Scikit-Qfit Documentation

Release

npdata

August 26, 2016

Contents

A gradient-orthogonal Q-polynomial representation of axially symmetric optical surfaces has been used for several years by designers. It improves upon the standard polynomial form by being simpler to interpret, using fewer terms to adequately define a surface, and sometimes even offering quicker convergence in design optimization. Q-polynomials were first introduced with the publication of:

- G W Forbes, Shape specification for axially symmetric optical surfaces, Opt. Express 15, 5218-5226 (2007)

The use of Q-polynomials was extended, by the original author, to address freeform shapes through the following articles:

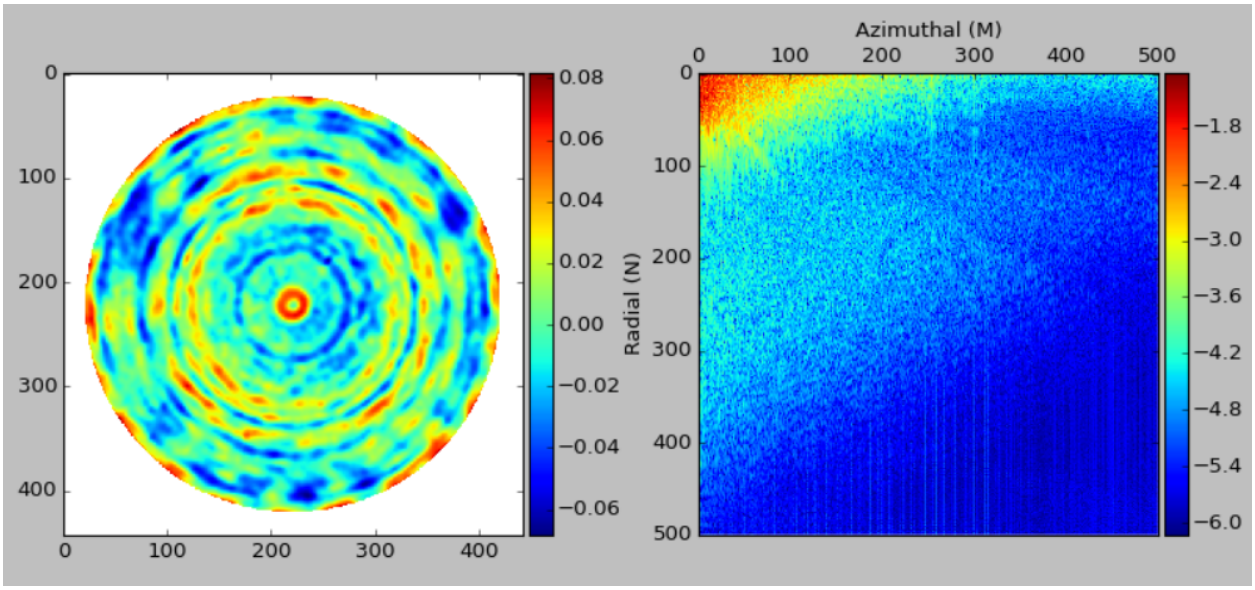
- Fitting freeform shapes with orthogonal bases, Opt. Express 21, 19061-19081 (2013)
- Characterizing the shape of freeform optics, Opt. Express 20(3), 2483-2499 (2012)
- Robust, efficient computational methods for axially symmetric optical aspheres, Opt. Express 18(19), 19700-19712 (2010)

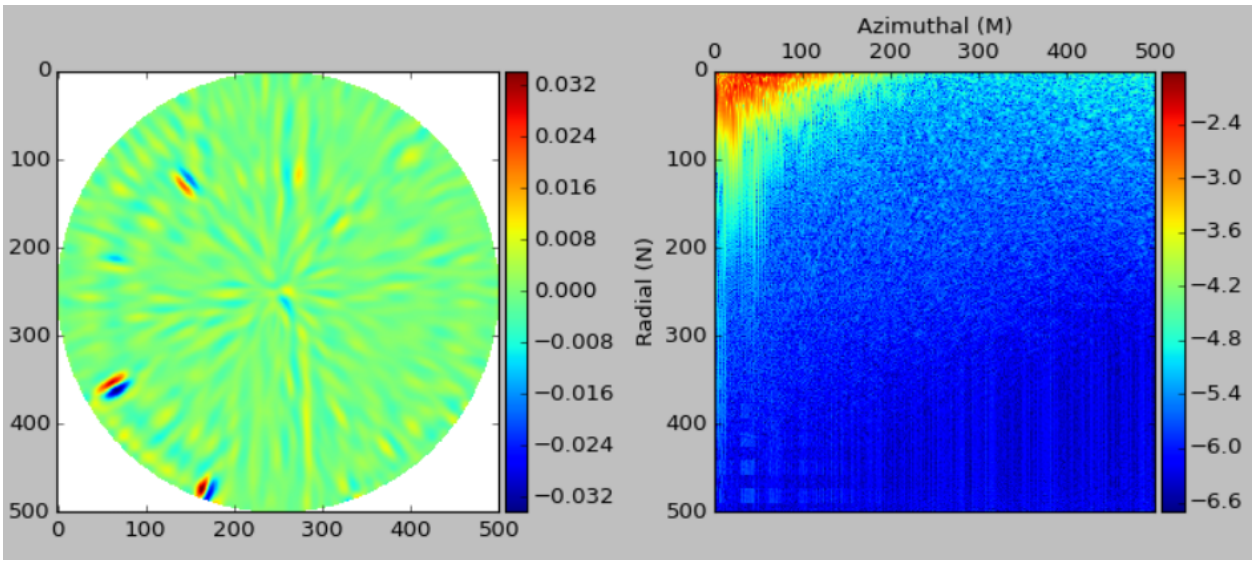
The implementation of this package follows the description in “Fitting freeform shapes with orthogonal bases”.

CHAPTER 1

Q-spectrum examples

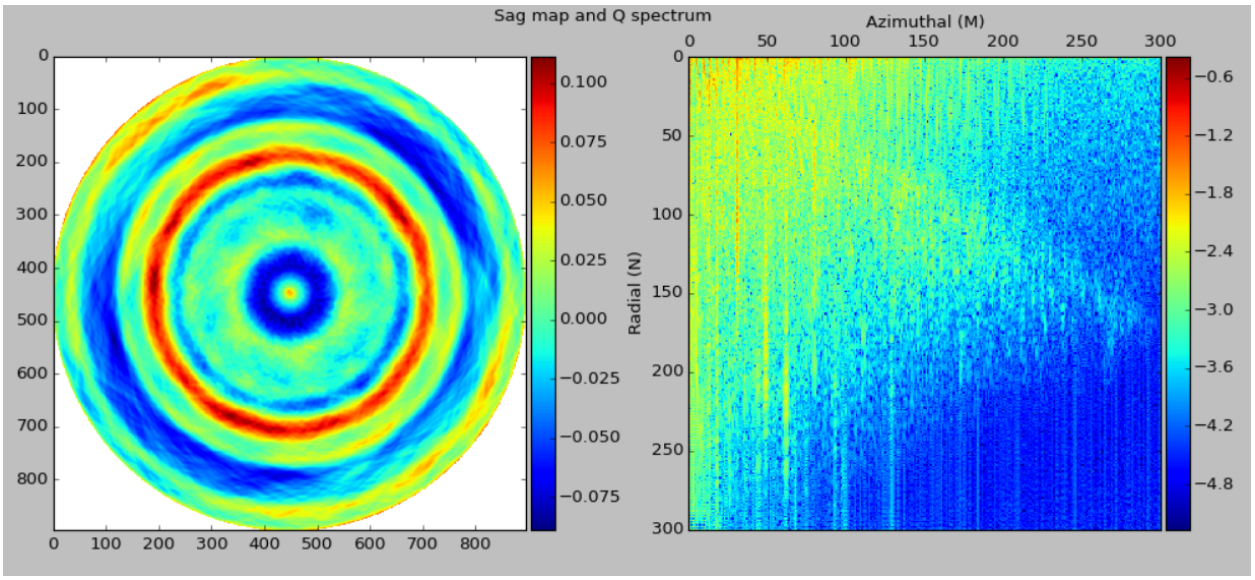
The Q-spectrum offers a natural way to quantify and investigate surface structure on parts with circular apertures. For example, spoke structure on the surface appears in specific columns of the spectrum whereas raster patterns appear as diagonal bands with a slope of one half. Rotationally symmetric structure, such a rings etc., are all contained in the m=0 column. The spectrum is computed with an FFT-like step that suffers similarly from aliasing unless a sufficiently large range of frequencies is computed.

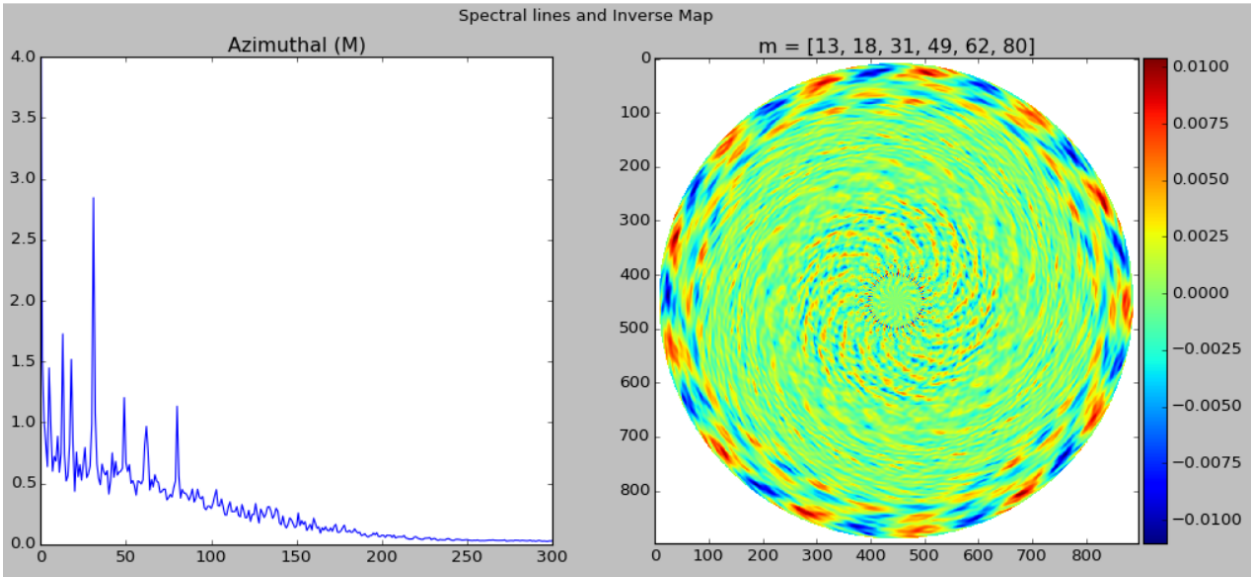




1.1 Generating maps

Interesting features in the spectrum can be extracted to create topography layers that quantify and highlight flaws in the surface. The same process can be used to monitor residuals in the data after subtracting fitted polynomial terms. In the example below the radial-azimuthal spectrum shows quite distinct columns at particular azimuthal orders that are even more evident by summing the squares of the coefficients to produce a spectral line plot. The last image shows the contribution of just a sample of the spectral lines in the original data map. The article "Fitting freeform shapes with orthogonal bases" provides more details on the types of spectral analysis that can be performed and the characteristic flaws associated with polishing techniques that employ either raster or rotational modes.





The above images were generated from data supplied by Mahr GmbH.

CHAPTER 2

Usage

To generate a Q-freeform spectrum from a data map, pass the coordinate axis x and y and 2-D array of data with shape (x.size,y.size) as arguments to the method qspec(). The azimuthal and radial spectrum limits are set by m_max and n_max respectively.

```
>>> import skqfit.qspectre as qf
>>> ...
>>> qspec = qf.qspec(x, y, zmap, m_max=500, n_max=500)
```

To observe the contribution of spectrum components it is necessary to use the sine and cosine elements in the forward processing step. After modifying the spectrum, an inverse process creates a data map along with optional x and y derivatives.

```
>>> qs = qf.QSpectrum()
>>> qs.data_map(x, y, zmap)
>>> a_nm, b_nm = qs.q_fit(mmax, nmax)
>>> ... # modify spectrum
>>> nmap, dfdx, dfdy = qs.build_map(x, y, a_nm, b_nm, inc_deriv=True)
```

A 1D trace across the fitted data can also be generated by passing a set of xv, yv arrays that represent the (x, y) coordinates.

```
>>> zv, dfdx, dfdy = qs.build_profile(xv, yv, a_nm, b_nm, inc_deriv=True)
```

CHAPTER 3

Code structure

The package only contains two modules. The Jacobi module provides support for normalised Jacobi polynomials that extend the range of usable parameters before an overflow condition is encountered. Even though the Q-fitting algorithm is procedural the implementation is via a Q-spectrum class as it allowed the spectrum to be iterated with different parameters without having to reload the data. The algorithm implementation refers to the relevant section of the reference documents.

3.1 The `skqfit.asmjacp` Module

References:

G W Forbes, "Characterizing the shape of freeform optics", Opt. Express 20(3), 2483-2499 (2012)

class skqfit.asmjacp.**AsymJacobiP**(*nmax*)

Bases: object

Generate asymmetric Jacobi like polynominals needed for the freeform fit as defined in the reference document [A.1]

The Jacobi P polynominals can generate large values that lead to a double overflow for large m and n values. It is tested to (1500, 1500) for (m,n). Exceeding these values may lead to overflow events.

The scipi.special.jacobi function is has a 1.5:1 performance advantage over the current implementation but it doesn't support the normalization and can lead to an overflow condition for n or m over 500.

build_recursion(*m*)

Build the recurssion coefficients and saves them as a sequence of tuples. These are the coefficients to build the m type polynominals up to order n.

jmat_u_x(*jmat*, *uv*, *xv*)

Builds the asymmetric Jacobi P polynomial as defined in [2] A.1 for all of the x values with the scaling factor of u**m which extends the range of usable (m,n) values before an overflow condition occurs.

jmat_x(*jmat*, *xv*)

Builds the asymmetric Jacobi P polynomial as defined in [2] A.1 for all a vector of x values.

jvec_u_x(*jvec*, *u*, *x*)

Builds the asymmetric Jacobi P polynomial for a single x as defined in [2] A.1 for all of the x values with the scaling factor of u**m which extends the range of usable (m,n) values.

jvec_x(*jvec*, *x*)

Builds the sequence of jacobi polynomials for the value x based on [A.2-5]

3.2 The skqfit.qspectre Module

Reference documents

[1] G W Forbes, "Fitting freeform shapes with orthogonal bases", Opt. Express 21, 19061-19081 (2013) [2] G W Forbes, "Characterizing the shape of freeform optics", Opt. Express 20(3), 2483-2499 (2012) [3] G W Forbes, "Robust, efficient computational methods for axially symmetric optical aspheres", Opt. Express 18(19), 19700-19712 (2010)

class skqfit.qspectre.**QSpectrum**(*m_max=None*, *n_max=None*)
Bases: object

Performs precomputation if Q spectrum limits are passed, otherwise it is delayed until the data is loaded. The class supports processing a data map or a pointer to a sag function that can be used for analytic testing.

Parameters: m_max, n_max: int

The azimuthal and radial spectrum order. Setting values above 1500 may lead to overflow events.

bfs_param()
Returns the fitted radius, curvature and centre.

Returns: radius, curvature, centre: float, float, (x,y) tuple

build_map(*x*, *y*, *a_nm*, *b_nm*, *curv=None*, *radius=None*, *centre=None*, *extend=1.0*, *interpolated=True*, *inc_deriv=False*)
Creates a 2D topography map and optional x and y derivate maps using the x and y axis vectors and the Q-freeform parameters.

Parameters:

x, y: array X, and Y axis values for the map to be created. The arrays must be sorted to increasing order.

a_nm, b_nm: 2D array The cosine and sine terms for the Q freeform polynominal

curv: float Nominal curvature for the part. If None uses the estimated value from the previous fit.

radius: float Defines the circular domain from the centre. If None uses the estimated value from the previous fit.

centre: (cx, cy) The centre of the part in axis coordinates. If None uses the estimated value from previous fit.

extend: float Generate a map over extend * radius from the centre

interpolated: boolean If True uses a high resolution regular polar grid to build the underlying data and a spline interpolation to extract the (x, y) grid, otherwise it evaluates each (x, y) point exactly. The non-interpolated solution is significanatly slower and only practical for smaller array sizes.

inc_deriv: boolean Return the X and Y derivatives as additional maps

Returns:

zmap: 2-D array Data map with shape (x.size, y.size)

xder: 2-D array X derivative map with shape (x.size, y.size) if inc_deriv is True, else None

yder: 2-D array Y derivative map with shape (x.size, y.size) if inc_deriv is True, else None

build_profile(*xv*, *yv*, *a_nm*, *b_nm*, *curv=None*, *radius=None*, *centre=None*, *extend=1.0*, *inc_deriv=False*)
Returns the nominal sag and optional x and y derivatives along a 1D trajectory of (x, y) coordinates.

Parameters:

x, y: arrays Arrays of values representing the (x, y) coordinates.

a_nm, b_nm: 2D array The cosine and sine terms for the Q freeform polynominal

curv: float Nominal curvature for the part. If None uses the estimated value from the previous fit.

radius: float Defines the circular domain from the centre. If None uses the estimated value from the previous fit.

centre: (cx, cy) The centre of the part in axis coordinates. If None uses the estimated value from previous fit.

extend: float Generate a map over extend * radius from the centre

inc_deriv: boolean Return the X and Y derivatives as additional maps

Returns:

zval: array Sag values for the (x, y) sequence

xder: array X derivative map for the (x, y) sequence if inc_deriv is True, else None

yder: array Y derivative map for the (x, y) sequence if inc_deriv is True, else None

build_q_spectrum(*m_max=None*, *n_max=None*)

Fits the departure from a best fit sphere to the Q-polynominals as defined in [1](1.1) and returns the root sum square of the azimuthal terms.

Parameters:

m_max, n_max: int The azimuthal and radial spectrum order. If None it uses the previous values and if not defined it matches the values to the pixel resolution.

Returns:

2D array The (m,n) matrix representation of the spectrum (sqrt(cos^2 + sin^2)) terms

data_map(*x*, *y*, *zmap*, *centre=None*, *radius=None*, *shrink_pixels=7*, *bfs_curv=None*)

Creates the spline interpolator for the map, determines the best fit sphere and minimum valid radius.

Parameters:

x, y: arrays The arrays are the X and Y axis values for the data map. The arrays must be sorted to increasing order.

zmap: array_like 2-D array of data with shape (x.size,y.size).

centre: (cx, cy) The centre of the part in axis coordinates. If None the centre is estimated by a centre of mass calculation

radius: float Defines the circular domain from the centre. If None it determines the maximum radius from the centre that contains no invalids (NAN).

shrink_pixels: int The estimated radius is reduced by 7 pixels to avoid edge effects with the spline interpolation. Ignored if the radius is specified.

q_fit(*m_max=None*, *n_max=None*)

Fits the departure from a best fit sphere to the Q-freeform polynominals as defined in [1](1.1) and returns the individual sine and cosine terms.

Parameters:

m_max, n_max: int The azimuthal and radial spectrum order. If None it uses the previous values and if not defined it matches the values to the pixel resolution. The maximum resolution supported is (1500, 1500)

Returns:

a_nm, b_nm: 2D array The (n,m) matrix representation of the cosine and sine terms

set_sag_fn(*sag_fn*, *radius*, *bfs_curv=None*)

A vectorized polar sag function that takes rho and theta as arguments.

This function is used to pass an analytic sag function to test the performance of the algorithm to a higher precision but can also be used to define the input map and bypass data_map().

skqfit.qspectre.**qspec**(*x*, *y*, *zmap*, *m_max=None*, *n_max=None*, *centre=None*, *radius=None*, *shrink_pixels=7*)

A wrapper function that creates the Q-spectrum object, loads and performs the fit of the data to the Q polynomials.

Parameters:

x, y: array_like The interpolator uses grid points defined by the coordinate arrays x, y. The arrays must be sorted to increasing order.

zmap: array_like 2-D array of data with shape (x.size,y.size)

m_max, n_max: int The azimuthal and radial spectrum order. If None, it uses the previous values and if not defined it matches the values to the pixel resolution.

centre: (cx, cy) The centre of the part in axis coordinates. If None the centre is estimated by a centre of mass calculation

radius: float Defines the circular domain from the centre. If None it determines the maximum radius from the centre that contains no invalids (NAN).

shrink_pixels: int The estimated radius is reduced by 7 pixels to avoid edge effects with the spline interpolation. Ignored if the radius is specified.

Returns:

2D array The (m,n) matrix representation of the spectrum (sqrt(cos^2 + sin^2)) terms

CHAPTER 4

Installation

4.1 Quick Installation

If you have pip installed, you should be able to install the latest stable release of `scikit-qfit` by running the following:

```
pip install scikit-qfit
```

4.2 Latest Software

The latest software can be downloaded from GitHub

4.3 Installation Dependencies

`scikit-qfit` requires that the following software packages to be installed:

- Python 2.7.6 or later.
- NumPy 1.8.2 or later.
- SciPy 0.13.3 or later.

CHAPTER 5

Contents

5.1 Modules

5.1.1 skqfit package

Submodules

skqfit.asmjacp module

References:

G W Forbes, “Characterizing the shape of freeform optics”, Opt. Express 20(3), 2483-2499 (2012)

class skqfit.asmjacp.**AsymJacobiP**(*nmax*)
Bases: object

Generate asymmetric Jacobi like polynominals needed for the freeform fit as defined in the reference document [A.1]

The Jacobi P polynominals can generate large values that lead to a double overflow for large m and n values. It is tested to (1500, 1500) for (m,n). Exceeding these values may lead to overflow events.

The scipi.special.jacobi function is has a 1.5:1 performance advantage over the current implementation but it doesn’t support the normalization and can lead to an overflow condition for n or m over 500.

build_recursion(*m*)
Build the recurssion coefficients and saves them as a sequence of tuples. These are the coefficients to build the m type polynominals up to order n.

jmat_u_x(*jmat*, *uv*, *xv*)
Builds the asymmetric Jacobi P polynomial as defined in [2] A.1 for all of the x values with the scaling factor of u**m which extends the range of usable (m,n) values before an overflow condition occurs.

jmat_x(*jmat*, *xv*)
Builds the asymmetric Jacobi P polynomial as defined in [2] A.1 for all a vector of x values.

jvec_u_x(*jvec*, *u*, *x*)
Builds the asymmetric Jacobi P polynomial for a single x as defined in [2] A.1 for all of the x values with the scaling factor of u**m which extends the range of usable (m,n) values.

jvec_x(*jvec*, *x*)
Builds the sequence of jacobi polynomials for the value x based on [A.2-5]

skqfit.qspectre module

Reference documents

[1] G W Forbes, "Fitting freeform shapes with orthogonal bases", Opt. Express 21, 19061-19081 (2013) [2] G W Forbes, "Characterizing the shape of freeform optics", Opt. Express 20(3), 2483-2499 (2012) [3] G W Forbes, "Robust, efficient computational methods for axially symmetric optical aspheres", Opt. Express 18(19), 19700-19712 (2010)

class skqfit.qspectre.**QSpectrum**(*m_max=None*, *n_max=None*)
Bases: object

Performs precomputation if Q spectrum limits are passed, otherwise it is delayed until the data is loaded. The class supports processing a data map or a pointer to a sag function that can be used for analytic testing.

Parameters: m_max, n_max: int

The azimuthal and radial spectrum order. Setting values above 1500 may lead to overflow events.

bfs_param()
Returns the fitted radius, curvature and centre.

Returns: radius, curvature, centre: float, float, (x,y) tuple

build_map(*x*, *y*, *a_nm*, *b_nm*, *curv=None*, *radius=None*, *centre=None*, *extend=1.0*, *interpolated=True*, *inc_deriv=False*)
Creates a 2D topography map and optional x and y derivate maps using the x and y axis vectors and the Q-freeform parameters.

Parameters:

x, y: array X, and Y axis values for the map to be created. The arrays must be sorted to increasing order.

a_nm, b_nm: 2D array The cosine and sine terms for the Q freeform polynominal

curv: float Nominal curvature for the part. If None uses the estimated value from the previous fit.

radius: float Defines the circular domain from the centre. If None uses the estimated value from the previous fit.

centre: (cx, cy) The centre of the part in axis coordinates. If None uses the estimated value from previous fit.

extend: float Generate a map over extend * radius from the centre

interpolated: boolean If True uses a high resolution regular polar grid to build the underlying data and a spline interpolation to extract the (x, y) grid, otherwise it evaluates each (x, y) point exactly. The non-interpolated solution is significanatly slower and only practical for smaller array sizes.

inc_deriv: boolean Return the X and Y derivatives as additional maps

Returns:

zmap: 2-D array Data map with shape (x.size, y.size)

xder: 2-D array X derivative map with shape (x.size, y.size) if inc_deriv is True, else None

yder: 2-D array Y derivative map with shape (x.size, y.size) if inc_deriv is True, else None

build_profile(*xv*, *yv*, *a_nm*, *b_nm*, *curv=None*, *radius=None*, *centre=None*, *extend=1.0*, *inc_deriv=False*)
Returns the nominal sag and optional x and y derivatives along a 1D trajectory of (x, y) coordinates.

Parameters:

x, y: arrays Arrays of values representing the (x, y) coordinates.

a_nm, b_nm: 2D array The cosine and sine terms for the Q freeform polynominal

curv: float Nominal curvature for the part. If None uses the estimated value from the previous fit.

radius: float Defines the circular domain from the centre. If None uses the estimated value from the previous fit.

centre: (cx, cy) The centre of the part in axis coordinates. If None uses the estimated value from previous fit.

extend: float Generate a map over extend * radius from the centre

inc_deriv: boolean Return the X and Y derivatives as additional maps

Returns:

zval: array Sag values for the (x, y) sequence

xder: array X derivative map for the (x, y) sequence if inc_deriv is True, else None

yder: array Y derivative map for the (x, y) sequence if inc_deriv is True, else None

build_q_spectrum(*m_max=None*, *n_max=None*)

Fits the departure from a best fit sphere to the Q-polynominals as defined in [1](1.1) and returns the root sum square of the azimuthal terms.

Parameters:

m_max, n_max: int The azimuthal and radial spectrum order. If None it uses the previous values and if not defined it matches the values to the pixel resolution.

Returns:

2D array The (m,n) matrix representation of the spectrum (sqrt(cos^2 + sin^2)) terms

data_map(*x*, *y*, *zmap*, *centre=None*, *radius=None*, *shrink_pixels=7*, *bfs_curv=None*)

Creates the spline interpolator for the map, determines the best fit sphere and minimum valid radius.

Parameters:

x, y: arrays The arrays are the X and Y axis values for the data map. The arrays must be sorted to increasing order.

zmap: array_like 2-D array of data with shape (x.size,y.size).

centre: (cx, cy) The centre of the part in axis coordinates. If None the centre is estimated by a centre of mass calculation

radius: float Defines the circular domain from the centre. If None it determines the maximum radius from the centre that contains no invalids (NAN).

shrink_pixels: int The estimated radius is reduced by 7 pixels to avoid edge effects with the spline interpolation. Ignored if the radius is specified.

q_fit(*m_max=None*, *n_max=None*)

Fits the departure from a best fit sphere to the Q-freeform polynominals as defined in [1](1.1) and returns the individual sine and cosine terms.

Parameters:

m_max, n_max: int The azimuthal and radial spectrum order. If None it uses the previous values and if not defined it matches the values to the pixel resolution. The maximum resolution supported is (1500, 1500)

Returns:

a_nm, b_nm: 2D array The (n,m) matrix representation of the cosine and sine terms

set_sag_fn(*sag_fn*, *radius*, *bfs_curv=None*)

A vectorized polar sag function that takes rho and theta as arguments.

This function is used to pass an analytic sag function to test the performance of the algorithm to a higher precision but can also be used to define the input map and bypass data_map().

skqfit.qspectre.**qspec**(*x*, *y*, *zmap*, *m_max=None*, *n_max=None*, *centre=None*, *radius=None*, *shrink_pixels=7*)

A wrapper function that creates the Q-spectrum object, loads and performs the fit of the data to the Q polynomials.

Parameters:

x, y: array_like The interpolator uses grid points defined by the coordinate arrays x, y. The arrays must be sorted to increasing order.

zmap: array_like 2-D array of data with shape (x.size,y.size)

m_max, n_max: int The azimuthal and radial spectrum order. If None, it uses the previous values and if not defined it matches the values to the pixel resolution.

centre: (cx, cy) The centre of the part in axis coordinates. If None the centre is estimated by a centre of mass calculation

radius: float Defines the circular domain from the centre. If None it determines the maximum radius from the centre that contains no invalids (NAN).

shrink_pixels: int The estimated radius is reduced by 7 pixels to avoid edge effects with the spline interpolation. Ignored if the radius is specified.

Returns:

2D array The (m,n) matrix representation of the spectrum (sqrt(cos^2 + sin^2)) terms

Module contents

5.2 License

```
The MIT License (MIT)

Copyright (c) 2015 npdata

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

5.3 Developers

- npdata <npdata@bigpond.com>

5.4 Changelog

5.4.1 Version 0.1.4

- Adds spectrum to topography functionality
- Updated documentation and minor fixes

5.4.2 Version 0.1.3

- Fixed error when data has invalids

5.4.3 Version 0.1.2

- Fixed error in interpolator call
- Added documentation to readthedocs

5.4.4 Version 0.1.1

- Corrections and addition to PyPi documentation
- Added example file

5.4.5 Version 0.1

- Initial public pre-alpha release

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

S

A

B

D

J

Q

S