The Moffat Point-Spread Function

- The PSF takes into account the spreading of the photons due to the atmospheric turbulence (seeing)

- It distributes the signal over an area larger than the real source ones (e.g., a star becomes a circle...)

- It can be analitically modelled.

Here we follow Trujillo et al., MNRAS, 2001, 328, 977.

Circular Moffat Point Spread Function:

$$\mathrm{PSF}(r) = \frac{\beta - 1}{\pi \alpha^2} \left[1 + \left(\frac{r}{\alpha} \right)^2 \right]^{-\beta},$$

where r is the distance from the center of the source, beta is a parameter of the function and the Full-Width Half-Maximum is:

$$\mathrm{FWHM} = 2\alpha\sqrt{2^{1/\beta} - 1},$$

Exercize: reproduce the upper panel of Fig. 1 in Trujillo 2001 (including the Gaussian)

Use alpha=2.1 ; for the gaussian, remember that FWHM=sigma, so mean is zero and sigma=1 in units of r/FWHM

Hint: on the x axis we have r/FWHM. So, generate r between a small number and 4, then multiply it by FWHM.

Now, substitute in Eq. 1 and produce a vector of r/f values as in Fig. 1. Note that the values of this vector do depend on beta.

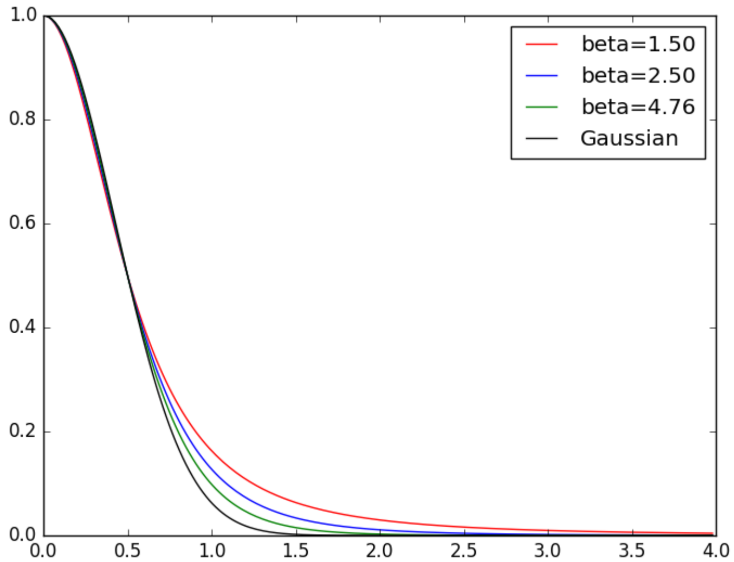
Note that on the y-axis, you have PSF(r)/PSF(0)

Try to use functions: they will be useful in the following exercizes.

```
def PSF(r, alpha, beta):
        res= ….
        return res
N=200
beta=1.50
alpha=1.75
 psf=np.zeros(N)
 fwhm=(2.*alpha*np.sqrt(2.**(1/b)-1))
 r=np.arange(0.01, 4, (4-0.01)/N)
  for i,x in enumerate(r):
    psf[i]=PSF(x*fwhm,alpha,b)


 psf/= psf[0]
….
```

You should get something like:



Question: could you use a lambda function?

Question: what is the effect of varying alpha?

Exercize: produce the bottom panel of Fig. 1

Final exercize: evaluate and plot the fraction of energy inside R, varying beta.

For doing this, we must do an area integral of PSF(r):

$$\int_0^{2\pi} d\varphi \int_0^R r\,PSF(r)\,dr = 2\pi \int_0^R r\,PSF(r)\,dr$$

This must be normalized to the same integral, done for $R \to \infty$ (Numerically, R very large)

Hints:

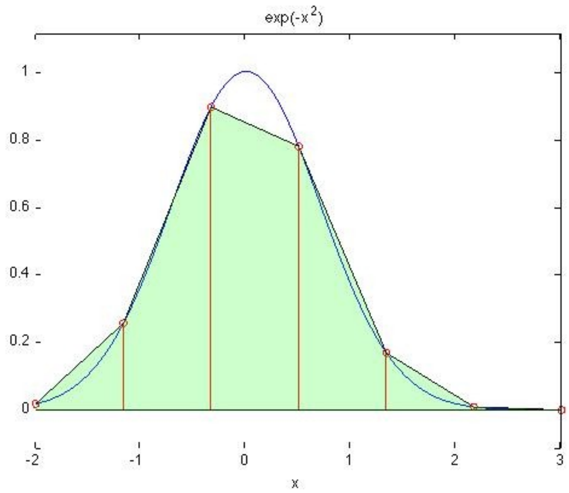
- write a function to evaluate PSF(r), with beta as a parameter

- write a function to evaluate the integral, at a given R, that must call the previous one

- write the main body of the code using such functions.

- do the integral using the trapezoidal rule

$$
\begin{aligned}
y = \int_a^b f(x)dx &\cong \frac{1}{2}\left[\sum_{i=1}^{n} f(x_{i-1})\Delta x + \sum_{i=1}^{n} f(x_i)\Delta x\right] \\
&= \frac{\Delta x}{2}\{[f(x_0)+f(x_1)]+[f(x_1)+f(x_2)]+ \\
&\quad \ldots + [f(x_{n-1})+f(x_n)]\} \\
&= \frac{\Delta x}{2}[f(x_0)+2f(x_1)+\ldots+2f(x_{n-1})+f(x_n)]
\end{aligned}
$$



- redo the integral using scipy.integrate. For example, the integral of:

$$\int_0^1 a\,x^2 + bx\,dx$$

with a=2, b=1 can be computed by this way:

```
>>> from scipy.integrate import quad
>>> def integrand(x, a, b):
...     return a*x**2 + b
...
>>> a = 2
>>> b = 1
```

```
>>> I = quad(integrand, 0, 1, args=(a,b))
>>> I
(1.6666666666666667, 1.8503717077085944e-14)
```

Note that there are a number of different integration methods in scipy.integrate!