EOS

python script to reduce reflectivity data
for Amor @ SINQ, PSI

Jochen Stahn

2024-03-15

---

eos is a python program used at the neutron reflectometer *Amor* at *SINQ, Paul Scherrer Institut, Switzerland* to turn *raw data* into reduced an *orso* compatible *reflectivity file*.

raw: *nexus hdf5* format:

- event arrays for detector and monitor events
- arrays for device propertise
- entries for instrument configuration

reduced: *orso reflectivity* format:

- header with information about
  - data origin
  - measurement conditions
  - reduction steps
- array with basic or expanded reflectivity data (in the simplest version: the *reflectivity curve*)

---

environment

eos (version 2.0 and later) was developen with python3.9.

The following (non-trivial) python modules are required:

- `numpy`
- `h5py`
- `orsopy`
- `numba`

usage

eos can read one or several *.hdf* files for one or several instrument settings, and it creates one or several reflectivity curves or intensity maps.

> **Warning**: Choosing wrong combinations can easily lead to huge data files. E.g. time-slizing and output of intensity maps might use several hundred GB.

eos is using command line arguments to

- find the raw data
- overwrite default values
- define the parameter range for reduction
- define reduction steps
- define the output path and name

communicate file numbers and path information

```
input data:
  -f FILEIDENTIFIER [FILEIDENTIFIER ...], --fileIdentifier FILEIDENTIFIER [FILEIDENTIFIER ..
                        file number(s) or offset (if negative)
  -n NORMALISATIONFILEIDENTIFIER [NORMALISATIONFILEIDENTIFIER ...], --normalisationFileIdent
                        file number(s) of normalisation measurement
  -d DATAPATH, --dataPath DATAPATH
                        relative path to directory with .hdf files
  -Y YEAR, --year YEAR  year the measurement was performed
  -sub SUBTRACT, --subtract SUBTRACT
                        R(q_z) curve to be subtracted (in .Rqz.ort format
```

minimum

purposes:

- fast access to human-readable meta data in the output header
- get an idea about q_z range and statistics

actions:

- read in one raw data file
- convert the event stream into an $I(\lambda, \alpha_f)$ map
- project this map onto q_z to give an $I(q_z)$ curve
- write this curve in *orso* format to disk

example: `> python eos.py -f 456 -o foo`
looks for the file `amor<year>n000456.hdf` in one of the default locations (`./`, `./raw/`, `../raw`, local raw data directory on Amor) and writes the output to `foo.Rqz.ort`.

with normalisation

purposes:

- fast access to human-readable meta data in the output header
- get a reduced and (partially) corrected reflectivity curve

actions:

- read in raw data file(s) and raw normalisation file(s)
- convert the normalisation measurement into a $N(\lambda, z_{\text{detector}})$ map containing information about guide and detector efficiencies, illuminated detector area and incoming intensity.
- convert the event stream into an $I(\lambda, \alpha_f)$ map
- normalisation: $R(\lambda, \alpha_f)_{la} = I(\lambda, \alpha_f)_{la} / N(\lambda, z_{\text{detector}})_{la}$.
- project this map onto q_z to give a $R(q_z)$ curve (not necessarily scaled)
- write this curve in *orso* format to disk

example: `> python eos.py -f 456 -n 123 -o foo`
looks for the files `amor<year>n000456.hdf` (reflectivity) and `amor<year>n000123.hdf` (normalisation) in one of the default locations (`./`, `./raw/`, `../raw`, local raw data directory on Amor) and writes the output to `foo.Rqz.ort`.

read multiple files

- **for the same instrument parameter set**

  The arguments of the keys `-f` and `-n` have the general form
  `<start1>[-<end1>[:<increment1]][,<start2>[-<end2>[:<increment2]],...]`
  Each number range is defined by a start value, an optional stop value and an optional increment. Various ranges are separated by a ‘,’.

  example: `20-25:2,28-30,40`
  resolves into the list
  `[20, 22, 24, 28, 29, 30, 40]`

  action: Effectively, the event streams found in the the various files are merged and processed together.

- **for different parameter sets, or to prevent merging**

  The key `-f` accepts more than one argument of the type defined above. The (set of) data file(s) related to one argument are merged and give one reflectivity curve (one `data_set`) in the output file. The reflectivity curves for more than one argument are separated in the output file by the separator `# data-set: <identifier>`.

example: `> python eos.py -f 20,21 30 -n 123 -o foo`
results in two reflectivity curves, the first made from files #20 and #21, the second from file #30. Both are saved in `foo.Rqz.ort`.

warning: `-n` does accept only one argument!

misc.

year The raw file name is created using the file number and the actual year. In case the data to be processed were recorded in a previous year, this must be explicitely stated with
`-Y <year>`.

path The default location for the output (and for starting the search for the input files) iis the present working directory. This can be altered by using the argument
`-d <path>`.

subtract q_z-dependent curve It is possible to provide a $R(q_z)$ curve in `.Rqz.ort` format to be subtracted from the reduced data. E.g. to emphasize the high-q region on a linear scale, or to illustrate changes in a series of measurements. The argument is
`-sub <filename>`.

---

output options

```
output:
  -o OUTPUTNAME, --outputName OUTPUTNAME
                        output file name (withot suffix)
  -of OUTPUTFORMAT [OUTPUTFORMAT ...], --outputFormat OUTPUTFORMAT [OUTPUTFORMAT ...]
  --offSpecular OFFSPECULAR
  -r QRESOLUTION, --qResolution QRESOLUTION
                        q_z resolution
  -ts TIMESLIZE [TIMESLIZE ...], --timeSlize TIMESLIZE [TIMESLIZE ...]
                        time slizing <interval> ,[<start> [,stop]]
  -s SCALE [SCALE ...], --scale SCALE [SCALE ...]
                        scaling factor for R(q_z)
  -S AUTOSCALE AUTOSCALE, --autoscale AUTOSCALE AUTOSCALE
                        scale to 1 in the given q_z range
```

output formats

Besides the default *orso* format `.Rqz.ort`, there is the option to write the $R(\lambda, \alpha_f)$ array and the related input, normalisation and mask arrays. This

output can help with the sample alignment or the readjustment of parameters (see below), or it can be used for debugging the data processing or instrument operation. The suffix of this output is `.Rlt.ort`

The format is chosen by using one or several of the arguments

- `Rqz.ort`, `Rlt.ort`, `Rqz.orb`, `Rlt.orb`
- `Rqz` (= `Rqz.ort` and `Rqz.orb`)
- `Rlt` (= `Rlt.ort` and `Rlt.orb`)
- `ort` (= `Rqz.ort` and `Rlt.ort`)
- `orb` (= `Rqz.orb` and `Rlt.orb`)

where `.orb` will be the future nexus comptibe output format.

q_z binning

The $R(\lambda, \alpha_f)$ arrays are projected onto a q_z grid, which is linear between $q_z = 0$ and $q_z = q_{\text{base}} = 0.1\,\text{Å}^{-1}$, and exponential for $q_z > q_{\text{base}}$. The bin boundaries are defined by:

$$q_{z\,i} \in [0,\, a,\, 2a,\, 3a,\, \ldots \hat{\imath}a] \qquad \forall \quad q_z \le q_{\text{base}}, \quad \hat{\imath} = q_{\text{base}}/a$$

$$q_{z\,\hat{\imath}+j} \in [q_{\text{base}} \cdot (1+a), q_{\text{base}} \cdot (1+a)^2, \ldots q_{\text{base}} \cdot (1+a)^j \ldots \qquad \forall \quad q_z > q_{\text{base}}$$

The **output resolution** a can be chosen with `-r` among the values
$a \in [0.005,\, 0.01,\, 0.02,\, 0.025,\, 0.04,\, 0.05,\, 0.1,\, 1]$
(this is restricted to ensure a *smooth* transition between the linear and exponential regions). The best instrument resolution is $\sigma_{q_z}/q_z = 2.2\,\%$.

intensity scaling

The argument `-s <value>` leads to a multiplication of all $R(q_z)$ curves with `<value>`. This is useful for one curve, only, or in combination with the `-S` argument.

$R(q_z)$ of the first reflectivity curve can be scaled to 1 in the q_z interval define by `-S <start> <stop>`. The following $R(q_z)$ curves are then scaled to match the respective previous one in the overlapping q_z range.

time-slizing

One (combined) data set can be chopped in slizes with the argument
`-ts <interval> [<start> [stop>]]` where `<interval>` is the time interval length in seconds. The chopping starts `<start>` seconds after the start of the measurement (default: 0) and ends at
`<stop>` seconds (default: end of the measurement).

All the resulting $R(q_z)$ curves are stored in one file, one after the other. An additional column is added with the start time of the respective slize.

example: `python -f 20-22 -n 123 -ts 60 1200 4000 -f foo`
The event streams of the measurements #20, #21 and #22 are merged. All events before $t = 1200\,\mathrm{s}$ with respect to the start of meausrement #20 are discarded. Then until $t = 4020\,\mathrm{s}$ (the starting time of the last slize is within the given interval) a $R(q_z)$ curve is generated for each 60 s interval.

---

masking

```
masks:
  -l LAMBDARANGE LAMBDARANGE, --lambdaRange LAMBDARANGE LAMBDARANGE
                        wavelength range
  -t THETARANGE THETARANGE, --thetaRange THETARANGE THETARANGE
                        absolute theta range
  -T THETARANGER THETARANGER, --thetaRangeR THETARANGER THETARANGER
                        relative theta range
  -y YRANGE YRANGE, --yRange YRANGE YRANGE
                        detector y range
  -q QZRANGE QZRANGE, --qzRange QZRANGE QZRANGE
                        q_z range
```

detector region The specularely reflected intensity illuminated the detector only in a limited region. To reduce noise, the **detector region of interest** is reduced by default to the inner horizontal channels of the detector and the vertical channels corresponding to the divergence of the beam. These values can be overwirtten by using the keys `-y` and `-t` for absolute finite angles or `-T` for the angular distance from the detector center.

λ The **wavelength band** can be limited using `-l`, where the arguments are the wavelengths with unit `angstrom`.

q_z The $\mathbf{q_z}$ **range** can de limited using `-q`, where the arguments are the momentum transfers with unit `1/angstrom`.

default values `-y 11 41 -l 2.0 15.0 -q 0.005 0.3`

For high-intensity specular neasurements, the angular region of interest is determined automatically. In most cases this corresponds to `-T -0.7 0.7`.

The wavelength range should be adapted to the measurement mode (divergent, focused, polarised).

---

overwrite parameters from the nexus file

```
overwrite:
  -cs CHOPPERSPEED, --chopperSpeed CHOPPERSPEED
                        chopper speed in rpm
  -cp CHOPPERPHASE, --chopperPhase CHOPPERPHASE
                        chopper phase
  -co CHOPPERPHASEOFFSET, --chopperPhaseOffset CHOPPERPHASEOFFSET
                        phase offset between chopper opening and trigger pulse
  -m MUOFFSET, --muOffset MUOFFSET
                        mu offset
  -mu MU, --mu MU       value of mu
  -nu NU, --nu NU       value of nu
  -sm SAMPLEMODEL, --sampleModel SAMPLEMODEL
                        1-line orso sample model description
```

purposes:

- debugging
- correctiion of wrong entries (due to communication problems)
- take into account misalignments

TODO list

- start and stop time of the measurement are not correct due to incomplete *.hdf* files.
- off-speculer measurements are not yet included
- background subtraction is missing
- several header parameters for *orso* compatibility are missing