Introduction to ufit

- a convenient scattering data evaluation tool

G. Brandl, P. Cermak
Forschungszentrum Jülich
What is ufit?

• Started as a private collection of data readers for evaluation scripts
• Evolved to a fitting tool with GUI
• Often repeated during beamtimes:
  – read in data from 1-n files
  – apply a model (Gauß or more) and fit
• Lots of solutions from different institutes, often idiosyncratic
• ufit tries to be more universal, and cover more than “just” quick view during experiments
Basic features

• Reads and fits mostly 1-dimensional data
• Complex models, fitting multiple datasets to one model
• Written in Python (2.7, recently ported to 3.5+ compatibility)
• Uses standard parts of scientific stack:
  – numpy, scipy
  – matplotlib
  – Qt 4 (porting to Qt 5 planned)
  – some specialized libraries for fitting
• Two interfaces: Scripting and GUI
Two interfaces

• GUIs are nicely discoverable and easy for quick look at data
  – but horribly tedious and error-prone for reproducible treatment
• Scripting is the most reproducible way of data evaluation
  – this is especially nice with notebooks: shown in session tomorrow
• ufit provides both interfaces, with shared functionality where useful
Sample scripting usage

# import all ufit API plus pylab
from ufit.lab import *

# set a template, so that data can be referenced by number only
set_datatemplate('path/to/data%04d.dat')
# read one dataset from a file, with given X and Y columns
data = read_data(104, 'A3', 'CNTS')

# create a model of a simple Gaussian peak with given initial guess
model = Background() + Gauss('peak', pos=97.5, ampl=100, fwhm=0.5)

# fit the model, then print and plot the result
result = model.fit(data)
result.printout()
result.plot()
show()
Supported data formats

- Simple multicolumn data (space/comma separated)
- MLZ NICOS (old/new)
- MLZ TRISP
- LLB binary 3-axis data
- ILL format 3-axis data
- ANSTO TAIPAN

- Add new formats: needs
  - reader
  - guesser
GUI: Main window
Basic GUI features

- Save/load sessions (everything you see)
- Group data together into a tree
- Customize, save, print plots
- IPython console: customize plots even further!
- Export raw data, fits, fit parameters as ASCII
- Save data plus model as Python for further use with scripting interface
GUI: Data loading

Data loading

Data template: auto
Data format: auto
Columns:
- X: auto
- Y: auto
- DY: auto
- Monitor: auto
- Monitor scale: 38000
Numbers: 4091-4092
Merge precision: 0
Merging type: Rebin
Put into group: Default

Load new data
Manage

Select one or more datasets:
- Default – 2 items
  1. Vanadium
     n124-01-1114-4091
  2. Vanadium
     n124-01-1114-4092
Data loading

- Main dataset columns: X, Y, ΔY, Normalization
- Relevant columns are normally autodetected
- All other columns, and scalar metadata, are also loaded and can be used later
- Mini-language for loading multiple files ("numors"), e.g. "4000>4010, 4011+4013, 4014-4017"
- Configurable merging of similar scans
- Same in scripting interface!
GUI: Data operations

- **Data operations** tab
  - Set title:
  - Set short name:
  - Clone to new:
  - Merge/Rebin (destructive):
  - Add a constant:
  - Scale Y with constant:
  - Shift X values:
  - Scale X with constant:
  - New moniter scale:
  - Subtract dataset:
  - Mask out points:
  - Fit limits:

- **Data operations options**
  - Float merge
  - Rebin
  - Add
  - Scale
  - Shift
  - Change

- **Graph**
  - Plot of counts vs. 2θ with peaks at 31.36° and 31.9°
GUI: Model creation

Data operations | Modeling | Fitting

Model definition: Gauss peaks only model

Add predefined model:
- Gauss
- Lorentz
- Voigt
- PseudoVoigt
- DH0
- Background
- SlopeBackground

Click "Check" to check model definition.

Reset | Check | Apply

Load new data

Select one or more datasets:
- Default – 2 items
  1. Vanadium
     in12-4-01-11144091
  2. Vanadium
     in12-4-01-11144092
Models

• Defined in Python with classes representing functions with arguments as parameters
  Gauss(...), DHO(...), ...

• Functions have a name that distinguishes same functions in the same model: Gauss('p1') + Gauss('p2')

• Functions can be added, multiplied, etc.

• Completely custom models can also be created referencing a Python lambda function

• Same in scripting interface!
GUI: Fitting

Converged. Reduced chi^2 = 3.27.

<table>
<thead>
<tr>
<th>Param</th>
<th>Value</th>
<th>Error</th>
<th>Fix</th>
<th>Expr</th>
<th>Min</th>
<th>Max</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>bgd</td>
<td>48.309</td>
<td>± 4.818</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>peak_pos</td>
<td>31.136</td>
<td>± 0.0089719</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>peak_ampl</td>
<td>1102.1</td>
<td>± 32.231</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>peak_wmhm</td>
<td>0.85075</td>
<td>± 0.015635</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parameters

• Initial guess from model

• Many options for parameters:
  – variable
  – variable with limits
  – fixing it after fitting
  – fixed, with Python expression
  – fixed, depending on value from metadata

• Same, and more, in scripting interface!
GUI: Multi-dataset options
Powerful multi-dataset options

• Copy one model to all datasets
• Fit all datasets at once
• Create a 2-dimensional mapping
• Create a dataset from parameters

• Has to be repeated after data changes...
GUI: Mappings
GUI: Data browser
Data browser

- Quick selection (important during experiments!)
- See all data from a directory
- Select many datasets, load directly from there
- Custom display based on metadata within files (temperature? kf? monochromator scattering order?)
Links

- Home:
  http://wiki.mlz-garching.de/ufit:index

- Documentation:
  http://pythonhosted.org/ufit/

- Tracker:
  http://wiki.mlz-garching.de/ufit:support
Thank you for your attention!