Repeating layers Api, slicing + interface profiles, Issue #2436

Internal Report
Scientific Computing Group of JCNS-MLZ

Last change May 15, 2020.

1 General

Design goal is to implement an Api that fulfills the following requirements

- The new Gui allows to define a multilayer structure with repeating internal content
- Currently there is no corresponding functionality within the core
- Therefore, need functionality to define a substructure
- This substructure should then be added to a multilayer with a defined number of repetitions
- How about fitting? Should a number of repetitions be fittable?
- How about parameter variations, i.e. varying thickness? Varying roughness? I have seen this in the literature. Also related to the suggestion by Artur that the roughness in the whole sample should in principle be parametrized in terms of just a few parameters. This seems to get very tricky though.
- I think the fitting is not really an issue. All solved via a callback method that generates an according sample

2 RefNX

- RefNX has such object called `Stack`, code taken from RefNX manual

```
# Make a multilayer by using a Stack Component
stack = Stack()
stack |= ti_layer
stack |= ni_layer
stack.repeats.value = 10

structure = air | stack | si(0, 0)
```

- `stack.repeats` is a parameter, seems that it is fitable in RefNX
- It seems that the `Stack` objects can be stacked
- RefNX supports varies interface profiles: Erf, Linear, Exponential, Tanh, Sin, Step
- Selection happens via

```
for i in range(NLayers):
  sample = sample | ti_Layer | ni_Layer

sample = sample | si_Substrate

for i in range(2 * NLayers + 2):
  sample[i].interfaces = refnx.reflect.Tanh()
```

- The precision can be adjusted with a shady parameter (the smaller the more precise)

```
  sample.contract = 0.0000001
```

- Agreement between Sliced + Nevot-Croce solution is not so satisfactory, had the same observation in own experiments
- In that light, the Tanh sliced solution agrees okay with BornAgain
- Different profiles in the same sample are possible

3 Design for BornAgain

- Question: Define number of repetitions in the substructure, or when adding to multilayer? Or both, i.e. default in substructure that can be overwritten when adding to a multilayer?
- One option for a rather involved refactoring would be to allow a multilayer to be added to a multilayer
  - This would then allow to specify a number of repetitions in the creation of such, or when adding it.
  - A downside is, that this requires some changes in the behavior of the current `MultiLayer` class, such as to change the checking of the semi-infinity condition for the first layer, or the absence of roughness for the first layer.
  - As a consequence, the complexity of some code would increase, increasing the risk of errors
- The second option is to introduce a new class, i.e. `MultiLayerStack`.
  - To this class one would, as for a `MultiLayer`, add layers, with and without roughness
  - Then we should also add the possibility to add another `MultiLayerStack` to a `MultiLayerStack`. This operation should obey the same logic as when adding it to a `MultiLayer`

3.1 Implementation

- Prefer second option, to this end do:
- Introduce new class `MultiLayerStack`
- Implement `MultiLayerStack::add(Layer &, LayerRoughness = nullptr )`
- Implement `MultiLayerStack::add(MultiLayerStack & )`
- Implement `MultiLayer::add(MultiLayerStack &)`
- **Idea:** Implement a unified interface for adding things to a `MultiLayer`, i.e. an overloaded `add` method. This method should accept either `Layer` objects, with and without roughness or a `MultiLayerStack`.

- **Unconventional idea:** Do not implement this in C++, but instead in a newly developed Python interface. This could be part of a large refactoring where the generated interface is replaced with a more *pythonic* interface (as suggested by Thomas back in January).

3.2 Roadmap

- Implement according to 3.1
- Adapt Ti-Ni bilayer example to show usage of the new api (as new example, alternative, where to show?)
- Use it in the fitting example (fitting/ex03/FitSpecularBasics.py)
- Create some tests (Unit test for `MultiLayerStack`, test sample creation. possibly take functional test, copy and rewrite with new api check against each other, or stored result)