

---

# TSC\_CM\_different\_prior

Unknown Author

February 3, 2014

Author: Karen Yin-Yee Ng

This program calculates the CM frame relic position constraints

```
In [37]: from __future__ import division
%autosave 60
%load_ext nbtoc
%nbtoc
from astropy import wcs
import astropy.coordinates as coord
import astropy.units as unit
from astropy.coordinates import Angle
import astropy.io.fits
from astropy.cosmology import FlatLambdaCDM
from astropy.coordinates import ICRS
import numpy as np

import time
import numpy.random as rand
import pdb
import pandas as pd
import sys
tstart = time.time()
#sys.path.append('/afs/sapphire.physics.ucdavis.edu/home/karenyng/Documents/Research_c
```

Autosaving every 60 seconds

The nbtoc extension is already loaded. To reload it, use:

```
%reload_ext nbtoc
```

```
In [38]: from wcs_ICRS import wcs_ICRS as WICRS
```

load homebrew modules

```
In [39]: from plotmod_dict import *
import cosmo
```

## Part I

# Initialization !!!!!

```
In [40]: #---initializations-----
data_path = "/Users/karenyng/Documents/Research/code"+\
            "/TSM/mercury_elGo/Feb_data/"
prefix = data_path + "ElGordo_"
oprefix = data_path + "polar_"
index = [ str(i) for i in range(20) ]
SE_centroid = ['01h02m38.38s', '-49d16m37.64s']
NW_relic = ICRS('01h02m46s -49d14m43s',
                unit=(unit.degree, unit.degree))
SE_relic = ICRS('01h03m01s -49d17m14s',
                unit=(unit.degree, unit.degree))

#Histogram bins
N_bins_2d = 130
N_bins_1d = 200
N_bins_TSM = 45
N_bins_alpha = 90

#specify the number of iterations
Iter = 500000
#Iter = 475000
par=['m_1',
     'm_2',
     'z_1',
     'z_2',
     'd_proj',
     'v_rad_obs',
     'alpha',
     'v_3d_obs',
     'd_3d',
     'v_3d_col',
     'd_max',
     'TSM_0',
     'TSM_1',
     'T',
     'prob']

fitspath = '/Users/karenyng/Documents/Research/code/'+\
            'ElGordo-Dynamics-Paper/Analysis/HSTlensing/'
fits = fitspath + 'header.fits'
getwcs = WICRS('01h03m01s -49d17m14s', fitsname=fits)
w = getwcs.wcs
```

load pickles into suitable formats

```
In [41]: data = load_pickles_to_df(par, prefix, index)
```

converting entry m\_1 to units of 1e14 m\_sun  
converting entry m\_2 to units of 1e14 m\_sun

calculate the separation of the CM from the two relics

```
In [42]: NW = ICRS('01h02m51.68s -49d15m04.40s')
SE = ICRS('01h02m38.38s -49d16m37.64s')
[NW_pix, SE_pix] = w.wcs_world2pix(np.array([[NW.ra.deg, NW.dec.deg],
                                             [SE.ra.deg, SE.dec.deg]]), 1)
```

Just use the mean of the mass blobs as the center of mass location

```
In [43]: NW_mean_mass = data['m_1'].mean()
SE_mean_mass = data['m_2'].mean()
cm_pix = (NW_pix * NW_mean_mass + SE_pix * SE_mean_mass) /\
(NW_mean_mass + SE_mean_mass)
[cm_wcs, junk] = w.wcs_pix2world(np.array([cm_pix, cm_pix]), 1)
cm_wcs = ICRS(ra=cm_wcs[0], dec=cm_wcs[1],
              unit=(unit.degree, unit.degree))
SE_relic_CM_sep = cm_wcs.separation(SE_relic)
NW_relic_CM_sep = cm_wcs.separation(NW_relic)
```

use  $\Lambda$ CDM

```
In [44]: cosmo = FlatLambdaCDM(H0=70, Om0=0.3)
r_proj_SE = cosmo.angular_diameter_distance(.87) * \
SE_relic_CM_sep.rad
r_proj_NW = cosmo.angular_diameter_distance(.87) * \
NW_relic_CM_sep.rad

print "SE relic separation from CM is {}".format(r_proj_SE)
print "NW relic separation from CM is {}".format(r_proj_NW)
```

```
SE relic separation from CM is 1.32742877372 Mpc
NW relic separation from CM is 0.47281510798 Mpc
```

Which is consistent with the picture given by Lindner et al. 's paper Fig 1.

## Part II

# How to do the relic calculation

## 1 Info from Lindner et al. 2013

“shock speed can be interpreted as an upper limit on the collision speed”

implies that this a relative speed between the two subclusters?

This is different than saying that the shock speed is in the CM frame.

### 1.1 the position of the center of mass and thus the separations of the relics to the CM

- using the full realizations takes up a lot of time for doing conversion between coordinates and calculating the physical separation since we have half a million realizations and the code is not vectorized - python loops are slow
- using the mean location of the center of mass which is a lot faster but will not fold the uncertainties in — maybe I should rewrite this part using Cython

## 1.2 the velocities of the relics

we do not know the time evolution of the speed but we can assume an average velocity to be either:

- the collisional velocity of the NW subcluster
- the collision velocity of the NW subcluster times a slow-down factor

In the center of mass frame, the speed of  $m_1$  and  $m_2$  are related by:

$$v_{2,CM} = -\frac{m_1}{m_2}v_{1,CM}$$

– (1)

but in our simulation, we only know about the relative velocities of the two subclusters,

$v_{rel} = v_1 - v_2$  – (2) which is true in any frame

Plug (1) in (2) to get

$$v_{1,CM} = \left( \frac{m_2}{m_1 + m_2} \right) v_{rel}$$
$$v_{2,CM} = - \left( \frac{m_1}{m_1 + m_2} \right) v_{rel}$$

## 1.3 Relic calculation - v.1 - just getting a feeling of how the distances compare

- using a mean location for the center of mass
- using NW  $v_{relic} = 4300 \text{ km/s} \pm 800 \text{ km/s}$  assuming this is in CM frame

Is this shock speed in 3D or 2D??? it makes a difference - should be in 3D... Calculation of the simulated separation for relic from the CM:

$$s_{relic} = v_{relic_i} t_{tsc_j} \cos(\alpha)$$

There will be 2 outputs from the calculations just from simulation since there are two TSCs. Unit conversion is done to convert units of (km / s \* Gyr) to Mpc

$$\text{Mpc} / \text{km} = 60 \times 60 \times 365 \times 24 \times 1e9 / (1e13 \times 1e6 \times 3.086)$$

## 1.4 Initialize the velocities

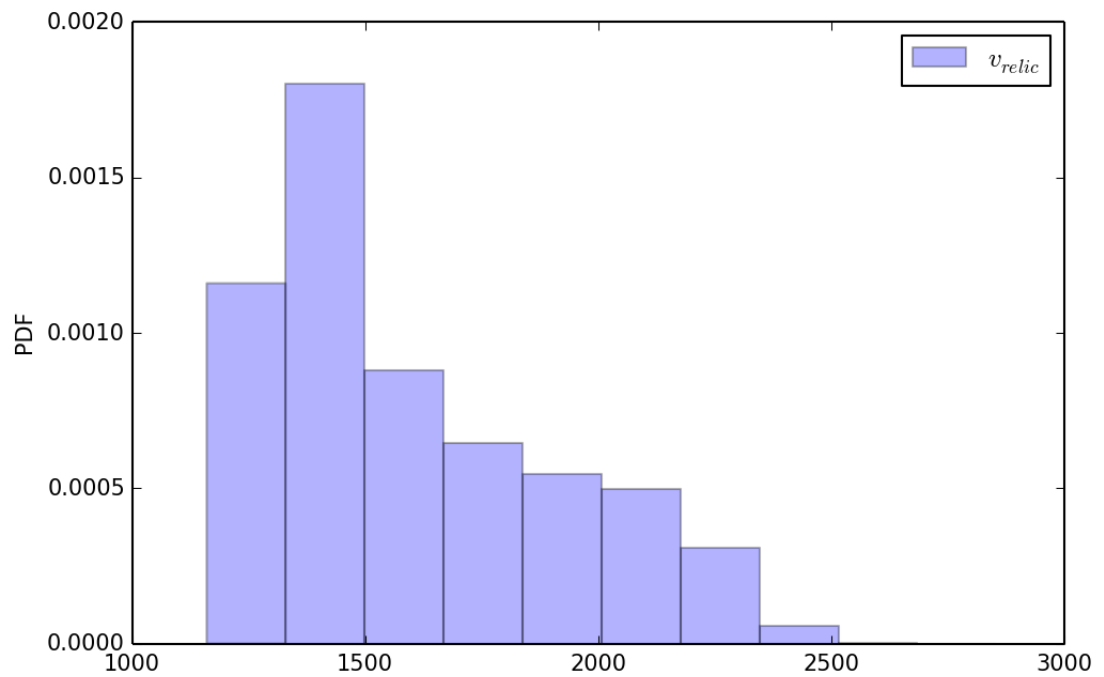
```
In [45]: data['v_relic'] = data['v_3d_col'] * SE_mean_mass /\n          (NW_mean_mass + SE_mean_mass)
```

## 1.5 See if the speed distribution looks alright

```
In [46]: j1, j2, j3 = plt.hist(data['v_relic'], histtype = 'bar',\n                               label = r'$v_{relic}$', alpha=0.3, normed=True)\nplt.ylabel('PDF')\nplt.legend(loc = 'best')
```

Out [46]:

<matplotlib.legend.Legend at 0x10e13cf90>



## 1.6 Initialize the lower bound of the predicted location

$$x_{NW} = \frac{m_{SE}}{m_{NW}} x_{SE}$$

$$x_{rel} = x_{NW} - x_{SE}$$

$$= x_{NW} - x_{SE}$$

$$x_{NW} = \frac{m_{SE}}{m_{NW} + m_{SE}} x_{rel}$$

```
In [47]: # convert m to Mpc , Myrs to seconds
unitconversion = 60 * 60 * 365 * 24 * 1e9 / (1e13 * 1e6 * 3.086)

data['r_proj0_min'] = data['d_3d'] * \
    np.cos(data['alpha'] / 180. * np.pi) * \
    (SE_mean_mass) / (SE_mean_mass + NW_mean_mass)
data['r_proj1_min'] = data['d_3d'] * \
    np.cos(data['alpha'] / 180. * np.pi) * \
    (SE_mean_mass) / (SE_mean_mass + NW_mean_mass)

data = data[~np.isnan(data['r_proj0_min'])]
data = data[~np.isnan(data['r_proj1_min'])]
```

## 1.7 Initialize the upper bound of the predicted location

```
In [48]: data['r_proj0_max'] = (data['v_relic'] * data['TSM_0'] *
                                np.cos(data['alpha'] / 180*np.pi)) * unitconversion
data['r_proj1_max'] = (data['v_relic'] * data['TSM_1'] *
                        np.cos(data['alpha'] / 180*np.pi)) * unitconversion
data = data[~np.isnan(data['r_proj0_max'])]
data = data[~np.isnan(data['r_proj1_max'])]
```

## 1.8 Include the uncertainty of the width of the relic

```
In [49]: rupper = r_proj_NW.value + 23/2/1000
rlower = r_proj_NW.value - 23/2/1000
print "range is {0}, {1}".format(rupper, rlower)
```

range is 0.48431510798, 0.48431510798

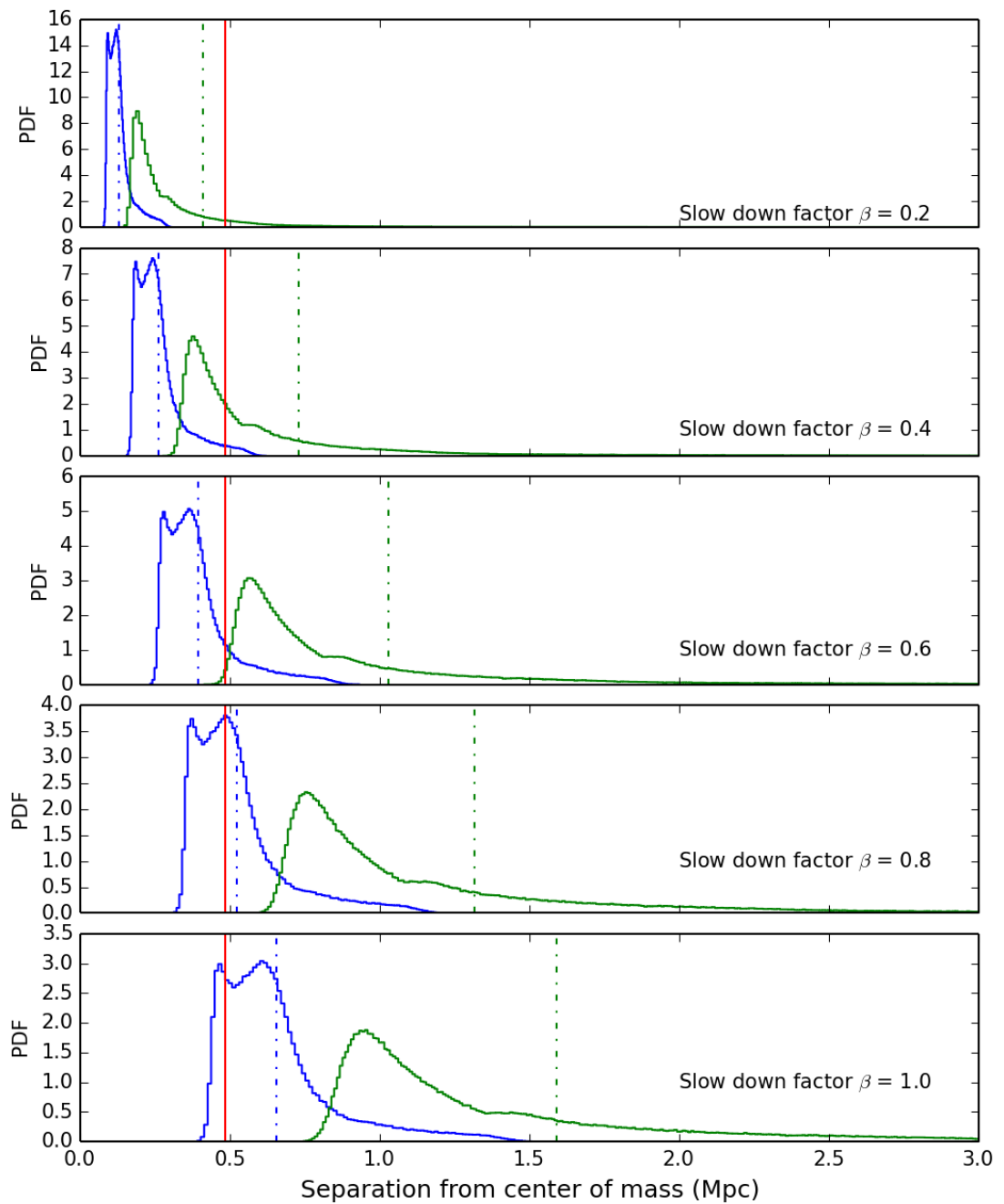
```
In [50]: def make_hist(this_ax, data1, data2, data1_mean, data2_mean, beta, i):
    j1, j2, j3 = this_ax.hist(data1,
                              histtype='step', bins=100, label=r'$s-min_0$',
                              normed=True)
    j1, j2, j3 = this_ax.hist(data2,
                              histtype='step', label=r'$s-min_1$',
                              bins = 1000, normed = True)
    this_ax.axvline(data1_mean, color='blue', ls='-.', label=r'$<s-min_0>$')
    this_ax.axvline(data2_mean, color='green', ls='-.', label=r'$<s-min_1>$')
    this_ax.axvspan(rlower, rupper, color='red', label=r'true $r_{proj}$')
    this_ax.set_xlim(0, 3)
    this_ax.set_ylabel('PDF')
    this_ax.text(2, 1, r"Slow down factor $\beta$ = {0}".format(beta),
                 va='center')
```

## 1.9 Create plot with varying beta ratio

```
In [56]: beta = np.arange(0.2, 1.1, 0.2)
f, ax = plt.subplots(beta.size, sharex=True)

for i in range(beta.size):
    # initialize the different ratio of velocities
    temp1 = data['v_3d_col'] * data['TSM_0'] * \
            np.cos(data['alpha'] / 180 * np.pi) * beta[i] * \
            unitconversion * SE_mean_mass / \
            (SE_mean_mass + NW_mean_mass)
    data1 = temp1[temp1 < 10]
    data1_mean = np.mean(data1)
    temp2 = data['v_3d_col'] * data['TSM_1'] * \
            np.cos(data['alpha'] / 180 * np.pi) * beta[i] * \
            unitconversion * SE_mean_mass / \
            (SE_mean_mass + NW_mean_mass)
    data2 = temp2[temp2 < 10]
    data2_mean = np.mean(data2)
    make_hist(ax[i], data1, data2, data1_mean, data2_mean, beta[i], i)

f.subplots_adjust(hspace=0.1)
f.set_size_inches(8, beta.size * 2)
plt.xlabel('Separation from center of mass (Mpc)', size = 12)
plt.savefig("default_prior_bounds.pdf", bbox_inches="tight")
```



turns out we need to use proper bin size !Let us zoom in on the part where the relic position is

### Part III

## Apply radio relic polarization prior

```
In [52]: radiomask1, count1 = radio_polar_prior(data['alpha'])
radiomask = radiomask1
r_data= data.copy(deep=True)
r_data = r_data[radiomask]
print 'length of data after applying prior is'+\
      '{0}'.format(len(r_data['d_3d']))
```

length of data after applying prior is 268971

## 1.10 Initialize the speeds for the data with polarization data

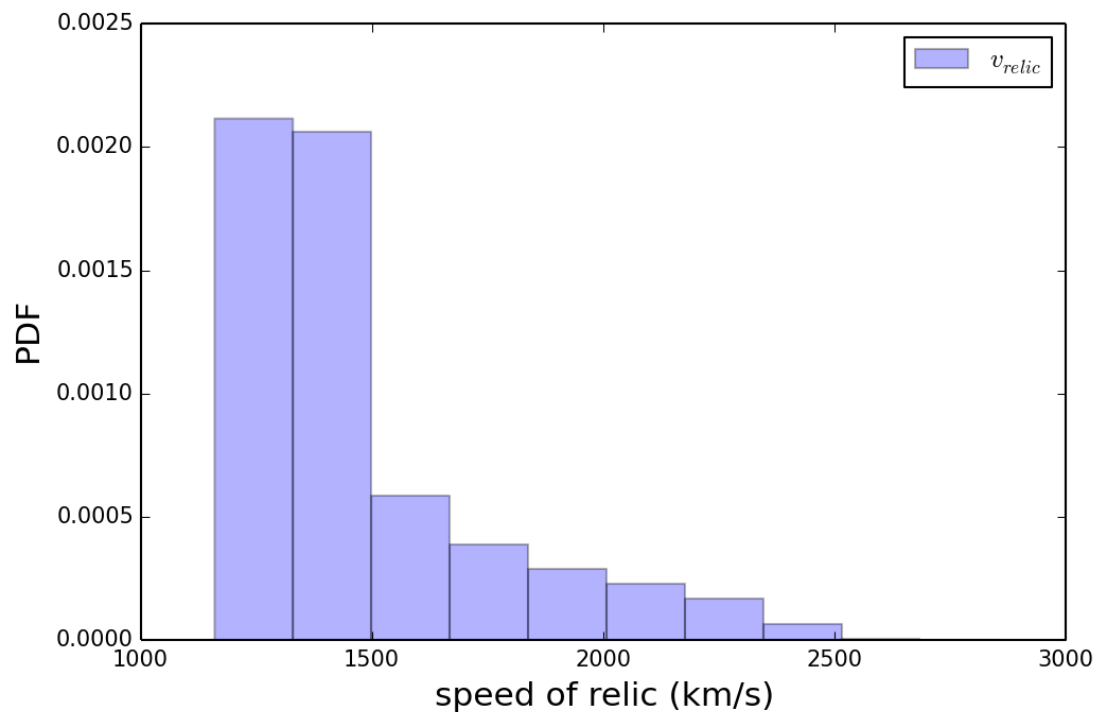
```
In [53]: r_data['v_relic'] = r_data['v_3d_col'] * SE_mean_mass / \
      (SE_mean_mass + NW_mean_mass)
```

## 1.11 See if the speed distribution looks alright

```
In [35]: j1, j2, j3 = plt.hist(r_data['v_relic'], histtype = 'bar',
      label = r'$v_{relic}$', alpha=0.3, normed=True)

plt.xlabel('speed of relic (km/s)', size=15)
plt.ylabel('PDF', size=15)
plt.legend(loc = 'best')
```

Out [35]:  
<matplotlib.legend.Legend at 0x104daled0>



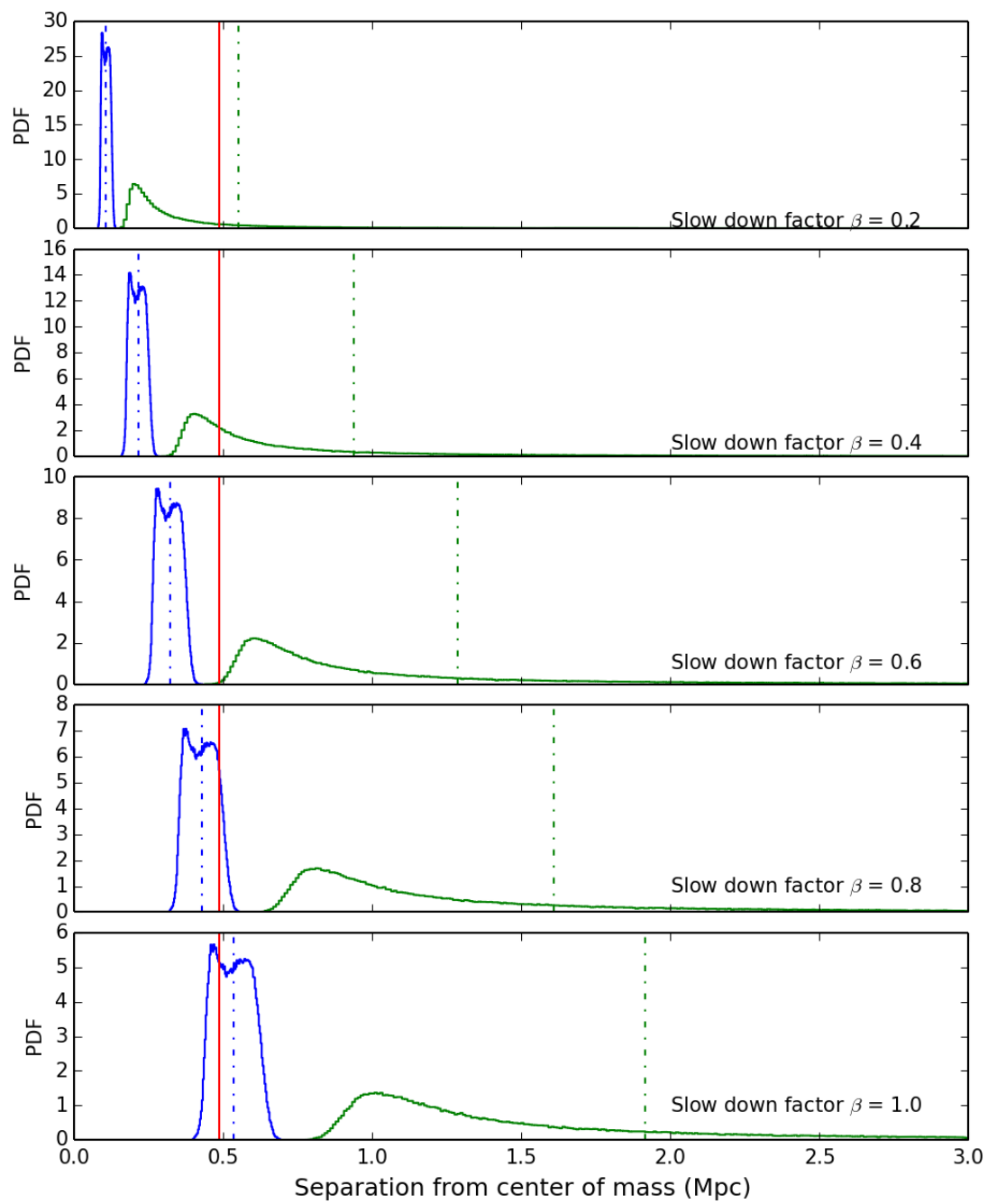


## 1.12 Calculate the predicted separation

```
In [55]: beta = np.arange(0.2, 1.1, 0.2)
f, ax = plt.subplots(beta.size, sharex=True)

for i in range(beta.size):
    # initialize the different ratio of velocities
    temp1 = r_data['v_3d_col'] * r_data['TSM_0'] * \
        np.cos(r_data['alpha'] / 180 * np.pi) * beta[i] * \
        unitconversion * SE_mean_mass /\
        (SE_mean_mass + NW_mean_mass)
    r_data1 = temp1[temp1 < 10]
    r_data1_mean = np.mean(r_data1)
    temp2 = r_data['v_3d_col'] * r_data['TSM_1'] * \
        np.cos(r_data['alpha'] / 180 * np.pi) * beta[i] * \
        unitconversion * SE_mean_mass /\
        (SE_mean_mass + NW_mean_mass)
    r_data2 = temp2[temp2 < 10]
    r_data2_mean = np.mean(r_data2)
    make_hist(ax[i], r_data1, r_data2, r_data1_mean,
        r_data2_mean, beta[i], i)

f.subplots_adjust(hspace=0.1)
f.set_size_inches(8, beta.size * 2)
plt.xlabel('Separation from center of mass (Mpc)', size = 12)
plt.savefig("polar_prior_bounds.pdf", bbox_inches="tight")
```



## Sort the different arrays to get the extreme values as bounds

```
In [198]: data.columns
```

```
Index([u'm_1', u'm_2', u'z_1', u'z_2', u'd_proj', u'v_rad_obs',
u'alpha', u'v_3d_obs', u'd_3d', u'v_3d_col', u'd_max', u'TSM_0',
u'TSM_1', u'T', u'prob', u'v_relic', u'r_proj0_min', u'r_proj1_min',
u'r_proj0_max', u'r_proj1_max'], dtype='object')
```

true location of relic from CM is  
0.47 Mpc

the bounds with default priors for outgoing scenario is  $0.37 \text{ Mpc} < r_{\text{relic}} < 1.56 \text{ Mpc}$

the bounds with default priors for incoming scenario is  $0.37 \text{ Mpc} < r_{\text{relic}} < 43616847.30 \text{ Mpc}$

the bounds with polarization priors for outgoing scenario is  $0.34 \text{ Mpc} < r_{\text{relic}} < 0.72 \text{ Mpc}$

\_\_\_\_\_

the bounds with polarization priors for incoming scenario is  
 $0.26 \text{ Mpc} < r_{\text{relic}} < 43616847.30 \text{ Mpc}$

```
In [204]: """
r_data['r_proj0_v2'] = (r_data['v_avg'] * r_data['TSM_0'] *
                        np.cos(r_data['alpha'] / 180*np.pi)) * unitconversion * \
                        (SE_mean_mass)/(NW_mean_mass + SE_mean_mass)
r_data['r_proj1_v2'] = (r_data['v_avg'] * r_data['TSM_1'] *
                        np.cos(r_data['alpha'] / 180*np.pi)) * unitconversion * \
                        (SE_mean_mass)/(NW_mean_mass + SE_mean_mass)

j1, j2, j3 = plt.hist(r_data['r_proj0_v2'][r_data['r_proj0_v2'] < 10],
                      histtype='step', bins=100, label=r'$r_{\text{min}_0}$',
                      normed=True)
j1, j2, j3 = plt.hist(r_data['r_proj1_v2'][r_data['r_proj1_v2'] < 10],
                      histtype='step', label=r'$r_{\text{min}_1}$',
                      bins = 100, normed = True)
plt.axvspan(rlower, rupper, color='red', label=r'true $r_{\text{proj}}$')
#plt.title(r'Projected relic location assuming'+\
#         '$v_{\text{relic}}$ \sim $v_{3D}(t_{\text{avg}})$')

plt.xlim(0, 2)
plt.ylabel('PDF', size=15)
plt.xlabel('Separation from center of mass (Mpc)', size=15)
plt.axvline(r_data['r_proj0_v2'][r_data['r_proj0_v2'] < 10].mean(),
            color='blue', ls='-.', label=r'$<r_{\text{min}_0}>$')
plt.axvline(r_data['r_proj1_v2'][r_data['r_proj1_v2'] < 10].mean(),
            color='green',
            ls='-.', label=r'$<r_{\text{min}_1}>$')
plt.legend(loc='upper right')
"""
```

```
Out [204]: "\nr_data['r_proj0_v2'] = (r_data['v_avg'] * r_data['TSM_0'] * \n
np.cos(r_data['alpha'] / 180*np.pi)) * unitconversion *
(SE_mean_mass)/(NW_mean_mass + SE_mean_mass)\nr_data['r_proj1_v2'] =
(r_data['v_avg'] * r_data['TSM_1'] *\n
np.cos(r_data['alpha'] / 180*np.pi)) * unitconversion *
(SE_mean_mass)/(NW_mean_mass + SE_mean_mass)\n\nj1, j2, j3 =
plt.hist(r_data['r_proj0_v2'][r_data['r_proj0_v2'] < 10], \n
histtype='step', bins=100, label=r'$r_{\text{min}_0}$', \n
normed=True)\nj1, j2, j3 =
plt.hist(r_data['r_proj1_v2'][r_data['r_proj1_v2'] < 10], \n
histtype='step', label=r'$r_{\text{min}_1}$', \n
bins =
100, normed = True)\nplt.axvspan(rlower, rupper, color='red',
label=r'true $r_{\text{proj}}$')\n#plt.title(r'Projected relic location
assuming'+#
'$v_{\text{relic}}$ \sim $v_{3D}(t_{\text{avg}})$')
')\n\nplt.xlim(0, 2)\nplt.ylabel('PDF',
size=15)\nplt.xlabel('Separation from center of mass (Mpc)',
size=15)\nplt.axvline(r_data['r_proj0_v2'][r_data['r_proj0_v2'] <
10].mean(), \n
color='blue', ls='-.', label=r'$<r_{\text{min}_0}>$')
\nplt.axvline(r_data['r_proj1_v2'][r_data['r_proj1_v2'] <
10].mean(), \n
color='green', \n
ls='-.', label=r'$<r_{\text{min}_1}>$')\nplt.legend(loc='upper right')\n"
```

```
In [205]: """
r_data['r_proj0_v2'] = (r_data['v_relic'] * r_data['TSM_0'] *
np.cos(r_data['alpha'] / 180*np.pi)) * unitconversion * \
(SE_mean_mass)/(NW_mean_mass + SE_mean_mass)
r_data['r_proj1_v2'] = (r_data['v_relic'] * r_data['TSM_1'] *
np.cos(r_data['alpha'] / 180*np.pi)) * unitconversion * \
(SE_mean_mass)/(NW_mean_mass + SE_mean_mass)

j1, j2, j3 = plt.hist(r_data['r_proj0_v2'][r_data['r_proj0_v2'] < 10],
histtype='step', bins=100, label=r'$r_{\max_0}$',
normed=True)
j1, j2, j3 = plt.hist(r_data['r_proj1_v2'][r_data['r_proj1_v2'] < 10],
histtype='step', label=r'$r_{\max_1}$',
bins=100, normed=True)

plt.xlim(0, 2.0)
plt.ylabel('PDF', size=15)
plt.xlabel('Separation from center of mass (Mpc)', size=15)
plt.axvspan(rlower, rupper, color='red', label=r'true $r_{\text{proj}}$')
plt.axvline(r_data['r_proj0_v2'][r_data['r_proj0_v2'] < 10].mean(),
color='blue', ls='-.', label=r'$<r_{\max_0}>$')
plt.axvline(r_data['r_proj1_v2'][r_data['r_proj1_v2'] < 10].mean(),
color='green', ls='-.', label=r'$<r_{\max_1}>$')
plt.legend(loc='best')
"""
```

```
Out [205]:
"\nr_data['r_proj0_v2'] = (r_data['v_relic'] * r_data['TSM_0'] * \n
np.cos(r_data['alpha'] / 180*np.pi)) * unitconversion *
(SE_mean_mass)/(NW_mean_mass + SE_mean_mass)\nr_data['r_proj1_v2'] =
(r_data['v_relic'] * r_data['TSM_1'] *\n
np.cos(r_data['alpha'] / 180*np.pi)) * unitconversion *
(SE_mean_mass)/(NW_mean_mass + SE_mean_mass)\n\nj1, j2, j3 =
plt.hist(r_data['r_proj0_v2'][r_data['r_proj0_v2'] < 10], \n
histtype='step', bins=100, label=r'$r_{\max_0}$',\n
normed=True)\nj1, j2, j3 =
plt.hist(r_data['r_proj1_v2'][r_data['r_proj1_v2'] < 10], \n
histtype='step', label=r'$r_{\max_1}$',\n
bins=100, normed=True)\n\nplt.xlim(0, 2.0)\nplt.ylabel('PDF',
size=15)\nplt.xlabel('Separation from center of mass (Mpc)',
size=15)\nplt.axvspan(rlower, rupper, color='red', label=r'true
$r_{\text{proj}}$')\nplt.axvline(r_data['r_proj0_v2'][r_data['r_proj0_v2'] <
10].mean(),\n
color='blue', ls='-.', label=r'$<r_{\max_0}>$')
)\nplt.axvline(r_data['r_proj1_v2'][r_data['r_proj1_v2'] <
10].mean(),\n
color='green', ls='-.',
label=r'$<r_{\max_1}>$')\nplt.legend(loc='best')\n"
```