COMPASS Documentation

Release py3 tag

COMPASS team

Sep 28, 2017

CONTENTS

1 Contents: 3
- 1.1 naga_context 3
  - 1.1.1 naga_context 3
- 1.2 naga_obj 3
  - 1.2.1 naga_obj_Int1D 3
  - 1.2.2 naga_obj_Int2D 4
  - 1.2.3 naga_obj_Int3D 5
  - 1.2.4 naga_obj_Int4D 6
  - 1.2.5 naga_obj_UInt1D 7
  - 1.2.6 naga_obj_UInt2D 8
  - 1.2.7 naga_obj_UInt3D 9
  - 1.2.8 naga_obj_UInt4D 9
  - 1.2.9 naga_obj_Float1D 10
  - 1.2.10 naga_obj_Float2D 12
  - 1.2.11 naga_obj_Float3D 15
  - 1.2.12 naga_obj_Float4D 16
  - 1.2.13 naga_obj_Double1D 18
  - 1.2.14 naga_obj_Double2D 20
  - 1.2.15 naga_obj_Double3D 22
  - 1.2.16 naga_obj_Double4D 24
  - 1.2.17 naga_obj_ComplexS1D 26
  - 1.2.18 naga_obj_ComplexS2D 27
  - 1.2.19 naga_obj_ComplexS3D 30
  - 1.2.20 naga_obj_ComplexS4D 31
  - 1.2.21 naga_obj_ComplexD1D 33
  - 1.2.22 naga_obj_ComplexD2D 34
  - 1.2.23 naga_obj_ComplexD3D 37
  - 1.2.24 naga_obj_ComplexD4D 38
- 1.3 naga_host_obj 40
  - 1.3.1 naga_host_obj_Int1D 40
  - 1.3.2 naga_host_obj_Int2D 40
  - 1.3.3 naga_host_obj_Int3D 41
  - 1.3.4 naga_host_obj_Int4D 41
  - 1.3.5 naga_host_obj_UInt1D 41
  - 1.3.6 naga_host_obj_UInt2D 42
  - 1.3.7 naga_host_obj_UInt3D 42
  - 1.3.8 naga_host_obj_UInt4D 42
  - 1.3.9 naga_host_obj_Float1D 43
  - 1.3.10 naga_host_obj_Float2D 43
  - 1.3.11 naga_host_obj_Float3D 44
  - 1.3.12 naga_host_obj_Float4D 44
  - 1.3.13 naga_host_obj_Double1D 44
  - 1.3.14 naga_host_obj_Double2D 45
  - 1.3.15 naga_host_obj_Double3D 45

1.3.16 naga_host_obj_Double4D . . . 46
1.3.17 naga_host_obj_ComplexS1D . . . 46
1.3.18 naga_host_obj_ComplexS2D . . . 46
1.3.19 naga_host_obj_ComplexS3D . . . 47
1.3.20 naga_host_obj_ComplexS4D . . . 47
1.3.21 naga_host_obj_ComplexD1D . . . 47
1.3.22 naga_host_obj_ComplexD2D . . . 48
1.3.23 naga_host_obj_ComplexD3D . . . 48
1.3.24 naga_host_obj_ComplexD4D . . . 49
1.4 naga_sparse_obj . . . 49
1.4.1 naga_sparse_obj_Double . . . 49
1.4.2 naga_sparse_obj_Float . . . 49
1.5 naga_magma . . . 49
1.5.1 getri_Double() . . . 49
1.5.2 getri_Float() . . . 49
1.5.3 getri_host_Double() . . . 50
1.5.4 getri_host_Float() . . . 50
1.5.5 potri_Double() . . . 50
1.5.6 potri_Float() . . . 50
1.5.7 potri_host_Double() . . . 50
1.5.8 potri_host_Float() . . . 50
1.5.9 svd_Double() . . . 50
1.5.10 svd_Float() . . . 50
1.5.11 svd_host_Double() . . . 50
1.5.12 svd_host_Float() . . . 50
1.5.13 syevd_Double() . . . 51
1.5.14 syevd_Float() . . . 51
1.5.15 syevd_host_Double() . . . 51
1.5.16 syevd_host_Float() . . . 51
1.6 naga_timer . . . 51
1.6.1 naga_timer . . . 51
1.6.2 threadSync() . . . 51
1.7 shesha_ao . . . 51
1.8 shesha_config . . . 56
1.9 shesha_constants . . . 70
1.10 shesha_init . . . 71
1.11 shesha_sim . . . 72
1.12 shesha_util . . . 73

2 Indices and tables **87**

Python Module Index **89**

Index **91**

Version PDF

CHAPTER

ONE

CONTENTS:

1.1 naga_context

1.1.1 naga_context

class naga_context.**naga_context**

get_activeDevice()
Return the index of actual activated device

get_cudaDriverGetVersion()
Return the version number of the installed CUDA driver

get_cudaRuntimeGetVersion()
Return the version number of the installed CUDA Runtime

get_device_names()
Return names of devices.

get_magma_info()
Return the information of the installed MAGMA

get_ndevice()
Return number of device.

set_activeDevice()
Activate a device.

newDevice – int device to activate silent – int (default=1)

set_activeDeviceForCpy()
Activate a device.

newDevice – int device to activate silent – int (default=1)

set_activeDeviceForce()
Activate a device.

newDevice – int device to activate silent – int (default=1)

1.2 naga_obj

1.2.1 naga_obj_Int1D

class naga_obj.**naga_obj_Int1D**

activateDevice()
Activate the device used by the current naga_obj.

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Int1D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Int1D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.int32) of 1 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.int32) of 1 dimension(s)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=1, dtype=np.int32_t] data): data – np.int32: data to copy from host to device

is_rng_init()

reset()
Set naga_obj to zero.

1.2.2 naga_obj_Int2D

class naga_obj.**naga_obj_Int2D**

activateDevice()
Activate the device used by the current naga_obj.

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Int2D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Int2D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.int32) of 2 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.int32) of 2 dimension(s)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=2, dtype=np.int32_t] data): data – np.int32: data to copy from host to device

is_rng_init()

reset()
Set naga_obj to zero.

1.2.3 naga_obj_Int3D

class naga_obj.**naga_obj_Int3D**

activateDevice()
Activate the device used by the current naga_obj.

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Int3D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Int3D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.int32) of 3 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.int32) of 3 dimension(s)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=3, dtype=np.int32_t] data): data – np.int32: data to copy from host to device

is_rng_init()

reset()
Set naga_obj to zero.

1.2.4 naga_obj_Int4D

class naga_obj.**naga_obj_Int4D**

activateDevice()
Activate the device used by the current naga_obj.

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Int4D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Int4D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.int32) of 4 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.int32) of 4 dimension(s)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=4, dtype=np.int32_t] data): data – np.int32: data to copy from host to device

is_rng_init()

reset()
Set naga_obj to zero.

1.2.5 naga_obj_UInt1D

class naga_obj.**naga_obj_UInt1D**

activateDevice()
Activate the device used by the current naga_obj.

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_UInt1D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_UInt1D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.uint32) of 1 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.uint32) of 1 dimension(s)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=1, dtype=np.uint32_t] data): data – np.uint32: data to copy from host to device

is_rng_init()

reset()
Set naga_obj to zero.

1.2.6 naga_obj_UInt2D

class naga_obj.**naga_obj_UInt2D**

activateDevice()
Activate the device used by the current naga_obj.

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_UInt2D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_UInt2D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.uint32) of 2 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.uint32) of 2 dimension(s)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=2, dtype=np.uint32_t] data): data – np.uint32: data to copy from host to device

is_rng_init()

reset()
Set naga_obj to zero.

1.2.7 naga_obj_UInt3D

class naga_obj.**naga_obj_UInt3D**

activateDevice()
Activate the device used by the current naga_obj.

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_UInt3D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_UInt3D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.uint32) of 3 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.uint32) of 3 dimension(s)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=3, dtype=np.uint32_t] data): data – np.uint32: data to copy from host to device

is_rng_init()

reset()
Set naga_obj to zero.

1.2.8 naga_obj_UInt4D

class naga_obj.**naga_obj_UInt4D**

activateDevice()
Activate the device used by the current naga_obj.

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_UInt4D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_UInt4D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.uint32) of 4 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.uint32) of 4 dimension(s)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=4, dtype=np.uint32_t] data): data – np.uint32: data to copy from host to device

is_rng_init()

reset()
Set naga_obj to zero.

1.2.9 naga_obj_Float1D

class naga_obj.**naga_obj_Float1D**

activateDevice()
Activate the device used by the current naga_obj.

asum()
Cublas asum. Return the sum of the absolute values of the data's elements

axpy()
cublas axpy

dest – naga_obj_Float1D alpha– np.float32 beta – np.float32 Return dest=alpha*self +dest

copy()
Cublas copy

src – naga_obj_Float1D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Float1D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Float1D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.float32) of 1 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.float32) of 1 dimension(s)

dot()
Cublas dot

src – naga_obj_Float1D return the dot product of src and self.

fft()

ger()
Cublas ger

Y – naga_obj_Float1D alpha – np.float32 (default = 1) A – naga_obj_Float2D (default = None)

Return A=alpha*self*t(y)+A

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=1, dtype=np.float32_t] data): data – np.float32: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

init_prng()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

is_rng_init()

montagn()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

nrm2()
Cublas nrm2. Return the Euclidean norm

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.float32: caling factor self = alpha.self

sum()
Return the sum of the data's elements

swap()
Cublas swap

src – naga_obj_Float1D Swap data contents of naga objects self and src.

1.2.10 naga_obj_Float2D

class naga_obj.**naga_obj_Float2D**

activateDevice()
Activate the device used by the current naga_obj.

asum()
Cublas asum. Return the sum of the absolute values of the data's elements

copy()
Cublas copy

src – naga_obj_Float2D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Float2D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Float2D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.float32) of 2 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.float32) of 2 dimension(s)

dgmm()
Cublas dgmm

X – naga_obj_Float1D side – char (default = 'l') C – naga_obj_Float2D (default = None)

Return self*diag(X) if sidec='l' diag(X)*self otherwise

dot()
Cublas dot

src – naga_obj_Float2D return the dot product of src and self.

fft()

geam()
Cublas geam

B – naga_obj_Float2D alpha – np.float32 (default = 1) beta – np.float32 (default = 0) opA – char (default = 'n') opB – char (default = 'n') C – naga_obj_Float2D (default = None)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix return C= alpha*opA(self)+beta*opB(B)

gemm()
Cublas gemm

B – naga_obj_Float2D opA – char (default = 'n') opB – char (default = 'n') alpha – np.float32 (default = 1) C – naga_obj_Float2D (default = None) beta – np.float32 (default = 0)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix Return C=alpha opA(self)*opB(B)+beta*C

gemv()
Cublas gemv

Vx – naga_obj_Float1D alpha – np.float32 (default = 1) Vy – naga_obj_Float1D (default = None) beta – np.float32 (default = 0) Return Vy=alpha*self*Vx+beta*Vy

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=2, dtype=np.float32_t] data): data – np.float32: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

init_prng()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

is_rng_init()

montagn()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

nrm2()
Cublas nrm2. Return the Euclidean norm

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.float32: caling factor self = alpha.self

sum()
Return the sum of the data's elements

swap()
Cublas swap

src – naga_obj_Float2D Swap data contents of naga objects self and src.

symm()
Cublas symm

B – naga_obj_Float2D side – char (default = 'l') alpha – np.float32 (default =1) C – naga_obj_Float2D (default = None) beta – np.float32 (default =0)

return alpha*A*B+beta*C if side='l' alpha*B*A+beta*C otherwise

symv()
Cublas symv

Vx – naga_obj_Float1D alpha – np.float32 (default = 1) Vy – naga_obj_Float1D (default = None) beta – np.float32 (default = 0) Return Vy=alpha*self*Vx+beta*Vy

syrk()
Cublas syrk

opA – char (default = 'n') alpha – np.float32 (default = 1) C – naga_obj_Float2D (default = None) beta – np.float32 (default = 0)

opA: transposition on matrix self 'n': no transposition 't':transpose matrix Return alpha*opA(self)*opA(self)T+beta*C

syrkx()
Cublas syrkx

B – naga_obj_Float2D opA – char (default = 'n') apha – np.float32 (default = 1) C – naga_obj_Float2D (default = None) beta – np.float32 (default = 0)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix Return alpha*opA(self)*opB(B)T+beta*C

transpose()

1.2.11 naga_obj_Float3D

class naga_obj.**naga_obj_Float3D**

activateDevice()
Activate the device used by the current naga_obj.

asum()
Cublas asum. Return the sum of the absolute values of the data's elements

copy()
Cublas copy

src – naga_obj_Float3D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Float3D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Float3D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.float32) of 3 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.float32) of 3 dimension(s)

dot()
Cublas dot

src – naga_obj_Float3D return the dot product of src and self.

fft()

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=3, dtype=np.float32_t] data): data – np.float32: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

init_prng()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

is_rng_init()

montagn()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

nrm2()
Cublas nrm2. Return the Euclidean norm

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.float32: caling factor self = alpha.self

sum()
Return the sum of the data's elements

swap()
Cublas swap

src – naga_obj_Float3D Swap data contents of naga objects self and src.

1.2.12 naga_obj_Float4D

class naga_obj.**naga_obj_Float4D**

activateDevice()
Activate the device used by the current naga_obj.

asum()
Cublas asum. Return the sum of the absolute values of the data's elements

copy()
Cublas copy

src – naga_obj_Float4D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Float4D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Float4D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.float32) of 4 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.float32) of 4 dimension(s)

dot()
Cublas dot

src – naga_obj_Float4D return the dot product of src and self.

fft()

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=4, dtype=np.float32_t] data): data – np.float32: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

init_prng()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

is_rng_init()

montagn()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

nrm2()
Cublas nrm2. Return the Euclidean norm

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.float32: caling factor self = alpha.self

sum()
Return the sum of the data's elements

swap()
Cublas swap

src – naga_obj_Float4D Swap data contents of naga objects self and src.

1.2.13 naga_obj_Double1D

class naga_obj.**naga_obj_Double1D**

activateDevice()
Activate the device used by the current naga_obj.

asum()
Cublas asum. Return the sum of the absolute values of the data's elements

axpy()
cublas axpy

dest – naga_obj_Double1D alpha– np.float64 beta – np.float64 Return dest=alpha*self +dest

copy()
Cublas copy

src – naga_obj_Double1D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Double1D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Double1D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.float64) of 1 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.float64) of 1 dimension(s)

dot()
Cublas dot

src – naga_obj_Double1D return the dot product of src and self.

fft()

ger()
Cublas ger

Y – naga_obj_Double1D alpha – np.float64 (default = 1) A – naga_obj_Double2D (default = None)

Return A=alpha*self*t(y)+A

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=1, dtype=np.float64_t] data): data – np.float64: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

init_prng()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

is_rng_init()

montagn()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

nrm2()
Cublas nrm2. Return the Euclidean norm

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.float64: caling factor self = alpha.self

sum()
Return the sum of the data's elements

swap()
Cublas swap

src – naga_obj_Double1D Swap data contents of naga objects self and src.

1.2.14 naga_obj_Double2D

class naga_obj.**naga_obj_Double2D**

activateDevice()
Activate the device used by the current naga_obj.

asum()
Cublas asum. Return the sum of the absolute values of the data's elements

copy()
Cublas copy

src – naga_obj_Double2D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Double2D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Double2D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.float64) of 2 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.float64) of 2 dimension(s)

dgmm()
Cublas dgmm

X – naga_obj_Double1D side – char (default = 'l') C – naga_obj_Double2D (default = None)

Return self*diag(X) if sidec='l' diag(X)*self otherwise

dot()
Cublas dot

src – naga_obj_Double2D return the dot product of src and self.

fft()

geam()
Cublas geam

B – naga_obj_Double2D alpha – np.float64 (default = 1) beta – np.float64 (default = 0) opA – char (default = 'n') opB – char (default = 'n') C – naga_obj_Double2D (default = None)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix return C= alpha*opA(self)+beta*opB(B)

gemm()
Cublas gemm

B – naga_obj_Double2D opA – char (default = 'n') opB – char (default = 'n') alpha – np.float64 (default = 1) C – naga_obj_Double2D (default = None) beta – np.float64 (default = 0)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix Return C=alpha opA(self)*opB(B)+beta*C

gemv()
Cublas gemv

Vx – naga_obj_Double1D alpha – np.float64 (default = 1) Vy – naga_obj_Double1D (default = None) beta – np.float64 (default = 0) Return Vy=alpha*self*Vx+beta*Vy

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=2, dtype=np.float64_t] data): data – np.float64: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

init_prng()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

is_rng_init()

montagn()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

nrm2()
Cublas nrm2. Return the Euclidean norm

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.float64: caling factor self = alpha.self

sum()
Return the sum of the data's elements

swap()
Cublas swap

src – naga_obj_Double2D Swap data contents of naga objects self and src.

symm()
Cublas symm

B – naga_obj_Double2D side – char (default = 'l') alpha – np.float64 (default =1) C – naga_obj_Double2D (default = None) beta – np.float64 (default =0)

return alpha*A*B+beta*C if side='l' alpha*B*A+beta*C otherwise

symv()
Cublas symv

Vx – naga_obj_Double1D alpha – np.float64 (default = 1) Vy – naga_obj_Double1D (default = None) beta – np.float64 (default = 0) Return Vy=alpha*self*Vx+beta*Vy

syrk()
Cublas syrk

opA – char (default = 'n') alpha – np.float64 (default = 1) C – naga_obj_Double2D (default = None) beta – np.float64 (default = 0)

opA: transposition on matrix self 'n': no transposition 't':transpose matrix Return alpha*opA(self)*opA(self)T+beta*C

syrkx()
Cublas syrkx

B – naga_obj_Double2D opA – char (default = 'n') apha – np.float64 (default = 1) C – naga_obj_Double2D (default = None) beta – np.float64 (default = 0)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix Return alpha*opA(self)*opB(B)T+beta*C

transpose()

1.2.15 naga_obj_Double3D

class naga_obj.**naga_obj_Double3D**

activateDevice()
Activate the device used by the current naga_obj.

asum()
Cublas asum. Return the sum of the absolute values of the data's elements

copy()
Cublas copy

src – naga_obj_Double3D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Double3D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Double3D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.float64) of 3 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.float64) of 3 dimension(s)

dot()
Cublas dot

src – naga_obj_Double3D return the dot product of src and self.

fft()

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=3, dtype=np.float64_t] data): data – np.float64: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

init_prng()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

is_rng_init()

montagn()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

nrm2()
Cublas nrm2. Return the Euclidean norm

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.float64: caling factor self = alpha.self

sum()
Return the sum of the data's elements

swap()
Cublas swap

src – naga_obj_Double3D Swap data contents of naga objects self and src.

1.2.16 `naga_obj_Double4D`

class naga_obj.**naga_obj_Double4D**

activateDevice()
Activate the device used by the current naga_obj.

asum()
Cublas asum. Return the sum of the absolute values of the data's elements

copy()
Cublas copy

src – naga_obj_Double4D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_Double4D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_Double4D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.float64) of 4 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.float64) of 4 dimension(s)

dot()
Cublas dot

src – naga_obj_Double4D return the dot product of src and self.

fft()

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=4, dtype=np.float64_t] data): data – np.float64: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

init_prng()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

is_rng_init()

montagn()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

nrm2()
Cublas nrm2. Return the Euclidean norm

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.float64: caling factor self = alpha.self

sum()
Return the sum of the data's elements

swap()
Cublas swap

src – naga_obj_Double4D Swap data contents of naga objects self and src.

1.2.17 naga_obj_ComplexS1D

class naga_obj.**naga_obj_ComplexS1D**

activateDevice()
Activate the device used by the current naga_obj.

axpy()
cublas axpy

dest – naga_obj_ComplexS1D alpha– np.complex64 beta – np.complex64 Return dest=alpha*self +dest

copy()
Cublas copy

src – naga_obj_ComplexS1D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_ComplexS1D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_ComplexS1D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.complex64) of 1 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.complex64) of 1 dimension(s)

dot()
Cublas dot

src – naga_obj_ComplexS1D return the dot product of src and self.

fft()
Compute fft, using "cufftExec"

dest – naga_obj (default = None) dir – integer (default 1)

dir: fft's direction if dest is None, inplace fft (only available for C2C fft)

Return dest= fft(self,dir)

ger()
Cublas ger

Y – naga_obj_ComplexS1D alpha – np.complex64 (default = 1) A – naga_obj_ComplexS2D (default = None)

Return A=alpha*self*t(y)+A

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=1, dtype=np.complex64_t] data): data – np.complex64: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

is_rng_init()

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.complex64: caling factor self = alpha.self

swap()
Cublas swap

src – naga_obj_ComplexS1D Swap data contents of naga objects self and src.

1.2.18 naga_obj_ComplexS2D

class naga_obj.**naga_obj_ComplexS2D**

activateDevice()
Activate the device used by the current naga_obj.

copy()
Cublas copy

src – naga_obj_ComplexS2D Copy data from src into self

copyFrom()

Copy the data from src to the current naga_obj.

src – naga_obj_ComplexS2D: object to copy the data from.

copyInto()

Copy data from current naga_obj to dest.

dest – naga_obj_ComplexS2D: object to copy the data into.

device2host()

Copy data from device to host.

return np.ndarray(dtype=np.complex64) of 2 dimension(s)

device2hostOpt()

Copy data from device o_data to host.

return np.ndarray(dtype=np.complex64) of 2 dimension(s)

dgmm()

Cublas dgmm

X – naga_obj_ComplexS1D side – char (default = 'l') C – naga_obj_ComplexS2D (default = None)

Return self*diag(X) if sidec='l' diag(X)*self otherwise

dot()

Cublas dot

src – naga_obj_ComplexS2D return the dot product of src and self.

fft()

Compute fft, using "cufftExec"

dest – naga_obj (default = None) dir – integer (default 1)

dir: fft's direction if dest is None, inplace fft (only available for C2C fft)

Return dest= fft(self,dir)

geam()

Cublas geam

B – naga_obj_ComplexS2D alpha – np.complex64 (default = 1) beta – np.complex64 (default = 0) opA – char (default = 'n') opB – char (default = 'n') C – naga_obj_ComplexS2D (default = None)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix return C= alpha*opA(self)+beta*opB(B)

gemm()

Cublas gemm

B – naga_obj_ComplexS2D opA – char (default = 'n') opB – char (default = 'n') alpha – np.complex64 (default = 1) C – naga_obj_ComplexS2D (default = None) beta – np.complex64 (default = 0)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix Return C=alpha opA(self)*opB(B)+beta*C

gemv()

Cublas gemv

Vx – naga_obj_ComplexS1D alpha – np.complex64 (default = 1) Vy – naga_obj_ComplexS1D (default = None) beta – np.complex64 (default = 0) Return Vy=alpha*self*Vx+beta*Vy

getCarma_ptr()

Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()

Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=2, dtype=np.complex64_t] data): data – np.complex64: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

is_rng_init()

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.complex64: caling factor self = alpha.self

swap()
Cublas swap

src – naga_obj_ComplexS2D Swap data contents of naga objects self and src.

symm()
Cublas symm

B – naga_obj_ComplexS2D side – char (default = 'l') alpha – np.complex64 (default =1) C – naga_obj_ComplexS2D (default = None) beta – np.complex64 (default =0)

return alpha*A*B+beta*C if side='l' alpha*B*A+beta*C otherwise

symv()
Cublas symv

Vx – naga_obj_ComplexS1D alpha – np.complex64 (default = 1) Vy – naga_obj_ComplexS1D (default = None) beta – np.complex64 (default = 0) Return Vy=alpha*self*Vx+beta*Vy

syrk()
Cublas syrk

opA – char (default = 'n') alpha – np.complex64 (default = 1) C – naga_obj_ComplexS2D (default = None) beta – np.complex64 (default = 0)

opA: transposition on matrix self 'n': no transposition 't':transpose matrix Return alpha*opA(self)*opA(self)T+beta*C

syrkx()
Cublas syrkx

B – naga_obj_ComplexS2D opA – char (default = 'n') apha – np.complex64 (default = 1) C – naga_obj_ComplexS2D (default = None) beta – np.complex64 (default = 0)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix Return alpha*opA(self)*opB(B)T+beta*C

transpose()

1.2.19 naga_obj_ComplexS3D

class naga_obj.**naga_obj_ComplexS3D**

activateDevice()
Activate the device used by the current naga_obj.

copy()
Cublas copy

src – naga_obj_ComplexS3D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_ComplexS3D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_ComplexS3D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.complex64) of 3 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.complex64) of 3 dimension(s)

dot()
Cublas dot

src – naga_obj_ComplexS3D return the dot product of src and self.

fft()
Compute fft, using "cufftExec"

dest – naga_obj (default = None) dir – integer (default 1)

dir: fft's direction if dest is None, inplace fft (only available for C2C fft)

Return dest= fft(self,dir)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=3, dtype=np.complex64_t] data): data – np.complex64: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

is_rng_init()

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.complex64: caling factor self = alpha.self

swap()
Cublas swap

src – naga_obj_ComplexS3D Swap data contents of naga objects self and src.

1.2.20 naga_obj_ComplexS4D

class naga_obj.**naga_obj_ComplexS4D**

activateDevice()
Activate the device used by the current naga_obj.

copy()
Cublas copy

src – naga_obj_ComplexS4D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_ComplexS4D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_ComplexS4D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.complex64) of 4 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.complex64) of 4 dimension(s)

dot()
Cublas dot

src – naga_obj_ComplexS4D return the dot product of src and self.

fft()
Compute fft, using "cufftExec"

dest – naga_obj (default = None) dir – integer (default 1)

dir: fft's direction if dest is None, inplace fft (only available for C2C fft)

Return dest= fft(self,dir)

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=4, dtype=np.complex64_t] data): data – np.complex64: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

is_rng_init()

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.complex64: caling factor self = alpha.self

swap()
Cublas swap

src – naga_obj_ComplexS4D Swap data contents of naga objects self and src.

1.2.21 naga_obj_ComplexD1D

class naga_obj.**naga_obj_ComplexD1D**

activateDevice()
Activate the device used by the current naga_obj.

axpy()
cublas axpy

dest – naga_obj_ComplexD1D alpha– np.complex128 beta – np.complex128 Return dest=alpha*self +dest

copy()
Cublas copy

src – naga_obj_ComplexD1D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_ComplexD1D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_ComplexD1D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.complex128) of 1 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.complex128) of 1 dimension(s)

dot()
Cublas dot

src – naga_obj_ComplexD1D return the dot product of src and self.

fft()
Compute fft, using "cufftExec"

dest – naga_obj (default = None) dir – integer (default 1)

dir: fft's direction if dest is None, inplace fft (only available for C2C fft)

Return dest= fft(self,dir)

ger()
Cublas ger

Y – naga_obj_ComplexD1D alpha – np.complex128 (default = 1) A – naga_obj_ComplexD2D (default = None)

Return A=alpha*self*t(y)+A

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=1, dtype=np.complex128_t] data): data – np.complex128: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

is_rng_init()

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.complex128: caling factor self = alpha.self

swap()
Cublas swap

src – naga_obj_ComplexD1D Swap data contents of naga objects self and src.

1.2.22 naga_obj_ComplexD2D

class naga_obj.**naga_obj_ComplexD2D**

activateDevice()
Activate the device used by the current naga_obj.

copy()
Cublas copy

src – naga_obj_ComplexD2D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_ComplexD2D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_ComplexD2D: object to copy the data into.

device2host()
Copy data from device to host.

return np.ndarray(dtype=np.complex128) of 2 dimension(s)

device2hostOpt()
Copy data from device o_data to host.

return np.ndarray(dtype=np.complex128) of 2 dimension(s)

dgmm()
Cublas dgmm

X – naga_obj_ComplexD1D side – char (default = 'l') C – naga_obj_ComplexD2D (default = None)

Return self*diag(X) if sidec='l' diag(X)*self otherwise

dot()
Cublas dot

src – naga_obj_ComplexD2D return the dot product of src and self.

fft()
Compute fft, using "cufftExec"

dest – naga_obj (default = None) dir – integer (default 1)

dir: fft's direction if dest is None, inplace fft (only available for C2C fft)

Return dest= fft(self,dir)

geam()
Cublas geam

B – naga_obj_ComplexD2D alpha – np.complex128 (default = 1) beta – np.complex128 (default = 0) opA – char (default = 'n') opB – char (default = 'n') C – naga_obj_ComplexD2D (default = None)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix return C= alpha*opA(self)+beta*opB(B)

gemm()
Cublas gemm

B – naga_obj_ComplexD2D opA – char (default = 'n') opB – char (default = 'n') alpha – np.complex128 (default = 1) C – naga_obj_ComplexD2D (default = None) beta – np.complex128 (default = 0)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix Return C=alpha opA(self)*opB(B)+beta*C

gemv()
Cublas gemv

Vx – naga_obj_ComplexD1D alpha – np.complex128 (default = 1) Vy – naga_obj_ComplexD1D (default = None) beta – np.complex128 (default = 0) Return Vy=alpha*self*Vx+beta*Vy

getCarma_ptr()
Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()
Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()
Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=2, dtype=np.complex128_t] data): data – np.complex128: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

is_rng_init()

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.complex128: caling factor self = alpha.self

swap()
Cublas swap

src – naga_obj_ComplexD2D Swap data contents of naga objects self and src.

symm()
Cublas symm

B – naga_obj_ComplexD2D side – char (default = 'l') alpha – np.complex128 (default =1) C – naga_obj_ComplexD2D (default = None) beta – np.complex128 (default =0)

return alpha*A*B+beta*C if side='l' alpha*B*A+beta*C otherwise

symv()
Cublas symv

Vx – naga_obj_ComplexD1D alpha – np.complex128 (default = 1) Vy – naga_obj_ComplexD1D (default = None) beta – np.complex128 (default = 0) Return Vy=alpha*self*Vx+beta*Vy

syrk()
Cublas syrk

opA – char (default = 'n') alpha – np.complex128 (default = 1) C – naga_obj_ComplexD2D (default = None) beta – np.complex128 (default = 0)

opA: transposition on matrix self 'n': no transposition 't':transpose matrix Return alpha*opA(self)*opA(self)T+beta*C

syrkx()

Cublas syrkx

B – naga_obj_ComplexD2D opA – char (default = 'n') apha – np.complex128 (default = 1) C – naga_obj_ComplexD2D (default = None) beta – np.complex128 (default = 0)

opA (opB): transposition on matrix self (B), 'n': no transposition 't':transpose matrix Return alpha*opA(self)*opB(B)T+beta*C

transpose()

1.2.23 `naga_obj_ComplexD3D`

class naga_obj.**naga_obj_ComplexD3D**

activateDevice()

Activate the device used by the current naga_obj.

copy()

Cublas copy

src – naga_obj_ComplexD3D Copy data from src into self

copyFrom()

Copy the data from src to the current naga_obj.

src – naga_obj_ComplexD3D: object to copy the data from.

copyInto()

Copy data from current naga_obj to dest.

dest – naga_obj_ComplexD3D: object to copy the data into.

device2host()

Copy data from device to host.

return np.ndarray(dtype=np.complex128) of 3 dimension(s)

device2hostOpt()

Copy data from device o_data to host.

return np.ndarray(dtype=np.complex128) of 3 dimension(s)

dot()

Cublas dot

src – naga_obj_ComplexD3D return the dot product of src and self.

fft()

Compute fft, using "cufftExec"

dest – naga_obj (default = None) dir – integer (default 1)

dir: fft's direction if dest is None, inplace fft (only available for C2C fft)

Return dest= fft(self,dir)

getCarma_ptr()

Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()

Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()

Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()
Return the device used by the current naga_obj.

getNbElem()
Return the number of elements of the naga object.

getValues()

get_Dims()
Return the dimensions of the naga_obj.

host2device()
Copy data from host to device.

host2device(np.ndarray[ndim=3, dtype=np.complex128_t] data): data – np.complex128: data to copy from host to device

imax()
Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()
Cublas amin

Return the smallest index of the minimum absolute magnitude element.

is_rng_init()

random()
Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()
Set naga_obj to zero.

scale()
Cublas scal

alpha – np.complex128: caling factor self = alpha.self

swap()
Cublas swap

src – naga_obj_ComplexD3D Swap data contents of naga objects self and src.

1.2.24 naga_obj_ComplexD4D

class naga_obj.**naga_obj_ComplexD4D**

activateDevice()
Activate the device used by the current naga_obj.

copy()
Cublas copy

src – naga_obj_ComplexD4D Copy data from src into self

copyFrom()
Copy the data from src to the current naga_obj.

src – naga_obj_ComplexD4D: object to copy the data from.

copyInto()
Copy data from current naga_obj to dest.

dest – naga_obj_ComplexD4D: object to copy the data into.

device2host()

Copy data from device to host.

return np.ndarray(dtype=np.complex128) of 4 dimension(s)

device2hostOpt()

Copy data from device o_data to host.

return np.ndarray(dtype=np.complex128) of 4 dimension(s)

dot()

Cublas dot

src – naga_obj_ComplexD4D return the dot product of src and self.

fft()

Compute fft, using "cufftExec"

dest – naga_obj (default = None) dir – integer (default 1)

dir: fft's direction if dest is None, inplace fft (only available for C2C fft)

Return dest= fft(self,dir)

getCarma_ptr()

Return the pointer value to the carma object of the naga (as an integer, type:uintptr_t).

getContext()

Return a pointer to the carma_context associated with the current naga_obj.

getData_ptr()

Return the pointer value to the naga data (as an integer, type:uintptr_t).

getDevice()

Return the device used by the current naga_obj.

getNbElem()

Return the number of elements of the naga object.

getValues()

get_Dims()

Return the dimensions of the naga_obj.

host2device()

Copy data from host to device.

host2device(np.ndarray[ndim=4, dtype=np.complex128_t] data): data – np.complex128: data to copy from host to device

imax()

Cublas amax.

Return the smallest index of the maximum absolute magnitude element.

imin()

Cublas amin

Return the smallest index of the minimum absolute magnitude element.

is_rng_init()

random()

Generate random values for this naga datas.

seed – integer: seed for random function (default:1234)

reset()

Set naga_obj to zero.

scale()
Cublas scal

alpha – np.complex128: caling factor self = alpha.self

swap()
Cublas swap

src – naga_obj_ComplexD4D Swap data contents of naga objects self and src.

1.3 naga_host_obj

1.3.1 naga_host_obj_Int1D

class naga_host_obj.**naga_host_obj_Int1D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Int1D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Int1D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.2 naga_host_obj_Int2D

class naga_host_obj.**naga_host_obj_Int2D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Int2D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Int2D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.3 naga_host_obj_Int3D

class naga_host_obj.**naga_host_obj_Int3D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Int3D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Int3D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.4 naga_host_obj_Int4D

class naga_host_obj.**naga_host_obj_Int4D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Int4D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Int4D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.5 naga_host_obj_UInt1D

class naga_host_obj.**naga_host_obj_UInt1D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_UInt1D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_UInt1D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.6 naga_host_obj_UInt2D

class naga_host_obj.**naga_host_obj_UInt2D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_UInt2D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_UInt2D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.7 naga_host_obj_UInt3D

class naga_host_obj.**naga_host_obj_UInt3D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_UInt3D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_UInt3D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.8 naga_host_obj_UInt4D

class naga_host_obj.**naga_host_obj_UInt4D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_UInt4D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_UInt4D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.9 naga_host_obj_Float1D

class naga_host_obj.**naga_host_obj_Float1D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Float1D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Float1D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.10 naga_host_obj_Float2D

class naga_host_obj.**naga_host_obj_Float2D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Float2D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Float2D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.11 naga_host_obj_Float3D

class naga_host_obj.**naga_host_obj_Float3D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Float3D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Float3D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.12 naga_host_obj_Float4D

class naga_host_obj.**naga_host_obj_Float4D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Float4D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Float4D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.13 naga_host_obj_Double1D

class naga_host_obj.**naga_host_obj_Double1D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Double1D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Double1D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.14 naga_host_obj_Double2D

class naga_host_obj.**naga_host_obj_Double2D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Double2D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Double2D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.15 naga_host_obj_Double3D

class naga_host_obj.**naga_host_obj_Double3D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Double3D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Double3D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.16 naga_host_obj_Double4D

class naga_host_obj.**naga_host_obj_Double4D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_Double4D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_Double4D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.17 naga_host_obj_ComplexS1D

class naga_host_obj.**naga_host_obj_ComplexS1D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_ComplexS1D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_ComplexS1D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.18 naga_host_obj_ComplexS2D

class naga_host_obj.**naga_host_obj_ComplexS2D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_ComplexS2D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_ComplexS2D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.19 naga_host_obj_ComplexS3D

class naga_host_obj.**naga_host_obj_ComplexS3D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_ComplexS3D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_ComplexS3D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.20 naga_host_obj_ComplexS4D

class naga_host_obj.**naga_host_obj_ComplexS4D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_ComplexS4D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_ComplexS4D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.21 naga_host_obj_ComplexD1D

class naga_host_obj.**naga_host_obj_ComplexD1D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_ComplexD1D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_ComplexD1D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.22 naga_host_obj_ComplexD2D

class naga_host_obj.**naga_host_obj_ComplexD2D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_ComplexD2D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_ComplexD2D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.23 naga_host_obj_ComplexD3D

class naga_host_obj.**naga_host_obj_ComplexD3D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_ComplexD3D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_ComplexD3D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.3.24 naga_host_obj_ComplexD4D

class naga_host_obj.**naga_host_obj_ComplexD4D**

cpy_obj_from()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

src – naga_obj_ComplexD4D: object to copy from

cpy_obj_into()
Copy into naga_host_obj (cpu storage) the data from a naga_obj (gpu storage).

dest – naga_obj_ComplexD4D: object to copy into

getData()

getNbElem()

get_Dims()

get_host_obj_ptr()
Return the pointer to the naga_host_obj.

setData()

1.4 naga_sparse_obj

1.4.1 naga_sparse_obj_Double

class naga_sparse_obj.**naga_sparse_obj_Double**

get_sparse()

1.4.2 naga_sparse_obj_Float

class naga_sparse_obj.**naga_sparse_obj_Float**

get_sparse()

1.5 naga_magma

1.5.1 getri_Double()

naga_magma.**getri_Double**()

1.5.2 getri_Float()

naga_magma.**getri_Float**()

1.5.3 `getri_host_Double()`

naga_magma.**getri_host_Double**()

1.5.4 `getri_host_Float()`

naga_magma.**getri_host_Float**()

1.5.5 `potri_Double()`

naga_magma.**potri_Double**()

1.5.6 `potri_Float()`

naga_magma.**potri_Float**()

1.5.7 `potri_host_Double()`

naga_magma.**potri_host_Double**()

1.5.8 `potri_host_Float()`

naga_magma.**potri_host_Float**()

1.5.9 `svd_Double()`

naga_magma.**svd_Double**()
Call carma_svd

1.5.10 `svd_Float()`

naga_magma.**svd_Float**()
Call carma_svd

1.5.11 `svd_host_Double()`

naga_magma.**svd_host_Double**()
Call carma_svd_cpu

naga_host_obj_Double2D mat: naga_host_obj_Double1D eigenvals: naga_host_obj_Double2D U: naga_host_obj_Double2D VT:

1.5.12 `svd_host_Float()`

naga_magma.**svd_host_Float**()
Call carma_svd_cpu

naga_host_obj_Float2D mat: naga_host_obj_Float1D eigenvals: naga_host_obj_Float2D U: naga_host_obj_Float2D VT:

1.5.13 syevd_Double()

naga_magma.**syevd_Double**()

1.5.14 syevd_Float()

naga_magma.**syevd_Float**()

1.5.15 syevd_host_Double()

naga_magma.**syevd_host_Double**()

1.5.16 syevd_host_Float()

naga_magma.**syevd_host_Float**()

1.6 naga_timer

1.6.1 naga_timer

class naga_timer.**naga_timer**

reset()

start()

stop()

1.6.2 threadSync()

naga_timer.**threadSync**()

1.7 shesha_ao

shesha_ao.basis.**command_on_Btt**(*rtc: Rtc.Rtc*, *dms: Dms.Dms*, *p_dms: list*, *p_geom: shesha_config.PGEOM.Param_geom*, *nfilt: int*)

Compute a command matrix in Btt modal basis (see error breakdown) and set it on the sutra_rtc. It computes by itself the volts to Btt matrix.

Parameters rtc: (Rtc) : rtc object

dms: (Dms): dms object

p_dms: (list of Param_dm): dms settings

p_geom: (Param_geom): geometry settings

nfilt: (int): number of modes to filter

shesha_ao.basis.**command_on_KL**(*rtc: Rtc.Rtc, dms: Dms.Dms, p_controller: shesha_config.PCONTROLLER.Param_controller, p_dms: typing.List[shesha_config.PDMS.Param_dm], p_geom: shesha_config.PGEOM.Param_geom, p_atmos: shesha_config.PATMOS.Param_atmos, p_tel: shesha_config.PTEL.Param_tel, nfilt: int*)

Compute a command matrix in KL modal basis and set it on the sutra_rtc. It computes by itself the volts to KL matrix.

Parameters rtc: (Rtc) : rtc object

dms: (Dms): dms object

p_dms: (list of Param_dm): dms settings

p_geom: (Param_geom): geometry settings

p_atmos : (Param_atmos) : atmos parameters

p_tel : (Param_tel) : telescope parameters

nfilt: (int): number of modes to filter

shesha_ao.basis.**compute_Btt**(*IFpzt*, *IFtt*)

Returns Btt to Volts and Volts to Btt matrices

Parameters IFpzt : (csr_matrix) : influence function matrix of pzt DM, sparse and arrange as (Npts in pup x nactus)

IFtt : (np.ndarray(ndim=2,dtype=np.float32)) : Influence function matrix of the TT mirror arrange as (Npts in pup x 2)

Returns

Btt : (np.ndarray(ndim=2,dtype=np.float32)) : Btt to Volts matrix

P : (np.ndarray(ndim=2,dtype=np.float32)) : Volts to Btt matrix

shesha_ao.basis.**compute_DMbasis**(*g_dm: Dms.Dms*, *p_dm: shesha_config.PDMS.Param_dm*, *p_geom: shesha_config.PGEOM.Param_geom*)

Compute a the DM basis as a sparse matrix :

- 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 = (csr_matrix) : DM IF basis

shesha_ao.basis.**compute_IFsparse**(*g_dm: Dms.Dms*, *p_dms: list*, *p_geom: shesha_config.PGEOM.Param_geom*)

Compute the influence functions of all DMs as a sparse matrix :

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

Parameters g_dm: (Dms) : Dms object

p_dms: (Param_dms) : dms settings

p_geom: (Param_geom) : geom settings

Returns IFbasis = (csr_matrix) : DM IF basis

`shesha_ao.basis.`**`compute_KL2V`**(*p_controller: shesha_config.PCONTROLLER.Param_controller, dms: Dms.Dms, p_dms: list, p_geom: shesha_config.PGEOM.Param_geom, p_atmos: shesha_config.PATMOS.Param_atmos, p_tel: shesha_config.PTEL.Param_tel*)

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

Parameters p_controller: (Param_controller) : p_controller settings

dms : (shesha_dms) : Dms object

p_dms: (list of Param_dm) : dms settings

p_geom : (Param_geom) : geometry parameters

p_atmos : (Param_atmos) : atmos parameters

p_tel : (Param_tel) : telescope parameters

Returns KL2V : (np.array(np.float32,dim=2)) : KL to Volt matrix

`shesha_ao.basis.`**`compute_cmat_with_Btt`**(*rtc: Rtc.Rtc, Btt: numpy.ndarray, nfilt: int*)

Compute a command matrix on the Btt basis and load it in the GPU

Parameters rtc: (Rtc): rtc object

Btt: (np.ndarray[ndim=2, dtype=np.float32]) : volts to Btt matrix

nfilt: (int): number of modes to filter

`shesha_ao.basis.`**`compute_cmat_with_KL`**(*rtc: Rtc.Rtc, KL2V: numpy.ndarray, nfilt: int*)

Compute a command matrix on the KL basis and load it in the GPU

Parameters rtc: (Rtc): rtc object

KL2V: (np.ndarray[ndim=2, dtype=np.float32]) : volts to KL matrix

nfilt: (int): number of modes to filter

`shesha_ao.cmats.`**`cmat_init`**(*ncontrol: int, rtc: Rtc.Rtc, p_controller: shesha_config.PCONTROLLER.Param_controller, p_wfss: typing.List[shesha_config.PWFS.Param_wfs], p_atmos: shesha_config.PATMOS.Param_atmos, p_tel: shesha_config.PTEL.Param_tel, p_dms: typing.List[shesha_config.PDMS.Param_dm], KL2V: numpy.ndarray = None, nmodes: int = 0*) → None

Compute the command matrix on the GPU

Parameters ncontrol: (int) :

rtc: (Rtc) :

p_controller: (Param_controller) : controller settings

p_wfss: (list of Param_wfs) : wfs settings

p_atmos: (Param_atmos) : atmos settings

p_tel : (Param_tel) : telescope settings

p_dms: (list of Param_dm) : dms settings

KL2V : (np.ndarray[ndim=2, dtype=np.float32]): (optional) KL to volts matrix (for KL cmat)

nmodes: (int) : (optional) number of kl modes shesha_ao.imats.**imat_geom**(*wfs: Sensors.Sensors, dms: Dms.Dms, p_wfss: typing.List[shesha_config.PWFS.Param_wfs], p_dms: typing.List[shesha_config.PDMS.Param_dm], p_controller: shesha_config.PCONTROLLER.Param_controller, meth: int = 0*) → numpy.ndarray

Compute the interaction matrix with a geometric method

Parameters wfs: (Sensors) : Sensors object

dms: (Dms) : Dms object

p_wfss: (list of Param_wfs) : wfs settings

p_dms: (list of Param_dm) : dms settings

p_controller: (Param_controller) : controller settings

meth: (int) : (optional) method type (0 or 1)

shesha_ao.imats.**imat_init**(*ncontrol: int*, *rtc: Rtc.Rtc*, *dms: Dms.Dms*, *p_dms: list*, *wfs: Sensors.Sensors*, *p_wfss: list*, *p_tel: shesha_config.PTEL.Param_tel*, *p_controller: shesha_config.PCONTROLLER.Param_controller*, *kl=None*, *dataBase: dict = {}*, *use_DB: bool = False*) → None

Initialize and compute the interaction matrix on the GPU

Parameters ncontrol: (int) : controller's index

rtc: (Rtc) : Rtc object

dms: (Dms) : Dms object

p_dms: (Param_dms) : dms settings

wfs: (Sensors) : Sensors object

p_wfss: (list of Param_wfs) : wfs settings

p_tel: (Param_tel) : telescope settings

p_controller: (Param_controller) : controller settings

kl:(np.array) : KL_matrix

dataBase:(dict): (optional) dict containing paths to files to load

use_DB:(bool) : (optional) use dataBase flag

shesha_ao.modopti.**openLoopSlp**(*tel: Telescope.Telescope*, *atmos: Atmos.Atmos*, *wfs: Sensors.Sensors*, *rtc: Rtc.Rtc*, *nrec: int*, *ncontrol: int*, *p_wfss: list*)

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

Parameters tel: (Telescope) : Telescope object

atmos: (Atmos) : Atmos object

wfs: (Sensors) : Sensors object

rtc: (Rtc) : Rtc object

nrec: (int) : number of samples to record

ncontrol: (int) : controller's index

p_wfss: (list of Param_wfs) : wfs settings

shesha_ao.tomo.**create_nact_geom**(*p_dm: shesha_config.PDMS.Param_dm*)

Compute the DM coupling matrix

Param p_dm : (Param_dm) : dm parameters

Returns Nact : (np.array(dtype=np.float64)) : the DM coupling matrix `shesha_ao.tomo.`**`create_piston_filter`**(*p_dm: shesha_config.PDMS.Param_dm*)

Create the piston filter matrix

Parameters p_dm: (Param_dm): dm settings

`shesha_ao.tomo.`**`do_tomo_matrices`**(*ncontrol: int, rtc: Rtc.Rtc, p_wfss: typing.List[shesha_config.PWFS.Param_wfs], dms: Dms.Dms, atmos: Atmos.Atmos, wfs: Sensors.Sensors, p_controller: shesha_config.PCONTROLLER.Param_controller, p_geom: shesha_config.PGEOM.Param_geom, p_dms: list, p_tel: shesha_config.PTEL.Param_tel, p_atmos: shesha_config.PATMOS.Param_atmos*)

Compute Cmm and Cphim matrices for the MV controller on GPU

Parameters ncontrol: (int): controller index

rtc: (Rtc) : rtc object

p_wfss: (list of Param_wfs) : wfs settings

dms: (Dms) : Dms object

atmos: (Atmos) : Atmos object

wfs: (Sensors) : Sensors object

p_controller: (Param_controller): controller settings

p_geom: (Param_geom) : geom settings

p_dms: (list of Param_dms) : dms settings

p_tel: (Param_tel) : telescope settings

p_atmos: (Param_atmos) : atmos settings

`shesha_ao.tomo.`**`selectDMforLayers`**(*p_atmos: shesha_config.PATMOS.Param_atmos*, *p_controller: shesha_config.PCONTROLLER.Param_controller*, *p_dms: list*)

For each atmos layer, select the DM which have to handle it in the Cphim computation for MV controller

Parameters p_atmos : (Param_atmos) : atmos parameters

p_controller : (Param_controller) : controller parameters

p_dms :(list of Param_dm) : dms parameters

Returns indlayersDM : (np.array(dtype=np.int32)) : for each atmos layer, the Dm number corresponding to it

Created on 1 aout 2017

@author: fferreira

`shesha_ao.wfs.`**`comp_new_fstop`**(*wfs: Sensors.Sensors*, *n: int*, *p_wfs: shesha_config.PWFS.Param_wfs*, *fssize: float*, *fstop: bytes*)

Compute a new field stop for pyrhr WFS

Parameters n : (int) : WFS index

wfs : (Param_wfs) : WFS parameters

fssize : (float) : field stop size [arcsec]

fstop : (string) : "square" or "round" (field stop shape)

`shesha_ao.wfs.`**`comp_new_pyr_ampl`**(*rtc: Rtc.Rtc*, *n: int*, *p_centroider: shesha_config.PCENTROIDER.Param_centroider*, *p_wfss: list*, *p_tel: shesha_config.PTEL.Param_tel*, *ampli: float*)

Set the pyramid modulation amplitude

Parameters rtc: (Rtc): rtc object

n : (int): centroider index

p_centroider : (Param_centroider) : pyr centroider settings

ampli : (float) : new amplitude in units of lambda/D

p_wfss : (list of Param_wfs) : list of wfs parameters

p_tel : (Param_tel) : Telescope parameters

`shesha_ao.wfs.`**`noise_cov`**(*nw: int*, *p_wfs: shesha_config.PWFS.Param_wfs*, *p_atmos: shesha_config.PATMOS.Param_atmos*, *p_tel: shesha_config.PTEL.Param_tel*)

Compute the diagonal of the noise covariance matrix for a SH WFS (arcsec^2) Photon noise: (pi^2/2)*(1/Nphotons)*(d/r0)^2 / (2*pi*d/lambda)^2 Electronic noise: (pi^2/3)*(wfs.noise^2/N^2photons)*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

1.8 shesha_config

Created on 13 juil. 2017

@author: vdeo

class `shesha_config.`**`Param_atmos`**

`set_L0`(*l*)

Set the L0 per layers

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

`set_alt`(*l*)

Set the altitudes of each layer

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

`set_deltax`(*l*)

Set the translation speed on axis x for each layer

Parameters **l** – (lit of float) : translation speed

`set_deltay`(*l*)

Set the translation speed on axis y for each layer

Parameters **l** – (lit of float) : translation speed

`set_dim_screens`(*l*)

Set the size of the phase screens

Parameters **l** – (lit of float) : phase screens sizes

`set_frac`(*l*)

Set the fraction of r0 for each layers

Parameters **l** – (lit of float) : fraction of r0

`set_nscreens`(*n*)

Set the number of turbulent layers

Parameters **n** – (long) number of screens.

set_pupixsize (*xsize*)
Set the pupil pixel size

Parameters **xsize** – (float) : pupil pixel size

set_r0 (*r*)
Set the global r0

Parameters **r** – (float) : global r0

set_seeds (*l*)
Set the seed for each layer

Parameters **l** – (lit of int) : seed

set_winddir (*l*)
Set the wind direction for each layer

Parameters **l** – (lit of float) : wind directions

set_windspeed (*l*)
Set the the wind speed for each layer

Parameters **l** – (list of float) : wind speeds

class shesha_config.**Param_centroider**

set_interpmat (*imap*)
Set the interp mat for corr centroider

Parameters **imap** – (np.ndarray[ndim=2, dtype=np.float32]) : sizey

set_method (*n*)

Set the method used by a pyr centroider: 0: nosinus global 1: sinus global 2: nosinus local 3: sinus local

Parameters **n** – (int) : method

set_nmax (*n*)
Set the nmax pixels used by a bpcog centroider

Parameters **n** – (int) : nmax

set_nwfs (*n*)
Set the index of the WFS handled by the centroider

Parameters **n** – (long) : WFS index

set_sizex (*n*)
Set the x size of inter mat for corr centroider

Parameters **n** – (int) : sizex

set_sizey (*n*)
Set the y size of interp mat for corr centroider

Parameters **n** – (int) : sizey

set_thresh (*t*)
Set the threshold used by a tcog centroider

Parameters **t** – (float) : thresh

set_type (*t*)
Set the centroider type

Parameters **t** – (string) : type

set_type_fct (*t*)
TODO: docstring

Parameters **t** – (string) : type

set_weights (*w*)
Set the weights used by a wcog cetroider

Parameters **w** – (np.ndarray[ndim=1, dtype=np.float32]) : weights

set_width (*t*)
Set the width of the gaussian used by a corr centroider

Parameters **t** – (float) : width

class shesha_config.**Param_controller**

set_TTcond (*m*)
Set the tiptilt condition number for cmat filtering with mv controller

Parameters **m** – (float) : tiptilt condition number

set_cmat (*cmat*)
Set the full control matrix

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

set_cured_ndivs (*n*)
Set the subdivision levels in cured

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

set_delay (*d*)
Set the loop delay expressed in frames

Parameters **d** – (float) :delay [frames]

set_gain (*g*)
Set the loop gain

Parameters **g** – (float) : loop gain

set_gmax (*g*)
Set the maximum gain for modal optimization

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

set_gmin (*g*)
Set the minimum gain for modal optimization

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

set_imat (*imat*)
Set the full interaction matrix

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

set_kl_imat (*n*)
Set type imat, for imat on kl set at 1

Parameters **k** – (int) : imat kl

set_klgain (*g*)
Set klgain for imatkl size = number of kl mode

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

set_maxcond (*m*)
Set the max condition number

Parameters **m** – (float) : max condition number

set_modopti(*n*)
Set the flag for modal optimization

Parameters **n** – (int) : flag for modal optimization

set_nactu(*l*)
Set the indices of dms

Parameters **l** – (np.ndarray[ndim=1, dtype=np.int32]) : indices of dms

set_ndm(*l*)
Set the indices of dms

Parameters **l** – (np.ndarray[ndim=1, dtype=np.int32]) : indices of dms

set_ngain(*n*)
Set the number of tested gains

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

set_nkl(*n*)
Set the number of KL modes used in imat_kl and used for computation of covmat in case of minimum variance controller

Parameters **n** – (long) : number of KL modes

set_nmodes(*n*)
Set the number of modes for M2V matrix (modal optimization)

Parameters **n** – (int) : number of modes

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

Parameters **n** – (int) : number of sample

set_nvalid(*l*)
Set the number of valid subaps

Parameters **l** – (list of int) : number of valid subaps

set_nwfs(*l*)
Set the indices of wfs

Parameters **l** – (np.ndarray[ndim=1, dtype=np.int32]) : indices of wfs

set_type(*t*)
Set the controller type

Parameters **t** – (string) : type

class shesha_config.**Param_dm**

set_alt(*a*)
set the conjugaison altitude

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

set_azbas(*r*)
Set the azimuthal array of the KL basis

Parameters **r** – (np.ndarray[ndim=1,dtype=np.float32_t]) : azimuthal array

set_center_name(*f*)
set the name of hdf5 influence file

Parameters **filename** – (str) : Hdf5 file influence name

set_coupling(*c*)
set the actuators coupling

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

set_cp (*r*)
Set the phi coordinates in carthesian grid

Parameters **r** – (np.ndarray[ndim=1,dtype=np.float32_t]) : phi coordinates in carthesian grid

set_cr (*r*)
Set the radial coordinates in carthesian grid

Parameters **r** – (np.ndarray[ndim=1,dtype=np.float32_t]) : radial coordinates in carthesian grid

set_cube_name (*cubename*)
set the name of influence cube in hdf5

Parameters **cubename** – (str) : name of influence cube

set_diam_dm (*di*)
set the name of dm diameter in file

Parameters **di** – (str) : name of diameter (meter) dm

set_diam_dm_proj (*dp*)
set the name of dm diameter projet on puille in file

Parameters **dp** – (str) : name of diameter (meter in pupil plan) dm

set_file_influ_hdf5 (*f*)
set the name of hdf5 influence file

Parameters **filename** – (str) : Hdf5 file influence name

set_gain (*g*)
Set the gain to apply to the actuators of the dm

Parameters **g** – (float) : gain

set_i1 (*i1*)
Set the X-position of the bottom left corner of each influence function

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

set_influ (*influ*)
Set the influence function

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

set_influType (*t*)
Set the influence function type for pzt DM

Parameters **t** – (str) : centroider type

set_influ_res (*res*)
set the name of influence fonction resolution in file

Parameters **res** – (str) : name of resoltion (meter/pixel) of influence

set_influpos (*ip*)
Set the influence functions pixels that contributes to each DM pixel

Parameters **ip** – (np.ndarray[ndim=1, drype=np.int32]) : influpos

set_influsize (*s*)
set the actuators influsize [pixels]

Parameters **s** – (int) : actuators influsize [pixels]

set_influstart (*n*)
Set the index where to start a new DM pixel shape in the array influpos to each DM pixel

Parameters `n` – (np.ndarray[ndim=1, drype=np.int32]) : influstart

set_j1 (*j1*)
Set the Y-position of the bottom left corner of each influence function

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

set_margin_in (*n*)
set the margin for inside actuator select (central obstruction)

Parameters `n` – (float) : unit is actuator pitch (+) for extra (-) for intra

set_margin_out (*n*)
set the margin for outside actuator select

Parameters `n` – (float) : unit is actuator pitch (+) for extra (-) for intra

set_n1 (*n*)
set the position of bottom left pixel in the largest support

Parameters `n` – (int) : actuators n1 [pixels]

set_n2 (*n*)
set the position of bottom right pixel in the largest support

Parameters `n` – (int) : actuators n2 [pixels]

set_nact (*n*)
set the number of actuator

Parameters `n` – (long) : number of actuators in the dm

set_ncp (*n*)
Set the dimension of grid (?)

Parameters `n` – (int) : dimension

set_ninflu (*n*)
Set the number of influence functions pixels that contributes to each DM pixel

Parameters `n` – (np.ndarray[ndim=1, drype=np.int32]) : ninflu

set_nkl (*n*)
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_npp (*n*)
Set the number of elements (?) for KL

Parameters `n` – (int) : number of elements

set_nr (*n*)
Set the number of radial points for KL

Parameters `n` – (int) : number of radial points

set_ntotact (*n*)
set the total number of actuators

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

set_ord (*n*)
Set the radial orders of the basis

Parameters `n` – (int) : radial order of the basis

set_outscl (*L0*)
Set the outer scale for KL with Von Karman spectrum

Parameters `L0` – (float) : outer scale [m]

set_pitch (*p*)
set the actuators pitch [pixels]

Parameters p – (float) : actuators pitch [pixels]

set_pupoffset (*off*)
Set the pupil offset in meters

Parameters off – (np.ndarray[ndim=1,dtype=np.float32_t]) : offsets [m]

set_puppixoffset (*off*)
Set the pupil offset in pixels

Parameters off – (np.ndarray[ndim=1,dtype=np.float32_t]) : offsets [pixels]

set_push4imat (*p*)
set the nominal voltage for imat

Parameters p – (float) : nominal voltage for imat

set_pzt_extent (*p*)
Set extent of pzt dm in pich unit default = 5

Parameters p – (int) : extent pzt dm

set_rabas (*r*)
Set the radial array of the KL basis

Parameters r – (np.ndarray[ndim=1,dtype=np.float32_t]) : radial array

set_thresh (*t*)
set the threshold on response for selection

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

set_type (*t*)
set the dm type

Parameters t – (str) : type of dm

set_type_kl (*t*)
Set the type of KL used for computation

Parameters t – (string) : KL types : kolmo or karman

set_type_pattern (*t*)
set the pattern type

Parameters t – (str) : type of pattern

set_unitpervolt (*u*)
set the Influence function sensitivity

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

set_x_name (*xname*)
set the name of x coord of influence fonction in file

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

set_xpos (*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 (*yname*)
set the name of y coord of influence fonction in file

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

set_ypos(*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

class shesha_config.**Param_geom**

set_apod(*a*)

Tells if the apodizer is used The apodizer is used if a is not 0

Parameters **a** – (int) boolean for apodizer

set_apod_file(*f*)
Set the path of apodizer file

Parameters **filename** – (str) : apodizer file name

set_apodizer(*s*)
Set the apodizer defined in spupil support

Parameters **s** – (np.ndarray[ndim=2, dtype=np.float32]) : apodizer

set_cent(*c*)
Set the central point of the simulation

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

set_ipupil(*s*)
Set the pupil in the biggest support

Parameters **s** – (np.ndarray[ndim=2, dtype=np.float32]) : pupil

set_is_init(*i*)
set the is_init flag

Parameters **i** – (bool) : is_init flag

set_mpupil(*s*)
Set the pupil in the middle support

Parameters **s** – (np.ndarray[ndim=2, dtype=np.float32]) : pupil

set_n(*s*)
Set the linear size of mpupil

Parameters **s** – (long) : coordinate (same in x and y) [pixel]

set_n1(*s*)
Set the bottom-left corner coordinates of the pupil in the ipupil support

Parameters **s** – (long) : coordinate (same in x and y) [pixel]

set_n2(*s*)
Set the upper-right corner coordinates of the pupil in the ipupil support

Parameters **s** – (long) : coordinate (same in x and y) [pixel]

set_p1(*s*)
Set the bottom-left corner coordinates of the pupil in the mpupil support

Parameters **s** – (long) : coordinate (same in x and y) [pixel]

set_p2(*s*)
Set the upper-right corner coordinates of the pupil in the mpupil support

Parameters **s** – (long) : coordinate (same in x and y) [pixel]

set_phase_ab_M1(*s*)
Set the phase aberration of the M1 defined in spupil support

Parameters **s** – (np.ndarray[ndim=2, dtype=np.float32]) : phase aberrations

set_phase_ab_M1_m(*s*)
Set the phase aberration of the M1 defined in mpupil support

Parameters **s** – (np.ndarray[ndim=2, dtype=np.float32]) : phase aberrations

set_pupdiam(*p*)
Set the linear size of total pupil

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

set_spupil(*s*)
Set the pupil in the smallest support

Parameters **s** – (np.ndarray[ndim=2, dtype=np.float32]) : pupil

set_ssize(*s*)
Set linear size of full image

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

set_zenithangle(*z*)
Set observations zenith angle

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

class shesha_config.**Param_loop**

set_devices(*devices*)
Set the list of GPU devices used

Parameters devices: (np.ndarray[ndim=1, dtype=np.int32_t]) : list of GPU devices

set_ittime(*t*)
Set iteration time

Parameters t: (float) :iteration time

set_niter(*n*)
Set the number of iteration

Parameters n: (long) : number of iteration

class shesha_config.**Param_target**

set_Lambda(*n*)
Set the wavelength of targets

Parameters **n** – (np.ndarray[ndim=2, dtype=np.float32]) : wavelength of targets

set_apod(*l*)
Set apodizer flag

Parameters **l** – (bool) : apod

set_dms_seen(*n*)
Set the dms_seen by the targets

Parameters **n** – (np.ndarray[ndim=2, dtype=np.int32]) : index of dms seen

set_mag(*n*)
Set the magnitudes of targets

Parameters **n** – (np.ndarray[ndim=2, dtype=np.float32]) : magnitudes

set_ntargets(*n*)
Set the number of targets

Parameters **n** – (long) number of targets

set_xpos(*n*)
Set the X-position of targets in the field [arcsec]

Parameters **n** – (np.ndarray[ndim=2, dtype=np.float32]) : X position of targets [arcsec]

set_ypos(*n*)
Set the Y-position of targets in the field [arcsec]

Parameters **n** – (np.ndarray[ndim=2, dtype=np.float32]): Y position of targets [arcsec]

set_zerop(*n*)
Set the zero point of targets

Parameters **n** – (float) : zero point of targets

class shesha_config.**Param_tel**

set_cobs(*c*)
Set the central obstruction ratio

Parameters **c** – (float) : central obstruction ratio

set_diam(*d*)
Set the telescope diameter

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

set_nbrmissing(*nb*)
Set the number of missing segments for EELT pupil

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

set_pupangle(*p*)
Set the rotation angle of pupil

Parameters **p** – (float) : rotation angle of pupil

set_referr(*ref*)
Set the std of reflectivity errors for EELT segments

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

set_spiders_type(*spider*)
Set the secondary supports type

Parameters **spider** – (str) : secondary supports type

set_std_piston(*piston*)
Set the std of piston errors for EELT segments

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

set_std_tt(*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(*spider*)
Set the secondary supports ratio

Parameters **spider** – (float) : secondary supports ratio

set_type_ap(*t*)
Set the EELT aperture type

Parameters **t** – (str) : EELT aperture type

set_vect_seg(*vect*)
Set the segment number for construct ELT pupil"

Parameters **vect** – (list of int32) : segment numbers

class shesha_config.**Param_wfs**(*error_budget=False*)

get_validsub()
Return both validsubsx and validsubsy

Returns (tuple) : (self._validsubsx, self._validsubsy)

set_G(*G*)
Set the magnifying factor

Parameters **G** – (float) : magnifying factor

set_Lambda(*L*)
Set the observation wavelength

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

set_Nfft(*n*)
Set the size of FFT support for a subap

Parameters **n** – (long) : size of FFT support

set_Ntot(*n*)
Set the size of hr image for a subap

Parameters **n** – (long) : size of hr image for a subap

set_altna(*a*)
Set the corresponding altitude

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

set_atmos_seen(*i*)
Tells if the wfs sees the atmosphere layers

Parameters **i** – (bool) :True if the WFS sees the atmosphere layers

set_azimuth(*data*)
TODO : docstring

set_beam(*data*)
TODO : docstring

set_beamsize(*b*)
Set the laser beam fwhm on-sky

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

set_binmap(*data*)
TODO : docstring

set_dms_seen(*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_dx(*dx*)
Set the X axis misalignment

Parameters **dx** – (float) : dx (pix)

set_dy(*dy*)
Set the Y axis misalignment

Parameters **dy** – (float) : dy (pix)

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

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

set_fluxPerSub (*data*)
Set the subap diameter (m)

Parameters **data** – (np.array(ndim=2, dtype=np.float32)) : subap diameter (m)

set_fracsub (*f*)
Set the minimal illumination fraction for valid subaps

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

set_fssize (*f*)
Set the size of field stop

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

set_fstop (*f*)
Set the size of field stop

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

set_ftbeam (*data*)
TODO : docstring

set_ftkernel (*data*)
TODO : docstring

set_gsalt (*g*)
Set the altitude of guide star

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

set_gsmag (*g*)
Set the magnitude of guide star

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

set_halfxy (*data*)
TODO : docstring

set_hrmap (*data*)
TODO : docstring

set_istart (*data*)
TODO : docstring

set_isvalid (*data*)
TODO : docstring

set_jstart (*data*)
TODO : docstring

set_kernel (*k*)
Set the attribute kernel

Parameters **k** – (float) :

set_laserpower (*l*)
Set the laser power

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

set_lgskern (*data*)
TODO : docstring

set_lgsreturnperwatt (*lpw*)
Set the return per watt factor

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

set_lltx(*l*)
Set the x position of llt

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

set_llty(*l*)
Set the y position of llt

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

set_noise(*n*)
Set the desired noise

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

set_nphotons(*n*)
Set number of photons per subap

Parameters **n** – (float) : number of photons per subap

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

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

set_npix(*n*)
Set the number of pixels per subap

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

set_nrebin(*n*)
Set the rebin factor from hr to binned image for a subap

Parameters **n** – (long) : rebin factor

set_nvalid(*n*)
Set the number of valid subapertures

Parameters **n** – (long) : number of valid subapertures

set_nxsub(*n*)
Set the linear number of subaps

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

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

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

set_optthroughput(*o*)
Set the wfs global throughput

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

set_pdiam(*n*)
Set the subap diameter in pixels

Parameters **n** – (long) : subap diam in pixels

set_phasemap(*data*)
TODO : docstring

set_pixsize(*p*)
Set the pixel size

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

set_prof1d(*data*)
TODO : docstring

set_profcum(*data*)
TODO : docstring

set_profna(*p*)
Set the sodium profile

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

set_proftype(*p*)
Set the type of sodium profile

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

set_pyr_ampl(*p*)
Set the pyramid wfs modulation amplitude radius

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

set_pyr_cx(*cx*)
Set the x position of modulation points for pyramid sensor

Parameters cx – (np.ndarray[ndim=1,dtype=np.floatt32_t) : x positions

set_pyr_cy(*cy*)
Set the y position of modulation points for pyramid sensor

Parameters cy – (np.ndarray[ndim=1,dtype=np.floatt32_t) : y positions

set_pyr_loc(*p*)
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(*p*)
Set the total number of point along modulation circle

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

set_pyr_pos(*data*)
TODO : docstring

set_pyr_pup_sep(*pyr_pup_sep*)
Set the pyramid pupil separation. (default: long(wfs.nxsub))

Parameters pyr_pup_sep – (long) : pyramid pupil separation wanted

set_pyrtype(*p*)
Set the type of pyramid,

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

set_qpixsize(*n*)
Set the quantum pixel size for the simulation

Parameters n – (float) : quantum pixel size

set_sincar(*data*)
TODO : docstring

set_subapd(*n*)
Set the subap diameter (m)

Parameters n – (float) : subap diameter (m)

set_submask(*data*)
TODO : docstring

set_thetaML(*thetaML*)
Set the rotation angle in the pupil

Parameters thetaML – (float) : rotation angle (rad)

set_type (*type_wfs*)
Set the type of wfs

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

set_validsubsx (*vx*)
Set the valid subapertures along X-axis

Parameters **vx** – (np.array(dim=1, dtype=np.int32)) : validsubsx

set_validsubsy (*vy*)
Set the valid subapertures along Y-axis

Parameters **vy** – (np.array(dim=1, dtype=np.int32)) : validsubsy

set_xpos (*x*)
Set the guide star x position on sky

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

set_ypos (*y*)
Set the guide star y position on sky

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

set_zerop (*z*)
Set the detector zero point

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

1.9 shesha_constants

Created on 5 juil. 2017

@author: vdeo

Numerical constants for shesha Config enumerations for safe-typing

class shesha_constants.**ApertureType**
Telescope apertures

class shesha_constants.**CentroiderType**
Centroider types

class shesha_constants.**ControllerType**
Controller types

class shesha_constants.**DmType**
Types of deformable mirrors

class shesha_constants.**FieldStopType**
WFS field stop

class shesha_constants.**InfluType**
Influence function types

class shesha_constants.**KLType**
Possible KLs for computations

class shesha_constants.**PatternType**
Types of Piezo DM patterns

class shesha_constants.**ProfType**
Sodium profile for LGS

class shesha_constants.**PyrCentroiderMethod**
Pyramid centroider methods Local flux normalization (eq SH quad-cell, ray optics. Ragazzonni

1996) Global flux normalization (Verinaud 2004, most > 2010 Pyr applications) Resulting (A+/-B-/+C-D)/(A+B+C+D) or sin((A+/-B-/+C-D)/(A+B+C+D))

class shesha_constants.**SpiderType**
Spiders

class shesha_constants.**TargetImageType**
Target Images

class shesha_constants.**WFSType**
WFS Types

shesha_constants.**check_enum**(*cls*, *name*)
Create a safe-type enum instance from bytes contents

1.10 shesha_init

Created on 13 juil. 2017

@author: vdeo

shesha_init.**atmos_init**(*context: naga_context.naga_context*, *p_atmos: shesha_config.PATMOS.Param_atmos*, *p_tel: shesha_config.PTEL.Param_tel*, *p_geom: shesha_config.PGEOM.Param_geom*, *ittime=None*, *p_wfss=None*, *sensors=None*, *p_target=None*, *dataBase={}*, *use_DB=False*)
TODO: docstring

shesha_init.**target_init**(*ctxt: naga_context.naga_context*, *telescope: Telescope.Telescope*, *p_target: shesha_config.PTARGET.Param_target*, *p_atmos: shesha_config.PATMOS.Param_atmos*, *p_tel: shesha_config.PTEL.Param_tel*, *p_geom: shesha_config.PGEOM.Param_geom*, *dm=None*, *brama=False*)
Create a cython target from parametres structures

Parameters ctxt: (naga_context) : telescope: (Telescope): Telescope object target: (Param_target) : target_settings p_atmos: (Param_atmos) : atmos settings p_tel: (Param_tel) : telescope settings p_geom: (Param_geom) : geom settings dm: (Param_dm) : (optional) dm settings brama: (bool): (optional) BRAMA flag

shesha_init.**dm_init**(*context: naga_context.naga_context, p_dms: typing.List[shesha_config.PDMS.Param_dm], p_tel: shesha_config.PTEL.Param_tel, p_geom: shesha_config.PGEOM.Param_geom, p_wfss: typing.List[shesha_config.PWFS.Param_wfs] = None*) → Dms.Dms
Create and initialize a Dms object on the gpu

Parameters context: (naga_context): context p_dms: (list of Param_dms) : dms settings p_tel: (Param_tel) : telescope settings p_geom: (Param_geom) : geom settings p_wfss: (list of Param_wfs) : wfs settings

shesha_init.**wfs_init**(*context: naga_context.naga_context*, *telescope: Telescope.Telescope*, *p_wfss: list*, *p_tel: shesha_config.PTEL.Param_tel*, *p_geom: shesha_config.PGEOM.Param_geom*, *p_dms=None*, *p_atmos=None*)
Create and initialise a Sensors object

Parameters context : (naga_context) telescope: (Telescope) : Telescope object p_wfss: (list of Param_wfs) : wfs settings p_tel: (Param_tel) : telescope settings p_geom: (Param_geom) : geom settings p_dms : (list of Param_dm) : (optional) dms settings p_atmos: (Param_atmos) : (optional) atmos settings shesha_init.**rtc_init**(*context: naga_context.naga_context, tel: Telescope.Telescope, wfs: Sensors.Sensors, dms: Dms.Dms, atmos: Atmos.Atmos, p_wfss: list, p_tel: shesha_config.PTEL.Param_tel, p_geom: shesha_config.PGEOM.Param_geom, p_atmos: shesha_config.PATMOS.Param_atmos, ittime: float, p_centroiders=None, p_controllers=None, p_dms=None, do_refslp=False, brama=False, tar=None, dataBase={}, use_DB=False*)

Initialize all the sutra_rtc objects : centroiders and controllers

Parameters context: (naga_context): context tel: (Telescope) : Telescope object wfs: (Sensors) : Sensors object dms: (Dms) : Dms object atmos: (Atmos) : Atmos object p_wfss: (list of Param_wfs) : wfs settings p_tel: (Param_tel) : telescope settings p_geom: (Param_geom) : geom settings p_atmos: (Param_atmos) : atmos settings ittime: (float) : iteration time [s] p_centroiders : (list of Param_centroider): (optional) centroiders settings p_controllers : (list of Param_controller): (optional) controllers settings p_dms: (list of Param_dms) : (optional) dms settings do_refslp : (bool): (optional) do ref slopes flag, default=False brama: (bool) : (optional) BRAMA flag tar: (Target) : (optional) dataBase: (dict): (optional) dict containig paths to files to load use_DB: (bool): use dataBase flag

Returns Rtc : (Rtc) : Rtc object

1.11 shesha_sim

class shesha_sim.**Bench**(**args, **kwargs*) → _O

Class Bench

Timed version of the simulator class using decorated overloads

class shesha_sim.**BenchBrama**(*filepath: str = None, use_DB: bool = False*) → None

Class BenchBrama

next(**, see_atmos: bool = False, nControl: int = 0*) → None

function next Iterates on centroiding and control, with optional parameters

Parameters **(int)** (*nControl*) – Controller number to use, default 0 (single control configurations)

class shesha_sim.**Simulator**(*filepath: str = None, use_DB: bool = False*) → None

init_sim() → None

TODO: docstring

load_from_file(*filepath: str*) → None

TODO: docstring

loop(*n=1, monitoring_freq=100, **kwargs*)

TODO: docstring

next(**, move_atmos: bool = True, see_atmos: bool = True, nControl: int = 0, tar_trace: typing.Iterable[int] = None, wfs_trace: typing.Iterable[int] = None, apply_control: bool = True*) → None

function next Iterates the AO loop, with optional parameters

Parameters

- **(bool)** (*apply_control*) – move the atmosphere for this iteration, default: True
- **(int)** (*nControl*) – Controller number to use, default 0 (single control configurations)

- **(None or list[int])** (*wfs_trace*) – list of targets to trace. None equivalent to all.
- **(None or list[int])** – list of WFS to trace. None equivalent to all.
- **(bool)** – (optional) if True (default), apply control on DMs

class shesha_sim.**SimulatorBrama** (*filepath: str = None*, *use_DB: bool = False*) → None
Class SimulatorBrama: Brama overloaded simulator _tar_init and _rtc_init to instantiate Brama classes instead of regular classes next() to call rtc/tar.publish()

1.12 shesha_util

Created on 13 juil. 2017

@author: vdeo Created on 3 aout 2017

@author: fferreira

shesha_util.dm_util.**createDoubleHexaPattern** (*pitch: float*, *supportSize: int*)
Creates a list of M actuator positions spread over an hexagonal grid. The number M is the number of points of this grid, it cannot be known before the procedure is called. Coordinates are centred around (0,0). The support that limits the grid is a square [-n/2,n/2].

Parameters pitch: (float) : distance in pixels between 2 adjacent actus

n: (float) [size in pixels of the support over which the coordinate list] should be returned.

Returns xy: (np.ndarray(dims=2,dtype=np.float32)) : xy[M,2] list of coodinates

shesha_util.dm_util.**createHexaPattern** (*pitch: float*, *supportSize: int*)
Creates a list of M actuator positions spread over an hexagonal grid. The number M is the number of points of this grid, it cannot be known before the procedure is called. Coordinates are centred around (0,0). The support that limits the grid is a square [-n/2,n/2].

Parameters pitch: (float) : distance in pixels between 2 adjacent actus

n: (float) [size in pixels of the support over which the coordinate list] should be returned.

Returns xy: (np.ndarray(dims=2,dtype=np.float32)) : xy[M,2] list of coodinates

shesha_util.dm_util.**createSquarePattern** (*pitch: float*, *nxact: int*)
Creates a list of M=nxact^2 actuator positions spread over an square grid. Coordinates are centred around (0,0).

Parameters pitch: (float) : distance in pixels between 2 adjacent actus

nxact: (int) : number of actu across the pupil diameter

Returns xy: (np.ndarray(dims=2,dtype=np.float32)) : xy[M,2] list of coodinates

shesha_util.dm_util.**dim_dm_patch** (*pupdiam: int, diam: float, type_dm: bytes, alt: float, xpos_wfs: typing.List[float], ypos_wfs: typing.List[float]*)
compute patchDiam for DM

Parameters pupdiam: (int) : pupil diameter

diam: (float) : telescope diameter

type_dm: (bytes) : type of dm

alt: (float) : altitude of dm

xpos_wfs: (list) : list of wfs xpos

ypos_wfs: (list) : list of wfs ypos

shesha_util.dm_util.**dim_dm_support**(*cent: float*, *extent: int*, *ssize: int*)
Compute the DM support dimensions

Parameters cent : (float): center of the pupil

extent: (float): size of the DM support

ssize: (int): size of ipupil support

shesha_util.dm_util.**make_zernike**(*nzer: int*, *size: int*, *diameter: int*, *xc=-1.0*, *yc=-1.0*, *ext=0*)
Compute the zernike modes

Parameters nzer: (int) : number of modes

size: (int) : size of the screen

diameter: (int) : pupil diameter

xc: (float) : (optional) x-position of the center

yc: (float) : (optional) y-position of the center

ext: (int) : (optional) extension

Returns z : (np.ndarray(ndims=3,dtype=np.float64)) : zernikes modes

shesha_util.dm_util.**select_actuators**(*xc: numpy.ndarray*, *yc: numpy.ndarray*, *nxact: int*, *pitch: int*, *cobs: float*, *margin_in: float*, *margin_out: float*, *N=None*)
Select the "valid" actuators according to the system geometry

Parameters xc: actuators x positions (origine in center of mirror)

yc: actuators y positions (origine in center of mirror)

nxact:

pitch:

cobs:

margin_in:

margin_out:

N:

Returns liste_fin: actuator indice selection for xpos/ypos

shesha_util.dm_util.**zernumero**(*zn: int*)
Returns the radial degree and the azimuthal number of zernike number zn, according to Noll numbering (Noll, JOSA, 1976)

Parameters zn: (int) : zernike number

Returns

rd: (int) : radial degrees

an: (int) : azimuthal numbers

shesha_util.hdf5_utils.**checkControlParams**(*savepath*, *config*, *pdict*, *matricesToLoad*)
Compare the current controller parameters to the database. If similar parameters are found, matricesToLoad dictionary is completed. Since all the controller matrices are computed together, we only check the parameters for the imat matrix : if we load imat, we load eigenv and U too.

Parameters config : (module) : simulation parameters

matricesToLoad : (dictionary) : matrices that will be load and their path `shesha_util.hdf5_utils.`**`checkDmsParams`**(*savepath*, *config*, *pdict*, *matricesToLoad*)

Compare the current controller parameters to the database. If similar parameters are found, matricesToLoad dictionary is completed. Since all the dms matrices are computed together, we only check the parameters for the pztok matrix : if we load pztok, we load pztnok too.

Parameters config : (module) : simulation parameters

matricesToLoad : (dictionary) : matrices that will be load and their path

`shesha_util.hdf5_utils.`**`checkMatricesDataBase`**(*savepath*, *config*, *param_dict*)

Check in the database if the current config have been already run. If so, return a dictionary containing the matrices to load and their path. Matrices which don't appear in the dictionary will be computed, stored and added to the database during the simulation. If the database doesn't exist, this function creates it.

Parameters savepath : (str) : path to the data repertory

config : (module) : simulation parameters

param_dict : (dictionary) : parameters dictionary

Returns matricesToLoad : (dictionary) : matrices that will be load and their path

`shesha_util.hdf5_utils.`**`checkTurbuParams`**(*savepath*, *config*, *pdict*, *matricesToLoad*)

Compare the current turbulence parameters to the database. If similar parameters are found, the matricesToLoad dictionary is completed. Since all the turbulence matrices are computed together, we only check the parameters for the A matrix : if we load A, we load B, istx and isty too.

Parameters config : (module) : simulation parameters

matricesToLoad : (dictionary) : matrices that will be load and their path

`shesha_util.hdf5_utils.`**`configFromH5`**(*filename*, *config*)

TODO: docstring

`shesha_util.hdf5_utils.`**`create_file_attributes`**(*filename*, *config*)

Create an hdf5 file wtih attributes corresponding to all simulation parameters

Param filename : (str) : full path + filename to create

config : () : simulation parameters

`shesha_util.hdf5_utils.`**`initDataBase`**(*savepath*, *param_dict*)

Initialize and create the database for all the saved matrices. This database will be placed on the top of the savepath and be named matricesDataBase.h5.

Parameters savepath : (str) : path to the data repertory

param_dict : (dictionary) : parameters dictionary

`shesha_util.hdf5_utils.`**`init_hdf5_files`**(*savepath*, *param_dict*, *matricesToLoad*)

TODO: docstring

`shesha_util.hdf5_utils.`**`load_AB_from_dataBase`**(*database*, *ind*)

Read and return A, B, istx and isty from the database

Parameters database: (dict): dictionary containing paths to matrices to load

ind: (int): layer index

`shesha_util.hdf5_utils.`**`load_dm_geom_from_dataBase`**(*database*, *ndm*)

Read and return the DM geometry

Parameters database: (dict): dictionary containing paths to matrices to load

ndm: (int): dm index

`shesha_util.hdf5_utils.`**`load_imat_from_dataBase`**(*database*)

Read and return the imat

Parameters database: (dict): dictionary containing paths to matrices to load `shesha_util.hdf5_utils.`**`params_dictionary`**`(`*config*`)`

Create and returns a dictionary of all the config parameters with the corresponding keys for further creation of database and save files

Parameters **`config`** – (module) : simulation parameters

Return param_dict (dictionary) : dictionary of parameters

`shesha_util.hdf5_utils.`**`readHdf5SingleDataset`**`(`*filename*, *datasetName='dataset'*`)`

Read a single dataset from an hdf5 file

Parameters filename: (str) : name of the file to read from

datasetName: (str) : name of the dataset to read (default="dataset")

`shesha_util.hdf5_utils.`**`save_AB_in_database`**`(`*k*, *A*, *B*, *istx*, *isty*`)`

Save A, B, istx and isty in the database

Parameters ind:

A:

B:

istx:

isty:

`shesha_util.hdf5_utils.`**`save_dm_geom_in_dataBase`**`(`*ndm*, *influpos*, *ninflu*, *influstart*, *i1*, *j1*, *ok*`)`

Save the DM geometry in the database

Parameters ndm:

influpos:

ninflu:

influstart:

i1:

j1:

`shesha_util.hdf5_utils.`**`save_h5`**`(`*filename*, *dataname*, *config*, *data*`)`

save_hdf5(filename, dataname, config, data) Create a hdf5 file and store data in it with full header from config parameters Usefull to backtrace data origins

Param filename: (str) : full path to the file

dataname : (str) : name of the data (imat, cmat…)

config : (module) : config parameters

data : np.array : data to save

`shesha_util.hdf5_utils.`**`save_hdf5`**`(`*filename*, *dataname*, *data*`)`

Create a dataset in an existing hdf5 file filename and store data in it

Param filename: (str) : full path to the file

dataname : (str) : name of the data (imat, cmat…)

data : np.array : data to save

`shesha_util.hdf5_utils.`**`save_imat_in_dataBase`**`(`*imat*`)`

Save the DM geometry in the database

Parameters imat: (np.ndarray): imat to save

`shesha_util.hdf5_utils.`**`updateDataBase`**`(`*h5file*, *savepath*, *matrix_type*`)`

Update the database adding a new row to the matrix_type database.

Parameters h5file : (str) : path to the new h5 file to add

savepath : (str) : path to the data directory

matrix_type [(str)][type of matrix to store ("A","B","istx","isty"] "istx","eigenv","imat","U" "pztok" or "pztnok")

`shesha_util.hdf5_utils.`**`validDataBase`**(*savepath*, *matricesToLoad*)
TODO: docstring

`shesha_util.hdf5_utils.`**`validFile`**(*filename*)
TODO: docstring

`shesha_util.hdf5_utils.`**`validInStore`**(*store*, *savepath*, *matricetype*)
TODO: docstring

`shesha_util.hdf5_utils.`**`writeHdf5SingleDataset`**(*filename*, *data*, *datasetName='dataset'*)
Write a hdf5 file containig a single field

If the file already exists, it will be overwritten

Parametres filename: (str) : name of the file to write

data: (np.ndarray) : content of the file

datasetName: (str) : name of the dataset to write (default="dataset")

Created on 3 aout 2017

@author: fferreira

`shesha_util.influ_util.`**`besel_orth`**(*m*, *n*, *phi*, *r*)
TODO: docstring

Parameters m:

n:

phi:

r:

Returns B:

`shesha_util.influ_util.`**`bessel_influence`**(*xx*, *yy*, *type_i=b'square'*)
TODO: docstring

Parameters xx:

yy:

type_i: (optional)

Returns influ

`shesha_util.influ_util.`**`makeBessel`**(*pitch: float*, *coupling: float*, *x: numpy.ndarray = None*, *y: numpy.ndarray = None*, *patternType: bytes = b'square'*)
Compute Bessel influence function

Parameters pitch: (float) : pitch of the DM expressed in pixels

coupling: (float) : coupling of the actuators

x: indices of influence function in relative position x local coordinates (float). 0 = top of the influence function

y: indices of influence function in relative position y local coordinates (float). 0 = top of the influence function

Returns influ: (np.ndarray(dims=3,dtype=np.float64)) : cube of the IF for each actuator `shesha_util.influ_util.makeBlacknutt(pitch: float, coupling: float, x=None, y=None)`

Compute Blacknutt influence function Attention, ici on ne peut pas choisir la valeur de coupling. La variable a ete laissee dans le code juste pour compatibilité avec les autres fonctions, mais elle n'est pas utilisee.

Parameters pitch: (float): pitch of the DM expressed in pixels

coupling: (float) : coupling of the actuators

x: indices of influence function in relative position x local coordinates (float). 0 = top of the influence function

y: indices of influence function in relative position y local coordinates (float). 0 = top of the influence function

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

`shesha_util.influ_util.makeGaussian(pitch: float, coupling: float, x=None, y=None)`

Compute Gaussian influence function. Coupling parameter is not taken into account

Parameters pitch: (float) : pitch of the DM expressed in pixels

coupling: (float) : coupling of the actuators

x: indices of influence function in relative position x local coordinates (float). 0 = top of the influence function

y: indices of influence function in relative position y local coordinates (float). 0 = top of the influence function

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

`shesha_util.influ_util.makeRadialSchwartz(pitch: float, coupling: float, x=None, y=None)`

Compute radial Schwartz influence function

Parameters pitch: (float) : pitch of the DM expressed in pixels

coupling: (float) : coupling of the actuators

x: indices of influence function in relative position x local coordinates (float). 0 = top of the influence function

y: indices of influence function in relative position y local coordinates (float). 0 = top of the influence function

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

`shesha_util.influ_util.makeRigaut(pitch: float, coupling: float, x=None, y=None)`

Compute 'Rigaut-like' influence function

Parameters pitch: (float) : pitch of the DM expressed in pixels

coupling: (float) : coupling of the actuators

x: indices of influence function in relative position x local coordinates (float). 0 = top of the influence function

y: indices of influence function in relative position y local coordinates (float). 0 = top of the influence function

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

`shesha_util.influ_util.makeSquareSchwartz(pitch: float, coupling: float, x=None, y=None)`

Compute Square Schwartz influence function

Parameters pitch: (float) : pitch of the DM expressed in pixels

coupling: (float) : coupling of the actuators

x: indices of influence function in relative position x local coordinates (float). 0 = top of the influence function y: indices of influence function in relative position y local coordinates (float). 0 = top of the influence function

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

`shesha_util.iterkolmo.`**`AB`**(*n*, *L0*, *deltax*, *deltay*, *rank=0*)

DOCUMENT AB, n, A, B, istencil This function initializes some matrices A, B and a list of stencil indexes istencil for iterative extrusion of a phase screen.

The method used is described by Fried & Clark in JOSA A, vol 25, no 2, p463, Feb 2008. The iteration is : x = A(z-zRef) + B.noise + zRef with z a vector containing "old" phase values from the initial screen, that are listed thanks to the indexes in istencil.

SEE ALSO: extrude createStencil Cxx Cxz Czz

`shesha_util.iterkolmo.`**`Cxx`**(*n*, *Zxn*, *Zyn*, *Xx*, *Xy*, *L0*)

Cxx computes the covariance matrix of the new phase vector x (new column for the phase screen).

`shesha_util.iterkolmo.`**`Cxz`**(*n*, *Zx*, *Zy*, *Xx*, *Xy*, *istencil*, *L0*)

Cxz computes the covariance matrix between the new phase vector x (new column for the phase screen), and the already known phase values z.

The known values z are the values of the phase screen that are pointed by the stencil indexes (istencil)

`shesha_util.iterkolmo.`**`Czz`**(*n*, *Zx*, *Zy*, *ist*, *L0*)

Czz computes the covariance matrix of the already known phase values z.

The known values z are the values of the phase screen that are pointed by the stencil indexes (istencil)

`shesha_util.iterkolmo.`**`asymp_macdo`**(*x*)

Computes a term involved in the computation of the phase struct function with a finite outer scale according to the Von-Karman model. The term involves the MacDonald function (modified bessel function of second kind) K_{5/6}(x), and the algorithm uses the asymptotic form for x ~ infinity.

Warnings :

- This function makes a floating point interrupt for x=0 and should not be used in this case.
- Works only for x>0.

`shesha_util.iterkolmo.`**`create_screen`**(*r0*, *pupixsize*, *screen_size*, *L0*, *A*, *B*, *ist*)

DOCUMENT create_screen screen = create_screen(r0,pupixsize,screen_size,&A,&B,&ist)

creates a phase screen and fill it with turbulence r0 : total r0 @ 0.5m pupixsize : pupil pixel size (in meters) screen_size : screen size (in pixels) A : A array for future extrude B : B array for future extrude ist : istencil array for future extrude

`shesha_util.iterkolmo.`**`create_screen_assist`**(*screen_size*, *L0*, *r0*)

screen_size : screen size (in pixels) L0 : L0 in pixel r0 : total r0 @ 0.5 microns

`shesha_util.iterkolmo.`**`create_stencil`**(*n*)

TODO: docstring

`shesha_util.iterkolmo.`**`extrude`**(*p*, *r0*, *A*, *B*, *istencil*)

DOCUMENT p1 = extrude(p,r0,A,B,istencil)

Extrudes a phase screen p1 from initial phase screen p. p1 prolongates p by 1 column on the right end. r0 is expressed in pixels

The method used is described by Fried & Clark in JOSA A, vol 25, no 2, p463, Feb 2008. The iteration is : x = A(z-zRef) + B.noise + zRef with z a vector containing "old" phase values from the initial screen, that are listed thanks to the indexes in istencil.

Examples n = 32; AB, n, A, B, istencil; p = array(0.0,n,n); p1 = extrude(p,r0,A,B,istencil); pli, p1

SEE ALSO: AB() createStencil() Cxx() Cxz() Czz()

`shesha_util.iterkolmo.`**`macdo_x56`**(*x*, *k=10*)

Computation of the function f(x) = x^(5/6)*K_{5/6}(x) using a series for the esimation of K_{5/6}, taken from Rod Conan thesis : K_a(x)=1/2 sum_{n=0}^infty

rac{(-1)^n}{n!} left(Gamma(-n-a) (x/2)^{2n+a} + Gamma(-n+a) (x/2)^{2n-a}

ight) , with a = 5/6.

Setting x22 = (x/2)^2, setting uda = (1/2)^a, and multiplying by x^a, this becomes : x^a * Ka(x) = 0.5 $ -1^n / n! [G(-n-a).uda x22^(n+a) + G(-n+a)/uda x22^n] Then we use the following recurrence formulae on the following quantities : G(-(n+1)-a) = G(-n-a) / -a-n-1 G(-(n+1)+a) = G(-n+a) / a-n-1 (n+1)! = n! * (n+1) x22^(n+1) = x22^n * x22 and at each iteration on n, one will use the values already computed at step (n-1). The values of G(a) and G(-a) are hardcoded instead of being computed.

The first term of the series has also been skipped, as it vanishes with another term in the expression of Dphi.

`shesha_util.iterkolmo.`**`phase_struct`**(*r*, *L0=None*)
TODO: docstring

`shesha_util.iterkolmo.`**`rodconan`**(*r*, *L0*)
The phase structure function is computed from the expression Dphi(r) = k1 * L0^(5./3) * (k2 - (2.pi.r/L0)^5/6 K_{5/6}(2.pi.r/L0))

For small r, the expression is computed from a development of K_5/6 near 0. The value of k2 is not used, as this same value appears in the series and cancels with k2. For large r, the expression is taken from an asymptotic form.

`shesha_util.iterkolmo.`**`stencil_size`**(*n*)
TODO: docstring

`shesha_util.iterkolmo.`**`stencil_size_array`**(*size*)
Compute_size2(np.ndarray[ndim=1, dtype=np.int64_t] size)

Compute the size of a stencil, given the screen size

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

Created on Thu Sep 8 15:42:43 2016 Functions for DM Python kl @author: translated by sdurand Compass Yorick translation

`shesha_util.kl_util.`**`gkl_fcom`**(*kers: numpy.ndarray*, *cobs: float*, *nf: int*)
This routine does the work : finding the eigenvalues and corresponding eigenvectors. Sort them and select the right one. It returns the KL modes : in polar coordinates : rabas as well as the associated variance : evals. It also returns a bunch of indices used to recover the modes in cartesian coordinates (nord, npo and ordd).

Parameters kerns : (np.ndarray[ndim= ,dtype=np.float32]) :

cobs : (float) : central obstruction

nf : (int) :

`shesha_util.kl_util.`**`make_azimuth`**(*nord: int*, *npp: int*) → numpy.ndarray
TODO: docstring

Parameters nord:

npp:

Returns azbas:

`shesha_util.kl_util.`**`make_kernels`**(*cobs: float*, *nr: int*, *radp: numpy.ndarray*, *kl_type: bytes*, *outscl: float = None*) → numpy.ndarray
This routine generates the kernel used to find the KL modes. The kernel constructed here should be simply a discretization of the continuous kernel. It needs rescaling before it is treated as a matrix for finding the eigen-values. The outer scale should be in units of the diameter of the telescope.

TODO:

Parameters cobs : (float): central obstruction

nr : (int) :

radp : (float) :

kl_type : (bytes) : “kolmo” or “karman”

outscl : (float) : outter scale for Von Karman spectrum

Returns kers :

`shesha_util.kl_util.`**`make_radii`**(*cobs: float*, *nr: int*) → float

TODO: docstring

Parameters cobs: (float) : central obstruction

nr : (int) :

`shesha_util.kl_util.`**`pcgeom`**(*nr*, *npp*, *cobs*, *ncp*, *ncmar*)

This routine builds a geom_struct. px and py are the x and y coordinates of points in the polar arrays. cr and cp are the r and phi coordinates of points in the cartesian grids. ncmar allows the possibility that there is a margin of ncmar points in the cartesian arrays outside the region of interest

TODO:

parameters nr:

npp:

cobs: (float) : central obstruction

ncp:

ncmar:

returns ncp:

ncmar:

px:

py:

cr:

cp:

pincx:

pincy:

pincw:

ap:

`shesha_util.kl_util.`**`piston_orth`**(*nr: int*) → numpy.ndarray

TODO: docstring

Parameters nr:

Returns s:

`shesha_util.kl_util.`**`polang`**(*r: numpy.ndarray*) → numpy.ndarray

This routine generates an array with the same dimensions as r, but containing the azimuthal values for a polar coordinate system.

TODO:

parameters r:

return p:

`shesha_util.kl_util.`**`radii`**(*nr: int*, *npp: int*, *cobs: float*) → numpy.ndarray

This routine generates an nr x npp array with npp copies of the radial coordinate array. Radial coordinate span the range from r=cobs to r=1 with successive annuli having equal areas (ie, the area between cobs and 1 is divided into nr equal rings, and the points are positioned at the half-area mark on each ring). There are no points on the border.

TODO:

parameters nr:

npp:

cobs: (float) : central obstruction

return r

`shesha_util.kl_util.`**`set_pctr`**(*dim: int*, *nr*, *npp*, *nkl: int*, *cobs: float*, *nord*, *ncmar=None*, *ncp=None*)

This routine calls pcgeom to build a geom_struct with the right initializations. bas is a gkl_basis_struct built with the gkl_bas routine. TODO:

Parameters dim:

nr:

npp:

nkl:

cobs:

nord:

ncmar: (optional)

ncp: (optional)

Returns

ncp

ncmar

px

py

cr

cp

pincx

pincy

pincw

ap

`shesha_util.kl_util.`**`setpincs`**(*ax: numpy.ndarray*, *ay: numpy.ndarray*, *px: numpy.ndarray*, *py: numpy.ndarray*, *cobs: float*) → typing.Tuple[[numpy.ndarray, numpy.ndarray], numpy.ndarray]

This routine determines a set of squares for interpolating from cartesian to polar coordinates, using only those points with cobs < r < 1 SEE ALSO : pcgeom

TODO:

parameters ax:

ay:

px:

py:

cobs: (float) : central obstruction

return pincx:

pincy:

pincw

Created on 13 juil. 2017

@author: vdeo

`shesha_util.make_apodizer.`**`make_apodizer`**(*dim*, *pupd*, *filename*, *angle*)

TODO doc

Parameters (int) : im:

(int) : pupd:

(str) : filename:

(float) : angle:

`shesha_util.make_pupil.`**`make_EELT`**(*dim*, *pupd*, *tel*, *N_seg=-1*)

Initialize the EELT pupil

Parameters dim: (long) : linear size of ???

pupd: (long) : linear size of total pupil

tel: (Param_tel) : Telescope structure

N_seg: (int)

TODO: complete TODO : add force rescal pup elt

`shesha_util.make_pupil.`**`make_VLT`**(*dim*, *pupd*, *tel*)

Initialize the VLT pupil

Parameters dim: (long) : linear size of ???

pupd: (long) : linear size of total pupil

tel: (Param_tel) : Telescope structure

`shesha_util.make_pupil.`**`make_phase_ab`**(*dim*, *pupd*, *tel*, *pup*)

Compute the EELT M1 phase aberration

Parameters dim: (long) : linear size of ???

pupd: (long) : linear size of total pupil

tel: (Param_tel) : Telescope structure

pup: (?)

TODO: complete

`shesha_util.make_pupil.`**`make_pupil`**(*dim*, *pupd*, *tel*, *xc=-1*, *yc=-1*, *real=0*)

Initialize the system pupil

Parameters dim: (long) : linear size of ???

pupd: (long) : linear size of total pupil

tel: (Param_tel) : Telescope structure

xc: (int)

yc: (int)

real: (int)

cobs: (float) : central obstruction ratio.

TODO: complete

`shesha_util.make_pupil.`**`make_pupil_generic`**(*dim*, *pupd*, *t_spiders=0.01*, *spiders_type=b'six'*, *xc=0*, *yc=0*, *real=0*, *cobs=0*)

Initialize the system pupil

Parameters dim: (long) : linear size of ???

pupd: (long) : linear size of total pupil

t_spiders: (float) : secondary supports ratio.

spiders_type: (str) : secondary supports type: "four" or "six".

xc: (int)

yc: (int)

real: (int)

cobs: (float) : central obstruction ratio.

TODO: complete

`shesha_util.rtc_util.`**`centroid_gain`**(*E*, *F*)

Returns the mean centroid gain

Parameters E : (np.array(dtype=np.float32)) : measurements from WFS

F : (np.array(dtype=np.float32)) : geometric measurements

Returns cgain : (float) : mean centroid gain between the sets of WFS measurements and geometric ones

`shesha_util.rtc_util.`**`create_interp_mat`**(*dimx: int*, *dimy: int*)

TODO doc

Parameters dimx: (int) :

dimy: (int) :

Created on 1 aout 2017

@author: fferreira

`shesha_util.utilities.`**`bin2d`**(*data_in*, *binfact*)

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 int, 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

`shesha_util.utilities.`**`dist`**(*dim*, *xc=-1*, *yc=-1*)

TODO: docstring

`shesha_util.utilities.`**`fft_goodsize`**(*s*)

find best size for a fft from size s

Parameters s: (int) size

`shesha_util.utilities.`**`makegaussian`**(*size*, *fwhm*, *xc=-1*, *yc=-1*, *norm=0*)

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

Parameters

- **size** – (int) :
- **fwhm** – (float) :
- **xc** – (float) : (optional) center position on x axis
- **yc** – (float) : (optional) center position on y axis
- **norm** – (int) : (optional) normalization

`shesha_util.utilities.`**`pad_array`**(*A*, *N*)

TODO: docstring

`shesha_util.utilities.`**`rebin`**(*a*, *shape*)

TODO: docstring

`shesha_util.utilities.`**`rotate`**(*im*, *ang*, *cx=-1*, *cy=-1*, *zoom=1.0*)

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: (float) : (optional) rotation center on x axis (default: image center)

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

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

`shesha_util.utilities.`**`rotate3d`**(*im*, *ang*, *cx=-1*, *cy=-1*, *zoom=1.0*)

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: (float) : (optional) rotation center on x axis (default: image center)

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

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

CHAPTER TWO

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

S

shesha_ao, 51
shesha_ao.basis, 51
shesha_ao.cmats, 53
shesha_ao.imats, 53
shesha_ao.modopti, 54
shesha_ao.tomo, 54
shesha_ao.wfs, 55
shesha_config, 56
shesha_constants, 70
shesha_init, 71
shesha_sim, 72
shesha_util, 73
shesha_util.dm_util, 73
shesha_util.hdf5_utils, 74
shesha_util.influ_util, 77
shesha_util.iterkolmo, 79
shesha_util.kl_util, 80
shesha_util.make_apodizer, 83
shesha_util.make_pupil, 83
shesha_util.rtc_util, 84
shesha_util.utilities, 84

INDEX

A

AB() (in module shesha_util.iterkolmo), 79
activateDevice() (naga_obj.naga_obj_ComplexD1D method), 33
activateDevice() (naga_obj.naga_obj_ComplexD2D method), 34
activateDevice() (naga_obj.naga_obj_ComplexD3D method), 37
activateDevice() (naga_obj.naga_obj_ComplexD4D method), 38
activateDevice() (naga_obj.naga_obj_ComplexS1D method), 26
activateDevice() (naga_obj.naga_obj_ComplexS2D method), 27
activateDevice() (naga_obj.naga_obj_ComplexS3D method), 30
activateDevice() (naga_obj.naga_obj_ComplexS4D method), 31
activateDevice() (naga_obj.naga_obj_Double1D method), 18
activateDevice() (naga_obj.naga_obj_Double2D method), 20
activateDevice() (naga_obj.naga_obj_Double3D method), 22
activateDevice() (naga_obj.naga_obj_Double4D method), 24
activateDevice() (naga_obj.naga_obj_Float1D method), 10
activateDevice() (naga_obj.naga_obj_Float2D method), 12
activateDevice() (naga_obj.naga_obj_Float3D method), 15
activateDevice() (naga_obj.naga_obj_Float4D method), 16
activateDevice() (naga_obj.naga_obj_Int1D method), 3
activateDevice() (naga_obj.naga_obj_Int2D method), 4
activateDevice() (naga_obj.naga_obj_Int3D method), 5
activateDevice() (naga_obj.naga_obj_Int4D method), 6
activateDevice() (naga_obj.naga_obj_UInt1D method), 7
activateDevice() (naga_obj.naga_obj_UInt2D method), 8
activateDevice() (naga_obj.naga_obj_UInt3D method), 9
activateDevice() (naga_obj.naga_obj_UInt4D method), 9
ApertureType (class in shesha_constants), 70
asum() (naga_obj.naga_obj_Double1D method), 18
asum() (naga_obj.naga_obj_Double2D method), 20
asum() (naga_obj.naga_obj_Double3D method), 22
asum() (naga_obj.naga_obj_Double4D method), 24
asum() (naga_obj.naga_obj_Float1D method), 10
asum() (naga_obj.naga_obj_Float2D method), 12
asum() (naga_obj.naga_obj_Float3D method), 15
asum() (naga_obj.naga_obj_Float4D method), 16
asymp_macdo() (in module shesha_util.iterkolmo), 79
atmos_init() (in module shesha_init), 71
axpy() (naga_obj.naga_obj_ComplexD1D method), 33
axpy() (naga_obj.naga_obj_ComplexS1D method), 26
axpy() (naga_obj.naga_obj_Double1D method), 18
axpy() (naga_obj.naga_obj_Float1D method), 10

B

Bench (class in shesha_sim), 72
BenchBrama (class in shesha_sim), 72
besel_orth() (in module shesha_util.influ_util), 77
bessel_influence() (in module shesha_util.influ_util), 77
bin2d() (in module shesha_util.utilities), 84

C

centroid_gain() (in module shesha_util.rtc_util), 84
CentroiderType (class in shesha_constants), 70
check_enum() (in module shesha_constants), 71
checkControlParams() (in module shesha_util.hdf5_utils), 74
checkDmsParams() (in module shesha_util.hdf5_utils), 74
checkMatricesDataBase() (in module shesha_util.hdf5_utils), 75
checkTurbuParams() (in module shesha_util.hdf5_utils), 75
cmat_init() (in module shesha_ao.cmats), 53
command_on_Btt() (in module shesha_ao.basis), 51
command_on_KL() (in module shesha_ao.basis), 51
comp_new_fstop() (in module shesha_ao.wfs), 55
comp_new_pyr_ampl() (in module shesha_ao.wfs), 55
compute_Btt() (in module shesha_ao.basis), 52
compute_cmat_with_Btt() (in module shesha_ao.basis), 53
compute_cmat_with_KL() (in module shesha_ao.basis), 53

compute_DMbasis() (in module shesha_ao.basis), 52
compute_IFsparse() (in module shesha_ao.basis), 52
compute_KL2V() (in module shesha_ao.basis), 53
configFromH5() (in module shesha_util.hdf5_utils), 75
ControllerType (class in shesha_constants), 70
copy() (naga_obj.naga_obj_ComplexD1D method), 33
copy() (naga_obj.naga_obj_ComplexD2D method), 34
copy() (naga_obj.naga_obj_ComplexD3D method), 37
copy() (naga_obj.naga_obj_ComplexD4D method), 38
copy() (naga_obj.naga_obj_ComplexS1D method), 26
copy() (naga_obj.naga_obj_ComplexS2D method), 27
copy() (naga_obj.naga_obj_ComplexS3D method), 30
copy() (naga_obj.naga_obj_ComplexS4D method), 31
copy() (naga_obj.naga_obj_Double1D method), 18
copy() (naga_obj.naga_obj_Double2D method), 20
copy() (naga_obj.naga_obj_Double3D method), 22
copy() (naga_obj.naga_obj_Double4D method), 24
copy() (naga_obj.naga_obj_Float1D method), 10
copy() (naga_obj.naga_obj_Float2D method), 12
copy() (naga_obj.naga_obj_Float3D method), 15
copy() (naga_obj.naga_obj_Float4D method), 16
copyFrom() (naga_obj.naga_obj_ComplexD1D method), 33
copyFrom() (naga_obj.naga_obj_ComplexD2D method), 34
copyFrom() (naga_obj.naga_obj_ComplexD3D method), 37
copyFrom() (naga_obj.naga_obj_ComplexD4D method), 38
copyFrom() (naga_obj.naga_obj_ComplexS1D method), 26
copyFrom() (naga_obj.naga_obj_ComplexS2D method), 27
copyFrom() (naga_obj.naga_obj_ComplexS3D method), 30
copyFrom() (naga_obj.naga_obj_ComplexS4D method), 31
copyFrom() (naga_obj.naga_obj_Double1D method), 18
copyFrom() (naga_obj.naga_obj_Double2D method), 20
copyFrom() (naga_obj.naga_obj_Double3D method), 23
copyFrom() (naga_obj.naga_obj_Double4D method), 24
copyFrom() (naga_obj.naga_obj_Float1D method), 11
copyFrom() (naga_obj.naga_obj_Float2D method), 12
copyFrom() (naga_obj.naga_obj_Float3D method), 15
copyFrom() (naga_obj.naga_obj_Float4D method), 17
copyFrom() (naga_obj.naga_obj_Int1D method), 4
copyFrom() (naga_obj.naga_obj_Int2D method), 4
copyFrom() (naga_obj.naga_obj_Int3D method), 5
copyFrom() (naga_obj.naga_obj_Int4D method), 6
copyFrom() (naga_obj.naga_obj_UInt1D method), 7
copyFrom() (naga_obj.naga_obj_UInt2D method), 8
copyFrom() (naga_obj.naga_obj_UInt3D method), 9
copyFrom() (naga_obj.naga_obj_UInt4D method), 10
copyInto() (naga_obj.naga_obj_ComplexD1D method), 33
copyInto() (naga_obj.naga_obj_ComplexD2D method), 35
copyInto() (naga_obj.naga_obj_ComplexD3D method), 37
copyInto() (naga_obj.naga_obj_ComplexD4D method), 38
copyInto() (naga_obj.naga_obj_ComplexS1D method), 26
copyInto() (naga_obj.naga_obj_ComplexS2D method), 28
copyInto() (naga_obj.naga_obj_ComplexS3D method), 30
copyInto() (naga_obj.naga_obj_ComplexS4D method), 31
copyInto() (naga_obj.naga_obj_Double1D method), 18
copyInto() (naga_obj.naga_obj_Double2D method), 20
copyInto() (naga_obj.naga_obj_Double3D method), 23
copyInto() (naga_obj.naga_obj_Double4D method), 24
copyInto() (naga_obj.naga_obj_Float1D method), 11
copyInto() (naga_obj.naga_obj_Float2D method), 12
copyInto() (naga_obj.naga_obj_Float3D method), 15
copyInto() (naga_obj.naga_obj_Float4D method), 17
copyInto() (naga_obj.naga_obj_Int1D method), 4
copyInto() (naga_obj.naga_obj_Int2D method), 4
copyInto() (naga_obj.naga_obj_Int3D method), 5
copyInto() (naga_obj.naga_obj_Int4D method), 6
copyInto() (naga_obj.naga_obj_UInt1D method), 7
copyInto() (naga_obj.naga_obj_UInt2D method), 8
copyInto() (naga_obj.naga_obj_UInt3D method), 9
copyInto() (naga_obj.naga_obj_UInt4D method), 10
cpy_obj_from() (naga_host_obj.naga_host_obj_ComplexD1D method), 47
cpy_obj_from() (naga_host_obj.naga_host_obj_ComplexD2D method), 48
cpy_obj_from() (naga_host_obj.naga_host_obj_ComplexD3D method), 48
cpy_obj_from() (naga_host_obj.naga_host_obj_ComplexD4D method), 49
cpy_obj_from() (naga_host_obj.naga_host_obj_ComplexS1D method), 46
cpy_obj_from() (naga_host_obj.naga_host_obj_ComplexS2D method), 46
cpy_obj_from() (naga_host_obj.naga_host_obj_ComplexS3D method), 47
cpy_obj_from() (naga_host_obj.naga_host_obj_ComplexS4D method), 47
cpy_obj_from() (naga_host_obj.naga_host_obj_Double1D method), 44
cpy_obj_from() (naga_host_obj.naga_host_obj_Double2D method), 45
cpy_obj_from() (naga_host_obj.naga_host_obj_Double3D method), 45
cpy_obj_from() (naga_host_obj.naga_host_obj_Double4D method), 46
cpy_obj_from() (naga_host_obj.naga_host_obj_Float1D method), 43

cpy_obj_from() (naga_host_obj.naga_host_obj_Float2D method), 43
cpy_obj_from() (naga_host_obj.naga_host_obj_Float3D method), 44
cpy_obj_from() (naga_host_obj.naga_host_obj_Float4D method), 44
cpy_obj_from() (naga_host_obj.naga_host_obj_Int1D method), 40
cpy_obj_from() (naga_host_obj.naga_host_obj_Int2D method), 40
cpy_obj_from() (naga_host_obj.naga_host_obj_Int3D method), 41
cpy_obj_from() (naga_host_obj.naga_host_obj_Int4D method), 41
cpy_obj_from() (naga_host_obj.naga_host_obj_UInt1D method), 41
cpy_obj_from() (naga_host_obj.naga_host_obj_UInt2D method), 42
cpy_obj_from() (naga_host_obj.naga_host_obj_UInt3D method), 42
cpy_obj_from() (naga_host_obj.naga_host_obj_UInt4D method), 42
cpy_obj_into() (naga_host_obj.naga_host_obj_ComplexD1D method), 48
cpy_obj_into() (naga_host_obj.naga_host_obj_ComplexD2D method), 48
cpy_obj_into() (naga_host_obj.naga_host_obj_ComplexD3D method), 48
cpy_obj_into() (naga_host_obj.naga_host_obj_ComplexD4D method), 49
cpy_obj_into() (naga_host_obj.naga_host_obj_ComplexS1D method), 46
cpy_obj_into() (naga_host_obj.naga_host_obj_ComplexS2D method), 46
cpy_obj_into() (naga_host_obj.naga_host_obj_ComplexS3D method), 47
cpy_obj_into() (naga_host_obj.naga_host_obj_ComplexS4D method), 47
cpy_obj_into() (naga_host_obj.naga_host_obj_Double1D method), 44
cpy_obj_into() (naga_host_obj.naga_host_obj_Double2D method), 45
cpy_obj_into() (naga_host_obj.naga_host_obj_Double3D method), 45
cpy_obj_into() (naga_host_obj.naga_host_obj_Double4D method), 46
cpy_obj_into() (naga_host_obj.naga_host_obj_Float1D method), 43
cpy_obj_into() (naga_host_obj.naga_host_obj_Float2D method), 43
cpy_obj_into() (naga_host_obj.naga_host_obj_Float3D method), 44
cpy_obj_into() (naga_host_obj.naga_host_obj_Float4D method), 44
cpy_obj_into() (naga_host_obj.naga_host_obj_Int1D method), 40
cpy_obj_into() (naga_host_obj.naga_host_obj_Int2D method), 40
cpy_obj_into() (naga_host_obj.naga_host_obj_Int3D method), 41
cpy_obj_into() (naga_host_obj.naga_host_obj_Int4D method), 41
cpy_obj_into() (naga_host_obj.naga_host_obj_UInt1D method), 41
cpy_obj_into() (naga_host_obj.naga_host_obj_UInt2D method), 42
cpy_obj_into() (naga_host_obj.naga_host_obj_UInt3D method), 42
cpy_obj_into() (naga_host_obj.naga_host_obj_UInt4D method), 43
create_file_attributes() (in module shesha_util.hdf5_utils), 75
create_interp_mat() (in module shesha_util.rtc_util), 84
create_nact_geom() (in module shesha_ao.tomo), 54
create_piston_filter() (in module shesha_ao.tomo), 54
create_screen() (in module shesha_util.iterkolmo), 79
create_screen_assist() (in module shesha_util.iterkolmo), 79
create_stencil() (in module shesha_util.iterkolmo), 79
createDoubleHexaPattern() (in module shesha_util.dm_util), 73
createHexaPattern() (in module shesha_util.dm_util), 73
createSquarePattern() (in module shesha_util.dm_util), 73
Cxx() (in module shesha_util.iterkolmo), 79
Cxz() (in module shesha_util.iterkolmo), 79
Czz() (in module shesha_util.iterkolmo), 79

D

device2host() (naga_obj.naga_obj_ComplexD1D method), 33
device2host() (naga_obj.naga_obj_ComplexD2D method), 35
device2host() (naga_obj.naga_obj_ComplexD3D method), 37
device2host() (naga_obj.naga_obj_ComplexD4D method), 38
device2host() (naga_obj.naga_obj_ComplexS1D method), 26
device2host() (naga_obj.naga_obj_ComplexS2D method), 28
device2host() (naga_obj.naga_obj_ComplexS3D method), 30
device2host() (naga_obj.naga_obj_ComplexS4D method), 31
device2host() (naga_obj.naga_obj_Double1D method), 18
device2host() (naga_obj.naga_obj_Double2D method), 20
device2host() (naga_obj.naga_obj_Double3D method), 23
device2host() (naga_obj.naga_obj_Double4D method), 24
device2host() (naga_obj.naga_obj_Float1D method), 11

device2host() (naga_obj.naga_obj_Float2D method), 12
device2host() (naga_obj.naga_obj_Float3D method), 15
device2host() (naga_obj.naga_obj_Float4D method), 17
device2host() (naga_obj.naga_obj_Int1D method), 4
device2host() (naga_obj.naga_obj_Int2D method), 5
device2host() (naga_obj.naga_obj_Int3D method), 5
device2host() (naga_obj.naga_obj_Int4D method), 6
device2host() (naga_obj.naga_obj_UInt1D method), 7
device2host() (naga_obj.naga_obj_UInt2D method), 8
device2host() (naga_obj.naga_obj_UInt3D method), 9
device2host() (naga_obj.naga_obj_UInt4D method), 10
device2hostOpt() (naga_obj.naga_obj_ComplexD1D method), 33
device2hostOpt() (naga_obj.naga_obj_ComplexD2D method), 35
device2hostOpt() (naga_obj.naga_obj_ComplexD3D method), 37
device2hostOpt() (naga_obj.naga_obj_ComplexD4D method), 39
device2hostOpt() (naga_obj.naga_obj_ComplexS1D method), 26
device2hostOpt() (naga_obj.naga_obj_ComplexS2D method), 28
device2hostOpt() (naga_obj.naga_obj_ComplexS3D method), 30
device2hostOpt() (naga_obj.naga_obj_ComplexS4D method), 32
device2hostOpt() (naga_obj.naga_obj_Double1D method), 18
device2hostOpt() (naga_obj.naga_obj_Double2D method), 20
device2hostOpt() (naga_obj.naga_obj_Double3D method), 23
device2hostOpt() (naga_obj.naga_obj_Double4D method), 24
device2hostOpt() (naga_obj.naga_obj_Float1D method), 11
device2hostOpt() (naga_obj.naga_obj_Float2D method), 13
device2hostOpt() (naga_obj.naga_obj_Float3D method), 15
device2hostOpt() (naga_obj.naga_obj_Float4D method), 17
device2hostOpt() (naga_obj.naga_obj_Int1D method), 4
device2hostOpt() (naga_obj.naga_obj_Int2D method), 5
device2hostOpt() (naga_obj.naga_obj_Int3D method), 5
device2hostOpt() (naga_obj.naga_obj_Int4D method), 6
device2hostOpt() (naga_obj.naga_obj_UInt1D method), 7
device2hostOpt() (naga_obj.naga_obj_UInt2D method), 8
device2hostOpt() (naga_obj.naga_obj_UInt3D method), 9
device2hostOpt() (naga_obj.naga_obj_UInt4D method), 10
dgmm() (naga_obj.naga_obj_ComplexD2D method), 35
dgmm() (naga_obj.naga_obj_ComplexS2D method), 28
dgmm() (naga_obj.naga_obj_Double2D method), 20
dgmm() (naga_obj.naga_obj_Float2D method), 13
dim_dm_patch() (in module shesha_util.dm_util), 73
dim_dm_support() (in module shesha_util.dm_util), 73
dist() (in module shesha_util.utilities), 84
dm_init() (in module shesha_init), 71
DmType (class in shesha_constants), 70
do_tomo_matrices() (in module shesha_ao.tomo), 55
dot() (naga_obj.naga_obj_ComplexD1D method), 33
dot() (naga_obj.naga_obj_ComplexD2D method), 35
dot() (naga_obj.naga_obj_ComplexD3D method), 37
dot() (naga_obj.naga_obj_ComplexD4D method), 39
dot() (naga_obj.naga_obj_ComplexS1D method), 26
dot() (naga_obj.naga_obj_ComplexS2D method), 28
dot() (naga_obj.naga_obj_ComplexS3D method), 30
dot() (naga_obj.naga_obj_ComplexS4D method), 32
dot() (naga_obj.naga_obj_Double1D method), 19
dot() (naga_obj.naga_obj_Double2D method), 20
dot() (naga_obj.naga_obj_Double3D method), 23
dot() (naga_obj.naga_obj_Double4D method), 24
dot() (naga_obj.naga_obj_Float1D method), 11
dot() (naga_obj.naga_obj_Float2D method), 13
dot() (naga_obj.naga_obj_Float3D method), 15
dot() (naga_obj.naga_obj_Float4D method), 17

E

extrude() (in module shesha_util.iterkolmo), 79

F

fft() (naga_obj.naga_obj_ComplexD1D method), 33
fft() (naga_obj.naga_obj_ComplexD2D method), 35
fft() (naga_obj.naga_obj_ComplexD3D method), 37
fft() (naga_obj.naga_obj_ComplexD4D method), 39
fft() (naga_obj.naga_obj_ComplexS1D method), 26
fft() (naga_obj.naga_obj_ComplexS2D method), 28
fft() (naga_obj.naga_obj_ComplexS3D method), 30
fft() (naga_obj.naga_obj_ComplexS4D method), 32
fft() (naga_obj.naga_obj_Double1D method), 19
fft() (naga_obj.naga_obj_Double2D method), 20
fft() (naga_obj.naga_obj_Double3D method), 23
fft() (naga_obj.naga_obj_Double4D method), 25
fft() (naga_obj.naga_obj_Float1D method), 11
fft() (naga_obj.naga_obj_Float2D method), 13
fft() (naga_obj.naga_obj_Float3D method), 15
fft() (naga_obj.naga_obj_Float4D method), 17
fft_goodsize() (in module shesha_util.utilities), 84
FieldStopType (class in shesha_constants), 70

G

geam() (naga_obj.naga_obj_ComplexD2D method), 35

geam() (naga_obj.naga_obj_ComplexS2D method), 28
geam() (naga_obj.naga_obj_Double2D method), 20
geam() (naga_obj.naga_obj_Float2D method), 13
gemm() (naga_obj.naga_obj_ComplexD2D method), 35
gemm() (naga_obj.naga_obj_ComplexS2D method), 28
gemm() (naga_obj.naga_obj_Double2D method), 21
gemm() (naga_obj.naga_obj_Float2D method), 13
gemv() (naga_obj.naga_obj_ComplexD2D method), 35
gemv() (naga_obj.naga_obj_ComplexS2D method), 28
gemv() (naga_obj.naga_obj_Double2D method), 21
gemv() (naga_obj.naga_obj_Float2D method), 13
ger() (naga_obj.naga_obj_ComplexD1D method), 33
ger() (naga_obj.naga_obj_ComplexS1D method), 26
ger() (naga_obj.naga_obj_Double1D method), 19
ger() (naga_obj.naga_obj_Float1D method), 11
get_activeDevice() (naga_context.naga_context method), 3
get_cudaDriverGetVersion() (naga_context.naga_context method), 3
get_cudaRuntimeGetVersion() (naga_context.naga_context method), 3
get_device_names() (naga_context.naga_context method), 3
get_Dims() (naga_host_obj.naga_host_obj_ComplexD1D method), 48
get_Dims() (naga_host_obj.naga_host_obj_ComplexD2D method), 48
get_Dims() (naga_host_obj.naga_host_obj_ComplexD3D method), 48
get_Dims() (naga_host_obj.naga_host_obj_ComplexD4D method), 49
get_Dims() (naga_host_obj.naga_host_obj_ComplexS1D method), 46
get_Dims() (naga_host_obj.naga_host_obj_ComplexS2D method), 47
get_Dims() (naga_host_obj.naga_host_obj_ComplexS3D method), 47
get_Dims() (naga_host_obj.naga_host_obj_ComplexS4D method), 47
get_Dims() (naga_host_obj.naga_host_obj_Double1D method), 45
get_Dims() (naga_host_obj.naga_host_obj_Double2D method), 45
get_Dims() (naga_host_obj.naga_host_obj_Double3D method), 45
get_Dims() (naga_host_obj.naga_host_obj_Double4D method), 46
get_Dims() (naga_host_obj.naga_host_obj_Float1D method), 43
get_Dims() (naga_host_obj.naga_host_obj_Float2D method), 43
get_Dims() (naga_host_obj.naga_host_obj_Float3D method), 44
get_Dims() (naga_host_obj.naga_host_obj_Float4D method), 44
get_Dims() (naga_host_obj.naga_host_obj_Int1D method), 40
get_Dims() (naga_host_obj.naga_host_obj_Int2D method), 40
get_Dims() (naga_host_obj.naga_host_obj_Int3D method), 41
get_Dims() (naga_host_obj.naga_host_obj_Int4D method), 41
get_Dims() (naga_host_obj.naga_host_obj_UInt1D method), 42
get_Dims() (naga_host_obj.naga_host_obj_UInt2D method), 42
get_Dims() (naga_host_obj.naga_host_obj_UInt3D method), 42
get_Dims() (naga_host_obj.naga_host_obj_UInt4D method), 43
get_Dims() (naga_obj.naga_obj_ComplexD1D method), 34
get_Dims() (naga_obj.naga_obj_ComplexD2D method), 36
get_Dims() (naga_obj.naga_obj_ComplexD3D method), 38
get_Dims() (naga_obj.naga_obj_ComplexD4D method), 39
get_Dims() (naga_obj.naga_obj_ComplexS1D method), 27
get_Dims() (naga_obj.naga_obj_ComplexS2D method), 29
get_Dims() (naga_obj.naga_obj_ComplexS3D method), 31
get_Dims() (naga_obj.naga_obj_ComplexS4D method), 32
get_Dims() (naga_obj.naga_obj_Double1D method), 19
get_Dims() (naga_obj.naga_obj_Double2D method), 21
get_Dims() (naga_obj.naga_obj_Double3D method), 23
get_Dims() (naga_obj.naga_obj_Double4D method), 25
get_Dims() (naga_obj.naga_obj_Float1D method), 11
get_Dims() (naga_obj.naga_obj_Float2D method), 13
get_Dims() (naga_obj.naga_obj_Float3D method), 16
get_Dims() (naga_obj.naga_obj_Float4D method), 17
get_Dims() (naga_obj.naga_obj_Int1D method), 4
get_Dims() (naga_obj.naga_obj_Int2D method), 5
get_Dims() (naga_obj.naga_obj_Int3D method), 6
get_Dims() (naga_obj.naga_obj_Int4D method), 7
get_Dims() (naga_obj.naga_obj_UInt1D method), 7
get_Dims() (naga_obj.naga_obj_UInt2D method), 8
get_Dims() (naga_obj.naga_obj_UInt3D method), 9
get_Dims() (naga_obj.naga_obj_UInt4D method), 10
get_host_obj_ptr() (naga_host_obj.naga_host_obj_ComplexD1D method), 48
get_host_obj_ptr() (naga_host_obj.naga_host_obj_ComplexD2D method), 48
get_host_obj_ptr() (naga_host_obj.naga_host_obj_ComplexD3D method), 48
get_host_obj_ptr() (naga_host_obj.naga_host_obj_ComplexD4D

method), 49
get_host_obj_ptr() (naga_host_obj.naga_host_obj_ComplexS1D method), 46
get_host_obj_ptr() (naga_host_obj.naga_host_obj_ComplexS2D method), 47
get_host_obj_ptr() (naga_host_obj.naga_host_obj_ComplexS3D method), 47
get_host_obj_ptr() (naga_host_obj.naga_host_obj_ComplexS4D method), 47
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Double1D method), 45
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Double2D method), 45
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Double3D method), 45
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Double4D method), 46
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Float1D method), 43
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Float2D method), 43
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Float3D method), 44
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Float4D method), 44
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Int1D method), 40
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Int2D method), 40
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Int3D method), 41
get_host_obj_ptr() (naga_host_obj.naga_host_obj_Int4D method), 41
get_host_obj_ptr() (naga_host_obj.naga_host_obj_UInt1D method), 42
get_host_obj_ptr() (naga_host_obj.naga_host_obj_UInt2D method), 42
get_host_obj_ptr() (naga_host_obj.naga_host_obj_UInt3D method), 42
get_host_obj_ptr() (naga_host_obj.naga_host_obj_UInt4D method), 43
get_magma_info() (naga_context.naga_context method), 3
get_ndevice() (naga_context.naga_context method), 3
get_sparse() (naga_sparse_obj.naga_sparse_obj_Double method), 49
get_sparse() (naga_sparse_obj.naga_sparse_obj_Float method), 49
get_validsub() (shesha_config.Param_wfs method), 66
getCarma_ptr() (naga_obj.naga_obj_ComplexD1D method), 33
getCarma_ptr() (naga_obj.naga_obj_ComplexD2D method), 35
getCarma_ptr() (naga_obj.naga_obj_ComplexD3D method), 37
getCarma_ptr() (naga_obj.naga_obj_ComplexD4D method), 39
getCarma_ptr() (naga_obj.naga_obj_ComplexS1D method), 26
getCarma_ptr() (naga_obj.naga_obj_ComplexS2D method), 28
getCarma_ptr() (naga_obj.naga_obj_ComplexS3D method), 30
getCarma_ptr() (naga_obj.naga_obj_ComplexS4D method), 32
getCarma_ptr() (naga_obj.naga_obj_Double1D method), 19
getCarma_ptr() (naga_obj.naga_obj_Double2D method), 21
getCarma_ptr() (naga_obj.naga_obj_Double3D method), 23
getCarma_ptr() (naga_obj.naga_obj_Double4D method), 25
getCarma_ptr() (naga_obj.naga_obj_Float1D method), 11
getCarma_ptr() (naga_obj.naga_obj_Float2D method), 13
getCarma_ptr() (naga_obj.naga_obj_Float3D method), 15
getCarma_ptr() (naga_obj.naga_obj_Float4D method), 17
getCarma_ptr() (naga_obj.naga_obj_Int1D method), 4
getCarma_ptr() (naga_obj.naga_obj_Int2D method), 5
getCarma_ptr() (naga_obj.naga_obj_Int3D method), 6
getCarma_ptr() (naga_obj.naga_obj_Int4D method), 6
getCarma_ptr() (naga_obj.naga_obj_UInt1D method), 7
getCarma_ptr() (naga_obj.naga_obj_UInt2D method), 8
getCarma_ptr() (naga_obj.naga_obj_UInt3D method), 9
getCarma_ptr() (naga_obj.naga_obj_UInt4D method), 10
getContext() (naga_obj.naga_obj_ComplexD1D method), 34
getContext() (naga_obj.naga_obj_ComplexD2D method), 35
getContext() (naga_obj.naga_obj_ComplexD3D method), 37
getContext() (naga_obj.naga_obj_ComplexD4D method), 39
getContext() (naga_obj.naga_obj_ComplexS1D method), 27
getContext() (naga_obj.naga_obj_ComplexS2D method), 28
getContext() (naga_obj.naga_obj_ComplexS3D method), 30
getContext() (naga_obj.naga_obj_ComplexS4D method), 32
getContext() (naga_obj.naga_obj_Double1D method), 19
getContext() (naga_obj.naga_obj_Double2D method), 21
getContext() (naga_obj.naga_obj_Double3D method), 23
getContext() (naga_obj.naga_obj_Double4D method),

25
getContext() (naga_obj.naga_obj_Float1D method), 11
getContext() (naga_obj.naga_obj_Float2D method), 13
getContext() (naga_obj.naga_obj_Float3D method), 15
getContext() (naga_obj.naga_obj_Float4D method), 17
getContext() (naga_obj.naga_obj_Int1D method), 4
getContext() (naga_obj.naga_obj_Int2D method), 5
getContext() (naga_obj.naga_obj_Int3D method), 6
getContext() (naga_obj.naga_obj_Int4D method), 6
getContext() (naga_obj.naga_obj_UInt1D method), 7
getContext() (naga_obj.naga_obj_UInt2D method), 8
getContext() (naga_obj.naga_obj_UInt3D method), 9
getContext() (naga_obj.naga_obj_UInt4D method), 10
getData() (naga_host_obj.naga_host_obj_ComplexD1D method), 48
getData() (naga_host_obj.naga_host_obj_ComplexD2D method), 48
getData() (naga_host_obj.naga_host_obj_ComplexD3D method), 48
getData() (naga_host_obj.naga_host_obj_ComplexD4D method), 49
getData() (naga_host_obj.naga_host_obj_ComplexS1D method), 46
getData() (naga_host_obj.naga_host_obj_ComplexS2D method), 46
getData() (naga_host_obj.naga_host_obj_ComplexS3D method), 47
getData() (naga_host_obj.naga_host_obj_ComplexS4D method), 47
getData() (naga_host_obj.naga_host_obj_Double1D method), 45
getData() (naga_host_obj.naga_host_obj_Double2D method), 45
getData() (naga_host_obj.naga_host_obj_Double3D method), 45
getData() (naga_host_obj.naga_host_obj_Double4D method), 46
getData() (naga_host_obj.naga_host_obj_Float1D method), 43
getData() (naga_host_obj.naga_host_obj_Float2D method), 43
getData() (naga_host_obj.naga_host_obj_Float3D method), 44
getData() (naga_host_obj.naga_host_obj_Float4D method), 44
getData() (naga_host_obj.naga_host_obj_Int1D method), 40
getData() (naga_host_obj.naga_host_obj_Int2D method), 40
getData() (naga_host_obj.naga_host_obj_Int3D method), 41
getData() (naga_host_obj.naga_host_obj_Int4D method), 41
getData() (naga_host_obj.naga_host_obj_UInt1D method), 41
getData() (naga_host_obj.naga_host_obj_UInt2D method), 42
getData() (naga_host_obj.naga_host_obj_UInt3D method), 42
getData() (naga_host_obj.naga_host_obj_UInt4D method), 43
getData_ptr() (naga_obj.naga_obj_ComplexD1D method), 34
getData_ptr() (naga_obj.naga_obj_ComplexD2D method), 35
getData_ptr() (naga_obj.naga_obj_ComplexD3D method), 37
getData_ptr() (naga_obj.naga_obj_ComplexD4D method), 39
getData_ptr() (naga_obj.naga_obj_ComplexS1D method), 27
getData_ptr() (naga_obj.naga_obj_ComplexS2D method), 28
getData_ptr() (naga_obj.naga_obj_ComplexS3D method), 30
getData_ptr() (naga_obj.naga_obj_ComplexS4D method), 32
getData_ptr() (naga_obj.naga_obj_Double1D method), 19
getData_ptr() (naga_obj.naga_obj_Double2D method), 21
getData_ptr() (naga_obj.naga_obj_Double3D method), 23
getData_ptr() (naga_obj.naga_obj_Double4D method), 25
getData_ptr() (naga_obj.naga_obj_Float1D method), 11
getData_ptr() (naga_obj.naga_obj_Float2D method), 13
getData_ptr() (naga_obj.naga_obj_Float3D method), 15
getData_ptr() (naga_obj.naga_obj_Float4D method), 17
getData_ptr() (naga_obj.naga_obj_Int1D method), 4
getData_ptr() (naga_obj.naga_obj_Int2D method), 5
getData_ptr() (naga_obj.naga_obj_Int3D method), 6
getData_ptr() (naga_obj.naga_obj_Int4D method), 6
getData_ptr() (naga_obj.naga_obj_UInt1D method), 7
getData_ptr() (naga_obj.naga_obj_UInt2D method), 8
getData_ptr() (naga_obj.naga_obj_UInt3D method), 9
getData_ptr() (naga_obj.naga_obj_UInt4D method), 10
getDevice() (naga_obj.naga_obj_ComplexD1D method), 34
getDevice() (naga_obj.naga_obj_ComplexD2D method), 36
getDevice() (naga_obj.naga_obj_ComplexD3D method), 37
getDevice() (naga_obj.naga_obj_ComplexD4D method), 39
getDevice() (naga_obj.naga_obj_ComplexS1D method), 27
getDevice() (naga_obj.naga_obj_ComplexS2D method), 29
getDevice() (naga_obj.naga_obj_ComplexS3D method), 30
getDevice() (naga_obj.naga_obj_ComplexS4D

method), 32
getDevice() (naga_obj.naga_obj_Double1D method), 19
getDevice() (naga_obj.naga_obj_Double2D method), 21
getDevice() (naga_obj.naga_obj_Double3D method), 23
getDevice() (naga_obj.naga_obj_Double4D method), 25
getDevice() (naga_obj.naga_obj_Float1D method), 11
getDevice() (naga_obj.naga_obj_Float2D method), 13
getDevice() (naga_obj.naga_obj_Float3D method), 15
getDevice() (naga_obj.naga_obj_Float4D method), 17
getDevice() (naga_obj.naga_obj_Int1D method), 4
getDevice() (naga_obj.naga_obj_Int2D method), 5
getDevice() (naga_obj.naga_obj_Int3D method), 6
getDevice() (naga_obj.naga_obj_Int4D method), 7
getDevice() (naga_obj.naga_obj_UInt1D method), 7
getDevice() (naga_obj.naga_obj_UInt2D method), 8
getDevice() (naga_obj.naga_obj_UInt3D method), 9
getDevice() (naga_obj.naga_obj_UInt4D method), 10
getNbElem() (naga_host_obj.naga_host_obj_ComplexD1D method), 48
getNbElem() (naga_host_obj.naga_host_obj_ComplexD2D method), 48
getNbElem() (naga_host_obj.naga_host_obj_ComplexD3D method), 48
getNbElem() (naga_host_obj.naga_host_obj_ComplexD4D method), 49
getNbElem() (naga_host_obj.naga_host_obj_ComplexS1D method), 46
getNbElem() (naga_host_obj.naga_host_obj_ComplexS2D method), 46
getNbElem() (naga_host_obj.naga_host_obj_ComplexS3D method), 47
getNbElem() (naga_host_obj.naga_host_obj_ComplexS4D method), 47
getNbElem() (naga_host_obj.naga_host_obj_Double1D method), 45
getNbElem() (naga_host_obj.naga_host_obj_Double2D method), 45
getNbElem() (naga_host_obj.naga_host_obj_Double3D method), 45
getNbElem() (naga_host_obj.naga_host_obj_Double4D method), 46
getNbElem() (naga_host_obj.naga_host_obj_Float1D method), 43
getNbElem() (naga_host_obj.naga_host_obj_Float2D method), 43
getNbElem() (naga_host_obj.naga_host_obj_Float3D method), 44
getNbElem() (naga_host_obj.naga_host_obj_Float4D method), 44
getNbElem() (naga_host_obj.naga_host_obj_Int1D method), 40
getNbElem() (naga_host_obj.naga_host_obj_Int2D method), 40
getNbElem() (naga_host_obj.naga_host_obj_Int3D method), 41
getNbElem() (naga_host_obj.naga_host_obj_Int4D method), 41
getNbElem() (naga_host_obj.naga_host_obj_UInt1D method), 41
getNbElem() (naga_host_obj.naga_host_obj_UInt2D method), 42
getNbElem() (naga_host_obj.naga_host_obj_UInt3D method), 42
getNbElem() (naga_host_obj.naga_host_obj_UInt4D method), 43
getNbElem() (naga_obj.naga_obj_ComplexD1D method), 34
getNbElem() (naga_obj.naga_obj_ComplexD2D method), 36
getNbElem() (naga_obj.naga_obj_ComplexD3D method), 38
getNbElem() (naga_obj.naga_obj_ComplexD4D method), 39
getNbElem() (naga_obj.naga_obj_ComplexS1D method), 27
getNbElem() (naga_obj.naga_obj_ComplexS2D method), 29
getNbElem() (naga_obj.naga_obj_ComplexS3D method), 31
getNbElem() (naga_obj.naga_obj_ComplexS4D method), 32
getNbElem() (naga_obj.naga_obj_Double1D method), 19
getNbElem() (naga_obj.naga_obj_Double2D method), 21
getNbElem() (naga_obj.naga_obj_Double3D method), 23
getNbElem() (naga_obj.naga_obj_Double4D method), 25
getNbElem() (naga_obj.naga_obj_Float1D method), 11
getNbElem() (naga_obj.naga_obj_Float2D method), 13
getNbElem() (naga_obj.naga_obj_Float3D method), 15
getNbElem() (naga_obj.naga_obj_Float4D method), 17
getNbElem() (naga_obj.naga_obj_Int1D method), 4
getNbElem() (naga_obj.naga_obj_Int2D method), 5
getNbElem() (naga_obj.naga_obj_Int3D method), 6
getNbElem() (naga_obj.naga_obj_Int4D method), 7
getNbElem() (naga_obj.naga_obj_UInt1D method), 7
getNbElem() (naga_obj.naga_obj_UInt2D method), 8
getNbElem() (naga_obj.naga_obj_UInt3D method), 9
getNbElem() (naga_obj.naga_obj_UInt4D method), 10
getri_Double() (in module naga_magma), 49
getri_Float() (in module naga_magma), 49
getri_host_Double() (in module naga_magma), 50
getri_host_Float() (in module naga_magma), 50
getValues() (naga_obj.naga_obj_ComplexD1D method), 34
getValues() (naga_obj.naga_obj_ComplexD2D method), 36
getValues() (naga_obj.naga_obj_ComplexD3D method), 38
getValues() (naga_obj.naga_obj_ComplexD4D

method), 39
getValues() (naga_obj.naga_obj_ComplexS1D method), 27
getValues() (naga_obj.naga_obj_ComplexS2D method), 29
getValues() (naga_obj.naga_obj_ComplexS3D method), 31
getValues() (naga_obj.naga_obj_ComplexS4D method), 32
getValues() (naga_obj.naga_obj_Double1D method), 19
getValues() (naga_obj.naga_obj_Double2D method), 21
getValues() (naga_obj.naga_obj_Double3D method), 23
getValues() (naga_obj.naga_obj_Double4D method), 25
getValues() (naga_obj.naga_obj_Float1D method), 11
getValues() (naga_obj.naga_obj_Float2D method), 13
getValues() (naga_obj.naga_obj_Float3D method), 15
getValues() (naga_obj.naga_obj_Float4D method), 17
getValues() (naga_obj.naga_obj_Int1D method), 4
getValues() (naga_obj.naga_obj_Int2D method), 5
getValues() (naga_obj.naga_obj_Int3D method), 6
getValues() (naga_obj.naga_obj_Int4D method), 7
getValues() (naga_obj.naga_obj_UInt1D method), 7
getValues() (naga_obj.naga_obj_UInt2D method), 8
getValues() (naga_obj.naga_obj_UInt3D method), 9
getValues() (naga_obj.naga_obj_UInt4D method), 10
gkl_fcom() (in module shesha_util.kl_util), 80

H

host2device() (naga_obj.naga_obj_ComplexD1D method), 34
host2device() (naga_obj.naga_obj_ComplexD2D method), 36
host2device() (naga_obj.naga_obj_ComplexD3D method), 38
host2device() (naga_obj.naga_obj_ComplexD4D method), 39
host2device() (naga_obj.naga_obj_ComplexS1D method), 27
host2device() (naga_obj.naga_obj_ComplexS2D method), 29
host2device() (naga_obj.naga_obj_ComplexS3D method), 31
host2device() (naga_obj.naga_obj_ComplexS4D method), 32
host2device() (naga_obj.naga_obj_Double1D method), 19
host2device() (naga_obj.naga_obj_Double2D method), 21
host2device() (naga_obj.naga_obj_Double3D method), 23
host2device() (naga_obj.naga_obj_Double4D method), 25
host2device() (naga_obj.naga_obj_Float1D method), 11
host2device() (naga_obj.naga_obj_Float2D method), 13
host2device() (naga_obj.naga_obj_Float3D method), 16
host2device() (naga_obj.naga_obj_Float4D method), 17
host2device() (naga_obj.naga_obj_Int1D method), 4
host2device() (naga_obj.naga_obj_Int2D method), 5
host2device() (naga_obj.naga_obj_Int3D method), 6
host2device() (naga_obj.naga_obj_Int4D method), 7
host2device() (naga_obj.naga_obj_UInt1D method), 8
host2device() (naga_obj.naga_obj_UInt2D method), 8
host2device() (naga_obj.naga_obj_UInt3D method), 9
host2device() (naga_obj.naga_obj_UInt4D method), 10

I

imat_geom() (in module shesha_ao.imats), 53
imat_init() (in module shesha_ao.imats), 54
imax() (naga_obj.naga_obj_ComplexD1D method), 34
imax() (naga_obj.naga_obj_ComplexD2D method), 36
imax() (naga_obj.naga_obj_ComplexD3D method), 38
imax() (naga_obj.naga_obj_ComplexD4D method), 39
imax() (naga_obj.naga_obj_ComplexS1D method), 27
imax() (naga_obj.naga_obj_ComplexS2D method), 29
imax() (naga_obj.naga_obj_ComplexS3D method), 31
imax() (naga_obj.naga_obj_ComplexS4D method), 32
imax() (naga_obj.naga_obj_Double1D method), 19
imax() (naga_obj.naga_obj_Double2D method), 21
imax() (naga_obj.naga_obj_Double3D method), 23
imax() (naga_obj.naga_obj_Double4D method), 25
imax() (naga_obj.naga_obj_Float1D method), 11
imax() (naga_obj.naga_obj_Float2D method), 14
imax() (naga_obj.naga_obj_Float3D method), 16
imax() (naga_obj.naga_obj_Float4D method), 17
imin() (naga_obj.naga_obj_ComplexD1D method), 34
imin() (naga_obj.naga_obj_ComplexD2D method), 36
imin() (naga_obj.naga_obj_ComplexD3D method), 38
imin() (naga_obj.naga_obj_ComplexD4D method), 39
imin() (naga_obj.naga_obj_ComplexS1D method), 27
imin() (naga_obj.naga_obj_ComplexS2D method), 29
imin() (naga_obj.naga_obj_ComplexS3D method), 31
imin() (naga_obj.naga_obj_ComplexS4D method), 32
imin() (naga_obj.naga_obj_Double1D method), 19
imin() (naga_obj.naga_obj_Double2D method), 21
imin() (naga_obj.naga_obj_Double3D method), 23
imin() (naga_obj.naga_obj_Double4D method), 25
imin() (naga_obj.naga_obj_Float1D method), 11
imin() (naga_obj.naga_obj_Float2D method), 14
imin() (naga_obj.naga_obj_Float3D method), 16
imin() (naga_obj.naga_obj_Float4D method), 17
InfluType (class in shesha_constants), 70
init_hdf5_files() (in module shesha_util.hdf5_utils), 75
init_prng() (naga_obj.naga_obj_Double1D method), 19
init_prng() (naga_obj.naga_obj_Double2D method), 21
init_prng() (naga_obj.naga_obj_Double3D method), 23
init_prng() (naga_obj.naga_obj_Double4D method), 25
init_prng() (naga_obj.naga_obj_Float1D method), 12
init_prng() (naga_obj.naga_obj_Float2D method), 14

init_prng() (naga_obj.naga_obj_Float3D method), 16
init_prng() (naga_obj.naga_obj_Float4D method), 17
init_sim() (shesha_sim.Simulator method), 72
initDataBase() (in module shesha_util.hdf5_utils), 75
is_rng_init() (naga_obj.naga_obj_ComplexD1D method), 34
is_rng_init() (naga_obj.naga_obj_ComplexD2D method), 36
is_rng_init() (naga_obj.naga_obj_ComplexD3D method), 38
is_rng_init() (naga_obj.naga_obj_ComplexD4D method), 39
is_rng_init() (naga_obj.naga_obj_ComplexS1D method), 27
is_rng_init() (naga_obj.naga_obj_ComplexS2D method), 29
is_rng_init() (naga_obj.naga_obj_ComplexS3D method), 31
is_rng_init() (naga_obj.naga_obj_ComplexS4D method), 32
is_rng_init() (naga_obj.naga_obj_Double1D method), 19
is_rng_init() (naga_obj.naga_obj_Double2D method), 21
is_rng_init() (naga_obj.naga_obj_Double3D method), 24
is_rng_init() (naga_obj.naga_obj_Double4D method), 25
is_rng_init() (naga_obj.naga_obj_Float1D method), 12
is_rng_init() (naga_obj.naga_obj_Float2D method), 14
is_rng_init() (naga_obj.naga_obj_Float3D method), 16
is_rng_init() (naga_obj.naga_obj_Float4D method), 18
is_rng_init() (naga_obj.naga_obj_Int1D method), 4
is_rng_init() (naga_obj.naga_obj_Int2D method), 5
is_rng_init() (naga_obj.naga_obj_Int3D method), 6
is_rng_init() (naga_obj.naga_obj_Int4D method), 7
is_rng_init() (naga_obj.naga_obj_UInt1D method), 8
is_rng_init() (naga_obj.naga_obj_UInt2D method), 8
is_rng_init() (naga_obj.naga_obj_UInt3D method), 9
is_rng_init() (naga_obj.naga_obj_UInt4D method), 10

K

KLType (class in shesha_constants), 70

L

load_AB_from_dataBase() (in module shesha_util.hdf5_utils), 75
load_dm_geom_from_dataBase() (in module shesha_util.hdf5_utils), 75
load_from_file() (shesha_sim.Simulator method), 72
load_imat_from_dataBase() (in module shesha_util.hdf5_utils), 75
loop() (shesha_sim.Simulator method), 72

M

macdo_x56() (in module shesha_util.iterkolmo), 79
make_apodizer() (in module shesha_util.make_apodizer), 83
make_azimuth() (in module shesha_util.kl_util), 80
make_EELT() (in module shesha_util.make_pupil), 83
make_kernels() (in module shesha_util.kl_util), 80
make_phase_ab() (in module shesha_util.make_pupil), 83
make_pupil() (in module shesha_util.make_pupil), 83
make_pupil_generic() (in module shesha_util.make_pupil), 84
make_radii() (in module shesha_util.kl_util), 81
make_VLT() (in module shesha_util.make_pupil), 83
make_zernike() (in module shesha_util.dm_util), 74
makeBessel() (in module shesha_util.influ_util), 77
makeBlacknutt() (in module shesha_util.influ_util), 77
makeGaussian() (in module shesha_util.influ_util), 78
makegaussian() (in module shesha_util.utilities), 84
makeRadialSchwartz() (in module shesha_util.influ_util), 78
makeRigaut() (in module shesha_util.influ_util), 78
makeSquareSchwartz() (in module shesha_util.influ_util), 78
montagn() (naga_obj.naga_obj_Double1D method), 19
montagn() (naga_obj.naga_obj_Double2D method), 21
montagn() (naga_obj.naga_obj_Double3D method), 24
montagn() (naga_obj.naga_obj_Double4D method), 25
montagn() (naga_obj.naga_obj_Float1D method), 12
montagn() (naga_obj.naga_obj_Float2D method), 14
montagn() (naga_obj.naga_obj_Float3D method), 16
montagn() (naga_obj.naga_obj_Float4D method), 18

N

naga_context (class in naga_context), 3
naga_host_obj_ComplexD1D (class in naga_host_obj), 47
naga_host_obj_ComplexD2D (class in naga_host_obj), 48
naga_host_obj_ComplexD3D (class in naga_host_obj), 48
naga_host_obj_ComplexD4D (class in naga_host_obj), 49
naga_host_obj_ComplexS1D (class in naga_host_obj), 46
naga_host_obj_ComplexS2D (class in naga_host_obj), 46
naga_host_obj_ComplexS3D (class in naga_host_obj), 47
naga_host_obj_ComplexS4D (class in naga_host_obj), 47
naga_host_obj_Double1D (class in naga_host_obj), 44
naga_host_obj_Double2D (class in naga_host_obj), 45
naga_host_obj_Double3D (class in naga_host_obj), 45
naga_host_obj_Double4D (class in naga_host_obj), 46
naga_host_obj_Float1D (class in naga_host_obj), 43
naga_host_obj_Float2D (class in naga_host_obj), 43
naga_host_obj_Float3D (class in naga_host_obj), 44
naga_host_obj_Float4D (class in naga_host_obj), 44
naga_host_obj_Int1D (class in naga_host_obj), 40
naga_host_obj_Int2D (class in naga_host_obj), 40
naga_host_obj_Int3D (class in naga_host_obj), 41

naga_host_obj_Int4D (class in naga_host_obj), 41
naga_host_obj_UInt1D (class in naga_host_obj), 41
naga_host_obj_UInt2D (class in naga_host_obj), 42
naga_host_obj_UInt3D (class in naga_host_obj), 42
naga_host_obj_UInt4D (class in naga_host_obj), 42
naga_obj_ComplexD1D (class in naga_obj), 33
naga_obj_ComplexD2D (class in naga_obj), 34
naga_obj_ComplexD3D (class in naga_obj), 37
naga_obj_ComplexD4D (class in naga_obj), 38
naga_obj_ComplexS1D (class in naga_obj), 26
naga_obj_ComplexS2D (class in naga_obj), 27
naga_obj_ComplexS3D (class in naga_obj), 30
naga_obj_ComplexS4D (class in naga_obj), 31
naga_obj_Double1D (class in naga_obj), 18
naga_obj_Double2D (class in naga_obj), 20
naga_obj_Double3D (class in naga_obj), 22
naga_obj_Double4D (class in naga_obj), 24
naga_obj_Float1D (class in naga_obj), 10
naga_obj_Float2D (class in naga_obj), 12
naga_obj_Float3D (class in naga_obj), 15
naga_obj_Float4D (class in naga_obj), 16
naga_obj_Int1D (class in naga_obj), 3
naga_obj_Int2D (class in naga_obj), 4
naga_obj_Int3D (class in naga_obj), 5
naga_obj_Int4D (class in naga_obj), 6
naga_obj_UInt1D (class in naga_obj), 7
naga_obj_UInt2D (class in naga_obj), 8
naga_obj_UInt3D (class in naga_obj), 9
naga_obj_UInt4D (class in naga_obj), 9
naga_sparse_obj_Double (class in naga_sparse_obj), 49
naga_sparse_obj_Float (class in naga_sparse_obj), 49
naga_timer (class in naga_timer), 51
next() (shesha_sim.BenchBrama method), 72
next() (shesha_sim.Simulator method), 72
noise_cov() (in module shesha_ao.wfs), 56
nrm2() (naga_obj.naga_obj_Double1D method), 19
nrm2() (naga_obj.naga_obj_Double2D method), 22
nrm2() (naga_obj.naga_obj_Double3D method), 24
nrm2() (naga_obj.naga_obj_Double4D method), 25
nrm2() (naga_obj.naga_obj_Float1D method), 12
nrm2() (naga_obj.naga_obj_Float2D method), 14
nrm2() (naga_obj.naga_obj_Float3D method), 16
nrm2() (naga_obj.naga_obj_Float4D method), 18

O

openLoopSlp() (in module shesha_ao.modopti), 54

P

pad_array() (in module shesha_util.utilities), 85
Param_atmos (class in shesha_config), 56
Param_centroider (class in shesha_config), 57
Param_controller (class in shesha_config), 58
Param_dm (class in shesha_config), 59
Param_geom (class in shesha_config), 63
Param_loop (class in shesha_config), 64
Param_target (class in shesha_config), 64
Param_tel (class in shesha_config), 65
Param_wfs (class in shesha_config), 66
params_dictionary() (in module shesha_util.hdf5_utils), 75
PatternType (class in shesha_constants), 70
pcgeom() (in module shesha_util.kl_util), 81
phase_struct() (in module shesha_util.iterkolmo), 80
piston_orth() (in module shesha_util.kl_util), 81
polang() (in module shesha_util.kl_util), 81
potri_Double() (in module naga_magma), 50
potri_Float() (in module naga_magma), 50
potri_host_Double() (in module naga_magma), 50
potri_host_Float() (in module naga_magma), 50
ProfType (class in shesha_constants), 70
PyrCentroiderMethod (class in shesha_constants), 70

R

radii() (in module shesha_util.kl_util), 81
random() (naga_obj.naga_obj_ComplexD1D method), 34
random() (naga_obj.naga_obj_ComplexD2D method), 36
random() (naga_obj.naga_obj_ComplexD3D method), 38
random() (naga_obj.naga_obj_ComplexD4D method), 39
random() (naga_obj.naga_obj_ComplexS1D method), 27
random() (naga_obj.naga_obj_ComplexS2D method), 29
random() (naga_obj.naga_obj_ComplexS3D method), 31
random() (naga_obj.naga_obj_ComplexS4D method), 32
random() (naga_obj.naga_obj_Double1D method), 19
random() (naga_obj.naga_obj_Double2D method), 22
random() (naga_obj.naga_obj_Double3D method), 24
random() (naga_obj.naga_obj_Double4D method), 25
random() (naga_obj.naga_obj_Float1D method), 12
random() (naga_obj.naga_obj_Float2D method), 14
random() (naga_obj.naga_obj_Float3D method), 16
random() (naga_obj.naga_obj_Float4D method), 18
readHdf5SingleDataset() (in module shesha_util.hdf5_utils), 76
rebin() (in module shesha_util.utilities), 85
reset() (naga_obj.naga_obj_ComplexD1D method), 34
reset() (naga_obj.naga_obj_ComplexD2D method), 36
reset() (naga_obj.naga_obj_ComplexD3D method), 38
reset() (naga_obj.naga_obj_ComplexD4D method), 39
reset() (naga_obj.naga_obj_ComplexS1D method), 27
reset() (naga_obj.naga_obj_ComplexS2D method), 29
reset() (naga_obj.naga_obj_ComplexS3D method), 31
reset() (naga_obj.naga_obj_ComplexS4D method), 32
reset() (naga_obj.naga_obj_Double1D method), 20
reset() (naga_obj.naga_obj_Double2D method), 22
reset() (naga_obj.naga_obj_Double3D method), 24
reset() (naga_obj.naga_obj_Double4D method), 25
reset() (naga_obj.naga_obj_Float1D method), 12
reset() (naga_obj.naga_obj_Float2D method), 14

reset() (naga_obj.naga_obj_Float3D method), 16
reset() (naga_obj.naga_obj_Float4D method), 18
reset() (naga_obj.naga_obj_Int1D method), 4
reset() (naga_obj.naga_obj_Int2D method), 5
reset() (naga_obj.naga_obj_Int3D method), 6
reset() (naga_obj.naga_obj_Int4D method), 7
reset() (naga_obj.naga_obj_UInt1D method), 8
reset() (naga_obj.naga_obj_UInt2D method), 8
reset() (naga_obj.naga_obj_UInt3D method), 9
reset() (naga_obj.naga_obj_UInt4D method), 10
reset() (naga_timer.naga_timer method), 51
rodconan() (in module shesha_util.iterkolmo), 80
rotate() (in module shesha_util.utilities), 85
rotate3d() (in module shesha_util.utilities), 85
rtc_init() (in module shesha_init), 71

S

save_AB_in_database() (in module shesha_util.hdf5_utils), 76
save_dm_geom_in_dataBase() (in module shesha_util.hdf5_utils), 76
save_h5() (in module shesha_util.hdf5_utils), 76
save_hdf5() (in module shesha_util.hdf5_utils), 76
save_imat_in_dataBase() (in module shesha_util.hdf5_utils), 76
scale() (naga_obj.naga_obj_ComplexD1D method), 34
scale() (naga_obj.naga_obj_ComplexD2D method), 36
scale() (naga_obj.naga_obj_ComplexD3D method), 38
scale() (naga_obj.naga_obj_ComplexD4D method), 39
scale() (naga_obj.naga_obj_ComplexS1D method), 27
scale() (naga_obj.naga_obj_ComplexS2D method), 29
scale() (naga_obj.naga_obj_ComplexS3D method), 31
scale() (naga_obj.naga_obj_ComplexS4D method), 32
scale() (naga_obj.naga_obj_Double1D method), 20
scale() (naga_obj.naga_obj_Double2D method), 22
scale() (naga_obj.naga_obj_Double3D method), 24
scale() (naga_obj.naga_obj_Double4D method), 25
scale() (naga_obj.naga_obj_Float1D method), 12
scale() (naga_obj.naga_obj_Float2D method), 14
scale() (naga_obj.naga_obj_Float3D method), 16
scale() (naga_obj.naga_obj_Float4D method), 18
select_actuators() (in module shesha_util.dm_util), 74
selectDMforLayers() (in module shesha_ao.tomo), 55
set_activeDevice() (naga_context.naga_context method), 3
set_activeDeviceForce() (naga_context.naga_context method), 3
set_activeDeviceForCpy() (naga_context.naga_context method), 3
set_alt() (shesha_config.Param_atmos method), 56
set_alt() (shesha_config.Param_dm method), 59
set_altna() (shesha_config.Param_wfs method), 66
set_apod() (shesha_config.Param_geom method), 63
set_apod() (shesha_config.Param_target method), 64
set_apod_file() (shesha_config.Param_geom method), 63
set_apodizer() (shesha_config.Param_geom method), 63
set_atmos_seen() (shesha_config.Param_wfs method), 66
set_azbas() (shesha_config.Param_dm method), 59
set_azimuth() (shesha_config.Param_wfs method), 66
set_beam() (shesha_config.Param_wfs method), 66
set_beamsize() (shesha_config.Param_wfs method), 66
set_binmap() (shesha_config.Param_wfs method), 66
set_cent() (shesha_config.Param_geom method), 63
set_center_name() (shesha_config.Param_dm method), 59
set_cmat() (shesha_config.Param_controller method), 58
set_cobs() (shesha_config.Param_tel method), 65
set_coupling() (shesha_config.Param_dm method), 59
set_cp() (shesha_config.Param_dm method), 60
set_cr() (shesha_config.Param_dm method), 60
set_cube_name() (shesha_config.Param_dm method), 60
set_cured_ndivs() (shesha_config.Param_controller method), 58
set_delay() (shesha_config.Param_controller method), 58
set_deltax() (shesha_config.Param_atmos method), 56
set_deltay() (shesha_config.Param_atmos method), 56
set_devices() (shesha_config.Param_loop method), 64
set_diam() (shesha_config.Param_tel method), 65
set_diam_dm() (shesha_config.Param_dm method), 60
set_diam_dm_proj() (shesha_config.Param_dm method), 60
set_dim_screens() (shesha_config.Param_atmos method), 56
set_dms_seen() (shesha_config.Param_target method), 64
set_dms_seen() (shesha_config.Param_wfs method), 66
set_dx() (shesha_config.Param_wfs method), 66
set_dy() (shesha_config.Param_wfs method), 66
set_error_budget() (shesha_config.Param_wfs method), 67
set_file_influ_hdf5() (shesha_config.Param_dm method), 60
set_fluxPerSub() (shesha_config.Param_wfs method), 67
set_frac() (shesha_config.Param_atmos method), 56
set_fracsub() (shesha_config.Param_wfs method), 67
set_fssize() (shesha_config.Param_wfs method), 67
set_fstop() (shesha_config.Param_wfs method), 67
set_ftbeam() (shesha_config.Param_wfs method), 67
set_ftkernel() (shesha_config.Param_wfs method), 67
set_G() (shesha_config.Param_wfs method), 66
set_gain() (shesha_config.Param_controller method), 58
set_gain() (shesha_config.Param_dm method), 60
set_gmax() (shesha_config.Param_controller method), 58
set_gmin() (shesha_config.Param_controller method), 58
set_gsalt() (shesha_config.Param_wfs method), 67
set_gsmag() (shesha_config.Param_wfs method), 67

set_halfxy() (shesha_config.Param_wfs method), 67
set_hrmap() (shesha_config.Param_wfs method), 67
set_i1() (shesha_config.Param_dm method), 60
set_imat() (shesha_config.Param_controller method), 58
set_influ() (shesha_config.Param_dm method), 60
set_influ_res() (shesha_config.Param_dm method), 60
set_influpos() (shesha_config.Param_dm method), 60
set_influsize() (shesha_config.Param_dm method), 60
set_influstart() (shesha_config.Param_dm method), 60
set_influType() (shesha_config.Param_dm method), 60
set_interpmat() (shesha_config.Param_centroider method), 57
set_ipupil() (shesha_config.Param_geom method), 63
set_is_init() (shesha_config.Param_geom method), 63
set_istart() (shesha_config.Param_wfs method), 67
set_isvalid() (shesha_config.Param_wfs method), 67
set_ittime() (shesha_config.Param_loop method), 64
set_j1() (shesha_config.Param_dm method), 61
set_jstart() (shesha_config.Param_wfs method), 67
set_kernel() (shesha_config.Param_wfs method), 67
set_kl_imat() (shesha_config.Param_controller method), 58
set_klgain() (shesha_config.Param_controller method), 58
set_L0() (shesha_config.Param_atmos method), 56
set_Lambda() (shesha_config.Param_target method), 64
set_Lambda() (shesha_config.Param_wfs method), 66
set_laserpower() (shesha_config.Param_wfs method), 67
set_lgskern() (shesha_config.Param_wfs method), 67
set_lgsreturnperwatt() (shesha_config.Param_wfs method), 67
set_lltx() (shesha_config.Param_wfs method), 68
set_llty() (shesha_config.Param_wfs method), 68
set_mag() (shesha_config.Param_target method), 64
set_margin_in() (shesha_config.Param_dm method), 61
set_margin_out() (shesha_config.Param_dm method), 61
set_maxcond() (shesha_config.Param_controller method), 58
set_method() (shesha_config.Param_centroider method), 57
set_modopti() (shesha_config.Param_controller method), 58
set_mpupil() (shesha_config.Param_geom method), 63
set_n() (shesha_config.Param_geom method), 63
set_n1() (shesha_config.Param_dm method), 61
set_n1() (shesha_config.Param_geom method), 63
set_n2() (shesha_config.Param_dm method), 61
set_n2() (shesha_config.Param_geom method), 63
set_nact() (shesha_config.Param_dm method), 61
set_nactu() (shesha_config.Param_controller method), 59
set_nbrmissing() (shesha_config.Param_tel method), 65
set_ncp() (shesha_config.Param_dm method), 61
set_ndm() (shesha_config.Param_controller method), 59
set_Nfft() (shesha_config.Param_wfs method), 66
set_ngain() (shesha_config.Param_controller method), 59
set_ninflu() (shesha_config.Param_dm method), 61
set_niter() (shesha_config.Param_loop method), 64
set_nkl() (shesha_config.Param_controller method), 59
set_nkl() (shesha_config.Param_dm method), 61
set_nmax() (shesha_config.Param_centroider method), 57
set_nmodes() (shesha_config.Param_controller method), 59
set_noise() (shesha_config.Param_wfs method), 68
set_nphotons() (shesha_config.Param_wfs method), 68
set_nphotons4imat() (shesha_config.Param_wfs method), 68
set_npix() (shesha_config.Param_wfs method), 68
set_npp() (shesha_config.Param_dm method), 61
set_nr() (shesha_config.Param_dm method), 61
set_nrebin() (shesha_config.Param_wfs method), 68
set_nrec() (shesha_config.Param_controller method), 59
set_nscreens() (shesha_config.Param_atmos method), 56
set_ntargets() (shesha_config.Param_target method), 64
set_Ntot() (shesha_config.Param_wfs method), 66
set_ntotact() (shesha_config.Param_dm method), 61
set_nvalid() (shesha_config.Param_controller method), 59
set_nvalid() (shesha_config.Param_wfs method), 68
set_nwfs() (shesha_config.Param_centroider method), 57
set_nwfs() (shesha_config.Param_controller method), 59
set_nxsub() (shesha_config.Param_wfs method), 68
set_openloop() (shesha_config.Param_wfs method), 68
set_optthroughput() (shesha_config.Param_wfs method), 68
set_ord() (shesha_config.Param_dm method), 61
set_outscl() (shesha_config.Param_dm method), 61
set_p1() (shesha_config.Param_geom method), 63
set_p2() (shesha_config.Param_geom method), 63
set_pctr() (in module shesha_util.kl_util), 82
set_pdiam() (shesha_config.Param_wfs method), 68
set_phase_ab_M1() (shesha_config.Param_geom method), 63
set_phase_ab_M1_m() (shesha_config.Param_geom method), 64
set_phasemap() (shesha_config.Param_wfs method), 68
set_pitch() (shesha_config.Param_dm method), 61
set_pixsize() (shesha_config.Param_wfs method), 68
set_prof1d() (shesha_config.Param_wfs method), 68
set_profcum() (shesha_config.Param_wfs method), 68
set_profna() (shesha_config.Param_wfs method), 69
set_proftype() (shesha_config.Param_wfs method), 69
set_pupangle() (shesha_config.Param_tel method), 65

set_pupdiam() (shesha_config.Param_geom method), 64
set_pupixsize() (shesha_config.Param_atmos method), 57
set_pupoffset() (shesha_config.Param_dm method), 62
set_puppixoffset() (shesha_config.Param_dm method), 62
set_push4imat() (shesha_config.Param_dm method), 62
set_pyr_ampl() (shesha_config.Param_wfs method), 69
set_pyr_cx() (shesha_config.Param_wfs method), 69
set_pyr_cy() (shesha_config.Param_wfs method), 69
set_pyr_loc() (shesha_config.Param_wfs method), 69
set_pyr_npts() (shesha_config.Param_wfs method), 69
set_pyr_pos() (shesha_config.Param_wfs method), 69
set_pyr_pup_sep() (shesha_config.Param_wfs method), 69
set_pyrtype() (shesha_config.Param_wfs method), 69
set_pzt_extent() (shesha_config.Param_dm method), 62
set_qpixsize() (shesha_config.Param_wfs method), 69
set_r0() (shesha_config.Param_atmos method), 57
set_rabas() (shesha_config.Param_dm method), 62
set_referr() (shesha_config.Param_tel method), 65
set_seeds() (shesha_config.Param_atmos method), 57
set_sincar() (shesha_config.Param_wfs method), 69
set_sizex() (shesha_config.Param_centroider method), 57
set_sizey() (shesha_config.Param_centroider method), 57
set_spiders_type() (shesha_config.Param_tel method), 65
set_spupil() (shesha_config.Param_geom method), 64
set_ssize() (shesha_config.Param_geom method), 64
set_std_piston() (shesha_config.Param_tel method), 65
set_std_tt() (shesha_config.Param_tel method), 65
set_subapd() (shesha_config.Param_wfs method), 69
set_submask() (shesha_config.Param_wfs method), 69
set_t_spiders() (shesha_config.Param_tel method), 65
set_thetaML() (shesha_config.Param_wfs method), 69
set_thresh() (shesha_config.Param_centroider method), 57
set_thresh() (shesha_config.Param_dm method), 62
set_TTcond() (shesha_config.Param_controller method), 58
set_type() (shesha_config.Param_centroider method), 57
set_type() (shesha_config.Param_controller method), 59
set_type() (shesha_config.Param_dm method), 62
set_type() (shesha_config.Param_wfs method), 70
set_type_ap() (shesha_config.Param_tel method), 65
set_type_fct() (shesha_config.Param_centroider method), 57
set_type_kl() (shesha_config.Param_dm method), 62
set_type_pattern() (shesha_config.Param_dm method), 62
set_unitpervolt() (shesha_config.Param_dm method), 62
set_validsubsx() (shesha_config.Param_wfs method), 70
set_validsubsy() (shesha_config.Param_wfs method), 70
set_vect_seg() (shesha_config.Param_tel method), 65
set_weights() (shesha_config.Param_centroider method), 58
set_width() (shesha_config.Param_centroider method), 58
set_winddir() (shesha_config.Param_atmos method), 57
set_windspeed() (shesha_config.Param_atmos method), 57
set_x_name() (shesha_config.Param_dm method), 62
set_xpos() (shesha_config.Param_dm method), 62
set_xpos() (shesha_config.Param_target method), 65
set_xpos() (shesha_config.Param_wfs method), 70
set_y_name() (shesha_config.Param_dm method), 62
set_ypos() (shesha_config.Param_dm method), 62
set_ypos() (shesha_config.Param_target method), 65
set_ypos() (shesha_config.Param_wfs method), 70
set_zenithangle() (shesha_config.Param_geom method), 64
set_zerop() (shesha_config.Param_target method), 65
set_zerop() (shesha_config.Param_wfs method), 70
setData() (naga_host_obj.naga_host_obj_ComplexD1D method), 48
setData() (naga_host_obj.naga_host_obj_ComplexD2D method), 48
setData() (naga_host_obj.naga_host_obj_ComplexD3D method), 49
setData() (naga_host_obj.naga_host_obj_ComplexD4D method), 49
setData() (naga_host_obj.naga_host_obj_ComplexS1D method), 46
setData() (naga_host_obj.naga_host_obj_ComplexS2D method), 47
setData() (naga_host_obj.naga_host_obj_ComplexS3D method), 47
setData() (naga_host_obj.naga_host_obj_ComplexS4D method), 47
setData() (naga_host_obj.naga_host_obj_Double1D method), 45
setData() (naga_host_obj.naga_host_obj_Double2D method), 45
setData() (naga_host_obj.naga_host_obj_Double3D method), 45
setData() (naga_host_obj.naga_host_obj_Double4D method), 46
setData() (naga_host_obj.naga_host_obj_Float1D method), 43
setData() (naga_host_obj.naga_host_obj_Float2D method), 44
setData() (naga_host_obj.naga_host_obj_Float3D method), 44
setData() (naga_host_obj.naga_host_obj_Float4D method), 44

setData() (naga_host_obj.naga_host_obj_Int1D method), 40
setData() (naga_host_obj.naga_host_obj_Int2D method), 40
setData() (naga_host_obj.naga_host_obj_Int3D method), 41
setData() (naga_host_obj.naga_host_obj_Int4D method), 41
setData() (naga_host_obj.naga_host_obj_UInt1D method), 42
setData() (naga_host_obj.naga_host_obj_UInt2D method), 42
setData() (naga_host_obj.naga_host_obj_UInt3D method), 42
setData() (naga_host_obj.naga_host_obj_UInt4D method), 43
setpincs() (in module shesha_util.kl_util), 82
shesha_ao (module), 51
shesha_ao.basis (module), 51
shesha_ao.cmats (module), 53
shesha_ao.imats (module), 53
shesha_ao.modopti (module), 54
shesha_ao.tomo (module), 54
shesha_ao.wfs (module), 55
shesha_config (module), 56
shesha_constants (module), 70
shesha_init (module), 71
shesha_sim (module), 72
shesha_util (module), 73
shesha_util.dm_util (module), 73
shesha_util.hdf5_utils (module), 74
shesha_util.influ_util (module), 77
shesha_util.iterkolmo (module), 79
shesha_util.kl_util (module), 80
shesha_util.make_apodizer (module), 83
shesha_util.make_pupil (module), 83
shesha_util.rtc_util (module), 84
shesha_util.utilities (module), 84
Simulator (class in shesha_sim), 72
SimulatorBrama (class in shesha_sim), 73
SpiderType (class in shesha_constants), 71
start() (naga_timer.naga_timer method), 51
stencil_size() (in module shesha_util.iterkolmo), 80
stencil_size_array() (in module shesha_util.iterkolmo), 80
stop() (naga_timer.naga_timer method), 51
sum() (naga_obj.naga_obj_Double1D method), 20
sum() (naga_obj.naga_obj_Double2D method), 22
sum() (naga_obj.naga_obj_Double3D method), 24
sum() (naga_obj.naga_obj_Double4D method), 25
sum() (naga_obj.naga_obj_Float1D method), 12
sum() (naga_obj.naga_obj_Float2D method), 14
sum() (naga_obj.naga_obj_Float3D method), 16
sum() (naga_obj.naga_obj_Float4D method), 18
svd_Double() (in module naga_magma), 50
svd_Float() (in module naga_magma), 50
svd_host_Double() (in module naga_magma), 50
svd_host_Float() (in module naga_magma), 50
swap() (naga_obj.naga_obj_ComplexD1D method), 34
swap() (naga_obj.naga_obj_ComplexD2D method), 36
swap() (naga_obj.naga_obj_ComplexD3D method), 38
swap() (naga_obj.naga_obj_ComplexD4D method), 40
swap() (naga_obj.naga_obj_ComplexS1D method), 27
swap() (naga_obj.naga_obj_ComplexS2D method), 29
swap() (naga_obj.naga_obj_ComplexS3D method), 31
swap() (naga_obj.naga_obj_ComplexS4D method), 33
swap() (naga_obj.naga_obj_Double1D method), 20
swap() (naga_obj.naga_obj_Double2D method), 22
swap() (naga_obj.naga_obj_Double3D method), 24
swap() (naga_obj.naga_obj_Double4D method), 26
swap() (naga_obj.naga_obj_Float1D method), 12
swap() (naga_obj.naga_obj_Float2D method), 14
swap() (naga_obj.naga_obj_Float3D method), 16
swap() (naga_obj.naga_obj_Float4D method), 18
syevd_Double() (in module naga_magma), 51
syevd_Float() (in module naga_magma), 51
syevd_host_Double() (in module naga_magma), 51
syevd_host_Float() (in module naga_magma), 51
symm() (naga_obj.naga_obj_ComplexD2D method), 36
symm() (naga_obj.naga_obj_ComplexS2D method), 29
symm() (naga_obj.naga_obj_Double2D method), 22
symm() (naga_obj.naga_obj_Float2D method), 14
symv() (naga_obj.naga_obj_ComplexD2D method), 36
symv() (naga_obj.naga_obj_ComplexS2D method), 29
symv() (naga_obj.naga_obj_Double2D method), 22
symv() (naga_obj.naga_obj_Float2D method), 14
syrk() (naga_obj.naga_obj_ComplexD2D method), 36
syrk() (naga_obj.naga_obj_ComplexS2D method), 29
syrk() (naga_obj.naga_obj_Double2D method), 22
syrk() (naga_obj.naga_obj_Float2D method), 14
syrkx() (naga_obj.naga_obj_ComplexD2D method), 37
syrkx() (naga_obj.naga_obj_ComplexS2D method), 30
syrkx() (naga_obj.naga_obj_Double2D method), 22
syrkx() (naga_obj.naga_obj_Float2D method), 14

T

target_init() (in module shesha_init), 71
TargetImageType (class in shesha_constants), 71
threadSync() (in module naga_timer), 51
transpose() (naga_obj.naga_obj_ComplexD2D method), 37
transpose() (naga_obj.naga_obj_ComplexS2D method), 30
transpose() (naga_obj.naga_obj_Double2D method), 22
transpose() (naga_obj.naga_obj_Float2D method), 15

U

updateDataBase() (in module shesha_util.hdf5_utils), 76

V

validDataBase() (in module shesha_util.hdf5_utils), 77
validFile() (in module shesha_util.hdf5_utils), 77
validInStore() (in module shesha_util.hdf5_utils), 77

W

wfs_init() (in module shesha_init), 71
WFSType (class in shesha_constants), 71
writeHdf5SingleDataset() (in module shesha_util.hdf5_utils), 77

Z

zernumero() (in module shesha_util.dm_util), 74