Direct Inversion Documentation

Release 1.0.10

Nikunj Patel

August 31, 2010

CONTENTS

1 DiRefl - An Overview **3**

2 Installing DiRefl **5**

3 Building DiRefl Code **7**

4 User's Guide to DiRefl **9**
- 4.1 Introduction . . . 9
- 4.2 A Simple Example . . . 9
- 4.3 Tutorial . . . 9

5 API Reference Guide **11**
- 5.1 Introduction . . . 11
- 5.2 Inversion . . . 11
- 5.3 Simulation . . . 19
- 5.4 Resolution . . . 20

6 Acknowledgement **29**

7 License **31**

8 Indices and Tables **33**

Bibliography **35**

Python Module Index **37**

Index **39**

The Direct Inversion Reflectometry (DiRefl) application generates a scattering length density (SLD) profile of a thin film or free form sample using two neutron scattering datasets without the need to perform a fit of the data. DiRefl also has a simulation capability for creating datasets from a simple model description of the sample material.

This section gives an overview on how to get started with Direct Inversion. Read this to have an idea about what are system requirements for using Direct Inversion package.

CHAPTER

ONE

DIREFL - AN OVERVIEW

Direct Inversion Reflectometry (DiRefl) applies phase reconstruction and direct inversion techniques to analyze the reflectivity datasets produced by the two neutron scattering experiments performed on a single or multi-layer sample sandwiched between incident and substrate layers whose characteristics are known. The only setup difference between the runs is that the user changes the material of one of the surrounding layers. Output from DiRefl is in the form of a SLD profile graph and other supporting plots that can be saved or printed. In addition, the user can load, edit, and save model information, load reflectometry datasets, and adjust several parameters that affect the qualitative results of the analysis.

CHAPTER

TWO

INSTALLING DIREFL

This tutorial will walk you through the process of installing Direfl.

In order to install Direfl on your system, you will need to perform the following steps:

CHAPTER

THREE

BUILDING DIREFL CODE

Details for building DiRefl goes here.

CHAPTER

FOUR

USER'S GUIDE TO DIREFL

This section gives an overview and introduction to Direct Inversion. Read this to have an idea about what Direct Inversion can do for you (and how) and if you want to know in detail about Direct Inversion, refer to the *Direct Inversion Modules Reference*.

4.1 Introduction

Direct Inversion Reflectometry (DiRefl) applies phase reconstruction and direct inversion techniques to analyze the reflectivity datasets produced by the two neutron scattering experiments performed on a single or multi-layer sample sandwiched between incident and substrate layers whose characteristics are known. The only setup difference between the runs is that the user changes the material of one of the surrounding layers. Output from DiRefl is in the form of a SLD profile graph and other supporting plots that can be saved or printed. In addition, the user can load, edit, and save model information, load reflectometry datasets, and adjust several parameters that affect the qualitative results of the analysis.

4.2 A Simple Example

4.3 Tutorial

CHAPTER

FIVE

API REFERENCE GUIDE

5.1 Introduction

5.2 Inversion

5.2.1 api.invert

Core classes and functions:

- **Interpolator** Class that performs data interpolation.
- **Inversion** Class that implements the inversion calculator.
- **SurroundVariation** Class that performs the surround variation calculation.
- **refl()** Reflectometry as a function of Qz and wavelength.
- **reconstruct()** Phase reconstruction by surround variation magic.
- **valid_f()** Calculate vector function using only the finite elements of the array.

Command line phase reconstruction phase inversion:

```
invert -u 2.07 -v 6.33 0 --Qmin 0.014 --thickness 1000 qrd1.refl qrd2.refl
```

Command line phase + inversion only:

```
invert --thickness=150 --Qmax 0.35 wsh02_re.dat
```

Scripts can use `reconstruct()` and `invert()`. For example:

```
>>> import invert
>>> substrate = 2.07
>>> f1, f2 = 0, -0.53
>>> phase = reflectometry.reconstruct("file1", "file2", substrate, f1, f2)
>>> inversion = invert.invert(data=(phase.Q, phase.RealR), thickness=200)
>>> inversion.plot()
>>> inversion.save("profile.dat")
```

The resulting profile has attributes for the input (*Q*, *RealR*) and the output (*z*, *rho*, *drho*). There are methods for plotting (*plot*, *plot_residual*) and storing (*save*). The analysis can be rerun with different attributes (*run(key=val,...)*).

See `reconstruct()` and `Inversion` for details.

The phase reconstruction algorithm is described in [Majkrzak2003]. The phase inversion algorithm is described in [Berk2009] and references therein. It is based on the partial differential equation solver described in [Sacks1993].

References

class api.invert.**Interpolator**(*xi*, *yi*, *porder=1*)
Construct an interpolation function from pairs (xi,yi).

class api.invert.**Inversion**(*data=None*, ***kw*)
Class that implements the inversion calculator.

This object holds the data and results associated with the direct inversion of the real value of the phase from a reflected signal.

Inversion converts a real reflectivity amplitude as computed by `reconstruct()` into a step profile of scattering length density as a function of depth. This process will only work for real-valued scattering potentials - with non-negligible absorption the results will be incorrect. With X-rays, the absorption is too high for this technique to be used successfully. For details on the underlying theory, see [Berk2009].

The following attributes and methods are of most interest:

Inputs:

Input Parameters	Description
data	The name of an input file or a pair of vectors (Q,RealR) where RealR is the real portion of the complex reflectivity amplitude.input filename or Q,RealR data (required).
thickness (400)	Defines the total thickness of the film of interest. If the value chosen is too small, the inverted profile will not be able to match the input reflection signal. If the thickness is too large, the film of interest should be properly reconstructed, but will be extended into a reconstructed substrate below the film.film thickness.
substrate (0)	It is the scattering length density of the substrate. The inversion calculation determines the scattering length densities (SLDs) within the profile relative to the SLD of the substrate. Entering the correct value of substrate will shift the profile back to the correct values.
bse (0)	It is the bound state energy correction factor. Films with large negative potentials at their base sometimes produce an incorrect inversion, as indicated by an incorrect value for the substrate portion of a film. A value of substrate SLD - bound state SLD seems to correct the reconstruction.
Qmin (0)	Minimum Q to use from data. Reduce *Qmax* to avoid contamination from noise at high Q and improve precision. However, doing this will reduce the size of the features that you are sensitive to in your profile.
Qmax (None)	Maximum Q to use from data. Increase *Qmin* to avoid values at low Q which will not have the correct phase reconstruction when Q is less than Qc^2 for both surround variation measurements used in the phase reconstruction calculation. Use this technique sparingly — the overall shape of the profile is sensitive to data at low Q.
backrefl (True)	Reflection measured through the substrate. It is True if the film is measured with an incident beam through the substrate rather than the surface.

Uncertainty controls:

Uncertainty is handled by averaging over *stages* inversions with noise added to the input data for each inversion. Usually the measurement uncertainty is estimated during data reduction and phase reconstruction, and Gaussian noise is added to the data. This is scaled by a factor of *noise* so the effects of noisier or quieter input are easy to estimate. If the uncertainty estimate is not available, 5% relative noise per point is assumed.

If *monitor* is specified, then Poisson noise is used instead, according to the following:

```
*noise* U[-1,1] (poisson(*monitor* |real R|)/*monitor* - |real R|)
```

That is, a value is pulled from the Poisson distribution of the expected counts, and the noise is the difference between this and the actual counts. This is further scaled by a fudge factor of *noise* and a further random uniform in [-1,1].

Uncertainty controls	Description
stages (4)	number of inversions to average over
noise (1)	noise scale factor
monitor (None)	incident beam intensity (poisson noise source)

Inversion controls:

Inversions controls	Description
rhopoints (128)	number of steps in the returned profile. If this value is too low, the profile will be coarse. If it is too high, the computation will take a long time. The additional smoothness generated by a high value of *rhopoints* is illusory — the information content of the profile is limited by the number of Q points which have been measured. Set *rhopoints* to (1/*dz*) for a step size near *dz* in the profile.
calcpoints (4)	number of internal steps per profile step. It is used internally to improve the accuracy of the calculation. For larger values of *rhopoints*, smaller values of *calcpoints* are feasible.
iters (6)	number of iterations to use for inversion. A value of 6 seems to work well. You can observe this by setting *showiters* to True and looking at the convergence of each stage of the averaging calculation.
showiters (False)	set to true to show inversion converging. Click the graph to move to the next stage.
ctf_window (0)	cosine transform smoothing. In practice, it is set to 0 for no smoothing.

Computed profile: The reflectivity computed from *z*, *rho* will not match the input data because the effect of the substrate has been removed in the process of reconstructing the phase. Instead, you will need to compute reflectivity from *rho-substrate* on the reversed profile. This is done in `refl()` when no surround material is selected, and can be used to show the difference between measured and inverted reflectivity. You may need to increase *calcpoints* or modify *thickness* to get a close match.

Computed profile	Description
Qinput, *RealRinput*	input data. The input data *Qinput*, *RealRinput* need to be placed on an even grid going from 0 to *Qmax* using linear interpolation. Values below *Qmin* are set to zero, and the number of points between *Qmin* and *Qmax* is preserved. This resampling works best when the input data are equally spaced, starting at k*dQ for some k.
Q, *RealR*, *dRealR*	output data. The returned *Q*, *RealR*, *dRealR* are the values averaged over multiple stages with added noise. The plots show this as the range of input variation used in approximating the profile variation.
z	represents the depth into the profile. *z* equals *thickness* at the substrate. If the thickness is correct, then *z* will be zero at the top of the film, but in practice the *thickness* value provided will be larger than the actual film thickness, and a portion of the vacuum will be included at the beginning of the profile.
rho	It is the SLD at depth *z* in units of 10^-6 inv A^2. It is calculated from the average of the inverted profiles from the noisy data sets, and includes the correction for the substrate SLD defined by *substrate*. The inverted *rho* will contain artifacts from the abrupt cutoff in the signal at *Qmin* and *Qmax*.
drho	It is the uncertainty in the SLD profile at depth *z*. It is calculated from the standard deviation of the inverted profiles from the noisy data sets. The uncertainty *drho* does not take into account the possible variation in the signal above *Qmax*.
signals	It is a list of the noisy (Q,RealR) input signals generated by the uncertainty controls.
profiles	It is a list of the corresponding (z,rho) profiles. The first stage is computed without noise, so *signals[0]* contains the meshed input and *profiles[0]* contains the output of the inversion process without additional noise.

Output methods: The primary output methods are

Output methods	Description
save	save the profile to a file.
show	show the profile on the screen.
plot	plot data and profile.
refl	compute reflectivity from profile.
run	run or rerun the inversion with new settings.

Additional methods for finer control of plots:

Output methods	Description
plot_data	plot just the data.
plot_profile	plot just the profile.
plot_residual	plot data minus theory.

chisq()
Compute normalized sum squared difference between original real R and the real R for the inverted profile.

plot(*details=False*, *phase=None*)
Plot the data and the inversion.

Parameters:

details: **boolean** If *details* is True, then plot the individual stages used to calculate the average, otherwise just plot the envelope.

phase: **boolean** If *phase* is a phase reconstruction object, plot the original measurements.

Returns: *None*

plot_input(*details=False*, *lowQ_inset=0*)
Plot the real R vs. the real R computed from inversion.

Parameters

details: **boolean** If *details* is True, then plot the individual stages used to calculate the average, otherwise just plot the envelope.

lowQ_inset: **intger** If *lowQ_inset* > 0, then plot a graph of Q, real R values below lowQ_inset, without scaling by Q**2.

Returns: *None*

plot_profile(*details=False*, ***kw*)
Plot the computed profiles.

Parameters:

details: **boolean** If *details* is True, then plot the individual stages used to calculate the average, otherwise just plot the envelope.

Returns: *None*

plot_residual(*details=False*)
Plot the residuals (inversion minus input).

Parameters:

details: **boolean** If *details* is True, then plot the individual stages used to calculate the average, otherwise just plot the envelope.

Returns: *None*

refl(*Q=None*, *surround=None*)
Return the complex reflectivity amplitude.

Parameters:

Q: **boolean** Use *Q* if provided, otherwise use the evenly spaced Q values used for the inversion.

surround: **boolean** If *surround* is provided, compute the reflectivity for the free film in the context of the substrate and the surround, otherwise compute the reflectivity of the reversed free film embedded in the substrate to match against the reflectivity amplitude supplied as input.

Returns: *None*

run(***kw*)
Run multiple inversions with resynthesized data for each.

All control keywords from the constructor can be used, except *data* and *outfile*. Sets *signals* to the list of noisy (Q, RealR) signals and sets *profiles* to the list of generated (z,rho) profiles.

save(*outfile=None*)
Save z,rho,drho to three column text file named *outfile*.

Parameters:

outfile: **file** If *outfile* is not provided, the name of the input file will be used, but with the extension replaced by '.amp'.

Returns: *None*

show()
Print z, rho, drho to the screen.

Q
Inverted profile calculation points

RealR
Average inversion free film reflectivity input

dRealR
Free film reflectivity input uncertainty

drho
Inverted SLD profile uncertainty

rho
Inverted SLD profile in 10^-6 * inv A^2 units

z
Inverted SLD profile depth in Angstroms

class api.invert.**SurroundVariation**(*file1*, *file2*, *u*, *v1*, *v2*, *stages=100*)
See reconstruct() for details.

Attributes:

Attributes	Description
Q, *RealR*, *ImagR*	real and imaginary reflectivity
dRealR, *dImagR*	Monte Carlo uncertainty estimate or None
Qin, *R1*, *R2*	input data
dR1, *dR2*	input uncertainty or None
name1, *name2*	input file names
save(file)	save output
show(), *plot()*	show Q, RealR, ImagR

clean()
Remove points which are NaN or Inf from the computed phase.

optimize(*z*, *rho_initial*)
Run a quasi-Newton optimizer on a discretized profile.

Parameters:

z: **boolean** Represents the depth into the profile. z equals thickness at the substrate.

rho_initial: **boolean** The initial profile *rho_initial* should come from direct inversion.

Returns:

rho: **(boolean, boolean)|** Returns the final profile rho which minimizes chisq.

plot_imaginary()

plot_measurement(*profile=None*)
Plot the data, and if available, the inverted theory.

plot_phase()

refl(*z*, *rho*, *resid=False*)
Return the reflectivities R1 and R2 for the film *z*,*rho* in the context of the substrate and surround variation.

Parameters:

z: **boolean** Represents the depth into the profile. z equals thickness at the substrate.

rho: **boolean** If the resolution is known, then return the convolved theory function.

resid: **boolean** If *resid* is True, then return the weighted residuals vector.

Returns:

R1, R2: **(boolean, boolean)|** Return the reflectivities R1 and R2 for the film z,*rho*.

save(*outfile=None*, *uncertainty=True*)
Save Q,RealR,ImagR to three column text file named *outfile*.

Parameters:

outfile: **file** Include dRealR,dImagR if they exist and if *uncertainty* is True, making a five column file.

uncertainity: **boolean** Include dRealR and dImagR if True.

Returns: *None*

show()
Print Q, RealR, ImagR to the screen.

api.invert.**invert**(***kw*)
Invert data returning an Inversion object.

If outfile is specified, save z,rho,drho to the named file. If plot=True, show a plot before returning

api.invert.**main**()
Drive phase reconstruction and direct inversion from the command line.

api.invert.**phase_shift**(*q*, *r*, *shift=0*)

api.invert.**plotamp**(*Q*, *r*, *dr=None*, *scaled=True*, *ylabel='Real R'*, ***kw*)
Plot Q, realR data.

api.invert.**plottitle**(*title*)

api.invert.**reconstruct**(*file1*, *file2*, *u*, *v1*, *v2*, *stages=100*)
Two reflectivity measurements of a film with different surrounding media $|r_1|^2$ and $|r_2|^2$ can be combined to compute the expected complex reflection amplitude r_reversed of the free standing film measured from the opposite side. The calculation can be done by varying the fronting media or by varying the backing media. For this code we only support measurements through a uniform substrate *u*, on two varying surrounding materials *v1*, *v2*.

We have to be careful about terminology. We will use the term substrate to mean the base on which we deposit our film of interest, and surface to be the material we put on the other side. The fronting or incident medium is the material through which the beam enters the sample. The backing material is the material on the other side. In back reflectivity, the fronting material is the substrate and the backing material is the surface. We are using u for the uniform substrate and v for the varying surface material.

In the experimental setup at the NCNR, we have a liquid resevoir which we can place above the film. We measure first with one liquid in the resevoir such as heavy water (D2O) and again with air or a contrasting liquid such as water (H2O). At approximately 100 um, the resevoir depth is much thicker than the effective coherence length of the neutron in the z direction, and so can be treated as a semi-infinite substrate, even when it is empty.

Note: You cannot simulate a semi-infinite substrate using a large but finitely thick material using the reflectometry calculation; at best the resulting reflection will be a high frequency signal which smooths after applying the resolution correction to a magnitude that is twice the reflection from a semi-infinite substrate.

The incident beam is measured through the substrate, and thus subject to the same absorption as the reflected beam. Refraction on entering and leaving the substrated is accounted for by a small adjustment to Q inside the reflectivity calculation.

When measuring reflectivity through the substrate, the beam enters the substrate from the side, refracts a little because of the steep angle of entry, reflects off the sample, and leaves through the other side of the substrate with an equal but opposite refraction. The reflectivity calculation takes this into account. Traveling through several centimeters of substrate, some of the beam will get absorbed. We account for this either by entering an

incident medium transmission coefficient in the reduction process, or by measuring the incident beam through the substrate so that it is subject to approximately the same absorption.

The phase cannot be properly computed for Q values which are below the critical edge Qc^2 for both surround variations. This problem can be avoided by choosing a substrate which is smaller than the surround on at least one of the measurements. This measurement will not have a critical edge at positive Q. In order to do a correct footprint correction the other measurement should use a substrate SLD greater than the surround SLD.

If the input file records uncertainty in the measurement, we perform a Monte Carlo uncertainty estimate of the reconstructed complex amplitude.

Inputs:

Input parameters	Description
file1, *file2*	reflectivity measurements at identical Q values. *file1* and *file2* can be pairs of vectors (q1,r1), (q2,r2) or files containing at least two columns (q,r), with the remaining columns such as dr, dq, and lambda ignored. If a third vector, dr, is present in both datasets, then an uncertainty estimate will be calculated for the reconstructed phase.
v1, *v2*	SLD of varying surrounds in *file1* and *file2*
u	SLD of the uniform substrate
stages	number of trials in Monte Carlo uncertainty estimate

Returns a `SurroundVariation` object with the following attributes:

Attributes	Description
RealR, *ImagR*	real and imaginary reflectivity
dRealR, *dImagR*	Monte Carlo uncertainty estimate
name1, *name2*	names of the input files
save(file)	save Q, RealR, ImagR to a file
show(), *plot()*	display the results

`api.invert.`**`refl`**(*Qz*, *depth*, *rho*, *mu=0*, *wavelength=1*, *sigma=0*)
Reflectometry as a function of Qz and wavelength.

Parameters:

Qz: **float|A** Scattering vector 4*pi*sin(theta)/wavelength. This is an array.

depth: **float|A** Thickness of each layer. The thickness of the incident medium and substrate are ignored.

rho, mu (uNb): **(float, float)|** Scattering length density and absorption of each layer.

wavelength: **float|A** Incident wavelength (angstrom).

sigma: **float|A** Interfacial roughness. This is the roughness between a layer and the subsequent layer. There is no interface associated with the substrate. The sigma array should have at least n-1 entries, though it may have n with the last entry ignored.

Returns *r* array of float

`api.invert.`**`remesh`**(*data*, *xmin*, *xmax*, *npts*, *left=None*, *right=None*)
Resample the data on a fixed grid.

`api.invert.`**`valid_f`**(*f*, *A*, *axis=0*)
Calculate vector function f using only the finite elements of the array *A*. *axis* is the axis over which the calculation should be performed, or None if the calculation should summarize the entire array.

5.3 Simulation

5.3.1 api.simulate

This code is intended to be used during your experimental proposal phase to check that you will be able to see the features of interest in your sample if you measure it by direct inversion.

Simulate classes and functions:

- **Simulation** Simulate phase-reconstruction and inversion.
- **wait()** Wait for the user to acknowledge the plot.

class api.simulate.**Simulation**(*sample=None*, *q=None*, *dq=None*, *urough=0*, *u=2.0699999999999998*, *v1=0*, *v2=6.3300000000000001*, *noise=0*, *seed=None*, *phase_args={}*, *invert_args={}*, *perfect_reconstruction=False*)

Parameters	Description
sample	structure [(rho1,d1), ..., (rhon, dn)]
q, *dq*	measurement points
u, *v1*, *v2*	uniform and varying SLDs for surround variation
noise	percentage noise in the measurement
seed	random number generator seed
perfect_reconstruction	use real(r) from free film rather than simulated reflectivity
phase_args	keyword arguments for reconstruction calculation
invert_args	keyword arguments for inversion calculation

The default values for the surround are set to u=Si (2.07), v1=Air (0), and v2=D2O (6.33). Noise and roughness are set to 0.

check()
Check phase and inversion.

check_inversion()
Check that the reconstructed profile matches the sample.

check_phase()
Check that the reconstructed phase is correct within noise.

phase_residual()
Return normalized residual from phase reconstruction.

plot()
Plot summary data.

plot_imaginary(*subplot=111*)
Plot the simulated phase (imaginary part).

plot_inversion(*subplot=111*)
Plot the phase of the inverted profile.

plot_measurement(*subplot=111*)
Plot the simulated data.

plot_phase_residual(*subplot=111*)
Plot the reconstructed phase residual.

plot_profile(*subplot=111*)
Plot the inverted profile.

plot_real(*subplot=111*)
Plot the simulated phase and the reconstructed phase (real).

run()
Reconstruct phase, invert and optimize profile.

sample_profile()

set(***kw*)
Reset or adjust input parameters, generating new sample data.

slab_profile()
Generate the sample profile.

api.simulate.**wait**(*msg=None*)
Wait for the user to acknowledge the plot.

5.4 Resolution

5.4.1 api.resolution

Given Q = 4 pi sin(theta)/lambda, the resolution in Q is determined by the dispersion angle theta and wavelength lambda. For monochromatic sources, the wavelength resolution is fixed and the angular resolution varies. For polychromatic sources, the wavelength resolution varies and the angular resolution is fixed.

The angular resolution is determined by the geometry (slit positions and openings and sample profile) with perhaps an additional contribution from sample warp. For monochromatic instruments, measurements are taken with fixed slits at low angles until the beam falls completely onto the sample. Then as the angle increases, slits are opened to preserve full illumination. At some point the slit openings exceed the beam width, and thus they are left fixed for all angles above this threshold.

When the sample is tiny, stray neutrons miss the sample and are not reflected onto the detector. This results in a resolution that is tighter than expected given the slit openings. If the sample width is available, we can use that to determine how much of the beam is intercepted by the sample, which we then use as an alternative second slit. This simple calculation isn't quite correct for very low Q, but in that case both reflected and transmitted neutrons will arrive at the detector, so the computed resolution of dQ=0 at Q=0 is good enough.

When the sample is warped, it may act to either focus or spread the incident beam. Some samples are diffuse scatters, which also acts to spread the beam. The degree of spread can be estimated from the full-width at half max (FWHM) of a rocking curve at known slit settings. The expected FWHM will be (s1+s2) / (2*(d_s1-d_s2)). The difference between this and the measured FWHM is the sample_broadening value. A second order effect is that at low angles the warping will cast shadows, changing the resolution and intensity in very complex ways.

For polychromatic time of flight instruments, the wavelength dispersion is determined by the reduction process, which bins different time channels between fastest and slowest, usually in a way that sets a fixed relative resolution dL/L for each bin.

Usage

resolution module defines two instrument types: `Monochromatic` and `Polychromatic`. These represent generic scanning and time of flight instruments, respectively. In addition, instruments at SNS and NCNR have many of their parameters predefined in `snsdata` and `ncnrdata`.

To perform a simulation or load a data set, an actual instrument must be created. For example:

```
>>> from resolution import Monochromatic
>>> instrument = Monochromatic(
        # instrument parameters
        instrument = "AND/R",
        radiation = "neutron",
        wavelength = 5.0042,
        dLoL=0.009,
        d_s1 = 230.0 + 1856.0,
        d_s2 = 230.0,
        # measurement parameters
        Tlo=0.5, slits_at_Tlo=0.2, slits_below=0.1,
        )
```

An instrument such as this can be used to compute a resolution function for arbitrary Q values, following the slit opening pattern defined by the instrument.

```
>>> resolution = instrument.resolution(T=linspace(0,5,51)
```

The resolution object *resolution* is of `Resolution`. Using it you can compute the resolution *dQ* for a given *Q* based on *T*, *dT*, *L*, and *dL*. Furthermore, it can easily be turned into a reflectometry probe for use in modeling:

```
>>> probe = resolution.probe()
```

Since creating the reflectometry probe is the usual case, a couple of driver functions are provided. For example, the following complete example loads and plots two data files:

```
>>> import pylab
>>> from ncnrdata import ANDR
>>> instrument = ANDR(Tlo=0.5, slits_at_Tlo=0.2, slits_below=0.1)
>>> probe1 = instrument.load('blg117.refl')
>>> probe2 = instrument.load('blg126.refl')
>>> probe1.plot()
>>> pylab.hold(True)
>>> probe2.plot()
>>> pylab.show()
```

Generating a simulation probe is similarly convenient:

```
>>> from ncnrdata import ANDR
>>> instrument = ANDR(Tlo=0.5, slits_at_Tlo=0.2, slits_below=0.1)
>>> probe = instrument.simulate(T=numpy.arange(0,5,100))
```

and for magnetic systems in polarized beam:

```
>>> probe = instrument.simulate_magnetic(T=numpy.arange(0,5,100))
```

When loading or simulating a data set, any of the instrument parameters and measurement geometry information can be specified, replacing the defaults within the instrument. For example, to include sample broadening effects in the resolution:

```
>>> from ncnrdata import ANDR
>>> instrument = ANDR(Tlo=0.5, slits_at_Tlo=0.2, slits_below=0.1)
>>> probe1 = instrument.load('blg117.refl', sample_broadening=0.1)
```

Properties of the instrument can be displayed:

```
>>> print ANDR.defaults()
>>> print instrument.defaults()
```

Details

Polychromatic measurements have the following attributes:

Attributes	Description
instrument	name of the instrument
radiation (xray or neutron)	source radiation type
T (degrees)	sample angle
slits (mm or mm,mm)	slit 1 and slit 2 openings
d_s1, *d_s2* (mm)	distance from sample to pre-sample slits 1 and 2; post-sample slits are ignored.
wavelength (Angstrom,Angstrom)	wavelength range for the measurement.
dLoL	constant relative wavelength dispersion; wavelength range and dispersion together determine the bins.
sample_width (mm)	width of sample; at low angle with tiny samples, stray neutrons miss the sample and are not reflected onto the detector, so the sample itself acts as a slit, therefore the width of the sample may be needed to compute the resolution correctly.
sample_broadening (degrees FWHM)	amount of angular divergence (+) or focusing (-) introduced by the sample; this is caused by sample warp, and may be read off of the rocking curve by subtracting (s1+s2)/2/(d_s1-d_s2) from the FWHM width of the rocking curve.

Monochromatic measurements have these additional or modified attributes:

Attributes	Description
instrument	name of the instrument
radiation (xray or neutron)	source radiation type
d_s1, *d_s2* (mm)	distance from sample to pre-sample slits 1 and 2; post-sample slits are ignored.
wavelength (Angstrom)	wavelength of the instrument
dLoL	constant relative wavelength dispersion; wavelength range and dispersion together determine the bins.
slits_at_Tlo (mm)	slit 1 and slit 2 openings at Tlo; this can be a scalar if both slits are open by the same amount, otherwise it is a pair (s1,s2).
slits_below, *slits_above*	slit 1 and slit 2 openings below Tlo and above Thi; again, these can be scalar if slit 1 and slit 2 are the same, otherwise they are each a pair (s1,s2). Below and above default to the values of the slits at Tlo and Thi respectively.
Tlo, *Thi* (degrees)	range of opening slits, or inf if slits are fixed.
sample_width (mm)	width of sample; at low angle with tiny samples, stray neutrons miss the sample and are not reflected onto the detector, so the sample itself acts as a slit, therefore the width of the sample may be needed to compute the resolution correctly
sample_broadening (degrees FWHM)	amount of angular divergence (+) or focusing (-) introduced by the sample; this is caused by sample warp, and may be read off of the rocking curve by subtracting (s1+s2)/2/(d_s1-d_s2) from the FWHM width of the rocking curve.

These parameters should be available in the reduced data file, or they can be found in the raw NeXus file.

GUI Usage

Graphical user interfaces follow different usage patterns from scripts. Here the emphasis will be on selecting a data set to process, displaying its default metadata and allowing the user to override it.

File loading should follow the pattern established in reflectometry reduction, with an extension registry and a fallback scheme whereby files can be checked in a predefined order. If the file cannot be loaded, then the next loader is

tried. This should be extended with the concept of a magic signature such as those used by graphics and sound file applications: preread the first block and run it through the signature check before trying to load it. For unrecognized extensions, all loaders can be tried.

The file loader should return an instrument instance with metadata initialized from the file header. This metadata can be displayed to the user along with a plot of the data and the resolution. When metadata values are changed, the resolution can be recomputed and the display updated. When the data is accepted, the final resolution calculation can be performed.

Calculations

Resolution in Q is computed from uncertainty in wavelength L and angle T using propogation of errors:

```
dQ**2 = (df/dL)**2 dL**2 + (df/dT)**2 dT**2
```

where:

```
f(L,T) = 4 pi sin(T)/L
df/dL = -4 pi sin(T)/L**2 = -Q/L
df/dT = 4 pi cos(T)/L = (4 pi sin(T) / L) / (sin(T)/cos(T)) = Q/tan(T)
```

yielding the traditional form:

```
(dQ/Q)**2 = (dL/L)**2 + (dT/tan(T))**2
```

Computationally, 1/tan(T) is infinity at T=0, so it is better to use the direct calculation:

```
dQ = (4 pi / L) sqrt( sin(T)**2 (dL/L)**2 + cos(T)**2 dT**2 )
```

Wavelength dispersion dL/L is usually constant (e.g., for AND/R it is 2% FWHM), but it can vary on time-of-flight instruments depending on how the data is binned.

Angular divergence dT comes primarily from the slit geometry, but can have broadening or focusing due to a warped sample. The slit contribution is dT = (s1+s2)/(2d) FWHM where s1,s2 are slit openings and d is the distance between slits. For tiny samples, the sample itself can act as a slit. If s_sample = sin(T)*w is smaller than s2 for some T, then use dT = (s1+s_sample)/(2(d+d_sample)) instead.

The sample broadening can be read off a rocking curve as w - (s1+s2)/(2d) where w is the measured FWHM of the peak. This constant should be added to the computed dT for all angles and slit geometries. You will not usually have this information on hand, but you can leave space for users to enter it if it is available.

FWHM can be converted to 1-sigma resolution using the scale factor of 1/sqrt(8 * log(2))

For opening slits, dT/T is held constant, so if you know s and To at the start of the opening slits region you can compute dT/To, and later scale that to your particular T:

```
dT(Q) = dT/To * T(Q)
```

Because d is fixed, that means s1(T) = s1(To) * T/To and s2(T) = s2(To) * T/To

class api.resolution.**Monochromatic**(**kw*)

Resolution calculator for scanning reflectometers.

calc_dT(*T*, *slits*, **kw*)

Compute the FWHM angular divergence in radians

d_s1, *d_s2* slit distances *sample_width* size of sample *sample_broadening* resolution changes from sample warping

Parameters:

T: **angle | A** Angle for measurement.

slits: **float** slits openings.

Returns: object of FWHM divergence

calc_slits(*T*, ***kw*)
Determines slit openings from measurement pattern.

If slits are fixed simply return the same slits for every angle, otherwise use an opening range [Tlo,Thi] and the value of the slits at the start of the opening to define the slits. Slits below Tlo and above Thi can be specified separately.

Parameters	Description
Tlo, *Thi*	angle range over which slits are opening
slits_at_Tlo	openings at the start of the range, or fixed opening
slits_below, *slits_above*	openings below and above the range

Use fixed_slits if available, otherwise use opening slits.

classmethod defaults()
Return default instrument properties as a printable string.

load(*filename*, ***kw*)
Load the data, returning the associated probe. This probe will contain Q, angle, wavelength, measured reflectivity and the associated uncertainties.

You can override instrument parameters using key=value. In particular, slit settings *slits_at_Tlo*, *Tlo*, *Thi*, and *slits_below*, and *slits_above* are used to define the angular divergence.

Parameters:

filename: **boolean** Name of file which holds data.

Returns:

Associated probe This probe will contain Q, angle, wavelength, measured reflectivity and the associated uncertainties.

resolution(*Q*, ***kw*)
Return the resolution for a given Q.

Resolution is an object with fields T, dT, L, dL, Q, dQ

Parameters:

Q: **float** Value for the measurement.

Returns:

object Return the resolution for a given Q. Resolution is an object with fields T, dT, L, dL, Q, dQ

simulate(*T=None*, *Q=None*, ***kw*)
Simulate the probe for a measurement.

Select the *Q* values or the angles *T* for the measurement, but not both.

Returns a probe with Q, angle, wavelength and the associated uncertainties, but not any data.

You can override instrument parameters using key=value. In particular, settings for *slits_at_Tlo*, *Tlo*, *Thi*, *slits_below*, and *slits_above* are used to define the angular divergence.

Parameters:

T: **angle | A** Angle for measurement.

Q: **float** Value for the measurement.

Returns:

Associated probe Returns a probe with Q, angle, wavelength and the associated uncertainties, but not any data.

simulate_magnetic(*T=None*, *Tguide=270*, *shared_beam=True*, ***kw*)
Simulate a polarized measurement probe.

Returns a probe with Q, angle, wavelength and the associated uncertainties, but not any data.

Guide field angle *Tguide* can be specified, as well as keyword arguments for the geometry of the probe cross sections such as *slits_at_Tlo*, *Tlo*, *Thi*, *slits_below*, and *slits_above* to define the angular divergence.

Parameters:

T: **angle | A** Angle for measurement.

Tguide: **float** Guide field angle for the measurement.

shared_beam: boolean

Returns:

Associated probe Returns a probe with Q, angle, wavelength and the associated uncertainties, but not any data.

class api.resolution.**Polychromatic**(***kw*)
Resolution calculator for multi-wavelength reflectometers.

calc_dT(*T*, *slits*, ***kw*)

classmethod defaults()
Return default instrument properties as a printable string.

load(*filename*, ***kw*)
Load the data, returning the associated probe. This probe will contain Q, angle, wavelength, measured reflectivity and the associated uncertainties.

You can override instrument parameters using key=value. In particular, slit settings *slits* and *T* define the angular divergence.

Parameters:

filename: **boolean** Name of file which holds data.

Returns:

Associated probe This probe will contain Q, angle, wavelength, measured reflectivity and the associated uncertainties.

resolution(*L*, *dL*, ***kw*)
Return the resolution of the measurement. Needs *Q*, *L*, *dL* specified as keywords.

Resolution is an object with fields T, dT, L, dL, Q, dQ

Parameters:

L: **float** Value for the measurement.

dL: float

Returns:

object Return the resolution for a given Q. Resolution is an object with fields T, dT, L, dL, Q, dQ

simulate(***kw*)
Simulate a measurement probe.

Returns a probe with Q, angle, wavelength and the associated uncertainties, but not any data.

You can override instrument parameters using key=value. In particular, slit settings *slits* and *T* define the angular divergence and *dLoL* defines the wavelength resolution.

Parameters:

Returns:

Associated probe Returns a probe with Q, angle, wavelength and the associated uncertainties, but not any data.

simulate_magnetic(*Tguide=270*, *shared_beam=True*, ***kw*)
Simulate a polarized measurement probe.

Returns a probe with Q, angle, wavelength and the associated uncertainties, but not any data.

Guide field angle *Tguide* can be specified, as well as keyword arguments for the geometry of the probe cross sections such as slit settings *slits* and *T* to define the angular divergence and *dLoL* to define the wavelength resolution.

Parameters:

Tguide: **float** Guide field angle for the measurement.

shared_beam: boolean

Returns:

Associated probe Returns a probe with Q, angle, wavelength and the associated uncertainties, but not any data.

class api.resolution.**Resolution**(*T*, *dT*, *L*, *dL*, *radiation='neutron'*)
Reflectometry resolution object.

T, *dT* (degrees) angle and FWHM divergence *L*, *dL* (Angstroms) wavelength and FWHM dispersion *Q*, *dQ* (inv Angstroms) calculated Q and 1-sigma resolution

probe(***kw*)
Return a reflectometry measurement object of the given resolution.

Q

dQ

api.resolution.**binedges**(*L*)
Construct bin edges E assuming that L represents the bin centers of a measured TOF data set.

The bins L are assumed to be spaced logarithmically with edges:

```
E[0] = min wavelength
E[i+1] = E[i] + dLoL*E[i]
```

and centers:

```
L[i] = (E[i]+E[i+1])/2
     = (E[i] + E[i]*(1+dLoL))/2
     = E[i]*(2 + dLoL)/2
```

so:

```
E[i] = L[i]*2/(2+dLoL)
E[n+1] = L[n]*2/(2+dLoL)*(1+dLoL)
```

api.resolution.**bins**(*low*, *high*, *dLoL*)
Return bin centers from low to high perserving a fixed resolution.

low,*high* are the minimum and maximum wavelength. *dLoL* is the desired resolution FWHM dL/L for the bins.

api.resolution.**binwidths**(*L*)

Construct dL assuming that L represents the bin centers of a measured TOF data set, and dL is the bin width.

The bins L are assumed to be spaced logarithmically with edges:

```
E[0] = min wavelength
E[i+1] = E[i] + dLoL*E[i]
```

and centers:

```
L[i] = (E[i]+E[i+1])/2
     = (E[i] + E[i]*(1+dLoL))/2
     = E[i]*(2 + dLoL)/2
```

so:

```
L[i+1]/L[i] = E[i+1]/E[i] = (1+dLoL)
dL[i] = E[i+1]-E[i] = (1+dLoL)*E[i]-E[i]
      = dLoL*E[i] = 2*dLoL/(2+dLoL)*L[i]
```

api.resolution.**divergence**(*T=None*, *slits=None*, *distance=None*, *sample_width=10000000000.0*, *sample_broadening=0*)

Calculate divergence due to slit and sample geometry.

Returns FWHM divergence in degrees.

T (degrees) incident angles *slits* (mm) s1,s2 slit openings for slit 1 and slit 2 *distance* (mm) d1,d2 distance from sample to slit 1 and slit 2 *sample_width* (mm) w, width of the sample *sample_broadening* (degrees FWHM) additional divergence caused by sample

Uses the following formula:

```
p = w * sin(radians(T))
dT = / (s1+s2) / 2 (d1-d2)   if p >= s2
     \ (s1+p) / 2 d1         otherwise
dT = dT + sample_broadening
```

where p is the projection of the sample into the beam.

sample_broadening can be estimated from W, the FWHM of a rocking curve:

```
sample_broadening = W - (s1+s2) / 2(d1-d2)
```

Note: default sample width is large but not infinite so that at T=0, sin(0)*sample_width returns 0 rather than NaN.

api.resolution.**opening_slits**(*T=None*, *slits_at_Tlo=None*, *Tlo=None*, *Thi=None*, *slits_below=None*, *slits_above=None*)

Compute the slit openings for the standard scanning reflectometer fixed-opening-fixed geometry.

Slits are assumed to be fixed below angle *Tlo* and above angle *Thi*, and opening at a constant dT/T between them.

Slit openings are recorded at *Tlo* as a tuple (s1,s2) or constant s=s1=s2. *Tlo* is optional for completely fixed slits. *Thi* is optional if there is no top limit to the fixed slits.

slits_below are the slits at $T <$ *Tlo*. *slits_above* are the slits at $T >$ *Thi*.

CHAPTER

SIX

ACKNOWLEDGEMENT

CHAPTER

SEVEN

LICENSE

CHAPTER

EIGHT

INDICES AND TABLES

- *genindex*
- *modindex*

BIBLIOGRAPHY

[Majkrzak2003] C. F. Majkrzak, N. F. Berk and U. A. Perez-Salas, "Phase-Sensitive Neutron Reflectometry", *Langmuir* 19, 7796-7810 (2003).

[Berk2009] N. F. Berk and C. F. Majkrzak, "Statistical analysis of phase-inversion neutron specular reflectivity", *Langmuir* 25, 4132-4144 (2009).

[Sacks1993] P.E. Sacks, *Wave Motion* 18, 21-30 (1993).

PYTHON MODULE INDEX

a

api.invert, 11
api.resolution, 20
api.simulate, 19

INDEX

A

api.invert (module), 11
api.resolution (module), 20
api.simulate (module), 19

B

binedges() (in module api.resolution), 26
bins() (in module api.resolution), 26
binwidths() (in module api.resolution), 27

C

calc_dT() (api.resolution.Monochromatic method), 23
calc_dT() (api.resolution.Polychromatic method), 25
calc_slits() (api.resolution.Monochromatic method), 24
check() (api.simulate.Simulation method), 19
check_inversion() (api.simulate.Simulation method), 19
check_phase() (api.simulate.Simulation method), 19
chisq() (api.invert.Inversion method), 14
clean() (api.invert.SurroundVariation method), 16

D

defaults() (api.resolution.Monochromatic class method), 24
defaults() (api.resolution.Polychromatic class method), 25
divergence() (in module api.resolution), 27
dQ (api.resolution.Resolution attribute), 26
dRealR (api.invert.Inversion attribute), 16
drho (api.invert.Inversion attribute), 16

I

Interpolator (class in api.invert), 12
Inversion (class in api.invert), 12
invert() (in module api.invert), 17

L

load() (api.resolution.Monochromatic method), 24
load() (api.resolution.Polychromatic method), 25

M

main() (in module api.invert), 17
Monochromatic (class in api.resolution), 23

O

opening_slits() (in module api.resolution), 27
optimize() (api.invert.SurroundVariation method), 16

P

phase_residual() (api.simulate.Simulation method), 19
phase_shift() (in module api.invert), 17
plot() (api.invert.Inversion method), 14
plot() (api.simulate.Simulation method), 19
plot_imaginary() (api.invert.SurroundVariation method), 16
plot_imaginary() (api.simulate.Simulation method), 19
plot_input() (api.invert.Inversion method), 14
plot_inversion() (api.simulate.Simulation method), 19
plot_measurement() (api.invert.SurroundVariation method), 16
plot_measurement() (api.simulate.Simulation method), 19
plot_phase() (api.invert.SurroundVariation method), 16
plot_phase_residual() (api.simulate.Simulation method), 19
plot_profile() (api.invert.Inversion method), 15
plot_profile() (api.simulate.Simulation method), 19
plot_real() (api.simulate.Simulation method), 19
plot_residual() (api.invert.Inversion method), 15
plotamp() (in module api.invert), 17
plottitle() (in module api.invert), 17
Polychromatic (class in api.resolution), 25
probe() (api.resolution.Resolution method), 26

Q

Q (api.invert.Inversion attribute), 15
Q (api.resolution.Resolution attribute), 26

R

RealR (api.invert.Inversion attribute), 15
reconstruct() (in module api.invert), 17
refl() (api.invert.Inversion method), 15
refl() (api.invert.SurroundVariation method), 16

refl() (in module api.invert), 18
remesh() (in module api.invert), 18
Resolution (class in api.resolution), 26
resolution() (api.resolution.Monochromatic method), 24
resolution() (api.resolution.Polychromatic method), 25
rho (api.invert.Inversion attribute), 16
run() (api.invert.Inversion method), 15
run() (api.simulate.Simulation method), 20

S

sample_profile() (api.simulate.Simulation method), 20
save() (api.invert.Inversion method), 15
save() (api.invert.SurroundVariation method), 17
set() (api.simulate.Simulation method), 20
show() (api.invert.Inversion method), 15
show() (api.invert.SurroundVariation method), 17
simulate() (api.resolution.Monochromatic method), 24
simulate() (api.resolution.Polychromatic method), 25
simulate_magnetic() (api.resolution.Monochromatic method), 25
simulate_magnetic() (api.resolution.Polychromatic method), 26
Simulation (class in api.simulate), 19
slab_profile() (api.simulate.Simulation method), 20
SurroundVariation (class in api.invert), 16

V

valid_f() (in module api.invert), 18

W

wait() (in module api.simulate), 20

Z

z (api.invert.Inversion attribute), 16