DIRECTION DES SCIENCES DE LA MATIERE
IRAMIS
Laboratoire Interdisciplinaire sur l'Organisation Nanométrique et Supramoléculaire

PySaxs
A Python module and GUI for SAXS data treatment

Olivier Taché

Collaborative work with :
O. Spalla, A. Thill, D. Sen, D. Carriere, F. Testard

Outline

Context :

- CEA-LIONS
- Small Angle X-Rays Scattering
- SAXS at LIONS
- SAXS data treatment

What is pySAXS ?

User Interface : GuiSAXS

- plots
- data treatment
- fitting by models

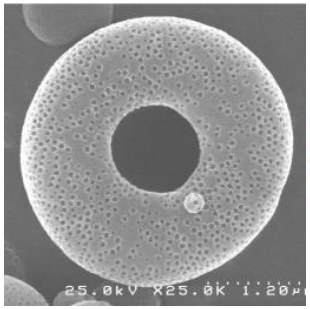
CEA / LIONS at Saclay

LIONS :

Laboratoire Interdisciplinaire sur l'Organisation Nanométrique et Supramoléculaire

(*Interdisciplinary Laboratory on Nanometric and Supramolecular Organization*)

Fundamental research on nanochimistry and nanoscience



- Knowledge of "nano-objects"
- Organized fluids and nano-structured solids
- 40 researchers (chemists, physicists, theoricians, computer scientists,…)
- Laboratory experiments
- High usage of synchrotrons and large instruments (Soleil, ERSF, LLB, ILL)

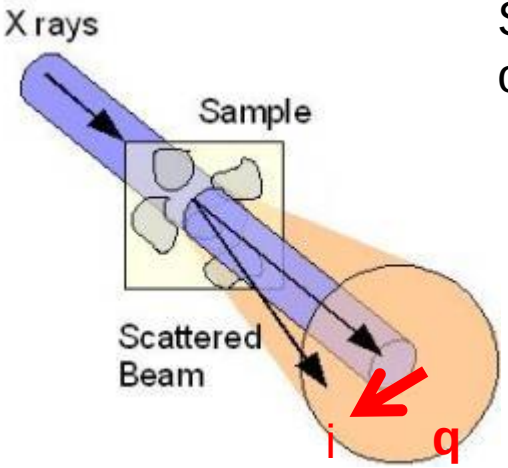
needs in computing :

- A flexible and powerful control-command system
  → TANGO with Python
- standardization for data processing
  →programming language **Python**
  → PySAXS for saxs data treatment

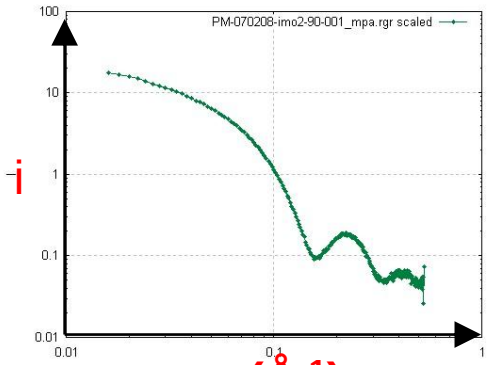
→ Internal Python learning courses for all new members

Small Angle X-Ray Scattering

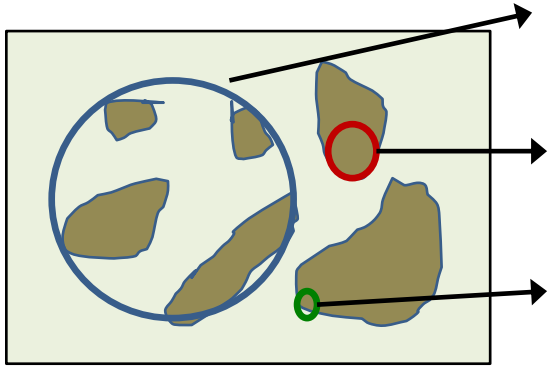
- A technic for characterization



Scattered X-Rays gives informations about fluctuation of electronic densities on heterogeneous matter.



$$q = \frac{4\pi}{\lambda} \sin \frac{\theta}{2}$$



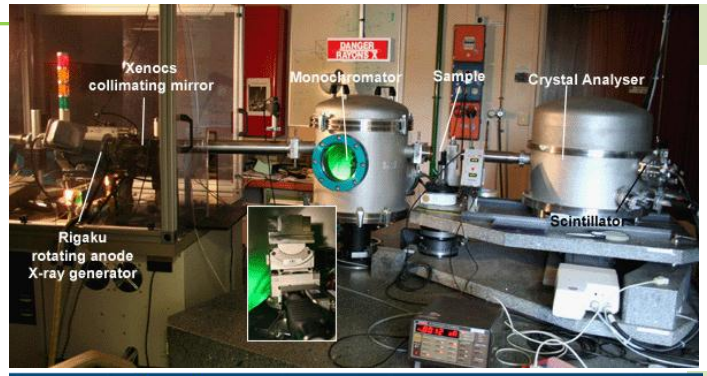
High q :
there is a contrast only at the **interface** between the two medias. information about the surfaces.

Intermediary zone :
elementary bricks in the systems. The **form factor** P(q) can be measured (size, shape and internal structure of one particle).

Low q :
the structural order can be obtained : it is the so-called **structure factor** S(q), interactions in the system.

→ Different q range are often necessary

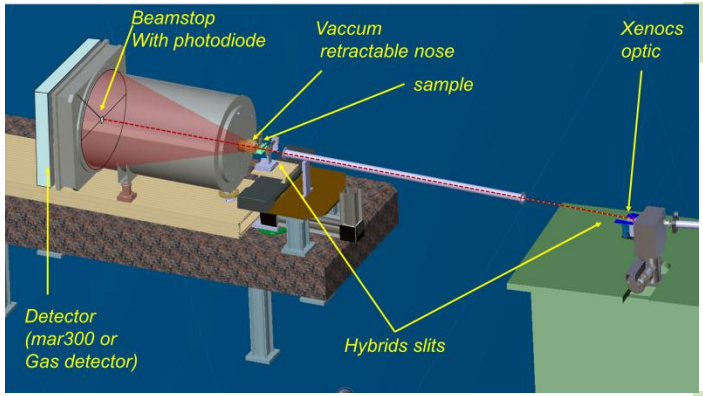
SAXS at LIONS : 3 experimentals setups



USAXS ultra small angles

q range : $2x10^{-4}$ to 10^{-1} A^{-1}

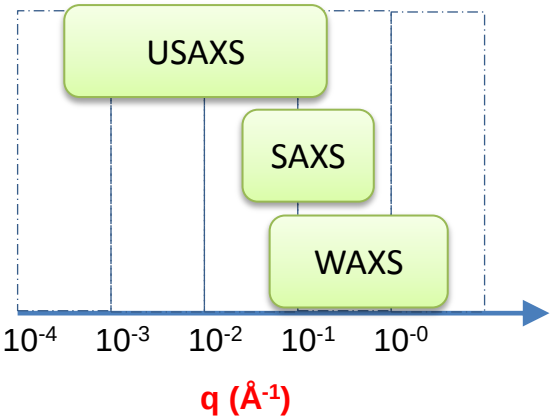
λ=0.154 nm
E=8keV
1D detector

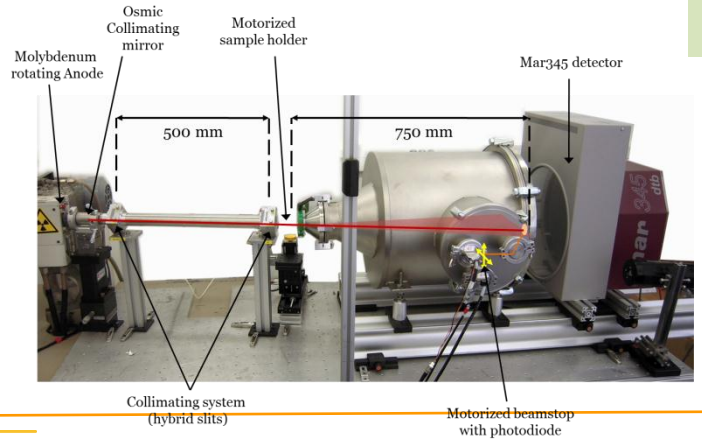


SAXS

q range : $2x10^{-2}$ to $7x10^{-1}$ A^{-1}

λ=0.154 nm
E=8 keV
2D detector





SAXS – WAXS (wide angles)

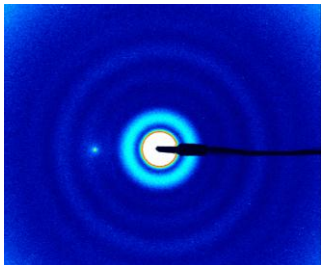
q range : $4x10^{-2}$ to 4 A^{-1}

λ=0.07 nm
E=17 keV
2D detector

SAXS Data treatment : what we have to do

1- for Images : data reduction

Using ImageJ (Java !) a software that manage images (with ROI, LUT)
With geometrical corrections

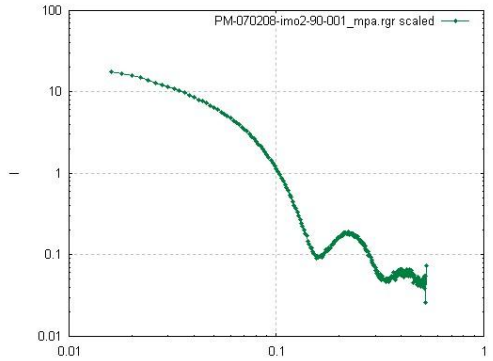


1- for USAXS (1D) : deconvolution (beam is not perfect)

With specific code

2- scaling in absolute intensities (taking in acount experimental parameters)

Very important if we want to compare datas from others experiments (synchrotron)
We can calculate Form factor and Structure Factor

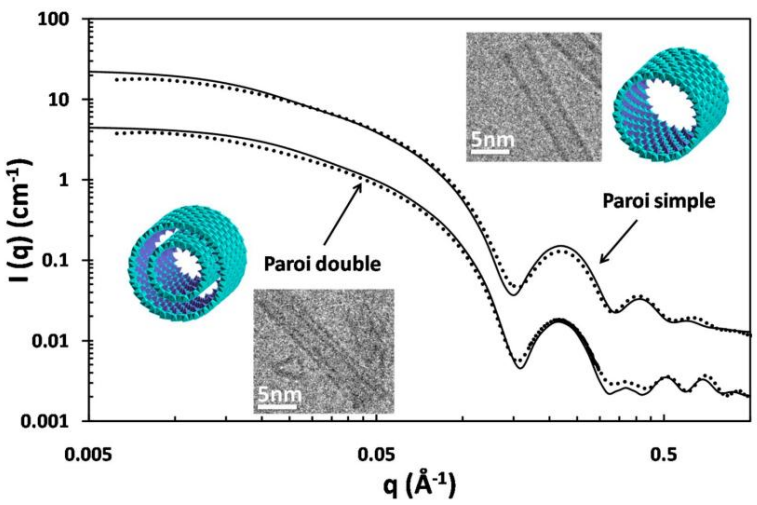


3- merging datas and substract background or solvant (ie water)

Merging datas with different scales (qrange or dq)

4- compare with predefined models

minimization for finding sample parameters
Home made models
or with source code we can check…



5- non automatic data processing for calculating form factor, structure,…

With source code optimized and tested

→ Home made software → PySAXS

Other softwares ?

Other SAXS data treatment softwares :

- Sasfit : for neutron, C language
- SOLEIL : foxtrot (integrated with the hardware)
- Igor routines (not free, code source)
- Matlab routines (not free, code source)
- BioXtas (python with a similar wxPython GUI)

With python, researchers can **validate** and modify the source code

With GuiSAXS, standard users can analyze **easily** datas

What is PySaxs ?

LS (LIONS Saxs) :

A special effort for a compilation of useful functions in Python

- calculating different form factors or structure factors :

```
def F1(q,R):
    """
    This function returns a scattering amplitude of a sphere of radius R for q
    """
    return (3.0*(numpy.sin(q*R)-q*R*numpy.cos(q*R)))/(q*R)**3.0
```

```
def P1(q,R):
    """
    This function returns the form factor of a sphere of radius R for q
    """
    if numpy.min(R)<=0.0:
        sys.stderr.write('can not compute for nul or negative sizes\n')
        return 1.0
    return F1(q,R)*F1(q,R)
```

Gives intensities (q range, parameters)

- For absolute intensities (scaling) processing

→ Can be used by researcher 'own' routines

What is PySaxs ?

Librairies of models:

based on a Class model,
coded by researchers
using combination of Form factors and structures factors (LS)

What is a model ?

I(q) function depending of parameters

list of parameters with description, defaults values
name of authors

Fitting functions based on scipy.optimize (simple with optimize.leastsq or with bounds : optimize.fmin_tnc)

Offering simple usage for fit :

```
from pySAXS.models import MonoSphere
sphere=MonoSphere()
y=array_of_experimentals_datas
res=sphere.fit(y)
```

How ?

1. A class model
2. And a mymodel.py file in the model directory

Available models (july 2011) :

Multilayer Cylinder
Porod
DC- Shells: semi-gaussian distribution
Spray Dried Grain
Imogolite Single Wall Si/Ge
Spheres poly-Gauss analytique
Gaussian
Porod Layer
Porod Curved
Core Shell Particle
Mono Ellipse
Mono Cylinder
Spheres : Semi-Gaussian distribution
Spheres Monodisperse
Spheres poly-Gauss
Imogolite Double Wall Si/Ge

Models

```
class MonoSphere(Model):
    '''

    class monoSphere from LSSca
    by OT 10/06/2009
    '''
    def MonoSphereFunction(self,q,par):
        """
        q array of q (A-1)
        par[0] radius of the sphere (A)
        par[1] scattering length density of sphere (cm-2)
        par[2] scattering length density of outside (cm-2)
        par[3] concentration of sphere (cm-3)
        """
        if len(par)!=4:
            sys.stderr.write("This function requires a list of 4 parameters")
            return -1.
        else:
            return par[3]*(par[1]-par[2])**2.*getV(par[0])*getV(par[0])*1e-48*P1(q,par[0])
            #sys.stderr.write(str(par[0]))
            #return P1(q,par[0])

    '''
    parameters definition
    Model(0,MonoSphere,Qlogspace(1e-4,1.,500.),([250.0,2e11,1e10,1.5e15]),
    ("radius (A)","scattering length density of sphere (cm-2)","scattering length density o
    ,("%f","%1.3e","%1.3e","%1.3e"),(True,True,False,False)),
    from LSSca
    '''
    IntensityFunc=MonoSphereFunction #function
    N=0
    q=Qlogspace(1e-4,1.,500.)       #q range(x scale)
    Arg=[250.0,2e11,1e10,1.5e15]            #list of defaults parameters
    Format=["%f","%1.3e","%1.3e","%1.3e"]       #list of c format
    istofit=[True,True,False,False]     #list of boolean for fitting
    name="Spheres Monodisperse"           #name of the model
    Doc=["radius (A)",\
         "scattering length density of sphere (cm-2)",\
         "scattering length density of outside (cm-2)",\
         "number concentration (cm-3)"] #list of description for parameters
```

What is PySaxs ?

A graphic user interface : GuiSAXS

no satisfaisant interface for data treatment and data manipulation

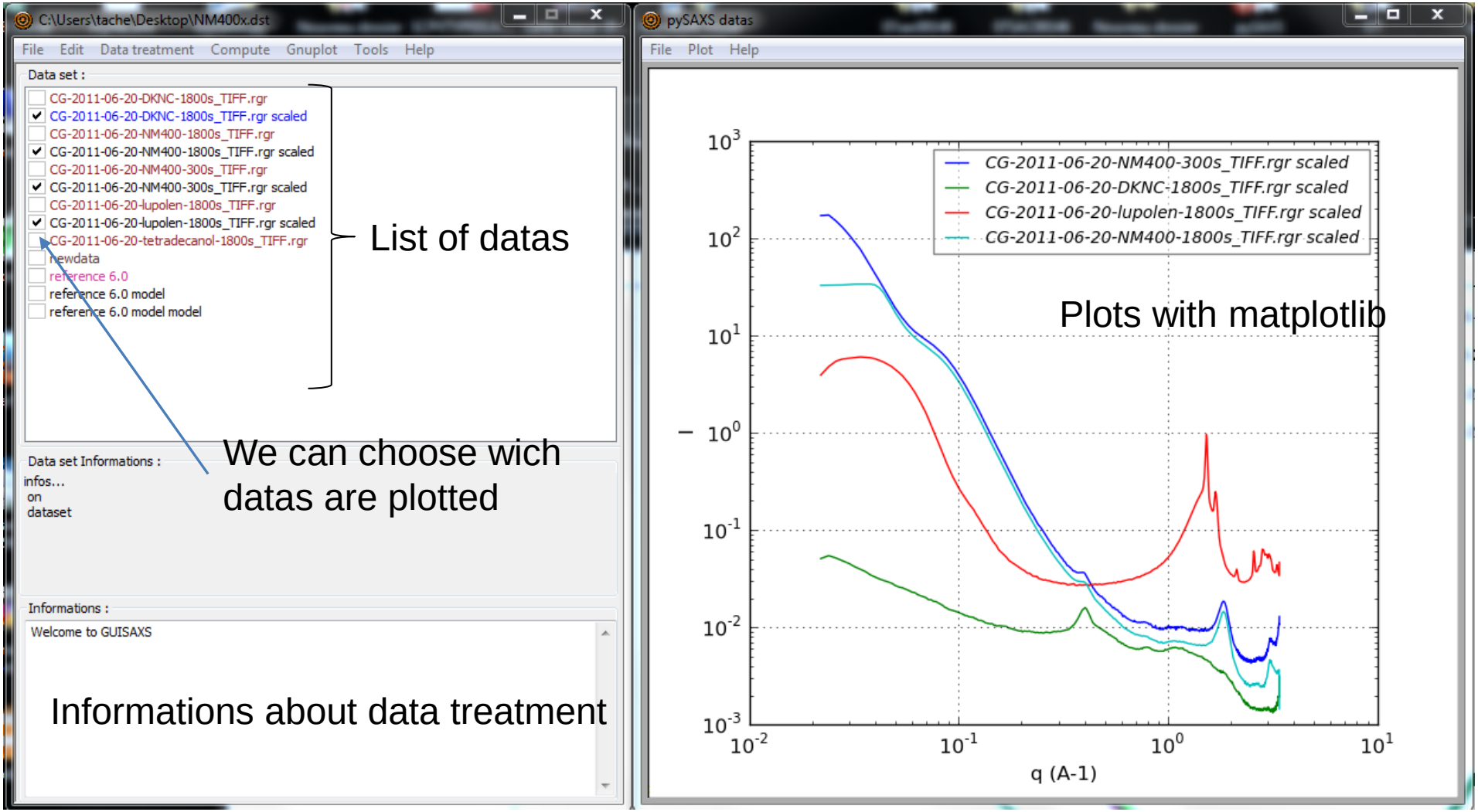
- import data from experiments (text file)
- scaling
- compare
- substraction or manipulation datas with different scales
- correct plotting tool (log scale)
- → gnuplot and matplotlib
- modeling

- informations about data treatment

Using wxPython :

Not a real choice
Works on windows and linux
No IDE : all the code is made by « hand »
→as much as possible : generic dialog boxes

GuiSAXS


C:\Users\tache\Desktop\NM400x.dst
File Edit Data treatment Compute Gnuplot Tools Help
Data set :
CG-2011-06-20-DKNC-1800s_TIFF.rgr
CG-2011-06-20-DKNC-1800s_TIFF.rgr scaled
CG-2011-06-20-NM400-1800s_TIFF.rgr
CG-2011-06-20-NM400-1800s_TIFF.rgr scaled
CG-2011-06-20-NM400-300s_TIFF.rgr
CG-2011-06-20-NM400-300s_TIFF.rgr scaled
CG-2011-06-20-lupolen-1800s_TIFF.rgr
CG-2011-06-20-lupolen-1800s_TIFF.rgr scaled
CG-2011-06-20-tetradecanol-1800s_TIFF.rgr
newdata
reference 6.0
reference 6.0 model
reference 6.0 model model
List of datas
We can choose wich datas are plotted
Data set Informations :
infos...
on
dataset
Informations :
Welcome to GUISAXS
Informations about data treatment
pySAXS datas
File Plot Help
CG-2011-06-20-NM400-300s_TIFF.rgr scaled
CG-2011-06-20-DKNC-1800s_TIFF.rgr scaled
CG-2011-06-20-lupolen-1800s_TIFF.rgr scaled
CG-2011-06-20-NM400-1800s_TIFF.rgr scaled
Plots with matplotlib
I
q (A-1)

GuiSAXS

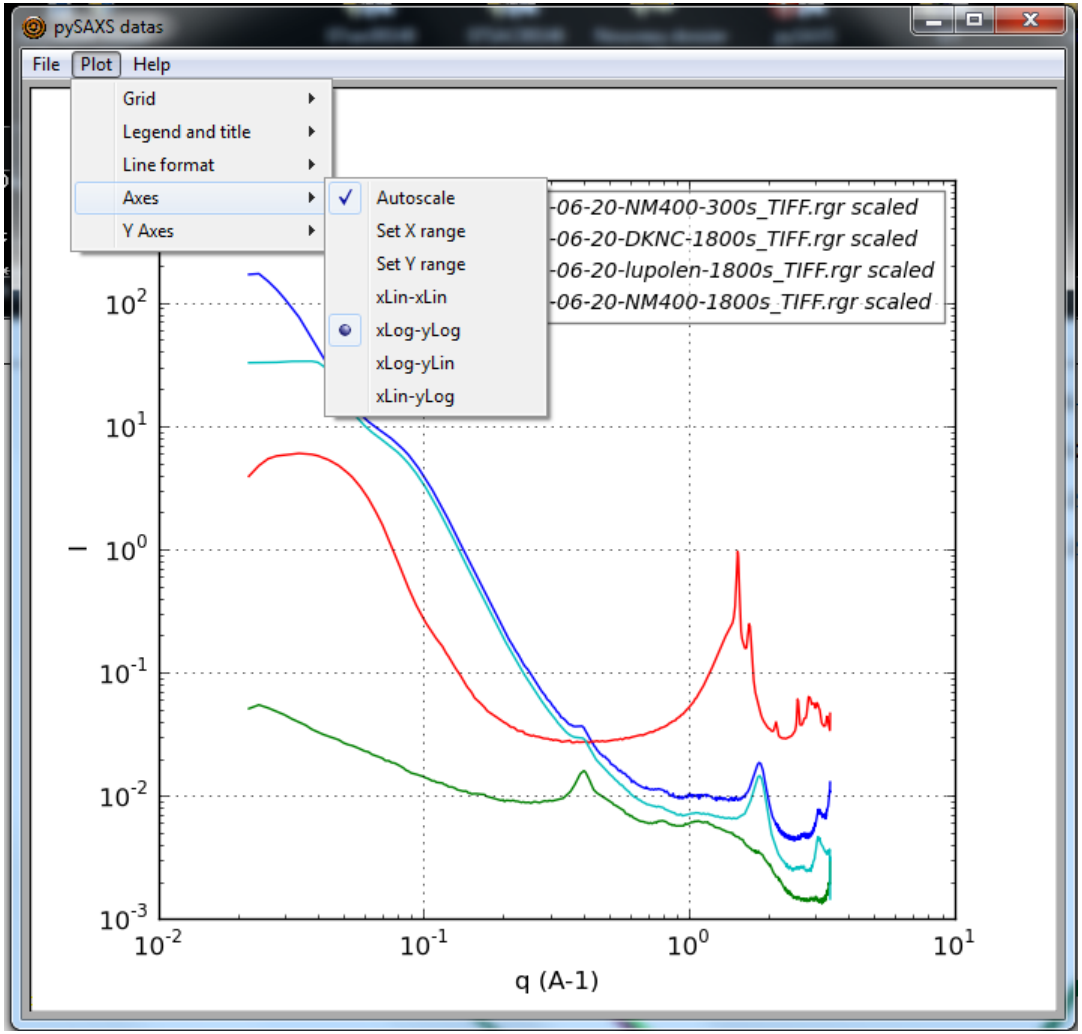
A matplotlib frame with a menu where you can :

- Add a grid
- Change legend and title
- Set the line format
- Set the axes scales
- Save as picture

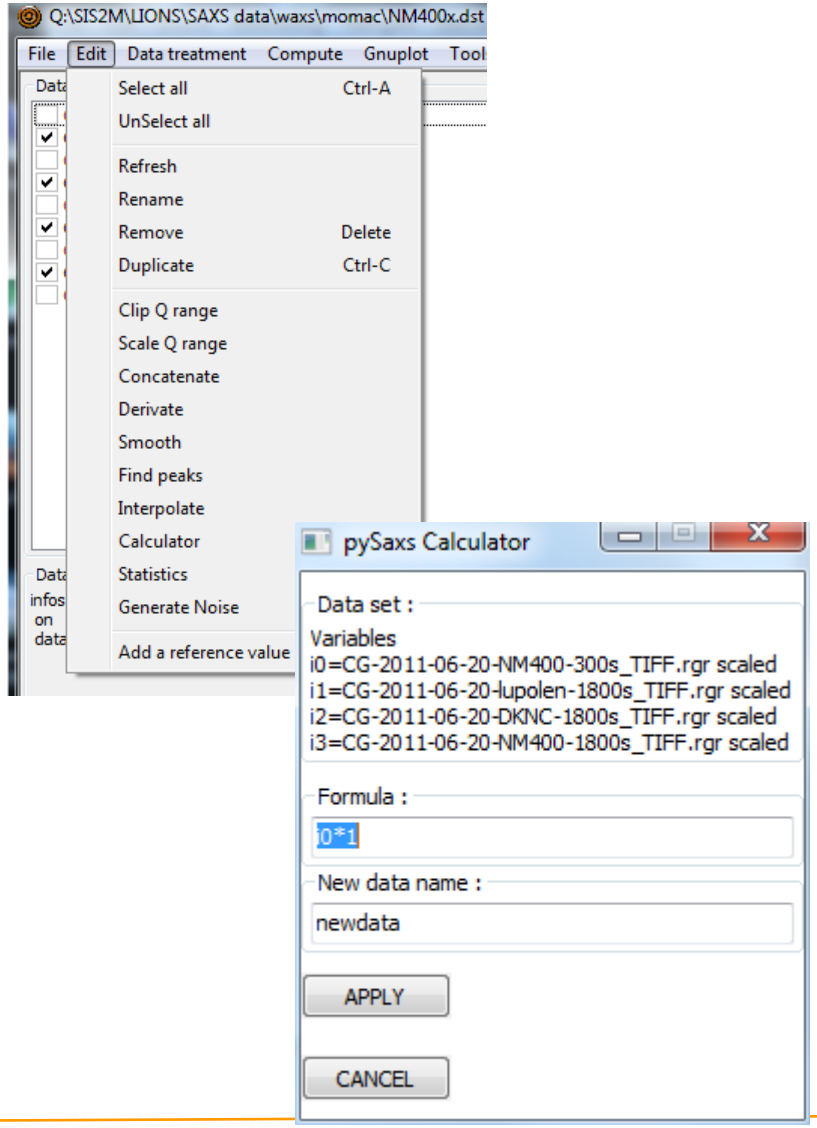
- Colors are automatics,

it is not possible to change them
Can be improved

OR you can use gnuplot windows



GuiSAXS : data manipulation



- Refresh : reload datas from file
- Rename
- Remove datas from list
- Duplicate

- Clip q range
- Scale : change scale (q*10)
- Concatenate
- Derivate
- Smooth datas
- Find peaks
- Interpolate (add points)
- Calculator :
  open a dialog box and let the user specify a formula for data manipulations
- Statistics
- Generate noise on datas
- Add a reference value : to compare with a flat datas

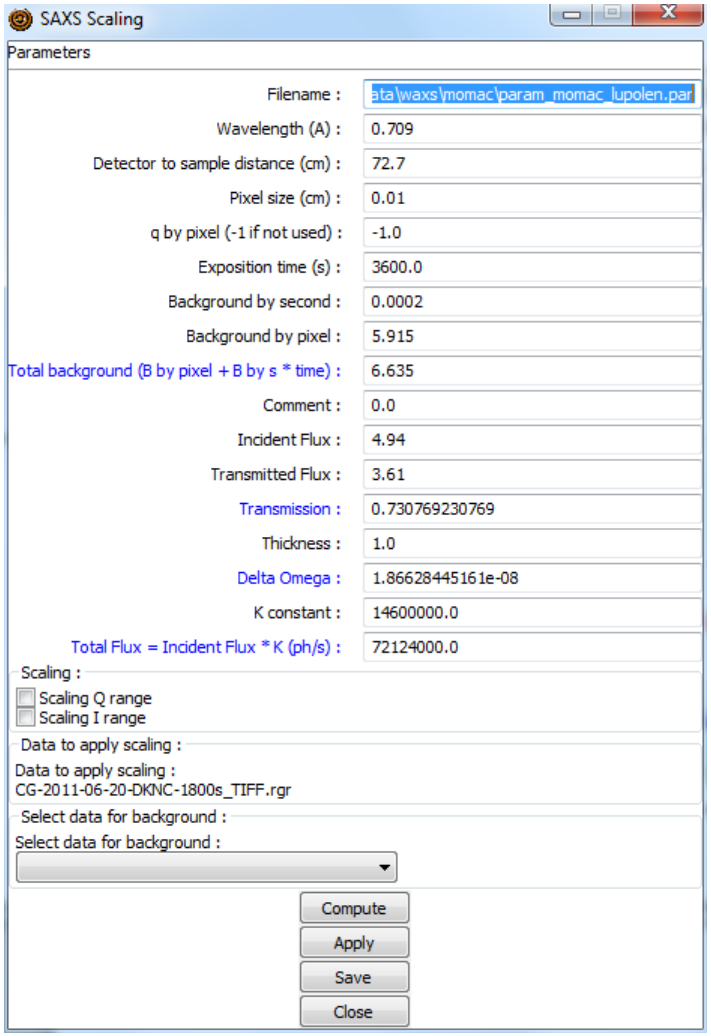
GuiSAXS : data scaling

$$I(q) = \frac{C_{ij}}{\phi_0 . dt} . \frac{1}{\Delta\Omega} . \frac{1}{e}$$

- C_{ij} is the number of counts detected on pixel ij during dt with background substracted
- $\phi_{0.}$ is the transmitted flux (photons/s) by the sample
  $\phi_{0=}\ \phi_{incident}$ .T.K
  T is the transmission of the sample
  K is the detector quantum efficiency
  $K = \frac{\eta_1}{\eta_2}$
  η_1, is the detector quantum efficiency for the counts C_{ij}
  η_2, is the detector quantum efficiency when measuring the incident beam.
- $\Delta\Omega$ is the solid angle covered by one pixel seen from the center of the sample.
  $\Delta\Omega = \frac{p^2}{D^2}$
  p is the pixel size and D the sample to detector distance
- e is the thickness of the sample (cm)

```
Intensity=(n-background)/(time * DeltaOmega * Transmission *
Thickness * Flux * K)
```

→ Absolute intensities are independant from experiment



Data substraction

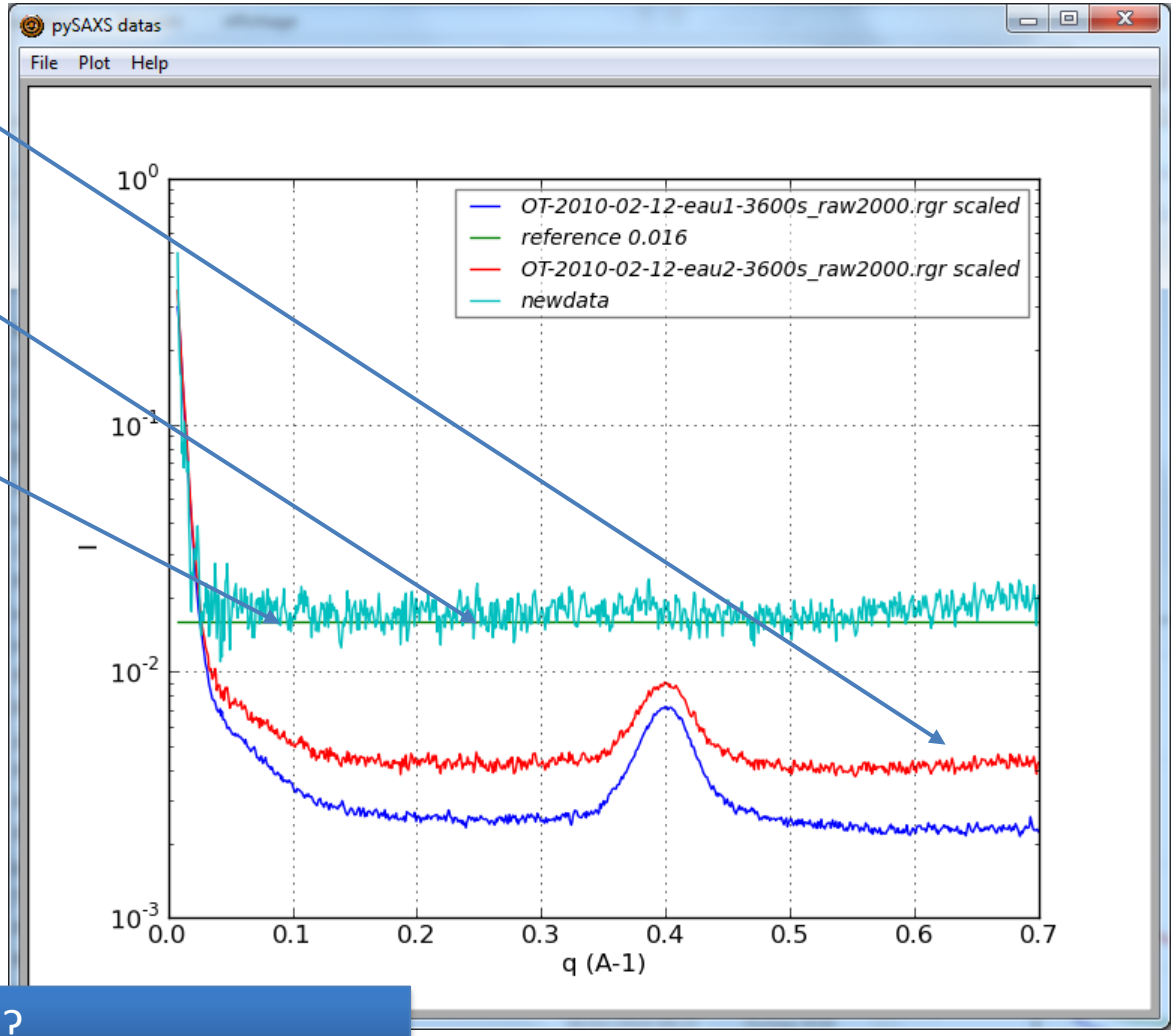
Data for 1mm and 2 mm thickness of water

Substraction (gives 1mm of water)

Reference (calculated value)

Data processing is done by using interpolation of datas

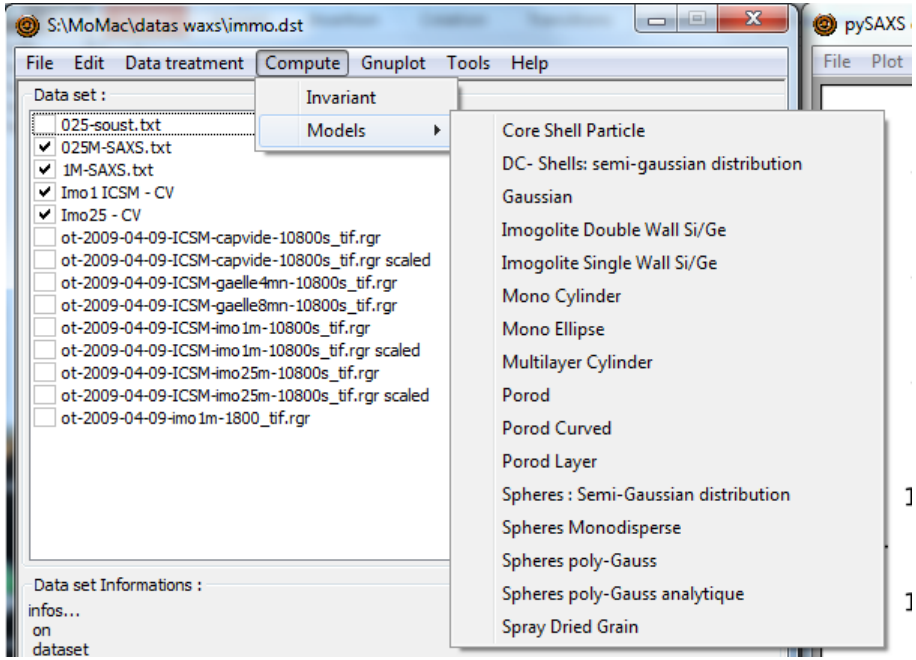
With propagation of measurement's errors



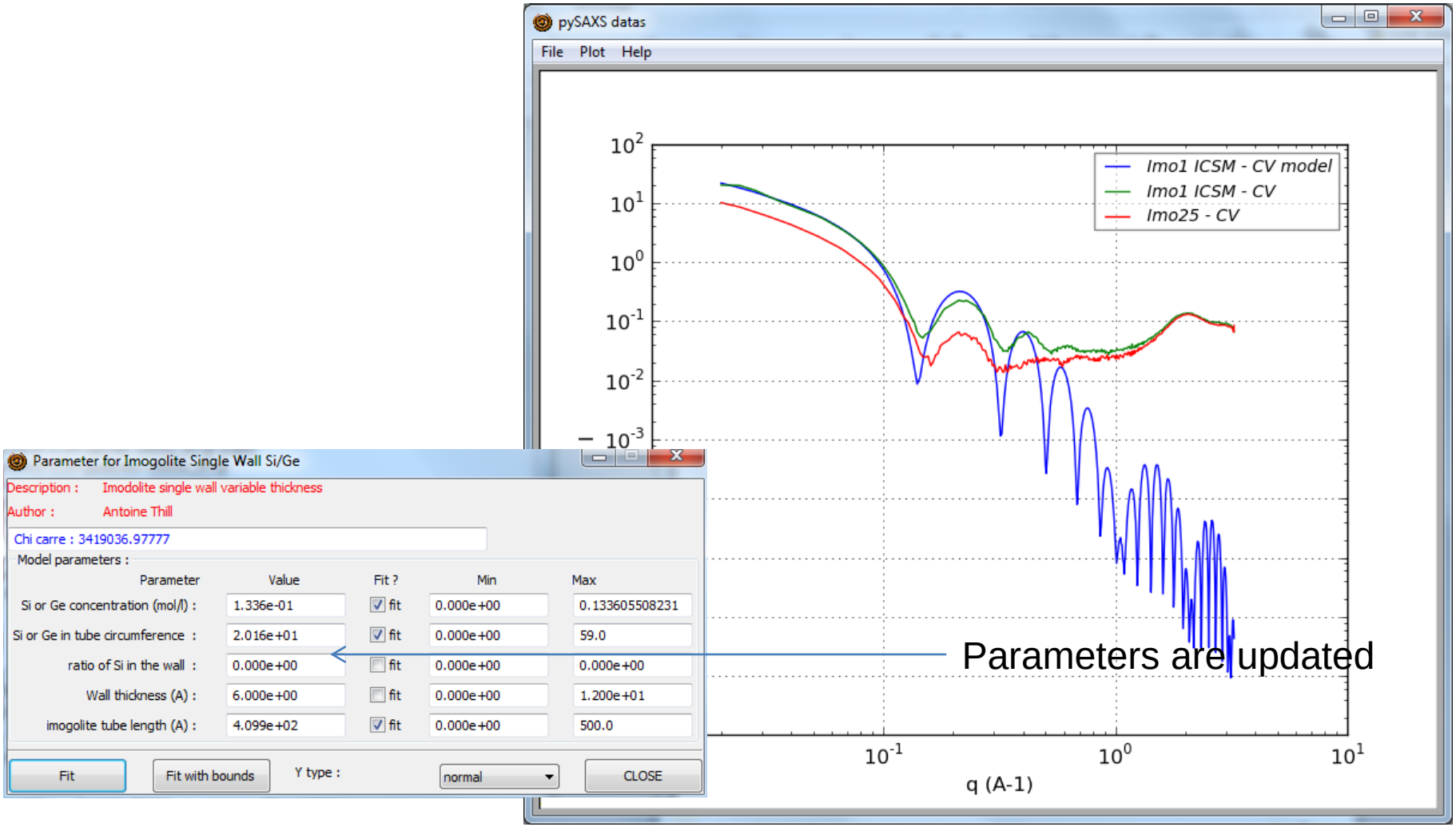
→ How could we do that in excel ?

Fitting with models

Models integrated automatically in the menu



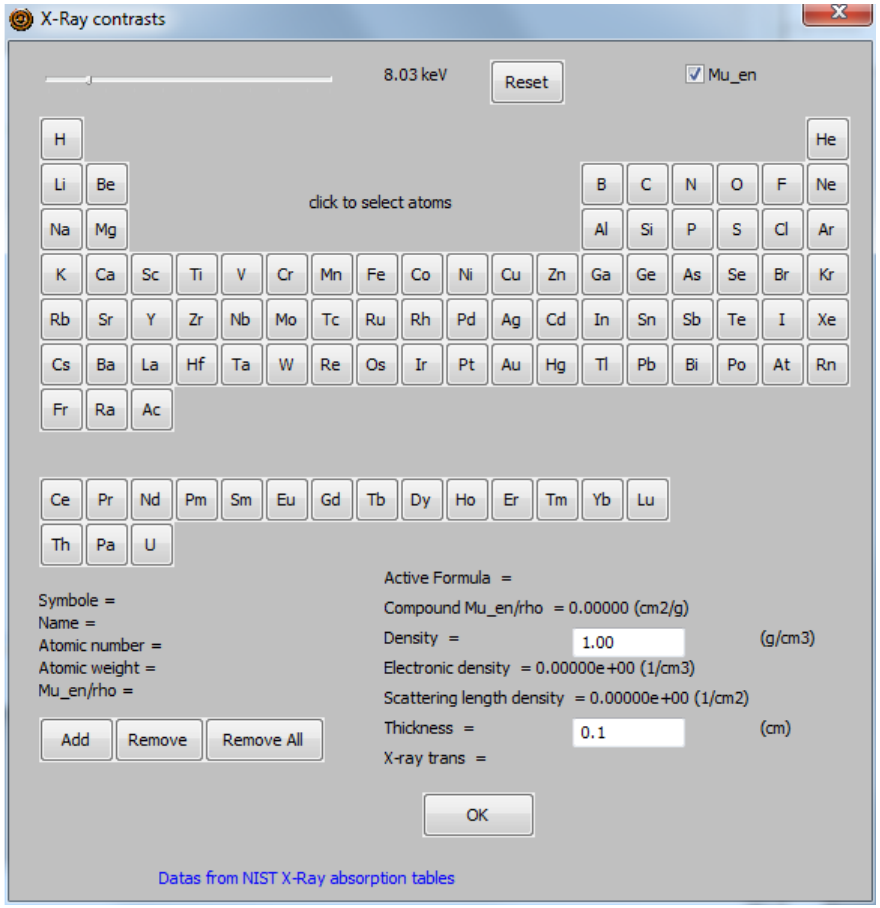
Fitting with models



Other functionalities :

- Datas saved as txt
- Datas saved by group (copy of memory in file)
- **Keeping measurements error bar**

- X-Ray contrasts dialog box : for calcutating transmission of sample depending on composition and x-ray energy



- PySAXS is given to users
- Easy Installation with PythonXY

Improvements ?

- Integration of the complete data treatment process ?