```
pip install astropy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Requirement already satisfied: astropy in /usr/local/lib/python3.8/dist-packages
Requirement already satisfied: pyerfa>=1.7.3 in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-pack
```

```
import numpy as np
import matplotlib.pyplot as plt
import math
import scipy as sp
import scipy.stats as stats
from astropy.io import fits
```

PART ONE

```
# Saving dark, flat, and target files of different exposure times into
# individual holders.

dark_15s = ['s0347.fits',
         's0348.fits',
         's0349.fits',
         's0350.fits',
         's0351.fits',
         's0352.fits',
         's0353.fits']

dark_44s = ['s0354.fits',
         's0355.fits',
         's0356.fits',
         's0357.fits',
         's0358.fits',
         's0359.fits',
         's0360.fits']

dark_150s = ['s0796.fits',
         's0797.fits',
         's0798.fits',
         's0799.fits',
         's0800.fits',
         's0801.fits',
         's0802.fits']

flat_44s = ['s0310.fits',
         's0311.fits',
         's0312.fits',
         's0313.fits',
```

```
        's0314.fits',
        's0315.fits',
        's0316.fits',
        's0317.fits',
        's0318.fits',
        's0319.fits']


tar_150s = ['s0576.fits',
        's0577.fits',
        's0578.fits',
        's0579.fits',
        's0580.fits',
        's0581.fits',
        's0582.fits',
        's0583.fits',
        's0584.fits',
        's0585.fits']

```

```
# Finding the median values for each of the three sets of dark frames (15 sec,
# 44 sec, and 150 sec exposure times).

def Med(files,extension):
    im = []
    for x in range(len(files)):
        im.append(fits.open(files[x])[extension].data)
    median = np.median(im,axis = 0)
    return median
```

```
# Applying function and saving values into new variables.

d15_med = Med(dark_15s,0)
d44_med = Med(dark_44s,0)
d150_med = Med(dark_150s,0)
```

```
# Subtracting the median dark values found above from the flat frames and target
# frames.

flat44s_d = []
for i in range(len(flat_44s)):
    flat44s_d.append(fits.open(flat_44s[i])[0].data - d44_med)

tar150s_d = []
for i in range(len(tar_150s)):
    tar150s_d.append(fits.open(tar_150s[i])[0].data - d150_med)
```

```
# Finding median flat frame after subtracting dark from all flat frames.
```

```
flats_med = np.median(flat44s_d,axis = 0)
```

```
# Create subsection of image above to find the median counts in the AO beam.
# Zooming in on the image created above and finding the median of the image.
# This median value is the median number of counts in the AO beam.

flat_zoom = flats_med[250:1250,600:1600]
zoom_med = np.median(flat_zoom)
```

```
# Creating a normalized flat by dividing the median flat image by the median of
# the subsection of the image.

flat_norm = flats_med/zoom_med
zoomed_norm_flat = flat_norm[250:1250,600:1600]
norm_med = np.median(flat_norm)
norm_std = np.std(flat_norm)
```

```
# This code ensures that no divisions by zero will occur when science frames are
# being divided by flat frames.

for  i in range(len(flat_norm)):
    for j in range(len(flat_norm[0])):
        if flat_norm[i][j] < 0.5:
            flat_norm[i][j] = 1
```

```
m_s_im = []
# This loop creates normalized science frames
for p in range(len(tar150s_d)):
    m_s_im.append(tar150s_d[p][250:1250,600:1600]/zoomed_norm_flat)
```

```
def centerfinder(s):
    keep = s

    for i in range(8):
        top = keep[0:int(len(keep)/2),0:len(keep[0])]
        bottom = keep[int(len(keep)/2):len(keep),0:len(keep[0])]
        if np.median(top)> np.median(bottom):
            keep = top
        else:
            keep = bottom


        left = keep[0:int(len(keep)),0:int(len(keep[0])/2)]
        right = keep[0:int(len(keep)),int(len(keep[0])/2):int(len(keep[0]))]
        if np.mean(left)> np.mean(right):
            keep = left
        else:
```

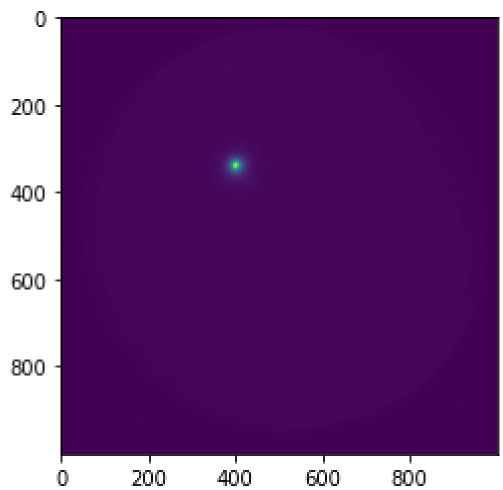
```
            keep = right

    for y in range(len(s)):
        for x in range(len(s[0])):
            if s[y][x] == np.max(keep):
                return [x,y]
```

```
for i in range(len(m_s_im)):
    for j in range(len(m_s_im[5])):
        for k in range(len(m_s_im[5][0])):
            if m_s_im[i][j][k] < 0:
                m_s_im[i][j][k] = 0

centerfinder(m_s_im[5])
```

```
[401, 338]
```

```
plt.imshow(m_s_im[5])
```

```
<matplotlib.image.AxesImage at 0x7f06b9423a30>
```



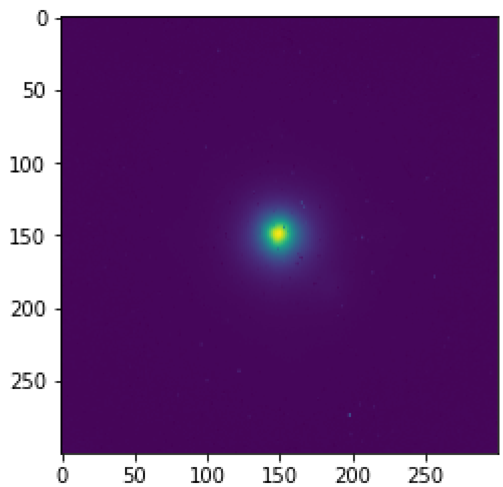

```
# Using centerfinder() function
# centers holds all info about where the center of each star is,
# snippets holds 100x100 pixels snippets centered on star center

centers = []
snippets = []
for f in range(len(m_s_im)):
    centers.append(centerfinder(m_s_im[f]))
```

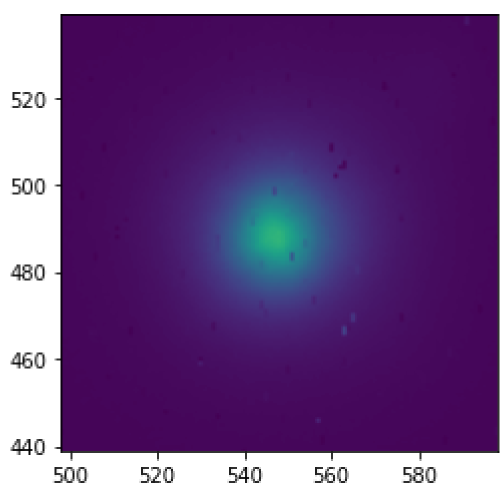
```
print(centers)
```

```
[[548, 489], [546, 487], [547, 487], [697, 334], [699, 331], [401, 338], [400, 3
```

```
# 'centers' is an array that holds information about the center of each
```

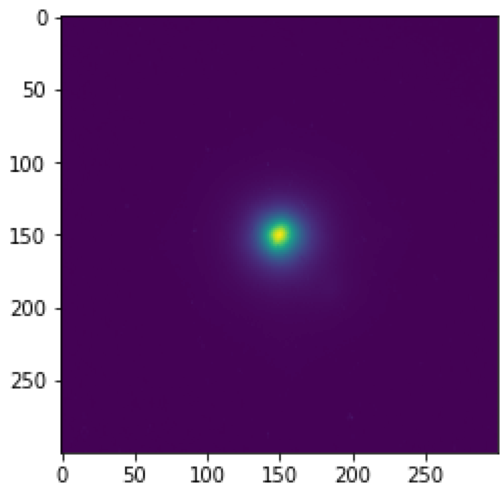
```
# normalized science image.

#append snippets with the desired sections of each normalized science image
for h in range(len(m_s_im)):
    snippets.append(m_s_im[h][(centers[h][1]-150):(centers[h][1]+150),(centers[h][

# Displaying image to see if star is centered
plt.imshow(snippets[0])
plt.show()

plt.imshow(m_s_im[0])
plt.axis([498,598,439,539])
plt.show()

```






```
# Combining all images with centered stars into one final image.

co_add= np.zeros([300,300])

for i in range(len(snippets)):
    co_add = co_add + snippets[i]

co_add = co_add/10
plt.imshow(co_add)
```

<matplotlib.image.AxesImage at 0x7f06b8e97f10>



```
psfx0 = snippets[0][150,0:300]
psfy0 = snippets[0][0:300,150]
```

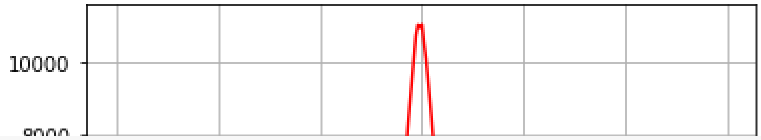
```
hm_psfx0 = (0.5)*psfx0.max()
hm_psfy0 = (0.5)*psfy0.max()
#these two values are the same! So are referred to as hm_psf0 in future lines

hm_psf0 = hm_psfx0
```

```
def FWHM(cut):
    hm = 0.5*cut.max()

    std = np.std(cut)
    val = []
    n = 0
    while len(val) != 2:
        n = n+1
        val = []
        for i in range(len(cut)):
            if cut[i] < hm + std/n and cut[i] > hm - std/n:
                val.append(i)

        if len(val) == 1:
            n = n-1
            val = []
            for i in range(len(cut)):
                if cut[i] < hm + std/n and cut[i] > hm - std/n:
                    val.append(i)

            if val[1] - val[0] < 5:
                return (val[-1] - val[-2])
                break
            else:
```

```
                return (val[1] - val[0])
                break





    return (val[1] - val[0])

```

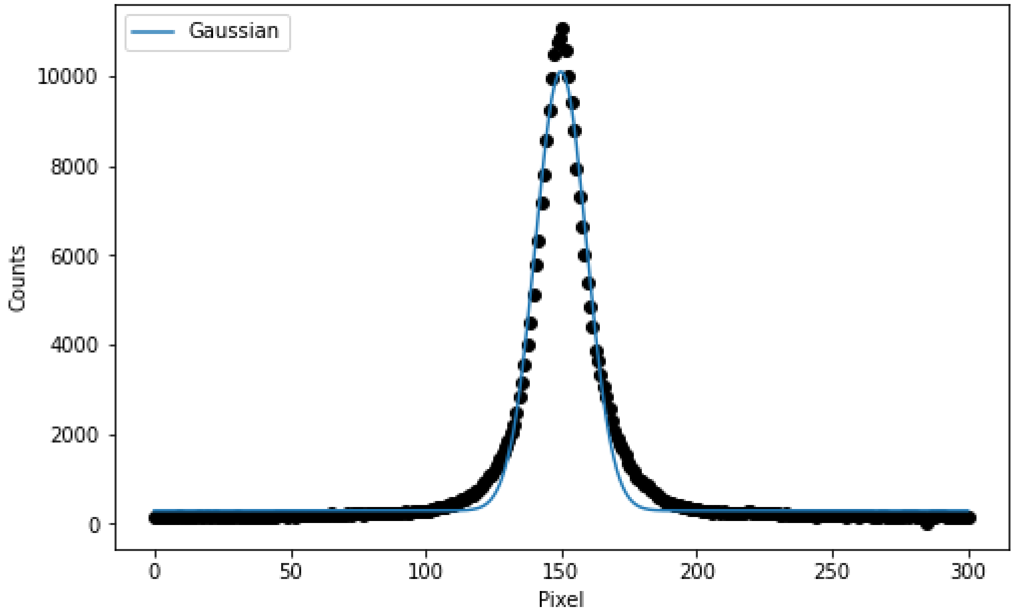
```
plt.plot(psfy0,color = 'red')
plt.axhline(y=hm_psf0,color = 'green')
print ('In the y direction the FWHM is',FWHM(psfy0))
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.grid()
plt.show()

plt.plot(psfx0,color = 'b')
plt.axhline(y=hm_psf0,color = 'orange')
print ('In the x direction the FWHM is',FWHM(psfx0))
plt.xlabel('Pixel Number')
plt.grid()
plt.ylabel('Counts')
plt.show()
```

```
In the y direction the FWHM is 21
```




```python
from astropy.modeling import models, fitting

xdata = np.linspace(0,300,300)
ydata = psfx0
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=np.mean(xdata), stddev=np.
fit_g = fitting.LevMarLSQFitter()
g_x = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko')
plt.plot(xdata, g_x(xdata), label='Gaussian')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
print('Gaussian Fit for the x direction')
plt.show()


xdata = np.linspace(0,300,300)
ydata = psfy0
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=50, stddev=np.std(xdata))
fit_g = fitting.LevMarLSQFitter()
g_y = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko')
plt.plot(xdata, g_y(xdata), label='Gaussian')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
print('Gaussian Fit for the y direction')
plt.show()
```

Gaussian Fit for the x direction



Gaussian Fit for the y direction


```
g_x
```

<CompoundModel(amplitude_0=9807.72937855, mean_0=149.84883466, stddev_0=9.46911548, amplitude_1=296.61035068)>

```
g_y
```

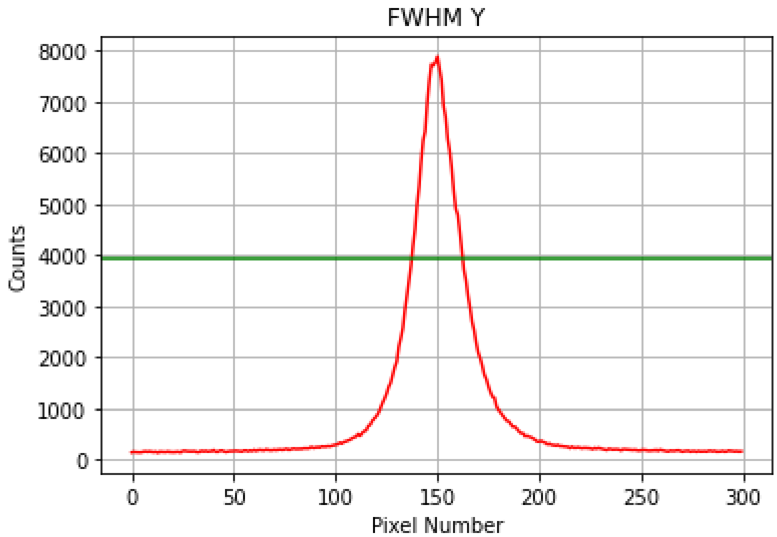
<CompoundModel(amplitude_0=9987.78236481, mean_0=149.92063103, stddev_0=9.75056617, amplitude_1=287.80551201)>

```
psfx3 = snippets[3][150,0:300]
psfy3 = snippets[3][0:300,150]
hm_psfx3 = (0.5)*psfx3.max()
hm_psfy3 = (0.5)*psfy3.max()
```

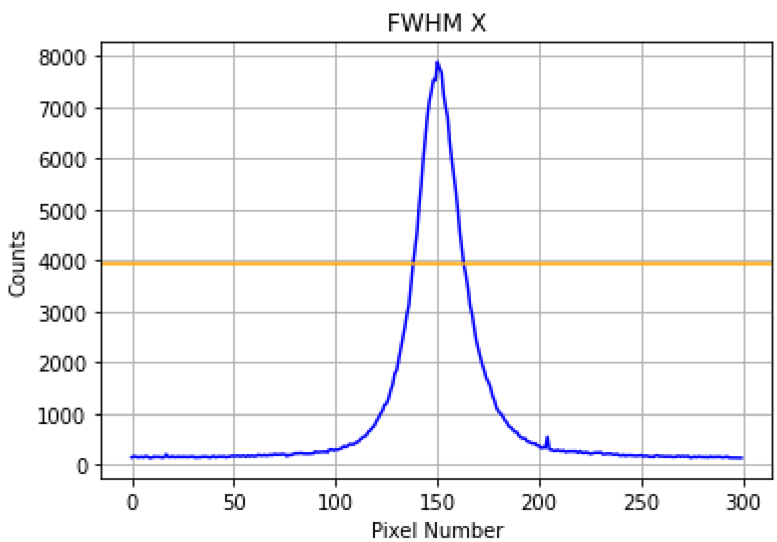
```
plt.plot(psfy3,color = 'red')
plt.axhline(y=hm_psfy3,color = 'green')
print('FWHM in Y-Direction: ',FWHM(psfy3))
plt.title("FWHM Y")
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.grid()
plt.show()

plt.plot(psfx3,color = 'b')
plt.axhline(y=hm_psfx3,color = 'orange')
print ('FWHM in X-Direction: ',FWHM(psfx3))
plt.title("FWHM X")
```

```
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.grid()
plt.show()
```

```
FWHM in Y-Direction:  25
```




```
FWHM in X-Direction:  25
```




```
xdata = np.linspace(0,300,300)
ydata = psfx3
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=np.mean(xdata), stddev=np.
fit_g = fitting.LevMarLSQFitter()
g_x3 = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko', label="Emission Profile")
plt.plot(xdata, g_x3(xdata), label='Gaussian Fit')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title('Gaussian Fit: X-Direction')
plt.grid()
```

```
plt.show()


xdata = np.linspace(0,300,300)
ydata = psfy3
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=50, stddev=np.std(xdata))
fit_g = fitting.LevMarLSQFitter()
g_y3 = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko', label="Emission Profile")
plt.plot(xdata, g_y3(xdata), label='Gaussian Fit')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title('Gaussian Fit: Y-Direction')
plt.grid()
plt.show()
```


```
g_x3
```

<CompoundModel(amplitude_0=6987.39909388, mean_0=151.09815486, stddev_0=12.07673428, amplitude_1=256.83767016)>

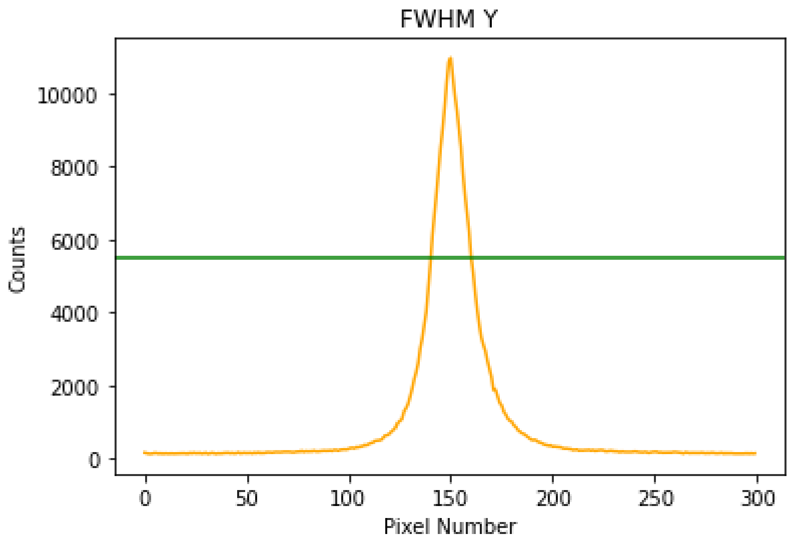
```
g_y3
```

<CompoundModel(amplitude_0=6996.54213072, mean_0=150.28753579, stddev_0=11.99664591, amplitude_1=253.13829597)>

```
psfx5 = snippets[5][150,0:300]
psfy5 = snippets[5][0:300,150]
hm_psfx5 = (0.5)*psfx5.max()
hm_psfy5 = (0.5)*psfy5.max()
```

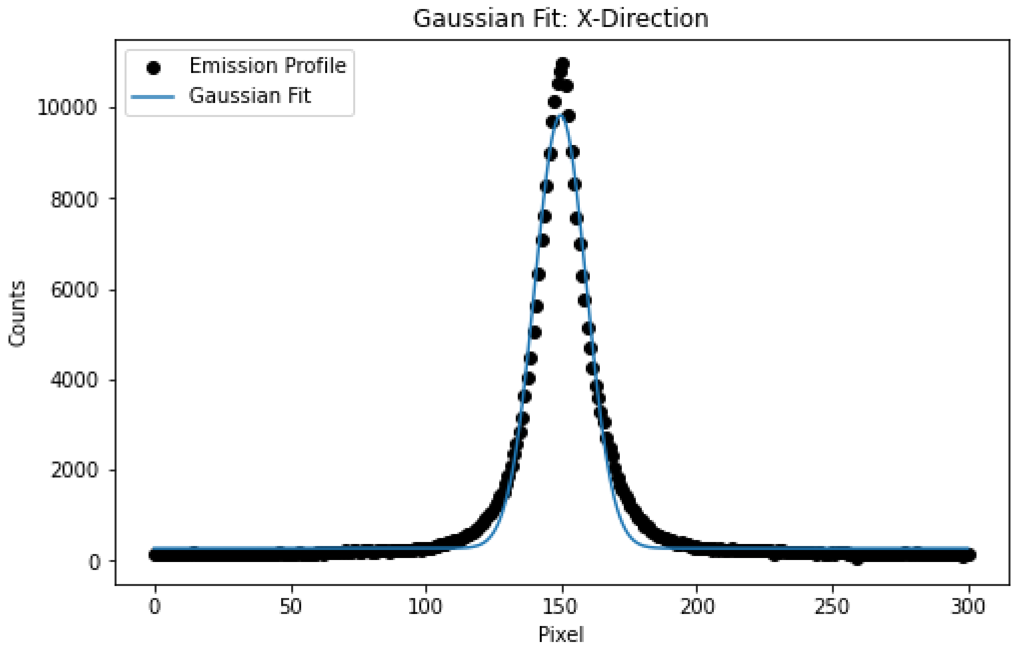
```
plt.plot(psfy5,color = 'orange')
plt.axhline(y=hm_psfy5,color = 'green')
print ('FWHM in Y-Direction: ',FWHM(psfy5))
plt.title("FWHM Y")
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.show()

plt.plot(psfx5,color = 'b')
plt.axhline(y=hm_psfx5,color = 'yellow')
print ('FWHM in X-Direction: ',FWHM(psfx5))
plt.title("FWHM X")
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.show()
```

```
FWHM in Y-Direction:  20
```

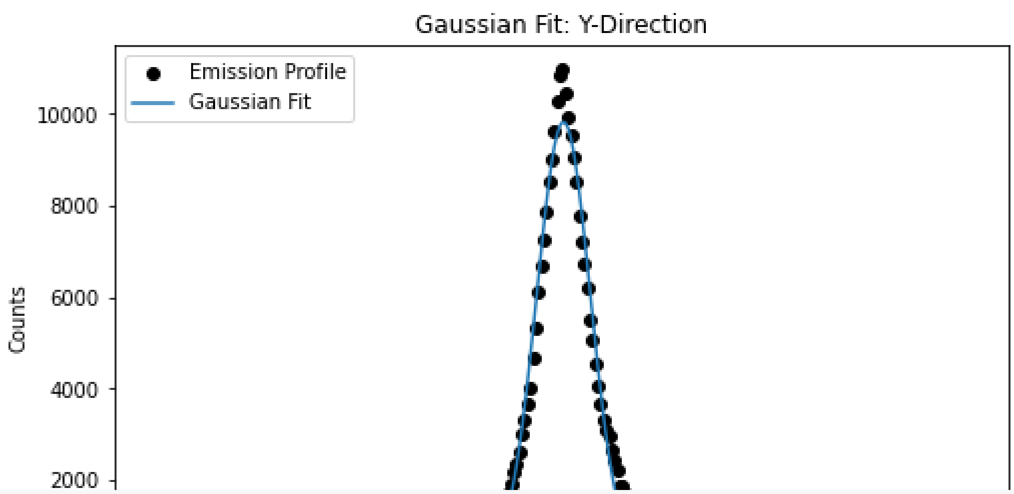



```
FWHM in X-Direction:  18
```

```
xdata = np.linspace(0,300,300)
ydata = psfx5
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=np.mean(xdata), stddev=np.
fit_g = fitting.LevMarLSQFitter()
g_x5 = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko', label="Emission Profile")
plt.plot(xdata, g_x5(xdata), label='Gaussian Fit')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title('Gaussian Fit: X-Direction')
plt.show()


xdata = np.linspace(0,300,300)
ydata = psfy5
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=50, stddev=np.std(xdata))
fit_g = fitting.LevMarLSQFitter()
g_y5 = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko', label="Emission Profile")
plt.plot(xdata, g_y5(xdata), label='Gaussian Fit')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title('Gaussian Fit: Y-Direction')
plt.show()
```




```
WARNING: The fit may be unsuccessful; check fit_info['message'] for more informa
WARNING:astropy:The fit may be unsuccessful; check fit_info['message'] for more i
```




```
g_x5
```

```
<CompoundModel(amplitude_0=9555.96762134, mean_0=149.72561841,
stddev_0=9.5578744, amplitude_1=273.24176652)>
```

```
g_y5
```

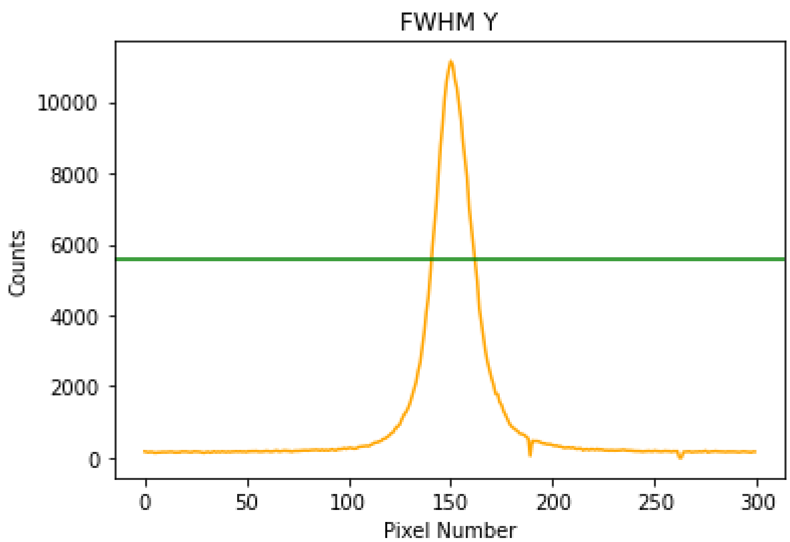
```
<CompoundModel(amplitude_0=9552.96647431, mean_0=150.80062487,
stddev_0=9.88539749, amplitude_1=275.12640495)>
```

```
psfx9 = snippets[9][150,0:300]
psfy9 = snippets[9][0:300,150]
hm_psfx9 = (0.5)*psfx9.max()
hm_psfy9 = (0.5)*psfy9.max()
```

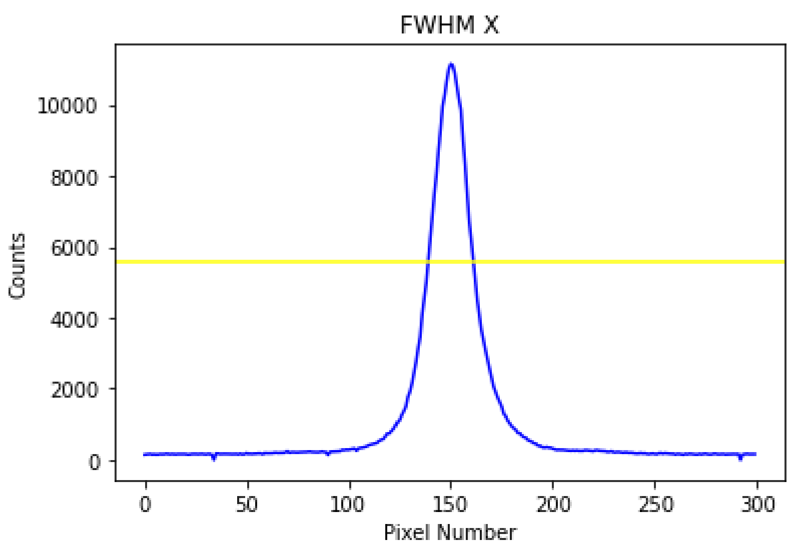
```
plt.plot(psfy9,color = 'orange')
```

```
plt.axhline(y=hm_psfy9,color = 'green')
print ('FWHM in Y-Direction: ',FWHM(psfy9))
plt.title("FWHM Y")
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.show()

plt.plot(psfx9,color = 'b')
plt.axhline(y=hm_psfx9,color = 'yellow')
print ('FWHM in X-Direction: ',FWHM(psfx9))
plt.title("FWHM X")
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.show()
```

```
FWHM in Y-Direction:  20
```

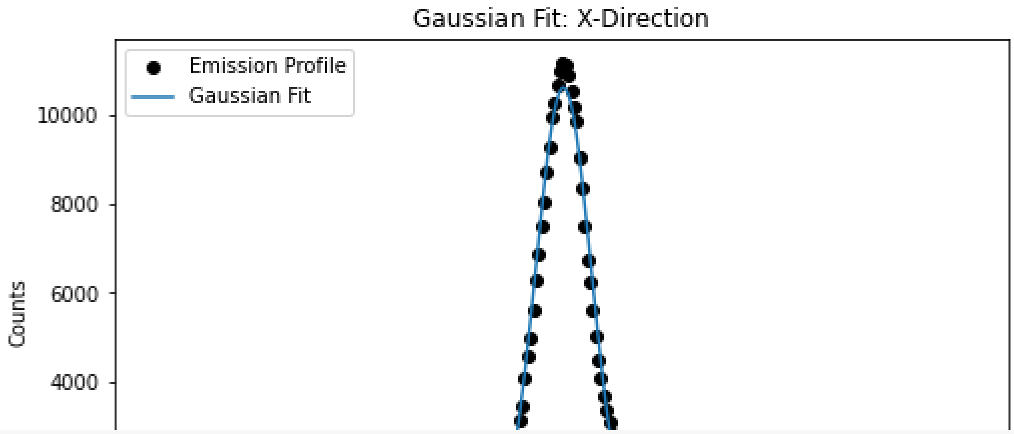



```
FWHM in X-Direction:  22
```




```
xdata = np.linspace(0,300,300)
ydata = psfx9
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=np.mean(xdata), stddev=np.
fit_g = fitting.LevMarLSQFitter()
```

```
g_x9 = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko', label="Emission Profile")
plt.plot(xdata, g_x9(xdata), label='Gaussian Fit')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title('Gaussian Fit: X-Direction')
plt.show()


xdata = np.linspace(0,300,300)
ydata = psfy9
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=50, stddev=np.std(xdata))
fit_g = fitting.LevMarLSQFitter()
g_y9 = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko', label="Emission Profile")
plt.plot(xdata, g_y9(xdata), label='Gaussian Fit')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title('Gaussian Fit: Y-Direction')
plt.show()
```




```
g_x9
```

<CompoundModel(amplitude_0=10349.17915377, mean_0=150.69518163, stddev_0=10.27463531, amplitude_1=266.94409771)>

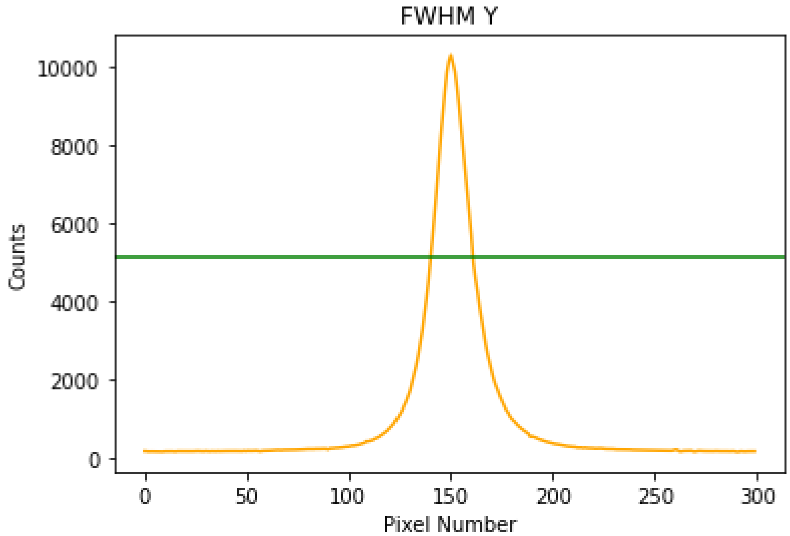
```
g_y9
```

<CompoundModel(amplitude_0=10227.75418275, mean_0=151.5645408, stddev_0=9.76196401, amplitude_1=264.2761535)>

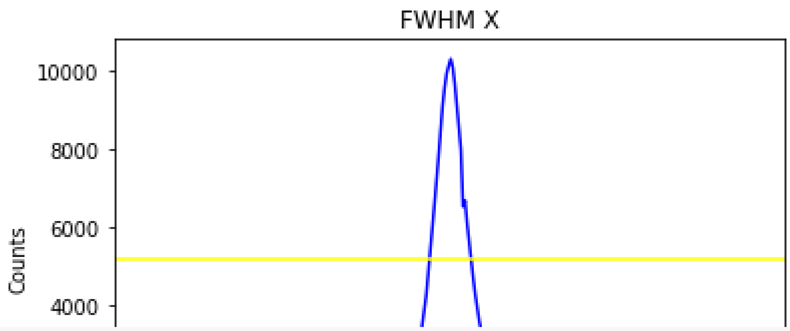
```
psfx = co_add[150,0:300]
psfy = co_add[0:300,150]
hm_psfx = (0.5)*psfx.max()
hm_psfy = (0.5)*psfy.max()
```

```
plt.plot(psfy,color = 'orange')
plt.axhline(y=hm_psfy,color = 'green')
print ('FWHM in Y-Direction: ',FWHM(psfy))
plt.title('FWHM Y')
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.show()

plt.plot(psfx,color = 'b')
plt.axhline(y=hm_psfx,color = 'yellow')
print ('FWHM in X-Direction: ',FWHM(psfx))
plt.title('FWHM X')
plt.xlabel('Pixel Number')
plt.ylabel('Counts')
plt.show()
```

```
FWHM in Y-Direction:  21
```

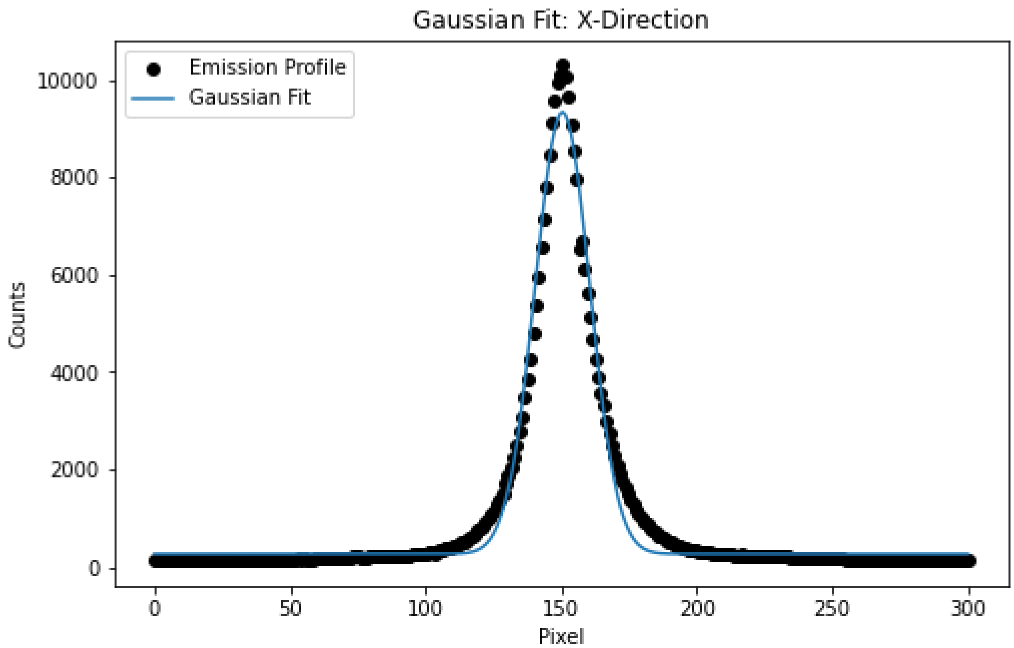


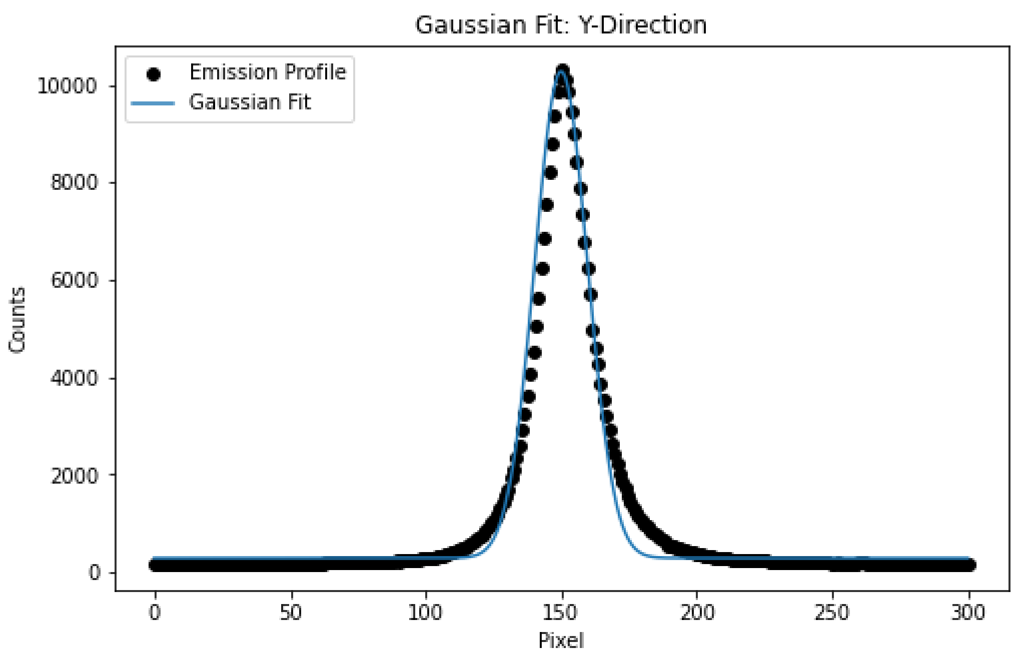

```
FWHM in X-Direction:  20
```




```
xdata = np.linspace(0,300,300)
ydata = psfx
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=np.mean(xdata), stddev=np.
fit_g = fitting.LevMarLSQFitter()
g_x = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko', label="Emission Profile")
plt.plot(xdata, g_x(xdata), label='Gaussian Fit')
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title('Gaussian Fit: X-Direction')
plt.show()


xdata = np.linspace(0,300,300)
ydata = psfy
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=50, stddev=np.std(xdata))
fit_g = fitting.LevMarLSQFitter()
g_y3 = fit_g(g_init, xdata, ydata)
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata,'ko', label="Emission Profile")
plt.plot(xdata, g_y(xdata), label='Gaussian Fit')
plt.xlabel('Pixel')
plt.ylabel('Counts')
```

```python
plt.legend(loc=2)
plt.title('Gaussian Fit: Y-Direction')
plt.show()
```






```python
g_x
```

```
<CompoundModel(amplitude_0=9051.95159099, mean_0=150.33536069,
stddev_0=10.12383687, amplitude_1=279.87173984)>
```

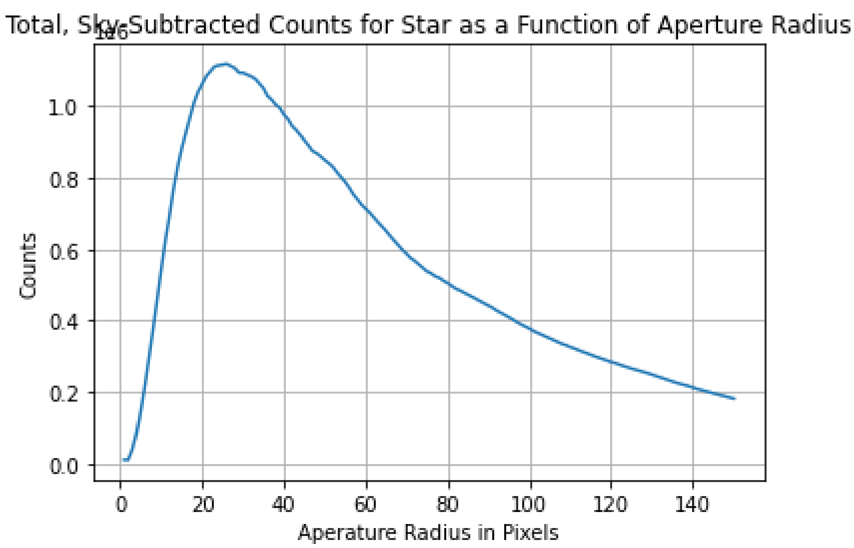
```python
g_y
```

```
<CompoundModel(amplitude_0=9987.78236481, mean_0=149.92063103,
stddev_0=9.75056617, amplitude_1=287.80551201)>
```

PART TWO

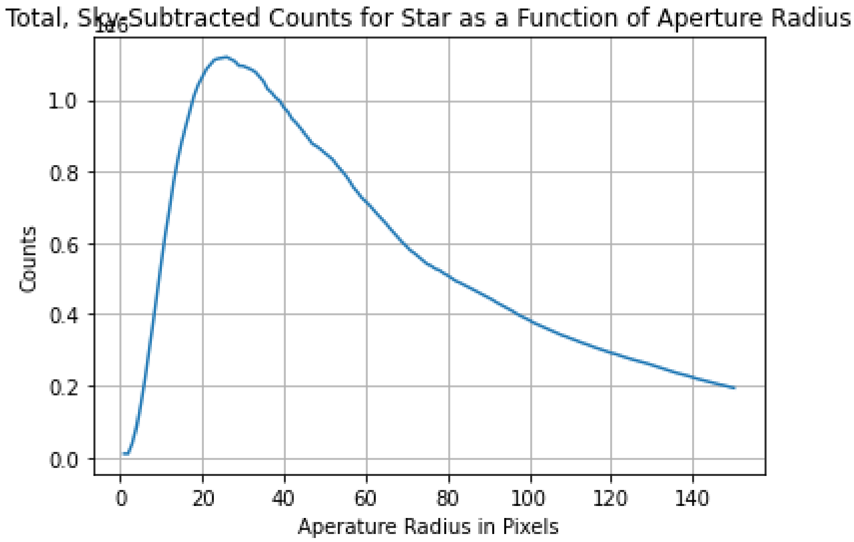
```
def func(data,y,x,n,s):
    pc = [data[y][x]]
    size = n
    local = [n]
    sky_sum = np.median(data[0:20*s,0:20*s])
    tot_sky = []
    fin = [pc[0]-sky_sum]

    while size < 150:
        start = data[x-size:x+size,y-size:y+size]
        sumz = size*size*np.median(start.flatten())
        sky = sky_sum*size*size
        tot_sky.append(np.abs(sky))
        pc.append(sumz)
        fin.append(sumz - sky)
        size = size + n
        local.append(size)

    plt.plot(local,fin)
    plt.grid()
    plt.title("Total, Sky-Subtracted Counts for Star as a Function of Aperture Radi
    plt.xlabel('Aperature Radius in Pixels')
    plt.ylabel('Counts')
```

```
func(co_add,150,150,1,1)
```

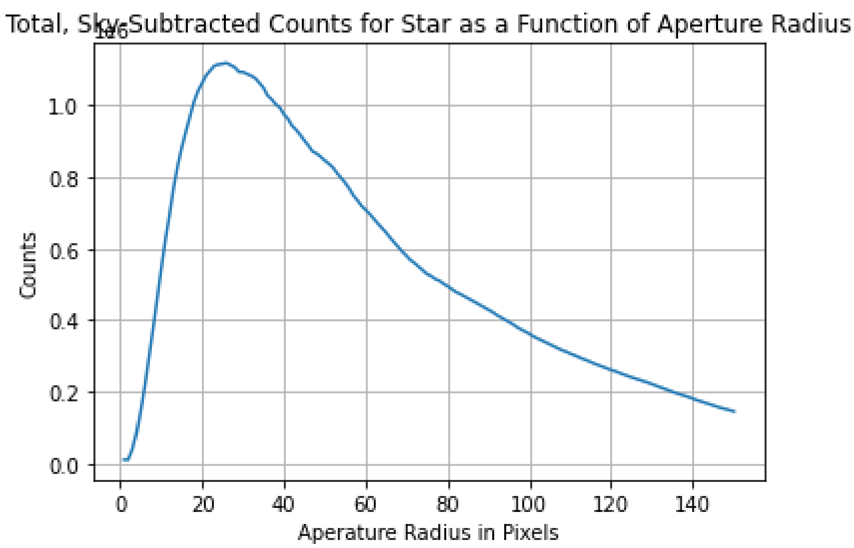



```
# Varying the location and size of the aperture used to measure the sky
```

```
func(co_add,150,150,1,2)
```



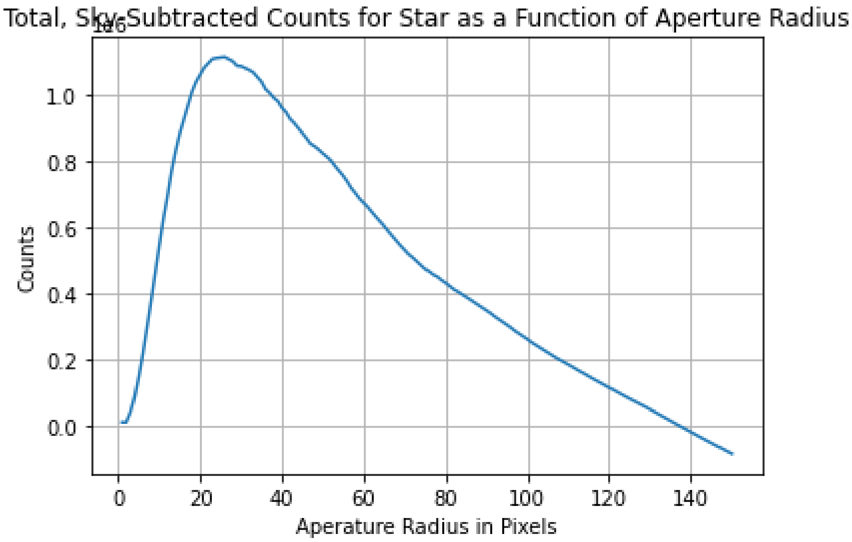

```
1 func(co_add,150,150,1,5)
```




```
1 func(co_add,150,150,1,7)
```

Total, Sky-Subtracted Counts for Star as a Function of Aperture Radius
1e6

```
func(co_add,150,150,1,10)
```



I will select an aperture of 25 within which to measure the total flux of the star. No matter how the location and size of the aperture being used to measure the sky varies, the graph has a peak around a radius of 25 pixels.

```
val = co_add[125:175,125:175]
vsum = val.sum()
dvsum = vsum/150
print("Total Counts/Second for Targeted Star: ",dvsum, "Counts/Second")
```

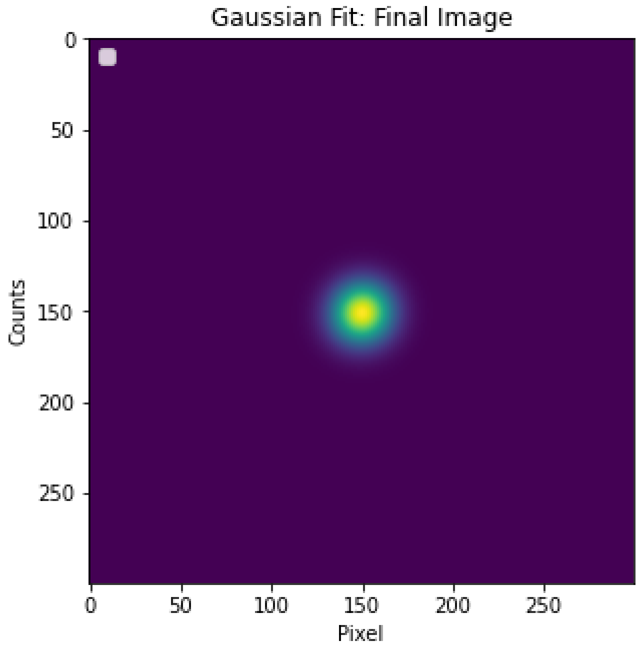
```
Total Counts/Second for Targeted Star:  45421.764173528034 Counts/Second
```

PART THREE

```
y_data = np.arange(300)
x_data = np.arange(300)

x, y = np.meshgrid(x_data,y_data)

z_data = co_add
g_init3 = models.Gaussian2D(amplitude=co_add.max(), x_mean=150., y_mean=150., x_st
                            )+ models.Const2D(amplitude=1.)
fit_g3 = fitting.LevMarLSQFitter()
g_3 = fit_g3(g_init3,x,y,z_data)
plt.figure(figsize=(8,5))

plt.imshow(g_3(x,y), label='Gaussian')
plt.xlabel('Pixel')
```

```
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title('Gaussian Fit: Final Image')
plt.show()
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```




```
g_3
```

```
<CompoundModel(amplitude_0=7867.28167245, x_mean_0=150.0927443,
y_mean_0=150.74957877, x_stddev_0=11.83045637, y_stddev_0=11.51437649,
theta_0=84.00115207, amplitude_1=194.30539993)>
```

```
(g_3(x,y)[125:175,125:175].sum() - (194.7*2500))/150
```

```
41989.58112719517
```

```
val.sum()/150
```

```
45421.764173528034
```