PySPOD: Python SPOD Documentation

Release 2.0.0

Gianmarco Mengaldo, Marcin Rogowski, Lisandro Dalcin, Romit M

May 05, 2024

CONTENTS:

- The **GitHub repository** of this package can be found at PySPOD along installation instructions, and how to get started.
- **Tutorials** can be found at PySPOD-Tutorials
- The package uses GitHub Actions for **continuous integration**.

CHAPTER

ONE

INDICES AND TABLE

- genindex
- modindex

CHAPTER

TWO

SPOD MODULE

2.1 SPOD base

Base module for the SPOD:

- The *Base.fit* method must be implemented in inherited classes

class Base(*params*, *weights=None*, *comm=None*)

Spectral Proper Orthogonal Decomposition base class.

property comm

Get the MPI communicator.

Returns
the MPI communicator.

Return type
mpi4py.MPI.Intracomm

compute_coeffs_op(*data*, *results_dir*, *modes_idx=None*, *tol=1e-10*, *svd=True*, *T_lb=None*, *T_ub=None*)

See method implementation in the spod.utils module.

compute_reconstruction(*coeffs_dir*, *time_idx=None*)

See method implementation in the spod.utils module.

define_weights()

Define and check weights.

property dim

Get the number of dimensions of the data matrix.

Returns
number of dimensions of the data matrix.

Return type
int

property dt

Get the time-step.

Returns
the time-step used by the SPOD algorithm.

Return type
double

property eigs

Get the eigenvalues of the SPOD matrix.

Returns
the eigenvalues from the eigendecomposition the SPOD matrix.

Return type
numpy.ndarray

property file_coeffs

Get the file path where coeffs are saved.

Returns
path to file where coeffs are saved.

Return type
str

property file_dynamics

Get the file path where reconstruction is saved.

Returns
path to file where reconstruction is saved.

Return type
str

find_nearest_coords(*coords*, *x*)

See method implementation in the postproc module.

find_nearest_freq(*freq_req*, *freq=None*)

See method implementation in the postproc module.

fit(*data_list*, **args*, ***kwargs*)

Fit the data using SPOD.

Parameters
data_list (*list*) – list containing data matrices for which to compute the SPOD.

property freq

Get the number of modes.

Returns
the number of modes computed by the SPOD algorithm.

Return type
int

property freq_idx_lb

Get the number of frequencies.

Returns
the number of frequencies computed by the SPOD algorithm.

Return type
int

property freq_idx_ub

Get the number of frequencies.

Returns
the number of frequencies computed by the SPOD algorithm.

Return type
int

generate_2d_data_video(*data*, *time_limits=[0, 10]*, *vars_idx=[0]*, *sampling=1*, *x1=None*, *x2=None*, *coastlines=''*, *figsize=(12, 8)*, *filename='data_video.mp4'*)

See method implementation in the postproc module.

get_data(*data*, *t_0=None*, *t_end=None*)

Get the original input data.

Returns
the matrix that contains the original snapshots.

Return type
numpy.ndarray

get_freq_axis()

Obtain frequency axis.

get_modes_at_freq(*freq_idx*)

See method implementation in the postproc module.

long_t_mean(*data*)

Get longtime mean.

property modes_dir

Get the dictionary containing the path to the SPOD modes saved.

Returns
the dictionary containing the path to the SPOD modes saved.

Return type
dict

property n_blocks

Get the number of blocks used.

Returns
the number of blocks used by the SPOD algorithms.

Return type
int

property n_dft

Get the number of DFT per block.

Returns
the number of DFT per block.

Return type
int

property n_freq

Get the number of frequencies.

Returns
the number of frequencies computed by the SPOD algorithm.

Return type
int

property n_modes

Get the number of modes.

Returns
the number of modes computed by the SPOD algorithm.

Return type
int

property n_modes_save

Get the number of modes.

Returns
the number of modes computed by the SPOD algorithm.

Return type
int

property nt

Get the number of time-steps of the data matrix.

Returns
the number of time-steps of the data matrix.

Return type
int

property nv

Get the number of variables of the data matrix.

Returns
the number of variables of the data matrix.

Return type
int

property nx

Get the number of spatial points of the data matrix.

Returns
the number of spatial points [dim1:] of the data matrix.

Return type
int

plot_2d_data(*data*, *time_idx=[0]*, *vars_idx=[0]*, *x1=None*, *x2=None*, *title=''*, *coastlines=''*, *figsize=(12, 8)*, *filename=None*, *origin=None*)

See method implementation in the postproc module.

plot_2d_mode_slice_vs_time(*freq_req*, *freq*, *vars_idx=[0]*, *modes_idx=[0]*, *x1=None*, *x2=None*, *max_each_mode=False*, *fftshift=False*, *title=''*, *figsize=(12, 8)*, *equal_axes=False*, *filename=None*)

See method implementation in the postproc module.

plot_2d_modes_at_frequency(*freq_req*, *freq*, *vars_idx=[0]*, *modes_idx=[0]*, *x1=None*, *x2=None*, *fftshift=False*, *imaginary=False*, *plot_max=False*, *coastlines=''*, *title=''*, *xticks=None*, *yticks=None*, *figsize=(12, 8)*, *equal_axes=False*, *filename=None*, *origin=None*, *pdf=None*, *shift180=False*)

See method implementation in the postproc module.

plot_3d_modes_slice_at_frequency(*freq_req*, *freq*, *vars_idx=[0]*, *modes_idx=[0]*, *x1=None*, *x2=None*, *x3=None*, *slice_dim=0*, *slice_id=None*, *fftshift=False*, *imaginary=False*, *plot_max=False*, *coastlines=''*, *title=''*, *xticks=None*, *yticks=None*, *figsize=(12, 8)*, *equal_axes=False*, *filename=None*, *origin=None*)

See method implementation in the postproc module.

plot_data_tracers(*data*, *coords_list*, *x=None*, *time_limits=[0, 10]*, *vars_idx=[0]*, *title=''*, *figsize=(12, 8)*, *filename=None*)

See method implementation in the postproc module.

plot_eigs(*title=''*, *figsize=(12, 8)*, *show_axes=True*, *equal_axes=False*, *filename=None*)

See method implementation in the postproc module.

plot_eigs_vs_frequency(*freq=None*, *title=''*, *xticks=None*, *yticks=None*, *show_axes=True*, *equal_axes=False*, *figsize=(12, 8)*, *filename=None*)

See method implementation in the postproc module.

plot_eigs_vs_period(*freq=None*, *title=''*, *xticks=None*, *yticks=None*, *show_axes=True*, *equal_axes=False*, *figsize=(12, 8)*, *filename=None*)

See method implementation in the postproc module.

plot_mode_tracers(*freq_req*, *freq*, *coords_list*, *x=None*, *vars_idx=[0]*, *modes_idx=[0]*, *fftshift=False*, *title=''*, *figsize=(12, 8)*, *filename=None*)

See method implementation in the postproc module.

property savedir_sim

Get the directory where results are saved.

Returns
path to directory where results are saved.

Return type
str

select_mean(*data*)

Select mean.

property shape

Get the shape of the data matrix.

Returns
shape of the data matrix.

Return type
int

property weights

Get the weights used to compute the inner product.

Returns
weight matrix used to compute the inner product.

Return type
numpy.ndarray

property xdim

Get the number of spatial dimensions of the data matrix.

Returns
number of spatial dimensions of the data matrix.

Return type
tuple(int,)

property xshape

Get the spatial shape of the data matrix.

Returns
spatial shape of the data matrix.

Return type
tuple(int,)

2.2 SPOD standard

Derived module from spod_base.py for standard SPOD.

class Standard(*params*, *weights=None*, *comm=None*)

Class that implements a distributed batch version of the Spectral Proper Orthogonal Decomposition algorithm to the input data.

The computation is performed on the *data* passed to the *fit* method of the *Standard* class, derived from the *Base* class.

fit(*data_list*, *variables=None*)

Class-specific method to fit the data matrix using the SPOD batch algorithm.

Parameters
data_list (*list*) – list containing data matrices for which to compute the SPOD.

2.3 SPOD streaming

Derived module from spod_base.py for streaming SPOD.

class Streaming(*params*, *weights=None*, *comm=None*)

Class that implements a distributed streaming version of the Spectral Proper Orthogonal Decomposition algorithm to the input data.

The computation is performed on the data passed to the *fit* method of the *Streaming* class, derived from the *Base* class.

fit(*data_list*)

Class-specific method to fit the data matrix using the SPOD streaming algorithm.

Parameters
data_list (*list*) – list containing data matrices for which to compute the SPOD.

2.4 SPOD utils

Utils for SPOD method.

check_orthogonality(*results_dir*, *mode_idx1*, *mode_idx2*, *freq_idx*, *dtype='double'*, *savedir=None*, *comm=None*)

Check orthogonality of SPOD modes.

Parameters

- **results_dir** (*str*) – path to results folder where to find SPOD modes.
- **mode_idx1** (*int*) – id first mode used for comparison.
- **mode_idx2** (*int*) – id second mode used for comparison.
- **freq_idx** (*int*) – frequency id to be used.
- **dtype** (*str*) – datatype to be used. Default is 'double'.
- **savedir** (*str*) – path where to save the data.
- **comm** (*MPI.Comm*) – MPI communicator.

Returns
orthogonality check and value.

Return type
bool, float

compute_coeffs_conv(*data*, *results_dir*, *modes_idx=None*, *freq_idx=None*, *T_lb=None*, *T_ub=None*, *tol=1e-10*, *svd=False*, *savedir=None*, *dtype='double'*, *comm=None*)

Continuously-discrete temporal expansion coefficients of SPOD modes via convolution.

Parameters

- **data** (*numpy.ndarray*) – data.
- **results_dir** (*str*) – path to results folder.
- **mode_idx** (*list*) – ids modes used for building coefficients. Default is None.
- **freq_idx** (*list*) – frequency ids to be used. Default is None.
- **T_lb** (*float*) – lower bound period. Default is None.
- **T_ub** (*float*) – upper bound period. Default is None.
- **tol** (*float*) – tolerance for pseudoinverse. Default is 1e-10.
- **svd** (*float*) – whether to use svd pseudoinverse. Default is None.
- **savedir** (*str*) – path where to save the data. Default is None.
- **dtype** (*str*) – datatype to be used. Default is 'double'.
- **comm** (*MPI.Comm*) – MPI communicator. Default is None.

Returns
where the file with the coefficients is saved, and associated folder.

Return type
str, str

compute_coeffs_op(*data*, *results_dir*, *modes_idx=None*, *freq_idx=None*, *T_lb=None*, *T_ub=None*, *tol=1e-10*, *svd=False*, *savedir=None*, *dtype='double'*, *comm=None*)

Compute coefficients through oblique projection.

Parameters

- **data** (*numpy.ndarray*) – data.
- **results_dir** (*str*) – path to results folder.
- **mode_idx** (*list*) – ids modes used for building coefficients. Default is None.
- **freq_idx** (*list*) – frequency ids to be used. Default is None.
- **T_lb** (*float*) – lower bound period. Default is None.
- **T_ub** (*float*) – upper bound period. Default is None.
- **tol** (*float*) – tolerance for pseudoinverse. Default is 1e-10.
- **svd** (*float*) – whether to use svd pseudoinverse. Default is None.
- **savedir** (*str*) – path where to save the data. Default is None.
- **dtype** (*str*) – datatype to be used. Default is 'double'.
- **comm** (*MPI.Comm*) – MPI communicator. Default is None.

Returns

where the file with the coefficients is saved, and associated folder.

Return type

str, str

compute_reconstruction(*coeffs_dir*, *time_idx*, *coeffs=None*, *savedir=None*, *filename=None*, *dtype='double'*, *comm=None*)

Reconstruct original data through oblique projection.

Parameters

- **coeffs_dir** (*str*) – path to coefficients folder.
- **time_idx** (*list*) – ids of times to be used for building reconstruction.
- **coeffs** (*list*) – coefficients. Default is None.
- **savedir** (*str*) – path where to save the data. Default is None.
- **filename** (*str*) – filename to use for saving reconstruction. Default is None.
- **dtype** (*str*) – datatype to be used. Default is 'double'.
- **comm** (*MPI.Comm*) – MPI communicator. Default is None.

Returns

where the file with the reconstruction is saved, and associated folder.

Return type

str, str

CHAPTER

THREE

UTILS MODULE

3.1 Postprocessing

Various postprocessing utilities.

find_nearest_coords(*coords*, *x*, *data_space_dim*)

Get nearest data coordinates to requested coordinates *coords*.

Parameters

- **coords** (*numpy.ndarray*) – coordinate requested.
- **x** (*list*) – data coordinates.
- **int** – spatial dimension of the data.

Returns

the nearest coordinate to the *coords* requested and its id.

Return type

tuple[numpy.ndarray, int]

find_nearest_freq(*freq_req*, *freq*)

Get nearest frequency to requested *freq_req* given an array of frequencies *freq*.

Parameters

- **freq_req** (*float*) – requested frequency.
- **freq** (*numpy.ndarray*) – array of frequencies.

Returns

the nearest frequency to the *freq_req* requested and its id.

Return type

tuple[float, int]

generate_2d_data_video(*X*, *time_limits=[0, 10]*, *vars_idx=[0]*, *sampling=1*, *x1=None*, *x2=None*, *coastlines=''*, *figsize=(12, 8)*, *path='CWD'*, *filename='data_video.mp4'*)

Make movie of 2D data.

Parameters

- **X** (*numpy.ndarray*) – 2D data to be plotted. First dimension must be time. Last dimension must be variable.
- **time_limits** (*2-element list*) – lower and upper time bounds to be used for video. Default is first 10 timeframes are used.

- **sampling** (*int*) – sample data every *sampling* timeframes. Default is 1 (use all timeframes).
- **x1** (*numpy.ndarray*) – x-axis coordinate. Default is None.
- **x2** (*numpy.ndarray*) – y-axis coordinate. Default is None.
- **coastlines** (*str*) – whether to overlay coastlines. Options are *regular* (longitude from 0 to 360) and *centred* (longitude from -180 to 180) Default is '' (no coastlines).
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).
- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.
- **filename** (*str*) – if specified, the plot is saved at *filename*.

get_data_from_file(*filename*)

Load data from file

Parameters
filename (*str*) – path from where to load data.

Returns
the requested data stored in *filename*

Return type
numpy.ndarray

get_modes_at_freq(*results_path*, *freq_idx*)

Get the matrix containing the SPOD modes at given frequency *freq_idx*, stored by [frequencies, spatial dimensions data, no. of variables, no. of modes].

Parameters

- **results_path** (*str*) – path to the files where the SPOD modes are stored.
- **freq_idx** (*int*) – frequency id requested.

Returns
the n_dims, n_vars, n_modes matrix containing the SPOD modes at requested frequency.

Return type
numpy.ndarray

plot_2d_data(*X*, *time_idx=[0]*, *vars_idx=[0]*, *x1=None*, *x2=None*, *xticks=None*, *yticks=None*, *equal_axes=False*, *title=''*, *coastlines=''*, *figsize=(12, 8)*, *path='CWD'*, *filename=None*, *origin=None*)

Plot 2D data.

Parameters

- **X** (*numpy.ndarray*) – 2D data to be plotted. First dimension must be time. Last dimension must be variable.
- **vars_idx** (*list*) – list of variables to plot. Default, first variable is plotted.
- **time_idx** (*list*) – list of time indices to plot. Default, first time index is plotted.
- **x1** (*numpy.ndarray*) – x-axis coordinate. Default is None.
- **x2** (*numpy.ndarray*) – y-axis coordinate. Default is None.
- **title** (*str*) – if specified, title of the plot. Default is ''.
- **coastlines** (*str*) – whether to overlay coastlines. Options are *regular* (longitude from 0 to 360) and *centred* (longitude from -180 to 180) Default is '' (no coastlines).
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).

- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.
- **filename** (*str*) – if specified, the plot is saved at *filename*.

plot_2d_modes_at_frequency(*results_path*, *freq_req*, *freq*, *vars_idx=[0]*, *modes_idx=[0]*, *x1=None*, *x2=None*, *limits_x1=(None,)*, *limits_x2=(None,)*, *fftshift=False*, *imaginary=False*, *plot_max=False*, *coastlines=''*, *title=''*, *xticks=None*, *yticks=None*, *cmap='coolwarm'*, *figsize=(12, 8)*, *equal_axes=False*, *path='CWD'*, *filename=None*, *origin=None*, *modes=None*, *pdf=None*, *shift180=False*)

Plot SPOD modes for 2D problems at a given frequency *freq_req*.

Parameters

- **results_path** (*str*) – file containing 2D SPOD modes.
- **freq_req** (*float*) – frequency to be plotted.
- **freq** (*numpy.ndarray*) – frequency array.
- **vars_idx** (*int or sequence(int)*) – variables to be plotted. Default, the first variable is plotted.
- **modes_idx** (*int or sequence(int)*) – modes to be plotted. Default, the first mode is plotted.
- **x1** (*numpy.ndarray*) – x-axis coordinate.
- **x2** (*numpy.ndarray*) – y-axis coordinate.
- **limits_x1** (*tuple*) – x-axis coordinate limits indices.
- **limits_x2** (*tuple*) – y-axis coordinate limits indices.
- **fftshift** (*bool*) – whether to perform fft-shifting. Default is False.
- **imaginary** (*bool*) – whether to plot imaginary part. Default is False
- **plot_max** (*bool*) – whether to plot a dot at maximum value of the plot. Default is False.
- **coastlines** (*str*) – whether to overlay coastlines. Options are *regular* (longitude from 0 to 360) and *centred* (longitude from -180 to 180). Default is '' (no coastlines).
- **title** (*str*) – if specified, title of the plot. Default is ''.
- **xticks** (*tuple or list*) – ticks to be set on x-axis. Default is None.
- **yticks** (*tuple or list*) – ticks to be set on y-axis. Default is None.
- **cmap** – contour map for plot. Default is 'coolwarm'.
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).
- **equal_axes** (*bool*) – if True, the axes will be equal. Default is False.
- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.
- **filename** (*str*) – if specified, the plot is saved at *filename*. Default is None.

plot_3d_modes_slice_at_frequency(*results_path*, *freq_req*, *freq*, *vars_idx=[0]*, *modes_idx=[0]*, *x1=None*, *x2=None*, *x3=None*, *slice_dim=0*, *slice_id=None*, *fftshift=False*, *imaginary=False*, *plot_max=False*, *coastlines=''*, *title=''*, *xticks=None*, *yticks=None*, *figsize=(12, 8)*, *equal_axes=False*, *path='CWD'*, *filename=None*, *origin=None*)

Plot SPOD modes for 3D problems at a given frequency *freq_req*.

Parameters

- **results_path** (*str*) – file containing 3D SPOD modes.
- **freq_req** (*float*) – frequency to be plotted.
- **freq** (*numpy.ndarray*) – frequency array.
- **vars_idx** (*int or sequence(int)*) – variables to be plotted. Default, the first variable is plotted.
- **modes_idx** (*int or sequence(int)*) – modes to be plotted. Default, the first mode is plotted.
- **x1** (*numpy.ndarray*) – x-axis coordinate. Default is None.
- **x2** (*numpy.ndarray*) – y-axis coordinate. Default is None.
- **x3** (*numpy.ndarray*) – z-axis coordinate. Default is None.
- **slice_dim** (*int*) – axis to slice. Either 0, 1, or 2. Default is 0.
- **slice_id** (*int*) – id of the slice to extract along *slice_dim*. Default is None. In this case, the slice_id is selected as the one that corresponds to the maximum value along *slice_dim*.
- **fftshift** (*bool*) – whether to perform fft-shifting. Default is False.
- **imaginary** (*bool*) – whether to plot imaginary part. Default is False
- **plot_max** (*bool*) – whether to plot a dot at maximum value of the plot. Default is False.
- **coastlines** (*str*) – whether to overlay coastlines. Options are *regular* (longitude from 0 to 360) and *centred* (longitude from -180 to 180) Default is '' (no coastlines).
- **title** (*str*) – if specified, title of the plot. Default is ''.
- **xticks** (*tuple or list*) – ticks to be set on x-axis. Default is None.
- **yticks** (*tuple or list*) – ticks to be set on y-axis. Default is None.
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).
- **equal_axes** (*bool*) – if True, the axes will be equal. Default is False.
- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.
- **filename** (*str*) – if specified, the plot is saved at *filename*. Default is None.

plot_data_tracers(*X*, *coords_list*, *x=None*, *time_limits=[0, 10]*, *vars_idx=[0]*, *title=''*, *figsize=(12, 8)*, *path='CWD'*, *filename=None*)

Plot data tracers for nD problems.

Parameters

- **X** (*numpy.ndarray*) – nD data.
- **coords_list** (*list(tuple(*),)*) – list of tuples containing coordinates to be plotted.
- **x** (*numpy.ndarray*) – data coordinates. Default is None.
- **time_limits** (*2-element list*) – lower and upper time bounds to be plotted. Default is first 10 timeframes are plotted.
- **title** (*str*) – if specified, title of the plot. Default is ''.
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).
- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.
- **filename** (*str*) – if specified, the plot is saved at *filename*. Default is None.

plot_eigs(*eigs*, *title=''*, *figsize=(12, 8)*, *show_axes=True*, *equal_axes=False*, *path='CWD'*, *filename=None*)

Plot eigenvalues *eigs*.

Parameters

- **eigs** (*ndarray*) – eigenvalues.
- **title** (*str*) – if specified, title of the plot.
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).
- **show_axes** (*bool*) – if True, the axes will be showed. Default is True.
- **equal_axes** (*bool*) – if True, the axes will be equal. Default is False.
- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.
- **filename** (*str*) – if specified, the plot is saved at *filename*. Default is None.

plot_eigs_vs_frequency(*eigs*, *freq*, *title=''*, *xticks=None*, *yticks=None*, *show_axes=True*, *equal_axes=False*, *figsize=(12, 8)*, *fontname='DejaVu Sans'*, *fontsize=16*, *path='CWD'*, *filename=None*)

Plot eigenvalues vs. frequency.

Parameters

- **eigs** (*ndarray*) – eigenvalues.
- **freq** (*ndarray*) – frequency vector to be used as the x-axis.
- **title** (*str*) – if specified, title of the plot.
- **xticks** (*tuple or list*) – ticks to be set on x-axis. Default is None.
- **yticks** (*tuple or list*) – ticks to be set on y-axis. Default is None.
- **show_axes** (*bool*) – if True, the axes will be showed. Default is True.
- **equal_axes** (*bool*) – if True, the axes will be equal. Default is False.
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).
- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.
- **filename** (*str*) – if specified, the plot is saved at *filename*. Default is None.

plot_eigs_vs_period(*eigs*, *freq*, *title=''*, *xticks=None*, *yticks=None*, *show_axes=True*, *equal_axes=False*, *figsize=(12, 8)*, *fontname='DejaVu Sans'*, *fontsize=16*, *path='CWD'*, *filename=None*)

Plot eigenvalues vs. period = 1 / freq.

Parameters

- **eigs** (*ndarray*) – eigenvalues.
- **freq** (*ndarray*) – frequency vector to be used as the x-axis.
- **title** (*str*) – if specified, title of the plot. Default is ‘’.
- **xticks** (*tuple or list*) – ticks to be set on x-axis. Default is None.
- **yticks** (*tuple or list*) – ticks to be set on y-axis. Default is None.
- **show_axes** (*bool*) – if True, the axes will be showed. Default is True.
- **equal_axes** (*bool*) – if True, the axes will be equal. Default is False.
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).
- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.

- **filename** (*str*) – if specified, the plot is saved at *filename*. Default is None.

plot_mode_tracers(*results_path*, *freq_req*, *freq*, *coords_list*, *x=None*, *vars_idx=[0]*, *modes_idx=[0]*, *fftshift=False*, *title=''*, *figsize=(12, 8)*, *path='CWD'*, *filename=None*)

Plot SPOD mode tracers for nD problems at given frequency *freq_req*.

Parameters

- **results_path** (*str*) – file containing nD SPOD modes.
- **freq_req** (*float*) – frequency to be plotted.
- **freq** (*numpy.ndarray*) – frequency array.
- **coords_list** (*list(tuple(*),)*) – list of tuples containing coordinates to be plotted.
- **x** (*numpy.ndarray*) – data coordinates. Default is None.
- **fftshift** (*bool*) – whether to perform fft-shifting. Default is False.
- **title** (*str*) – if specified, title of the plot. Default is ‘’.
- **figsize** (*tuple(int,int)*) – size of the figure (width,height). Default is (12,8).
- **path** (*str*) – if specified, the plot is saved at *path*. Default is CWD.
- **filename** (*str*) – if specified, the plot is saved at *filename*. Default is None.

3.2 Weights

Module implementing weights for standard cases.

apply_normalization(*data*, *weights*, *n_vars*, *method='variance'*, *comm=None*)

Normalization of weights if required by data variance.

Parameters

- **data** (*numpy.ndarray*) – data.
- **weights** (*numpy.ndarray*) – weights.
- **n_vars** (*int*) – number of variables.
- **method** (*int*) – normalization method. Default is ‘variance’.
- **comm** (*MPI.Comm*) – MPI communicator.

Returns
the normalized weights.

Return type
numpy.ndarray

custom(***kwargs*)

Customized weights to be implemented by user if required. Note, weights must have the same dimension as the data flattened spatial dimension (i.e. if we have two spatial dimensions, with length 10, and 20, respectively, and we have two variables, this function must return a np.ndarray of dimension = 10 x 20 x 2 = 400).

geo_trapz_2D(*x1_dim*, *x2_dim*, *n_vars*, ***kwargs*)

2D integration weights for geospatial data via trapezoidal rule.

Parameters

- **x1_dim** (*numpy.ndarray*) – first spatial coordinate.

- **x2_dim** (*numpy.ndarray*) – second spatial coordinate.
- **n_vars** (*int*) – number of variables.

Returns
the computed weights.

Return type
numpy.ndarray

geo_trapz_3D(*x1_dim*, *x2_dim*, *x3_dim*, *n_vars*, ***kwargs*)

3D integration weights for geospatial data via trapezoidal rule.

Parameters

- **x1_dim** (*numpy.ndarray*) – first spatial coordinate.
- **x2_dim** (*numpy.ndarray*) – second spatial coordinate.
- **x3_dim** (*numpy.ndarray*) – third spatial coordinate.
- **n_vars** (*int*) – number of variables.

Returns
the computed weights.

Return type
numpy.ndarray

PYTHON MODULE INDEX

p

INDEX

P

S

W

X