

Images & turbulence — 17

-> Perturbed wavefront generation

The well-known FFT method allows us to generate phase screens $\varphi(\vec{r})$, where $\vec{r}$ is the two-dimensional position within the phase screen, assuming usually either a Kolmogorov or a von Karman spectrum $\Phi_\varphi(\vec{\nu})$, where $\vec{\nu}$ is the two-dimensional spatial frequency, from which is computed the modulus of $\tilde{\varphi}(\vec{\nu})$, the Fourier transform of $\varphi(\vec{r})$. Assuming the near-field approximation and small phase perturbation [3], the von Karman/Kolmogorov spectrum is given by

$$\Phi_\varphi(\vec{\nu}) = 0.0229 r_0^{-\frac{5}{3}} \left(\nu^2 + \frac{1}{\mathcal{L}_0^2} \right)^{-\frac{11}{6}}, \quad (1)$$

where r_0 is the Fried parameter and $\mathcal{L}_0$ is the wavefront outer scale (infinite for the Kolmogorov model). Within the framework of the classical FFT-based technique, a turbulent phase screen $\varphi_L(\vec{r})$ of physical length L is obtained by taking the inverse FFT of $\tilde{\varphi}_L(\vec{\nu})$, the modulus of which is obtained from Eq. (1) by applying the definition of the power spectrum, which is

$$\begin{aligned} \Phi_\varphi(\vec{\nu}) &= \lim_{L \rightarrow \infty} \left(\frac{\langle |\tilde{\varphi}_L(\nu)|^2 \rangle}{L^2} \right) \\ \Rightarrow |\tilde{\varphi}_L(\nu)| &\simeq L r_0^{-\frac{5}{6}} \sqrt{0.0228} \left(\nu^2 + \frac{1}{\mathcal{L}_0^2} \right)^{-\frac{11}{12}}, \quad (2) \end{aligned}$$

and which phase is random and uniformly distributed.

(From Carbillet & Riccardi, sec. 2: read it as well...)

(the same manipulation as before is applied here in order to obtain the numerical formulation here below.)

The obtained phase screen is thus numerically written

$$\begin{aligned} \varphi_L(i,j) &= \sqrt{2} \sqrt{0.0228} \left(\frac{L}{r_0} \right)^{\frac{5}{6}} \left\{ \text{FFT}^{-1} \left[\left(k^2 + l^2 \right. \right. \right. \\ &\quad \left. \left. \left. + \left(\frac{L}{\mathcal{L}_0} \right)^2 \right)^{-\frac{11}{12}} \exp\{i\theta(k,l)\} \right] \right\}, \quad (3) \end{aligned}$$

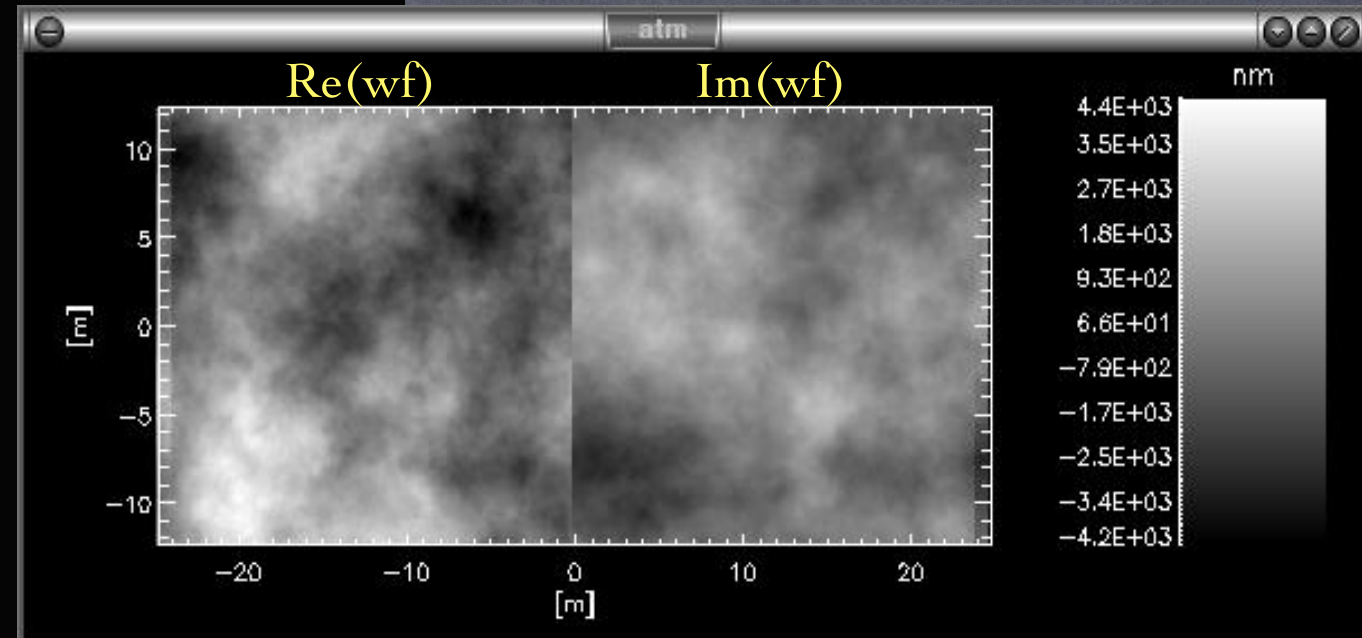
where i and j are the indices in the direct space, k and l are the indices in the FFT space, $\{ \}$ stands for either *real part of* or *imaginary part of*, i is the imaginary unit, and θ is the random uniformly distributed phase (between $-\pi$ and π). The factor $\sqrt{2}$ comes from the fact that here we use both the real and imaginary parts of the original complex generated FFT phase screens, which are independent one from the other [4]. This kind of phase screen suffers, however, from the lack of spatial frequencies lower than the inverse of the necessarily finite length L of the simulated array.

Images & turbulence — 18

```

1 function wfgeneration, dim, length, L0, r0, lambda, SEED=seed
2 ;
3 ; wave-front (wf) generation following von Karman model
4 ; (infinite L0 -Kolmogorov model- not allowed here).
5 ;
6 ; dim      = wf linear dimension [px],
7 ; length   = wf physical length [m],
8 ; L0       = wf outer-scale [m],
9 ; seed     = random generation seed (OPTIONAL),
10 ; r0       = Fried parameter at wavelength 'lambda' [m],
11 ; lambda   = wavelength at which r0 is defined.
12 ;
13 ; Marcel Carillet [marcel.carillet@unice.fr],
14 ; lab. Lagrange (UCA, OCA, CNRS), Feb. 2013.
15 ;
16 ; Last modification: March 2025.
17 ;
18 theta = (randomu(seed,dim,dim)-.5) * 2*!PI ; rnd uniformly distributed phase
19 ; (=argument), between -PI and +PI,
20 ; of the complex number describing
21 ; the FFT of the phase screen phi
22 freq   = dist(dim) ; spatial frequency array
23 modul  = sqrt(2)*sqrt(.0228)*(length/r0)^(5/6.)*(freq^2+(length/L0)^2)^(-11/12.)
24 ; von Kármán model
25 phi    = fft(modul*exp(complex(0,1)*theta), /INVERSE)
26 ; compute phase screen phi
27 wf      = phi*lambda/(2*!PI) ; phase screen phi => wavefront wf
28 wf      -= mean(wf) ; force mean to zero
29
30 return, wf ; deliver 2 independent wf:
31 ; float(wf) & imaginary(wf)
32 end

```



[with, e.g.: dim=100px, L=2m,
r0=10cm, L0=20m, $\lambda=500\text{nm}$]

Images & turbulence — 19

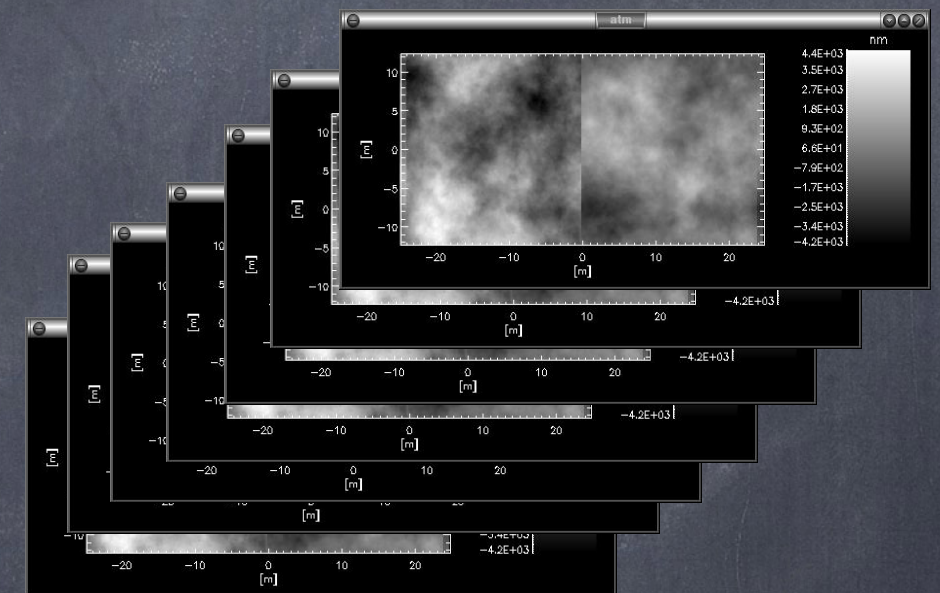
wf generation:

generate a cube of statistically independent wf (typically 100)...

=> compute mean *rms* for different [r_0 , L_0]

[with, e.g: dim=100px, L=2m, $r_0=5|10|20$ cm, $L_0=200|20|2$ m, $\lambda=500$ nm]

```
1 function wfcube2, dim, length, L0, r0, lambda, n_wf, filewf
2
3 ;+
4 ; example of use:
5 ; dim      = 128L      ; [px] wf dimension
6 ; length   = 2.        ; [m] wf physical dimension
7 ; L0       = 27.       ; [m] outerscale of turbulence
8 ; r0       = .1        ; [m] Fried parameter
9 ; lambda   = 500E-9    ; [m] r0 wavelength
10 ; n_wf     = 100L      ; nb of generated wf
11 ; filewf   = 'cube.sav' ; cube of wf filename
12 ;
13 ; print, wfcube2(128L, 2., 27., .1, 500E-9, 100L, 'wf_r0=10cm_L0=10m.sav')*1E9
14 ; -> compute the cube of wf, save it, and print the rms value in nm
15 ;
16 ; sub-routines needed:
17 ; wfgeneration.pro, compute_rms.pro
18 ;
19 ; Marcel Carillet [marcel.carillet@unice.fr],
20 ; lab. Lagrange (UCA, OCA, CNRS), Feb. 2018.
21 ; Last modification: 11th March 2024
22 ;-
23
24 ; preliminary
25 cube = fltarr(dim,dim,n_wf) ; initialize cube of wf
26
27 ; compute and save cube of wf
28 for i=0, n_wf/2-1 do begin ; generate wf
29     wf = wfgeneration(dim, length, L0, r0, lambda, SEED=seed)
30     cube[*,*,2*i] = float(wf)
31     cube[*,*,2*i+1] = imaginary(wf)
32 endfor
33 save, cube, FILE=filewf ; save cube of wf to disk
34
35 ; compute mean rms
36 rms = compute_rms(cube) ; compute rms
37
38 return, rms ; return back
39 end
```



```
function compute_rms, cube
; cube: cube of wavefronts (square wf, no pupil!)

n_wf = (size(cube))[3]
rms = fltarr(n_wf)

for i=0,n_wf-1 do begin
    toto = moment(cube[*,*,i], SDEV=dummy)
    rms[i] = dummy
endfor

rms_moy = mean(rms)

return, rms_moy
end
```

Logbook "Adaptive Optics"

-
+ PSD(r_0 , L_0)
+ $\Rightarrow$ influence of r_0 and L_0
-
+ rms(r_0 , L_0)
+ $\Rightarrow$ influence of r_0 and L_0
-

(à suivre...)

Images & turbulence — 20



$$\Psi(\vec{r}) = A \exp(i\Phi(\vec{r}))$$

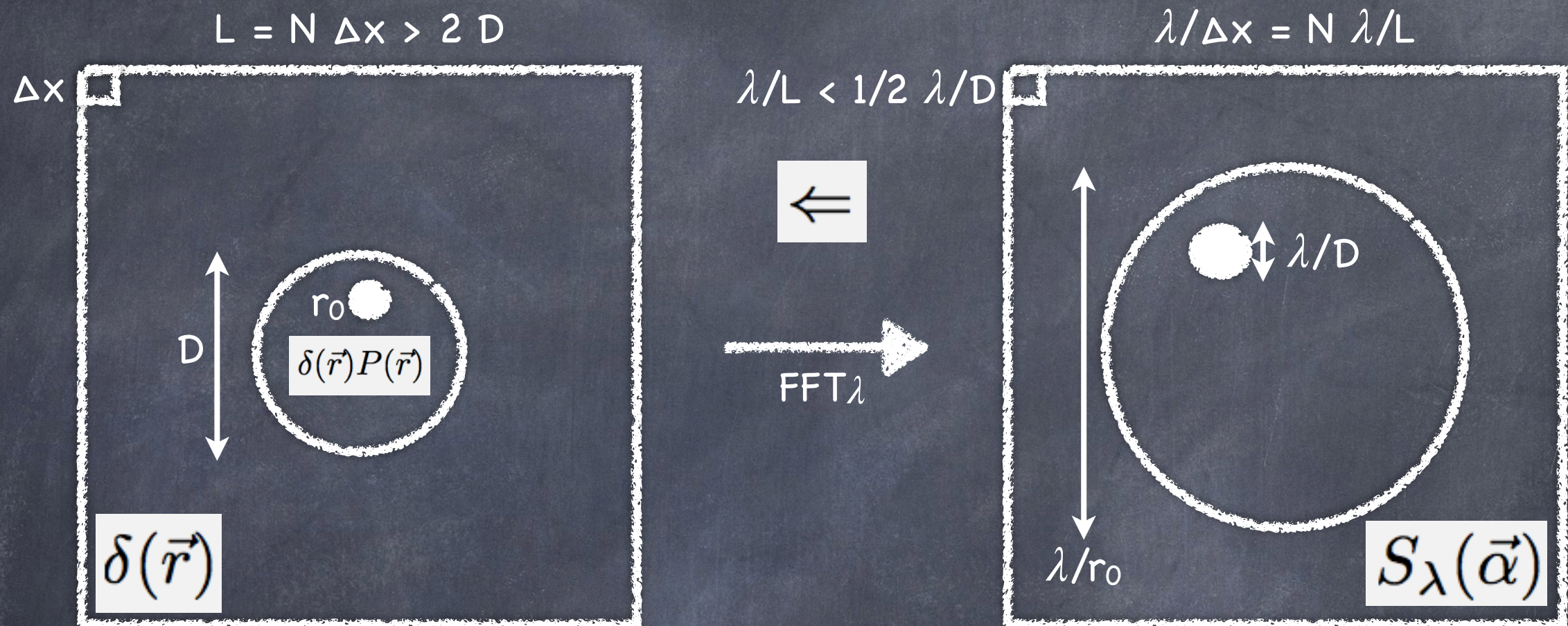
$$\Psi(\vec{r}) \times P(\vec{r}) = A \exp(i\Phi(\vec{r})) \times P(\vec{r})$$

$$S_\lambda(\vec{\alpha}) \propto ||FT\{\Psi(\vec{r}) \times P(\vec{r})\}||^2$$

$$\Rightarrow S_\lambda(\vec{\alpha}) \propto ||FT\{A \exp(i\Phi(\vec{r})) \times P(\vec{r})\}||^2$$

$$A = 1 \text{ and } \Psi(\vec{r}) = \frac{2\Pi}{\lambda} \delta(\vec{r}) \Rightarrow S_\lambda(\vec{\alpha}) \propto ||FT\{\exp\left(i\frac{2\Pi}{\lambda} \delta(\vec{r})\right) \times P(\vec{r})\}||^2$$

Images & turbulence — 21



Shannon (=Nyquist) criterium

=> the image pixel λ/L must be at most half the resolution element (resel!) λ/D
 (in other words : one must have AT LEAST 2 image pixels per λ/D)

=> the simulated wavefronts must be at least twice the telescope diameter ($L > 2D$)

In addition

- λ/r_0 should be smaller than $\lambda/\Delta x$ (=> N large enough)

Images & turbulence — 22

```
function wfimg2, diam, obs, lambda_psf, filewf, filepsf
;+
; example of use:
;   diam      = 64L           ; [px] telescope pupil dimension
;   obs       = 0. [0-1]     ; (linear) obscuration ratio
;   lambda_psf= 500E-9        ; [m] PSF wavelength
;   filewf    = 'cube.sav'    ; cube of wf filename
;   filepsf   = 'cube_psf.sav'; cube of PSFs filename
;   print, wfimg2(diam,obs,lambda_psf,filewf,filepsf)
;   -> compute the cube of PSFs, save it, and tell how it went
;
; sub-routines needed: make_PSF.pro, wfgeneration.pro, makepup.pro
;
; Marcel Carillet [marcel.carillet@unice.fr], Lagrange (UniCA, OCA, CNRS)
; written: Feb. 2018, last modified: March 11th 2024.
;-

; preliminaries
restore, filewf           ; restores variable 'cube' containing nn wf
dim= (size(cube))(1)      ; linear size of wf
nn = (size(cube))(3)      ; nb of wf
cube_psf=filtarr(dim,dim,nn) ; initialize cube of PSFs

; compute and save PSFs
pup = makepup(dim,diam,obs) ; compute entrance pupil
for i=0, nn-1L do cube_psf[:,*,i] = make_PSF(pup,cube[:,*,i],lambda_psf)
                                ; compute the PSF corresponding to each wf
save, cube_psf, FI=filepsf    ; save cube of PSFs to disk

; return back
return, 'Cube of PSFs '+filepsf+' saved on disk...'
end
```

image formation:

1- cube of instantaneous PSFs (500nm & H-band)

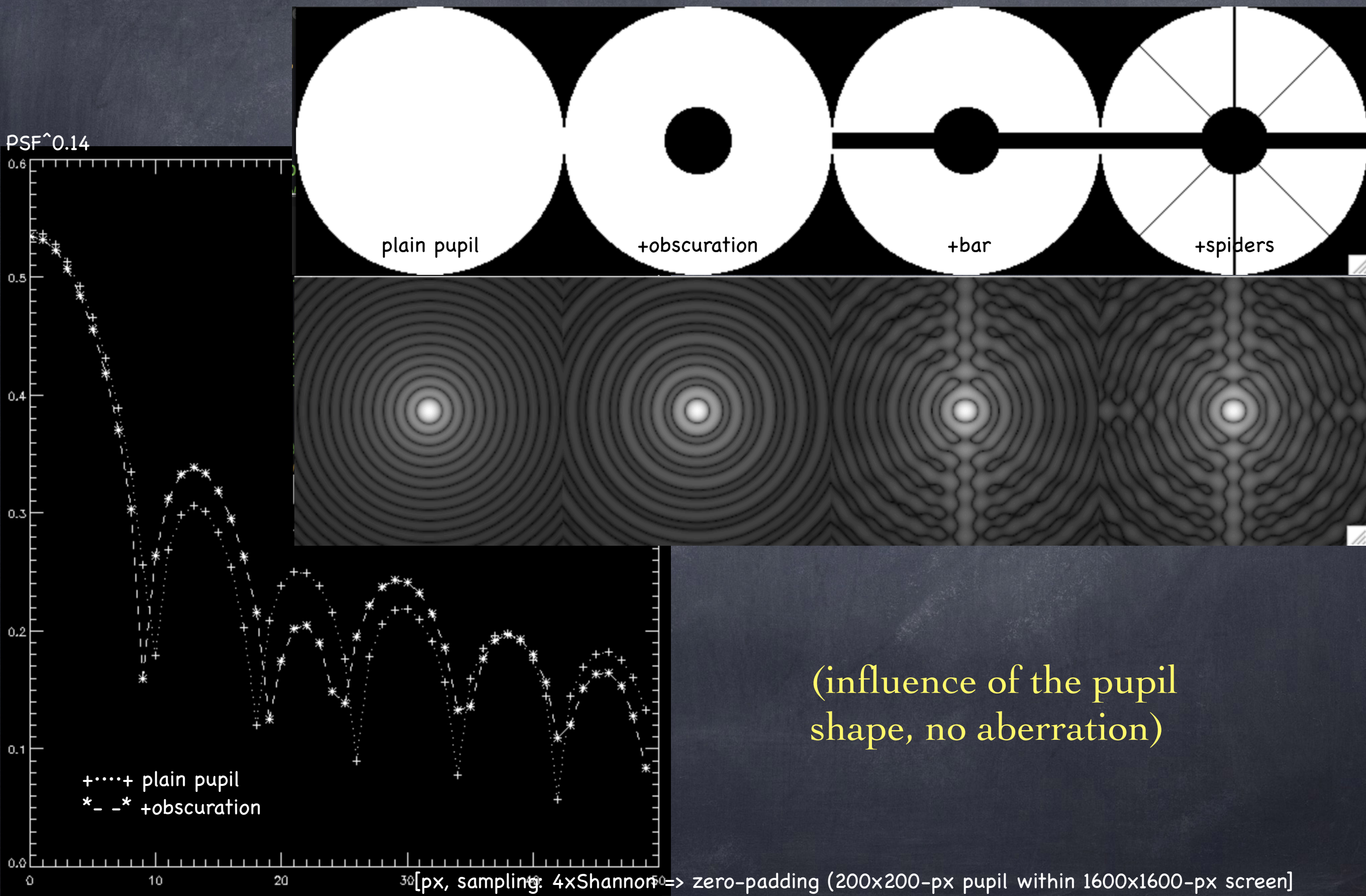
```
function make_PSF, pup, wf, lambda
;+
; PSF computation from a wavefront
;
; pup      = input pupil,
; wf       = input wavefront [float],
; lambda   = wavelength at which PSF is computed.
; PSF = make_PSF(pup, wf, lambda)
; -> compute the PSF corresponding to wf and pup, at wavelength lambda
;
; Marcel Carillet [marcel.carillet@unice.fr],
; UMR 7293 Lagrange (UNS/CNRS/OCA), Feb. 2013.
; Last modification: March 11th 2024
;-

; preliminary
dim = (size(wf))[1]

; compute PSF
psf = (abs(fft(pup*exp(complex(0,1)*2*!PI/lambda*wf*pup))))^2
; NB: (abs(fft(pup*exp(complex(0,1)*2*!PI/lambda*wf))))^2 would suffice
psf = shift(psf, dim/2, dim/2)

; return back
return, psf
end
```

Images & turbulence — 23



Images & turbulence — 24

```

[IDL> restore, 'PSF_r0=10cm_L0=10m_lambda=500nm.sav'
[IDL> help
% At $MAIN$
CUBE_PSF      FLOAT      = Array[128, 128, 100]
I             INT        =      100
Compiled Procedures:
  $MAIN$

Compiled Functions:
  COMPUTE_RMS DIST      MAKEPUP      MAKE_PSF      WFCUBE2      WFGENERATION      WFIMG2

[IDL> window, XS=512, YS=512, /FREE
[IDL> for i=0,99 do tvscl, rebin(cube_PSF[:, :, i], 512, 512, /SAMPLE)

[IDL> longexp=total(cube_PSF, 3)
[IDL> tvscl, rebin(longexp, 512, 512, /SAMPLE)^.1
    
```

$$\lambda/L = 1/2 \lambda/D$$

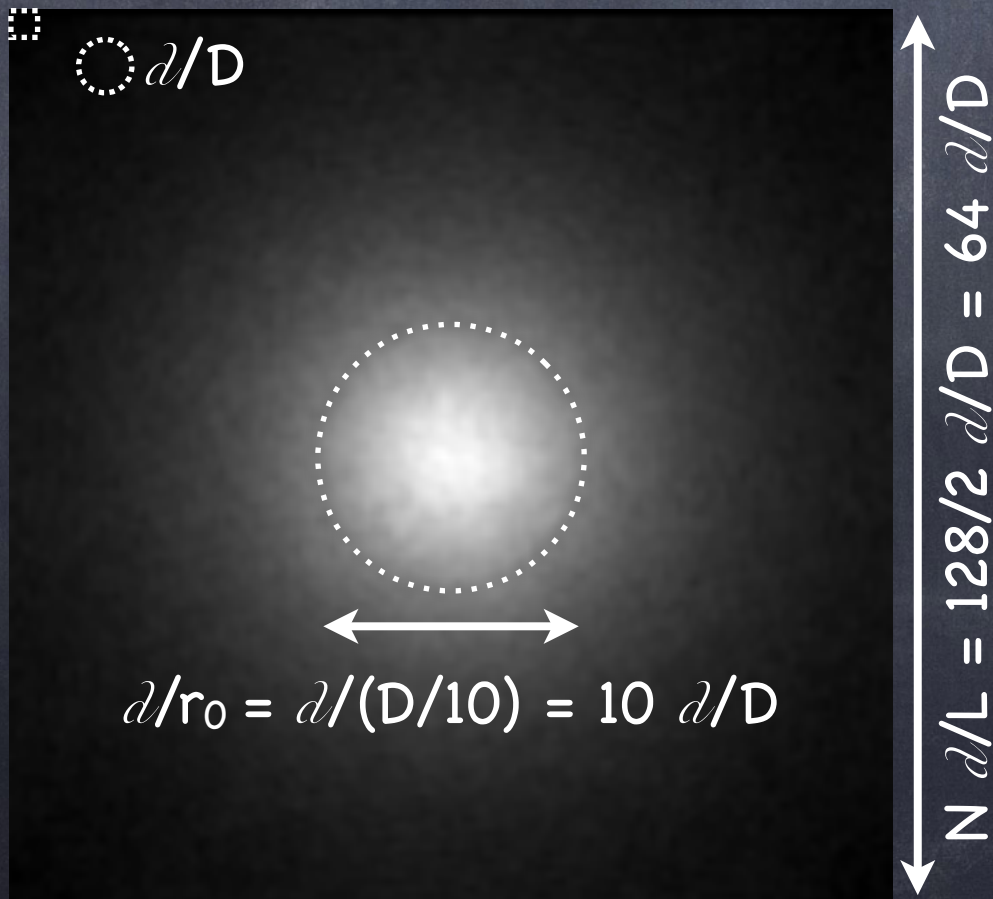


image formation:

- 1- cube of instantaneous PSFs (500nm & band H)
- 2- long-exposure PSF

Images & turbulence — 25

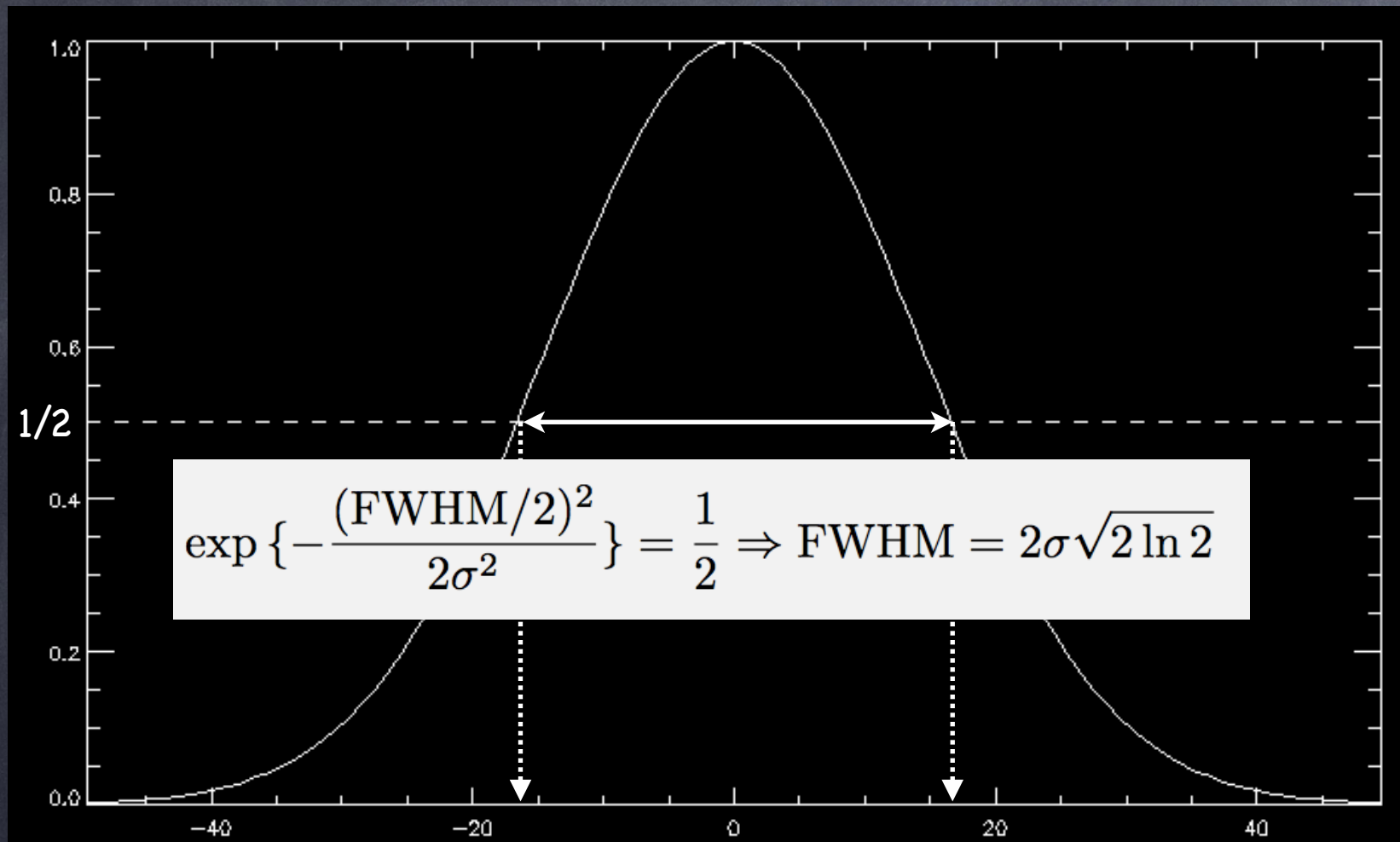


image formation:

1- cube of instantaneous PSFs (500nm & H-band)
2- long-exposure PSFs
3- fit with gaussian and compare FWHM vs. λ/r_0 (seeing), also in function of the outerscale L_0 .

→ Also read Martinez...

```
[IDL> res=gauss2dfit(longexp,a)
% Program caused arithmetic error: Floating underflow
[IDL> print, 2*((a[2]+a[3])/2)*sqrt(2*a*log(2))
      15.9637
IDL> █
```

In this example, the FWHM is $\approx 16\text{px}$ and, since we have here: $1\text{px}=(\lambda/D)/2$, we have hence: $\text{FWHM} \approx 8 (\lambda/D)$ [i.e. $8*0.1'' \approx 0.8''$ here (@500nm)]

Images & turbulence — 26

On the Difference between Seeing and Image Quality: When the Turbulence Outer Scale Enters the Game

Patrice Martinez¹
Johann Kolb¹
Marc Sarazin¹
Andrei Tokovinin²

¹ ESO

² Cerro-Tololo Inter American Observatory,
Chile

We attempt to clarify the frequent confusion between seeing and image quality for large telescopes. The full width at half maximum of a stellar image is commonly considered to be equal to the atmospheric seeing. However the outer scale of the turbulence, which corresponds to a reduction in the low frequency content of the phase perturbation spectrum, plays a significant role in the improvement of image quality at the focus of a telescope. The image quality is therefore different (and in some cases by a large factor) from the atmospheric seeing that can be measured by dedicated seeing monitors, such as a differential image motion monitor.

of telescope diameters and wavelengths. We show that this dependence is efficiently predicated by a simple approximate formula introduced in the literature in 2002. The practical consequences for operation of large telescopes are discussed and an application to on-sky data is presented.

Background and definitions

In practice the resolution of ground-based telescopes is limited by the atmospheric turbulence, called “seeing”. It is traditionally characterised by the Fried parameter (r_0) – the diameter of a telescope such that its diffraction-limited resolution equals the seeing resolution. The well-known Kolmogorov turbulence model describes the shape of the atmospheric long-exposure point spread function (PSF), and many other phenomena, by this single parameter r_0 . This model predicts the dependence¹ of the PSF FWHM (denoted ϵ_0) on wavelength (λ) and inversely on the Fried parameter, r_0 , where r_0 depends on wavelength (to

A finite L_0 reduces the variance of the low order modes of the turbulence, and in particular decreases the image motion (the tip-tilt). The result is a decrease of the FWHM of the PSF. In the von Kàrmàn model, r_0 describes the high frequency asymptotic behaviour of the spectrum where L_0 has no effect, and thus r_0 loses its sense of an equivalent wavefront coherence diameter. The differential image motion monitors (DIMM; Sarazin & Roddier, 1990) are devices that are commonly used to measure the seeing at astronomical sites. The DIMM delivers an estimate of r_0 based on measuring wavefront distortions at scales of ~ 0.1 m, where L_0 has no effect. By contrast, the absolute image motion and long-exposure PSFs are affected by large-scale distortions and depend on L_0 . In this context the Kolmogorov expression for ϵ_0 ¹ is therefore no longer valid.

Proving the von Kàrmàn model experimentally would be a difficult and eventually futile goal as large-scale wavefront perturbations are anything but stationary. However, the increasing number of esti-

-- -- -- -- --
Logbook "Adaptive Optics"
-- -- -- -- --

-
+ PSD(r_0 , L_0)
+ $\Rightarrow$ influence of r_0 and L_0
-
+ rms(r_0 , L_0)
+ $\Rightarrow$ influence of r_0 and L_0
-
+ FWHM(r_0 or $\lambda \Rightarrow r_0$, L_0)
+ $\Rightarrow$ influence of r_0 and L_0
+ $\Rightarrow$ comparison with the "seeing" λ/r_0

(à suivre...)