aotools Documentation

Release 0.1.0

Andrew Reeves

Apr 20, 2017

Contents

Contents:

CHAPTER 1

Phase Screen Creation

Submodules

Phase Screen Creation

aotools.turbulence.phasescreen.**ft_phase_screen**(*r0*, *N*, *delta*, *L0*, *l0*, *FFT=None*)

Creates a random phase screen with Von Karmen statistics. (Schmidt 2010)

Parameters

- **r0** (*float*) – r0 parameter of scrn in metres
- **N** (*int*) – Size of phase scrn in pxls
- **delta** (*float*) – size in Metres of each pxl
- **L0** (*float*) – Size of outer-scale in metres
- **l0** (*float*) – inner scale in metres

Returns numpy array representing phase screen

Return type ndarray

aotools.turbulence.phasescreen.**ft_sh_phase_screen**(*r0*, *N*, *delta*, *L0*, *l0*, *FFT=None*)

Creates a random phase screen with Von Karmen statistics with added sub-harmonics to augment tip-tilt modes. (Schmidt 2010)

Parameters

- **r0** (*float*) – r0 parameter of scrn in metres
- **N** (*int*) – Size of phase scrn in pxls
- **delta** (*float*) – size in Metres of each pxl
- **L0** (*float*) – Size of outer-scale in metres
- **l0** (*float*) – inner scale in metres

Returns numpy array representing phase screen

Return type ndarray

aotools.turbulence.phasescreen.**ift2**(*G*, *delta_f*, *FFT=None*)

Wrapper for inverse fourier transform

Parameters

- **G** – data to transform
- **delta_f** – pixel seperation
- **FFT** (*FFT object, optional*) – An accelerated FFT object

An implementation of the "infinite phase screen", as deduced by Francois Assemat and Richard W. Wilson, 2006.

class aotools.turbulence.infinitephasescreen.**PhaseScreen**(*nSize*, *pxlScale*, *r0*, *L0*, *nCol=2*, *random_seed=None*)

Bases: object

A "Phase Screen" for use in AO simulation.

This represents the phase addition light experiences when passing through atmospheric turbulence. Unlike other phase screen generation techniques that translate a large static screen, this method keeps a small section of phase, and extends it as neccessary for as many steps as required. This can significantly reduce memory consuption at the expense of more processing power required.

The technique is described in a paper by Assemat and Wilson, 2006. It essentially assumes that there are two matrices, "A" and "B", that can be used to extend an existing phase screen. A single row or column of new phase can be represented by

X = A.Z + B.b

where X is the new phase vector, Z is some number of columns of the existing screen, and b is a random vector with gaussian statistics.

This object calculates the A and B matrices using an expression of the phase covariance when it is initialised. Calculating A is straightforward through the relationship:

A = Cov_xz . (Cov_zz)^(-1).

B is less trivial.

BB^t = Cov_xx - A.Cov_zx

(where B^t is the transpose of B) is a symmetric matrix, hence B can be expressed as

B = UL,

where U and L are obtained from the svd for BB^t

U, w, U^t = svd(BB^t)

L is a diagonal matrix where the diagonal elements are w^(1/2).

On initialisation an initial phase screen is calculated using an FFT based method. When 'addRows' is called, a new vector of phase is added to the phase screen using *nCols* columns of previous phase. Assemat & Wilson claim that two columns are adequate for good atmospheric statistics. The phase in the screen data is always accessed as *<phasescreen>.scrn*.

Parameters

- **nSize** (*int*) – Size of phase screen (NxN)
- **pxlScale** (*float*) – Size of each phase pixel in metres

- **r0** (*float*) – fried parameter (metres)
- **L0** (*float*) – Outer scale (metres)
- **nCol** (*int, optional*) – Number of columns to use to continue screen, default is 2

addRow(*nRows=1*, *axis=0*)
Adds new rows to the phase screen and removes old ones.

Parameters

- **nRows** (*int*) – Number of rows to add
- **axis** (*int*) – Axis to add new rows (can be 0 (default) or 1)

makeAMatrix()
Calculates the "A" matrix, that uses the existing data to find a new component of the new phase vector. This is for propagating in axis 0.

makeBMatrix()
Calculates the "B" matrix, that turns a random vector into a component of the new phase.

Finds a B matrix for the case of generating new data on each side of the phase screen.

makeInitialScrn()
Makes the initial screen usign FFT method that can be extended

makeNewPhase(*nRows*, *axis=0*)
Makes new rows or columns of phase.

Parameters

- **nRows** (*int*) – Number of rows to add (can be positive or negative)
- **axis** (*int*) – Axis to add new rows (can be 0 (default) or 1)

makeSingleBMatrix(*cov_xx*, *cov_zx*, *A_mat*)
Makes the B matrix for a single direction

Parameters

- **cov_xx** – Matrix of XX covariance
- **cov_zx** – Matrix of ZX covariance
- **A_mat** – Corresponding A matrix

makeXXCovMatrix()
Uses the seperation between the new phase points to calculate the theoretical covariance between them

makeXXSeperation()
Calculates a matrix where each element is the seperation in metres between points in the new phase data points.

Returns Array of seperations

Return type ndarray

makeXZCovMat()
Uses the seperation between new and existing phase points to calculate the theoretical covariance between them

makeXZSeperation()
Calculates a matrix where each element is the seperation in metres between points in the new phase data and the existing data.

Returns Array of seperations

Return type ndarray

makeZXCovMatrix()
Uses the seperation between the existing and new phase points to calculate the theoretical covariance between them

makeZXSeperation()
Calculates a matrix where each element is the seperation in metres between points in the existing phase data and the new data.

Returns Array of seperations

Return type ndarray

makeZZCovMat()
Uses the seperation between the existing phase points to calculate the theoretical covariance between them

makeZZSeperation()
Calculates a matrix where each element is the seperation in metres between points in the the existing data.

Returns Array of seperations

Return type ndarray

moveScrn(*translation*)
Translates the phase screen a given distance in metres. Interpolates if required.

Parameters **translation** (*tuple*) – Distance to translate screen in axis 0 and 1, in metres.

aotools.turbulence.infinitephasescreen.**phaseCovariance**(*r*, *r0*, *L0*)
Calculate the phase covariance between two points seperated by *r*, in turbulence with a given *r0 and 'L0*. Uses equation 5 from Assemat and Wilson, 2006.

Parameters

- **r** (*float, ndarray*) – Seperation between points in metres (can be ndarray)
- **r0** (*float*) – Fried parameter of turbulence in metres
- **L0** (*float*) – Outer scale of turbulence in metres

CHAPTER 2

Centroiders

Centroider module

aotools.image_processing.centroiders.**brightestPxl**(*img*, *threshold*, ***kwargs*)

Centroids using brightest Pixel Algorithm (A. G. Basden et al, MNRAS, 2011)

Finds the nPxlsth brightest pixel, subtracts that value from frame, sets anything below 0 to 0, and finally takes centroid.

Parameters

- **img** (*ndarray*) – 2d or greater rank array of imgs to centroid
- **threshold** (*float*) – Percentage of pixels to use for centroid

Returns Array of centroid values

Return type ndarray

aotools.image_processing.centroiders.**centreOfGravity**(*img*, *threshold=0*, ***kwargs*)

Centroids an image, or an array of images. Centroids over the last 2 dimensions. Sets all values under "threshold*max_value" to zero before centroiding

Parameters

- **img** (*ndarray*) – 2d or greater rank array of imgs to centroid
- **threshold** (*float*) – Percentage of max value under which pixels set to 0

Returns Array of centroid values

Return type ndarray

aotools.image_processing.centroiders.**corrConvolve**(*x*, *y*)

2D convolution using FFT, use to generate cross-correlations.

Parameters

- **x** (*array*) – subap image

- **y** (*array*) – reference image

Returns cross-correlation of x and y

Return type ndarray

aotools.image_processing.centroiders.**correlation**(*im*, *threshold*, *ref*)

Correlation Centroider, currently only works for 3d im shape. Performs a simple thresholded COM on the correlation.

Parameters

- **im** – subap images (t, y, x)
- **threshold_fac** – threshold for COM (0=all pixels, 1=brightest pixel)
- **ref** – reference image (t, y, x)

Returns centroids of im, given as x, y

Return type ndarray

aotools.image_processing.centroiders.**quadCell**(*img*, ***kwargs*)

Centroider to be used for 2x2 images.

Parameters **img** – 2d or greater rank array, where centroiding performed over last 2 dimensions

Returns Array of centroid values

Return type ndarray

CHAPTER 3

Normalised Fourier Transforms

aotools.fouriertransform module

Module containing useful FFT based function and classes

aotools.fouriertransform.**ft**(*data*, *delta*)
A properly scaled 1-D FFT

Parameters

- **data** (*ndarray*) – An array on which to perform the FFT
- **delta** (*float*) – Spacing between elements

Returns scaled FFT

Return type ndarray

aotools.fouriertransform.**ft2**(*data*, *delta*)
A properly scaled 2-D FFT

Parameters

- **data** (*ndarray*) – An array on which to perform the FFT
- **delta** (*float*) – Spacing between elements

Returns scaled FFT

Return type ndarray

aotools.fouriertransform.**ift**(*DATA*, *delta_f*)
Scaled inverse 1-D FFT

Parameters

- **DATA** (*ndarray*) – Data in Fourier Space to transform
- **delta_f** (*ndarray*) – Frequency spacing of grid

Returns Scaled data in real space

Return type ndarray

aotools.fouriertransform.**ift2**(*DATA*, *delta_f*)
Scaled inverse 2-D FFT

Parameters

- **DATA** (*ndarray*) – Data in Fourier Space to transform
- **delta_f** (*ndarray*) – Frequency spacing of grid

Returns Scaled data in real space

Return type ndarray

aotools.fouriertransform.**irft**(*DATA*, *delta_f*)
Scaled real inverse 1-D FFT

Parameters

- **DATA** (*ndarray*) – Data in Fourier Space to transform
- **delta_f** (*ndarray*) – Frequency spacing of grid

Returns Scaled data in real space

Return type ndarray

aotools.fouriertransform.**irft2**(*DATA*, *delta_f*)
Scaled inverse real 2-D FFT

Parameters

- **DATA** (*ndarray*) – Data in Fourier Space to transform
- **delta_f** (*ndarray*) – Frequency spacing of grid

Returns Scaled data in real space

Return type ndarray

aotools.fouriertransform.**rft**(*data*, *delta*)
A properly scaled real 1-D FFT

Parameters

- **data** (*ndarray*) – An array on which to perform the FFT
- **delta** (*float*) – Spacing between elements

Returns scaled FFT

Return type ndarray

aotools.fouriertransform.**rft2**(*data*, *delta*)
A properly scaled, real 2-D FFT

Parameters

- **data** (*ndarray*) – An array on which to perform the FFT
- **delta** (*float*) – Spacing between elements

Returns scaled FFT

Return type ndarray

CHAPTER 4

Wavefront Sensor Analysis

WFS module

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

Q

R