Help on a few PASSATA features

Guido Agapito
guido.agapito@inaf.it
INAF Osservatorio Astrofisico di Arcetri, Largo E. Fermi 5, Firenze, Italy

CONTENTS

1 INTRODUCTION

Here we report the help to set up and use a few PASSATA (Agapito et al. 2016) features. In general, this help is written with the more advanced function/classe in mind, like `factory` class and function/classes contained in the `cloop` directory, but it can be extended in the case that `init` and `properties` of the processing and data objects are directly used paying attention to the variable names .

2 ZENITH ANGLE

`params.main.zentihAngleInDeg` is used in `factory` class to scale altitudes of atmospheric layers and sources (also sodium layer altitude when kernels of elongated SH spots are computed).

3 ATMOSPHERE

3.1 Maximum simulation time

Atmosphere evolution time is in general limited by the size of the phase screen (it can be set with `params.atmo.pixel_phasescreens`, and its default value is 8192L) and the wind speed, but `params.atmo.cycle_screens` can be set (=1B) to start from the beginning of the screen when the end is reached. This gives a discontinuity unless phase screens are cyclic (`params.atmo.make_cycle`).

3.2 Delay given by light propagation through the layers

`params.atmo.extra_delta_time` can be used to set up the `extra_delta_time` property in `atmo_evolution` class. For example, it can be set as:

```
params.atmo.extra_delta_time = params.atmo.heights / 299792458d
```

to consider the different time in which light from a star passes through the atmospheric layers. Instead if an upward propagation is considered it can be set as:

```
params.atmo.extra_delta_time = (max(params.atmo.heights) -
params.atmo.heights) / 299792458d
```

3.3 Cubes of phase

Use of cubes of phase. The atmo dictionary must be changed like this:

```
{atmo,
    filename:          'path/file.fits',
```

```
    wavelengthInNm:   2200.,
    rad:              1B
}
```

Fits file must be an array of `[n_pixel,n_pixel,n_iterations]`, where `n_pixel = params.main.pixel_pupil` and `n_iterations = params.main.total_time / params.main.time_step + 1` and if rad is 1B then the array values will be considered as radians of phase at wavelength equal to wavelengthInNm (otherwise it considers them to be in nm).
Moreover:

- `wind_speed` and `wind_direction` dictionaries are not used when a cube is used.
- set `params.control.int_gain` to a vector of 0 to measure the cubes without correction.

4 DEFORMABLE MIRRORS

4.1 DM set up

A set of functions can be used to set up the DM influence functions. They are:

- in PASSATA, `lib/compute_kl_ifunc.pro`, `compute_mixed_ifunc.pro` and `compute_zern_ifunc.pro`. These can be also used to set up DM influence functions at the beginning of the simulations without saving them on disk for example using as DM parameters:
```
{DM,
    type      : 'mixed'     ; 'kl' or 'zernike' or 'mixed'
    nmodes    : 1500,       ; number of modes
    nzern     : 3,          ; number zernike of modes
    npixels   : 160,        ; must be the same of
main.pixel_pupil
    obsratio  : 0.16,       ; central obscuration relative size
    diaratio  : 1.00,       ; diameter relative ratio w.r.t.
npixels
    height    : 0           ; DM height [m]
}
```
- In IdlTools `oaa_lib_ao_lib/make_modal_base_from_ifs.pro`, `make_modal_base_from_ifs_fft.pro` used together with a set of zonal influence functions and `ifunc` class.

4.2 DM misalignments

Several misalignment can be add to DMs:

- X/Y shifts: `params.dm.shiftXYinPixel`, a vector of 2 elements
- rotations: `params.dm.rotInDeg`
- magnifications: `params.dm.magnification`, 1 means no magnification

5 WAVEFRONT SENSORS

5.1 WFS types

WFS types available are:

- Shack-Hartmann: `classes/sh__define.pro`, `sh_gpu__define.pro`, `sh_shift__define.pro` and `sh_tilt__define.pro`
- Pyramid: `classes/modulated_pyramid__define.pro`, `modulated_pyramid_gpu__define.pro` and `pyr_tilt__define.pro`
- Ideal (that means a sensor that measures first derivative of incoming phase): `classes/ideal_wfs__define.pro`
- Perfect (that means a sensor that measures modal decomposition of incoming phase): `classes/modalanalysis_wfs__define.pro`
- Linearized focal-plane technique (LIFT, see Meimon et al. 2010): `classes/lift__define.pro`
  This one is a slope/mode computation method for a whole pupil WFS. Note that specific help and additional functions may be required to make it work properly.

Note that each one of this WFS is coupled with a slope computer class (`sh_slopec__define.pro`, `pyr_slopec__define.pro`, `ideal_wfs_slopec__define.pro` and `modalanalysis_slopec__define.pro`).

5.1.1 Valid sub-apertures

Shack-Hartmann (also valid for Ideal): this info is managed in the `class/sh_slopec__define.pro` class by the object `subapdata` (`class/subapdata__define.pro`). This object has a set of properties that are used to define the valid sub-apertures, `idxs` and `map`. `idxs` is a $n_{subap} \times n_{pixels}$ array with the pixels of the detector for each valid sub-aperture (for n_{subap} valid sub-apertures). `map` is a n_{subap} vector with the indices of the valid sub-apertures. Typically these arrays are computed by the `detect_subaps` method of `classes/sh__define.pro` class and saved on disk in the `subaps` directory in the calibration directory tree.

Pyramid: this info is managed in the `class/pyr_slopec__define.pro` class by the object `pupdata` (`class/pupdata__define.pro`). This object has a property that is used to define the valid sub-apertures, `ind_pup`. `ind_pup` is a $4 \times n_{subap}$ array with the 4 pixels of the detector (we are considering 4 sub-pupils) for each valid sub-aperture (for n_{subap} valid sub-apertures). Typically this array is computed by the function `lib/pupil_acquire.pro` and saved on disk in the `pupils` directory in the calibration directory tree.

5.2 Extended sources

- **Shack-Hartmann** sensor:

- Sodium elongation: the following three dictionaries are used to introduce the sodium elongation in the LGS WFS. Note that `zlayer.zfocus` and `zlayer.theta` must be determined off-line to get a zero tip, tilt and focus signal on the LGS WFS.

```
{ zlayer,
    func_type    : 'SIN'
    constant_tag : 'na_profile_20080717-06_40_56_x'
; [m] layer heights, tag name of the fits file stored in
params.main.root_dir+'/data/'
    zfocus       : 92092.6
; [m] focus value used as reference
    theta        : [-0.00617147, -0.0228771]
; [arcsec] tip/tilt value used as reference
}

{ zprofile,
    func_type    : 'SIN'
    constant_tag : 'na_profile_20080717-06_40_56_y'
; [m] layer intensity, tag name of the fits file stored in
params.main.root_dir+'/data/'
}

{ launcher,
; LGS launcher position
    position  : [1.25, 5.35, 96.0]
; position [x,y,z] meters (x eq y position) original position:
x=1250mm, y=5350mm, z=96000mm (x/y angle 13.15degree)
    sizeArcsec: 0.826
; this launcher dimension gives a total spot size of 1.2"
; square summed with the seeing of 0.87"
(sqrt(1.2^2.-0.87^2.)=0.826499 )
}
```

- Random jitter of the laser spot: the following three dictionaries are used to introduce a random tip/tilt residual (after removing the atmospheric tip/tilt residual) on the LGS WFSs with a standard deviation that is lgsttres.amp [nm]

```
{tt_modalanalysis,        ; modal analysis to remove
atmospheric Tip/Tilt from LGS SHS
 type      : 'zernike'
 nmodes    : 2
 npixels   : 160
 obsratio  : 0.16
 diaratio  : 1            ; DM height [m]
}

{tt_DM,                   ; zernike DM to remove atmospheric
Tip/Tilt from LGS SHS
```

```
    type       : 'zernike'
    nmodes     : 2
    npixels    : 160
    obsratio   : 0.16
    diaratio   : 1
    height     : 0
  ; DM height [m]
  }

  {lgsttres,                 ; disturbance to add random Tip/Tilt
  on LGS SHS
      func_type: 'RANDOM',
      amp: [737.,737.],      ; total 106 mas RMS on sky -->
  sqrt(106.^2./2.)*4.848e-9*8.118/4.*1e9
      nmodes: 2,
      height: 0,
      dm_type: 'zernike',
      dm_npixels: 160,
      dm_obsratio: 0.16,
      PRECISION: 0B,
      seed: 1
  }
```

- **Pyramid**:
  2D and 3D extended objects for the PWFS can be used setting up an extended_source object (`PASSATA/classes/extended_source__define.pro`) with get_extended_source of factory class and set it in the PWFS object with the set_extended_source method. Note that PWFS modulation is disabled with extended sources. Reference SPIE paper Esposito et al. 2016.

  ```
  extsource =
  factory.get_extended_source(params.extended_object)
  wfs.set_extended_source, extsource
  ```

  Below an example of the parameter set:

  ```
  {extended_object,
      polar_coordinate: [0.0, 0]
      height: 90000                ; source altitude in m for
  atmosphere propagation (!VALUES.F_INFINITY for cylindrical
  propagation, 90000 for sodium layer cone effect)
      magnitude : 9
      wavelengthInNm: 589
      type : 'GAUSS',              ; see obj_type in compute2d method
  of extended_source object
      sampling_type : 'CARTESIAN',  ; see sampling_type in compute2d
  method of extended_source object
      size_obj : 0.56              ; 2D size, for "TOPHAT" type it is
  the diameter
  ```

```
    multiples_fwhm : 1.0          ; extended object 2D sampling in
multiple of lambda/D (DL FWHM)
    show_source: 0B               ; if set display the extended
source points
    layerHeight: [0]              ; sodium layers altitude in m with
respect to focusHeight, a single elements means a 2D source
    intensityProfile: [1]         ; vector of sodium layers intensity
(total = 1), a single elements means a 2D source
    focusHeight: 90000            ; sodium layer focus altitude in m
}
```

Note that the first 4 keys must be the same as `params.wfs_source`.

5.3 Chromaticity

- **Shack-Hartmann** (note this a relatively old feature and now PASSATA could be updated considering the features illustrated in WFS misalignments section): `params.sh.xytilt` vector can be used to introduce a chromatic aberration on the focal plane while `params.sh.xyshift` vector can be used to introduce a chromatic aberration on the pupil plane. They cannot be set together. `tiltWavelengthInNm` (vector of waveleghts in nm, for each one a different WFS object will be built), `qe_factor_tilt` (vector of relative intensities, sum equal to 1) vectors are also required. Moreover these parameters:

```
     dm_npixels = 176     ; DM parameters to introduce tip/tilt in
focal plane, same number as in params.main.pixel_pupil
     dm_obsratio = 0.16d  ; DM parameters to introduce tip/tilt in
focal plane, same obscuration ratio of the pupil
     dm_type = 'zernike'  ; DM parameters to introduce tip/tilt in
focal plane, always 'zernike'
     nmodes = 2           ; DM parameters to introduce tip/tilt in
focal plane, always 2
     height = 0           ; DM parameters to introduce tip/tilt in
focal plane, always 0
```

  are required in case of `xytilt, and resize_fact` in case of `xyshift` (if a resize after the shift is required, it can be used to get shifts of less than 1 pixel).
- **Pyramid**:
  Below we report the pyramid dictionary for a GPI case with 60 sub-apertures on the diameter, chromaticity on 2 wavelengths (`tiltWavelengthInNm`) and a chromatic aberration on focal plane (`xyTilt`) and on pupil plane (`pup_shifts`):

```
{pyramid,
     pup_diam: 60.                    ; Pupil diameter in
subapertures
     pup_dist: 72.                    ; Requested separatoin between
pupil centers, in subapertures
     fov     : 2.1                    ; Requested field-of-view
[arcsec]
     fov_errinf : 0.1                 ; Maximum error in reducing fov
     fov_errsup : 3.                  ; Maximum error in reducing fov
```

```
     mod_amp = 3.0                    ; Modulation radius (in
lambda/D units)
     output_resolution: 140           ; Output sampling [usually
corresponding to CCD pixels]
     fft_res = 3.0                    ; pyramid focal-plane PSF
sampling in lambda/D units
     wavelengthInNm: 750              ; [nm] Pyramid wavelength (in
this case is not used,
                                      ; instead tiltWavelengthInNm
vector is used)
     tiltWavelengthInNm = [600.,900]  ; vector of waveleghts in nm,
for each one a different PWFS object will be build
     xyTilt = [[-200,0.],[200,0.]]    ; focal plane tip,tilt in nm
RSM, a couple for each PWFS
     qe_factor_tilt = [0.5,0.5]       ; vector of throughput, one
forr each PWFS, total must be 1
     pup_shifts = [[-1.0,0.0],[1.0,0.0]] ; pupil plane shift in
pixels, a couple for each PWFS
     func_type = 'SIN'                ; do not change this
     dm_npixels = 176                 ; DM parameters to introduce
tip/tilt in focal plane
     dm_obsratio = 0.16d              ;       ''          ''
''
     dm_type = 'zernike'              ;       ''          ''
''
     nmodes = 2                       ;       ''          ''
''
     height = 0                       ;       ''          ''
''
     }
```

5.4 WFS misalignments

- **Shack-Hartmann** sensor:
  - X/Y shifts: `params.sh.xShiftPhInPixel` and `yShiftPhInPixel` (`axShiftPhInPixel` and `ayShiftPhInPixel` that are used to avoid the automatic procedures to consider them for restoring a calibration that consider them)
  - rotations: `params.sh.rotAnglePhInDeg` (arotAnglePhInDeg that is used to avoid the automatic procedures to consider it for restoring a calibration that consider it)
- **Pyramid**:
  - X/Y shifts: `params.pyramidpup_shifts`, a vector of 2 elements
  - different X/Y shifts for each sub-pupil: `params.pyramid.pyr_tlt_coeff`, a matrix of 4 rows and 2 columns. Nominal value for this parameter is: `params.pyramid.pyr_tlt_coeff = [[1,1],[-1,1],[-1,-1],[1,-1]]`. To get a positive X shift of one pixel the nominal value of the first sub-pupil must be summed to

0.5/param.pyramid.pup_dist as: params.pyramid.pyr_tlt_coeff = [[1+0.5/param.pyramid.pup_dist,1],[-1,1],[-1,-1],[1,-1]]. Note that in this case the selection of valid sub-apertures can be refined considering a few options (see lib/pupil_acquire.pro).

5.5 WFS defects

Pyramid edges and tip defect (0 phase) in lambda/D unit can be introduced with params.pyramid.pyrEdgeDefLd and params.pyramid.pyrTipDefLd keys.

6 DETECTOR

- automatic update of detector parameters: auto_params_management method of the ccd class can be used to checks detector size, update a few detector parameters in function of params.detector.name (using IdlTools/oaa_lib/ao_lib/calc_detector_noise.pro function and, for background level, params.detector.sky_bg_norm key)
- Charge diffusion: params.detector.charge_diffusion and charge_diffusion_fwhm (this value in pixel FWHM) are used to add a gaussian charge diffusion in the ccd class.
- Pixel gains: params.detector.pixelGains_tag = 'tag_name' can be used to set a map of pixel gains (stored in params.main.root_dir+'/data/'). Instead, a vector of 4 elements, params.detector.quadrantsGains, can be used to set up pixel gains different per quadrant.
- Clocked Induced Charge (CIC) noise: can be set with params.detector.cic_noise and params.detector.cic_level.
- Charge Transmission Efficiency (CTE) noise: can be set with params.detector.cte_mat
- output of the detector can be equal to the input intensity if params.detector.doNotChangeI is set to 1b.

7 SLOPE COMPUTATION

Several options are available for a Shack Hartmann sensor:

- The default slope computation is Center of Gravity (CoG).
- a quad cell mode can be enabled, in case of a SCAO NGS mode → params.slopec.quadcell_mode = 1B.
- a Weighted Center of Gravity, in case of a SCAO NGS mode → params.slopec.weightedPixRad = 1, half width half maximum in pixel of the gaussian weight.

- a Windowing Center of Gravity, in case of a SCAO NGS mode → `params.slopec.weightedPixRad = 1` and `params.slopec.windowing = 1B`.
- correlation: `params.slopec.correlation`, `params.slopec.corrThr` and `params.slopec.corrWindowSidePix`. It requires a correlation template `corr_template` property of `sh_slopec` class.

8 CONTROL

Two control types are available, integrator (`params.control.type = 'INT'`) and infinite impulse response filter (`params.control.type = 'IIR'`).

- **INTEGRATOR**: in case of integrator a integrator gain vector is required, `params.control.int_gain`, and, optionally, a forgetting factor vector, `params.control.ff`, of the same size of the gain vector can be defined to get leaky integrators (a paper about these integrators is Agapito et al. 2019) instead of pure integrator.
    - OMG: `params.control.opt_dt` this value in seconds is used to get a recurring optimized modal gain vactor (similar but not exactly the same as Agapito et al. 2021). In this case `intcontrol_opt` class is selected instead of `intcontrol`
- **IIR filter**: in case of infinite impulse response filter a file saved using the `iirfilter` class (`save` method) is required, `params.control.iir_tag = 'tag_name'`.
  This class can store an arbitrary number of filters (it has been coded to have one filter for each mode) equal to the property `nfilter`. Methods like `set_num`, `set_den`, `set_zeros`, `set_poles` and `set_gain` can be used to set the filter parameters (filter coefficients of filter roots and gain). The case with a single pole and a gain per filter is equivalent to the integrator described in the previous point.
  Note that this kind of filter is typically used to get additional degrees of freedom in the temporal control that can be useful to reject structure vibrations or deal with particular features of the input disturbances (a paper about these filters is Agapito et al. 2012).

Then there is an additional control developed to reject vibrations:

- Adaptive vibration cancellation algorithm (Muradore et al. SPIE 2012), example for a SCAO system and a single vibration:

```
{avc,
    freq: freq,                         ; vibration frequency
    sinusSingleFreq: sinusSingleFreq, ; set it to 1B if vibration
spectrum is narrow, 0B if it is broad
    modes: modes,                       ; index of the mode affected by
the vibration
    estTFparams: 0B                     ; set it to 1B to estimate
closed loop transfer function parameters, to 0B to rely on the
theoretic value based on control parameters
}
```

Pseudo open loop control for MCAO system is code in classes/maory_polcrev_rtc__define.pro, maory_rtc__define.pro and maory_rtc_2step__define.pro.

9 VIBRATION AND ABERRATION

params.vibrations is used to add a static or dynamic aberration. Examples of this for a SCAO simulation (for MCAO multiple vibrations/aberrations can be set up with params.disturbance1, params.disturbance2, … and for each one of the LGS WFS paths, params.lgs_disturbance1, params.lgs_disturbance2, ...) are:

- tip/tilt vibrations (but it can be of any mode/modes):

```
{vibrations,
    func_type: 'VIB_PSD',                   ; type of function, see
classes/func_generator__define.pro
    vib_data: 'MacaoSinfoniVibrations',    ; vibration PSD tag name of
the fits file stored in params.main.root_dir+'/vibrations/'
    continuous_psd: 1B,                     ; PSD can be continuous as
in this case of made of a combination of sinusoidal signals
    nmodes: 2,                              ; number of modes starting
from the first one of the selected modal base
    height: 0,                              ; conjugation altitude of
the aberration
    influence_function:'VLT_ifunc_160p',   ; modal base
    PRECISION: 0B,
    seed: 1
}
```

- static aberration:

```
{vibrations,
    func_type          :'SIN',              ; type of function, see
classes/func_generator__define.pro
    constant           : [0,0,0,250,0,0,0,0,0,0], ; aberration modal
vector in nm RMS
    height             : 0.,                ; conjugation altitude of
the aberration
    influence_function : 'VLT_ifunc_160p', ; modal base
    nmodes             : 10L,               ; number of modes starting
from the first one of the selected modal base
    verbose            : 0B
}
```

- static aberration restored from disk:

```
{vibrations,
    map_tag: 'tag_name', ; tag name of the fits file stored in
params.main.root_dir+'/data/'
    height: 0.          ; conjugation altitude of the aberration
}
```

10 AUTOMATIC TAG NAME SELECTION

Parameters dictionary contains several tag names, typically of valid sub-aperture vector, slope null vector, interaction matrix, reconstruction matrix, … A set of procedure can be used to automatically selects them:

- `lib/give_me_the_tags_lngs.pro`
- `lib/give_me_the_tags_mcao.pro`
- `lib/give_me_the_tags_scao.pro`

if `'auto'` string is used for the tag names.
Note that **this is optional** and users can always select this name by hand or by its own functions.

Examples of how to use this can be found in `updateParams` methods of `cloop/base_scao_loop__define.pro`, `cloop/base_2wfs_loop__define.pro` and `cloop/base_mcao_loop__define.pro`.
Note that in `cloop/base_mcao_loop__define.pro` the approach is different because reconstruction and projection matrices (here we are using a pseudo-open loop control) are saved in directories named as timestamp with a `recmat.fits` file (in case of reconstruction matrices) and with a parameters dictionary file inside. This dictionary is the one used during the calibration and it is generally used by the `lib/search_mcao_mat.pro` function to find the desired matrix.

11 DATASTORE CLASS

This class is used to collect and save on disk a simulation, but it can also be used to restore the data and the parameters dictionary with `restore` (function) method (that uses `restore` and `restore_tracknum` procedure methods) and can be used to list keys, `keys` and `HasKey` methods, return values and times, `values` and `times` methods, compute average values, `mean` method, make some plots, `plot` method, ...

12 DICTIONARIES MANAGEMENT

A few functions are available to manage parameters and/or dictionaries, main ones are:

- `lib/combine_params.pro`
- `lib/compare_dict.pro`
- `lib/duplicate_params.pro`
- `lib/make_params_permutations.pro`
- `lib/read_params_file.pro`
- `lib/search_keys_dict.pro`

13 HOW TO RUN A SET OF SIMULATIONS

Sometimes it is useful to run a set of simulations with a single main file where only a few parameters change: for example to estimate performance with different seeing values.

- The first option is to set up a for cycle in the main file and update the desired parameters at each step of the simulations.
- The second option is to build a list of parameters dictionary to be used in a for cycle. The list can be built using the function `lib/make_params_permutations.pro`. Example:

  ```
  ; parameters to be explored
  params_to_explore = list()
  params_to_explore.add,
  list('wfs_source','magnitude',[12,15,16,17,18,19,20.])
  params_to_explore.add,
  list('detector','dt',[2d-3,2d-3,2d-3,2d-3,2d-3,2d-3,3d-3])
  params_list = make_params_permutations(params_to_explore,
  permutations_matrix=permutations_matrix, /no_permutations)
  ```

  Here using `permutations_matrix` array and `no_permutations` keyword several options can be set up. For example in the case above star magnitude and detector integration time are combined one-to-one and a `params_list` with all the combinations can be produced removing the `no_permutations` keyword. This function is also used in `lib/iterate_dictionary.pro`. In this case `combine_params` function must be used to get the i-th dictionary combining the original parameters dictionary and the i-th element of `params_list`.
- The third option requires the use of the `expand` keyword of `lib/read_params_file.pro` function in the parameters file. In this case for each parameter key to be explored with different values the syntax must be:

  ```
  time_step: iterate( [0.001,0.002,0.005d], 1)
  ```

  where ther vector collects the values to be explored and the scalar value the “group”. This “group” can be used to get a one-to-one combination (so no permutations) with other parameters (for example `params.main.time_step` and `params.detector.dt`). In this case the output of `read_params_file` is a list of dictionaries.

14 MODAL ANALYSIS

When the dictionary `params.modalanalysis` is set in a SCAO simulation (NGS or LGS+NGS), residual turbulence is decomposed on the selected modal base at each step and a `resMod` variable is added to the datastore (this is done by the `addModalAnalysis`

method of `base_loop` class).
Procedure `lib/do_modal_plot.pro` can then be used to plot a figure with curves of modal turbulence and residual standard deviation (or RMS) starting from the data stored in the `datastore` object (`store` in the procedure call). This figure is known colloquially as "modal plot".

A couple of examples of `params.modalanalysis:`

```
{modalanalysis,
   phase2modes_tag: 'VLT_ifunc_160p_inv' ; tag name of the inverse
of the VLT_ifunc_160p modal base
}

{modalanalysis,
    type       : 'kl'        ; KL modal base
    nmodes     : 1000        ; with 1000 modes
    npixels    : 160         ; 160x160 pixels
    obsratio   : 0.16        ; 16% of central obscuration
    diaratio   : 1           ; diameter is 100% of npixels
}
```

Note that for a MCAO simulation the same kind of analysis can be done offline, see sec. 16.

15 PSF VISUALIZATION

An example of how a PSF can be visualized is reported below. It uses functions and procedures from IdlTools library. A few notes:

- `psf` is the PSF array, it can also be a cube.
- `psf_resolution` is the padding coefficient used in the FFT to compute the PSF.
- `scale` is an output and it is the pixel scale of the PSF.
- `profile` is an output and it is a list of structures (because psf can be a cube of PSFs) with `off_axis_angle`, `prof_res` and `ee`.
- `FWHM` is the FWHM computed from the profile.

```
maxval_psf = max(psf)
minval_psf = maxval_psf*1e-4 ; this is used to set up the dynamic range in
the figure
range_psf = 1 ; this is used to set up the portion of FoV shown in the figure

loadct, 3 ; red color scale
nwin = 1 ; window number
xsize = 800 ; window size
ysize = 600

psf_show, psf, wavelengthInM, diameterInM, $
          /noshift, /log, /as, /sh, /inv, /noproplot, $
          psf_resolution=psf_resolution, range_psf=range_psf, $
```

```
          minval_psf=minval_psf, maxval_psf=maxval_psf, $
          nwin=nwin, xsize=xsize, ysize=ysize, $
          profile=profile, scale=scale

maxProfPsf = max(profile[0].prof_res)

window, nwin+1, xsize=xsize, ysize=ysize
plot_io, profile[0].off_axis_angle, profile[0].prof_res/maxProfPsf,
xra=[0,range_psf], yra=[1e-6,1], $
         xtit='!17Off-axis angle !4[!17arcsec!4]!17', ytit='!17Normalized
profile', ytickformat='exponent', $
         title='!17wavelength: '+strtrim(wavelengthInM*1e9,2)+'nm', thick=2

window, nwin+2, xsize=xsize, ysize=ysize
plot, profile[0].off_axis_angle, profile[0].ee, xst=17, xra=[0,range_psf], $
      xtit='!17Off-axis angle !4[!17arcsec!4]!17', ytit='!17Encircled
Energy', $
      title='!17wavelength: '+strtrim(wavelengthInM*1e9,2)+'nm', thick=2

FWHM = calc_fwhm_from_prof(profile[0].prof_res, profile[0].off_axis_angle)
```

16 SIMULATION REPRODUCIBILITY

PASSATA simulation can be reproduced to make off-line PSF computation, modal decomposition of residual phase and residual phase cube saves. There are a set of functions/procedures/classes to do so. They are:

- `lib/scao_data_analysis.pro` (and `analyse_scao_saved_data.pro` and `analyse_lgs_saved_data.pro`) that can be used to compute PSF on the line of sight of the NGS starting from a saved simulation or a list of saved simulations.
- `lib/collect_offaxis_data.pro` (and `compute_off_axis_cube.pro`, `compute_off_axis_init.pro`, `compute_off_axis_modal_analysis.pro`, `compute_off_axis_psf.pro` and `compute_off_axis_resphase.pro`) that can be used to compute PSF, residual cubes or modal decomposition of residual phase in any direction of the FoV (defined by `params.atmo.mcao_fov`).

So, in principle, performance of the simulation can be estimated off-line. This is useful to speed up the simulation, reduce its memory requirements and change wavelengths, coordinates, … at a later time.