Example *molgroups* script with *Refl1D* interface

D. Hoogerheide, 9/30/2024, 1/28/2025

Introduction to *molgroups*

The *molgroups* package is used for a variety of applications, including neutron reflectometry, X-ray diffraction, and small angle neutron scattering data analysis. It is well developed for lipid bilayer / protein systems, but can be used for nearly any application where filling space with different molecular components is desired. The package contains the following modules:

- `molgroups.mol`: contains the base code
- `molgroups.components`: tools for dealing with molecular components, which are an adaptation of the `periodictable.Molecule` object, which calculates SLDs from a chemical formula (including labile hydrogens) and a molecular volume. Many of the components are pieces of lipids, e.g. acyl chains or headgroups. The most important object is the `Component` object, which extends `Molecule` with an object length.
- `molgroups.lipids`: a library of lipids for use with lipid bilayer models. Lipids not in the library can be easily constructed from their constituent `Component`s.
- `molgroups.refl1d_interface`: a group of shims connecting the base `molgroups` objects to *Refl1D* layers.

There are two ways to use *molgroups* in *Refl1D*:

- Using the `refl1d_interface` module (e.g. this example script) to define the molecular groups. Advantages include:
  1. The ability to serialize the model (to JSON) for inclusion in metadata and more robust reloading later.
  2. Automatic inclusion of plots in the *Refl1D* GUI interface, including uncertainty plots as DREAM fits run to completion.
- Using `molgroups.refl1d_interface.functionalprofile`. The main advantage of this method is flexibility. The `refl1d_interface` module does not allow all possible types of logic between the base groups. The disadvantages are that GUI plots are not automatically included and models are saved using `pickle` instead of being serializable, leading to less robust (environment-dependent) reloading. In addition, intimate knowledge of the base code may be required.

Introduction to the example script: tiox_dopc_refl1d_interface.py

This example script uses the `refl1d_interface` module. It loads two data sets representing the same DOPC lipid bilayer membrane on a silicon / silicon oxide/ titanium oxide substrate. All the parameters except the scattering length density of the medium are coupled.

The structure of the lipid bilayer membrane is that of a solid-supported membrane. In `molgroups.refl1d_interface`, this object is represented by a `SolidSupportedBilayer` object, with an underlying `molgroups.mol.ssBLM` object.

For illustration purposes, this example also contains three other groups:

1. An additional surface group (`VolumeFractionBox`) on the titanium oxide, to illustrate how other films can be added through the `molgroups` architecture, with the volume fraction and length varying.
2. An overlayer, or free-floating bilayer (`Bilayer`), independent of the surface.
3. A freeform Hermite spline (`Freeform`), e.g. a protein, that can replace some of the lipids if there is overfilling of space in the lipid bilayer region.

Each object in `molgroups.refl1d_interface` has two types of attributes:

1. Parameters. These are connected to *Refl1D* or *bumps* `Parameter` objects and can be fit parameters. For the `SolidSupportedBilayer` object, these may be the completeness (volume fraction) of the bilayer, the thickness of the inner or outer leaflet acyl chains, the distance from the substrate, etc.
2. Reference points. These are calculated parameters and can be used to connect other objects. For example, in this script, the substrate surface, the top surface of the additional surface layer, and the top surface of the bilayer are all used as reference points. These are non-fittable `Parameter` objects, so they can be referenced as part of a parameter definition, but cannot be directly set. In other words, they always appear on the right side of a parameter operation.

Note that the "minimal" script can be used to actually fit the data files provided with a bilayer-only model.

Import section

```
"""Example script using molgroups Refl1D interface objects"""

import numpy as np
from refl1d.names import Parameter, SLD, Slab, FitProblem, load4
from molgroups import components as cmp
from molgroups.refl1d_interface import (SolidSupportedBilayer,
                                        Freeform,
                                        MolgroupsLayer,
                                        MolgroupsStack,
                                        MolgroupsExperiment,
                                        Bilayer,
                                        VolumeFractionBox)

from periodictable.fasta import Sequence
```

Loading data files

Data files are loaded as `refl1d.probe.Probe` objects as usual.

```
## === Probes/data files ===
probe_d2o = load4('ch061_d2o_ph7.refl', back_reflectivity=True, name='D2O')
probe_h2o = load4('ch060_h2o_ph7.refl', back_reflectivity=True, name='H2O')

# Probe parameters
probes = [probe_d2o, probe_h2o]

# Probe parameters
```

```
intensity = Parameter(name='intensity', value=0.8).range(0.65, 1.0)
sample_broadening = Parameter(name='sample broadening', value=0.0).range(-0.003,
0.02)
theta_offset = Parameter(name='theta offset', value=0.0).range(-0.02, 0.02)

# apply background and intensity to all probes
for probe in probes:
    probe.background.limits = (-np.inf, np.inf)
    probe.background.range(-1e-6, 1e-5)
    probe.intensity = intensity

    # if probes support these
    probe.sample_broadening = sample_broadening
    probe.theta_offset = theta_offset
```

Parameter definitions

Parameters do not have to be defined ahead of time, but it can help with making a script more readable.

```
## === Structural parameters ===

vf_bilayer = Parameter(name='volume fraction bilayer', value=0.9).range(0.0, 1.0)
l_lipid1 = Parameter(name='inner acyl chain thickness', value=10.0).range(8, 30)
l_lipid2 = Parameter(name='outer acyl chain thickness', value=10.0).range(8, 18)
l_submembrane = Parameter(name='submembrane thickness', value=10.0).range(0, 50)
sigma = Parameter(name='bilayer roughness', value=5).range(0.5, 9)
global_rough = Parameter(name ='substrate roughness', value=5).range(2, 9)
tiox_rough = Parameter(name='titanium oxide roughness', value=4).range(2, 9)
d_oxide = Parameter(name='silicon oxide layer thickness', value=10).range(5, 30)
d_tiox =  Parameter(name='titanium oxide layer thickness', value=110).range(100,
200)

dz_overlayer = Parameter(name='separation distance of overlayer',
value=10.0).range(0, 30)
vf_overlayer = Parameter(name='volume fraction overlayer', value=0.9).range(0.0,
1.0)

l_surface_group = Parameter(name='surface group length', value=10).range(5, 15)
vf_surface_group = Parameter(name='surface group volume fraction',
value=0.5).range(0, 1)
rho_surface_group = Parameter(name='surface group rho', value=4).range(3, 5)
```

Spline-specific parameters. Here the spline is intended to represent the density of peptide near the lipid bilayer, using the amino acid sequence 'I AM A PEPTIDE IN LIPIDS'. Here we use the `Sequence.D2Osld` method to calculate the SLD of the example peptide in pure solvent.

```
peptide = Sequence(name='peptide',
                   sequence='I AM A PEPTIDE IN LIPIDS',
                   type='aa')
```

```
rhoH_peptide = peptide.D2Osld(1, 0)
rhoD_peptide = peptide.D2Osld(1, 1)
```

The spline control points are defined by lists of parameters. In this case we are only fitting the volume fraction parameters (dVf) and not the positional adjustments (dDp) to the control points.

```
CONTROLPOINTS = 12
SPACING = 15.0
dDp = [0.0] * CONTROLPOINTS
dVf = [0.0] * CONTROLPOINTS
for i in range(CONTROLPOINTS):
    dDp[i] = Parameter(name='dDp'+str(i), value=0.0) #.range(-1 * SPACING / 3., 
SPACING / 3.)
for i in range(0, CONTROLPOINTS-1):
    dVf[i] = Parameter(name='dVf'+str(i), value=0.0).range(-0.001, 1.0)
```

Bulk material definitions and parameters

```
## === Materials ===

# Material definitions
d2o = SLD(name='d2o', rho=6.3000, irho=0.0000)
h2o = SLD(name='h2o', rho=-0.56, irho=0.0000)
tiox = SLD(name='tiox', rho=2.1630, irho=0.0000)
siox = SLD(name='siox', rho=4.1000, irho=0.0000)
silicon = SLD(name='silicon', rho=2.0690, irho=0.0000)

# Material SLD parameters
d2o.rho.range(5.3000, 6.36)
h2o.rho.range(-0.56, 0.6)
tiox.rho.range(1.2, 3.2)
siox.rho.range(2.8, 4.8)
```

Molecular group definitions and connections

This is an example of how to define a lipid based on the base `Component` objects available in `molgroups.components`. Here a DOPC lipid is constructed from the PC headgroup, 2 oleoyl tails, and methyl groups. The lipid list will be used later. It can contain any number of lipids, with the ratios defined by `lipid_nf`.

```
## === Molecular groups ===

DOPC = cmp.Lipid(name='DOPC', headgroup=cmp.pc, tails=2 * [cmp.oleoyl], methyls=
[cmp.methyl])
lipidlist = [DOPC]
lipid_nf = [1.0]
```

The contrast function

Frequently in soft matter problems, the scattering length densities of the soft materials depend on the contrast, due to labile hydrogens. Thus the molgroups structure definition is wrapped in a function that is contrast-aware. Each part of this function will be described below.

```
def bilayer(substrate, contrast):

    blm = SolidSupportedBilayer(name='bilayer',
                                overlap=overlap,
                                lipids=lipidlist,
                                inner_lipid_nf=lipid_nf,
                                outer_lipid_nf=lipid_nf,
                                rho_substrate=tiox.rho,
                                l_siox=0.0,
                                vf_bilayer=vf_bilayer,
                                l_lipid1=l_lipid1,
                                l_lipid2=l_lipid2,
                                l_submembrane=l_submembrane,
                                substrate_rough=tiox_rough,
                                sigma=sigma)

    surface_group = VolumeFractionBox(name='surface group',
                                      z=blm.substrate_surface + 0.5 *
l_surface_group,
                                      rhoH=rho_surface_group,
                                      rhoD=rho_surface_group,
                                      volume_fraction=vf_surface_group,
                                      length=l_surface_group,
                                      sigma_bottom=tiox_rough,
                                      sigma_top=tiox_rough)

    blm.l_submembrane = surface_group.length + l_submembrane

    ol = Bilayer(name='overlayer',
                lipids=lipidlist,
                inner_lipid_nf=lipid_nf,
                outer_lipid_nf=lipid_nf,
                startz=blm.outer_headgroup_top + dz_overlayer,
                vf_bilayer=vf_overlayer,
                l_lipid1=l_lipid1,
                l_lipid2=l_lipid2,
                sigma=sigma)

    spline = Freeform(name='spline',
                      dSpacing=SPACING,
                      startz=surface_group.top_surface,
                      Dp=dDp,
                      Vf=dVf,
                      rhoH=rhoH_peptide,
                     rhoD=rhoD_peptide,
                     sigma=sigma,
                     nf=vf_bilayer)

    mollayer = MolgroupsLayer(base_group=blm,
                             add_groups=[surface_group, ol],
                             overlay_groups=[spline],
                             thickness=thickness,
                             contrast=contrast,
                             name='bilayer layer ' + contrast.name,
                             contrasts=[d2o, h2o],
                             substrate = substrate)

    return MolgroupsStack(substrate=substrate,
                          molgroups_layer=mollayer,
                          name=mollayer.name)
```

Solid Supported Bilayers

Defining the solid-supported bilayer, which comprises space-filling substrate group (volume fraction = 1) and a lipid bilayer membrane. A few notes:

- The `overlap` parameter is the amount of overlap between the substrate group and the underlying *Refl1D* `Slab`. Later, this amount will be subtracted from the thickness of that slab.
- The ssBLM can support arbitrarily complex lipid compositions, defined by a list of `Lipid` groups and lists (of equal length) of the number fraction of those lipids. Here there is only one lipid so the number fractions are unity. Number fractions can differ by leaflet. Note that the number fractions are exposed as `Parameter` objects that can be fit.
- `rho_substrate` is a parameter linking the SLD of the underlying *Refl1D* `Slab` to that of the substrate group of the ssBLM.
- `l_siox` is set to zero here. This is used when the underlying substrate is silicon and a native oxide is present. As shown below, *molgroups* offers a different way of adding a surface group.
- `vf_bilayer` is the completeness (volume fraction) of the bilayer
- `l_lipid1` is the thickness of the inner acyl chains
- `l_lipid2` is the thickness of the outer acyl chains
- `l_submembrane` is the separation between the substrate surface and the bottom of the inner leaflet headgroups. The bottom is defined as the position that is a distance, from the inner hydrophobic interface, equal to the weighted average length of the headgroups.
- `substrate_rough` is the roughness of the substrate layer
- `sigma` is the roughness of the bilayer (sigma, not FWHM)

```
    blm = SolidSupportedBilayer(name='bilayer',
                          overlap=overlap,
                          lipids=lipidlist,
                          inner_lipid_nf=lipid_nf,
                          outer_lipid_nf=lipid_nf,
                          rho_substrate=tiox.rho,
                          l_siox=0.0,
                          vf_bilayer=vf_bilayer,
                          l_lipid1=l_lipid1,
                          l_lipid2=l_lipid2,
                          l_submembrane=l_submembrane,
                          substrate_rough=tiox_rough,
                          sigma=sigma)
```

Volume Fraction box functions

Here we define a partial film on the surface of the titanium oxide substrate layer:

- `z` is the position of the center of the film (modeled as a "box function" defined by two error functions). Here we use a reference point `blm.substrate_surface`, the surface of the titanium oxide surface, plus half the length of the new surface group which is a predefined parameter.
- `rhoH` is the SLD of the group in pure H2O.
- `rhoD` is the SLD of the group in pure D2O. For a D2O/H2O mixture, the SLD of the material is calculated by linear interpolation.
- `volume_fraction` is the volume fraction of the film.
- `length` is the thickness of the film.
- `sigma_bottom` is the roughness of the bottom of the film
- `sigma_top` is the roughness of the top of the film

```
    surface_group = VolumeFractionBox(name='surface group',
                                      z=blm.substrate_surface + 0.5 *
l_surface_group,
                                      rhoH=rho_surface_group,
                                      rhoD=rho_surface_group,
                                      volume_fraction=vf_surface_group,
                                      length=l_surface_group,
                                      sigma_bottom=tiox_rough,
                                      sigma_top=tiox_rough)
```

Because we now have an intervening film, the meaning of the submembrane thickness has changed:

```
    blm.l_submembrane = surface_group.length + l_submembrane
```

Free floating bilayers

Perhaps our sample preparation is imperfect and we have a lipid overlayer. Differences from the ssBLM model are:

- the absence of the substrate
- `startz` is the bottom of the inner headgroups. This is connected to the underlying ssBLM object's `outer_headgroup_top` reference point, plus a separation distance parameter `dz_overlayer`.
- The thicknesses and roughness are assumed to be the same as the underlying bilayer.

```
    ol = Bilayer(name='overlayer',
                 lipids=lipidlist,
                 inner_lipid_nf=lipid_nf,
                 outer_lipid_nf=lipid_nf,
                 startz=blm.outer_headgroup_top + dz_overlayer,
                 vf_bilayer=vf_overlayer,
                 l_lipid1=l_lipid1,
                 l_lipid2=l_lipid2,
                 sigma=sigma)
```

Freeform models

Just for fun, we add in a freeform model. The `Freeform` object is designed to connect smoothly to error function roughnesses of underlying layers. This might represent protein density, for example:

- `dSpacing` is the spacing between spline control points.
- `startz` is the position of the underlying interface to connect (i.e. control point 0)
- `Dp` is a **list** of parameters representing positional adjustments to the control points
- `Vf` is a **list** of parameters representing the volume fraction at each control point
- `rhoH` and `rhoD` are the SLDs of the material in pure H2O and D2O, respectively.
- `sigma` is the roughness of the underlying film
- `nf` is a parameter common to all *molgroups* objects and represents the number fraction, an overall scaling parameter. In this case, it is linked to `vf_bilayer` so that the amount of the freeform material, e.g. protein, scales with the bilayer completeness.

```
    spline = Freeform(name='spline',
                      dSpacing=SPACING,
                      startz=surface_group.top_surface,
                      Dp=dDp,
                      Vf=dVf,
                      rhoH=rhoH_peptide,
                      rhoD=rhoD_peptide,
                      sigma=sigma,
                      nf=vf_bilayer)
```

Molgroups samples

Then we create a layer that contains the *molgroups* objects. It is also okay to forego the use of `make_samples` and create the objects independently. The `MolgroupsLayer` object can be thought of as a canvas on which the individual molecular groups are added in a piecewise fashion. Its arguments are:

- `base_group` is the group that connects to the underlying `Slab` object, in this case `layer_tiox`. It is updated first.
- `add_groups` are independent groups that are updated in the order they are given.
- `overlay_groups` are similarly added; however, if there is any overfilling of the canvas anywhere, the excess volume fraction will be removed from the already added groups (`base_group` and `add_groups`).

- `thickness` is the extent of the canvas (the z-axis). In general this should be large enough to contain the entire groups even for the most extreme values of the parameters.
- `substrate` is a *Refl1D* `Slab` or layer stack that represents the substrate. Make sure that any thicknesses of underlying slabs is reduced by `overlap`.

The returned sample objects are *Refl1D* stacks that combine the substrate, the molgroups canvas, and an automatically generated semi-infinite layer with bulk contrast SLD.

```
    mollayer = MolgroupsLayer(base_group=blm,
                            add_groups=[surface_group, ol],
                            overlay_groups=[spline],
                            thickness=thickness,
                            contrast=contrast,
                            name='bilayer layer ' + contrast.name,
                            contrasts=[d2o, h2o],
                            substrate = substrate)

    return MolgroupsStack(substrate=substrate,
                        molgroups_layer=mollayer,
                        name=mollayer.name)
```

Creating the samples using the contrast function

First we define the substrate layers. Note how the thickness of `layer_tiox` is reduced by `overlap` to account for the part in the Molgroups layer

```
## == Sample layer stack ==

layer_silicon = Slab(material=silicon, thickness=0.0000, interface=global_rough)
layer_siox = Slab(material=siox, thickness=d_oxide, interface=global_rough)
layer_tiox = Slab(material=tiox, thickness=d_tiox - overlap, interface=0.00)

substrate = layer_silicon | layer_siox | layer_tiox
```

Then we use the contrast function to create the samples

```
sample_d2o, sample_h2o = [bilayer(substrate, contrast) for contrast in [d2o, h2o]]
```

Create the *Refl1D* experiments

Creating *Refl1D* experiments looks similar to a typical script, except we use the `MolgroupsExperiment` object. This automatically registers custom plotting routines for visualization in the *Refl1D* GUI.

```
## === Problem definition ===
step = False
```

```
STEPSIZE=0.5

model_d2o = MolgroupsExperiment(sample=sample_d2o, probe=probe_d2o, dz=STEPSIZE,
step_interfaces = step)
model_h2o = MolgroupsExperiment(sample=sample_h2o, probe=probe_h2o, dz=STEPSIZE,
step_interfaces = step)

problem = FitProblem([model_d2o, model_h2o])
```