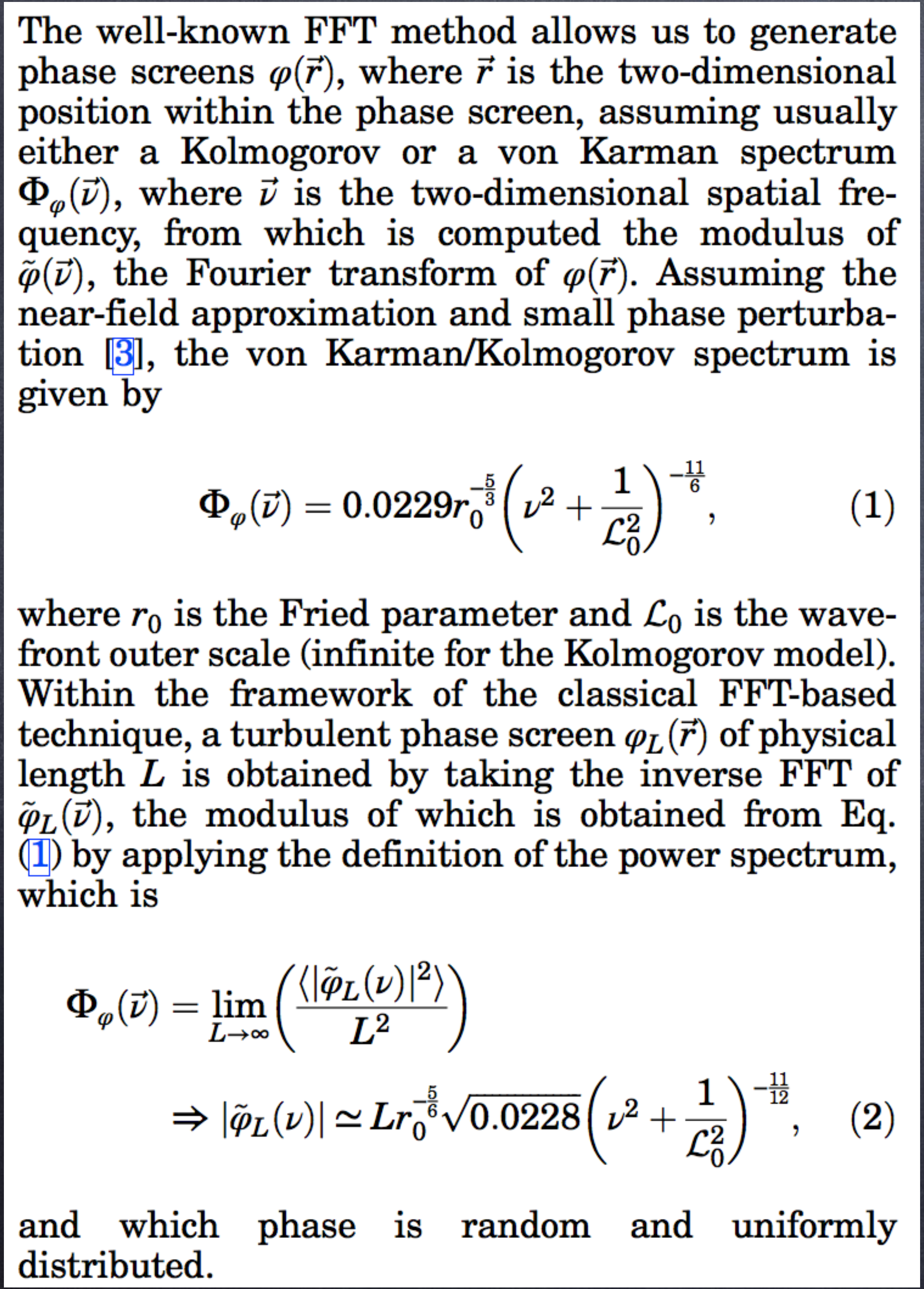
Images & turbulence - 16

-> 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

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

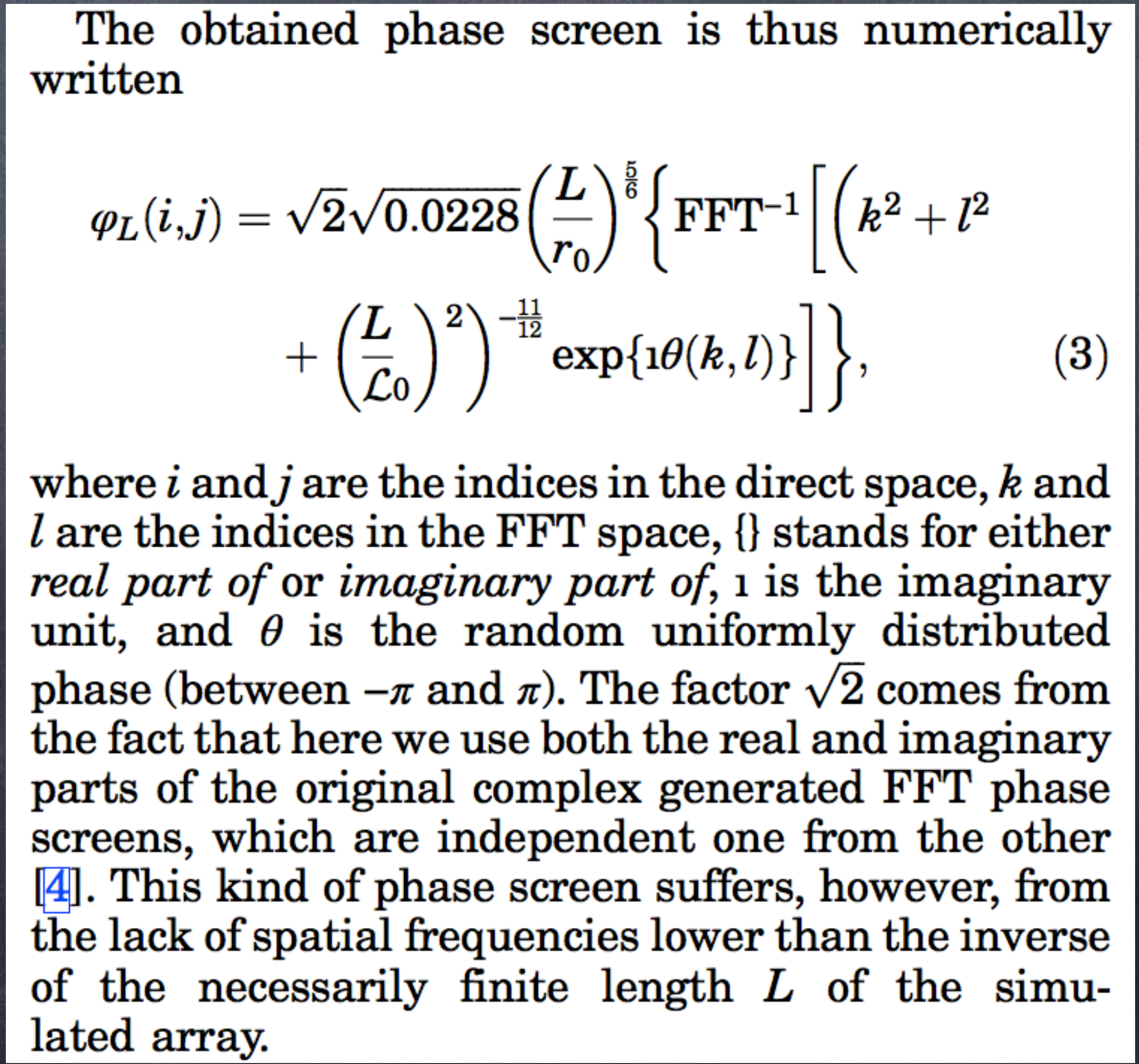
where r_0 is the Fried parameter and $\mathcal{L}_0$ is the wavefront outer scale (infinite for the Kolmogorov model). Within the framework of the classical FFT-based technique, a turbulent phase screen $\varphi_L(\vec{r})$ of physical length L is obtained by taking the inverse FFT of $\tilde{\varphi}_L(\vec{\nu})$,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

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

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.)

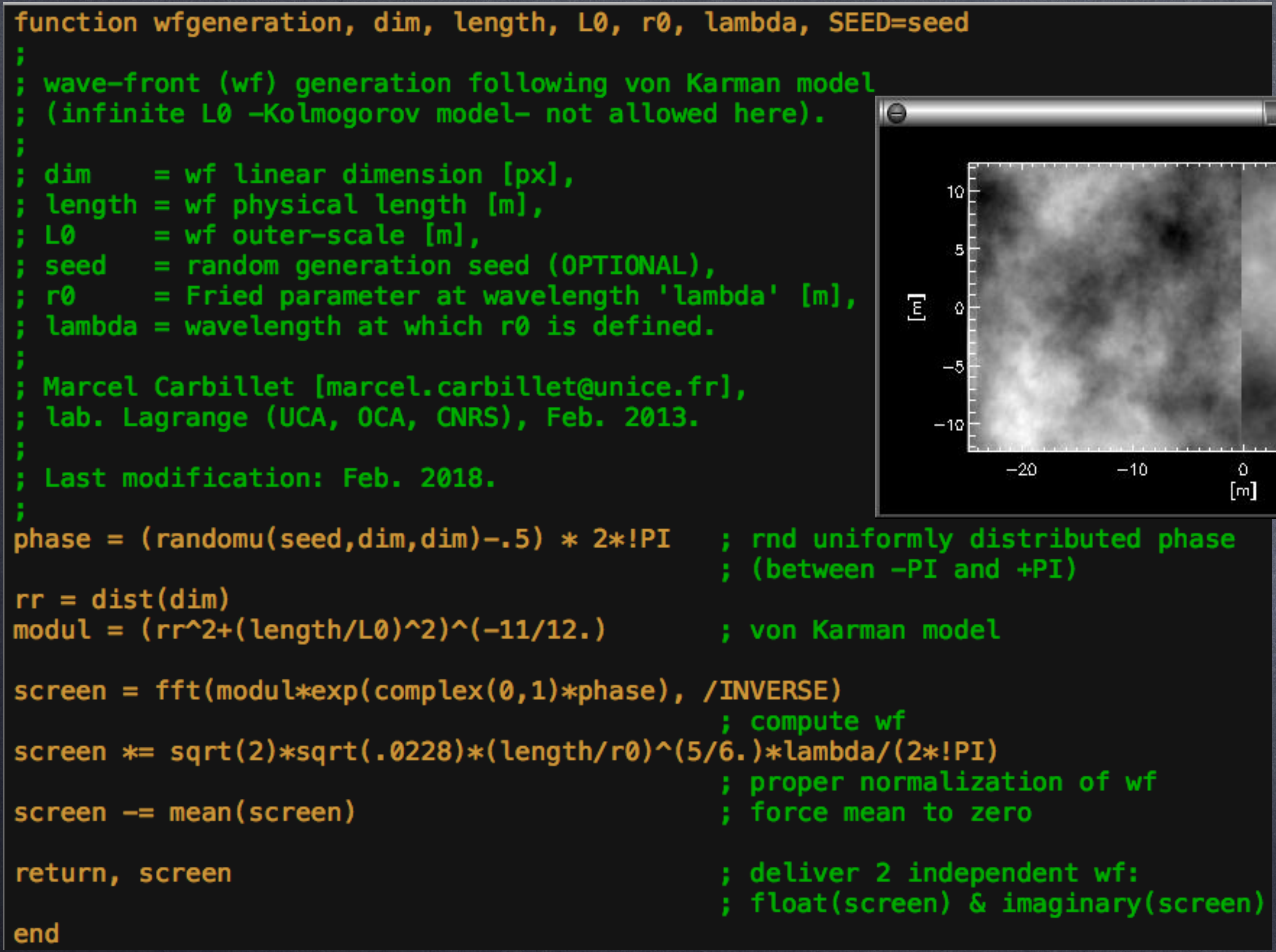


The obtained phase screen is thus numerically written

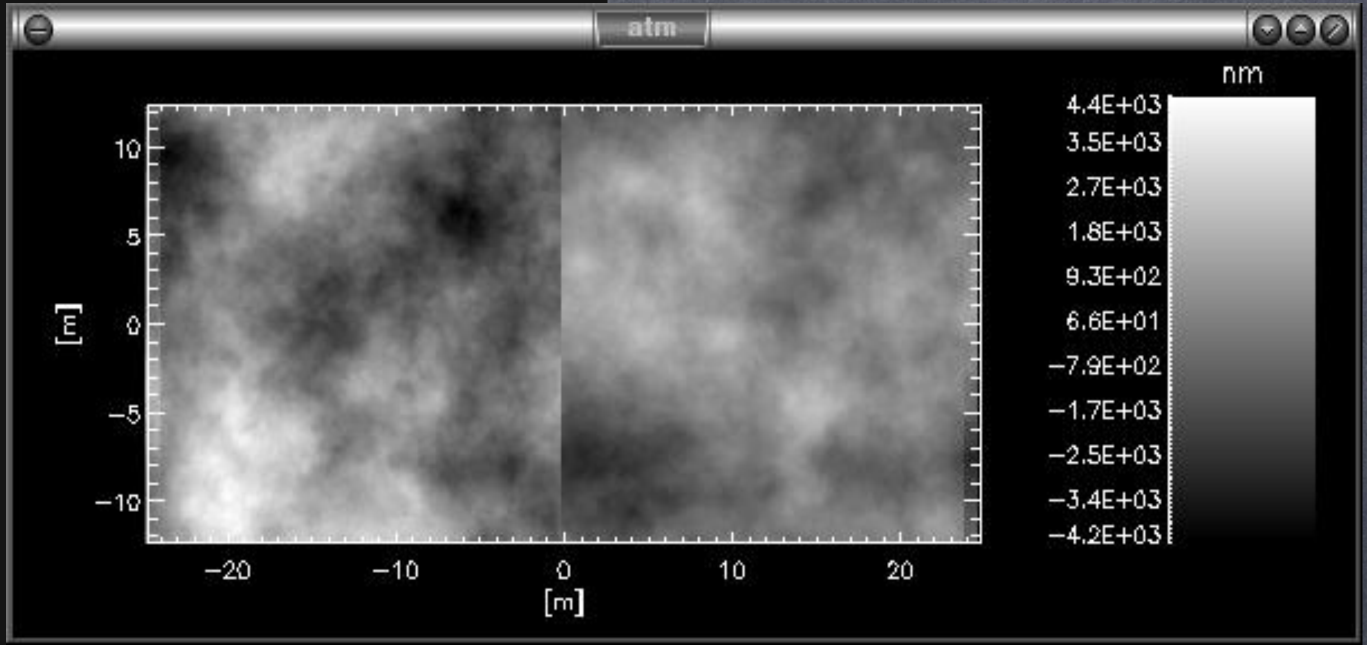
$$\varphi_L(i,j) = \sqrt{2}\sqrt{0.0228}\left(\frac{L}{r_0}\right)^{\frac{5}{6}} \left\{ \mathrm{FFT}^{-1}\left[\left(k^2 + l^2 + \left(\frac{L}{\mathcal{L}_0}\right)^2\right)^{-\frac{11}{12}} \exp\{\imath\theta(k,l)\}\right]\right\}, \qquad (3)$$

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*, $\imath$ 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 - 17

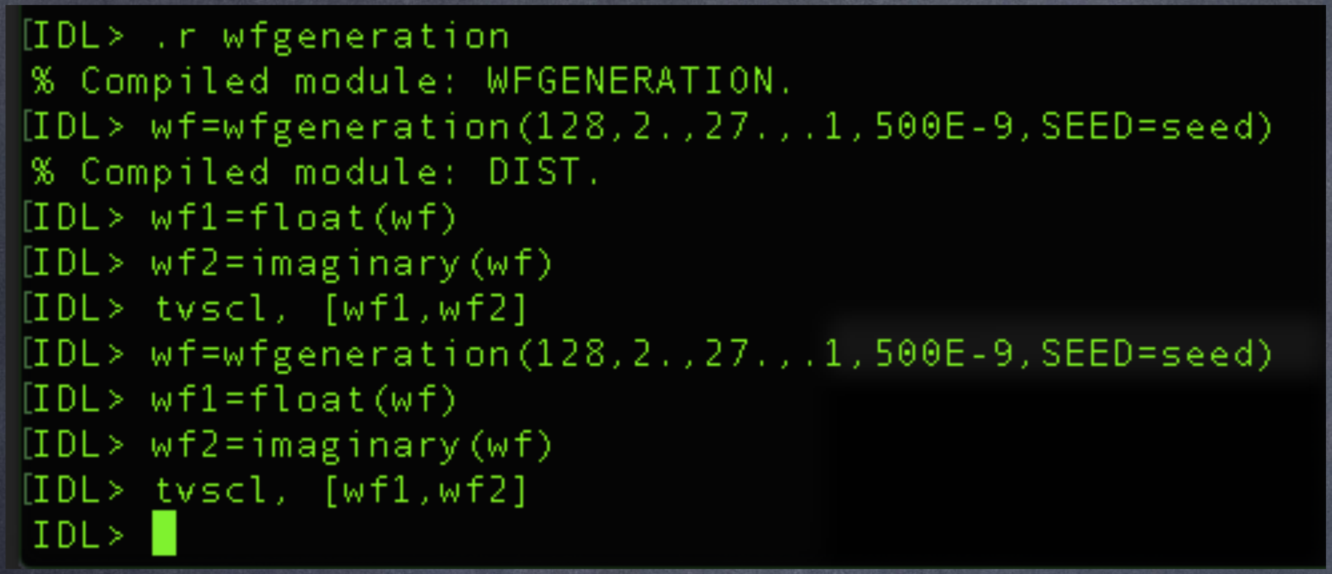


```
function wfgeneration, dim, length, L0, r0, lambda, SEED=seed
;
; wave-front (wf) generation following von Karman model
; (infinite L0 -Kolmogorov model- not allowed here).
;
; dim    = wf linear dimension [px],
; length = wf physical length [m],
; L0     = wf outer-scale [m],
; seed   = random generation seed (OPTIONAL),
; r0     = Fried parameter at wavelength 'lambda' [m],
; lambda = wavelength at which r0 is defined.
;
; Marcel Carbillet [marcel.carbillet@unice.fr],
; lab. Lagrange (UCA, OCA, CNRS), Feb. 2013.
;
; Last modification: Feb. 2018.
;
phase = (randomu(seed,dim,dim)-.5) * 2*!PI    ; rnd uniformly distributed phase
                                              ; (between -PI and +PI)
rr = dist(dim)
modul = (rr^2+(length/L0)^2)^(-11/12.)        ; von Karman model

screen = fft(modul*exp(complex(0,1)*phase), /INVERSE)
                                              ; compute wf
screen *= sqrt(2)*sqrt(.0228)*(length/r0)^(5/6.)*lambda/(2*!PI)
                                              ; proper normalization of wf
screen -= mean(screen)                        ; force mean to zero

return, screen                                ; deliver 2 independent wf:
                                              ; float(screen) & imaginary(screen)
end
```

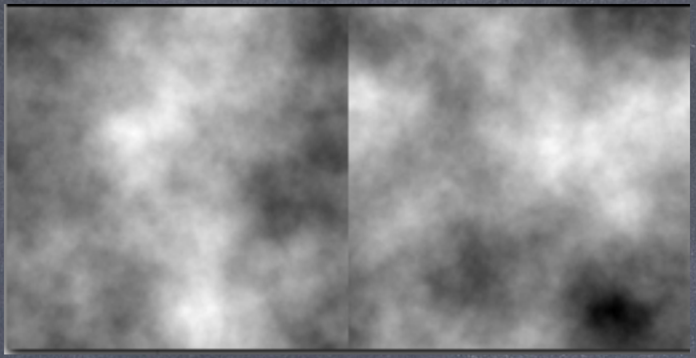


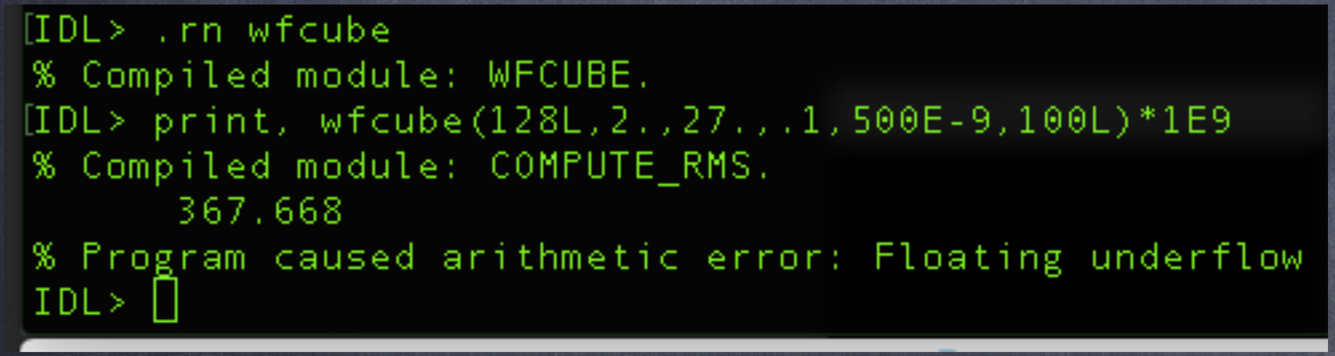
<u>wf generation:</u>
generate a cube of statistically independent wf (typically 100)... => compute mean *rms* for different [r_0, L_0]

Images & turbulence - 18

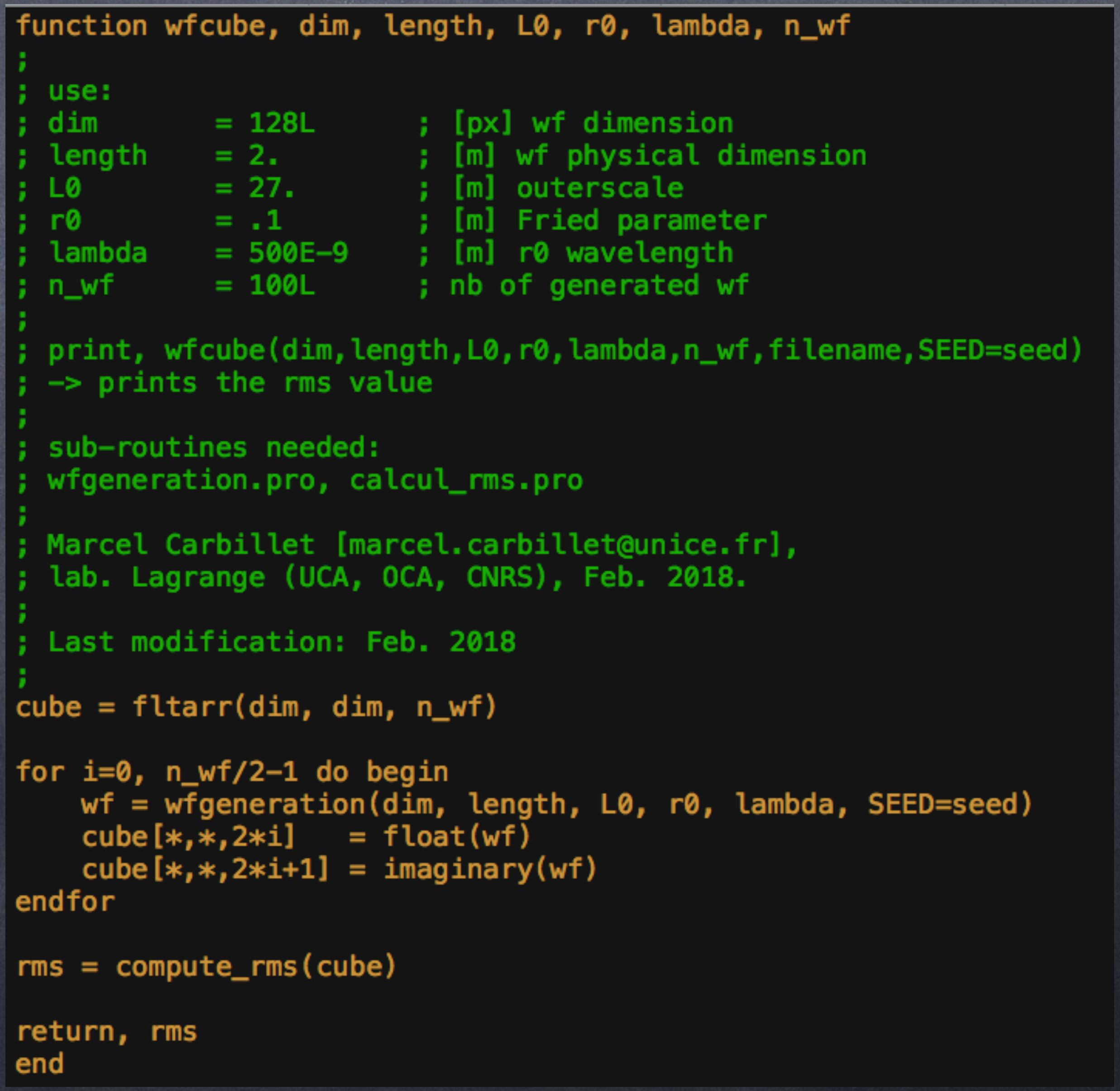


```
[IDL> .r wfgeneration
% Compiled module: WFGENERATION.
[IDL> wf=wfgeneration(128,2.,27.,.1,500E-9,SEED=seed)
% Compiled module: DIST.
[IDL> wf1=float(wf)
[IDL> wf2=imaginary(wf)
[IDL> tvscl, [wf1,wf2]
[IDL> wf=wfgeneration(128,2.,27.,.1,500E-9,SEED=seed)
[IDL> wf1=float(wf)
[IDL> wf2=imaginary(wf)
[IDL> tvscl, [wf1,wf2]
 IDL> 
```

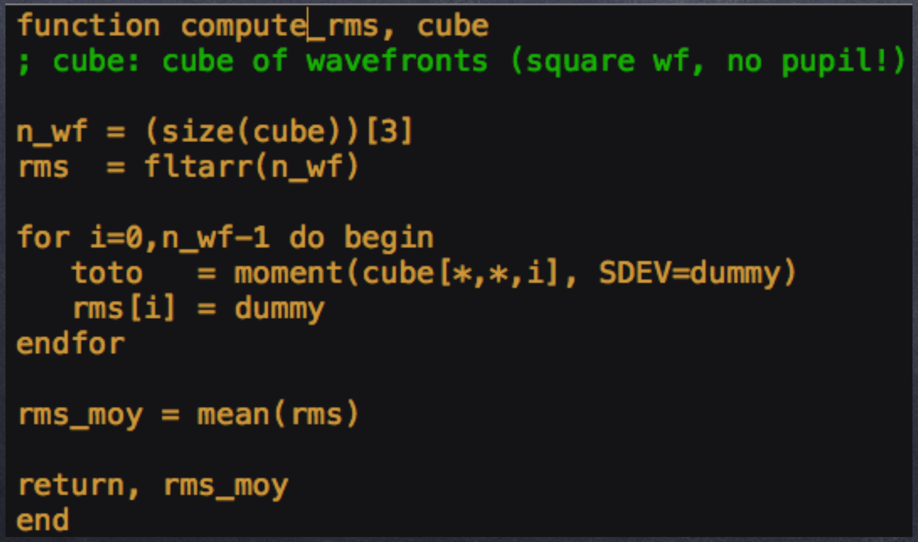




```
[IDL> .rn wfcube
% Compiled module: WFCUBE.
[IDL> print, wfcube(128L,2.,27.,.1,500E-9,100L)*1E9
% Compiled module: COMPUTE_RMS.
      367.668
% Program caused arithmetic error: Floating underflow
 IDL> 
```

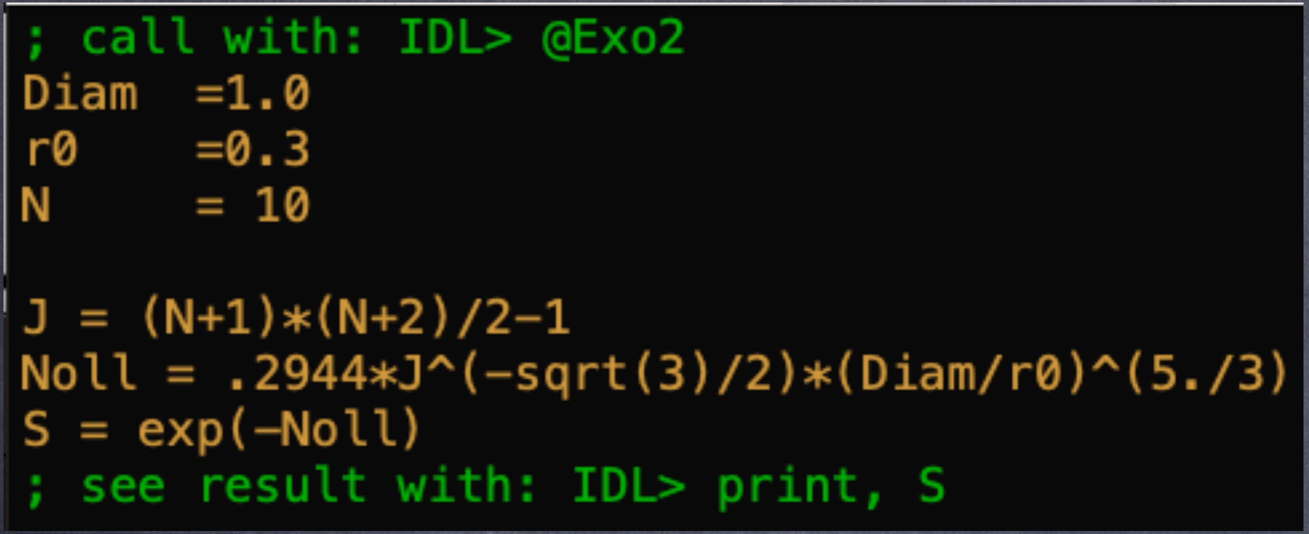


```
function wfcube, dim, length, L0, r0, lambda, n_wf
;
; use:
; dim       = 128L      ; [px] wf dimension
; length    = 2.        ; [m] wf physical dimension
; L0        = 27.       ; [m] outerscale
; r0        = .1        ; [m] Fried parameter
; lambda    = 500E-9    ; [m] r0 wavelength
; n_wf      = 100L      ; nb of generated wf
;
; print, wfcube(dim,length,L0,r0,lambda,n_wf,filename,SEED=seed)
; -> prints the rms value
;
; sub-routines needed:
; wfgeneration.pro, calcul_rms.pro
;
; Marcel Carbillet [marcel.carbillet@unice.fr],
; lab. Lagrange (UCA, OCA, CNRS), Feb. 2018.
;
; Last modification: Feb. 2018
;
cube = fltarr(dim, dim, n_wf)

for i=0, n_wf/2-1 do begin
    wf = wfgeneration(dim, length, L0, r0, lambda, SEED=seed)
    cube[*,*,2*i]   = float(wf)
    cube[*,*,2*i+1] = imaginary(wf)
endfor

rms = compute_rms(cube)

return, rms
end
```



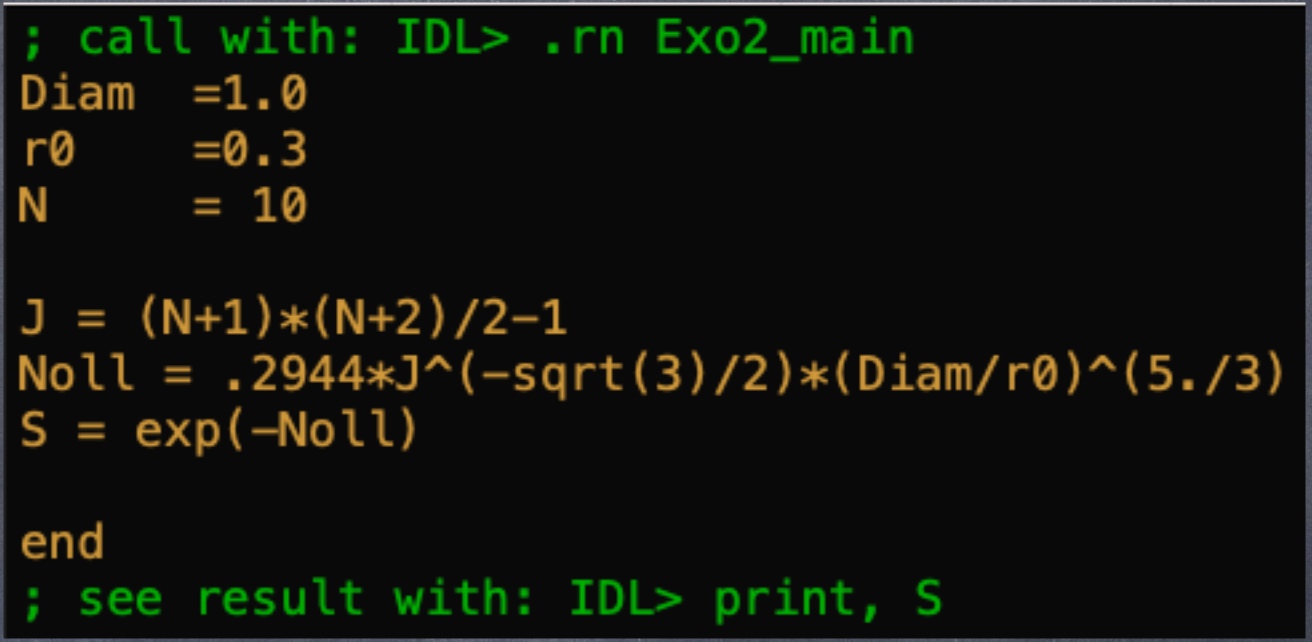
```
function compute_rms, cube
; cube: cube of wavefronts (square wf, no pupil!)

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
```

(IDL: 4 kind of routines/scripts)



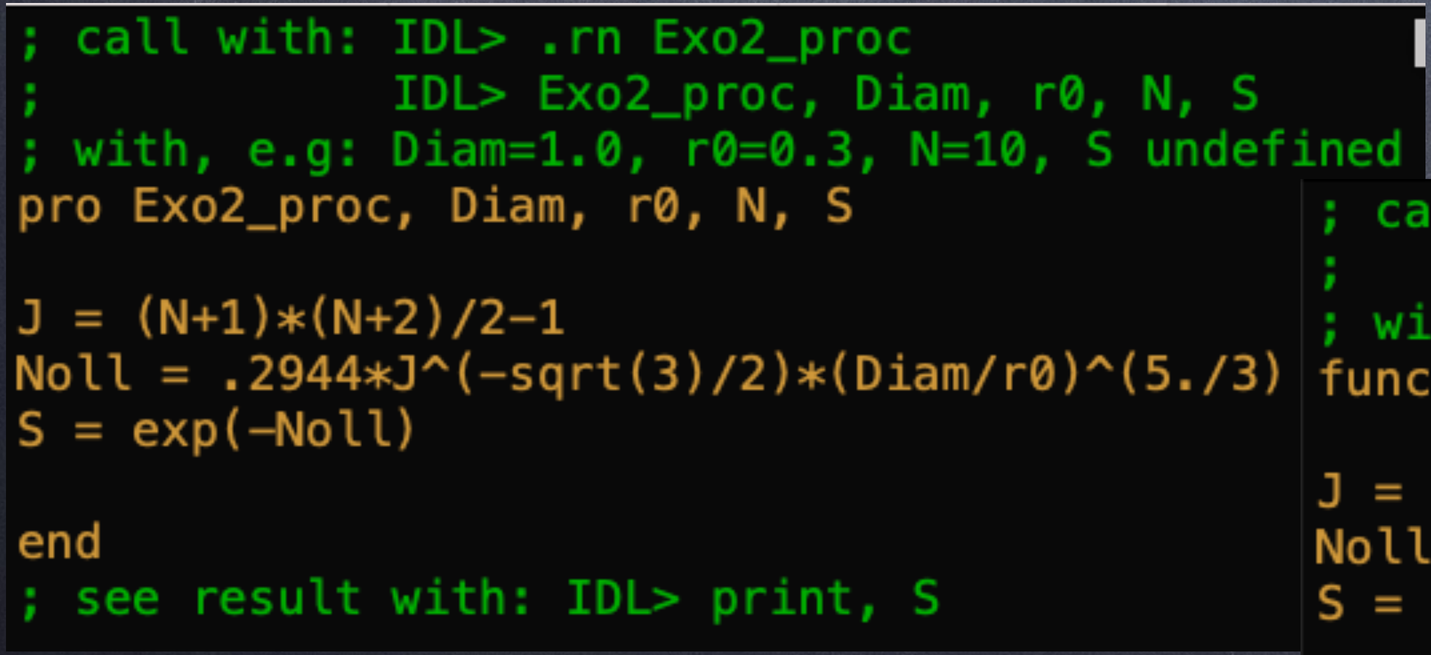
```
; call with: IDL> @Exo2
Diam  =1.0
r0    =0.3
N     = 10

J = (N+1)*(N+2)/2-1
Noll = .2944*J^(-sqrt(3)/2)*(Diam/r0)^(5./3)
S = exp(-Noll)
; see result with: IDL> print, S
```

batch: all variables are accessible.



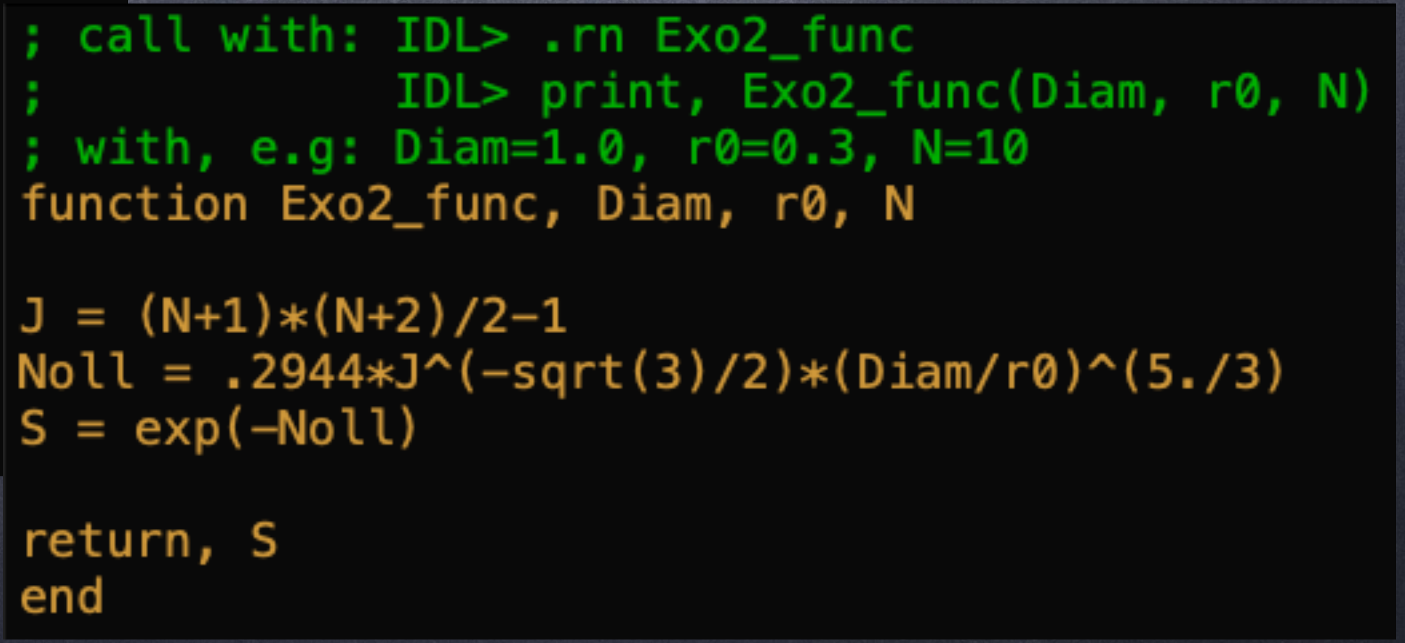
```
; call with: IDL> .rn Exo2_main
Diam  =1.0
r0    =0.3
N     = 10

J = (N+1)*(N+2)/2-1
Noll = .2944*J^(-sqrt(3)/2)*(Diam/r0)^(5./3)
S = exp(-Noll)

end
; see result with: IDL> print, S
```

main: idem (« .r » : run ; « .rn » : run new).



```
; call with: IDL> .rn Exo2_proc
;             IDL> Exo2_proc, Diam, r0, N, S
; with, e.g: Diam=1.0, r0=0.3, N=10, S undefined
pro Exo2_proc, Diam, r0, N, S

J = (N+1)*(N+2)/2-1
Noll = .2944*J^(-sqrt(3)/2)*(Diam/r0)^(5./3)
S = exp(-Noll)

end
; see result with: IDL> print, S
```

procedure: (input/output) parameters are accessible, but variables defined within the procedure are not.



```
; call with: IDL> .rn Exo2_func
;             IDL> print, Exo2_func(Diam, r0, N)
; with, e.g: Diam=1.0, r0=0.3, N=10
function Exo2_func, Diam, r0, N

J = (N+1)*(N+2)/2-1
Noll = .2944*J^(-sqrt(3)/2)*(Diam/r0)^(5./3)
S = exp(-Noll)

return, S
end
```

function: no output parameters, inside variables not accessible, result of the function returned.

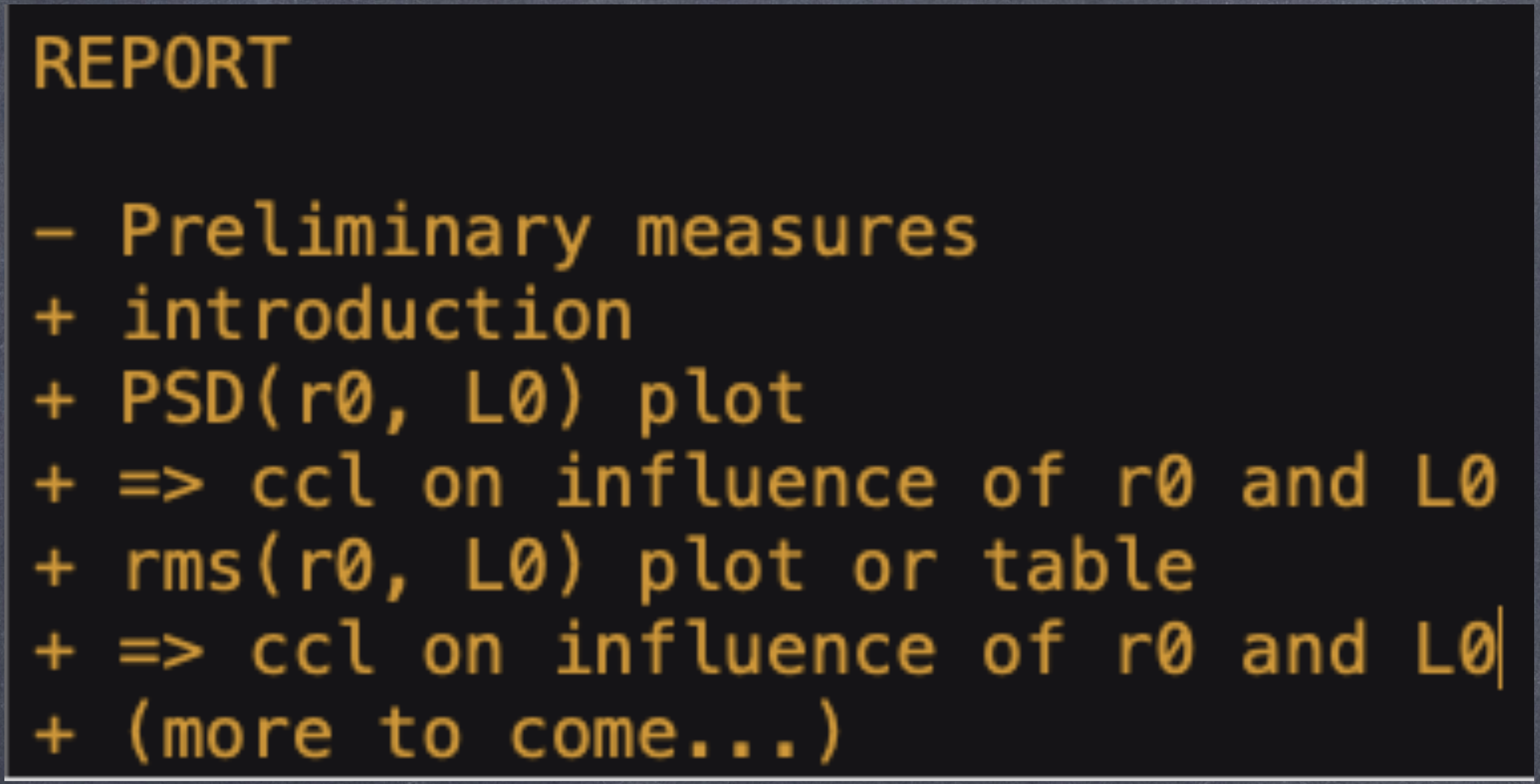
(IDL: other useful remarks)

- IDL help is called with: IDL>> ?
- '?' opens with a defined browser the file 'idl.htm', here: **.../exelis/.../idl/idl.htm**
- This file can also be found with the unix command 'find':

  unix>> cd /

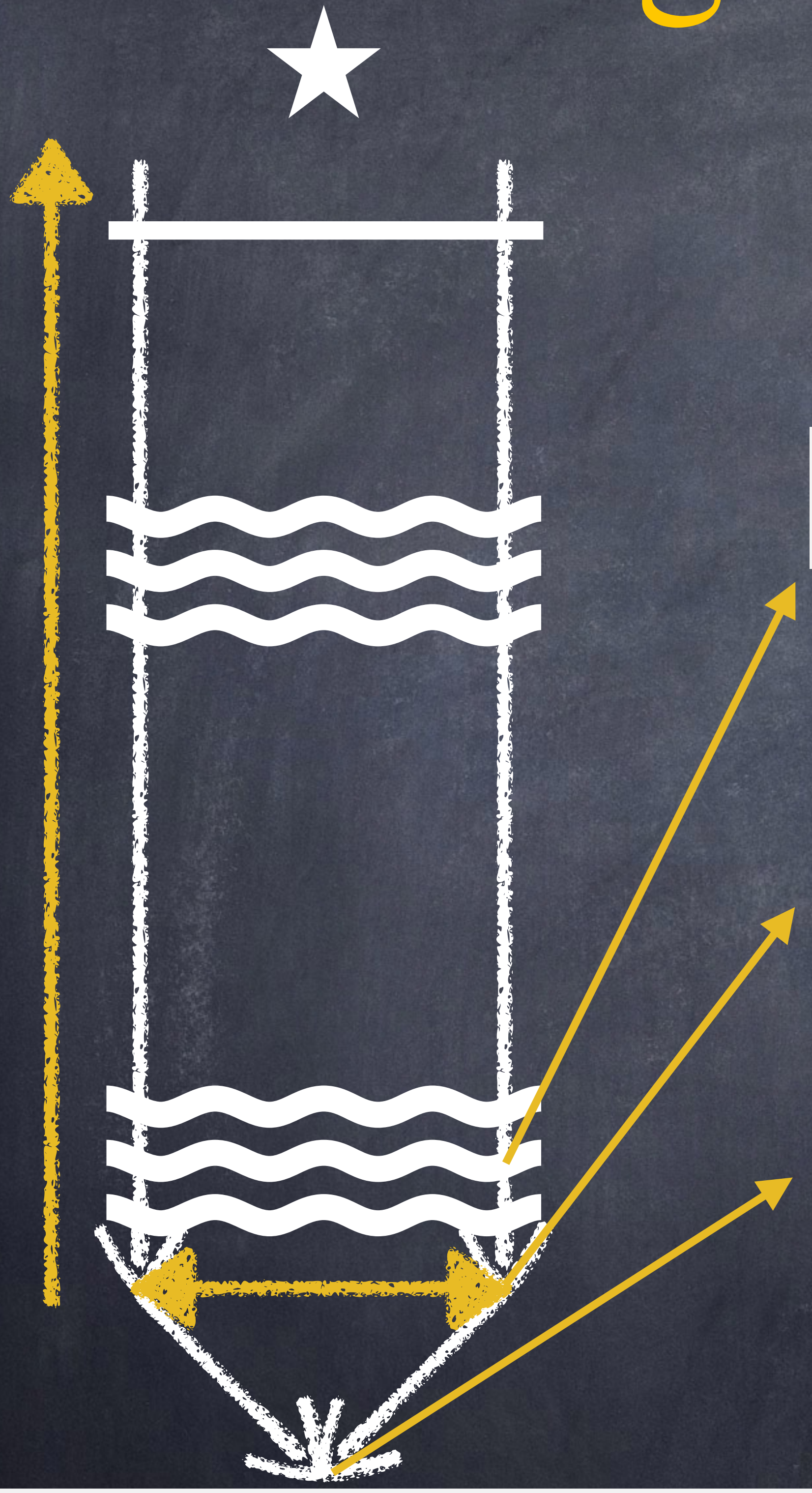
  unix>> find . -name idl.htm
- See also (for routines which are part of a third library): **IDL>> doc_library, 'routine_name'**
- Return to main level of programming after a crash: **retall**
- Details on a variable xxx: **idl> help, xxx**

  (see all variables: **idl> help**)
- Close last opened window: **idl> wdelete**



```
REPORT

- Preliminary measures
+ introduction
+ PSD(r0, L0) plot
+ => ccl on influence of r0 and L0
+ rms(r0, L0) plot or table
+ => ccl on influence of r0 and L0
+ (more to come...)
```

Images & turbulence - 19



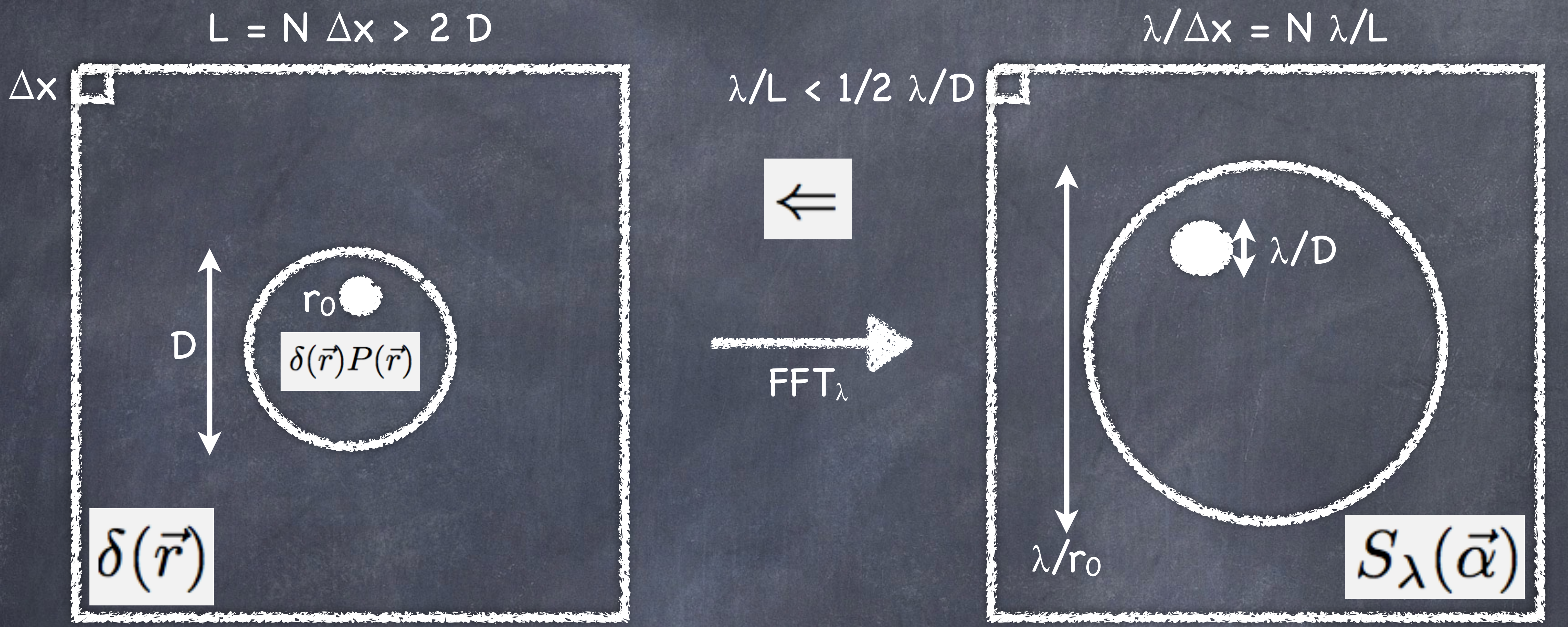
$$\Psi(\vec{r}) = A\ \exp\left(\imath\Phi(\vec{r})\right)$$

$$P(\vec{r}) \Rightarrow A\ P(\vec{r})\ \exp\left(\imath\Phi(\vec{r})P(\vec{r})\right)$$

$$S_\lambda(\vec{\alpha}) \propto \|FT\{A\ P(\vec{r})\ \exp\left(\imath\Phi(\vec{r})P(\vec{r})\right)\}\|^2$$

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

Images & turbulence - 20



$L = N \Delta x > 2D$
$\lambda/\Delta x = N \lambda/L$
Δx
$\lambda/L < 1/2\ \lambda/D$
$\Leftarrow$
D
r_0
$\delta(\vec{r})P(\vec{r})$
FFT_λ
λ/D
λ/r_0
$\delta(\vec{r})$
$S_\lambda(\vec{\alpha})$

Shannon (=Nyquist) criterium

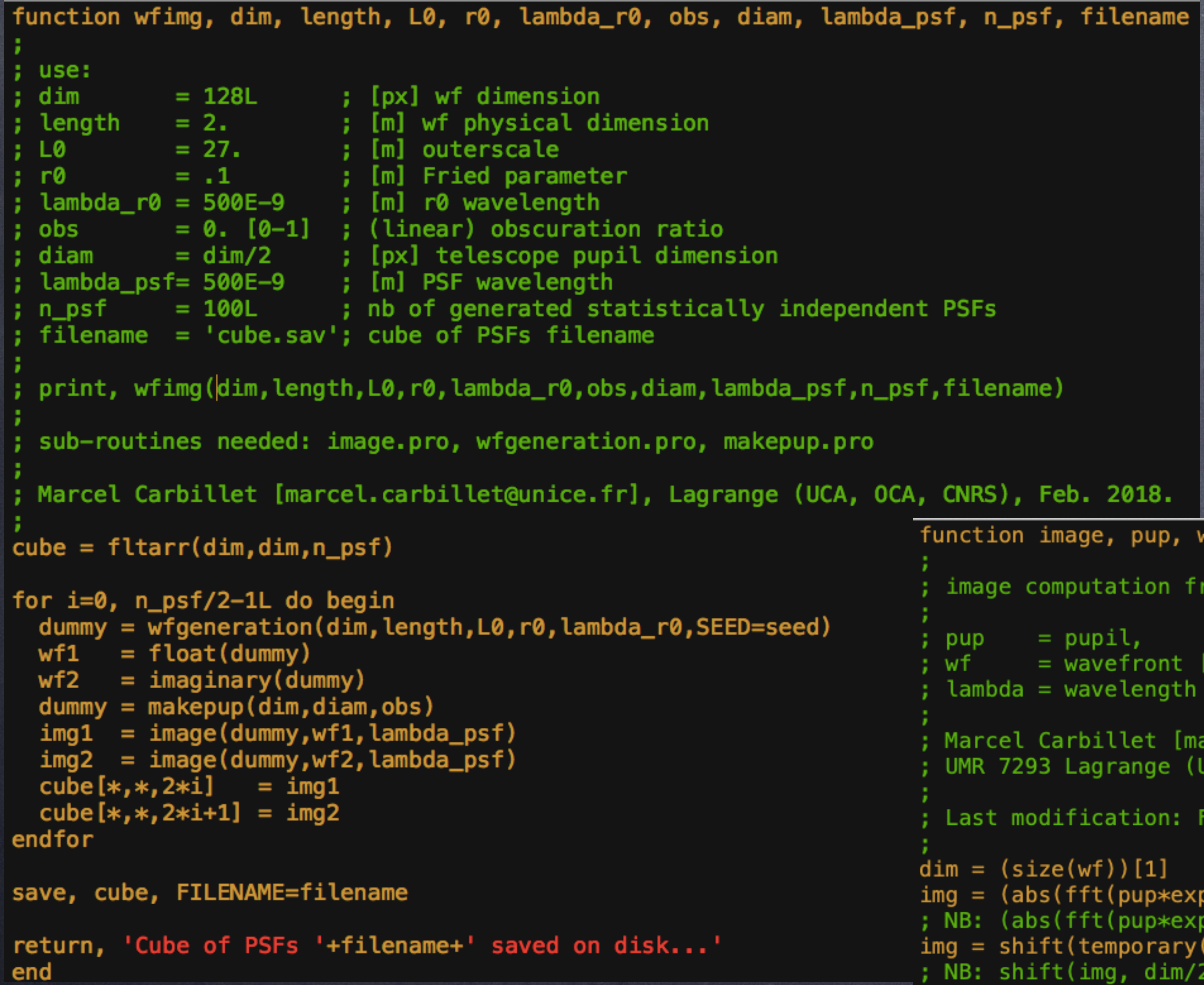
=> the image pixel λ/L must be at most half the resolution element (resel!) λ/D
(in other words : one must have AT LEAST 2 image pixels per λ/D)

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

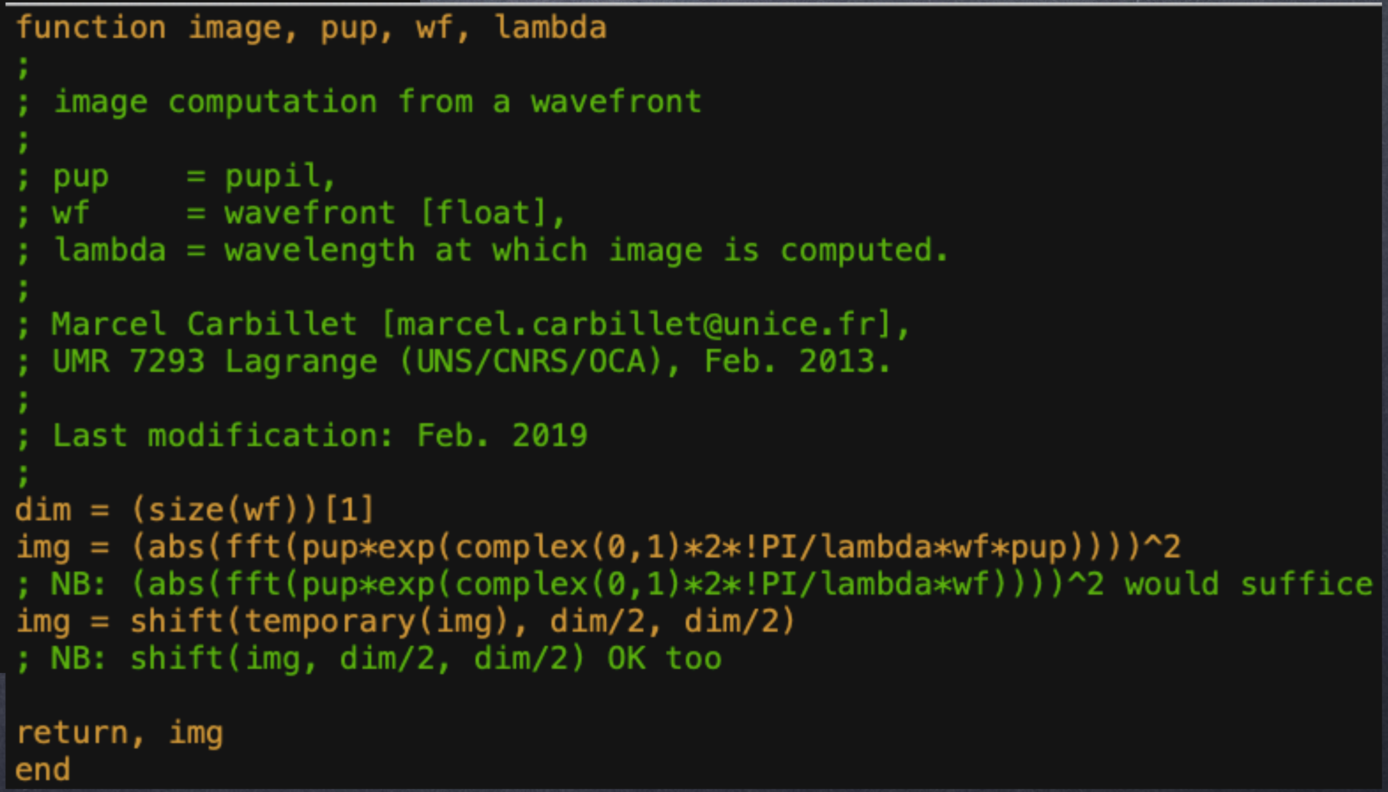
In addition

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

Images & turbulence - 21

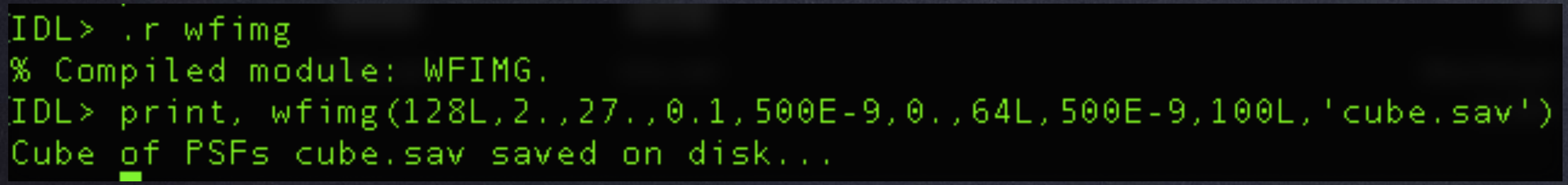


```
function wfimg, dim, length, L0, r0, lambda_r0, obs, diam, lambda_psf, n_psf, filename
;
; use:
; dim        = 128L      ; [px] wf dimension
; length     = 2.        ; [m] wf physical dimension
; L0         = 27.       ; [m] outerscale
; r0         = .1        ; [m] Fried parameter
; lambda_r0 = 500E-9     ; [m] r0 wavelength
; obs        = 0. [0-1]  ; (linear) obscuration ratio
; diam       = dim/2     ; [px] telescope pupil dimension
; lambda_psf= 500E-9     ; [m] PSF wavelength
; n_psf      = 100L      ; nb of generated statistically independent PSFs
; filename   = 'cube.sav'; cube of PSFs filename
;
; print, wfimg(dim,length,L0,r0,lambda_r0,obs,diam,lambda_psf,n_psf,filename)
;
; sub-routines needed: image.pro, wfgeneration.pro, makepup.pro
;
; Marcel Carbillet [marcel.carbillet@unice.fr], Lagrange (UCA, OCA, CNRS), Feb. 2018.
;
cube = fltarr(dim,dim,n_psf)

for i=0, n_psf/2-1L do begin
  dummy = wfgeneration(dim,length,L0,r0,lambda_r0,SEED=seed)
  wf1   = float(dummy)
  wf2   = imaginary(dummy)
  dummy = makepup(dim,diam,obs)
  img1  = image(dummy,wf1,lambda_psf)
  img2  = image(dummy,wf2,lambda_psf)
  cube[*,*,2*i]   = img1
  cube[*,*,2*i+1] = img2
endfor

save, cube, FILENAME=filename

return, 'Cube of PSFs '+filename+' saved on disk...'
end
```



```
function image, pup, wf, lambda
;
; image computation from a wavefront
;
; pup    = pupil,
; wf     = wavefront [float],
; lambda = wavelength at which image is computed.
;
; Marcel Carbillet [marcel.carbillet@unice.fr],
; UMR 7293 Lagrange (UNS/CNRS/OCA), Feb. 2013.
;
; Last modification: Feb. 2019
;
dim = (size(wf))[1]
img = (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
img = shift(temporary(img), dim/2, dim/2)
; NB: shift(img, dim/2, dim/2) OK too

return, img
end
```

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



```
IDL> .r wfimg
% Compiled module: WFIMG.
IDL> print, wfimg(128L,2.,27.,0.1,500E-9,0.,64L,500E-9,100L,'cube.sav')
Cube of PSFs cube.sav saved on disk...
```