
COMPASS (SHESHA) Documentation

Release r791

COMPASS team

May 30, 2016

CONTENTS

1	Contents:	3
1.1	shesha	3
1.2	shesha_atmos	4
1.3	shesha_dms	5
1.4	shesha_param	9
1.5	shesha_rtc	25
1.6	shesha_sensors	34
1.7	shesha_target	40
2	Indices and tables	43
	Index	45

Version PDF

CONTENTS:

1.1 shesha

1.1.1 bin2d()

`shesha.bin2d()`

Returns the input 2D array “array”, binned with the binning factor “binfact”. The input array X and/or Y dimensions needs not to be a multiple of “binfact”; The final/edge pixels are in effect replicated if needed. This routine prepares the parameters and calls the C routine `_bin2d`. The input array can be of type long, float or double. Last modified: Dec 15, 2003. Author: F.Rigaut SEE ALSO: `_bin2d`

Parameters `data_in`: (np.ndarray) : data to binned

`binfact`: (int) : binning factor

1.1.2 indices()

`shesha.indices (int dim1, int dim2=-1)`

Return a `dimxdimx2` array. First plane is the X indices of the pixels in the `dimxdim` array. Second plane contains the Y indices.

Inspired by the Python `scipy` routine of the same name.

New (June 12 2002): `dim` can either be :

- a single number `N` (e.g. 128) in which case the returned array are square (`NxN`)
- a Yorick array size, e.g. `[#dimension,N1,N2]`, in which case the returned array are `N1xN2`
- a vector `[N1,N2]`, same result as previous case

F.Rigaut 2002/04/03 SEE ALSO: `span`

Parameters

- **dim1** – (int) : first dimension
- **dim2** – (int) : (optional) second dimension

1.1.3 makegaussian()

`shesha.makegaussian (size, fwhm, xc, yc)`

Returns a centered gaussian of specified size and fwhm. norm returns normalized 2d gaussian

Parameters `size`: (int) :

`fwhm`: (float) :

`xc`: (int) : (optional) center position on x axis

`yc`: (int) : (optional) center position on y axis

norm: (int) : (optional) normalization

1.2 shesha_atmos

1.2.1 Atmos

`class shesha_atmos.Atmos`

add_screen()

Add a screen to the atmos object.

Parameters size: (float) : dimension of the screen (size x size)

amplitude: (float) : frac

altitude: (float) : altitude of the screen in meters

windspeed: (float) : windspeed of the screen [m/s]

winddir: (float) : wind direction (deg)

deltax: (float) : extrude deltax pixels in the x-direction at each iteration

deltay: (float) : extrude deltax pixels in the y-direction at each iteration

device: (int) : device number

del_screen()

Delete a screen from the atmos object

Parameters alt – (float) : altitude of the screen to delete

disp()

Display the screen phase at a given altitude

Parameters alt – (float) : altitude of the screen to display

get_screen()

Return a numpy array containing the turbulence at a given altitude

Parameters alt – (float) :altitude of the screen to get

list_alt()

Display the list of the screens altitude

move_atmos()

Move the turbulence in the atmos screen following previous loaded paramters such as windspeed and wind direction

1.2.2 atmos_init()

`shesha_atmos.atmos_init()`

atmos_init(naga_context c, Param_atmos atm, Param_tel tel, Param_geom geom, Param_loop loop, wfss=None, Param_target target=None, int rank=0, int clean=1, dict load={})

Create and initialise an atmos object

Parameters c: (naga_context) : context

tel: (Param_tel) : telescope settings

geom: (Param_geom) : geometry settings

loop: (Param_loop) : loop settings

wfss: (list of Param_wfs) : (optional) wfs settings
sensors: (Sensors) : (optional) Sensors object on GPU
target: (Param_target) : (optional) target_settings
overwrite: (int) : (optional) overwrite data files if overwrite=1 (default 1)
rank: (int) : (optional) rank of the process (default=0)
clean: (clean) : (optional) ? (default=1)
load: (dict) : (optional) ? (default={})

1.3 shesha_dms

1.3.1 Dms

class shesha_dms.Dms

add_dm()

Add a dm into a Dms object

Parameters type_dm: (str) : dm type to remove,

alt: (float) : dm conjugaison altitude to remove,

ninflu: (long) : ,

influsize: (long) : ,

ninflupos: (long) : ,

npts: (long) : ,

push4imat: (float) : ,

device: (int) : device where the DM will be create (default=-1):

comp_oneactu()

Compute the shape of the dm when pushing the nactu actuator

Parameters type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

nactu: (int) : actuator number pushed

ampli: (float) : amplitude

computeKLbasis()

Compute a Karhunen-Loeve basis for the dm:

- compute the phase covariance matrix on the actuators using Kolmogorov
- compute the geometric covariance matrix
- double diagonalisation to obtain KL basis

Parameters type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

xpos: (np.ndarray[ndim=1,dtype=np.float32_t]) : x-position of actuators

ypos: (np.ndarray[ndim=1,dtype=np.float32_t]) : y-position of actuators

indx_pup: (np.ndarray[ndim=1,dtype=np.int32_t]) : indices of where(pup)

dim: (long) : number of where(pup)

norm: (float) : normalization factor

ampli: (float) : amplitude

getComm()

Return the voltage command of the sutra_dm

Parameters type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

Returns data : (np.ndarray(dims=1,dtype=np.float32)) : voltage vector

getInflu()

Return the influence functions of the DM

Parameters type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

Returns data : (np.ndarray(dims=3,dtype=np.float32)) : influence functions

get_KLbasis()

Return the klbasis computed by computeKLbasis

Parameters type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

Returns KLbasis : (np.ndarray(dims=2,dtype=np.float32)) : the KL basis

get_dm()

Return the shape of the dm

Parameters type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

Returns data : (np.ndarray(dims=2,dtype=np.float32)) : DM shape

load_kl()

Load all the arrays computed during the initialization for a kl DM in a sutra_dms object

Parameters alt: (float) : dm conjugaison altitude

rabas: (np.ndarray[ndim=1,dtype=np.float32_t]) : TODO

azbas: (np.ndarray[ndim=1,dtype=np.float32_t]) :

ord: (np.ndarray[ndim=1,dtype=np.int32_t]) :

cr: (np.ndarray[ndim=1,dtype=np.float32_t]) :

cp: (np.ndarray[ndim=1,dtype=np.float32_t]) :

load_pzt()

Load all the arrays computed during the initialization for a pzt DM in a sutra_dms object

Parameters alt: (float) : dm conjugaison altitude

influ: (np.ndarray[ndim=3,dtype=np.float32_t]) : influence functions

influpos: (np.ndarray[ndim=1,dtype=np.int32_t]) : positions of the IF

npoints: (np.ndarray[ndim=1,dtype=np.int32_t]) [for each pixel on the DM screen,]
the number of IF which impact on this pixel

istart: (np.ndarray[ndim=1,dtype=np.int32_t]) :

xoff: (np.ndarray[ndim=1,dtype=np.int32_t]) : x-offset

yoff: (np.ndarray[ndim=1,dtype=np.int32_t]) : y-offset

kern: (np.ndarray[ndim=1,dtype=np.float32_t]) : convoltuon kernel

load_tt()

Load all the arrays computed during the initialization for a tt DM in a sutra_dms object

Parameters alt: (float) : dm conjugaison altitude

influ: (np.ndarray[ndim=3,dtype=np.float32_t]) : influence functions

oneactu()

Push on on the nactu actuator of the DM with ampli amplitude and compute the corresponding shape

Parameters type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

nactu: (int) : actuator number

ampli: (float): amplitude

remove_dm()

Remove a dm from a Dms object

Parameters type_dm: (str) : dm type to remove

alt: (float) : dm conjugaison altitude to remove

resetdm()

Reset the shape of the DM to 0

Parameters type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

set_comm()

Set the voltage command on a sutra_dm

type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

comm: (np.ndarray[ndim=1,dtype=np.float32_t]) : voltage vector

shape_dm: (bool) : perform the dm_shape after the load (default=False)

set_full_comm()

Set the voltage command

comm: (np.ndarray[ndim=1,dtype=np.float32_t]) : voltage vector

shape_dm: (bool) : perform the dm_shape after the load (default=True)

shape_dm()

Compute the shape of the DM in a sutra_dm object

type_dm: (str) : dm type

alt: (float) : dm conjugaison altitude

1.3.2 comp_dmgeom()

shesha_dms.comp_dmgeom()

Compute the geometry of a DM : positions of actuators and influence functions

Parameters dm: (Param_dm) : dm settings

geom: (Param_geom) : geom settings

1.3.3 `computeDMbasis()`

`shesha_dms.computeDMbasis()`

Compute a the DM basis []

- push on each actuator
- get the corresponding dm shape
- apply pupil mask and store in a column

Parameters `g_dm`: (Dms) : Dms object

`p_dm`: (Param_dm) : dm settings

`p_geom`: (Param_geom) : geom settings

Returns `IFbasis = (np.ndarray((indx_valid.size,Nactu),dtype=np.float32))` : DM IF basis

1.3.4 `compute_klbasis()`

`shesha_dms.compute_klbasis()`

Compute a Karhunen-Loeve basis for the dm:

- compute the phase covariance matrix on the actuators using Kolmogorov
- compute the geometric covariance matrix
- double diagonalisation to obtain KL basis

Parameters `g_dm`: (Dms) : Dms object

`p_dm`: (Param_dm) : dm settings

`p_geom`: (Param_geom) : geom settings

`p_atmos`: (Param_atmos) : atmos settings

`p_tel`: (Param_tel) : telescope settings

1.3.5 `dm_init()`

`shesha_dms.dm_init()`

Create and initialize a Dms object on the gpu

Parameters `p_dms`: (list of Param_dms) : dms settings

`p_wfs`: (Param_wfs) : wfs settings

`p_geom`: (Param_geom) : geom settings

`p_tel`: (Param_tel) : telescope settings

1.3.6 `make_pzt_dm()`

`shesha_dms.make_pzt_dm()`

Compute the actuators positions and the influence functions for a pzt DM

Parameters `p_dm`: (Param_dm) : dm settings

`geom`: (Param_geom) : geom settings

`p_tel`: (Param_tel) : tel settings

`irc`: factor for influence size

Returns influ: (np.ndarray(dims=3,dtype=np.float64)) : cube of the IF for each actuator

1.4 shesha_param

1.4.1 Param_loop

class shesha_param.Param_loop

ittime
iteration time (in sec)

niter
number of iterations

set_ittime()
Set iteration time

Parameters t: (float) :iteration time

set_niter()
Set the number of iteration

Parameters n: (long) : number of iteration

1.4.2 Param_tel

class shesha_param.Param_tel

cobs
central obstruction ratio.

diam
telescope diameter (in meters).

nbrmissing
number of missing segments for EELT pupil (max is 20).

pupangle
rotation angle of pupil.

referr
std of reflectivity errors for EELT segments (fraction).

set_cobs()
set the central obstruction ratio

Parameters c – (float) : central obstruction ratio

set_diam()
set the telescope diameter

Parameters d – (float) : telescope diameter (in meters)

set_nbrmissing()
set the number of missing segments for EELT pupil

Parameters nb – (long) : number of missing segments for EELT pupil (max is 20)

set_pupangle()
set the rotation angle of pupil

Parameters p – (float) : rotation angle of pupil

set_referr()
set the std of reflectivity errors for EELT segments
Parameters **ref** – (float) : std of reflectivity errors for EELT segments (fraction)

set_spiders_type()
set the secondary supports type
Parameters **spider** – (str) : secondary supports type

set_std_piston()
set the std of piston errors for EELT segments
Parameters **piston** – (float) : std of piston errors for EELT segments

set_std_tt()
set the std of tip-tilt errors for EELT segments
Parameters **tt** – (float) : std of tip-tilt errors for EELT segments

set_t_spiders()
set the secondary supports ratio
Parameters **spider** – (float) : secondary supports ratio

set_type_ap()
set the EELT aperture type
Parameters **t** – (str) : EELT aperture type

spiders_type
secondary supports type: “four” or “six”.

std_piston
std of piston errors for EELT segments

std_tt
std of tip-tilt errors for EELT segments

t_spiders
secondary supports ratio.

type_ap
EELT aperture type: “Nominal”, “BP1”, “BP3”, “BP5” (for back-up plan with 1, 3, or 5 missing annulus).

1.4.3 Param_geom

class shesha_param.Param_geom

cent
central point of the simulation.

geom_init()
Initialize the system geometry
Parameters **tel**: (Param_tel) : telescope settings
 pupdiam: (long) : linear size of total pupil
 apod: (int) : apodizer

get_ipupil()
return the full pupil support

get_mpupil()
return the padded pupil

get_n ()
Return the linear size of the medium pupil

get_n1 ()
Return the min(x,y) for valid points for the total pupil

get_n2 ()
Return the max(x,y) for valid points for the total pupil

get_p1 ()
Return the min(x,y) for valid points for the medium pupil

get_p2 ()
Return the max(x,y) for valid points for the medium pupil

get_spupil ()
return the small pupil

pupdiam
linear size of total pupil (in pixels).

set_cent ()
Set the central point of the simulation

Parameters c – (float) : central point of the simulation.

set_pupdiam ()
Set the linear size of total pupil

Parameters p – (long) : linear size of total pupil (in pixels).

set_ssize ()
Set linear size of full image

Parameters s – (long) : linear size of full image (in pixels).

set_zenithangle ()
Set observations zenith angle

Parameters z – (float) : observations zenith angle (in deg).

ssize
linear size of full image (in pixels).

zenithangle
observations zenith angle (in deg).

1.4.4 Param_wfs

class shesha_param.Param_wfs

Lambda
observation wavelength (in μm) for a subap.

atmos_seen
1 if the WFS sees the atmosphere layers

beamsize
laser beam fwhm on-sky (in arcsec).

dms_seen
index of dms seen by the WFS

error_budget
If True, enable error budget analysis for the simulation

fracsub
minimal illumination fraction for valid subaps.

fssize
size of field stop in arcsec.

fstop
Fields of the wfs diaphragm shape : “square” or “none”

gsalt
altitude of guide star (in m) 0 if ngs.

gsmag
magnitude of guide star.

laserpower
laser power in W.

lgsreturnperwatt
return per watt factor (high season : 10 ph/cm2/s/W).

lltx
x position (in meters) of llt.

llty
y position (in meters) of llt.

noise
desired noise : < 0 = no noise / 0 = photon only / > 0 photon + ron.

nphotons4imat
number of photons per subap used for doing imat

npix
number of pixels per subap.

nxsub
linear number of subaps.

openloop
1 if in “open-loop” mode (i.e. does not see dm).

optthroughput
wfs global throughput.

pixsize
pixel size (in arcsec) for a subap.

proftype
type of sodium profile “gauss”, “exp”, etc ...

pyr_ampl
pyramid wfs modulation amplitude radius [arcsec].

pyr_loc
Location of modulation, before/after the field stop. valid value are “before” or “after” (default “after”).

pyr_npts
total number of point along modulation circle [unitless].

pyrtype
Type of pyramid, either 0 for “Pyramid” or 1 for “RoofPrism”.

set_lambda()
Set the observation wavelength

Parameters **L** – (float) : observation wavelength (in μm) for a subap

set_altna()
Set the corresponding altitude

Parameters **a** – (np.ndarray[ndim=1,dtype=np.float32]) : corresponding altitude

set_atmos_seen()
Tells if the wfs sees the atmosphere layers

Parameters **i** – (int) : 1 if the WFS sees the atmosphere layers

set_beamsize()
Set the laser beam fwhm on-sky

Parameters **b** – (float) : laser beam fwhm on-sky (in arcsec)

set_dms_seen()
Set the index of dms seen by the WFS

Parameters **dms_seen** – (np.ndarray[ndim=1,dtype=np.int32_t]) : index of dms seen by the WFS

set_errorBudget()
Set the error budget flag : if True, enable error budget analysis for this simulation

Parameters **error_budget** – (bool) : error budget flag

set_fracsub()
Set the minimal illumination fraction for valid subaps

Parameters **f** – (float) : minimal illumination fraction for valid subaps

set_fssize()
Set the size of field stop

Parameters **f** – (float) : size of field stop in arcsec

set_fstop()
Set the size of field stop

Parameters **f** – (str) : size of field stop in arcsec

set_gsalt()
Set the altitude of guide star

Parameters **g** – (float) : altitude of guide star (in m) 0 if ngs

set_gsmag()
Set the magnitude of guide star

Parameters **g** – (float) : magnitude of guide star

set_kernel()
Set the attribute kernel

Parameters **k** – (float) :

set_laserpower()
Set the laser power

Parameters **l** – (float) : laser power in W

set_lgsreturnperwatt()
Set the return per watt factor

Parameters **lpw** – (float) : return per watt factor (high season : 10 ph/cm2/s/W)

set_lltx()
Set the x position of llt

Parameters **l** – (float) : x position (in meters) of llt

set_llty()
Set the y position of llt

Parameters **l** – (float) : y position (in meters) of llt

set_noise()
Set the desired noise

Parameters **n** – (float) : desired noise : < 0 = no noise / 0 = photon only / > 0 photon + ron

set_nphotons4imat()
Set the desired numner of photons used for doing imat

Parameters **nphot** – (float) : desired number of photons

set_npix()
Set the number of pixels per subap

Parameters **n** – (long) : number of pixels per subap

set_nxsub()
Set the linear number of subaps

Parameters **n** – (long) : linear number of subaps

set_openloop()
Set the loop state (open or closed)

Parameters **o** – (long) : 1 if in “open-loop” mode (i.e. does not see dm)

set_optthroughput()
Set the wfs global throughput

Parameters **o** – (float) : wfs global throughput

set_pixsize()
Set the pixel size

Parameters **p** – (float) : pixel size (in arcsec) for a subap

set_profna()
Set the sodium profile

Parameters **p** – (np.ndarray[ndim=1,dtype=np.float32]) : sodium profile

set_proftype()
Set the type of sodium profile

Parameters **p** – (str) : type of sodium profile “gauss”, “exp”, etc ...

set_pyr_ampl()
Set the pyramid wfs modulation amplitude radius

Parameters **p** – (float) : pyramid wfs modulation amplitude radius (in arsec)

set_pyr_loc()
Set the location of modulation

Parameters **p** – (str) : location of modulation, before/after the field stop. valid value are “before” or “after” (default “after”)

set_pyr_npts()
Set the total number of point along modulation circle

Parameters **p** – (long) : total number of point along modulation circle

set_pyrtype()
Set the type of pyramid,

Parameters **p** – (str) : type of pyramid, either 0 for “Pyramid” or 1 for “RoofPrism”

set_type()
Set the type of wfs

Parameters **t** – (str) : type of wfs (“sh” or “pyr”)

set_xpos()

Set the guide star x position on sky

Parameters **x** – (float) : guide star x position on sky (in arcsec)

set_ypos()

Set the guide star y position on sky

Parameters **y** – (float) : guide star y position on sky (in arcsec)

set_zerop()

Set the detector zero point

Parameters **z** – (float) : detector zero point

type_wfs

type of wfs : “sh” or “pyr”.

xpos

guide star x position on sky (in arcsec).

ypos

guide star x position on sky (in arcsec).

zerop

detector zero point.

1.4.5 Param_atmos

class shesha_param.Param_atmos

L0

L0 per layers in meters.

alt

altitudes of each layer.

deltax

x translation speed (in pix / iteration) for each layer.

deltay

y translation speed (in pix / iteration) for each layer.

dim_screens

linear size of phase screens.

frac

fraction of r0 for each layer.

nscreens

number of turbulent layers.

pupixsize

pupil pixel size (in meters).

r0

global r0.

set_L0()

Set the L0 per layers

Parameters **l** – (lit of float) : L0 for each layers

set_alt()

Set the altitudes of each layer

Parameters **l** – (lit of float) : altitudes

set_deltax()
Set the translation speed on axis x for each layer
Parameters **1** – (lit of float) : translation speed

set_deltay()
Set the translation speed on axis y for each layer
Parameters **1** – (lit of float) : translation speed

set_dim_screens()
Set the size of the phase screens
Parameters **1** – (lit of float) : phase screens sizes

set_frac()
Set the fraction of r0 for each layers
Parameters **1** – (lit of float) : fraction of r0

set_nscreens()
Set the number of turbulent layers
Parameters **n** – (long) number of screens.

set_pupixsize()
Set the pupil pixel size
Parameters **xsize** – (float) : pupil pixel size

set_r0()
Set the global r0
Parameters **r** – (float) : global r0

set_seeds()
Set the seed for each layer
Parameters **1** – (lit of int) : seed

set_winddir()
Set the wind direction for each layer
Parameters **1** – (lit of float) : wind directions

set_windspeed()
Set the the wind speed for each layer
Parameters **1** – (lit of float) : wind speeds

winddir
wind directions of each layer.

windspeed
wind speeds of each layer.

1.4.6 Param_dm

class shesha_param.**Param_dm**

alt
conjugaison altitude (im m)

center_name
filename for influ hdf5 file

coupling
actuators coupling (<0.3)

cube_name
name for influence cube in hdf5

file_influ_hdf5
filename for influ hdf5 file

hyst
actuators hysteresis (<1.)

margin
outside margin for actuator select

margin_in
inside margin for actuator select

margin_out
outside margin for actuator select

nact
number of actuators in the diameter

nk1
number of kl modes

pupoffset
2.

push4imat
nominal voltage for imat

set_alt()
set the conjugaison altitude

Parameters a – (float) : conjugaison altitude (im m)

set_center_name()
set the name of hdf5 influence file

Parameters filename – (str) : Hdf5 file influence name

set_coupling()
set the actuators coupling

Parameters c – (float) : actuators coupling (<0.3)

set_cube_name()
set the name of influence cube in hdf5

Parameters cubename – (str) : name of influence cube

set_file_influ_hdf5()
set the name of hdf5 influence file

Parameters filename – (str) : Hdf5 file influence name

set_i1()
TODO doc

Parameters i1 – (np.ndarray[ndim=1,dtype=np.int32_t]) :

set_influ()
Set the influence function

Parameters influ – (np.ndarray[ndim=3,dtype=np.float32_t]) : influence function

set_j1()
TODO doc

Parameters j1 – (np.ndarray[ndim=1,dtype=np.int32_t]) :

set_margin()

set the margin for outside actuator select

Parameters **n** – (float) : pupille diametre ratio for actuator select

set_margin_in()

set the margin for inside actuator select (central obstruction)

Parameters **n** – (float) : pupille diametre ratio for actuator select

set_margin_out()

set the margin for outside actuator select

Parameters **n** – (float) : pupille diametre ratio for actuator select

set_nact()

set the number of actuator

Parameters **n** – (long) : number of actuators in the diameter

set_ntotact()

set the total number of actuators

Parameters **n** – (long) : total number of actuators

set_pattern()

set the pattern type

Parameters **t** – (str) : type of pattern

set_push4imat()

set the nominal voltage for imat

Parameters **p** – (float) : nominal voltage for imat

set_thresh()

set the threshold on response for selection

Parameters **t** – (float) : threshold on response for selection (<1)

set_type()

set the dm type

Parameters **t** – (str) : type of dm

set_unitpervolt()

set the Influence function sensitivity

Parameters **u** – (float) : Influence function sensitivity in unit/volt

set_x_name()

set the name of x coord of influence fonction in file

Parameters **t** – (str) : name of x coord of influence

set_xpos()

Set the x positions of influ functions (lower left corner)

Parameters **xpos** – (np.ndarray[ndim=1,dtype=np.float32_t]) : x positions of influ functions

set_y_name()

set the name of y coord of influence fonction in file

Parameters **yname** – (str) : name of y coord of influence

set_ypos()

Set the y positions of influ functions (lower left corner)

Parameters **ypos** – (np.ndarray[ndim=1,dtype=np.float32_t]) : y positions of influ functions

thresh
threshold on response for selection (<1)

type_dm
type of dm

type_pattern
type of pattern

unitpervolt
Influence function sensitivity in unit/volt. Optional [0.01] Stackarray: mic/volt, Tip-tilt: arcsec/volt.

x_name
name for x coord of influence

y_name
name for y coord of influence

1.4.7 Param_target

class shesha_param.Param_target

Lambda
observation wavelength for each target

apod
boolean for apodizer

dms_seen
index of dms seen by the target

mag
magnitude for each target

ntargets
number of targets

set_Lambda()
Set the observation wavelength

Parameters 1 – (list of float) : observation wavelength for each target

set_apod()
Tells if the apodizer is used

The apodizer is used if a is not 0 :param a: (int) boolean for apodizer

set_dms_seen()
set the dms seen by the target

Parameters 1 – (list of int) : index for each dm

set_mag()
set the magnitude

Parameters 1 – (list of float) : magnitude for each target

set_nTargets()
Set the number of targets

Parameters n – (int) : number of targets

set_xpos()
Set the x positions on sky (in arcsec)

Parameters 1 – (list of float) : x positions on sky for each target

set_ypos ()

Set the y positions on sky (in arcsec)

Parameters **l** – (list of float) : y positions on sky for each target

set_zerop ()

Set the detector zero point

Parameters **z** – (float) : detector zero point

xpos

x positions on sky (in arcsec) for each target

ypos

y positions on sky (in arcsec) for each target

zerop

target flux for magnitude 0

1.4.8 Param_rtc

class shesha_param.Param_rtc

set_centroiders ()

Set the centroiders

Parameters **l** – (list of Param_centroider) : centroiders settings

set_controllers ()

Set the controller

Parameters **l** – (list of Param_controller) : controllers settings

set_nwfs ()

Set the number of wfs

Parameters **n** – (int) number of wfs

1.4.9 Param_centroider

class shesha_param.Param_centroider

interpmat

optional reference function(s) used for corr centroiding

nmax

number of brightest pixels

nwfs

index of wfs in y_wfs structure on which we want to do centroiding

set_nmax ()

Set the number of brightest pixels to use for bpcog

Parameters **n**: (int) : number of brightest pixels

set_nwfs ()

Set the index of wfs

Parameters **n** – (int) : index of wfs

set_sizex ()

Set sizex parameters for corr centroider (interp_mat size)

Parameters **s**: (long) : x size

set_sizey()
Set sizey parameters for corr centroider (interp_mat size)
Parameters *s*: (long) : y size

set_thresh()
Set the threshold for tcog
Parameters *t*: (float) : threshold

set_type()
Set the centroider type :param *t*: (str) : centroider type

set_type_fct()
Set the type of ref function
Parameters *f* – (str) : type of ref function

set_weights()
Set the weights to use with wcog or corr
Parameters *w*: (np.ndarray[ndim=3 ,dtype=np.float32_t]) : weights

set_width()
Set the width of the Gaussian
Parameters *w* – (float) : width of the gaussian

size_x
x-size for inter mat (corr)

size_y
x-size for inter mat (corr)

thresh
Threshold

type_centro
type of centroiding cog, tcog, bpcog, wcog, corr

type_fct
type of ref function gauss, file, model

weights
optional reference function(s) used for centroiding

width
width of the Gaussian

1.4.10 Param_controller

class shesha_param.Param_controller

TTcond
tipilt condition number for cmat filtering with mv controller

cmat
full control matrix

cured_ndivs
subdivision levels in cured

delay
loop delay [frames]

gain
loop gain

gmax
Maximum gain for modal optimization

gmin
Minimum gain for modal optimization

imat
full interaction matrix

maxcond
max condition number

modopti
Flag for modal optimization

nactu
number of controled actuator per dm

ndm
index of dms in controller

ngain
Number of tested gains

nmodes
Number of modes for M2V matrix (modal optimization)

nrec
Number of sample of open loop slopes for modal optimization computation

nvalid
number of valid subaps per wfs

nwfs
index of wfss in controller

set_TTcond()
Set the tipilt condition number for cmat filtering with mv controller
:param : (float) : tipilt condition number

set_cmat()
Set the full control matrix
Parameters **cmat** – (np.ndarray[ndim=2,dtype=np.float32_t]) : full control matrix

set_cured_ndivs()
Set the subdivision levels in cured
Parameters **c** – (long) : subdivision levels in cured

set_delay()
Set the loop delay expressed in frames
Parameters **d**: (float) :delay [frames]

set_gain()
Set the loop gain
Parameters **g**: (float) : loop gain

set_gmax()
Set the maximum gain for modal optimization
Parameters **g** – (float) : maximum gain for modal optimization

set_gmin()
Set the minimum gain for modal optimization
Parameters **g** – (float) : minimum gain for modal optimization

set_imat()
Set the full interaction matrix
Parameters **imat** – (np.ndarray[ndim=2,dtype=np.float32_t]) : full interaction matrix

set_maxcond()
Set the max condition number
:param : (float) : max condition number

set_modopti()
Set the flag for modal optimization
Parameters **m** – (int) : flag for modal optimization

set_nactu()
Set the number of controlled actuator
Parameters **l** – (list of int) : number of controlled actuator per dm

set_ndm()
Set the indices of dms
Parameters **l** – (list of int) : indices of dms

set_ngain()
Set the number of tested gains
Parameters **n** – (int) : number of tested gains

set_nkl()
Set the number of KL modes used for computation of covmat in case of minimum variance controller
Parameters **n** – (long) : number of KL modes

set_nmodes()
Set the number of modes for M2V matrix (modal optimization)
Parameters **n** – (int) : number of modes

set_nrec()
Set the number of sample of open loop slopes for modal optimization computation
Parameters **n** – (int) : number of sample

set_nvalid()
Set the number of valid subaps
Parameters **l** – (list of int) : number of valid subaps per wfs

set_nwfs()
Set the indices of wfs
Parameters **l** – (list of int) : indices of wfs

type_control
type of controller

1.4.11 get_classAttributes()

`shesha_param.get_classAttributes(Param_class)`
Return all the attribute names of the given Param_class
Parameters **Param_class** – shesha parameters class
Returns list of strings (attributes names)
Example `import shesha as ao` `get_classAttributes(ao.Param_wfs)`

1.4.12 `indices()`

`shesha_param.indices()`
DOCUMENT `indices(dim)`

Return a `dimxdimx2` array. First plane is the X indices of the pixels in the `dimxdim` array. Second plane contains the Y indices. Inspired by the Python `scipy` routine of the same name. New (June 12 2002), `dim` can either be:

- a single number `N` (e.g. 128) in which case the returned array are square (`NxN`)
- a Yorick array size, e.g. `[#dimension,N1,N2]`, in which case the returned array are `N1xN2`
- a vector `[N1,N2]`, same result as previous case

F.Rigaut 2002/04/03

Parameters `dim1`: (int) : first dimension
`dim2`: (int) : (optional) second dimension

1.4.13 `make_apodizer()`

`shesha_param.make_apodizer()`
TODO doc

Parameters (int) : `im`:
(int) : `pupd`:
(str) : `filename`:
(float) : `angle`:

1.4.14 `makegaussian()`

`shesha_param.makegaussian(size, fwhm, xc, yc)`

Returns a centered gaussian of specified size and `fwhm`. `norm` returns normalized 2d gaussian

Parameters `size`: (int) :
`fwhm`: (float) :
`xc`: (int) : (optional) center position on x axis
`yc`: (int) : (optional) center position on y axis
`norm`: (int) : (optional) normalization

1.4.15 `rotate()`

`shesha_param.rotate()`

Rotates an image of an angle “ang” (in DEGREES).

The center of rotation is `cx,cy`. A zoom factor can be applied.

(`cx,cy`) can be omitted :one will assume one rotates around the center of the image. If zoom is not specified, the default value of 1.0 is taken.

Parameters `im`: (`np.ndarray[ndim=3,dtype=np.float32_t]`) : array to rotate
`ang`: (float) : rotation angle (in degrees)
`cx`: (int) : (optional) rotation center on x axis (default: image center)
`cy`: (int) : (optional) rotation center on x axis (default: image center)

zoom: (float) : (opional) zoom factor (default =1.0)

1.4.16 rotate3d()

shesha_param.**rotate3d**()

Rotates an image of an angle “ang” (in DEGREES).

The center of rotation is cx,cy. A zoom factor can be applied.

(cx,cy) can be omitted :one will assume one rotates around the center of the image. If zoom is not specified, the default value of 1.0 is taken.

modif dg : allow to rotate a cube of images with one angle per image

Parameters im: (np.ndarray[ndim=3,dtype=np.float32_t]) : array to rotate

ang: (np.ndarray[ndim=1,dtype=np.float32_t]) : rotation angle (in degrees)

cx: (int) : (optional) rotation center on x axis (default: image center)

cy: (int) : (optional) rotation center on x axis (default: image center)

zoom: (float) : (opional) zoom factor (default =1.0)

1.5 shesha_rtc

1.5.1 Rtc

class shesha_rtc.**Rtc**

add_Controller()

Add a controller in the sutra_controller vector of the RTC on the GPU

Parameters nactu: (int) : number of actuators

delay: (float) : loop delay

type_control: (str) : controller’s type

dms: (Dms) : sutra_dms object (GPU)

type_dmseen: (char**) : dms indices controled by the controller

alt: (np.ndarray[ndim=1,dtype=np.float32_t]) : altitudes of the dms seen

ndm: (int) : number of dms controled

Nphi: (long) : number of pixels in the pupil (used in geo controler case only)

add_centroider()

Add a centroider in the sutra_centroiders vector of the RTC on the GPU

Parameters sensor: (Sensors) : sutra_sensors object (GPU)

nwfs : (long) : number of wfs

nvalid: (long) : number of valid subaps

type_centro: (str) : centroider’s type

offset: (float) :

scale: (float) :

applycontrol ()

Compute the DMs shapes from the commands computed in a `sutra_controller_object`. From the command vector, it computes the voltage command (adding pertrubation voltages, taking delay into account) and then apply it to the dms

Parameters `ncontro: (int)` : controller index

buildcmat ()

Compute the command matrix in a `sutra_controller_ls` object

Parameters `ncontro: (int)` : controller index

`nfilt: (int)` : number of modes to filter

`filt_tt: (int)` : (optional) flag to filter TT

buildcmatmv ()

Compute the command matrix in a `sutra_controller_mv` object

Parameters `ncontro: (int)` : controller index

`cond: (float)` : conditioning factor for the Cmm inversion

docentroids ()

Compute the centroids with `sutra_controller #ncontrol` object

Parameters `ncontrol: (optional)` controller's index

docentroids_geom ()

Compute the geometric centroids with `sutra_controller #ncontrol` object

Parameters `ncontrol: (optional)` controller's index

docontrol ()

Compute the command to apply on the DMs on a `sutra_controller` object

Parameters `ncontro: (int)` : controller index

docontrol_geo ()

Compute the command to apply on the DMs on a `sutra_controller_geo` object for the target direction

Parameters `ncontro: (int)` : controller index

docontrol_geo_onwfs ()

Compute the command to apply on the DMs on a `sutra_controller_geo` object for the wfs direction

Parameters `ncontro: (int)` : controller index

doimat ()

Compute the interaction matrix

Parameters `ncontro: (int)` : controller index

`g_dms: (Dms)` : Dms object

doimat_geom ()

Compute the interaction matrix by using a geometric centroiding method

Parameters `ncontro: (int)` : controller index

`g_dms: (Dms)` : Dms object

`geom: (int)` : type of geometric method (0 or 1)

getCenbuff ()

Return the centroids buffer from a `sutra_controller_ls` object. This buffer contains centroids from iteration `i-delay` to current iteration.

Parameters `ncontro: (int)` : controller index

Returns `data : (np.ndarray[ndim=2,dtype=np.float32_t])` : centroids buffer

getCentroids ()

Return the centroids computed by the `sutra_rtc` object

Parameters `ncontro`: (int) : controller's index

Returns `data` : (np.ndarray[ndim=1,dtype=np.float32_t]) : centroids (arcsec)

getCmmEigenvals ()

Return the eigen values of the Cmm decomposition in a `sutra_controller_mv` object

Parameters `ncontro`: (int) : controller index

Returns `eigenvals` : (np.ndarray[ndim=1,dtype=np.float32_t]) : eigenvalues

getCom ()

Return the command vector from a `sutra_controller` object

Parameters `ncontro`: (int) : controller index

Returns `data` : (np.ndarray[ndim=1,dtype=np.float32_t]) : command vector

getEigenvals ()

Return the eigen values of the imat decomposition in a `sutra_controller` object

Parameters `ncontro`: (int) : controller index

Returns `eigenvals` : (np.ndarray[ndim=1,dtype=np.float32_t]) : eigenvalues

getErr ()

Return the command increment (`cmat*slopes`) from a `sutra_controller_ls` object

Parameters `ncontro`: (int) : controller index

Returns `data` : (np.ndarray[ndim=1,dtype=np.float32_t]) : command increment

getU ()

Return the eigen modes matrix of the imat decomposition from a `sutra_controller_ls` object

Parameters `ncontro`: (int) : controller index

Returns `U` : (np.ndarray[ndim=2,dtype=np.float32_t]) : eigen modes matrix

getVoltage ()

Return the voltage vector that will be effectively applied to the DMs

Parameters `ncontro`: (int) : controller index

Returns `data` : (np.ndarray[ndim=1,dtype=np.float32_t]) : voltage vector

get_IFsparse ()

Get the influence functions matrix computed by the geo controller Return a `scipy.sparse` object which shape is (nactus,Npts in the pupil)

Parameters `ncontro`: (int) : controller index

Returns `IF` : (scipy.sparse) : influence functions matrix

get_cmat ()

Return the command matrix from a `sutra_controller` object

Parameters `ncontro`: (int) : controller index

Returns `cmat` : (np.ndarray[ndim=2,dtype=np.float32_t]) : command matrix

get_cmm ()

Return the Cmm matrix from a `sutra_controller_mv` object

Parameters `ncontro`: (int) : controller index

Returns `cmm` : (np.ndarray[ndim=2,dtype=np.float32_t]) : Cmm matrix

get_cphim ()

Return the Cphim matrix from a `sutra_controller_mv` object

:parameters; ncontro: (int) : controller index

Returns cphim : (np.ndarray[ndim=2,dtype=np.float32_t]) : Cphim matrix

get_imat()

Return the interaction matrix of a sutra_controller object

Parameters ncontro: (int) : controller index

Returns imat : (np.ndarray[ndim=2,dtype=np.float32_t]) : interaction matrix

get_mgain()

Return modal gains from sutra_controller

Parameters ncontro: (int) : controller index

Returns mgain : (np.ndarray[ndim=1,dtype=np.float32_t]) : modal gains

get_nfiltered()

Get the number of filtered modes for cmat computation

Parameters ncontro: (int) : controller index p_rtc: (Param_rtc) : rtc parameters

getcentroids()

Return the centroids computed by the sutra_rtc object If ncontrol <= d_control.size, return rtc.d_centroids Else, compute centroids from wfs[nwfs] with centroider[ncontrol]

Parameters ncontrol: (int) : controller's index

g_wfs: (Sensors) : (optional) sutra_sensors object

nwfs: (int) : (optional) number of wfs

Returns data : (np.ndarray[ndim=1,dtype=np.float32_t]) : centroids (arcsec)

getolmeas()

Return the reconstructed open-loop measurement from a sutra_controller_mv object

Parameters ncontro: (int) : controller index

Returns data : (np.ndarray[ndim=1,dtype=np.float32_t]) : reconstructed open-loop

imat_svd()

Compute the singular value decomposition of the interaction matrix

Parameters ncontro – controller index

init_modalOpti()

Initialize the modal optimization controller : compute the slopes-to-modes matrix and the transfer functions

Parameters ncontro: (int) : controller index

nmodes: (int) : number of modes

nrec: (int) : number of recorded open slopes measurements

M2V: (np.ndarray[ndim=2,dtype=np.float32_t]) : modes-to-volt matrix

gmin: (float) : minimum gain for modal optimization

gmax: (float) : maximum gain for modal optimization

ngain: (int) : Number of tested gains

Fs: (float) : sampling frequency

init_proj()

Initialize the projection matrix for sutra_controller_geo object. The projection matrix is $(IFt.IF)^*(-1) * IFt$ where IF is the DMs influence functions matrix

Parameters ncontro: (int) : controller index

dms: (Dms) : Dms object

indx_dm: (np.ndarray[ndim=1,dtype=np.int32_t]) : indices of where(pup) on DM screen

unitpervolt: (np.ndarray[ndim=1,dtype=np.float32_t]) : unitpervolt DM parameter

indx_pup: (np.ndarray[ndim=1,dtype=np.int32_t]) : indices of where(pup) on ipupil screen

loadOpenLoop()

Load an array of recoded open-loop measurements for modal optimization

Parameters ncontro: (int) : controller index

ol_slopes: (np.ndarray[ndim=2, dtype=np.float32_t]) : open-loop slopes

loadnoisemat()

Load the noise vector on a sutra_controller_mv object

Parameters ncontro: (int) : controller index

N: (np.ndarray[ndim=1,dtype=np.float32_t]) : noise vector

modalControlOptimization()

Compute the command matrix with modal control optimization

Parameter ncontro: controller index

rmcontrol()

Remove a controller

sensors_compslopes()

Compute the slopes in a sutra_wfs object. This function is equivalent to docentroids() but the centroids are stored in the sutra_wfs object instead of the sutra_rtc object

Parameters ncentro: (int) : centroider index

sensors_initbcube()

Initialize npix in the sutra_centroider_corr object (useless ?)

Parameters ncentro: (int) : centroider's index

sensors_initcorr()

Initialize sutra_centroider_corr object

Parameters ncentro: (int) : centroider's index

w: (np.ndarray[ndim=1,dtype=np.float32_t]) : weight

corr_norm: (np.ndarray[ndim=2,dtype=np.float32_t]) :

sizeX: (int) :

sizeY: (int) :

interpMat: ([ndim=2,dtype=np.float32_t]) :

sensors_initweights()

Load the weight array in sutra_centroider_wcog object

Parameters ncentro: (int) : centroider's index

w: (np.ndarray[ndim=2, dtype=np.float32_t]) : weight

setCentroids()

Set the centroids vector of a sutra_controller object to centro

Parameters ncontro: (int) : controller index centro: (np.ndarray[ndim=1,dtype=np.float32_t]) : centroids vector

setCom()

Set the command vector of a sutra_controller object to comvec

Parameters ncontro: (int) : controller index

setEigenvals()

Set the eigen values of the imat decomposition in a sutra_controller_ls object

Parameters ncontro: (int) : controller index

eigenvals: (np.ndarray[ndim=1,dtype=np.float32_t]) : eigen values

setU()

Set the eigen modes matrix of the imat decomposition in a sutra_controller_ls object

Parameters ncontro: (int) : controller index

U: (np.ndarray[ndim=2,dtype=np.float32_t]) : eigen modes matrix

set_cmat()

Set the command matrix on a sutra_controller object

Parameters ncontro: (int) : controller index

data: (np.ndarray[ndim=2,dtype=np.float32_t]) : command matrix to use

set_cmm()

Set the Cmm matrix on a sutra_controller_mv object

Parameters ncontro: (int) : controller index

data: (np.ndarray[ndim=2,dtype=np.float32_t]) : Cmm matrix

set_decayFactor()

Set the decay factor on a sutra_controller_generic object

Parameters ncontro: (int) : controller index

decay: (np.ndarray[ndim=1,dtype=np.float32_t]) : ask to Rico

set_gain()

Set the loop gain in sutra_controller object

Parameters ncontro: (int) : controller index

gain: (float) : loop gain

set_imat()

Set the interaction matrix on a sutra_controller object

Parameters ncontro: (int) : controller index

data: (np.ndarray[ndim=2,dtype=np.float32_t]) : interaction matrix to use

set_matE()

Set the matrix E on a sutra_controller_generic object

Parameters ncontro: (int) : controller index

matE: (np.ndarray[ndim=2,dtype=np.float32_t]) : ask to Rico

set_mgain()

Set modal gains in sutra_controller object

Parameters ncontro: (int) : controller index

mgain: (np.ndarray[ndim=1,dtype=np.float32_t]) : modal gains

set_openloop()

Set the openloop state to a sutra_controller object

Parameters ncontro: (int) : controller index

openloop: state of the controller

setnmax()

set the number of brightest pixels to consider for bpcog centroider

Parameters ncentro: (int) : centroider's index

nmax: (int) : number of brightest pixels

setthresh()

set threshold for the centroider #ncentro

Parameters ncentro: (int) : centroider's index

thresh: (float) : threshold

1.5.2 cmat_init()

shesha_rtc.cmat_init()

Compute the command matrix on the GPU

Parameters ncontro: (int) :

g_rtc: (Rtc) :

p_rtc: (Param_rtc) : rtc settings

p_wfs: (list of Param_wfs) : wfs settings

p_tel: (Param_tel) : telescope settings

clean: (int) : (optional) clean datafiles (imat, U, eigenv)

simul_name: (str) : (optional) simulation's name, use for data files' path

load: (dict) : (optional) dictionary of matrices to load and their path

1.5.3 compute_KL2V()

shesha_rtc.compute_KL2V()

Compute the Karhunen-Loeve to Volt matrix (transfer matrix between the KL space and volt space for a pzt dm)

Parameters p_dms: (list of Param_dm) : dms settings

controller: (Param_controller) : controller settings

1.5.4 correct_dm()

shesha_rtc.correct_dm()

Correct the geometry of the DMs using the imat (filter unseen actuators)

Parameters p_dms: (list of Param_dm) : dms settings

g_dms: (Dms) : Dms object

p_control: (Param_controller) : controller settings

p_geom: (Param_geom) : geom settings

imat: (np.ndarray) : interaction matrix

simul_name: (str) : simulation's name, use for data files' path

load: (dict) : (optional) dictionary of matrices to load and their path

1.5.5 create_interp_mat()

`shesha_rtc.create_interp_mat()`
TODO doc

Parameters dimx: (int) :
dimy: (int) :

1.5.6 create_nact_geom()

`shesha_rtc.create_nact_geom()`
Compute the DM coupling matrix

Param p_dms : (list of Param_dm) : dms parameters ndm : (int) : dm number
Returns Nact : (np.array(dtype=np.float64)) : the DM coupling matrix

1.5.7 create_piston_filter()

`shesha_rtc.create_piston_filter()`

1.5.8 doTomoMatrices()

`shesha_rtc.doTomoMatrices()`
Compute Cmm and Cphim matrices for the MV controller on GPU

Parameters g_wfs: (Sensors) :
p_wfs: (list of Param_wfs) : wfs settings
g_dms: (Dms) :
p_dms: (list of Param_dms) : dms settings
p_geom: (Param_geom) : geom settings
p_atmos: (Param_atmos) : atmos settings
g_atmos: (Atmos) :
p_tel: (Param_tel) : telescope settings

1.5.9 get_r0()

`shesha_rtc.get_r0()`
Compute r0 at lambda2 from r0 value at lambda1

Parameters r0_at_lambda1: (float) : r0 value at lambda1
lambda1: (float) : lambda1
lambda2: (float) : lambda2

1.5.10 imat_geom()

`shesha_rtc.imat_geom()`
Compute the interaction matrix with a geometric method

Parameters g_wfs: (Sensors) : Sensors object
p_wfs: (list of Param_wfs) : wfs settings
p_control: (Param_controller) : controller settings
g_dms: (Dms) : Dms object
p_dms: (list of Param_dm) : dms settings
meth: (int) : (optional) method type (0 or 1)

1.5.11 `imat_init()`

`shesha_rtc.imat_init()`

Initialize and compute the interaction matrix on the GPU

Parameters ncontro: (int) : controller's index
g_rtc: (Rtc) : Rtc object
p_rtc: (Param_rtc) : rtc settings
g_dms: (Dms) : Dms object
g_wfs: (Sensors) : Sensors object
p_wfs: (list of Param_wfs) : wfs settings
p_tel: (Param_tel) : telescope settings
clean: (int) : (optional) : clean datafiles (imat, U, eigenv)
simul_name: (str) : (optional) simulation's name, use for data files' path
load: (dict) : (optional) dictionary of matrices to load and their path

1.5.12 `manual_imat()`

`shesha_rtc.manual_imat()`

Compute the interaction matrix 'manually', ie without `sutra_rtc.doimat` method

Parameters g_rtc: (Rtc) : Rtc object
g_wfs: (Sensors) : Sensors object
g_dms: (Dms) : Dms object
p_dms: (list of Param_dm) : dm settings

1.5.13 `openLoopSlp()`

`shesha_rtc.openLoopSlp()`

Return a set of recorded open-loop slopes, usefull for modal control optimization

Parameters g_tel: (Telescope) : Telescope object
g_atm: (Atmos) : Atmos object
g_rtc: (Rtc) : Rtc object
nrec: (int) : number of samples to record
ncontro: (int) : controller's index
g_wfs: (Sensors) : Sensors object
p_wfs: (list of Param_wfs) : wfs settings

p_tar: (Param_target) : target settings

g_tar: (Target) : Target object

1.5.14 rtc_init()

shesha_rtc.rtc_init()

Initialize all the sutra_rtc objects : centroiders and controllers

Parameters g_tel: (Telescope) : Telescope object

g_wfs: (Sensors) : Sensors object

p_wfs: (list of Param_wfs) : wfs settings

g_dms: (Dms) : Dms object

p_dms: (list of Param_dms) : dms settings

p_geom: (Param_geom) : geom settings

p_atmos: (Param_atmos) : atmos settings

g_atmos: (Atmos) : Atmos object

p_tel: (Param_tel) : telescope settings

p_loop: (Param_loop) : loop settings

p_tar: (Param_target) : (optional) target settings

clean: (int) : (optional) clean datafiles (imat, U, eigenv, pztok, pztok)

brama: (int) : (optional) not implemented yet

doimat: (int) : (optional) force imat computation

simul_name: (str) : (optional) simulation's name, use for path to save data (imat, U...)

load: (dict) : (optional) dictionary of matrices to load and their path

Returns Rtc : (Rtc) : Rtc object

1.6 shesha_sensors

1.6.1 Sensors

class shesha_sensors.Sensors

Constructor: Sensors(nsensors,type_data,npup,nxsub,nvalid,nphase,pdiam,npix,nrebin,nfft,nftota,nphot,lgs,odevice,comm_si

Parameters nsensors: (int) :

type_data: list of strings) :

npup: (np.ndarray[ndim=1,dtype=np.int64_t]) :

nxsub: (np.ndarray[ndim=1,dtype=np.int64_t]) :

nvalid: (np.ndarray[ndim=1,dtype=np.int64_t]) :

nphase: (np.ndarray[ndim=1,dtype=np.int64_t]) :

pdiam: (np.ndarray[ndim=1,dtype=np.float32_t]) :

npix: (np.ndarray[ndim=1,dtype=np.int64_t]) :

nrebin: (np.ndarray[ndim=1,dtype=np.int64_t]) :

nfft: (np.ndarray[ndim=1,dtype=np.int64_t]) :

ntota: (np.ndarray[ndim=1,dtype=np.int64_t]) :
 nphot: (np.ndarray[ndim=1,dtype=np.float32_t]) :
 nphot4imat: (np.ndarray[ndim=1,dtype=np.float32_t]) :
 lgs: (np.ndarray[ndim=1,dtype=np.int32_t]) :
 odevice: (int) :
 comm_size: (int) : MPI communicator size
 rank: (int) : process rank

get_amplifoc()

Return the ‘amplifoc’ array of a given wfs

Parameters *n* – (int) : number of the wfs to get the ‘amplifoc’ from

get_amplifoc_pyr()

Return the ‘amplifoc’ array of a given wfs

Parameters *n* – (int) : number of the wfs to get the ‘amplifoc’ from

get_bincube()

Return the ‘bincube’ array of a given wfs

Parameters *n* – (int) : number of the wfs to get the ‘bincube’ from

get_bincubeNotNoisy()

Return the ‘bincube_not_noisy’ array of a given wfs. It’s the bincube before noise has been added

Parameters *n* – (int) : number of the wfs to get the ‘bincube_not_noisy’ from

get_bining()

Return the ‘binimg’ array of a given wfs

:param *n*: (int) : number of the wfs to get the ‘binimg’ from

:options for raw image computation *tel* (Telescope) : shesha telescope *atmos* (Atmos) : shesha atmos *dms* (Dms) : shesha dms

get_camclipup()

Return the ‘camclipup’ array of a given wfs

Parameters *n* – (int) : number of the wfs to get the ‘camclipup’ from

get_camclipup_pyr()

Return the ‘camclipup’ array of a given wfs in the pyr case

Parameters *n* – (int) : number of the wfs to get the ‘camclipup’ from

get_ftlgskern()

Return the ftlgskern array of a given wfs

Parameters *n* – (int) : number of the wfs to get the phase from

get_fttotim_pyr()

Return the ‘fttotim’ array of a given wfs

Parameters *n* – (int) : number of the wfs to get the ‘amplifoc’ from

get_hrimg_pyr()

Return the phase array of a given wfs

Parameters *n* – (int) : number of the wfs to get the phase from

get_imgtele()

Return the ‘image_telemetry’ array of a given wfs

Parameters *n* – (int) : number of the wfs to get the ‘image_telemetry’ from

:options for raw image computation tel (Telescope) : shesha telescope atmos (Atmos) : shesha atmos dms (Dms) : shesha dms

get_lgskern ()

Return the lgskern array of a given wfs

Parameters n – (int) : number of the wfs to get the phase from

get_offsets ()

Return the 'offset' array of a given wfs

Parameters n – (int) : number of the wfs to get the 'offset' from

get_phase ()

Return the phase array of a given wfs

Parameters n – (int) : number of the wfs to get the phase from

get_pyrimg ()

Return the image of a pyr wfs

Parameters n – (int) : number of the wfs to get the image from

get_pyrimg_hr ()

Return the high res image of a pyr wfs

Parameters n – (int) : number of the wfs to get the image from

get_rank ()

Return the rank of one of the sensors wfs

Parameters n – (int) : index of the wfs to get the rank for

get_slopes ()

Return the 'slopes' array of a given wfs

Parameters n – (int) : number of the wfs to get the 'slopes' from

get_subsum ()

Return the 'subsum' array of a given wfs

Parameters n – (int) : number of the wfs to get the 'subsum' from

reset_phase ()

Reset the phase's array of a given wfs

Parameters n – (int) : index of the given wfs

sensors_addlayer ()

Call function add_layer from the sutra_source of a sutra_wfs of the Sensors

Parameters i: (int) :

type_dm: (string) :

alt: (float) :

xoff: (float) :

yoff: (float) :

sensors_comping ()

TODO doc

Parameters n – (in) : index of the wfs

sensors_initarr ()

Call the function wfs_initarrays from a sutra_wfs of the Sensors

Parameters n: (int) : index of the wfs

wfs: (Param_wfs) :

sensors_initgs()

Call the function sensors_initgs

Parameters xpos: (np.ndarray[ndim=1,dtype=np.float32_t]) :

ypos: (np.ndarray[ndim=1,dtype=np.float32_t]) :

Lambda: (np.ndarray[ndim=1,dtype=np.float32_t]) :

mag: (np.ndarray[ndim=1,dtype=np.float32_t]) :

zerop: (float) :

size: (np.ndarray[ndim=1,dtype=np.int64_t]) :

noise: (np.ndarray[ndim=1,dtype=np.float32_t]) :

seed: (np.ndarray[ndim=1,dtype=np.int64_t]) :

sensors_trace()

Does the raytracing for the wfs phase screen in sutra_wfs

Parameters n: (int) :

type_trace: (str) ["all"] [raytracing across atmos and dms seen] "dm" : raytracing across dms seen only "atmos" : raytracing across atmos only

tel: (Telescope) :(optional) Telescope object

atmos: (Atmos) :(optional) Atmos object

dms: (Dms) : (optional) Dms object

rst: (int) : (optional) reset before raytracing if rst = 1

set_bincube()

Set the bincube of the WFS numner n

Parameters n: (int) : WFS number data: (np.ndarray[ndim=3,dtype=np.float32_t]) : bincube to use

set_phase()

Set the phase array of a given wfs

Parameters

- **n** – (int) : number of the wfs to get the phase from
- **data** – (np.ndarray) : the phase to set

slopes_geom()

Compute the geometric slopes in a sutra_wfs object

Parameters nsensor: (int) : wfs number

param t (int) : method (0 or 1)

1.6.2 bin2d()

shesha_sensors.bin2d()

Returns the input 2D array “array”, binned with the binning factor “binfact”. The input array X and/or Y dimensions needs not to be a multiple of “binfact”; The final/edge pixels are in effect replicated if needed. This routine prepares the parameters and calls the C routine `_bin2d`. The input array can be of type long, float or double. Last modified: Dec 15, 2003. Author: F.Rigaut SEE ALSO: `_bin2d`

Parmeters data_in: (np.ndarray) : data to binned

binfact: (int) : binning factor

1.6.3 `fft_goodsize()`

`shesha_sensors.fft_goodsize()`
find best size for a fft from size s

Parameters s: (long) size

1.6.4 `init_wfs_geom()`

`shesha_sensors.init_wfs_geom()`
Compute the geometry of WFSs: valid subaps, positions of the subaps, flux per subap, etc...

Parameters wfs: (Param_wfs) : wfs settings

wfs0: (Param_wfs) : reference wfs settings

n: (int) : index of the wfs (diplay information purpose only)

atmos: (Param_atmos) : atmos settings

tel: (Param_tel) : telescope settings

geom: (Param_geom) : geom settings

target: (Param_target) : target settings

loop: (Param_loop) : loop settings

init: (int) : (optional)

verbose: (int) : (optional) display informations if 0

1.6.5 `make_lgs_prof1d()`

`shesha_sensors.make_lgs_prof1d()`
same as `prep_lgs_prof` but cpu only. original routine from rico

Parameters p_tel: (Param_tel) : telescope settings

prof: (np.ndarray[`dtype=np.float32`]) : Na profile intensity, in arbitrary units

h: (np.ndarray[`dtype=np.float32`]) : altitude, in meters. h MUST be an array with EQUALLY spaced elements.

beam: (float) : size in arcsec of the laser beam

center: (string) : either “image” or “fourier” depending on where the centre should be.

1.6.6 `noise_cov()`

`shesha_sensors.noise_cov()`
Compute the diagonal of the noise covariance matrix for a SH WFS (arcsec^2)
Photon noise: $(\pi^2/2) * (1/N_{\text{photons}}) * (d/r0)^2 / (2 * \pi * d / \lambda)^2$ Electronic noise:
 $(\pi^2/3) * (wfs.noise^2 / N^2_{\text{photons}}) * wfs.npix^2 * (wfs.npix * wfs.pixsize * d / \lambda)^2 / (2 * \pi * d / \lambda)^2$

Parameters nw: wfs number p_wfs: (Param_wfs) : wfs settings p_atmos: (Param_atmos) :
atmos settings p_tel: (Param_tel) : telescope settings

Returns cov : (np.ndarray(`ndim=1, dtype=np.float64`)) : noise covariance diagonal

1.6.7 `prep_lgs_prof()`

`shesha_sensors.prep_lgs_prof()`

The function returns an image array(double,n,n) of a laser beacon elongated by perspective effect. It is obtained by convolution of a gaussian of width “lgsWidth” arcseconds, with the line of the sodium profile “prof”. The altitude of the profile is the array “h”.

parameters nsensors: (int) : wfs index
 p_tel: (Param_tel) : telescope settings
 prof: (np.ndarray[dtype=np.float32]) : Na profile intensity, in arbitrary units
 h: (np.ndarray[dtype=np.float32]) : altitude, in meters. h MUST be an array with EQUALLY spaced elements.
 beam: (float) : size in arcsec of the laser beam
 center: (string) : either “image” or “fourier” depending on where the centre should be.

Computation of LGS spot from the sodium profile: Everything is done here in 1D, because the Na profile is the result of the convolution of a function $P(x,y) = \text{profile}(x) \cdot \text{dirac}(y)$ by a gaussian function, for which variables x and y can be split : $\exp(-(x^2+y^2)/2.s^2) = \exp(-x^2/2.s^2) \cdot \exp(-y^2/2.s^2)$ The convolution is (symbol $\$$ denotes integral) $C(X,Y) = \$\$ \exp(-x^2/2.s^2) \cdot \exp(-y^2/2.s^2) \cdot \text{profile}(x-X) \cdot \text{dirac}(y-Y) dx dy$ First one performs the integration along y $C(X,Y) = \exp(-Y^2/2.s^2) \$ \exp(-x^2/2.s^2) \cdot \text{profile}(x-X) dx$ which shows that the profile can be computed by - convolving the 1-D profile - multiplying it in the 2nd dimension by a gaussian function

If one has to undersample the initial profile, then some structures may be “lost”. In this case, it’s better to try to “save” those structures by re-sampling the integral of the profile, and then derivating it afterwards. Now, if the initial profile is a coarse one, and that one has to oversample it, then a simple re-sampling of the profile is adequate.

1.6.8 `type_present()`

`shesha_sensors.type_present()`

Check the present types in a list

Parameters liste: (list of str) : list of types

pyr: (int) : set to 1 if the list contains “pyr” (0 else)
 roof: (int): set to 1 if the list contains “roof” (0 else)
 sh: (int) : set to 1 if the list contains “sh” (0 else)
 geo: (int) : set to 1 if the list contains “geo” (0 else)

return 1 if the wfs type is present (0 else)

1.6.9 `wfs_init()`

`shesha_sensors.wfs_init()`

Create and initialise a Sensors object

Parameters wfs: (list of Param_wfs) : wfs settings

p_atmos: (Param_atmos) : atmos settings
 p_tel: (Param_tel) : telescope settings
 p_geom: (Param_geom) : geom settings
 p_target: (Param_target) : target settings

p_loop: (Param_loop) : loop settings
dm: (list of Param_dm) : (optional) dms settings [=None]
comm_size: (int) : (optional) communicator size [=1]
rank: (int) : (optional) process rank [=0]

1.6.10 wheremax()

shesha_sensors.wheremax()

return the index of the maximum value of the list

Parameters **liste** – (list of values) : values to get the index of the maximum from

1.7 shesha_target

1.7.1 Target

class shesha_target.Target

Lambda

observation wavelength for each target

add_layer()

Add a phase screen dm or atmos as layers of turbulence

Parameters **n**: (int) : index of the target

l_type: (str) : “atmos” or “dm”

alt: (float) : altitude

xoff: (float) : x-offset

yoff: (float) : y-offset

apod

boolean for apodizer

atmos_trace()

Raytracing of the target through the atmosphere

Parameters **nTarget**: (int) : index of the target

atm: (atmos) : atmos to get through

tel: (Telescope) : telescope

dmtrace()

Raytracing of the target through the dms

Parameters **ntar**: (int) : index of the target

dms: (Dms) : dms to go through

reset: (int) : if >0, reset the screen before raytracing

do_phase_var: (int) : if 0, doesn't take the screen into account in the phase average
(unused)

get_amplipup()

Return the complex amplitude in the pupil plane of the target.

Parameters **nTarget** – (int) : index of the target

get_image()

Return the image from the target (or long exposure image according to the requested type)

Parameters `nTarget`: (int) : index of the target

`type_im`: (str) : type of the image to get (“se” or “le”)

`puponly`: (int) : if 1, image computed from phase on the pupil only

`comp_le`: (bool) : if False (default), the computed image is not taken into account in the LE image

get_phase()

Return the phase’s screen of the target

Parameters `nTarget` – (int) : index of the target

get_phasetele()

Return the telemetry phase of the target

Parameters `nTarget` – (int) : index of the target

Return data (np.ndarray(ndim=2,np.float32)) : phase screen

get_strehl()

Compute and return the target’s strehl

Parameters `nTarget` – (int) : index of the target

Return strehl (np.array(4,dtype=np.float32)) : [Strehl SE, Strehl LE, instantaneous phase variance over the pupil, average phase variance over the pupil]

init_strehlmeter()

Initialise target’s strehl

Parameters `nTarget` – (int) : index of the target

mag

magnitude for each target

ntargets

number of targets

reset_phase()

Reset the phase’s screen of the target

Parameters `nTarget` – (int) : index of the target

reset_strehl()

Reset the target’s strehl

Parameters `nTarget` – (int) : index of the target

set_phase()

Set the phase’s screen of the target

Parameters

- `nTarget` – (int) : index of the target
- `data` – (np.ndarray[ndim=2,dtype=np.float32_t]) : phase screen

xpos

x positions on sky (in arcsec) for each target

ypos

y positions on sky (in arcsec) for each target

1.7.2 `target_init()`

`shesha_target.target_init()`

Create a cython target from parametres structures

Parameters `ctxt: (naga_context)` :

`atm: (Param_atmos)` : atmos settings

`geom: (Param_geom)` : geom settings

`wfs: (Param_wfs)` : wfs settings

`dm: (Param_dm)` : dm settings

INDICES AND TABLES

- `genindex`
- `search`

A

add_centroider() (shesha_rtc.Rtc method), 25
 add_Controller() (shesha_rtc.Rtc method), 25
 add_dm() (shesha_dms.Dms method), 5
 add_layer() (shesha_target.Target method), 40
 add_screen() (shesha_atmos.Atmos method), 4
 alt (shesha_param.Param_atmos attribute), 15
 alt (shesha_param.Param_dm attribute), 16
 apod (shesha_param.Param_target attribute), 19
 apod (shesha_target.Target attribute), 40
 applycontrol() (shesha_rtc.Rtc method), 25
 Atmos (class in shesha_atmos), 4
 atmos_init() (in module shesha_atmos), 4
 atmos_seen (shesha_param.Param_wfs attribute), 11
 atmos_trace() (shesha_target.Target method), 40

B

beamsize (shesha_param.Param_wfs attribute), 11
 bin2d() (in module shesha), 3
 bin2d() (in module shesha_sensors), 37
 buildcmat() (shesha_rtc.Rtc method), 26
 buildcmatmv() (shesha_rtc.Rtc method), 26

C

cent (shesha_param.Param_geom attribute), 10
 center_name (shesha_param.Param_dm attribute), 16
 cmat (shesha_param.Param_controller attribute), 21
 cmat_init() (in module shesha_rtc), 31
 cobs (shesha_param.Param_tel attribute), 9
 comp_dmgeom() (in module shesha_dms), 7
 comp_oneactu() (shesha_dms.Dms method), 5
 compute_KL2V() (in module shesha_rtc), 31
 compute_klbasis() (in module shesha_dms), 8
 computeDMbasis() (in module shesha_dms), 8
 computeKLbasis() (shesha_dms.Dms method), 5
 correct_dm() (in module shesha_rtc), 31
 coupling (shesha_param.Param_dm attribute), 16
 create_interp_mat() (in module shesha_rtc), 32
 create_nact_geom() (in module shesha_rtc), 32
 create_piston_filter() (in module shesha_rtc), 32
 cube_name (shesha_param.Param_dm attribute), 16
 cured_ndivs (shesha_param.Param_controller attribute), 21

D

del_screen() (shesha_atmos.Atmos method), 4

delay (shesha_param.Param_controller attribute), 21
 deltax (shesha_param.Param_atmos attribute), 15
 deltax (shesha_param.Param_atmos attribute), 15
 diam (shesha_param.Param_tel attribute), 9
 dim_screens (shesha_param.Param_atmos attribute), 15
 disp() (shesha_atmos.Atmos method), 4
 dm_init() (in module shesha_dms), 8
 Dms (class in shesha_dms), 5
 dms_seen (shesha_param.Param_target attribute), 19
 dms_seen (shesha_param.Param_wfs attribute), 11
 dmtrace() (shesha_target.Target method), 40
 docentroids() (shesha_rtc.Rtc method), 26
 docentroids_geom() (shesha_rtc.Rtc method), 26
 docontrol() (shesha_rtc.Rtc method), 26
 docontrol_geo() (shesha_rtc.Rtc method), 26
 docontrol_geo_onwfs() (shesha_rtc.Rtc method), 26
 doimat() (shesha_rtc.Rtc method), 26
 doimat_geom() (shesha_rtc.Rtc method), 26
 doTomoMatrices() (in module shesha_rtc), 32

E

error_budget (shesha_param.Param_wfs attribute), 11

F

fft_goodsizes() (in module shesha_sensors), 38
 file_influ_hdf5 (shesha_param.Param_dm attribute), 17
 frac (shesha_param.Param_atmos attribute), 15
 fracsub (shesha_param.Param_wfs attribute), 11
 fssize (shesha_param.Param_wfs attribute), 12
 fstop (shesha_param.Param_wfs attribute), 12

G

gain (shesha_param.Param_controller attribute), 21
 geom_init() (shesha_param.Param_geom method), 10
 get_amplifoc() (shesha_sensors.Sensors method), 35
 get_amplifoc_pyr() (shesha_sensors.Sensors method), 35
 get_amplipup() (shesha_target.Target method), 40
 get_bincube() (shesha_sensors.Sensors method), 35
 get_bincubeNotNoisy() (shesha_sensors.Sensors method), 35
 get_bining() (shesha_sensors.Sensors method), 35
 get_camplipup() (shesha_sensors.Sensors method), 35
 get_camplipup_pyr() (shesha_sensors.Sensors method), 35

get_classAttributes() (in module shesha_param), 23
 get_cmat() (shesha_rtc.Rtc method), 27
 get_cmm() (shesha_rtc.Rtc method), 27
 get_cphim() (shesha_rtc.Rtc method), 27
 get_dm() (shesha_dms.Dms method), 6
 get_ftlgskern() (shesha_sensors.Sensors method), 35
 get_fitotim_pyr() (shesha_sensors.Sensors method), 35
 get_hrimg_pyr() (shesha_sensors.Sensors method), 35
 get_IFsparse() (shesha_rtc.Rtc method), 27
 get_image() (shesha_target.Target method), 40
 get_imat() (shesha_rtc.Rtc method), 28
 get_imgtele() (shesha_sensors.Sensors method), 35
 get_ipupil() (shesha_param.Param_geom method), 10
 get_KLbasis() (shesha_dms.Dms method), 6
 get_lgskern() (shesha_sensors.Sensors method), 36
 get_mgain() (shesha_rtc.Rtc method), 28
 get_mpupil() (shesha_param.Param_geom method), 10
 get_n() (shesha_param.Param_geom method), 10
 get_n1() (shesha_param.Param_geom method), 11
 get_n2() (shesha_param.Param_geom method), 11
 get_nfiltered() (shesha_rtc.Rtc method), 28
 get_offsets() (shesha_sensors.Sensors method), 36
 get_p1() (shesha_param.Param_geom method), 11
 get_p2() (shesha_param.Param_geom method), 11
 get_phase() (shesha_sensors.Sensors method), 36
 get_phase() (shesha_target.Target method), 41
 get_phasetele() (shesha_target.Target method), 41
 get_pyrimg() (shesha_sensors.Sensors method), 36
 get_pyrimghr() (shesha_sensors.Sensors method), 36
 get_r0() (in module shesha_rtc), 32
 get_rank() (shesha_sensors.Sensors method), 36
 get_screen() (shesha_atmos.Atmos method), 4
 get_slopes() (shesha_sensors.Sensors method), 36
 get_spupil() (shesha_param.Param_geom method), 11
 get_strehl() (shesha_target.Target method), 41
 get_subsum() (shesha_sensors.Sensors method), 36
 getCenbuff() (shesha_rtc.Rtc method), 26
 getCentroids() (shesha_rtc.Rtc method), 26
 getcentroids() (shesha_rtc.Rtc method), 28
 getCmmEigenvals() (shesha_rtc.Rtc method), 27
 getCom() (shesha_rtc.Rtc method), 27
 getComm() (shesha_dms.Dms method), 6
 getEigenvals() (shesha_rtc.Rtc method), 27
 getErr() (shesha_rtc.Rtc method), 27
 getInflu() (shesha_dms.Dms method), 6
 getolmeas() (shesha_rtc.Rtc method), 28
 getU() (shesha_rtc.Rtc method), 27
 getVoltage() (shesha_rtc.Rtc method), 27
 gmax (shesha_param.Param_controller attribute), 21
 gmin (shesha_param.Param_controller attribute), 22
 gsalt (shesha_param.Param_wfs attribute), 12
 gsmag (shesha_param.Param_wfs attribute), 12

H

hyst (shesha_param.Param_dm attribute), 17

I

imat (shesha_param.Param_controller attribute), 22

imat_geom() (in module shesha_rtc), 32
 imat_init() (in module shesha_rtc), 33
 imat_svd() (shesha_rtc.Rtc method), 28
 indices() (in module shesha), 3
 indices() (in module shesha_param), 24
 init_modalOpti() (shesha_rtc.Rtc method), 28
 init_proj() (shesha_rtc.Rtc method), 28
 init_strehlmeter() (shesha_target.Target method), 41
 init_wfs_geom() (in module shesha_sensors), 38
 interpmat (shesha_param.Param_centroider attribute), 20
 ittime (shesha_param.Param_loop attribute), 9

L

L0 (shesha_param.Param_atmos attribute), 15
 Lambda (shesha_param.Param_target attribute), 19
 Lambda (shesha_param.Param_wfs attribute), 11
 Lambda (shesha_target.Target attribute), 40
 laserpower (shesha_param.Param_wfs attribute), 12
 lgsreturnperwatt (shesha_param.Param_wfs attribute), 12
 list_alt() (shesha_atmos.Atmos method), 4
 lltx (shesha_param.Param_wfs attribute), 12
 llty (shesha_param.Param_wfs attribute), 12
 load_kl() (shesha_dms.Dms method), 6
 load_pzt() (shesha_dms.Dms method), 6
 load_tt() (shesha_dms.Dms method), 7
 loadnoisemat() (shesha_rtc.Rtc method), 29
 loadOpenLoop() (shesha_rtc.Rtc method), 29

M

mag (shesha_param.Param_target attribute), 19
 mag (shesha_target.Target attribute), 41
 make_apodizer() (in module shesha_param), 24
 make_lgs_prof1d() (in module shesha_sensors), 38
 make_pzt_dm() (in module shesha_dms), 8
 makegaussian() (in module shesha), 3
 makegaussian() (in module shesha_param), 24
 manual_imat() (in module shesha_rtc), 33
 margin (shesha_param.Param_dm attribute), 17
 margin_in (shesha_param.Param_dm attribute), 17
 margin_out (shesha_param.Param_dm attribute), 17
 maxcond (shesha_param.Param_controller attribute), 22
 modalControlOptimization() (shesha_rtc.Rtc method), 29
 modopti (shesha_param.Param_controller attribute), 22
 move_atmos() (shesha_atmos.Atmos method), 4

N

nact (shesha_param.Param_dm attribute), 17
 nactu (shesha_param.Param_controller attribute), 22
 nbrmissing (shesha_param.Param_tel attribute), 9
 ndm (shesha_param.Param_controller attribute), 22
 ngain (shesha_param.Param_controller attribute), 22
 niter (shesha_param.Param_loop attribute), 9
 nkl (shesha_param.Param_dm attribute), 17
 nmax (shesha_param.Param_centroider attribute), 20

nmodes (shesha_param.Param_controller attribute), 22
 noise (shesha_param.Param_wfs attribute), 12
 noise_cov() (in module shesha_sensors), 38
 nphotons4imat (shesha_param.Param_wfs attribute), 12
 npix (shesha_param.Param_wfs attribute), 12
 nrec (shesha_param.Param_controller attribute), 22
 nscreens (shesha_param.Param_atmos attribute), 15
 ntargets (shesha_param.Param_target attribute), 19
 ntargets (shesha_target.Target attribute), 41
 nvalid (shesha_param.Param_controller attribute), 22
 nwfs (shesha_param.Param_centroider attribute), 20
 nwfs (shesha_param.Param_controller attribute), 22
 nxsub (shesha_param.Param_wfs attribute), 12

O

oneactu() (shesha_dms.Dms method), 7
 openloop (shesha_param.Param_wfs attribute), 12
 openLoopSlp() (in module shesha_rtc), 33
 optthroughput (shesha_param.Param_wfs attribute), 12

P

Param_atmos (class in shesha_param), 15
 Param_centroider (class in shesha_param), 20
 Param_controller (class in shesha_param), 21
 Param_dm (class in shesha_param), 16
 Param_geom (class in shesha_param), 10
 Param_loop (class in shesha_param), 9
 Param_rtc (class in shesha_param), 20
 Param_target (class in shesha_param), 19
 Param_tel (class in shesha_param), 9
 Param_wfs (class in shesha_param), 11
 pixsize (shesha_param.Param_wfs attribute), 12
 prep_lgs_prof() (in module shesha_sensors), 39
 proftype (shesha_param.Param_wfs attribute), 12
 pupangle (shesha_param.Param_tel attribute), 9
 pupdiam (shesha_param.Param_geom attribute), 11
 pupixsize (shesha_param.Param_atmos attribute), 15
 pupoffset (shesha_param.Param_dm attribute), 17
 push4imat (shesha_param.Param_dm attribute), 17
 pyr_ampl (shesha_param.Param_wfs attribute), 12
 pyr_loc (shesha_param.Param_wfs attribute), 12
 pyr_npts (shesha_param.Param_wfs attribute), 12
 pyrtype (shesha_param.Param_wfs attribute), 12

R

r0 (shesha_param.Param_atmos attribute), 15
 referr (shesha_param.Param_tel attribute), 9
 remove_dm() (shesha_dms.Dms method), 7
 reset_phase() (shesha_sensors.Sensors method), 36
 reset_phase() (shesha_target.Target method), 41
 reset_strehl() (shesha_target.Target method), 41
 resetdm() (shesha_dms.Dms method), 7
 rmcontrol() (shesha_rtc.Rtc method), 29
 rotate() (in module shesha_param), 24
 rotate3d() (in module shesha_param), 25
 Rtc (class in shesha_rtc), 25
 rtc_init() (in module shesha_rtc), 34

S

Sensors (class in shesha_sensors), 34
 sensors_addlayer() (shesha_sensors.Sensors method), 36
 sensors_compimg() (shesha_sensors.Sensors method), 36
 sensors_compslopes() (shesha_rtc.Rtc method), 29
 sensors_initarr() (shesha_sensors.Sensors method), 36
 sensors_initbcube() (shesha_rtc.Rtc method), 29
 sensors_initcorr() (shesha_rtc.Rtc method), 29
 sensors_initgs() (shesha_sensors.Sensors method), 36
 sensors_initweights() (shesha_rtc.Rtc method), 29
 sensors_trace() (shesha_sensors.Sensors method), 37
 set_alt() (shesha_param.Param_atmos method), 15
 set_alt() (shesha_param.Param_dm method), 17
 set_altna() (shesha_param.Param_wfs method), 12
 set_apod() (shesha_param.Param_target method), 19
 set_atmos_seen() (shesha_param.Param_wfs method), 13
 set_beamsize() (shesha_param.Param_wfs method), 13
 set_bincube() (shesha_sensors.Sensors method), 37
 set_cent() (shesha_param.Param_geom method), 11
 set_center_name() (shesha_param.Param_dm method), 17
 set_centroiders() (shesha_param.Param_rtc method), 20
 set_cmat() (shesha_param.Param_controller method), 22
 set_cmat() (shesha_rtc.Rtc method), 30
 set_cmm() (shesha_rtc.Rtc method), 30
 set_cobs() (shesha_param.Param_tel method), 9
 set_comm() (shesha_dms.Dms method), 7
 set_controllers() (shesha_param.Param_rtc method), 20
 set_coupling() (shesha_param.Param_dm method), 17
 set_cube_name() (shesha_param.Param_dm method), 17
 set_cured_ndivs() (shesha_param.Param_controller method), 22
 set_decayFactor() (shesha_rtc.Rtc method), 30
 set_delay() (shesha_param.Param_controller method), 22
 set_deltax() (shesha_param.Param_atmos method), 15
 set_deltay() (shesha_param.Param_atmos method), 16
 set_diam() (shesha_param.Param_tel method), 9
 set_dim_screens() (shesha_param.Param_atmos method), 16
 set_dms_seen() (shesha_param.Param_target method), 19
 set_dms_seen() (shesha_param.Param_wfs method), 13
 set_errorBudget() (shesha_param.Param_wfs method), 13
 set_file_influ_hdf5() (shesha_param.Param_dm method), 17
 set_frac() (shesha_param.Param_atmos method), 16
 set_fracsub() (shesha_param.Param_wfs method), 13
 set_fssize() (shesha_param.Param_wfs method), 13
 set_fstop() (shesha_param.Param_wfs method), 13
 set_full_comm() (shesha_dms.Dms method), 7

`set_gain()` (shesha_param.Param_controller method), 22

`set_gain()` (shesha_rtc.Rtc method), 30

`set_gmax()` (shesha_param.Param_controller method), 22

`set_gmin()` (shesha_param.Param_controller method), 22

`set_gsalt()` (shesha_param.Param_wfs method), 13

`set_gsmag()` (shesha_param.Param_wfs method), 13

`set_i1()` (shesha_param.Param_dm method), 17

`set_imat()` (shesha_param.Param_controller method), 22

`set_imat()` (shesha_rtc.Rtc method), 30

`set_influ()` (shesha_param.Param_dm method), 17

`set_ittime()` (shesha_param.Param_loop method), 9

`set_j1()` (shesha_param.Param_dm method), 17

`set_kernel()` (shesha_param.Param_wfs method), 13

`set_L0()` (shesha_param.Param_atmos method), 15

`set_Lambda()` (shesha_param.Param_target method), 19

`set_Lambda()` (shesha_param.Param_wfs method), 12

`set_laserpower()` (shesha_param.Param_wfs method), 13

`set_lgsreturnperwatt()` (shesha_param.Param_wfs method), 13

`set_lltx()` (shesha_param.Param_wfs method), 13

`set_llty()` (shesha_param.Param_wfs method), 13

`set_mag()` (shesha_param.Param_target method), 19

`set_margin()` (shesha_param.Param_dm method), 17

`set_margin_in()` (shesha_param.Param_dm method), 18

`set_margin_out()` (shesha_param.Param_dm method), 18

`set_matE()` (shesha_rtc.Rtc method), 30

`set_maxcond()` (shesha_param.Param_controller method), 23

`set_mgain()` (shesha_rtc.Rtc method), 30

`set_modopti()` (shesha_param.Param_controller method), 23

`set_nact()` (shesha_param.Param_dm method), 18

`set_nactu()` (shesha_param.Param_controller method), 23

`set_nbrmissing()` (shesha_param.Param_tel method), 9

`set_ndm()` (shesha_param.Param_controller method), 23

`set_ngain()` (shesha_param.Param_controller method), 23

`set_niter()` (shesha_param.Param_loop method), 9

`set_nkl()` (shesha_param.Param_controller method), 23

`set_nmax()` (shesha_param.Param_centroider method), 20

`set_nmodes()` (shesha_param.Param_controller method), 23

`set_noise()` (shesha_param.Param_wfs method), 13

`set_nphotons4imat()` (shesha_param.Param_wfs method), 14

`set_npix()` (shesha_param.Param_wfs method), 14

`set_nrec()` (shesha_param.Param_controller method), 23

`set_nscreens()` (shesha_param.Param_atmos method), 16

`set_nTargets()` (shesha_param.Param_target method), 19

`set_ntotact()` (shesha_param.Param_dm method), 18

`set_nvalid()` (shesha_param.Param_controller method), 23

`set_nwfs()` (shesha_param.Param_centroider method), 20

`set_nwfs()` (shesha_param.Param_controller method), 23

`set_nwfs()` (shesha_param.Param_rtc method), 20

`set_nxsub()` (shesha_param.Param_wfs method), 14

`set_openloop()` (shesha_param.Param_wfs method), 14

`set_openloop()` (shesha_rtc.Rtc method), 30

`set_optthroughput()` (shesha_param.Param_wfs method), 14

`set_pattern()` (shesha_param.Param_dm method), 18

`set_phase()` (shesha_sensors.Sensors method), 37

`set_phase()` (shesha_target.Target method), 41

`set_pixsize()` (shesha_param.Param_wfs method), 14

`set_profna()` (shesha_param.Param_wfs method), 14

`set_proftype()` (shesha_param.Param_wfs method), 14

`set_pupangle()` (shesha_param.Param_tel method), 9

`set_pupdiam()` (shesha_param.Param_geom method), 11

`set_pupixsize()` (shesha_param.Param_atmos method), 16

`set_push4imat()` (shesha_param.Param_dm method), 18

`set_pyr_ampl()` (shesha_param.Param_wfs method), 14

`set_pyr_loc()` (shesha_param.Param_wfs method), 14

`set_pyr_npts()` (shesha_param.Param_wfs method), 14

`set_pyrtype()` (shesha_param.Param_wfs method), 14

`set_r0()` (shesha_param.Param_atmos method), 16

`set_referr()` (shesha_param.Param_tel method), 9

`set_seeds()` (shesha_param.Param_atmos method), 16

`set_sizeX()` (shesha_param.Param_centroider method), 20

`set_sizey()` (shesha_param.Param_centroider method), 20

`set_spiders_type()` (shesha_param.Param_tel method), 10

`set_ssize()` (shesha_param.Param_geom method), 11

`set_std_piston()` (shesha_param.Param_tel method), 10

`set_std_tt()` (shesha_param.Param_tel method), 10

`set_t_spiders()` (shesha_param.Param_tel method), 10

`set_thresh()` (shesha_param.Param_centroider method), 21

`set_thresh()` (shesha_param.Param_dm method), 18

`set_TTcond()` (shesha_param.Param_controller method), 22

`set_type()` (shesha_param.Param_centroider method), 21

`set_type()` (shesha_param.Param_dm method), 18

`set_type()` (shesha_param.Param_wfs method), 14

`set_type_ap()` (shesha_param.Param_tel method), 10

`set_type_fct()` (shesha_param.Param_centroider

method), 21
 set_unitpervolt() (shesha_param.Param_dm method), 18
 set_weights() (shesha_param.Param_centroider method), 21
 set_width() (shesha_param.Param_centroider method), 21
 set_winddir() (shesha_param.Param_atmos method), 16
 set_windspeed() (shesha_param.Param_atmos method), 16
 set_x_name() (shesha_param.Param_dm method), 18
 set_xpos() (shesha_param.Param_dm method), 18
 set_xpos() (shesha_param.Param_target method), 19
 set_xpos() (shesha_param.Param_wfs method), 14
 set_y_name() (shesha_param.Param_dm method), 18
 set_ypos() (shesha_param.Param_dm method), 18
 set_ypos() (shesha_param.Param_target method), 19
 set_ypos() (shesha_param.Param_wfs method), 15
 set_zenithangle() (shesha_param.Param_geom method), 11
 set_zerop() (shesha_param.Param_target method), 20
 set_zerop() (shesha_param.Param_wfs method), 15
 setCentroids() (shesha_rtc.Rtc method), 29
 setCom() (shesha_rtc.Rtc method), 29
 setEigenvals() (shesha_rtc.Rtc method), 30
 setnmax() (shesha_rtc.Rtc method), 31
 setthresh() (shesha_rtc.Rtc method), 31
 setU() (shesha_rtc.Rtc method), 30
 shape_dm() (shesha_dms.Dms method), 7
 sizex (shesha_param.Param_centroider attribute), 21
 sizey (shesha_param.Param_centroider attribute), 21
 slopes_geom() (shesha_sensors.Sensors method), 37
 spiders_type (shesha_param.Param_tel attribute), 10
 ssize (shesha_param.Param_geom attribute), 11
 std_piston (shesha_param.Param_tel attribute), 10
 std_tt (shesha_param.Param_tel attribute), 10

T

t_spiders (shesha_param.Param_tel attribute), 10
 Target (class in shesha_target), 40
 target_init() (in module shesha_target), 42
 thresh (shesha_param.Param_centroider attribute), 21
 thresh (shesha_param.Param_dm attribute), 18
 TTcond (shesha_param.Param_controller attribute), 21
 type_ap (shesha_param.Param_tel attribute), 10
 type_centro (shesha_param.Param_centroider attribute), 21
 type_control (shesha_param.Param_controller attribute), 23
 type_dm (shesha_param.Param_dm attribute), 19
 type_fct (shesha_param.Param_centroider attribute), 21
 type_pattern (shesha_param.Param_dm attribute), 19
 type_present() (in module shesha_sensors), 39
 type_wfs (shesha_param.Param_wfs attribute), 15

U

unitpervolt (shesha_param.Param_dm attribute), 19

W

weights (shesha_param.Param_centroider attribute), 21
 wfs_init() (in module shesha_sensors), 39
 wheremax() (in module shesha_sensors), 40
 width (shesha_param.Param_centroider attribute), 21
 winddir (shesha_param.Param_atmos attribute), 16
 windspeed (shesha_param.Param_atmos attribute), 16

X

x_name (shesha_param.Param_dm attribute), 19
 xpos (shesha_param.Param_target attribute), 20
 xpos (shesha_param.Param_wfs attribute), 15
 xpos (shesha_target.Target attribute), 41

Y

y_name (shesha_param.Param_dm attribute), 19
 ypos (shesha_param.Param_target attribute), 20
 ypos (shesha_param.Param_wfs attribute), 15
 ypos (shesha_target.Target attribute), 41

Z

zenithangle (shesha_param.Param_geom attribute), 11
 zerop (shesha_param.Param_target attribute), 20
 zerop (shesha_param.Param_wfs attribute), 15