Sound Field Analysis Toolbox

Release 88e0b9b

SFA Toolbox Developers

Sep 13, 2017

Contents

The Sound Field Analysis Toolbox for NumPy/Python provides implementations of various techniques for the analysis of sound fields and beamforming using microphone arrays.

Source code and issue tracker: https://github.com/spatialaudio/sfa-numpy

License: MIT – see the file `LICENSE` for details.

Quick start:

- Install NumPy, SciPy and for the examples Matplotlib
- `git clone https://github.com/spatialaudio/sfa-numpy.git`
- `cd sfa-numpy`
- `python setup.py install --user`

1 Usage

1.1 Requirements

Obviously, you'll need Python[1]. We normally use Python 3.x, but it *should* also work with Python 2.x. NumPy[2] and SciPy[3] are needed for the calculations. If you also want to plot the resulting sound fields, you'll need matplotlib[4].

[1] http://www.python.org/
[2] http://www.numpy.org/
[3] http://www.scipy.org/scipylib/
[4] http://matplotlib.org/

Instead of installing all of them separately, you should probably get a Python distribution that already includes everything, e.g. Anaconda[5].

1.2 How to Get Started

Various examples are located in the directory

2 Modal Beamforming

Submodules for modal beamforming

2.1 Angular

`micarray.modal.angular.`**`sht_matrix`**(*N*, *azi*, *elev*, *weights=None*)
    (N+1)**2 x M SHT matrix

`micarray.modal.angular.`**`Legendre_matrix`**(*N*, *ctheta*)
    (N+1) x M matrix of weighted Legendre Polynominals 2*n+1/4*pi * P_n(ctheta)

`micarray.modal.angular.`**`grid_equal_angle`**(*n*)
    equi_angular grid on sphere. (cf. Rafaely book, sec.3.2)

`micarray.modal.angular.`**`grid_gauss`**(*n*)
    Gauss-Legendre sampling points on sphere. (cf. Rafaely book, sec.3.3)

2.2 Radial

`micarray.modal.radial.`**`spherical_pw`**(*N*, *k*, *r*, *setup*)
    Radial coefficients for a plane wave

Computes the radial component of the spherical harmonics expansion of a plane wave impinging on a spherical array.

$$\mathring{P}_n(k) = 4\pi i^n b_n(kr)$$

Parameters

- **N** (*int*) – Maximum order.
- **k** (*array_like*) – Wavenumber.
- **r** (*float*) – Radius of microphone array.
- **setup** (*{'open', 'card', 'rigid'}*) – Array configuration (open, cardioids, rigid).

Returns *numpy.ndarray* – Radial weights for all orders up to N and the given wavenumbers.

`micarray.modal.radial.`**`spherical_ps`**(*N*, *k*, *r*, *rs*, *setup*)
    Radial coefficients for a point source

Computes the radial component of the spherical harmonics expansion of a point source impinging on a spherical array.

$$\mathring{P}_n(k) = 4\pi(-i)kh_n^{(2)}(kr_s)b_n(kr)$$

Parameters

- **N** (*int*) – Maximum order.

[5] http://docs.continuum.io/anaconda/

- **k** (*array_like*) – Wavenumber.
- **r** (*float*) – Radius of microphone array.
- **rs** (*float*) – Distance of source.
- **setup** (*{'open', 'card', 'rigid'}*) – Array configuration (open, cardioids, rigid).

Returns *numpy.ndarray* – Radial weights for all orders up to N and the given wavenumbers.

`micarray.modal.radial.`**`weights`**(*N*, *kr*, *setup*)

Radial weighing functions

Computes the radial weighting functions for diferent array types (cf. eq.(2.62), Rafaely 2015).

For instance for an rigid array

$$b_n(kr) = j_n(kr) - \frac{j_n'(kr)}{h_n^{(2)\prime}(kr)} h_n^{(2)}(kr)$$

Parameters

- **N** (*int*) – Maximum order.
- **kr** (*array_like*) – Wavenumber * radius.
- **setup** (*{'open', 'card', 'rigid'}*) – Array configuration (open, cardioids, rigid).

Returns *numpy.ndarray* – Radial weights for all orders up to N and the given wavenumbers.

`micarray.modal.radial.`**`regularize`**(*dn*, *a0*, *method*)

(cf. Rettberg, Spors : DAGA 2014)

`micarray.modal.radial.`**`diagonal_mode_mat`**(*bk*)

3 Utilities

`micarray.util.`**`norm_of_columns`**(*A*, *p=2*)

Vector p-norm of each column.

`micarray.util.`**`coherence_of_columns`**(*A*)

Mutual coherence of columns of A.

`micarray.util.`**`asarray_1d`**(*a*, ***kwargs*)

Squeeze the input and check if the result is one-dimensional.

Returns *a* converted to a `numpy.ndarray`[6] and stripped of all singleton dimensions. Scalars are "upgraded" to 1D arrays. The result must have exactly one dimension. If not, an error is raised.

[6] https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray