iFit : a simple generic data analysis framework

Example use with full Reitveld analysis using virtual experiments.

E. Farhi <farhi@ill.fr>

<ifit.mccode.org>

History and development

First release August 2011, but designed 2 prototypes since 2003.
Actively developed since 2009.

Development team: me
Beta-testers and friendly users: *P. Willendrup, L. Udby, H. Jacobsen*.

Licence: EUPL. ILL software.

Funding: none (~10% of my time).

Project size: Core is about 25 kLOC. Total 55 kLOC with docs and contrib.
Code is 3 Mb size only + doc and example data files.

Stability: good (most features are implemented and functional).

Rationale: one object to hold data sets, one to hold models, and methods to handle both (load, save, pure math, fitting, plot, ...)

Built with Matlab: distributed as well as standalone executable (no need for Matlab license, for *Linux, MacOSX, Windows*). Debian packages.

What can iFit do for you

- Handle data sets typically up to a few Gb.
- Import/export any text-based and a variety of binary data file (*load/save*)

  including NeXus/HDF, EDF, CBF/imgCIF, SPE, McStas, event files...
- Display data sets (*plot*) in 1D, 2D, 3D
- Apply mathematical operators (50+) directly on data sets (+-/*, ...).
- Carry signal, axes, error bars, … along operations
- Record data sets history along operations
- Optimize any multi-parameter problem (*fmin*).
- Fit model function onto data sets (*fits*).

iFit consists in the following parts :

iData

to store data sets (e.g. files), display and manipulate them.
Any dimensionality/size

iFunc

to store models (e.g. fit functions), display and fit to data sets.
Symbolic evaluation, any dimensionality

Libraries for Loaders, Optimisation and pre-defined Models

Examples

Get documentation
>> doc(iData)

Import data files (37 formats supported)
>> a=iData('filename'); b=iData('directory'); c=iData(''); *% file selector*

Manipulate data sets (and models)
Just like numbers, *e.g.* d=a+b
Resolves axes intersection, rebinning, dimension mismatch .. transparently
List methods with *methods iData* and *methods iFunc*

Display data sets
>> disp(a); plot(a);

Save data sets (29 formats supported)
>> save(a, 'filename.ext'); save (iData,'formats') *% lists all supported formats*

Fit to a model
>> fits(a, gauss); fits(a, gauss+gauss); fits(iData) *% get the list of models*
>> m=iFunc('expr. p, x,y,z...'); edit(iFunc); *% edit/create a new model*
>> m=convn(gauss,lorz); plot(m); *% a Voigt function*

Powder Rietveld refinement basics

Currently, a powder structure is refined from a diffractogram as follows:

- **Import** the raw data.
- **Normalize** detector efficiency with a reference sample (Vanadium).
- **Subtract** empty cell making use of the sample transmission.
- Define an **initial structure** *SpcG abc, αβγ XYZ...*
- Estimate the instrument **resolution** parameters *UVW* from measurement configuration $\alpha_{1\text{-}3}$ η and Caglioti equations. These *UVW* parameters are often pre-determined from the instrument calibration.
- For each powder structure parameter set, compute a perfect diffractogram and **convolute** each Bragg peak by a Voigt function of width=*f(θ,UVW)* depending on the angle. Vertical divergence adds **asymmetry** and **broadening**. The background level is subtracted for each iteration (e.g. quadratic).
- Minimize the **least-square** criteria using e.g. gradient or simulated annealing optimisers. The whole process is the Rietveld method.

In practice, the background level (not a constant) and peak asymmetry are additional parameters together with the powder structure.

Powder Rietveld refinement theory

G. Caglioti et al, *NIM* **3** (1958) 223
A.W. Hewat, *NIM* **127** (1975) 361
L.D. Cussen, *NIM A* **554** (2005) 406

Integral

$$L \propto L_0 = \frac{\alpha_1 \beta \alpha_2 \alpha_3 \cot\theta_M}{\sqrt{\alpha_1^2 + \alpha_2^2 + 4\beta^2}}$$

Width

$$H_{Peak} \propto \frac{L}{A_{1/2}}$$

$$A_{1/2}^2 = U \tan^2\theta_S + V \tan\theta_S + W$$

$$U = \frac{4(\alpha_1^2\alpha_2^2 + \alpha_1^2\beta^2 + \alpha_2^2\beta^2)}{\tan^2\theta_M(\alpha_1^2 + \alpha_2^2 + 4\beta^2)}$$

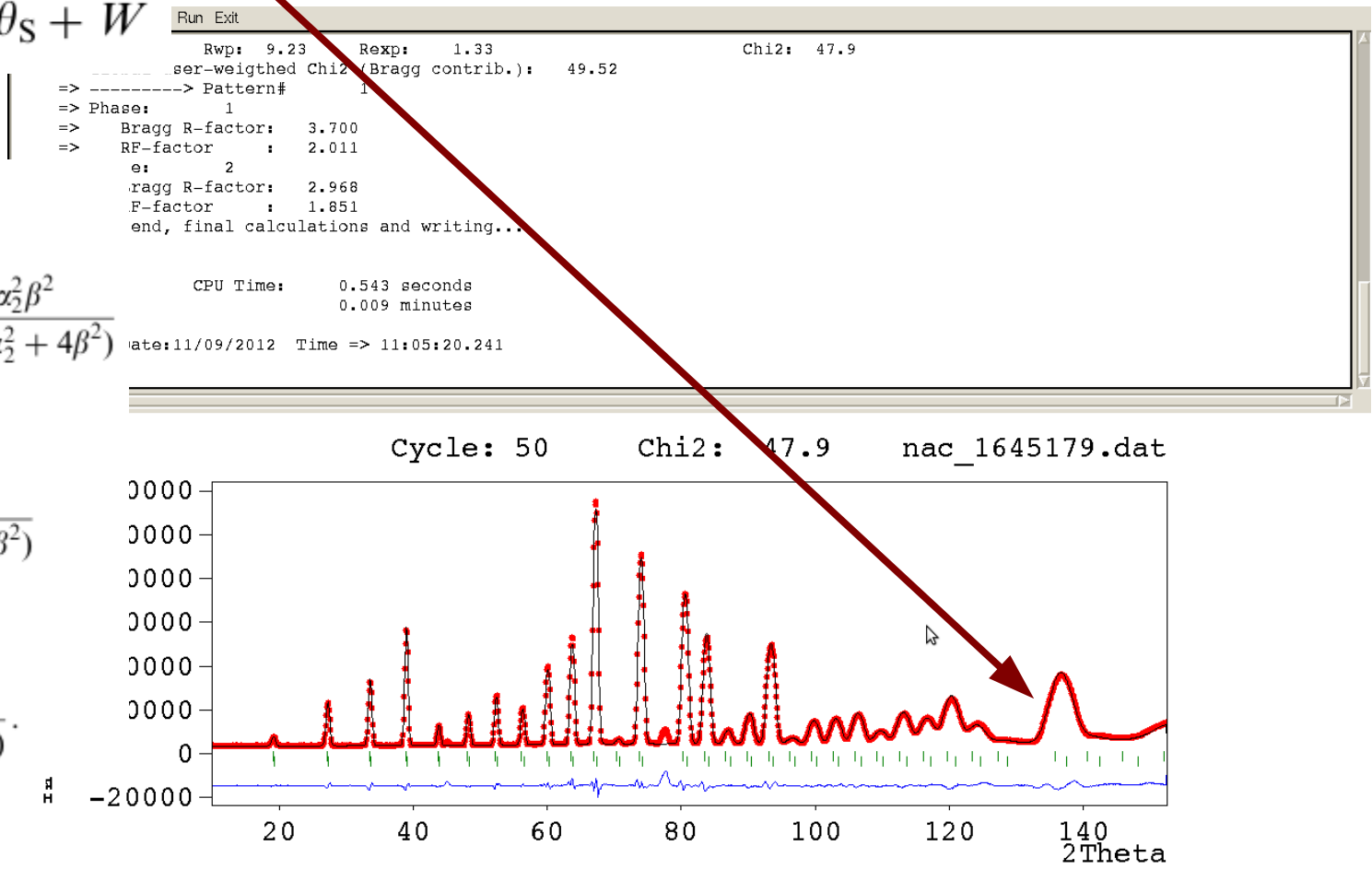
$$= \frac{4}{\tan^2\theta_M}\left(\frac{1}{\alpha_1^2} + \frac{1}{\alpha_2^2} + \frac{1}{\beta^2}\right)\frac{\alpha_1^2\alpha_2^2\beta^2}{(\alpha_1^2 + \alpha_2^2 + 4\beta^2)}$$

$$V = \frac{4\alpha_2^2(\alpha_1^2 + 2\beta^2)}{\tan\theta_M(\alpha_1^2 + \alpha_2^2 + 4\beta^2)}$$

$$= \frac{4}{\tan\theta_M}\left(\frac{1}{\beta^2} + \frac{2}{\alpha_1^2}\right)\frac{\alpha_1^2\alpha_2^2\beta^2}{(\alpha_1^2 + \alpha_2^2 + 4\beta^2)}$$

$$W = \alpha_3^2 + \frac{(\alpha_1^2\alpha_2^2 + 4\beta^2\alpha_2^2)}{(\alpha_1^2 + \alpha_2^2 + 4\beta^2)}$$

$$= \alpha_3^2 + \left(\frac{1}{\beta^2} + \frac{4}{\alpha_1^2}\right)\frac{\alpha_1^2\alpha_2^2\beta^2}{(\alpha_1^2 + \alpha_2^2 + 4\beta^2)}.$$



Is it possible to reduce the number of parameters to fit and improve the refinement quality/robustness ?

Yes (I think)

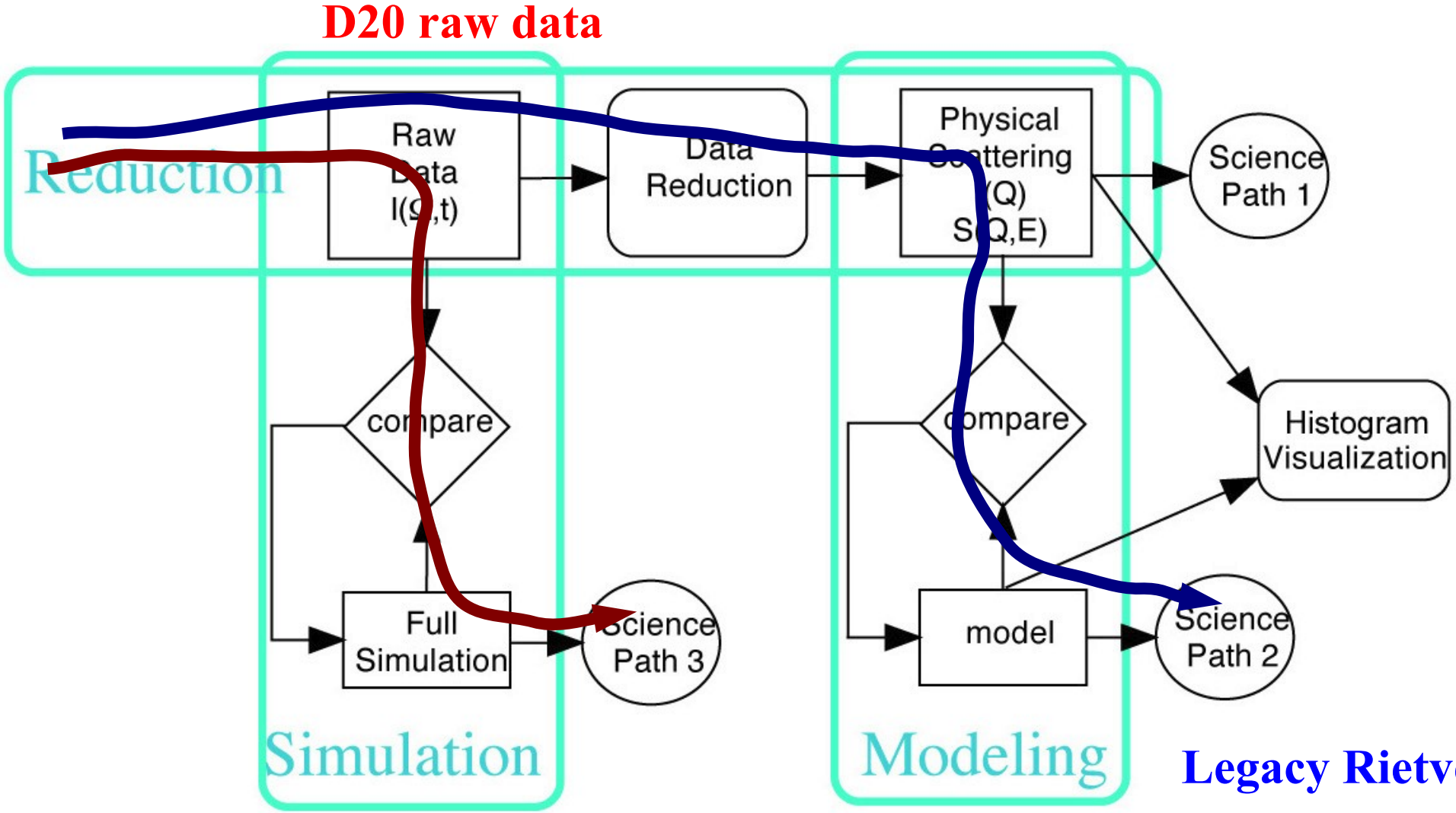
- Use ***McStas*** to model the diffractometer, including container, …
  No more UVW, no explicit analytical approximation.
- Use the *PowderN* (McStas sample) to model the diffraction from the powder. Requires a reflection list.
- Use ***CrysFML*** (FullProf) routines to relate the powder structure parameters to the reflection list needed by *PowderN*
- Use ***iFit*** to transparently import both measurement and simulated data sets, compare them (least-squares) and optimise parameters.

Let's try that …

Resources: 2 ½ months student placement this summer
Student: ***Marta Bolsa-Ferruz***

Legacy Reduction-Analysis *vs.* Full simulation

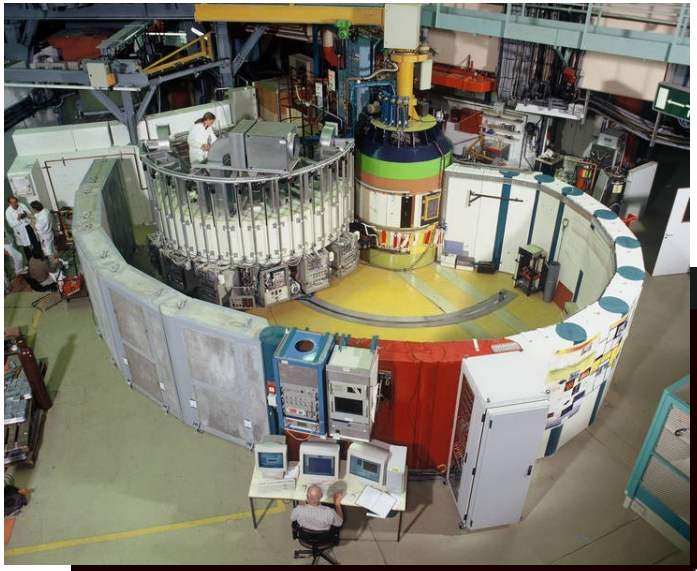
We have used iFit to test a full simulation data analysis procedure for powders.



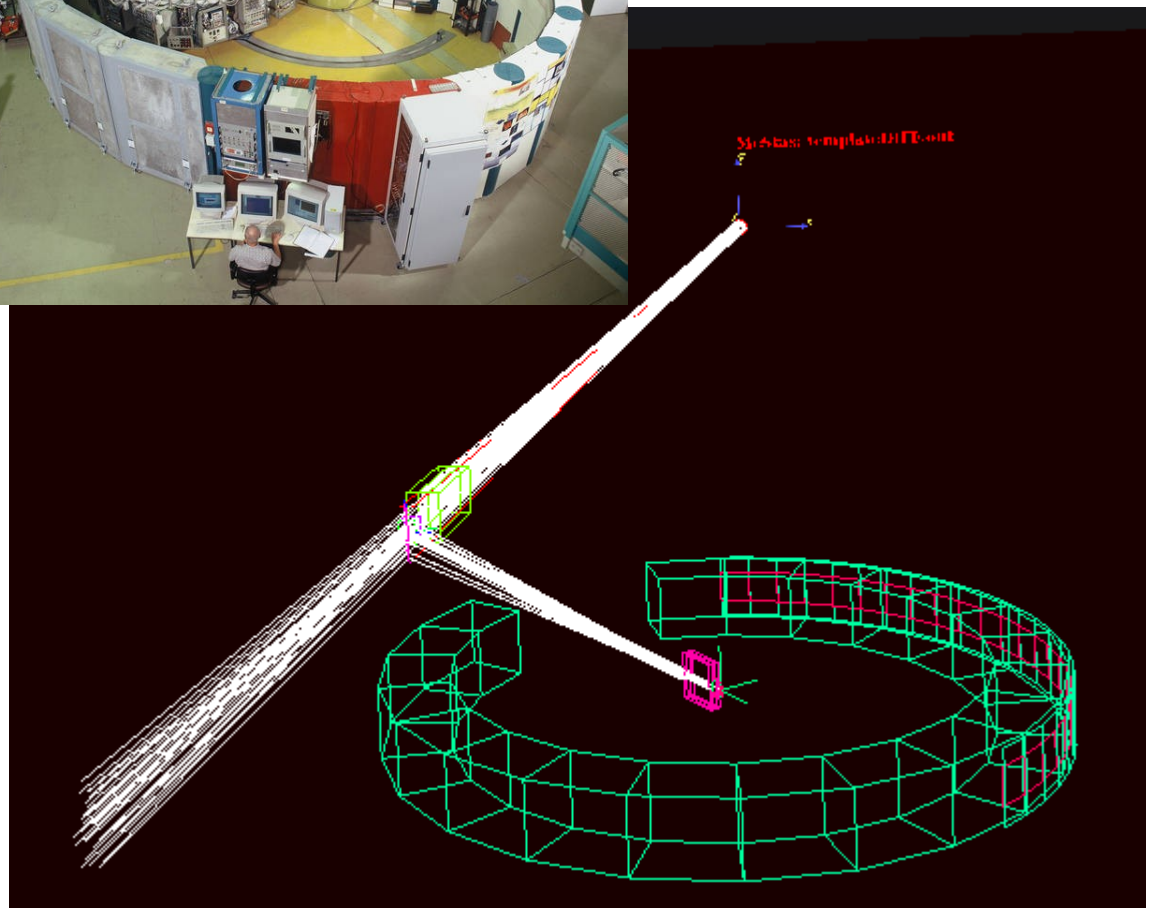
McStas with CrysFML calls
Optimisation with iFit (used swarm here)
Least-Square criteria

Diffratometer virtual experiment

We set-up a full simulation of a power sample in a D20 model.

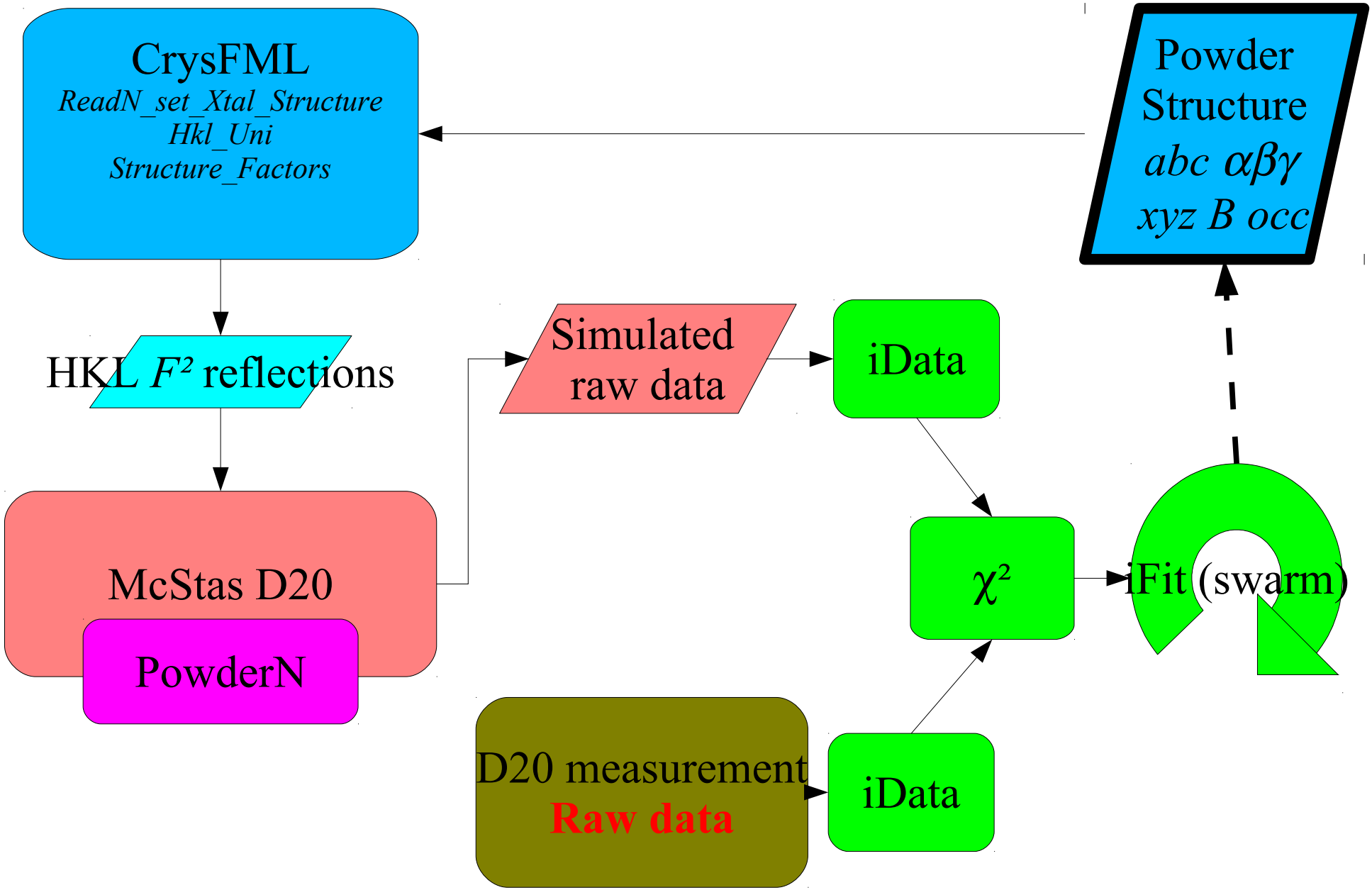


D20 thermal diffractometer:
17 m from the ILL RHF reactor
λ = 2.41 Å, Cu002, take-of 42°
No collimation $\alpha_{1\text{-}3}$
Powder $Na_2Ca_3Al_2F_{14}$ ϕ=7mm
V container
Detector: ^{3}He microstrip 5-150°



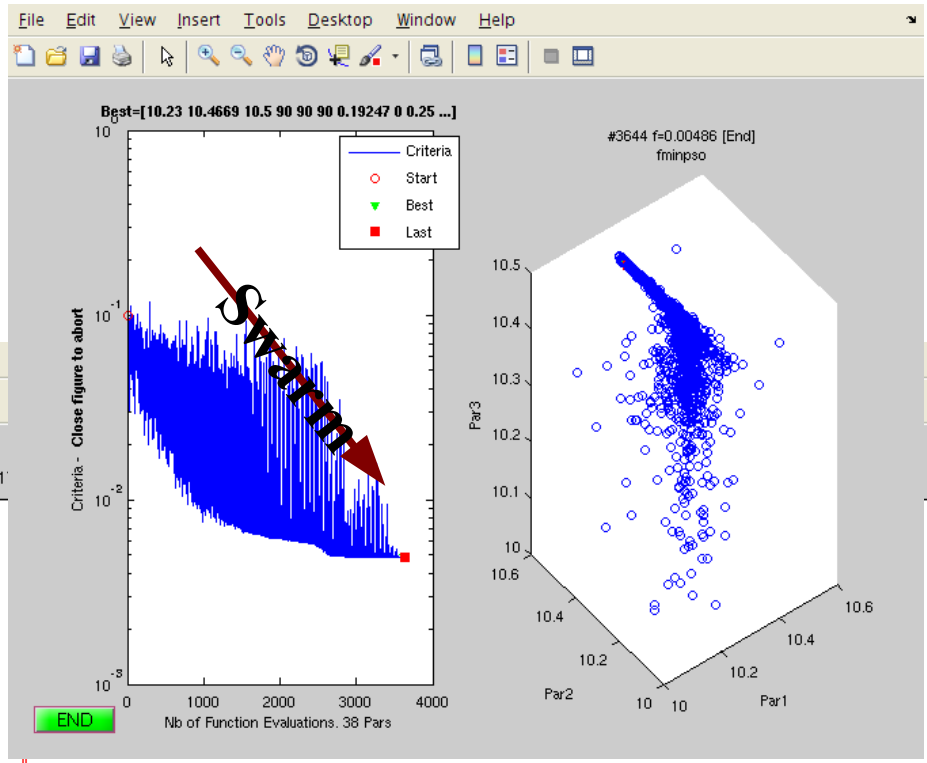
Virtual experiment:
Built with *templateDIFF* from McStas 1.12c using the PowderN component.

Full simulation Rietveld



Full simulation Rietveld : refinement

iFit displays the refinement steps in criteria value, *abc* values, and the two diffractograms (measured+simulated). Then we wait for convergence...



Results and Lessons learned with the new Rietveld

Results:

- The procedure is **fully functional** and automatic. Only requires the initial structure and the instrument model (which is given with *McStas*).
- The **only** added background is a constant, to model the reactor hall noise level.
- The final diffractogram fits the data (convergence achieved). Peaks shapes and background are well accounted for.
- Does not contain any 'trick', to fit asymmetry, background, non Gaussian/Voigt shapes, ...

But:

- The refinement takes much longer than a legacy one (100x longer)
- The refinement is currently less satisfactory than the legacy one.

This new Rietveld methodology **will be greatly improved** in the next months (speed, accuracy), and may represent a usefull **complementary method** to e.g. FullProf and GSAS. Also, it may prove particularly efficient when peaks are **overlaping**, or when **background** level is difficult to fit, for instance with a mixed liquid+powder phases where we would use PowderN+Isotropic_Sqw.

Conclusion

About iFit

- A lightweight, simple, intuitive yet efficient framework.
- Deployable as binary and Matlab source code
- Can link seamlessly to other software, e.g. Fullprof, McStas, …
- Everything needed for data reduction and analysis, *any* dimensionality and data type.

About "virtual experiment Rietveld"

- Very promising.
- Intrinsically includes instrument effetcs (resolution, environment, spurions, …).
- Will be extended to Laue and SANS experiments.

<http://ifit.mccode.org>

E. Farhi et al, to appear in *J. Neut. Res.* **17** (2012).

<http://www.mcstas.org>

<http://forge.ill.fr/projects/crysfml>