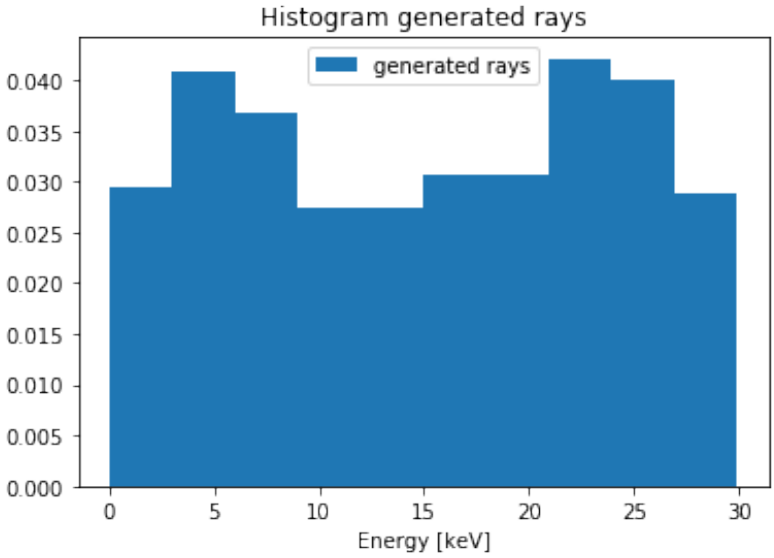
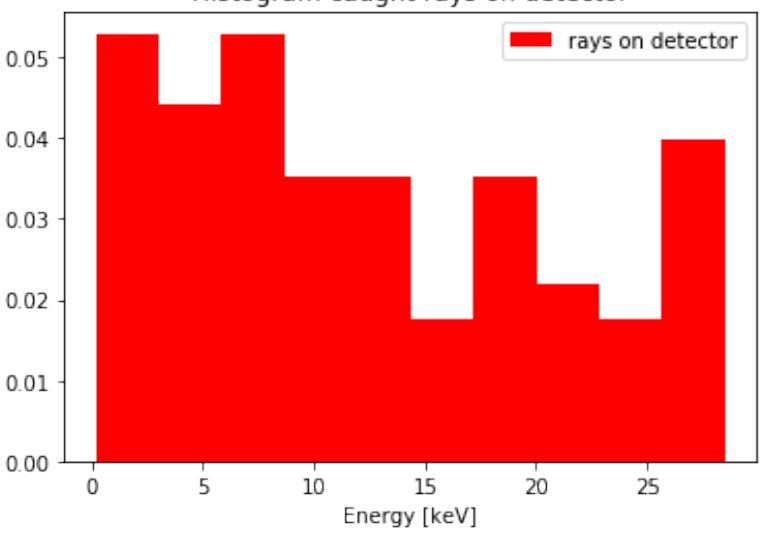
FOXSISIM

Energy response for a 2-shell Wolter-I figure for different focal lengths.

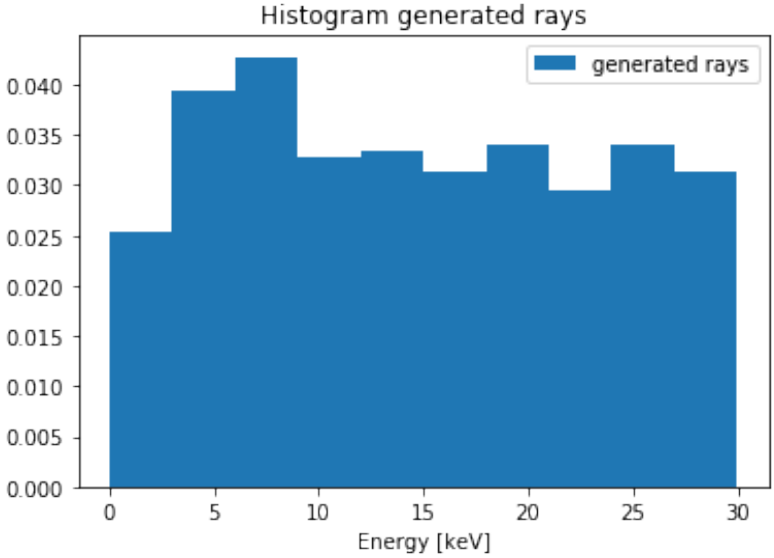
Milo - Jun, 2017

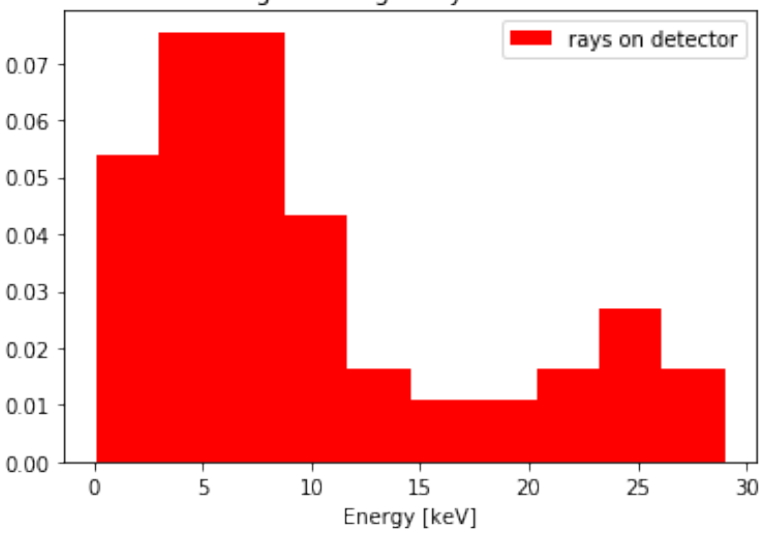
F = 1500



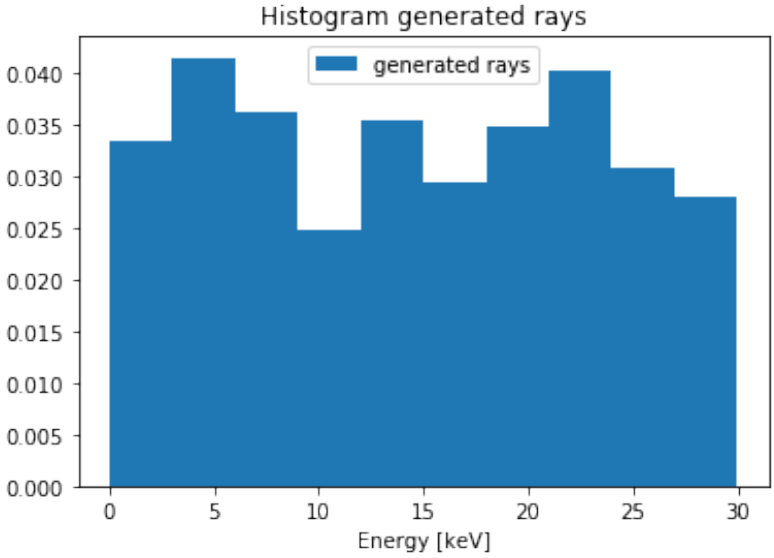


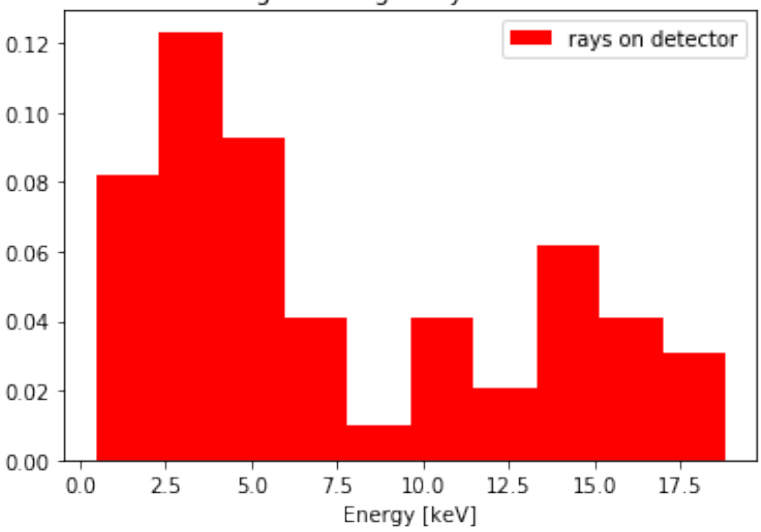
F = 1000



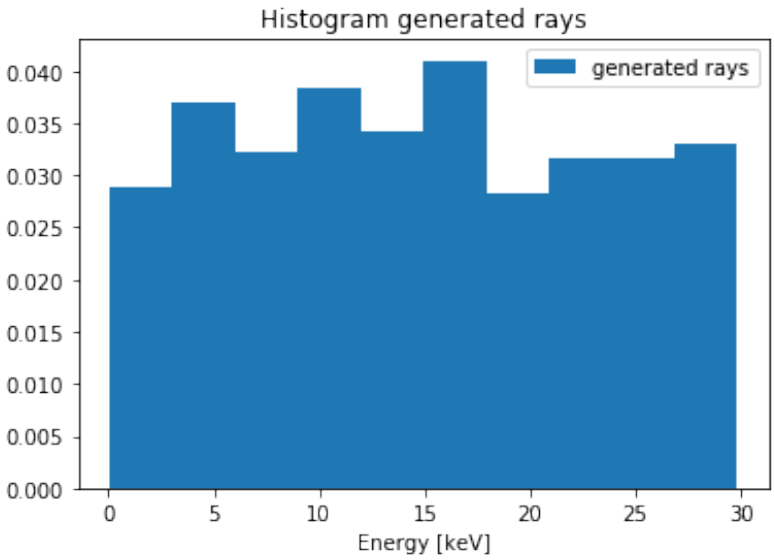


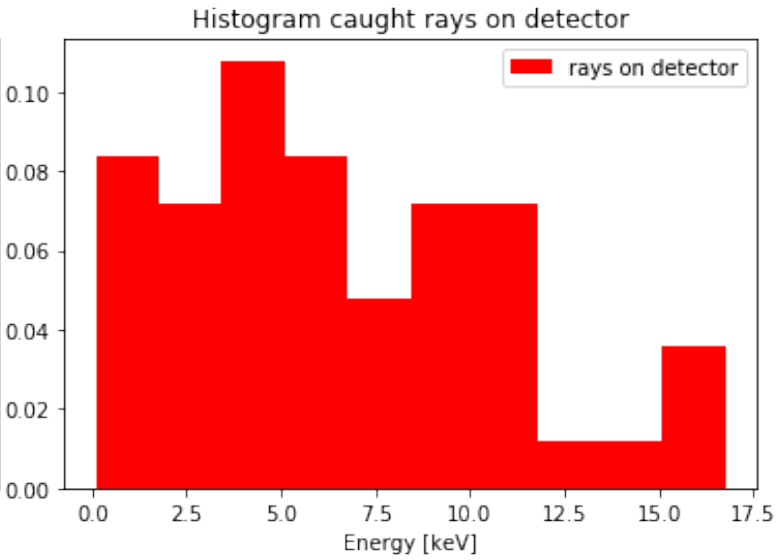
F = 500



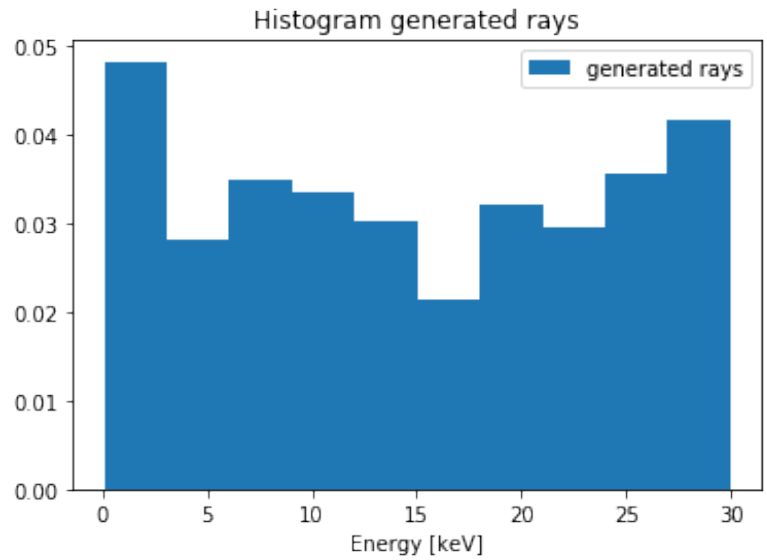


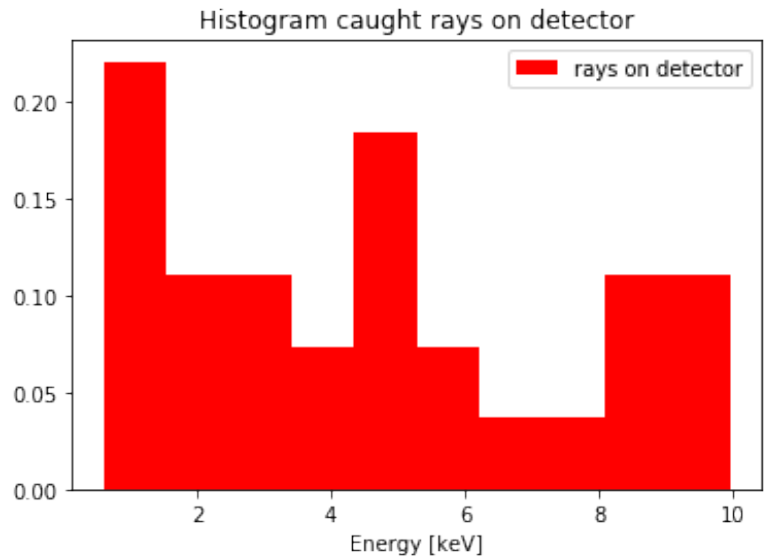
F = 400



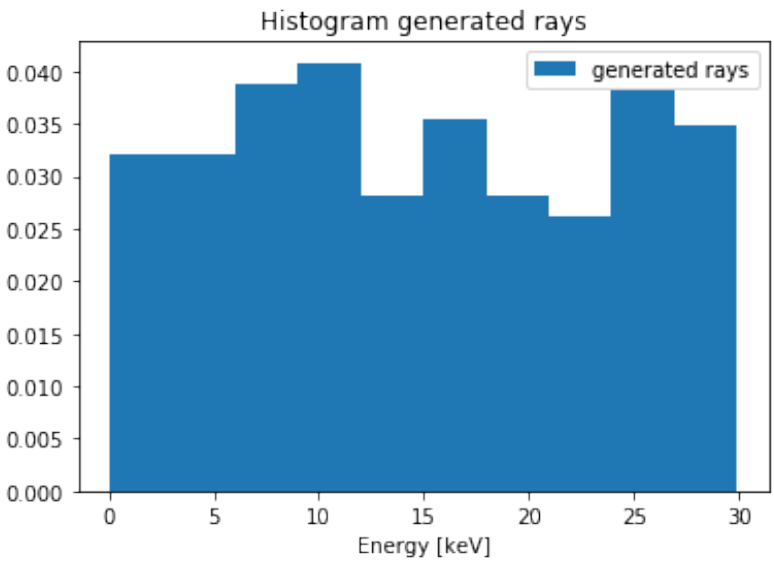


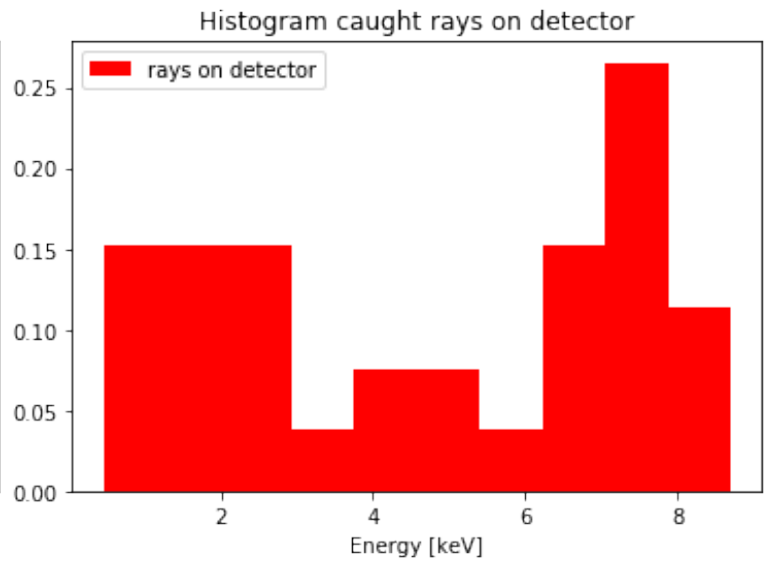
F = 300



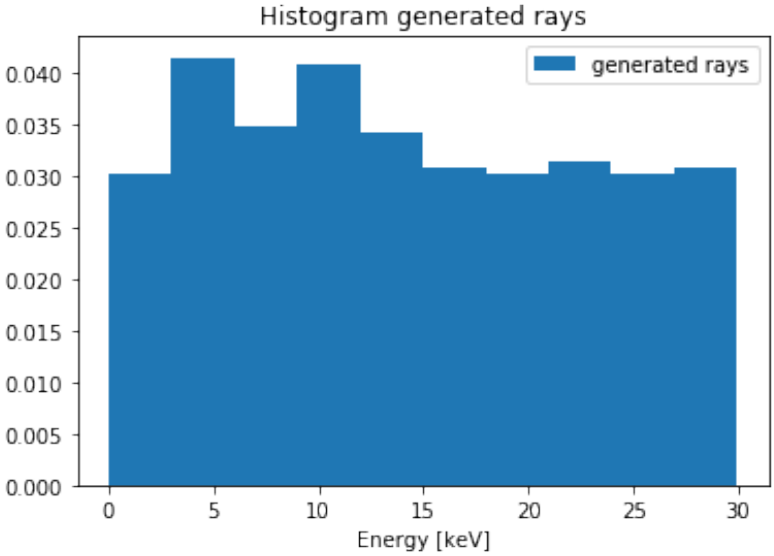


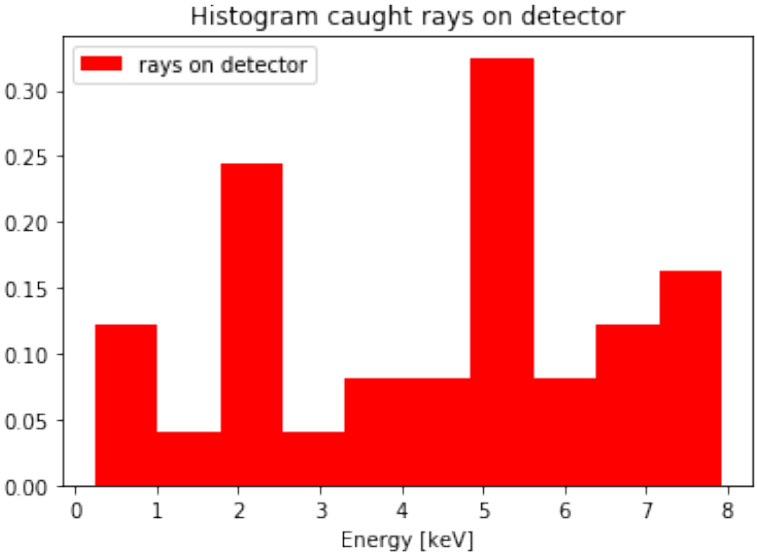
F = 250



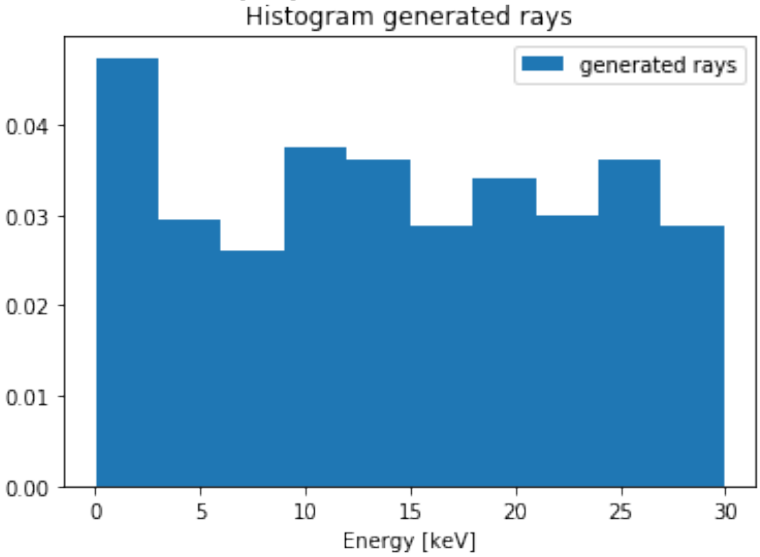


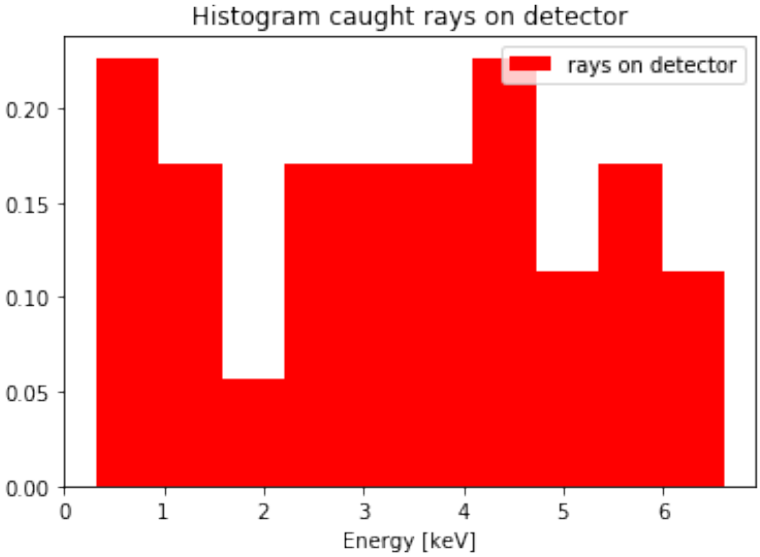
F = 200 - FOXSI rocket



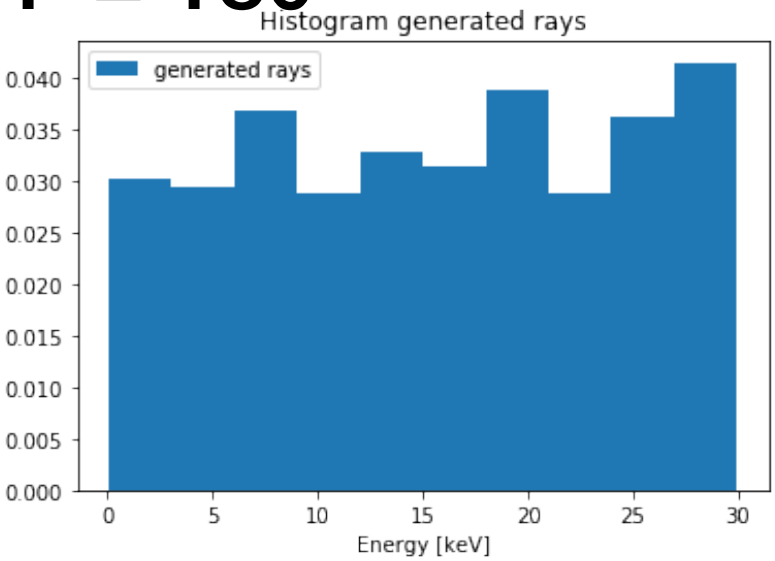


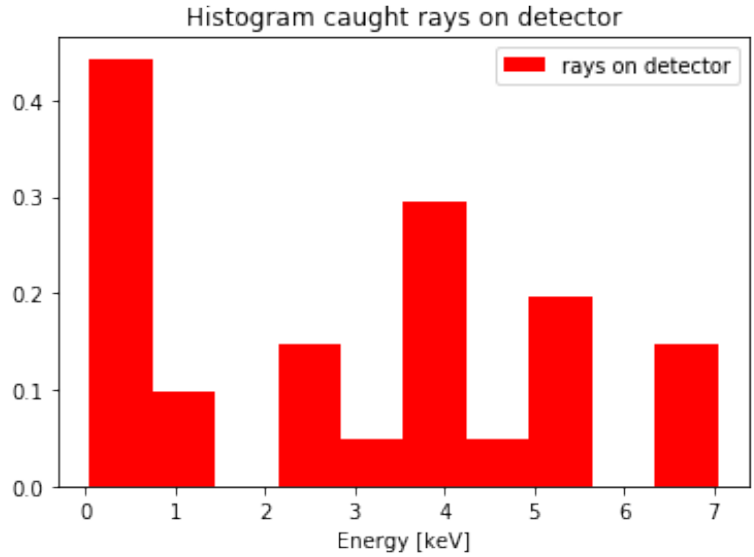
F = 190



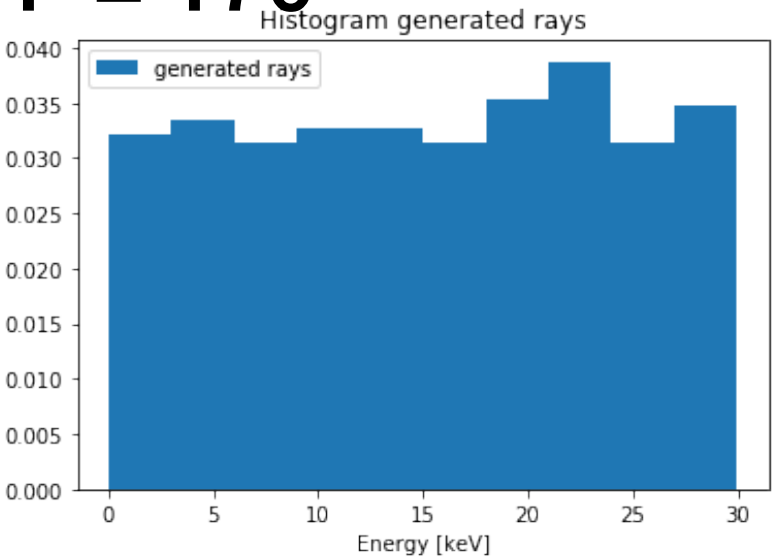


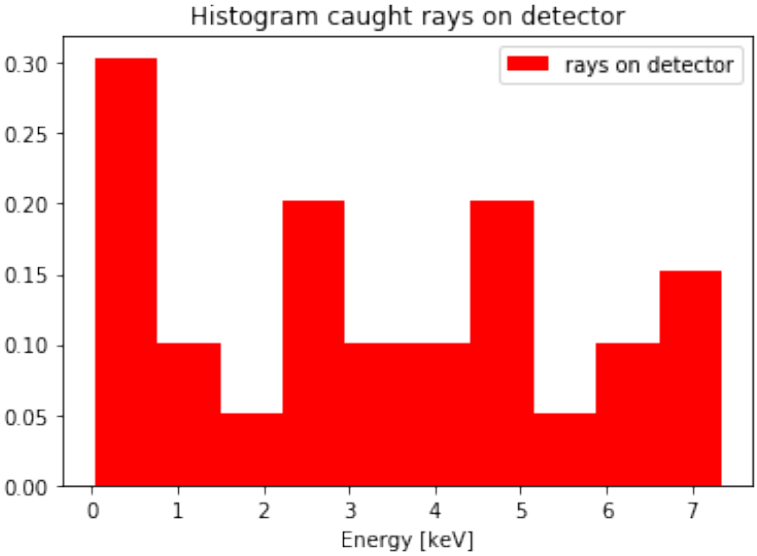
F = 180



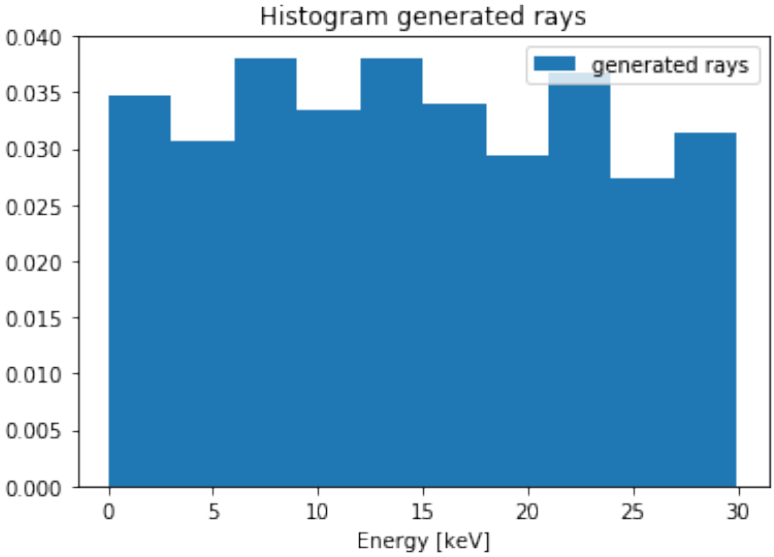


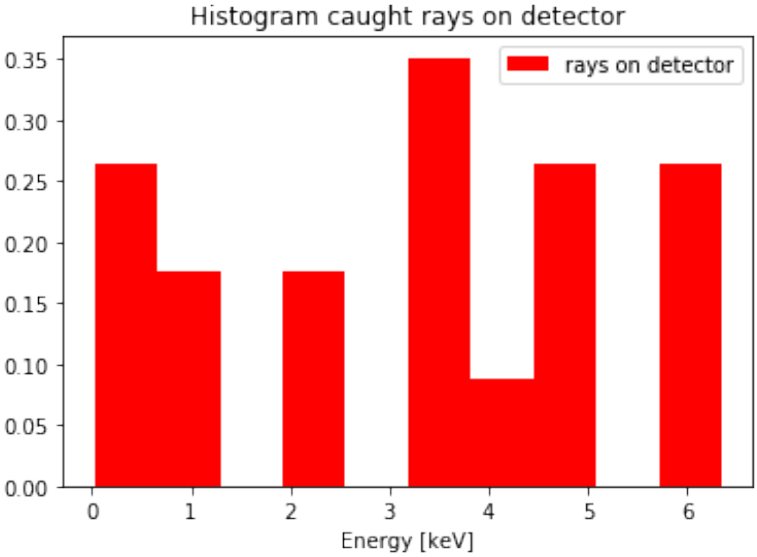
F = 170



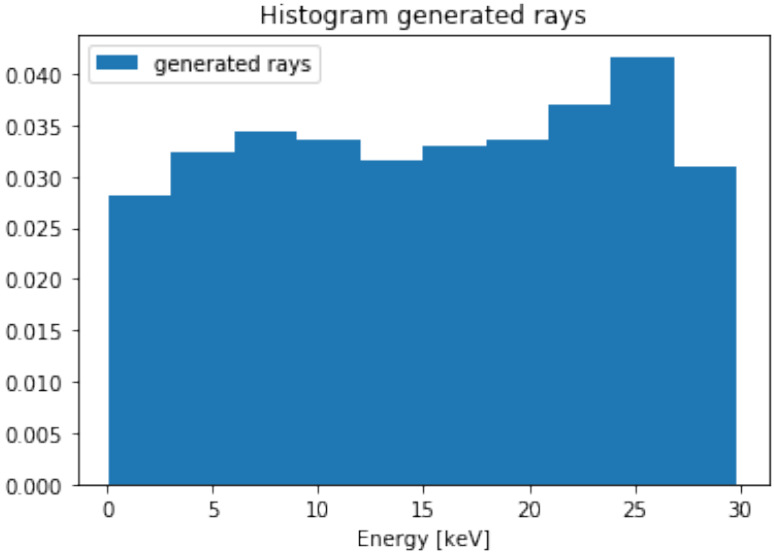


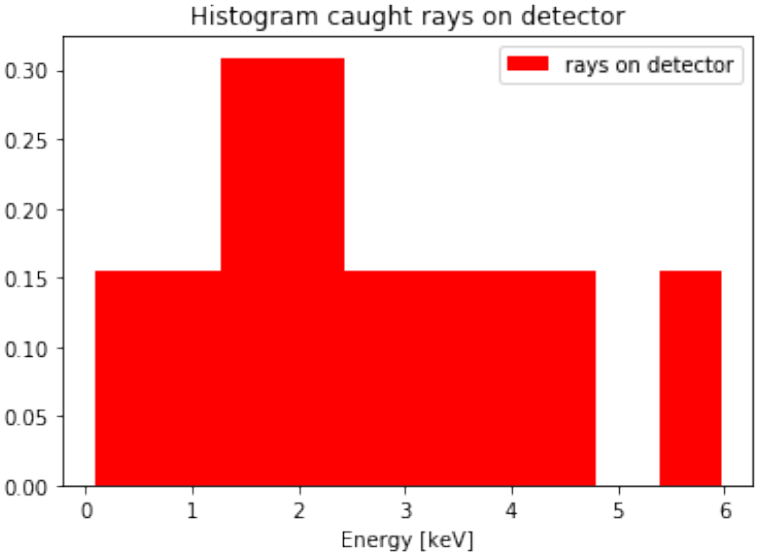
F = 160



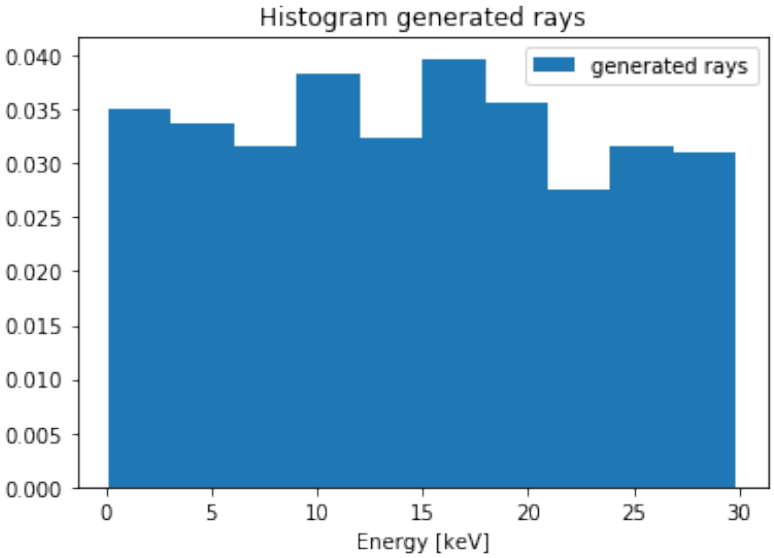


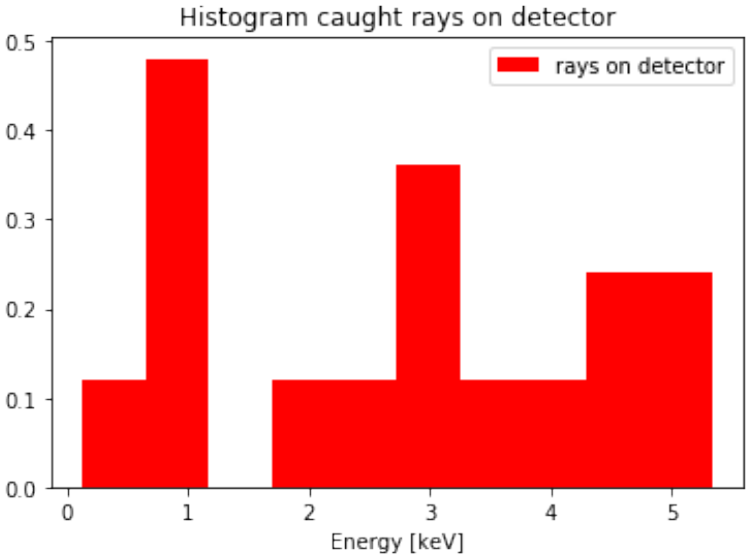
F = 150



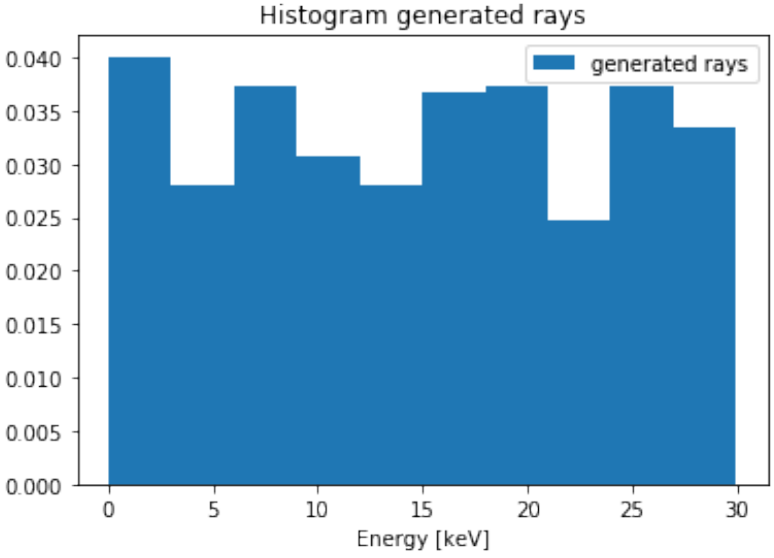


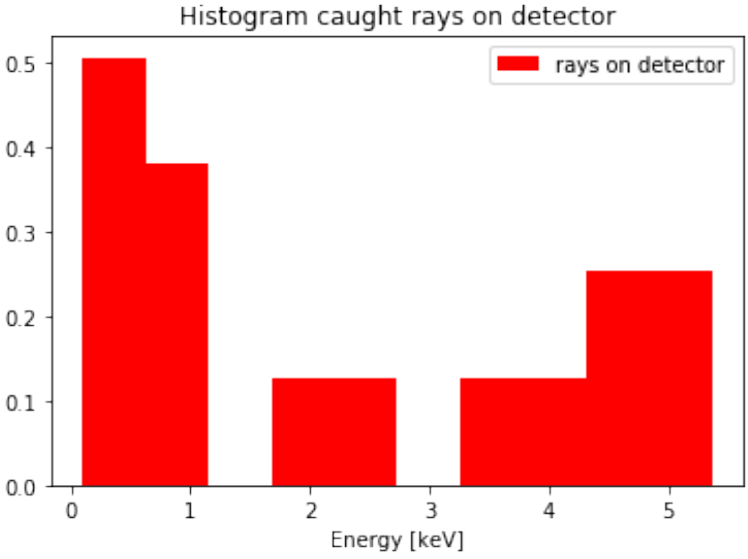
F = 140



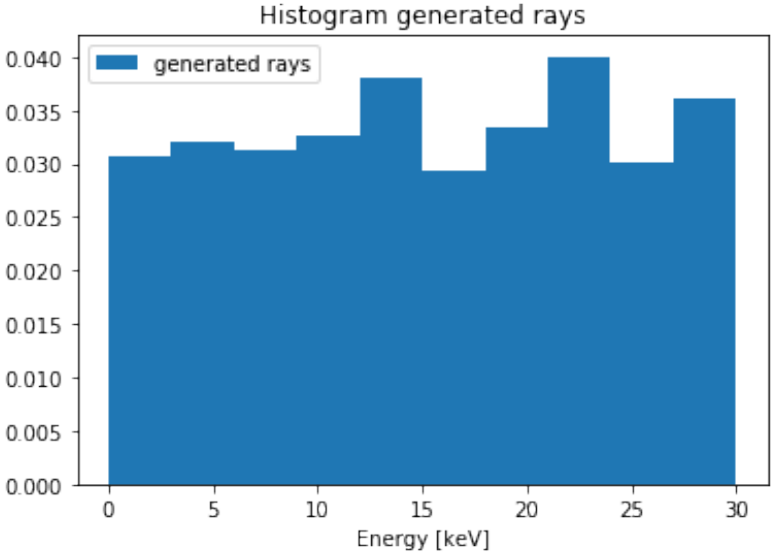


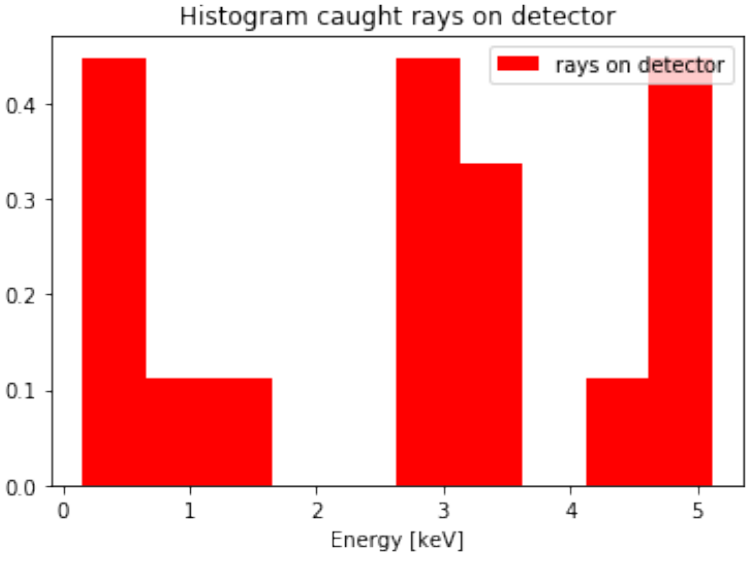
F = 130



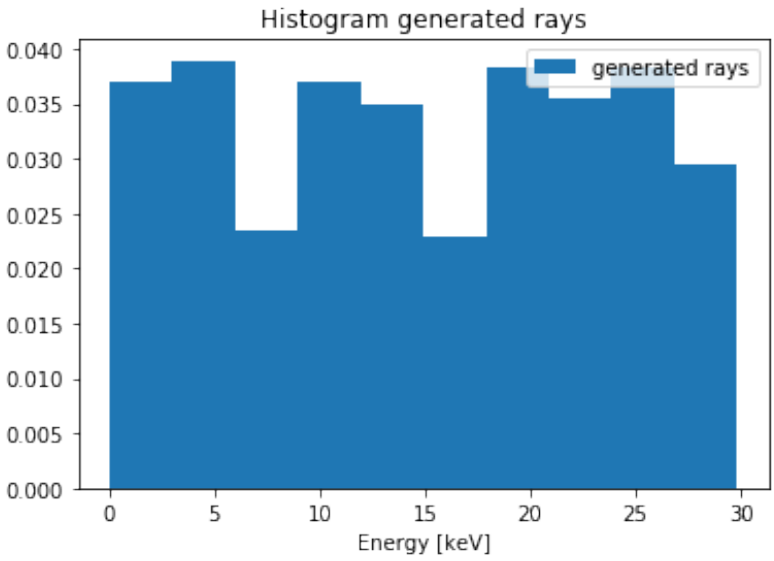


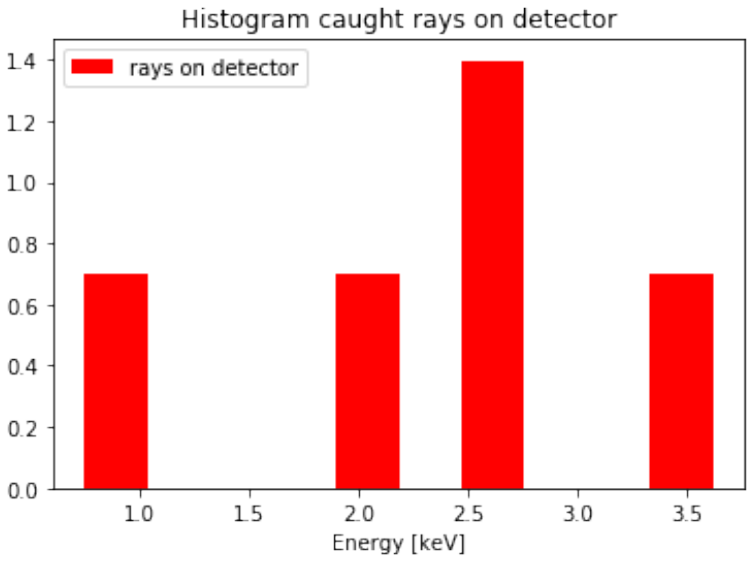
F = 120



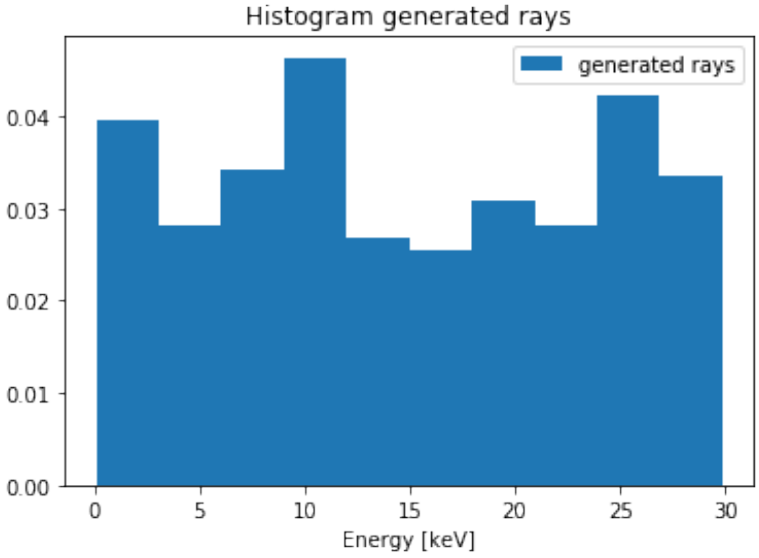


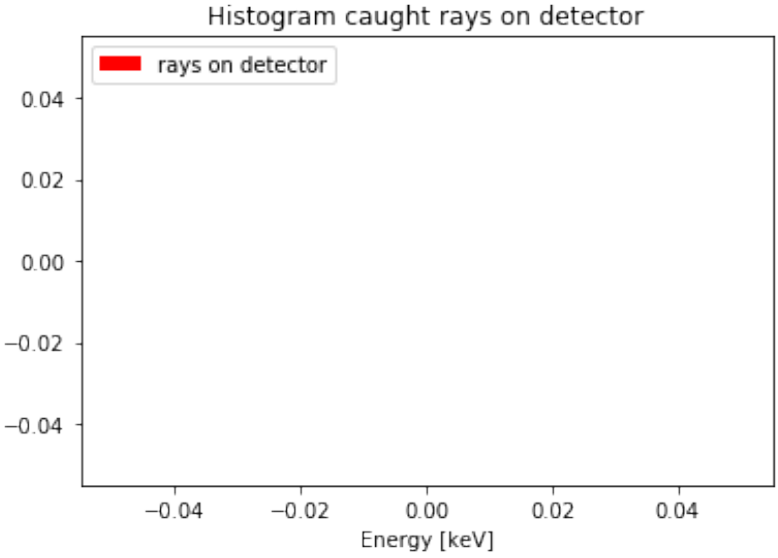
F = 110


Histogram generated rays
generated rays
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
5
10
15
20
25
30
Energy [keV]


Histogram caught rays on detector
rays on detector
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
Energy [keV]

F = 100


Histogram generated rays
generated rays
0.04
0.03
0.02
0.01
0.00
0
5
10
15
20
25
30
Energy [keV]


Histogram caught rays on detector
rays on detector
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
Energy [keV]

Piece of code used to generate these plots

```
from foxsisim.module import Module
from foxsisim.detector import Detector
from foxsisim.source import Source
from foxsisim.plotting import plot,scatterHist
import matplotlib.pyplot as plt
import numpy as np

max_energy = 30.0
def spectrum(z):
    if (type(z) is not type([1])) and (type(z) is not type(np.array(1))):
        x = np.array([z])
    else:
        x = np.array(z)
    return np.piecewise(x, [x < 0, ((x < max_energy) & (x > 0)), (x >= max_energy)], [0, 1./max_energy, 0])

source_distance = -1e4 ##cm
source = Source(type='point', center=[0, 0, source_distance])
source.loadSpectrum(spectrum)
energies = np.arange(-10, 60, 0.1)
plt.plot(energies, source._spectrum(energies))
plt.xlabel('Energy [keV]')
plt.title('Source Spectrum')
plt.show()

module = Module(radii=[5.151,4.9],focal=200.0)
detector = Detector(width=8,
                    height=8,
                    normal = [0,0,1],
                    center = [0,0,230],
                    reso = [1024,1024])

rays = source.generateRays(module.targetFront, 500)

plt.figure()
plt.hist([ray.energy for ray in rays], normed=True, label='generated rays')
plt.xlabel('Energy [keV]')
plt.title('Histogram generated rays')
plt.legend()
plt.show()

from datetime import datetime
tstart = datetime.now()

# pass rays through module
module.passRays(rays, robust=True)
# catch rays at detector
detector.catchRays(rays)


rays_on_detector = len(detector.rays)
print('Number of rays on Detector ' + str(rays_on_detector))
print('Time total: ' + str((datetime.now() - tstart).seconds) + ' seconds')
print('Time per ray (s): ' + str(rays_on_detector / float((datetime.now() - tstart).seconds)))

drays = [ray for ray in rays if ray.des[2]==230.0]
scatterHist(drays)
plt.show()

plt.figure()
plt.hist([ray.energy for ray in detector.rays], normed=True, label='rays on detector',color='r')
plt.xlabel('Energy [keV]')
plt.title('Histogram caught rays on detector')
plt.legend()
plt.show()
```