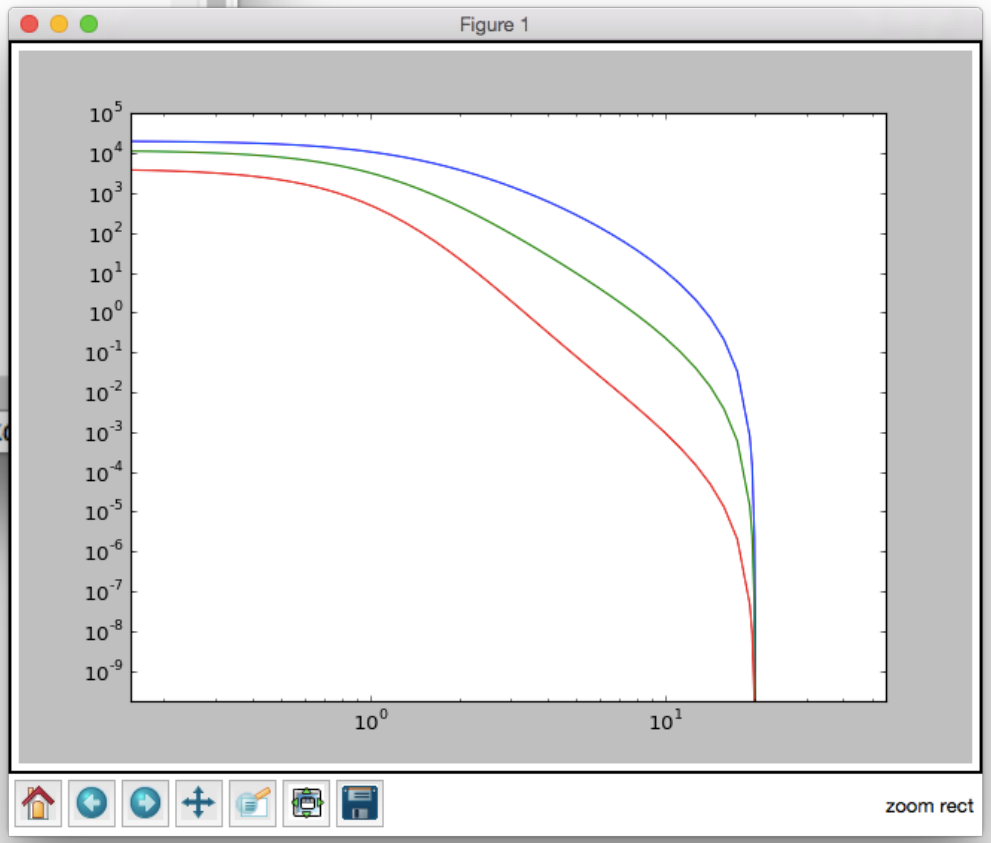
```
 |       v2rj, v2tj : radial and tangential component of mean-square velocity
 |                    profile
 |       r0j, raj : radii (King, anisotropy)
 |       Kj : kinetic energy
 |       Krj, Ktj : radial/tangential component of kinetic energy
 |
 |      Projected multi-mass models:
 |      ----------------------------
 |       Properties of each component:
 |       Sigmaj : surface (mass) density
 |       v2zj : line-of-sight mean-square velocity profile
 |       v2Rj, v2Tj : radial and tangential component on the plane of the sky
 |                    of the mean-square velocity profile
 |
 |      Examples:
 |      =========
 |
 |      Construct a Woolley model with phi0 = 7 and print r_t/r_0 and r_v/r_h
 |
 |      >>> k = limepy(7, 0)
 |      >>> print k.rt/k.r0, k.rv/k.rh
 |      >>> 19.1293426415 1.17783663028
 |
 |      Construct a Michie-King model and print ratio of anisotropy radius over
 |      half-mass radius and the Polyachenko & Shukhman (1981) anisotropy
 |      parameter
 |
 |      >>> a = limepy(7, 1, ra=5)
 |      >>> print a.ra/a.rh, 2*a.Kr/a.Kt
 |      >>> 1.03377960149 1.36280949941
 |
 |      Create a Wilson model with phi0 = 12 in Henon/N-body units: G = M =
 |      r_v = 1 and print the normalisation constant A of the DF and the
 |      value of the DF in the centre:
 |
 |      >>> w = limepy(12, 2, scale=True, GS=1, MS=1, RS=1, scale_radius='rv')
 |      >>> print w.A, w.df(0,0)
 |      >>> [ 0.00800902] [ 1303.40270676]
 |
 |      Multi-mass model in physical units with r_h = 3 pc and M = 10^5 M_sun
 |      and print central densities of each bin over the total central density
 |      and the half-mass radius + half-mass radius in projection
 |
 |      >>> m = limepy(7, 1, mj=[0.3,1,5], Mj=[9,3,1], scale=True, project=True)
 |      >>> print m.alpha, m.rh, m.rhp
 |      >>> [ 0.30721416  0.14103549  0.55175035] 3.0 2.25494426759
 |
 |  df(self, *arg)
 |      Returns the value of the normalised DF at a given position in phase
 |      space, can only be called after solving Poisson's equation
 |
 |      Arguments can be:
 |        - r, v                  (isotropic single-mass models)
 |        - r, v, j               (isotropic multi-mass models)
 |        - r, v, theta, j        (anisotropic models)
 |        - x, y, z, vx, vy, vz, j (all models)
 |
 |      Here j specifies the mass bin, j=0 for single mass
 |      Works with scalar and array input
 |
 |  interp_phi(self, r)
 |      Returns interpolated potential at r, works on scalar and arrays
(END)
```

Plot

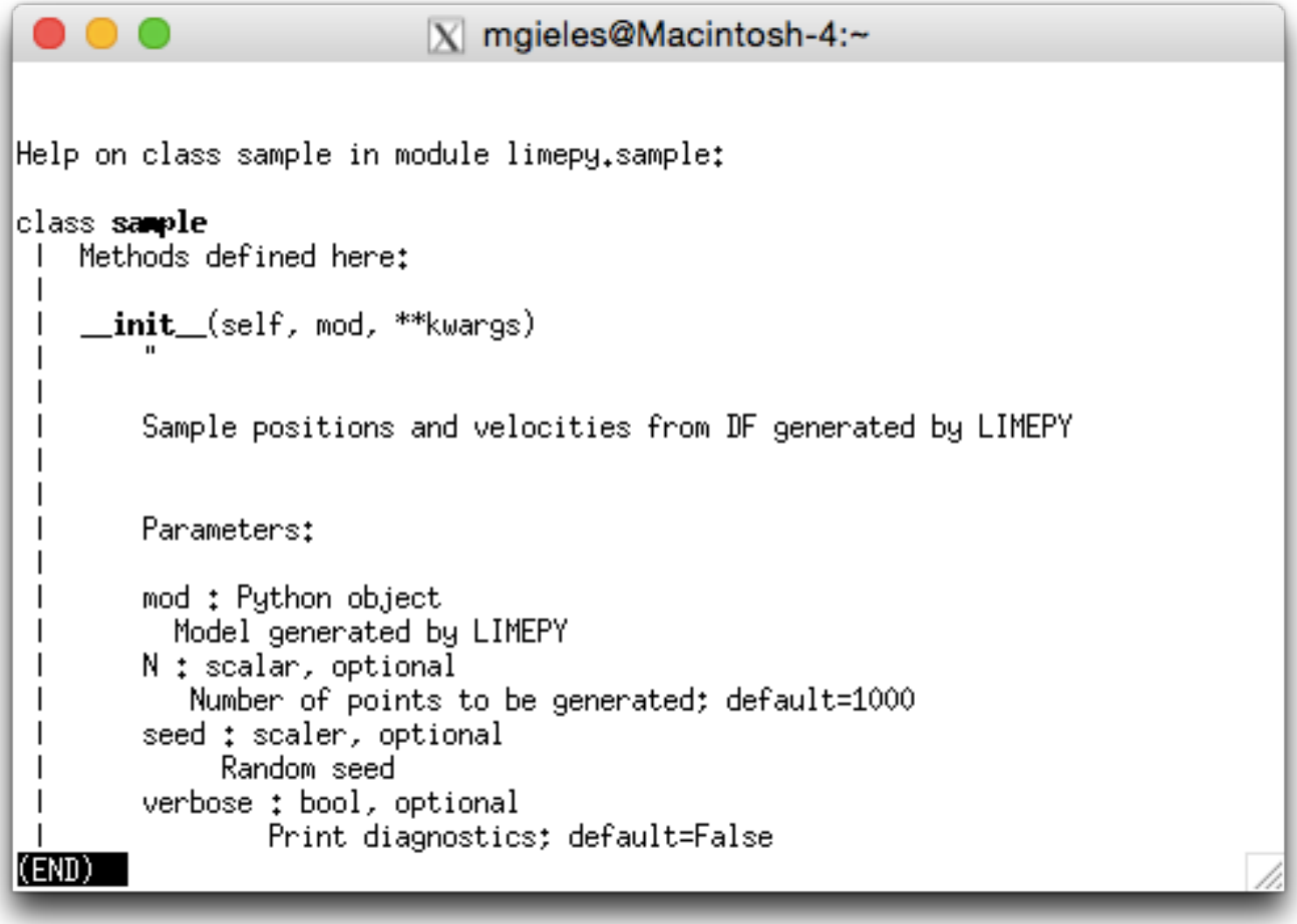
examplep.py

```
from __future__ import division
from limepy import limepy
import matplotlib.pylab as plt

plt.ion()
plt.clf()
plt.loglog()

phi0, g, ra = 7, 1.5, 20
mj, Mj      = [0.3, 0.8, 1.3], [1, 0.1, 0.01]
M, rh       = 1e6, 3
m = limepy(phi0, g, ra=ra, mj=mj, Mj=Mj, scale=True, MS=M, RS=rh, \
           scale_radius='rh', verbose=True)

for i in range(m.nmbin):
    plt.plot(m.r, m.rhoj[i])
```

```
-:---  examplep.py    All L12    (Python)
Wrote /Users/mgieles/Dropbox/projects/limepy/multi_iter/exc
```



>>> help(sample)


mgieles@Macintosh-4:~
Help on class sample in module limepy.sample:
class sample
| Methods defined here:
|
| __init__(self, mod, **kwargs)
| "
|
| Sample positions and velocities from DF generated by LIMEPY
|
|
| Parameters:
|
| mod : Python object
| Model generated by LIMEPY
| N : scalar, optional
| Number of points to be generated; default=1000
| seed : scaler, optional
| Random seed
| verbose : bool, optional
| Print diagnostics; default=False
(END)

Plot

examplep.py

```
from __future__ import division
from limepy import limepy, sample
import matplotlib.pylab as plt

plt.ion()
plt.clf()

phi0, g, ra = 7, 1.5, 20
mj, Mj      = [0.3, 0.8, 1.3], [1, 0.1, 0.01]
M, rh       = 1e4, 3
m = limepy(phi0, g, ra=ra, mj=mj, Mj=Mj, scale=True, MS=M, RS=rh, \
           scale_radius='rh', verbose=True)

ic = sample(m, seed=123)

plt.plot(ic.x, ic.y, '.')
plt.plot(ic.x[ic.Nstart[2]:ic.Nend[2]], ic.y[ic.Nstart[2]:ic.Nend[2]], 'r.')
```

-:--- examplep.py All L17 (Python)