pystog Documentation

Release 0.1.3

Marshall McDonnell

Jun 04, 2023

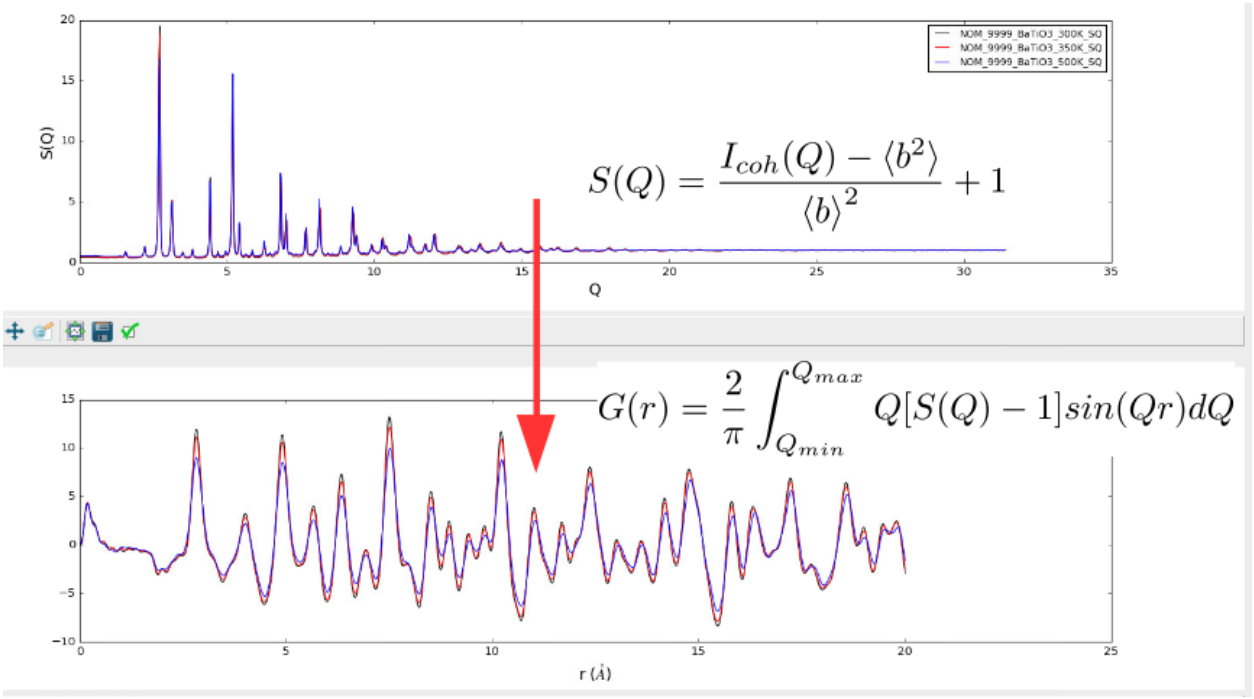
Contents:

CHAPTER 1

About

From total scattering functions, we have reciprocal-space structure factors and real-space pair distribution functions that are related via a Fourier transform. PyStoG is a package that allows for:

1. Converting between the various functions used by different "communities" (ie researchers who study crystalline versus amorphous or glass materials). Conversions are for either real-space or reciprocal-space.
2. Perform the transform between the different available functions of choice
3. Fourier filter to remove spurious artificats in the data (ie aphysical, sub-angstrom low-r peaks in G(r) from experiments)



The name **PyStoG** comes from the fact that this is a *Pythonized* version of **StoG**, a ~30 year old Fortran program that is part of the RMCProfile software suite. **StoG** means **"S(Q) to G(r)"** for the fact that it takes recirpocal-space S(Q) patterns from files and transforms them into a single G(r) pattern. The original *StoG* program has been developed, in reverse chronological order, by:

- Matthew Tucker and Martin Dove (~2009)
- Spencer Howells (~1989)
- Jack Carpenter (prior to 1989)

A current state of the **StoG** program is kept in the *fortran* directory of this package.

This project was initially just a "sandbox" for taking the capabilities of **StoG** and migrating them over to the Mantid Framework. With more and more use cases, **PyStoG** was further developed as the stand-alone project it is now. Yet, migration to the Mantid Framework is still a goal since it feeds into the ADDIE project.

PyStoG is not a Python replica of StoG but has the same goal as StoG. Yet, has capabilites that surpass StoG such as multiple input/output real and reciprocal space functions and modules allowing for re-construction of the workflow for processing total scattering data.

CHAPTER 2

Installation

2.1 Requirements

- Python (3.6, 3.7, 3.8, and 3.9 are tested)
- numpy

2.2 Using PyPI

Installation is available via *pip*.

```
pip install pystog
```

or for a local install

```
pip install pystog --user #locally installed in $HOME/.local
```

2.3 Using python setup.py

A setup.py is available to install but it is recommended to do this in a virtual environment. For system install, use *PyPi* instead.

```
python setup.py install
```

2.4 Development

For a development environment, you can use virtualenv to setup an isolated environemnt, namely *ENV*:

```
python -m virtualenv /path/to/ENV
source /path/to/ENV/bin/activate
pip install pystog
```

Also, direnv is a useful and recommended way to manage the virtual environment. You can simply use an *.envrc* file which contains *layout python<version>* where <*version*> == 2 or 3 for the python version you would like (3 is recommended).

Then, once inside the development directory, just install via *pip* as described above.

2.5 Tests

From the parent directory of the module, run:

```
python tests/runner.py
```

CHAPTER 3

Modules

List of PyStoG modules:

3.1 Converter

This module defines the Converter class that converts functions in the same space

class pystog.converter.**Converter**

The Converter class is used to convert between either different reciprocal space functions or different real space functions

Examples

```
>>> import numpy
>>> from pystog import Converter
>>> converter = Converter()
>>> q, sq = numpy.loadtxt("my_sofq_file.txt",unpack=True)
>>> fq, dfq = converter.S_to_F(q, sq)
>>> r, gr = numpy.loadtxt("my_gofr_file.txt",unpack=True)
>>> kwargs = {'rho' : 1.0}
>>> gr_keen, dgr_keen = converter.g_to_GK(r, gr, **kwargs)
```

DCS_to_F(*q*, *dcs*, *ddcs=None*, ***kwargs*)

Convert $\frac{d\sigma}{d\Omega}(Q)$ to $Q[S(Q)-1]$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **dcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **ddcs** (*numpy.array or list*) – uncertainty vector

Returns ($Q[S(Q)-1]$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

DCS_to_FK(*q*, *dcs*, *ddcs=None*, ***kwargs*)
Convert $\frac{d\sigma}{d\Omega}(Q)$ to $F(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **ddcs** (*numpy.array or list*) – uncertainty vector

Returns ($F(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

DCS_to_S(*q*, *dcs*, *ddcs=None*, ***kwargs*)
Convert $\frac{d\sigma}{d\Omega}(Q)$ to $S(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **dcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **ddcs** (*numpy.array or list*) – uncertainty vector

Returns ($S(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

FK_to_DCS(*q*, *fq*, *dfq=None*, ***kwargs*)
Convert $F(Q)$ to $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $F(Q)$ vector
- **dfq** (*numpy.array or list*) – uncertainty vector

Returns ($\frac{d\sigma}{d\Omega}(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

FK_to_F(*q*, *fq_keen*, *dfq_keen=None*, ***kwargs*)
Convert $F(Q)$ to $Q[S(Q)-1]$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq_keen** (*numpy.array or list*) – $F(Q)$ vector
- **dfq_keen** (*numpy.array or list*) – uncertainty vector

Returns ($Q[S(Q)-1]$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

FK_to_S(*q*, *fq_keen*, *dfq_keen=None*, ***kwargs*)
Convert $F(Q)$ to $S(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq_keen** (*numpy.array or list*) – $F(Q)$ vector
- **dfq_keen** (*numpy.array or list*) – uncertainty vector

Returns ($S(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

F_to_DCS(*q*, *fq*, *dfq=None*, ***kwargs*)
Converts from $Q[S(Q)-1]$ to $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q)-1]$ vector
- **dfq** (*numpy.array or list*) – uncertainty vector

Returns ($\frac{d\sigma}{d\Omega}(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

F_to_FK(*q*, *fq*, *dfq=None*, ***kwargs*)
Converts from $Q[S(Q)-1]$ to $F(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q)-1]$ vector
- **dfq** (*numpy.array or list*) – uncertainty vector

Returns ($F(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

F_to_S(*q*, *fq*, *dfq=None*, ***kwargs*)
Converts from $Q[S(Q)-1]$ to $S(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q)-1]$ vector
- **dfq** (*numpy.array or list*) – uncertainty vector

Returns ($S(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

GK_to_G(*r*, *gr*, *dgr=None*, ***kwargs*)
Convert $G_{KeenVersion}(r)$ to $G_{PDFFIT}(r)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **dgr** (*numpy.array or list*) – uncertainty vector $\Delta g(r)$

Returns $G_{PDFFIT}(r)$ vector, uncertainty vector

Return type (numpy.array, numpy.array)

GK_to_g(*r*, *gr*, *dgr=None*, ***kwargs*)
Convert $G_{KeenVersion}(r)$ to $g(r)$

Parameters

- **r** (*numpy.array or list*) – r-space vector

- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **dgr** (*numpy.array or list*) – uncertainty vector $\Delta g(r)$

Returns $g(r)$ vector, uncertainty vector

Return type (numpy.array, numpy.array)

G_to_GK(*r*, *gr*, *dgr=None*, ***kwargs*)
Convert $G_{PDFFIT}(r)$ to $G_{KeenVersion}(r)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **dgr** (*numpy.array or list*) – uncertainty vector $\Delta g(r)$

Returns $G_{KeenVersion}(r)$ vector, uncertainty vector

Return type (numpy.array, numpy.array)

G_to_g(*r*, *gr*, *dgr=None*, ***kwargs*)
Convert $G_{PDFFIT}(r)$ to $g(r)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **dgr** (*numpy.array or list*) – uncertainty vector $\Delta g(r)$

Returns $g(r)$ vector, uncertainty vector

Return type (numpy.array, numpy.array)

S_to_DCS(*q*, *sq*, *dsq=None*, ***kwargs*)
Convert $S(Q)$ to $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **sq** (*numpy.array or list*) – $S(Q)$ vector
- **dsq** (*numpy.array or list*) – uncertainty vector

Returns ($\frac{d\sigma}{d\Omega}(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

S_to_F(*q*, *sq*, *dsq=None*, ***kwargs*)
Convert $S(Q)$ to $Q[S(Q)-1]$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **sq** (*numpy.array or list*) – $S(Q)$ vector
- **dfq** (*numpy.array or list*) – uncertainty vector
- **dsq** (*numpy.array or list*) – uncertainty vector

Returns ($Q[S(Q)-1]$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

S_to_FK(*q*, *sq*, *dsq=None*, ***kwargs*)
Convert $S(Q)$ to $F(Q)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **sq** (*numpy.array or list*) – $S(Q)$ vector
- **dsq** (*numpy.array or list*) – uncertainty vector

Returns ($F(Q)$ vector, uncertainty vector)

Return type (numpy.array, numpy.array)

g_to_G(*r*, *gr*, *dgr=None*, ***kwargs*)
Convert $g(r)$ to $G_{PDFFIT}(r)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **dgr** (*numpy.array or list*) – uncertainty vector $\Delta g(r)$

Returns $G_{PDFFIT}(r)$ vector, uncertainty vector

Return type (numpy.array, numpy.array)

g_to_GK(*r*, *gr*, *dgr=None*, ***kwargs*)
Convert $g(r)$ to $G_{KeenVersion}(r)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **dgr** (*numpy.array or list*) – uncertainty vector $\Delta g(r)$

Returns $G_{KeenVersion}(r)$ vector, uncertainty vector

Return type (numpy.array, numpy.array)

3.2 Transformer

This module defines the Transformer class that performs the Fourier transforms

class pystog.transformer.**Transformer**
The Transformer class is used to Fourier transform between the difference spaces. Either: a reciprocal space function -> real space function or a real space function -> reciprocal space function

Examples

```
>>> import numpy
>>> from pystog import Transformer
>>> transformer = Transformer()
>>> q, sq = numpy.loadtxt("my_sofq_file.txt",unpack=True)
>>> r = numpy.linspace(0., 25., 2500)
>>> r, gr, dgr = transformer.S_to_G(q, sq, r)
>>> q = numpy.linspace(0., 25., 2500)
>>> q, sq, dsq = transformer.G_to_S(r, gr, q)
```

DCS_to_G(*q*, *dcs*, *r*, *ddcs=None*, ***kwargs*)
Transforms from reciprocal space $\frac{d\sigma}{d\Omega}(Q)$ to real space $G_{PDFFIT}(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **dcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **ddcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ uncertainties

Returns r, $G_{PDFFIT}(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

DCS_to_GK(*q*, *dcs*, *r*, *ddcs=None*, ***kwargs*)
Transforms from reciprocal space $\frac{d\sigma}{d\Omega}(Q)$ to real space $G_{KeenVersion}(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **dcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **ddcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ uncertainties

Returns r, $G_{KeenVersion}(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

DCS_to_g(*q*, *dcs*, *r*, *ddcs=None*, ***kwargs*)
Transforms from reciprocal space $\frac{d\sigma}{d\Omega}(Q)$ to real space $g(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **dcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **ddcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ uncertainties

Returns r, $g(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

FK_to_G(*q*, *fq_keen*, *r*, *dfq_keen=None*, ***kwargs*)
Transforms from reciprocal space $F(Q)$ to real space $G_{PDFFIT}(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq_keen** (*numpy.array or list*) – $F(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dfq_keen** (*numpy.array or list*) – $F(Q)$ vector uncertainties

Returns r, $G_{PDFFIT}(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

FK_to_GK(*q*, *fq_keen*, *r*, *dfq_keen=None*, ***kwargs*)
Transforms from reciprocal space $F(Q)$ to real space $G_{KeenVersion}(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq_keen** (*numpy.array or list*) – $F(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dfq_keen** (*numpy.array or list*) – $F(Q)$ vector uncertainties

Returns r, $G_{KeenVersion}(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

FK_to_g(*q*, *fq_keen*, *r*, *dfq_keen=None*, ***kwargs*)
Transforms from reciprocal space $F(Q)$ to real space $g(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq_keen** (*numpy.array or list*) – $F(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dfq_keen** (*numpy.array or list*) – $F(Q)$ vector uncertainties

Returns r, $g(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

F_to_G(*q*, *fq*, *r*, *dfq=None*, ***kwargs*)
Transforms from reciprocal space $Q[S(Q)-1]$ to real space $G_{PDFFIT}(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q)-1]$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dfq** (*numpy.array or list*) – uncertainty on $Q[S(Q)-1]$

Returns r, $G_{PDFFIT}(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

F_to_GK(*q*, *fq*, *r*, *dfq=None*, ***kwargs*)
Transforms from reciprocal space $Q[S(Q)-1]$ to real space $G_{KeenVersion}(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q)-1]$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dfq** (*numpy.array or list*) – uncertainty on $Q[S(Q)-1]$

Returns r, $G_{KeenVersion}(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

F_to_g(*q*, *fq*, *r*, *dfq=None*, ***kwargs*)
Transforms from reciprocal space $Q[S(Q)-1]$ to real space $g(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q)-1]$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dfq** (*numpy.array or list*) – uncertainty on $Q[S(Q)-1]$

Returns r, $g(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

GK_to_DCS(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $G_{KeenVersion}(r)$ to reciprocal space $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ uncertainties

Returns Q, $\frac{d\sigma}{d\Omega}(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

GK_to_F(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $G_{KeenVersion}(r)$ to reciprocal space $Q[S(Q)-1]$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ uncertainties

Returns Q, $Q[S(Q)-1]$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

GK_to_FK(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $G_{KeenVersion}(r)$ to reciprocal space $F(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ uncertainties

Returns Q, $F(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

GK_to_S(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $G_{KeenVersion}(r)$ to reciprocal space $S(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector

- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ uncertainties

Returns Q, $S(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

G_to_DCS(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $G_{PDFFIT}(r)$ to reciprocal space $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ uncertainties

Returns Q, $\frac{d\sigma}{d\Omega}(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

G_to_F(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $G_{PDFFIT}(r)$ to reciprocal space $Q[S(Q) - 1]$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ uncertainties

Returns Q, $Q[S(Q) - 1]$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

G_to_FK(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $G_{PDFFIT}(r)$ to reciprocal space $F(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ uncertainties

Returns Q, $F(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

G_to_S(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $G_{PDFFIT}(r)$ to reciprocal space $S(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector

- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ uncertainties

Returns Q, $S(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

S_to_G(*q*, *sq*, *r*, *dsq=None*, ***kwargs*)
Transforms from reciprocal space $S(Q)$ to real space $G_{PDFFIT}(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **sq** (*numpy.array or list*) – $S(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dsq** (*numpy.array or list*) – $S(Q)$ uncertainties

Returns r, $G_{PDFFIT}(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

S_to_GK(*q*, *sq*, *r*, *dsq=None*, ***kwargs*)
Transforms from reciprocal space $S(Q)$ to real space $G_{KeenVersion}(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **sq** (*numpy.array or list*) – $S(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dsq** (*numpy.array or list*) – $S(Q)$ uncertainties

Returns r, $G_{KeenVersion}(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

S_to_g(*q*, *sq*, *r*, *dsq=None*, ***kwargs*)
Transforms from reciprocal space $S(Q)$ to real space $g(r)$

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **sq** (*numpy.array or list*) – $S(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector
- **dsq** (*numpy.array or list*) – $S(Q)$ uncertainties

Returns r, $g(r)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

apply_cropping(*x*, *y*, *xmin*, *xmax*, *dy=None*)
Utility to crop x and y based on xmin and xmax along x. Provides the capability to specify the (Qmin,Qmax) or (Rmin,Rmax) in the Fourier transform

Parameters

- **x** (*numpy.array or list*) – domain vector
- **y** (*numpy.array or list*) – range vector
- **xmin** (*float*) – minimum x-value for crop

- **xmax** (*float*) – maximum x-value for crop
- **dy** (*numpy.array or list*) – uncertainty vector

Returns vector pair (x,y) with cropping applied

Return type (numpy.array, numpy.array, numpy.array)

fourier_transform(*xin*, *yin*, *xout*, *xmin=None*, *xmax=None*, *dyin=None*, ***kwargs*)
The Fourier transform function. The kwargs argument allows for different modifications: Lorch dampening, omitted low-x range correction,

Parameters

- **xin** (*numpy.array or list*) – domain vector for space to be transformed from
- **yin** (*numpy.array or list*) – range vector for space to be transformed from
- **xout** (*numpy.array or list*) – domain vector for space to be transformed to
- **xmin** (*float*) – minimum x-value for crop
- **xmax** (*float*) – maximum x-value for crop
- **dyin** (*numpy.array or list*) – uncertainty vector for yin

Returns vector pair of transformed domain, range vectors, and uncertainties

Return type numpy.array, numpy.array, numpy.array

g_to_DCS(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $g(r)$ to reciprocal space $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $g(r)$ uncertainties

Returns Q, $\frac{d\sigma}{d\Omega}(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

g_to_F(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $g(r)$ to reciprocal space $Q[S(Q)-1]$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $g(r)$ uncertainties

Returns Q, $Q[S(Q)-1]$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

g_to_FK(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $g(r)$ to reciprocal space $F(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector

- **gr** (*numpy.array or list*) – $g(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $g(r)$ uncertainties

Returns Q, $F(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

g_to_S(*r*, *gr*, *q*, *dgr=None*, ***kwargs*)
Transforms from real space $g(r)$ to reciprocal space $S(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dgr** (*numpy.array or list*) – $g(r)$ uncertainties

Returns Q, $S(Q)$, and uncertainties

Return type numpy.array, numpy.array, numpy.array

3.3 FourierFilter

This module defines the FourierFilter class that performs the Fourier filter for a given range to exclude.

class pystog.fourier_filter.**FourierFilter**
The FourierFilter class is used to exlude a given range in the current function by a back Fourier Transform of that section, followed by a difference from the non-excluded function, and then a forward transform of the difference function Can currently do: a real space function -> reciprocal space function -> real space function

Examples

```
>>> import numpy
>>> from pystog import FourierFilter
>>> ff = FourierFilter()
>>> r, gr = numpy.loadtxt("my_gofr_file.txt",unpack=True)
>>> q = numpy.linspace(0., 25., 2500)
>>> q, sq = transformer.G_to_S(r, gr, q)
>>> q_ft, sq_ft, q, sq, r, gr = ff.G_using_F(r, gr, q, sq, 1.5)
```

GK_using_DCS(*r*, *gr*, *q*, *dcs*, *cutoff*, *dgr=None*, *ddcs=None*, ***kwargs*)
Fourier filters real space $G_{KeenVersion}(r)$ using the reciprocal space $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q and $\frac{d\sigma}{d\Omega}(Q)$ for the 0. to cutoff transform, the corrected Q and $\frac{d\sigma}{d\Omega}(Q)$, and the filtered r and $G_{KeenVersion}(r)$.

Thus, [Q_{FF}, $\frac{d\sigma}{d\Omega}(Q)_{FF}$, Q, $\frac{d\sigma}{d\Omega}(Q)$, r_{FF}, $G_{KeenVersion}(r)_{FF}$]

Return type tuple of numpy.array

GK_using_F(*r*, *gr*, *q*, *fq*, *cutoff*, *dgr=None*, *dfq=None*, ***kwargs*)
Fourier filters real space $G_{KeenVersion}(r)$ using the reciprocal space $Q[S(Q) - 1]$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q) - 1]$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q **and** $Q[S(Q) - 1]$ for the 0. to cutoff transform, the corrected Q and $Q[S(Q) - 1]$, and the filtered r and $G_{KeenVersion}(r)$.

Thus, [

$$Q_{FF}, Q[S(Q) - 1]_{FF}, Q, Q[S(Q) - 1], r_{FF}, G_{KeenVersion}(r)_{FF}$$

]

Return type tuple of numpy.array

GK_using_FK(*r*, *gr*, *q*, *fq*, *cutoff*, *dgr=None*, *dfq=None*, ***kwargs*)
Fourier filters real space $G_{KeenVersion}(r)$ using the reciprocal space $F(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $F(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q **and** $F(Q)$ for the 0. to cutoff transform, the corrected Q and $F(Q)$, and the filtered r and $G_{KeenVersion}(r)$.

Thus, [

$$Q_{FF}, F(Q)_{FF}, Q, F(Q), r_{FF}, G_{KeenVersion}(r)_{FF}$$

]

Return type tuple of numpy.array

GK_using_S(*r*, *gr*, *q*, *sq*, *cutoff*, *dgr=None*, *dsq=None*, ***kwargs*)
Fourier filters real space $G_{KeenVersion}(r)$ using the reciprocal space $S(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{KeenVersion}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $S(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q and $S(Q)$ for the 0. to cutoff transform, the corrected Q and $S(Q)$, and the filtered r and $G_{KeenVersion}(r)$.

Thus, [

$$Q_{FF}, S(Q)_{FF}, Q, S(Q), r_{FF}, G_{KeenVersion}(r)_{FF}$$

]

Return type tuple of numpy.array

G_using_DCS(*r*, *gr*, *q*, *dcs*, *cutoff*, *dgr=None*, *ddcs=None*, ***kwargs*)

Fourier filters real space $G_{PDFFIT}(r)$ using the reciprocal space $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **dcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q and $\frac{d\sigma}{d\Omega}(Q)$ for the 0. to cutoff transform, the corrected Q and $\frac{d\sigma}{d\Omega}(Q)$, and the filtered r and $G_{PDFFIT}(r)$.

Thus, $[Q_{FF}, \frac{d\sigma}{d\Omega}(Q)_{FF}, Q, \frac{d\sigma}{d\Omega}(Q), r_{FF}, G_{PDFFIT}(r)_{FF}]$

Return type tuple of numpy.array

G_using_F(*r*, *gr*, *q*, *fq*, *cutoff*, *dgr=None*, *dfq=None*, ***kwargs*)

Fourier filters real space $G_{PDFFIT}(r)$ using the reciprocal space $Q[S(Q) - 1]$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q) - 1]$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q and $Q[S(Q) - 1]$ for the 0. to cutoff transform, the corrected Q and $Q[S(Q) - 1]$, and the filtered r and $G_{PDFFIT}(r)$.

Thus, [

$$Q_{FF}, Q[S(Q) - 1]_{FF}, Q, Q[S(Q) - 1], r_{FF}, G_{PDFFIT}(r)_{FF}$$

]

Return type tuple of numpy.array

G_using_FK(*r*, *gr*, *q*, *fq*, *cutoff*, *dgr=None*, *dfq=None*, ***kwargs*)
Fourier filters real space $G_{PDFFIT}(r)$ using the reciprocal space $F(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $F(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q **and** $F(Q)$ for the 0. to cutoff transform, the corrected Q and $F(Q)$, and the filtered r and $G_{PDFFIT}(r)$.

Thus, [

Q_{FF}, $F(Q)_{FF}$, Q, $F(Q)$, r_{FF}, $G_{PDFFIT}(r)_{FF}$

]

Return type tuple of numpy.array

G_using_S(*r*, *gr*, *q*, *sq*, *cutoff*, *dgr=None*, *dsq=None*, ***kwargs*)
Fourier filters real space $G_{PDFFIT}(r)$ using the reciprocal space $S(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $G_{PDFFIT}(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $S(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q **and** $S(Q)$ for the 0. to cutoff transform, the corrected Q and $S(Q)$, and the filtered r and $G_{PDFFIT}(r)$.

Thus, [

Q_{FF}, $S(Q)_{FF}$, Q, $S(Q)$, r_{FF}, $G_{PDFFIT}(r)_{FF}$

]

Return type tuple of numpy.array

g_using_DCS(*r*, *gr*, *q*, *dcs*, *cutoff*, *dgr=None*, *ddcs=None*, ***kwargs*)
Fourier filters real space $g(r)$ using the reciprocal space $\frac{d\sigma}{d\Omega}(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector

- **dcs** (*numpy.array or list*) – $\frac{d\sigma}{d\Omega}(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q and $\frac{d\sigma}{d\Omega}(Q)$ for the 0. to cutoff transform, the corrected Q and $\frac{d\sigma}{d\Omega}(Q)$, and the filtered r and $g(r)$.

Thus, [Q_{FF}, $\frac{d\sigma}{d\Omega}(Q)_{FF}$, Q, $\frac{d\sigma}{d\Omega}(Q)$, r_{FF}, $g(r)_{FF}$]

Return type tuple of numpy.array

g_using_F(*r*, *gr*, *q*, *fq*, *cutoff*, *dgr=None*, *dfq=None*, ***kwargs*)
Fourier filters real space $g(r)$ using the reciprocal space $Q[S(Q) - 1]$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $Q[S(Q) - 1]$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q **and** $Q[S(Q) - 1]$ for the 0. to cutoff transform, the corrected Q and $Q[S(Q) - 1]$, and the filtered r and $g(r)$.

Thus, [

$$Q_{FF}, Q[S(Q) - 1]_{FF}, Q, Q[S(Q) - 1], r_{FF}, g(r)_{FF}$$

]

Return type tuple of numpy.array

g_using_FK(*r*, *gr*, *q*, *fq*, *cutoff*, *dgr=None*, *dfq=None*, ***kwargs*)
Fourier filters real space $g(r)$ using the reciprocal space $F(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **fq** (*numpy.array or list*) – $F(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q and $F(Q)$ for the 0. to cutoff transform, the corrected Q and $F(Q)$, and the filtered r and $g(r)$.

Thus, [Q_{FF}, $F(Q)_{FF}$, Q, $F(Q)$, r_{FF}, $g(r)_{FF}$]

Return type tuple of numpy.array

g_using_S(*r*, *gr*, *q*, *sq*, *cutoff*, *dgr=None*, *dsq=None*, ***kwargs*)
Fourier filters real space $g(r)$ using the reciprocal space $S(Q)$

Parameters

- **r** (*numpy.array or list*) – r-space vector
- **gr** (*numpy.array or list*) – $g(r)$ vector
- **q** (*numpy.array or list*) – Q-space vector
- **sq** (*numpy.array or list*) – $S(Q)$ vector
- **cutoff** (*float*) – The r_{max} value to filter from 0. to cutoff

Returns

A tuple of the Q and $S(Q)$ for the 0. to cutoff transform, the corrected Q and $S(Q)$, and the filtered r and $g(r)$.

Thus, [Q_{FF}, $S(Q)_{FF}$, Q, $S(Q)$, r_{FF}, $g(r)_{FF}$]

Return type tuple of numpy.array

3.4 StoG

This module defines the StoG class that tries to replicate the previous stog program behavior in an organized fashion with the ability to re-construct the workflow.

exception pystog.stog.**NoInputFilesException**
Exception when no files are given to process

class pystog.stog.**StoG**(***kwargs*)
The StoG class is used to put together the Converter, Transformer, and FourierFilter class functionalities to reproduce the original **stog** Fortran program behavior. This class is meant to put together the functionality of the classes into higher-level calls to construct workflows for merging and processing multiple recprical space functions into a final output real space function.

This pythonized-version of the original Fortran **stog** uses numpy for data storage, organization, and manipulation "under the hood".

Examples

```
>>> import json
>>> from pystog import StoG
>>> with open("../data/examples/argon_pystog.json", 'r') as f:
>>>     kwargs = json.load(f)
>>> stog = StoG(**kwargs)
>>> stog.read_all_data()
>>> stog.merge_data()
>>> stog.write_out_merged_data()
```

GKofR_title
The title of the $G_{KeenVersion}(r)$ with all corrections applied.

Getter Returns the current title for this function

Setter Sets the title for this function

Type str

add_dataset(*info*, *yscale=1.0*, *yoffset=0.0*, *xoffset=0.0*, *ydecimals=16*, ***kwargs*)
Takes the info with the dataset and manipulations, such as scales, offsets, cropping, etc., and creates an invidual numpy array for the pattern.

Parameters

- **info** (*dict*) – Dict with information for dataset (filename, manipulations, etc.)
- **yscale** (*float*) – Scale factor for the Y data (i.e. $S(Q)$, $F(Q)$, etc.)
- **yoffset** (*float*) – Offset factor for the Y data (i.e. $S(Q)$, $F(Q)$, etc.)
- **xoffset** – Offset factor for the X data (i.e. Q)

append_file(*new_file*)
Appends a file to the file list

Parameters **new_file** (*str*) – New file name to append

Returns File list with appended new_file

Return type list

apply_lorch(*q*, *sq*, *r*)
Performs the Fourier transform using the Lorch dampening correction on the merged $S(Q)$ from the **sq_master** dictionary to generate the desired real space function with this correction. The results from both reciprocal space and real space are:

1. Saved back to the respective "master" dictionaries
2. Saved to files via the **stem_name**
3. Returned from function

Parameters

- **q** (*numpy.array or list*) – Q-space vector
- **sq** (*numpy.array or list*) – $S(Q)$ vector
- **r** (*numpy.array or list*) – r-space vector

Returns Returns a tuple with r and selected real space function

Return type tuple of numpy.array

static apply_scales_and_offset(*x*, *y*, *dy=None*, *yscale=1.0*, *yoffset=0.0*, *xoffset=0.0*)
Applies scales to the Y-axis and offsets to both X and Y axes.

Parameters

- **x** (*numpy.array or list*) – X-axis data
- **y** (*numpy.array or list*) – Y-axis data
- **yscale** (*float*) – Y-axis scale factor
- **yoffset** (*float*) – Y-axis offset factor
- **xoffset** (*float*) – X-axis offset factor

Returns X and Y vectors after scales and offsets applied

Return type numpy.array pair

bcoh_sqrd
The average coherent scattering length, squared: $\langle b_{coh} \rangle^2 = (\sum_i c_i b_{coh,i})(\sum_i c_i b^*_{coh,i})$ where the subscript i implies for atom type i, c_i is the concentration of i, $b_{coh,i}$ is the coherent scattering length of i, and $b^*_{coh,i}$ is the complex coherent scattering length of i

The real part of the $b_{coh,i}$ term can be found from the **Coh b** column of the NIST neutron scattering length and cross section table found here: https://www.ncnr.nist.gov/resources/n-lengths/list.html

Units are in fm in the table for the $b_{coh,i}$ term. Thus, $\langle b_{coh} \rangle^2$ has units of fm^2 (and what PyStoG expects). NOTE: 100 fm^2 == 1 $barn$.

Getter Return the value of $\langle b_{coh} \rangle^2$

Setter Set the value for $\langle b_{coh} \rangle^2$

Type float

btot_sqrd

The average coherent scattering length, squared: $\langle b_{tot}^2 \rangle = \sum_i c_i b_{tot,i}^2 = \frac{1}{4\pi} \sum_i c_i \sigma_{tot,i}$ where the subscript i implies for atom type i, c_i is the concentration of i and $\sigma_{tot,i}$ is the total cross-section of i

The real part of the $b_{coh,i}$ term can be found from the **Scatt xs** column of the NIST neutron scattering length and cross section table found here: https://www.ncnr.nist.gov/resources/n-lengths/list.html

Units are in $barn$ (=100 fm^2) in the table for the $\sigma_{tot,i}$ term. Thus, you must multiply $\sum_i c_i b_{tot,i}^2$ by 100 to go from $barn$ to fm^2 (what PyStoG expects).

Getter Return the value of $\sum_i c_i b_{tot,i}^2$

Setter Set the value for $\sum_i c_i b_{tot,i}^2$

Type float

converter = None

The **converter** attribute defines the Converter which is used to do all inner conversions necessary to go from the input reciprocal space functions, produce diagnostics for the selected real space functions, and finally output the desired real space function Must of type *pystog.converter.Converter*

density

The number density used (atoms/^{3}) used for the ρ_0 term in the equations

Getter Return the density value

Setter Set the density value

Type float

dr

The real space function X axis data, r-space vector

Getter Return the r vector

Setter Set the r vector

Type numpy.array

extend_file_list(*new_files*)

Extend the file list with a list of new files

Parameters **new_files** – List of new files

Returns File list extended by new_files

Return type list

extract(*hdf_file*, *path*, *index=None*)

extract_path_from_title(*hdf_file*, *title*, *title_path='title'*, *wksp_path='workspace'*)

extract_xy(*hdf_file*, *xpath*, *ypath*, ***kwargs*)

files

The files that contain the reciprocal space data to merge together.

Getter Current list of files to merge

Setter Set the list of files to merg

Type list

filter = None
The **filter** attribute defines the FourierFilter which is used to do all Fourier filtering if the **fourier_filter_cutoff** attribute is supplied. Must of type *pystog.fourier_filter.FourierFilter*

fourier_filter()
Performs the Fourier filter on the **sq_master** pattern to generate the desired real space function with this correction. The results from both reciprocal space and real space are:

1. Saved back to the respective "master" dictionaries
2. Saved to files via the **stem_name**
3. Returned from function

Returns Returns a tuple with r, the selected real space function, Q, and $S(Q)$ functions

Return type tuple of numpy.array

fourier_filter_cutoff
This sets the cutoff in r-space for the Fourier filter. The minimum is automatically 0.0. Thus, from 0.0 to **fourier_filter_cutoff** is reverse transfomed, subtracted in reciprocal space, and then the difference is back-transformed.

See *pystog.fourier_filter.FourierFilter* for more information.

Getter Return currently set cutoff value

Setter Set cutoff value

Type float

fq_title
The title of the $F(Q)$ function after merging and a fourier filter correction.

Getter Returns the current title for this function

Setter Sets the title for this function

Type str

gr_ft_title
The title for the real space function after both merging and a fourier filter correction

Getter Returns the current title for this function

Setter Sets the title for this function

Type str

gr_lorch_title
The title for the real space function with the lorch correction

Getter Returns the current title for this function

Setter Sets the title for this function

Type str

gr_master
The "master" dictionary for the real space functions that are generated for each processing step.

Getter Returns the current "master" real space functions dictionary generated up to the current step in the workflow.

Setter Sets the "master" real space function dictionary

Type dict[str:numpy.ndarray]

gr_title
The title of the real space function directly after merging the reciprocal space functions without any further corrections.

Getter Returns the current title for this function

Setter Sets the title for this function

Type str

lorch_flag
This sets the option to perform the Lorch dampening correction for the Q range. Generally, will help reduce Fourier "ripples", or AKA "Gibbs phenomenon", due to discontinuity at Q_{max} if the reciprocal space function is not at the $Q- > \infty$ limit. Yet, will also broaden real space function peaks, possibly dubiously.

See *pystog.transformer.Transformer* **fourier_transform** and **_low_x_correction** methods for where this is applied.

Getter Return bool of applying the Lorch dampening correction

Setter Set whether the correction is applied or not

Type bool

low_q_correction
This sets the option to perform a low-Q correction for the omitted Q range.

See *pystog.transformer.Transformer* **_low_x_correction** method for more information.

Getter Return bool of applying the low-Q correction

Setter Set whether the correction is applied or not

Type bool

merge_data()
Merges the reciprocal space data stored in the **reciprocal_individuals** numpy array into a single, merged recirocal space function. Stores the S(Q) result in **sq_master** dictionary using **sq_title** (default: "S(Q) Merged").

Also, converts this merged $S(Q)$ into $Q[S(Q) - 1]$ via the **Converter** class and applies any modification specified in **merged_opts** dict attribute, specified by the **'Q[S(Q)-1]'** key of the dict. If there is modification, this modified $Q[S(Q) - 1]$ will be converted to $S(Q)$ and replace the $S(Q)$ directly after merge.

Example dict of **merged_opts** for scaling of $S(Q)$ by 2 and then offsetting $Q[S(Q) - 1]$ by 5:

```
{
    "Merging": {
        "Y": {
            "Offset": 0.0,
            "Scale": 2.0
        },
        "Q[S(Q)-1]": {
            "Y": "Offset": 5.0,
            "Scale": 1.0
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

...

merged_opts
This sets the options to perform after merging the reciprocal space functions together, such as an overall offset and scale.

Getter Return the options currently set

Setter Set the options for the merged pattern

Type dict

q_master
The "master" dictionary for the domain :math:Q‘ of the reciprocal space functions that are generated for each processing step.

Getter Returns the current "master" Q reciprocal space functions dictionary generated up to the current step in the workflow.

Setter Sets the "master" Q dictionary

Type dict[str:numpy.ndarray]

qmax
The Q_{max} value to use for the Fourier transforms (from recirocal space -> real space). This overrides **xmax** attribute if **qmax < xmax**.

Getter Returns the current set value

Setter Set the qmax value for the Fourier transforms

Type float

qmin
The Q_{min} value to use for the Fourier transforms (from recirocal space -> real space). This overrides **xmin** attribute if **xmin < qmin**.

Getter Returns the current set value

Setter Set the Q_{min} value for the Fourier transforms

Type float

qsq_minus_one_title
The title of the $Q[S(Q) - 1]$ function directly after merging the reciprocal space functions without any further corrections.

Getter Returns the current title for this function

Setter Sets the title for this function

Type str

r_master
The "master" dictionary for the domain :math:r‘ of the real space functions that are generated for each processing step.

Getter Returns the current "master" r real space functions dictionary generated up to the current step in the workflow.

Setter Sets the "master" r dictionary

Type dict[str:numpy.ndarray]

rdelta
The Δr for the r-space vector

Getter Return Δr value

Setter Set the Δr value and update r-space vector via the **dr** attribute

Type value

read_all_data(***kwargs*)
Reads all the data from the **files** attribute Uses the **read_dataset** method on each file.

Will append all datasets to the numpy storage array, **reciprocal_individuals**, and also convert to $S(Q)$ and add to the **sq_individuals** numpy storage array in **add_dataset** method via **read_dataset** method.

read_all_nexus_file_banks()

read_dataset(*info*, *xcol=0*, *ycol=1*, *dycol=2*, *sep='\\s+'*, *skiprows=2*, ***kwargs*)
Reads an individual file and uses the **add_dataset** method to apply all dataset manipulations, such as scales, offsets, cropping, etc.

Parameters

- **info** (*dict*) – Dict with information for dataset (filename, manipulations, etc.)
- **xcol** (*int*) – Column in data file for X-axis
- **ycol** (*int*) – Column in data file for Y-axis
- **dycol** (*int*) – Column in data file for Y uncertainty
- **sep** (*raw string*) – Separator for the file used by numpy.loadtxt
- **skiprows** (*int*) – Number of rows to skip. Passed to numpy.loadtxt

read_nexus_file_by_bank(*nexus_file*, *bank*, *title*, ***kwargs*)
Reads an individual file bank by bank and uses the **extract_xy** to handle extraction and **extract_path_from_title** to obtain coord paths

real_space_function
The real space function to use throughoutt the processing

Getter Returns the currently select real space function

Setter Set the selected real space function and updates other title attributes that rely on this in their name.

Type str

reciprocal_individuals
The storage array for the input reciprocal space functions loaded from **files** and with the loading processing from **add_dataset** class method.

Getter Returns the current individual, input reciprocal space functions numpy array. The dimenions is $3xN * M$ where N is the number of patterns stored and M is the length of the patterns.

Setter Sets the numpy array

Type numpy.ndarray

rmax
The R_{max} valuefor the r-space vector

Getter Return R_{max} value

Setter Set the R_{max} value and update r-space vector via the **dr** attribute

Type value

rmin
The R_{min} valuefor the r-space vector

Getter Return R_{min} value

Setter Set the R_{min} value and update r-space vector via the **dr** attribute

Type value

save_xy (*filename*, *xdata*, *ydata*)

sq_ft_title
The title of the $S(Q)$ function after merging and a fourier filter correction.

Getter Returns the current title for this function

Setter Sets the title for this function

Type str

sq_individuals
The storage array for the $S(Q)$ generated from each input reciprocal space dataset in **reciprocal_individuals** array.

Getter Returns the current individual $S(Q)$ reciprocal space functions numpy array. The dimenions is $3xN * M$ where N is the number of patterns stored and M is the length of the patterns.

Setter Sets the numpy array

Type numpy.ndarray

sq_master
The "master" dictionary for the $S(Q)$ reciprocal space functions that are generated for each processing step.

Getter Returns the current "master" $S(Q)$ reciprocal space functions dictionary generated up to the current step in the workflow.

Setter Sets the "master" $S(Q)$ function dictionary

Type dict[str:numpy.ndarray]

sq_title
The title of the $S(Q)$ function directly after merging the reciprocal space functions without any further corrections.

Getter Returns the current title for this function

Setter Sets the title for this function

Type str

stem_name
A stem name to prefix for all output files. Replicates the **stog** Fortran program behavior.

Getter Return the currently set stem name

Setter Set the stem name for output files

Type str

transform_merged()
Performs the Fourier transform on the merged **sq_master** pattern to generate the desired real space function with this correction. The results for real space are: the domain is saved to the **r_master** dictionary and the range is saved to the **gr_master** dictionary, with both using the **gr_title** for the key of the dictionaries.

transformer = None
The **transformer** attribute defines the Transformer which is used to do all Fourier transforms necessary from reciprocal space to real space and vice versa. Must of type `pystog.transformer.Transformer`

write_out_ft(*filename=None*)
Helper function for writing out the Fourier filter correction.

Parameters **filename** (*str*) – Filename to write to

write_out_ft_gr(*filename=None*)
Helper function for writing out the Fourier filtered real space function

Parameters **filename** (*str*) – Filename to write to

write_out_ft_sq(*filename=None*)
Helper function for writing out the Fourier filtered $S(Q)$

Parameters **filename** (*str*) – Filename to write to

write_out_lorched_gr(*filename=None*)
Helper function for writing out the Lorch dampened real space function

Parameters **filename** (*str*) – Filename to write to

write_out_merged_gr(*filename=None*)
Helper function for writing out the merged real space function

Parameters **filename** (*str*) – Filename to write to

write_out_merged_sq(*filename=None*)
Helper function for writing out the merged $S(Q)$

Parameters **filename** (*str*) – Filename to write to

write_out_rmc_fq(*filename=None*)
Helper function for writing out the output $F(Q)$

Parameters **filename** (*str*) – Filename to write to

write_out_rmc_gr(*filename=None*)
Helper function for writing out the output $G_{KeenVersion}(Q)$

Parameters **filename** (*str*) – Filename to write to

xmax
The maximum X value of all datasets to use for Fourier transforms (from recirocal space -> real space)

Getter Returns the current set value

Setter Set the xmax value for the Fourier transforms

Type float

xmin
The minimum X value of all datasets to use for Fourier transforms (from recirocal space -> real space)

Getter Returns the current set value

Setter Set the xmin value for the Fourier transforms

Type float

Python Module Index

p

Index

L

M

N

P

Q

R

S

T

W

X