PyXRD

A model for the simulation of 1-dimensional X-ray diffraction patterns of disordered layered minerals

Software Manual

Last update: October 15th 2015

Mathijs Dumon
Ghent University
Department of Geology and Soil Science (WE13)
Krijgslaan 281/S8
B-9000 Ghent
Belgium

Table of contents

1 Introduction

PyXRD is a computer model developed in Python for the simulation of 1-dimensional X-ray diffraction patterns for mixed-layer minerals. It has been developed keeping a multi-specimen full profile fitting strategy in mind. It allows for (semi-)quantification of mixed-layer phases by combining several observed XRD patterns and can perform automatic parameter refinements using several aglorithms.

This document provides for a general overview of the theoretical background on which this model is based, an overview of the actual implementation (code-wise) and instructions on how the general user interface (GUI) written in GTK can be used to create and modify models.

For more detailed information we kindly refer to the source documentation and if that is failing, the source code itself.

If any mistakes are discovered in this document please inform me at mathijs.dumon@ugent.be

2 License

3 Installation

3.1 Windows

As of PyXRD v0.6.2 there are two options for windows users: (i) either manually install all the dependencies and PyXRD or (ii) use a custom installer which will install PyXRD and all of its dependencies for you.

Note that when using the bundled installer, DEAP is not installed. You will still need to follow the instructions in section 3.1.3

3.1.1 Bundled installer

Download and run the bundled installer:

https://github.com/mathijs-dumon/PyXRD/releases/download/v0.6.7/PyXRD-0.6.7-win32-bundle.exe

For most of the dependencies this does not require input from your side. However, for the Numpy and Scipy libraries there is currently no easy way to completely automate the installation. As a result, you will have to click 'Next' and/or 'Finish' a few times to complete the installation for these libraries.

Note that this is still an experimental feature. If you encounter problems, please report them by e-mail (mathijs.dumon@ugent.be) and continue by following the instructions for manual installation below.

3.1.2 Manual install

The installation is a bit lengthy because PyXRD depends on a number of third-party python modules. Work has started to create a unified installer, but for now you'll have to install them manually. These are the dependencies (more recent versions should also work, except for python which needs to be version 2.7):

- Python: http://www.python.org/ftp/python/2.7.8/python-2.7.8.msi
- PyGTK: http://ftp.gnome.org/pub/GNOME/binaries/win32/pygtk/2.24/pygtk-all-in-one-2.24.2.win32-py2.7.msi
- Numpy: http://sourceforge.net/projects/numpy/files/NumPy/1.7.0/numpy-1.7.0-win32-superpack-python2.7.exe/download
- Scipy: http://sourceforge.net/projects/scipy/files/scipy/0.14.0/scipy-0.14.0-win32-superpack-python2.7.exe/download
- Matplotlib: https://downloads.sourceforge.net/project/matplotlib/matplotlib/matplotlib-1.2.1/matplotlib-1.2.1.win32-py2.7.exe
- Pyparsing: http://sourceforge.net/projects/pyparsing/files/pyparsing/pyparsing-2.0.3/pyparsing-2.0.3.win32-py2.7.exe/download

- Setuptools: https://bootstrap.pypa.io/ez_setup.py
    1. Download the script somewhere you can find it (e.g. the desktop)
    2. Open a command line as administrator
       (Start button → Search → enter 'cmd.exe' → right-click the command line icon and select 'Run as adiministrator')
    3. Enter the following command (replace the path to ez_setup to where you have downloaded it):
       ```
       C:\Python27\python.exe c:\users\myusername\Desktop\ez_setup.py
       ```
       This assumes you have installed python in C:\Python27 (the default location), if not change the command accordingly.

Finally download and install PyXRD:
https://github.com/mathijs-dumon/PyXRD/releases/download/v0.6.7/PyXRD-0.6.7.win32.exe

3.1.3 Installation of DEAP

You can optionally install DEAP which will provide evolutionary refinement algorithms:

1. Open a command line as administrator
   (Start button → Search → enter 'cmd.exe' → right-click the command line icon and select 'Run as adiministrator')
2. Enter the following command:
   ```
   C:\Python27\Scripts\easy_install.exe deap
   ```

3.2 Linux

Installation on linux should be straightforward, first install Python 2.7 and PyGTK:

- Debian/Ubuntu/...
  ```
  sudo apt-get install python python-gtk2
  ```
- Fedora/Red Hat/... (untested)
  ```
  sudo yum install python python-gtk
  ```
- OpenSuSE (untested)
  ```
  sudo yum install python python-gtk
  ```

Then you can choose to either install the dependencies from your package manager repositories or using pip or easy_install. Usually it is preferable to use the binaries from the package manager as you will not need to install them by hand. The dependencies are:

- Numpy >= 1.7.0
- Scipy >= 0.14.0
- Matplotlib >= 1.2.1
- Pyparsing >= 2.0.4
- Setuptools

And the corresponding commands would be:

- Debian/Ubuntu/...
  ```
  sudo apt-get install python-numpy python-scipy python-matplotlib
  ```
- Fedora/Red Hat/... (untested)
  ```
  sudo yum install python python-gtk python-numpy python-scipy  python-matplotlib
  ```
- OpenSuSE (untested)
  ```
  sudo yum install python python-gtk python-numpy python-scipy  python-matplotlib
  ```

Once this has been completed, open a terminal and enter these commands:

```
sudo easy_install pip
pip install --user 'pyxrd>=0.6.7'
```

This will install everything under your '~/.local' folder. To run PyXRD type in the following:

```
~/.local/bin/PyXRD
```

or make a shortcut to this command.

4 Theoretical background

4.1 General mathematical formalism

The mathematical formalisms on which PyXRD is based is described in a number of articles and books published over the years. Relevant references can be found at the end of this section. Below an overview will be given of the different parts of the general matrix formalism.

The general matrix formalism (described in detail in Drits & Tchoubar (1990)) allows to calculate the diffraction pattern for a single mixed-layer phase A as:

$$I_A(s) \propto \frac{wf_A}{\overline{\rho_A}\cdot\overline{V_A}^2}\cdot\Xi\cdot\sum_{n=1}^{M}\alpha(n)\int_{\varphi}\Re\,Tr\{[F]\cdot[W]\cdot[R]\}\cdot\overline{T_n(U)}\cdot\bar{p}(\varphi)\cdot d\varphi \qquad \text{(eq. box 4.1)}$$

in which:

$I_A(s)$	the intensity diffracted by phase A at reciprocal space value s
wf_a	the weigth fraction of phase A in the mixture
$\overline{\rho_A}$	is the density for an average unit cell in mixed layer phase A composed of G different components,calculated as $\sum_G \frac{M_g\cdot W_g}{V_g}$, in which M_g is the total atomic mass for a unit cell of component g, V_g is the volume of the unit cell for component g and W_g is the relative fraction of components of type g in the mixed-layer phase A
$\overline{V_A}$	is the average unit cell for phase A, calculated as $\sum_G W_g\cdot V_g$
Ξ	is the Lorentz-polarisation factor, including a correction for the imperfect orientation of particles (also known as the powder ring distribution factor), (see chapter 4.6)
$\alpha(n)$	is the fraction of crystallites with a coherent scattering domain size (CSDS) of n layers (see chapter 4.5)
$[F]$	the structure factor matrix (see chapter 4.3);
$[W]$	the weight fraction matrix (see chapter 4.8);
$[R]$	is defined as $[R]=[I]+2\cdot\sum_{m=1}^{M-1}\frac{M-m}{M}\cdot[Q]^m$ in which [I] is the identity matrix, M the number of layers in the considered CSDS and [Q] is the nearest neighbour phase difference matrix (see chapter 4.4)
$\overline{T_n}(U)$	is the mean area of the coherent scatter domain with n layers, which can be approximated by $n\cdot\bar{c}=n\cdot\sum_G c_g\cdot W_g$ in which $\bar{c}$ is the average d-spacing and c_g is the d-spacing of g-th component
$\bar{p}(\varphi)$	is the (normalized) probability of finding a layer deviating φ radians from perfect orientation

In the following sections, the calculation of each of the matrices and functions is explained in more detail.

4.2 Length of the reciprocal vector: s

The length of the reciprocal vector (commonly denoted as $\vec{s}$) can be expressed as:

$$|\vec{s}|=s=\frac{2\cdot\sin\theta}{\lambda} \qquad \textit{(eq. box 4.2)}$$

in which:

θ	the angle of the incident X-ray bundle
λ	the wavelength of the X-ray waves

It relates with Bragg's formula like this:

$$2 \cdot d \cdot \sin(\theta) = n \cdot \lambda$$
$$\frac{2 \cdot \sin(\theta)}{\lambda} = \frac{n}{d} = s \qquad \textit{(eq. box 4.3)}$$

4.3 Calculation of the structure factor matrix F

4.3.1 General

The actual size of the F and Q matrices depends on the Reichweite of the model. Therefore this section is split in two parts: first the setup of the structure factor matrix for R0 and R1 models is explained, and then it is explained how that matrix can be scaled to match the higher Reichweite models. The basis for this elaboration is based for the larger part on principles and examples found in Drits & Tchoubar (1990).

i. Layout for R0 and R1 models

The structure factor matrix F then has the following definition:

$$F = \begin{bmatrix} F_1 F_1^* & F_2 F_1^* & \dots & F_G F_1^* \\ F_1 F_2^* & F_2 F_2^* & \dots & F_G F_2^* \\ \dots & \dots & \dots & \dots \\ F_1 F_G^* & F_2 F_G^* & \dots & F_G F_G^* \end{bmatrix} \qquad \textit{(eq. box 4.4)}$$

in which:

F_g	the structure factor for the g-th component (see 4.3.2)
F_g^*	its complex conjugate

The complete structure factor matrix F can be constructed from simpler matrices. First we create a 1D matrix F_a containing the structure factors for each component:

$$F_a = \begin{bmatrix} F_1 & F_2 & \dots & F_G \end{bmatrix} \qquad \textit{(eq. box 4.5)}$$

After this we create another 1D matrix F_b which is the transpose-conjugated form of matrix Fa:

$$F_b = F_a^{*T} = \begin{bmatrix} F_1^* \\ F_2^* \\ \dots \\ F_G^* \end{bmatrix} \qquad \textit{(eq. box 4.6)}$$

The structure factor matrix F can then be constructed by multiplying matrices Fb with Fa:

$$F = F_b \cdot F_a = F_a^{*T} \cdot F_a$$

$$F = \begin{bmatrix} F_1^* \\ F_2^* \\ \cdots \\ F_G^* \end{bmatrix} \cdot \begin{bmatrix} F_1 & F_2 & \dots & F_G \end{bmatrix}$$

$$F = \begin{bmatrix} F_1 F_1^* & F_2 F_1^* & \dots & F_G F_1^* \\ F_1 F_2^* & F_2 F_2^* & \dots & F_G F_2^* \\ \cdots & \dots & \dots & \dots \\ F_1 F_G^* & F_2 F_G^* & \dots & F_G F_G^* \end{bmatrix}$$

(eq. box 4.7)

ii. Scaling for higher Reichweite models

For models with R > 1 the matrix needs to be of size G^R. To accomplish this, each pair of structure factors $F_i \cdot F_j^*$ is replaced with a sub-matrix of size G^{R-1} of the form:

$$F_{ij} = \begin{bmatrix} F_i F_j^* & F_i F_j^* & \dots & F_i F_j^* \\ F_i F_j^* & F_i F_j^* & \dots & F_i F_j^* \\ \dots & \dots & \dots & \dots \\ F_i F_j^* & F_i F_j^* & \dots & F_i F_j^* \end{bmatrix}$$

(eq. box 4.8)

With other words, each pair of structure factors is replaced with a 'sub-matrix' of size G^{R-1} in which each element is a duplicate of the replaced pair of structure factors.

4.3.2 Structure factor F_n for a component

The structure factor characterizing the X-ray scattering by an infinite, three-dimensional atomic motif can, in general, be written as:

$$F_{hkl}(s) = \sum_m f_m(s) \cdot \exp(2 \cdot \pi \cdot i \cdot (\frac{h \cdot x_m}{a} + \frac{k \cdot y_m}{b} + \frac{l \cdot z_m}{c}))$$

(eq. box 4.9)

in which:

m	the number of atoms in the motif of the component
f_m(s)	the scattering factor for the m-th atom
x_m, y_m, z_m	the position of the m-th atom along the X, Y and Z axes of the motif (in nm)
h, k, l	the miller indices of the reflection being calculated
a, b, c	unit cell dimensions along the X, Y and Z axes of the motif (in nm)
s	the length of the reciprocal vector (see chapter 4.2)
i	the unreal number (= $\sqrt{(-1)}$)

However, since we are only interested in 00l-reflections, the h and k terms can be dropped and the structure factor for this 1-dimensional atomic motif can be written as:

$$F_n = \sum_m f_m(s)\cdot\exp\left(2\cdot\pi\cdot i\cdot\frac{l\cdot z_m}{c}\right) \tag{eq. box 4.10}$$

According to Bragg's law, we can write (also see chapter 4.2):

$$\frac{2\cdot\sin(\theta)}{\lambda} = \frac{n}{c} = s \tag{eq. box 4.11}$$

Combining the above two relations and setting n=1 and l=1, we can write:

$$F_n = \sum_m f_m(s)\cdot\exp(2\cdot\pi\cdot i\cdot z_m\cdot s) \tag{eq. box 4.12}$$

in which:

m	the number of atoms in the motif of the component
f_m(s)	the scattering factor for the m-th atom (see chapter 4.3.3)
z_m	the position of the m-th atom along the Z axis of the motif
s	the length of the reciprocal vector (see chapter 4.2)
i	the unreal number (= $\sqrt{(-1)}$)

The complex conjugate of the structure factor is (see chapter 4.3):

$$F_n^* = \sum_m f_m(s)\cdot\exp(-2\cdot\pi\cdot i\cdot z_m\cdot s) \tag{eq. box 4.13}$$

This exponential relation can be transformed using Euler's formula into the sine and cosine form:

$$F_n = \sum_m f_m(s)\cdot\left[\cos(2\cdot\pi\cdot i\cdot z_m\cdot s) + i\cdot\sin(2\cdot\pi\cdot i\cdot z_m\cdot s)\right] \tag{eq. box 4.14}$$

The complex conjugate of this formula can then be written as:

$$F_n^* = \sum_m f_m(s)\cdot\left[\cos(2\cdot\pi\cdot i\cdot z_m\cdot s) - i\cdot\sin(2\cdot\pi\cdot i\cdot z_m\cdot s)\right] \tag{eq. box 4.15}$$

4.3.3 Atomic scattering factor for a single atom: f_m

The atomic scattering factor for a single atom is calculated using the Cromer-Mann coëfficiënts as published in Waasmaier and Kirfel (1995). The Debye constants are set to:

0	for neutral atoms
2	for anions
1.5	for cations

The atomic scattering factor is calculated as follows:

$$f_m(s) = P_m\cdot\left[c + \sum_{i=1}^{5}\left(a_i\cdot\exp\left(-b_i\cdot\frac{s^2}{2\cdot 10}\right)\right)\right] * \exp(-B_m * s^2) \tag{eq. box 4.16}$$

in which:

P_m	the number of atoms per unit cell at this z location (can be > 1 due to the projection on the Z-axis)
c	constant of the exponential approximation of the scattering factor
a_i	the i-th a factor of the exponential approximation of the scattering factor
b_i	the i-th b factor of the exponential approximation of the scattering factor
s	the length of the reciprocal vector (see chapter 4.2)
B_m	the Debye constant for the m-th atom

The factor 10 in the exponential part of the summation is there to convert the units of the reciprocal vector s from nanometer to Ångstrom (factor 10).

4.4 Calculation of the phase factor matrix Q

4.4.1 General

As explained in section 4.3.1, the actual size of the F and Q matrices depends on the Reichweite of the model. This section is also split in two parts: first the setup of the phase factor matrix for R0 and R1 models is explained, and then it is explained how that matrix can be scaled to match the higher Reichweite models.

i. Layout for R0 and R1 models

The phase factor matrix Q is the Hadamard product (element-wise product, not the regular matrix multiplication) of two other matrixes:

$$Q=[\varphi]\circ P \qquad \textit{(eq. box 4.17)}$$

of which the first term can be defined as follows:

$$[\varphi]=\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1n} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_{n1} & \varphi_{n2} & \cdots & \varphi_{nn} \end{bmatrix} \qquad \textit{(eq. box 4.18)}$$

in which:

φij	the phase difference between the i-th and j-th component in the mixed-layer phase

The matrix P contains the probability parameters. Its calculation is detailed in chapter 4.8 for different values of R and will not be discussed in further detail here.

ii. Scaling for higher Reichweite models

For models with R > 1 the $[\varphi]$ matrix needs to be of size G^R. To accomplish this, each phase difference $[\varphi_{ij}]$ is replaced with a sub-matrix of size G^{R-1} of the form:

$$[\varphi_{ij}]=\begin{bmatrix} \varphi_{ij} & \varphi_{ij} & \cdots & \varphi_{ij} \\ \varphi_{ij} & \varphi_{ij} & \cdots & \varphi_{ij} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_{ij} & \varphi_{ij} & \cdots & \varphi_{ij} \end{bmatrix} \qquad \textit{(eq. box 4.19)}$$

With other words, each phase difference is replaced with a sub-matrix of size G^{R-1} in which each element is a duplicate of the replaced phase difference.

4.4.2 Phase difference between the i-th and j-th components φ_{ij}

The phase difference depends on the distance between the components. If we define the i-th component as the one preceeding the j-th component, the phase difference depends on the basal spacing of that i-th component (and not of the j-th component):

$$\varphi=e^{2\cdot\pi\cdot s\cdot d_{001}\cdot i}\cdot e^{-2\cdot(\pi\cdot s\cdot\delta_{001})^2}=e^{2\cdot\pi\cdot s\cdot(d_{001}\cdot i-\pi\cdot s\cdot\delta_{001}^2)} \qquad \textit{(eq. box 4.20)}$$

in which:

s	the length of the reciprocal vector (see chapter 4.2)
i	the unreal number (= $\sqrt{(-1)}$)
d_{001}	the basal spacing of the i-th component (nm)
δ_{001}	the variation in the basal spacing of the i-th component (nm)

In Plançon (2002) an equation can be found on how to insert variable d-spacings. A similar formulation can also be found in Drits & Tchoubar (1990, page 89), albeit with different constants. The latter has been implemented in PyXRD, as it matched with the output from Sybilla. It assumes a Gaussian distribution of the d-spacing around the default d spacing.

$$F'_n=\sum_m f_m(s)\cdot e^{2\cdot\pi\cdot i\cdot z_m\cdot s}\cdot e^{-2\cdot\pi^2\cdot\delta_{001}^2\cdot s^2} \qquad \text{(eq. box 4.21)}$$

in which:

m	the number of atoms in the motif of the component
$\boldsymbol{f_m(s)}$	the scattering factor for the m-th atom (see chapter 4.3.3)
$\mathbf{z_m}$	the position of the m-th atom along the Z axis of the motif
s	the length of the reciprocal vector (see chapter 4.2)
δ_{001}^2	the variable d-spacing standard deviation (assuming a Gaussian distribution)
i	the unreal number (= $\sqrt{(-1)}$)

4.5 Calculation of the coherent scattering domain size (CSDS) distribution function α

Several function types have been proposed in the past. Their definitions are detailed below. Currently in the model only a generic log-normal distribution and the log-normal distribution as proposed in Drits et al. (1997) are implemented.

In general, the arithmetic mean CSDS can be calculated from whatever CSDS distribution function is used by:

$$\bar{N}=\sum_{n=1}^{N_{max}} n\cdot\alpha(n) \qquad \textit{(eq. box 4.22)}$$

4.5.1 Ergun model (not implemented)

One of the first CSDS distributions proposed is that of Ergun (1970):

$$\alpha(n)=\mathit{exp}\left(\frac{2-N}{\delta}\right) \qquad \textit{(eq. box 4.23)}$$

in which:

n	the CSDS value of interest
δ	the mean defect-free number of layers

However, this simple model is not used often anymore and has been replaced by a log-normal CSDS distribution, as detailed in Drits et al. (1997).

4.5.2 Log-normal models

The log-normal models assume a log-normal distribution of CSDS values. The basic definition is:

$$\alpha(n)=\sqrt{\frac{2\cdot\pi}{B^2\cdot n^2}}\cdot\mathit{exp}\left(\frac{-(\mathit{log}(n)-(A))^2}{2\cdot B^2}\right) \qquad \textit{(eq. box 4.24)}$$

in which:

n	the CSDS value of interest
A	the mean of the probability density function, defined as: $A=a_1\cdot\mathit{log}(\bar{N})+a_2$ in which a_1 and a_2 are empirical constants and N is the average CSDS.
B^2	the variance of the probability density function, defined as: $B^2=b_1\cdot\mathit{log}(\bar{N})+b_2$ in which b_1 and b_2 are empirical constants and N is the average CSDS.

This model is implemented in PyXRD, together with a model which has pre-set values for the a_1, a_2, b_1 and b_2 parameters (according to Drits et al. (1997)):

a_1 = 0.9485
a_2 = 0.0170
b_1 = 0.1032
b_2 = 0.0034

4.6 Preferred orientation

A correction for preferred orientation of phases should be applied. The basis for these corrections was laid by Reynolds (1986) and the importance of these corrections has recently been reiterated upon by (Dohrmann et al., 2009). The effect of preferred orientation (calculated as ψ) is usually grouped with the Lorentz-Polarisation factor into Ξ:

$$\begin{aligned} S &= \sqrt{\frac{S_1^2+S_2^2}{4}} \\ Q &= \frac{S}{\sqrt{8}\cdot \sin(\theta)\cdot\sigma^*} \\ \psi &= \operatorname{erf}\frac{(Q)\cdot\sqrt{2\cdot\pi}}{2\cdot\sigma^*\cdot S} - 2\cdot \sin(\theta)\cdot\left(\frac{1-e^{-Q^2}}{S^2}\right) \\ \Xi &= \frac{1+\cos^2(2\theta)}{\sin(2\theta)}\cdot\psi \end{aligned} \qquad \textit{(eq. box 4.25)}$$

We also need to describe the distribution of the orientation of the particles in our sample. If our sample is well-oriented, we can assume a Gaussian distribution with a standard deviation σ^* (Dohrmann et al., 2009) so that the distribution fuction $\bar{p}(\varphi)$ becomes:

$$\bar{p}(\varphi) = \frac{1}{\sigma^*\cdot\sqrt{2\cdot\pi}}\cdot e^{\frac{-\varphi^2}{2\cdot(\sigma^*)^2}} \qquad \textit{(eq. box 4.26)}$$

In the general equation, this function needs to be integrated over the entire domain of φ leading to:

$$\int_{\varphi_{min}}^{\varphi_{max}} \bar{p}(\varphi)\,d\varphi = \frac{1}{2}\cdot\left[1+\operatorname{erf}\left(\frac{\varphi_{max}}{\sigma^*\cdot\sqrt{2}}\right)-\operatorname{erf}\left(\frac{\varphi_{min}}{\sigma^*\cdot\sqrt{2}}\right)\right] \qquad \textit{(eq. box 4.27)}$$

Since for an oriented sample $\varphi_{min}=0$ and $\varphi_{max}\gg\sigma^*$, we can approximate this:

$$\int_{\varphi_{min}}^{\varphi_{max}} \bar{p}(\varphi)\,d\varphi \simeq \frac{1}{2}\cdot[1+1-0]=1 \qquad \textit{(eq. box 4.28)}$$

As a result, for an oriented sample, equation 4.1 becomes:

$$I_A(s) \propto \frac{wf_A}{\overline{\rho_A}\cdot\overline{V_A}^2}\cdot\Xi\cdot\sum_{n=1}^{M}\alpha(n)\cdot\Re\, Tr\{[F]\cdot[W]\cdot[R]\}\cdot\overline{T_n(U)} \qquad \textit{(eq. box 4.29)}$$

4.7 Optimization of the formalism

The equation 4.29 can be further optimized to allow for a more efficient calculation process.

If we drop in the relations given for $\overline{T}_n(U)$ and $[R]$ detailed in chapter 4.1, we arrive at:

$$I_A(s) \propto \frac{wf_A}{\overline{\rho_A} \cdot \overline{V_A}^2} \cdot \Xi \cdot \sum_{n=1}^{M} \alpha(n) \cdot n \cdot \bar{c} \cdot \Re\, Tr \left\{ [F] \cdot [W] \cdot \left[[I] + 2 \cdot \sum_{m=1}^{n-1} \frac{n-m}{n} \cdot [Q]^m \right] \right\} \qquad \textit{(eq. box 4.30)}$$

Since $\alpha(n)$ and n are real numbers we can bring them inside the $\Re\, Tr$ operators, and since $\bar{c}$ is independent of n we can take it out of the outermost summation, leading to:

$$I_A(s) \propto \frac{\bar{c} \cdot wf_A}{\overline{\rho_A} \cdot \overline{V_A}^2} \cdot \Xi \cdot \sum_{n=1}^{M} \Re\, Tr \left\{ [F] \cdot [W] \cdot \alpha(n) \cdot n \cdot \bar{c} \cdot \left[[I] + 2 \cdot \sum_{m=1}^{n-1} \frac{n-m}{n} \cdot [Q]^m \right] \right\} \qquad \textit{(eq. box 4.31)}$$

In addition, since [F] and [W] are independent of n, we can take them out of the outer summation (together with the $\Re\, Tr$ operators), and re-write this as:

$$I_A(s) \propto \frac{\bar{c} \cdot wf_A}{\overline{\rho_A} \cdot \overline{V_A}^2} \cdot \Xi \cdot \Re\, Tr \left\{ [F] \cdot [W] \cdot \sum_{n=1}^{M} \alpha(n) \cdot n \cdot \left[[I] + 2 \cdot \sum_{m=1}^{n-1} \frac{n-m}{n} \cdot [Q]^m \right] \right\} \qquad \textit{(eq. box 4.32)}$$

At this point we have already reduced calculating $[F] \cdot [W]$ M times to calculating it only once. But we can still improve this a little bit.

We can distribute the $\alpha(n) \cdot n$ term over the sum of the identity matrix and the inner summation, which cancels the division by n in there. If we also split this summation of a sum in the sum of two summations, we arrive at:

$$I_A(s) \propto \frac{\bar{c} \cdot wf_A}{\overline{\rho_A} \cdot \overline{V_A}^2} \cdot \Xi \cdot \Re\, Tr \left\{ [F] \cdot [W] \cdot \left[\sum_{n=1}^{M} \alpha(n) \cdot n \cdot [I] + 2 \cdot \sum_{n=1}^{M} \sum_{m=1}^{M-1} \alpha(n) \cdot (n-m) \cdot [Q]^m \right] \right\}$$

(eq. box 4.33)

Since the arithmetic mean for the CSDS is defined as $\overline{M} = \sum_{n=1}^{M} \alpha(n) \cdot n$ and $[I]$ is independent of n, we can change this to:

$$I_A(s) \propto \frac{\bar{c} \cdot wf_A}{\overline{\rho_A} \cdot \overline{V_A}^2} \cdot \Xi \cdot \Re\, Tr \left\{ [F] \cdot [W] \cdot \left[\overline{M} \cdot [I] + \underbrace{2 \cdot \sum_{n=1}^{M} \sum_{m=1}^{M-1} \alpha(n) \cdot (n-m) \cdot [Q]^m}_{[S]} \right] \right\} \qquad \textit{(eq. box 4.34)}$$

The double summation term, indicated in equation 4.34 as [S], can now be considered separately. When we write this summation in full, we arrive at a sum of M – 1 summations, of which the first summation will have one term, and the last will have M-1 terms, or:

$$\begin{aligned}[S] &= \sum_{n=1}^{M}\sum_{m=1}^{M-1}\alpha(n)\cdot(n-m)\cdot[Q]^{m} = \\ &\alpha(2)\cdot\left[(2-1)\cdot[Q]^{1}\right] \\ &+\alpha(3)\cdot\left[(3-1)\cdot[Q]^{1}+(3-2)\cdot[Q]^{2}\right] \\ &+\ldots \\ &+\alpha\underbrace{(M)\cdot\left[(M-1)\cdot[Q]^{1}+(M-2)\cdot[Q]^{2}+\ldots+(M-(M-1))\cdot[Q]^{M-1}\right]}_{(M-1)\text{ terms}}\end{aligned} \qquad \textit{(eq. box 4.35)}$$

Since these terms have a lot of Q^{n} matrices in common, summing them in this way is rather time consuming (especially when Q has a high rank). This can be solved by re-grouping these common $[Q]^{n}$ terms as follows:

$$\begin{aligned}[S] &= \sum_{n=1}^{M}\sum_{m=1}^{M-1}\alpha(n)\cdot(n-m)\cdot[Q]^{m} = \\ &\underbrace{\left[\alpha(2)\cdot(2-1)+\alpha(3)\cdot(3-1)+\ldots+\alpha(M)\cdot(M-1)\right]\cdot[Q]^{1}}_{(M-1)\text{ terms}} \\ &+\underbrace{\left[\alpha(3)\cdot(3-2)+\alpha(4)\cdot(4-2)+\ldots+\alpha(M)\cdot(M-2)\right]\cdot[Q]^{2}}_{(M-2)\text{ terms}} \\ &+\ldots \\ &+\underbrace{\left[\alpha(M)\cdot(M-(M-1))\right]\cdot[Q]^{M-1}}_{1\text{ term}}\end{aligned} \qquad \textit{(eq. box 4.36)}$$

Which can be written as a summation again like this:

$$[S] = \sum_{n=1}^{M}\sum_{m=1}^{M-1}\alpha(n)\cdot(n-m)\cdot[Q]^{m} = \sum_{n=1}^{M}[Q]^{n}\cdot\sum_{m=n+1}^{M}\alpha(m)\cdot(m-n) \qquad \textit{(eq. box 4.37)}$$

Now each $[Q]^{n}$ matrix is multiplied only once with a single factor for each value of n, which is a huge performance gain compared to multiplying each $[Q]^{m}$ matrix (M-1) times.

4.8 Probability models

4.8.1 Introduction

The probability models are responsible for calculating the W and P matrices (see previous sections) containing the relative weight fractions and probability parameters respectively. The probability parameters are derived using Markovian statistics. The rank of the matrices depends on the number of components G in the mixed-layer and the Reichweite R.

The Reichweite or Reach is an important concept. The value for R denotes what number of previous components (in a stack of components) still influence the probability determining the type of the following component. With other words, for:

R=0;	the type of the next component does not depend on the previous components,
R=1;	the type of the next component depends on the type of the previous component,
R=2;	the type of the next component depends on the types of the previous two components,
etc.	

There are a number of general relations between the weight fractions W and probabilities P detailed below. They are valid regardless of the value of R or G. These are detailed below. For a more in-depth explanation we refer to Drits & Tchoubar (1990). For stacks composed of G types of layers, we can write:

$$\begin{array}{lll} W_i = N_i / N_{max} & \text{where } i & \in \{1, 2, \ldots, G\} \\ W_{ij} = N_{ij} / (N_{max} - 1) & \text{where } i, j & \in \{1, 2, \ldots, G\} \\ W_{ijk} = N_{ijk} / (N_{max} - 2) & \text{where } i, j, k & \in \{1, 2, \ldots, G\} \\ \quad \vdots & & \\ \quad \text{etc}. & \ldots & \ldots \end{array} \quad \text{(eq. box 4.38)}$$

with:

$$\begin{array}{lll} W_{ij} = W_i \cdot P_{ij} & W_{ijk} = W_{ij} \cdot P_{ijk} & \ldots \\ \sum_i^G W_i = 1 & \sum_i^G \sum_j^G W_{ij} = 1 & \ldots \\ \sum_j^G P_{ij} = 1 & \sum_k^G P_{ijk} = 1 & \ldots \end{array} \quad \text{(eq. box 4.39)}$$

In the following sections a description is made how to setup the matrices used in the calculations for different combinations of R and G. An important step in this process is the selection of independent parameters. A modeller is free to choose these, however for PyXRD it was chosen to make the definition of the independent parameters as much as possible identical to those used in the Sybilla® model created by Chevron ETC, to facilitate (future) exchange of models and comparison of results.

i. W and P matrix layouts

The general expressions for the W and P matrix for different combinations of R and G can be generalized. To illustrate this, the matrices for an R0 or R1 model and for an R2 with G components model are shown below.

The expressions for the W and P matrix for R0 and R1 models are:

$$W=\begin{bmatrix} W_1 & 0 & \dots & 0 \\ 0 & W_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & W_G \end{bmatrix} \qquad \textit{(eq. box 4.40)}$$

$$P=\begin{bmatrix} P_{11} & P_{12} & \dots & P_{1G} \\ P_{21} & P_{22} & \dots & P_{2G} \\ \vdots & \vdots & & \vdots \\ P_{G1} & P_{G2} & \dots & P_{GG} \end{bmatrix} \qquad \textit{(eq. box 4.41)}$$

Observe that these matrices are of size GxG.

For an R2 model, the basic layout is preserved, however each of the elements in the W and P matrices (e.g. W_1 or P_{11}) are replaced by sub-matrices of size GxG:

$$W_i=\begin{bmatrix} W_{i1} & 0 & \dots & 0 \\ 0 & W_{i2} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & W_{iG} \end{bmatrix}$$

$$P_{ij}=\begin{bmatrix} 0 & 0 & \dots & 0 & \\ 0 & 0 & & 0 & \\ \vdots & \vdots & & \vdots & \\ P_{ij1} & P_{ij2} & \dots & P_{ijG} & \leftarrow \text{j-th row of the matrix} \\ \vdots & \vdots & & \vdots & \\ 0 & 0 & \dots & 0 & \end{bmatrix} \qquad \textit{(eq. box 4.42)}$$

This increases the size of the complete W or P matrix to G^2xG^2.

In general, for a model with a Reichweite R and G components, the final W and P matrix will have a size of G^RxG^R. They can be created using the general layout of equations 4.42 and by replacing the elements recursively G-1 times by sub-matrices of the following generalised form:

$$W_{\dots}=\begin{bmatrix} W_{\dots 1} & 0 & \dots & 0 \\ 0 & W_{\dots 2} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & W_{\dots G} \end{bmatrix}$$

$$P_{\dots}=\begin{bmatrix} 0 & 0 & \dots & 0 & \\ 0 & 0 & & 0 & \\ \vdots & \vdots & & \vdots & \\ P_{\dots 1} & P_{\dots 2} & \dots & P_{\dots G} & \leftarrow \text{j-th row of the matrix} \\ \vdots & \vdots & & \vdots & \\ 0 & 0 & \dots & 0 & \end{bmatrix}$$ *(eq. box 4.43)*

The "…" indices are to be replaced with the indices of the parameters the sub-matrix is replacing in the parent matrix. E.g. when replacing the initial W_i elements for an R3 model, the "..." index would be replaced with the value of the "i" index of the replaced W_i parameter. The result will be a matrix containing W_{ij} parameters. Since we are creating a matrix for an R3 model, there need to be in total 2 rounds of replacements. This means we need to replace every W_{ij} parameter with another sub-matrix in which the "..." index is this time replaced with the value ij index values of the replaced W_{ij} parameter. We have then made G-1 recursive replacements and the matrix will have a size of $G^3 x G^3$ and contain parameters of the form W_{ijk}.

ii. Parameter and matrix indexation

Simple relationships can be derived to obtain the matrix column and row indexes from the parameter indexes and vice-versa.

W parameter indexes to column and row indexes:

$$x=y=\sum_{i=1}^{Z} z_i \cdot G^{R'-i}$$ *(eq. box 4.44)*

in which:

x	the W matrix column index
y	the W matrix row index
Z	the number of indexes in the W parameter (e.g. 2 for W_{11})
zi	the value of the i-th W parameter index (e.g. $z_1 = 2$ for W_{21})
G	the number of components
R'	R, but with a lower limit of 1, as in: $R':((R>1)\rightarrow R)\wedge((R\leqslant 1)\rightarrow 1)$

The opposite relation for W parameters is:

$$z_i=\left(\left\lfloor \frac{x-1}{G^{R-i}} \right\rfloor \bmod G\right)+1$$ *(eq. box 4.45)*

in which:

zi	the value of the i-th W parameter index (e.g. $z_1 = 2$ for W_{21})
G	the number of components
R'	R, but with a lower limit of 1, as in: $R':((R>1)\rightarrow R)\wedge((R\leqslant 1)\rightarrow 1)$
x	the W matrix column index (can be replaced with the row index since x=y)

P parameter indexes to column and row indexes:

$$x=\sum_{i=1}^{Z-1} z_i \cdot G^{R'-i}$$

$$y=\sum_{i=2}^{Z} z_i \cdot G^{R'-i}$$

(eq. box 4.46)

in which:

Z	the number of indexes in the P parameter (e.g. 3 for P_{123})
zi	the value of the i-th W parameter index (e.g. $z_1 = 3$ for P_{321})
G	the number of components
R'	R, but with a lower limit of 1, as in: $R':((R>1)\rightarrow R)\wedge((R\leqslant 1)\rightarrow 1)$
x	the P matrix column index
y	the P matrix row index

The opposite relation for P parameters is:

for $i \in \{1,2,\ldots,G\}$

$$z_i=\left(\left\lfloor \frac{x-1}{G^{R-i}} \right\rfloor \bmod G\right)+1$$

for i = G:

$$z_i=\left(\left\lfloor \frac{y-1}{G^{R-i}} \right\rfloor \bmod G\right)+1$$

(eq. box 4.47)

in which:

zi	the value of the i-th W parameter index (e.g. $z_1 = 2$ for W_{21})
G	the number of components
R'	R, but with a lower limit of 1, as in: $R':((R>1)\rightarrow R)\wedge((R\leqslant 1)\rightarrow 1)$
x	the W matrix column index (can be replaced with the row index since x=y)

4.8.2 R0 models – random interlayering

In this model the chance of finding any type of layer after another type of layer equals the relative weight of that layer in the whole stack. This type of disorder is actually a specific type of R1 ordering where for a stack containing G different types of layers we can write:

$$\begin{array}{l} P_{ga}=W_a \quad \text{and} \quad W_{ga}=W_g*P_{ga}=W_g*W_a \\ P_{gb}=W_b \quad \text{and} \quad W_{gb}=W_g*P_{gb}=W_g*W_b \\ P_{gc}=W_c \quad \text{and} \quad W_{gb}=W_g*P_{gc}=W_g*W_c \\ \ldots \\ P_{gG}=W_G \quad \text{and} \quad W_{gG}=W_g*P_{gG}=W_g*W_G \quad \text{with}\, g\in\{1,2,\ldots,G\} \end{array} \qquad \textit{(eq. box 4.48)}$$

In addition to the above relations, the sum of all the relative weights should equal one (see equation box 4.39):

$$\sum_{i=1}^{G} W_i=1$$

Therefore, since we have G weight fraction parameters and we have G+1 relations, we only need to choose G-1 independent parameters to be able to calculate all weight fractions and related probabilities. These G-1 parameters are chosen as such so that they can all be described by the following weight fraction definition:

$$Fw_g=\frac{W_g}{\sum_{i=g}^{G} W_i} \quad \text{with}\, g\in\{1,2,\ldots,G-1\} \qquad \textit{(eq. box 4.49)}$$

Using the above definition we can derive that:

$$W_g=Fw_g*\sum_{i=g}^{G} W_i \quad \text{for every } g \in\{1, 2, \ldots, G\} \qquad \textit{(eq. box 4.50)}$$

and using equation 4.39, we can also derive that:

$$\begin{array}{l} \underbrace{\sum_{i=1}^{G} W_i}_{=1}=\sum_{i=1}^{g-1} W_i+\sum_{i=g}^{G} W_i \\ \sum_{i=g}^{G} W_i=1-\sum_{i=1}^{g-1} W_i \quad \text{for every } g \in\{1, 2, \ldots, G\} \end{array} \qquad \textit{(eq. box 4.51)}$$

If we are considering the first ratio $Fw_{1,}$ then in equation 4.50 g = 1 and, using equation 4.39, can be written as:

$$\begin{array}{l} W_1=Fw_1\cdot\underbrace{\sum_{i=1}^{G} W_i}_{=1} \\ W_1=Fw_1 \end{array} \qquad \textit{(eq. box 4.52)}$$

Or, with other words, the first fraction equals the weight fraction for the first component. Knowing this, the other weight fractions can be derived in sequence by combining the relations in boxes 4.50 and 4.51. E.g. for g = 2 we can write:

$$W_2=Fw_2\cdot\sum_{i=2}^{G} W_i \qquad \textit{(eq. box 4.53)}$$

and since:

$$\sum_{i=2}^{G} W_i = 1 - \sum_{i=1}^{1} W_i = 1 - W_1$$ *(eq. box 4.54)*

the first expression becomes:

$$W_2 = Fw_2 \cdot (1 - W_1)$$ *(eq. box 4.55)*

As can be derived from the above equations, these R0 expression can be extended for any number of components. In fact, the implementation in PyXRD comprises a generalised probability model that can handle any number of layers. In practice however, there is little need for a 100 component mixed layer model, so the upper limit has been set to 6 different components (for now).

4.8.3 R1 models

PyXRD has R1 models available for mixed-layers with up to 4 different components.

i. Two-component R1 model

For two-component (G=2) R1 models, the independent parameters are W_1 and $P_{11}(\text{if } W_1 \leqslant 0.5)$ or $P_{22}(\text{if } W_1 > 0.5)$. The other parameters in this model can then be calculated using these two parameters:

$$
\begin{aligned}
&W_2 = 1 - W_1 \\
&\text{if } W_0 \leqslant 0.5: \\
&\quad P_{12} = 1 - P_{11} \\
&\quad P_{21} = W_1 \cdot \frac{P_{12}}{W_2} \\
&\quad P_{22} = 1 - P_{21} \\
&\text{if } W_0 > 0.5: \\
&\quad P_{21} = 1 - P_{22} \\
&\quad P_{12} = W_2 \cdot \frac{P_{21}}{W_1} \\
&\quad P_{11} = 1 - P_{12}
\end{aligned}
\qquad \textit{(eq. box 4.56)}
$$

ii. Three-component R1 model

For three-component (G=3) R1 models, the first two independent parameters are identical to the two-component model: W_1 and either $P_{11}(\text{if } W_1 \leqslant 0.5)$ or $P_{xx}(\text{if } W_1 > 0.5)$, the latter is defined as:

$$
P_{xx} = \frac{W_{22} + W_{23} + W_{32} + W_{33}}{W_2 + W_3} \qquad \textit{(eq. box 4.57)}
$$

The other four parameters are chosen to be the following fractions:

$$
\begin{aligned}
&Fw_1 = \frac{W_2}{W_3 + W_2} \\
&Fw_2 = \frac{W_{22} + W_{23}}{W_{22} + W_{23} + W_{32} + W_{33}} \\
&Fw_3 = \frac{W_{22}}{W_{22} + W_{23}} \\
&Fw_4 = \frac{W_{23}}{W_{32} + W_{33}}
\end{aligned}
\qquad \textit{(eq. box 4.58)}
$$

The primary weight fractions W_2 and W_3 can be calculated as follows:

$$
\begin{aligned}
&W_2 = (1 - W_1) \cdot Fw_1 \\
&W_3 = 1 - W_1 - W_2
\end{aligned}
\qquad \textit{(eq. box 4.59)}
$$

If $W_1 \leqslant 0.5$:

Since:

$$W_1 = W_{11} + W_{21} + W_{31} \Rightarrow P_{11} = \frac{W_1 - W_{21} - W_{31}}{W_1}$$

$$W_2 = W_{12} + W_{22} + W_{32} \Rightarrow P_{12} = \frac{W_2 - W_{22} - W_{32}}{W_1}$$

$$W_3 = W_{13} + W_{23} + W_{33} \Rightarrow P_{13} = \frac{W_3 - W_{23} - W_{33}}{W_1}$$

(eq. box 4.60)

We can write:

$$(P_{12} + P_{13}) \cdot W_1 = W_2 + W_3 - (W_{22} + W_{23} + W_{32} + W_{33}) = W_2 + W_3 - W_{xx}$$

$$W_{xx} = W_2 + W_3 - W_1 \cdot (P_{12} + P_{13})$$

Since $P_{11} + P_{12} + P_{13} = 1$ we can change this to:

$$W_{xx} = W_2 + W_3 - W_1 \cdot (1 - P_{11})$$

If $W_1 > 0.5$:

$$W_{xx} = (1.0 - W_1) \cdot P_{xx}$$

(eq. box 4.61)

From that point on the calculation is fairly straightforward:

$$W_{22} = W_{xx} \cdot Fw_2 \cdot Fw_3$$

$$W_{23} = W_{xx} \cdot Fw_2 \cdot (1 - Fw_3)$$

$$P_{22} = \frac{W_{22}}{W_2} \quad \text{if } W_2 > 0 \text{ else } 0$$

$$P_{21} = 1 - P_{22} - P_{23}$$

$$W_{32} = W_{xx} \cdot (1 - Fw_2) \cdot Fw_4$$

$$W_{33} = W_{xx} \cdot (1 - Fw_2) \cdot (1 - Fw_4)$$

$$P_{32} = \frac{W_{32}}{W_3} \quad \text{if } W_3 > 0 \text{ else } 0$$

(eq. box 4.62)

$$P_{33} = \frac{W_{33}}{W_3} \quad \text{if } W_3 > 0 \text{ else } 0$$

$$P_{31} = 1 - P_{33} - P_{31}$$

$$P_{12} = \frac{W_2 - W_{22} - W_{32}}{W_1} \quad \text{if } W_1 > 0 \text{ else } 0$$

$$P_{13} = \frac{W_3 - W_{23} - W_{33}}{W_1} \quad \text{if } W_1 > 0 \text{ else } 0$$

$$P_{11} = 1 - P_{12} - P_{13} \quad \text{if } W_1 > 0.5 \text{ else } P_{11} \text{ is given}$$

At this point we know all probability parameters P_{ij}. Together with the primary weight fractions W_1, W_2 and W_3, it possible to calculate any unknown W_{ij} using the equations from box 4.39.

iii. Four-component R1 model

As with previous R1 models, the first two independent parameters are W_1 and either $P_{11}(\text{if } W_1 \leqslant 0.5)$ or $P_{22}(\text{if } W_1 > 0.5)$. The other 10 parameters are chosen to be the following fractions:

$$Fw_1 = \frac{W_2}{W_2 + W_3 + W_4} \qquad Fw_2 = \frac{W_3}{W_3 + W_4}$$
$$Fw_3 = \frac{W_{22} + W_{23} + W_{24}}{\sum_{i=2}^{4} \sum_{j=2}^{4} W_{ij}} \qquad Fw_4 = \frac{W_{32} + W_{33} + W_{34}}{\sum_{i=3}^{4} \sum_{j=2}^{4} W_{ij}}$$

(eq. box 4.63)

$$Fw_{22} = \frac{W_{22}}{W_{22} + W_{23} + W_{24}} \qquad Fw_{23} = \frac{W_{23}}{W_{23} + W_{24}}$$
$$Fw_{32} = \frac{W_{32}}{W_{32} + W_{33} + W_{34}} \qquad Fw_{33} = \frac{W_{33}}{W_{33} + W_{34}}$$
$$Fw_{42} = \frac{W_{42}}{W_{42} + W_{43} + W_{44}} \qquad Fw_{43} = \frac{W_{43}}{W_{43} + W_{44}}$$

Again, the primary weight fractions can be easily calculated from these fractions:

$$W_2 = (1 - W_1) \cdot Fw_1$$
$$W_3 = (1 - W_1 - W_2) \cdot Fw_2 \qquad \text{(eq. box 4.64)}$$
$$W_4 = 1 - W_1 - W_2 - W_3$$

Then, if $W_1 \leq 0.5$, we can calculate W_{xx} using the following formula, derived in the same way as for the 3 component R1 model:

$$W_{xx} = W_1 \cdot (P_{11} - 1) + W_2 + W_3 + W_4 \qquad \text{(eq. box 4.65)}$$

If W1 > 0.5 , Wxx is calculated as:

$$W_{xx} = P_{xx} \cdot (W_2 + W_3 + W_4) \qquad \text{(eq. box 4.66)}$$

From that point on the calculation is again fairly straightforward, first we calculate a number of partial sums:

$$W_{2x} = W_{22} + W_{23} + W_{24} = W_{xx} \cdot Fw_3$$
$$W_{yx} = W_{32} + W_{33} + W_{34} + W_{42} + W_{43} + W_{44} = (W_{xx} - W_{2x}) \qquad \text{(eq. box 4.67)}$$
$$W_{3x} = W_{32} + W_{33} + W_{34} = W_{yx} \cdot Fw_4$$
$$W_{4x} = W_{42} + W_{43} + W_{44} = W_{yx} - W_{3x}$$

Then we can calculate some of the weight fractions and all of the probabilities as follows:

$$W_{22}=Fw_{22}\cdot W_{2x}$$
$$W_{23}=Fw_{23}\cdot(W_{2x}-W_{22})$$
$$W_{24}=W_{2x}-W_{22}-W_{23}$$

$$W_{32}=Fw_{32}\cdot W_{3x}$$
$$W_{33}=Fw_{33}\cdot(W_{3x}-W_{32})$$
$$W_{34}=W_{3x}-W_{32}-W_{33}$$

(eq. box 4.68)

$$W_{42}=Fw_{42}\cdot W_{4x}$$
$$W_{43}=Fw_{43}\cdot(W_{4x}-W_{42})$$
$$W_{44}=W_{4x}-W_{42}-W_{43}$$

$$P_{23}=W_{23}/W_2 \quad \text{or if } W_2=0:\ P_{23}=0$$
$$P_{24}=W_{24}/W_2 \quad \text{or if } W_2=0:\ P_{24}=0$$
$$P_{21}=1-P_{22}-P_{23}-P_{24}$$

$$P_{32}=W_{32}/W_3 \quad \text{or if } W_3=0:\ P_{32}=0$$
$$P_{33}=W_{33}/W_3 \quad \text{or if } W_3=0:\ P_{33}=0$$
$$P_{34}=W_{34}/W_3 \quad \text{or if } W_3=0:\ P_{34}=0$$
$$P_{31}=1-P_{32}-P_{33}-P_{34}$$

$$P_{42}=\frac{W_{42}}{W_4} \quad \text{or if } W_4=0:\ P_{42}=0$$
$$P_{43}=\frac{W_{43}}{W_4} \quad \text{or if } W_4=0:\ P_{43}=0$$
$$P_{44}=\frac{W_{44}}{W_4} \quad \text{or if } W_4=0:\ P_{44}=0$$
$$P_{41}=1-P_{42}-P_{43}-P_{44}$$

(eq. box 4.69)

$$P_{12}=\frac{W_2-W_{22}-W_{32}-W_{42}}{W_1} \quad \text{or if } W_1=0:\ P_{12}=0$$
$$P_{13}=\frac{W_3-W_{23}-W_{33}-W_{43}}{W_1} \quad \text{or if } W_1=0:\ P_{13}=0$$
$$P_{14}=\frac{W_4-W_{24}-W_{34}-W_{44}}{W_1} \quad \text{or if } W_1=0:\ P_{14}=0$$
$$P_{11}=1-P_{12}-P_{13}-P_{14} \quad \text{if } W_1>0.5 \text{ else } P_{11} \text{ is given}$$

At this point we know all probability parameters P_{ij}. Together with the primary weight fractions W_1, W_2, W_3 and W_4, it possible to calculate any unknown W_{ij} using the equations from box 4.39.

4.8.4 R2 models

PyXRD has R2 models available for models with up to 3 different components.

i. Two-component R2 model

This is one of the few exceptions where the choice of independent parameters differs from Sybilla. The reason is that Sybilla implements a restricted version, not allowing for the full range of possibilities. PyXRD does not have these constraints. The result is that PyXRD requires two more parameters (4 in tolal) compared with Sybilla.
As with previous models, the first independent parameter is W_1 and either $P_{112}(\text{if } W_1 \leqslant 2/3)$ or $P_{211}(\text{if } W_1 > 2/3)$, P_{21} and either $P_{122}(\text{if } W_1 \leqslant 0.5)$ or $P_{211}(\text{if } W_1 > 0.5)$.

The calculation proceeds as follows:

$$\begin{array}{lll}
W_2 = 1 - W_1 & \text{If } W_1 \leqslant 2/3: & \text{If } W_1 \leqslant 0.5: \\
P_{22} = 1 - P_{21} & \quad P_{211} = P_{112} \cdot \dfrac{W_{11}}{W_{21}} & \quad P_{221} = P_{211} \cdot \dfrac{W_{12}}{W_{22}} \\
 & \text{else:} & \text{else:} \\
W_{21} = W_2 \cdot P_{21} & \quad P_{112} = P_{211} \cdot \dfrac{W_{21}}{W_{11}} & \quad P_{122} = P_{221} \cdot \dfrac{W_{22}}{W_{12}} \\
W_{22} = W_2 \cdot P_{22} & P_{212} = 1 - P_{211} & P_{121} = 1 - P_{122} \\
W_{12} = W_{21} & P_{111} = 1 - P_{112} & P_{222} = 1 - P_{221} \\
W_{11} = W_1 - W_{21} & &
\end{array}$$

(eq. box 4.70)

ii. Three-component R2 model

The three-component R2 model is restricted in the sense that the weight fraction of the first component must be equal to or larger then 50% and no 2nd or 3d type component can follow or precede another 2nd or 3d type component. This restriction results into the following relations:

$$\begin{array}{l}
P_{22} = P_{23} = 0 \quad \Rightarrow P_{21} = 1 \\
P_{32} = P_{33} = 0 \quad \Rightarrow P_{31} = 1 \\
\\
P_{222} = P_{223} = 0 \quad \Rightarrow P_{221} = 1 \\
P_{232} = P_{233} = 0 \quad \Rightarrow P_{231} = 1 \\
P_{322} = P_{323} = 0 \quad \Rightarrow P_{321} = 1 \\
P_{332} = P_{333} = 0 \quad \Rightarrow P_{331} = 1 \\
\\
W_{21} = W_{12} = W_2 \\
W_{31} = W_{13} = W_3 \\
W_{11} = W_1
\end{array}$$

(eq. box 4.71)

Because of these restrictions, the number of independent variables is reduced to only 6. They are chosen to be W_1, either $P_{111}(\text{if } 0.5 \leqslant W_1 \leqslant 2/3)$ or $P_{212}(\text{if } 2/3 \leqslant W_1 \leqslant 1)$, and the following four fractions:

$$Fw_1 = \frac{W_1}{W_1 + W_2}$$
$$Fw_2 = \frac{W_{212} + W_{213}}{W_{212} + W_{213} + W_{312} + W_{313}}$$
$$Fw_3 = \frac{W_{212}}{W_{212} + W_{213}}$$
$$Fw_4 = \frac{W_{312}}{W_{312} + W_{313}}$$

(eq. box 4.72)

The calculation of the other parameters proceeds as follows:

$$W_2 = Fw_1 \cdot (1 - W_1)$$
$$W_3 = 1 - W_1 - W_2$$

$$P_{212} = \frac{Fw_2 \cdot Fw_1}{W_1} \cdot [W_{11} \cdot (P_{111} - 1) + 2] \quad \text{if } 0.5 \leqslant W_1 \leqslant 2/3$$
$$P_{213} = P_{212} \cdot \left(\frac{1}{Fw_3} - 1\right) \quad \text{or if } Fw_3 = 0 \Rightarrow P_{213} = 1$$

$$W_{212} = P_{212} \cdot W_{21} = P_{212} \cdot W_2$$
$$W_{213} = P_{213} \cdot W_2$$
$$W_{211} = 1 - W_{212} - W_{213}$$

$$W_{312} + W_{313} = \frac{1 - Fw_2}{Fw_3 \cdot Fw_4} \cdot W_{212} \quad \text{or if } Fw_3 = 0 \text{ or } Fw_4 = 0 \Rightarrow W_{312} + W_{313} = 0$$
$$W_{312} = Fw_4 \cdot (W_{312} + W_{313})$$
$$W_{313} = \left(\frac{1}{Fw_4} - 1\right) \cdot W_{312} \quad \text{or if } Fw_4 = 0 \Rightarrow W_{313} = 1$$
$$W_{311} = 1 - W_{312} - W_{313}$$

$$W_{111} = W_{11} - W_{211} - W_{311}$$

The remaining unknown parameters can be derived using the general equations as in equations 4.38 and 4.39.

4.8.5 R3 models

PyXRD has a single (restricted) R3 probability model built-in for a 2 component mixed layer.

Because of the restrictions only 2 independent variables are needed for this model; W_1 and $P_{1111}(\text{if } 2/3 \leqslant W_1 < 3/4)$ or $P_{2112}(\text{if } 3/4 \leqslant W_1 \leqslant 1)$.

The restrictions for this model, that:

- only mixed layers with more than 2/3 of the first layer type can be described
- no two layers of the second type occur after each other
- the probability of finding a layer of the first type in between two layers of the second type is zero

This translates into the following conditions:

$$2/3 \leqslant W_1 \leqslant 1$$
$$P_{22} = P_{212} = 0$$
$$P_{21} = P_{211} = 1$$

The probabilities below are undefined but are set to zero or one to make the matrix valid. The actual value actually has no meaning since the weight fractions they should be multiplied with equal zero anyway (e.g. $W_{2211} = W_{22} \cdot P_{221} \cdot P_{2211}$ and W_{22} is zero since P_{22} is zero):

$$P_{1121} = P_{1211} = P_{2211} = P_{2121} = P_{2221} = P_{1221} = 0$$
$$P_{1122} = P_{1212} = P_{2212} = P_{2122} = P_{2222} = P_{1222} = 1$$

The remaining probabilities and weight fractions can be calculated as follows and using equations 4.38 and 4.39:

$$W_2 = 1 - W_1$$

$\text{if } W_1 < 3/4$:

P_{1111} is given

$$P_{1112} = 1 - P_{1111}$$
$$P_{2111} = P_{1112} \cdot \frac{W_1 - 2 \cdot W_2}{W_2}$$
$$P_{2112} = 1 - P_{2111}$$

$\text{if } W_1 \geqslant 3/4$:

P_{2112} is given

$$P_{2111} = 1 - P_{2112}$$
$$P_{1111} = P_{2111} \cdot \frac{W_2}{W_1 - 2 \cdot W_2}$$
$$P_{1112} = 1 - P_{1111}$$

$$W_{111} = 3 \cdot W_1 - 2$$
$$W_{212} = W_{221} = W_{222} = W_{122} = 0$$
$$W_{211} = W_{121} = W_{112} = 1 - W_1$$

5 Software layout

The general software layout of PyXRD is schematically presented in Figure 1.

The most basic layer is inside the 'calculations' module. This module contains all the basic functions and data structures to allow fast calculations of X-ray diffraction patterns of disordered lamellar structures. In effect it is an implementation of the matrix algorithm as presented in the previous chapters. However, this layer is too basic to be of immediate use to a non-technical user. A GUI has been created to allow easier manipulation of objects. This is where the mvc framework comes into play.

The mvc framework consists of three layers: a model layer containing all the 'logic' and wrapping the data structures present in the calculation layer. On top of this layer two more layers are provided: a controller and a view. The view is the visual representation of a (or part of a) model(s) while the controller provides the link between that view and the underlying model(s). The mvc pattern is a common paradigm which allows to separate so-called 'business logic' aspects from GUI aspects, making it easier to debug these separately.

In the PyXRD source tree the models, views and controllers are grouped topic-wise, meaning that all views, controllers and models for a specific part (e.g. atoms, projects, phases, …) are grouped in a module named accordingly. Common code is provided by the mvc module. In future releases the mvc module might become a separate dependency. For now it is included with PyXRD.

The actual calculations (as presented in the previous chapters) are fully separated from the model layer into the calculations module. This makes it easier to spread the calculations over several processes, making efficient use of multi-core processors.

Aside from these layers, there are (more technical) aspects which are not covered in this manual like input/output, plotting, refinement support, etc. For details on these we refer to the source code and the documentation therein.

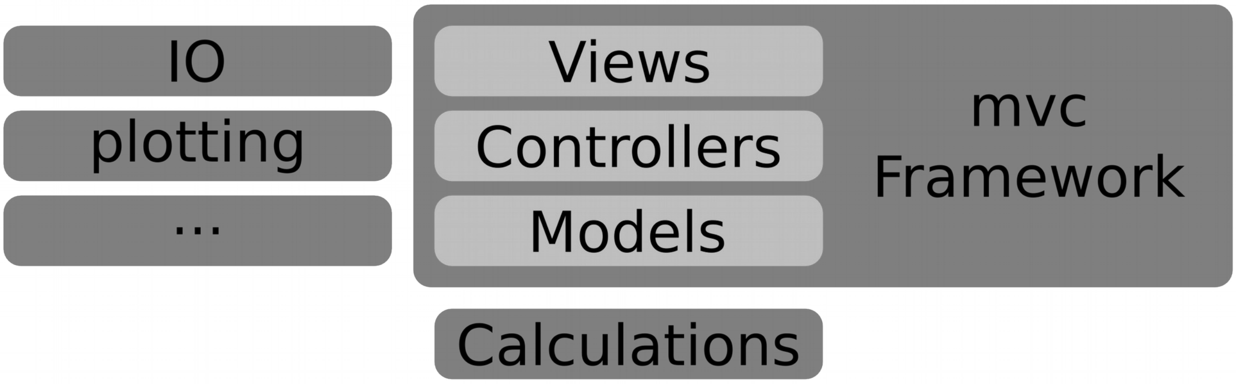


Figure 1: PyXRD's most important software layers

5.1 Model hierarchy

5.1.1 Overview

An overview of the model hierarchy can be found in Figure 2. The topmost model is the *Project*, which holds references to *AtomType*s, *Phase*s, *Specimens* and *Mixtures*. Each of these can have references to each other and several other objects each of which are discussed in more detail below.

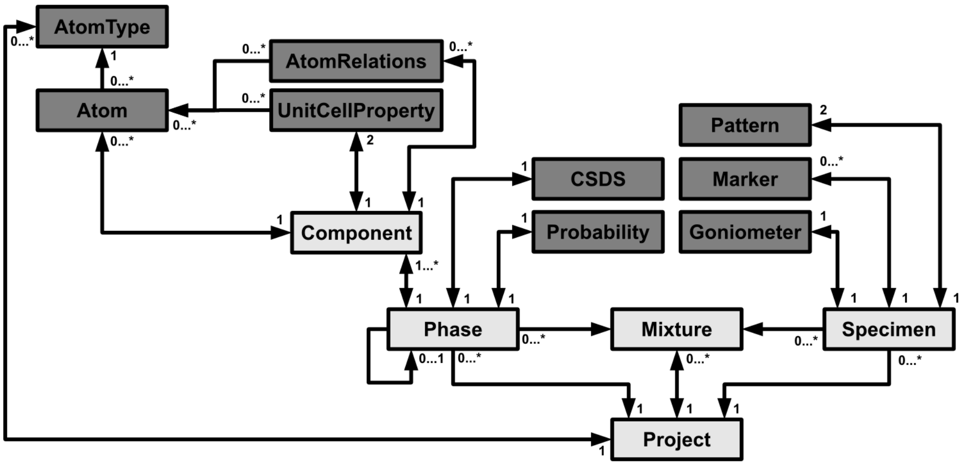


Figure 2: Overview of the model hierarchy in PyXRD

5.1.2 AtomTypes and Atoms

The most basic building block is the *AtomType*. This object bundles all the physical constants (e.g. charge, atomic weight, scattering factors, ...) for a single ion (e.g. Fe^{2+}, Fe^{3+}, …) or for a small enough molecule (i.e. H_2O or glycol). When a new project is created a default list of these AtomTypes is loaded, using the atomic scattering factors as published in (Waasmaier and Kirfel, 1995).

Atom objects hold a reference to one of these AtomTypes and has additional information about the position of that atom in the structure (a z coordinate) and its multiplicity (as atoms are projected, we can group them together). Lists of these atom objects are used in the Component object to describe the layers and interlayers. They also support expendable interlayers in which the z coordinate of the Atom is recalculated keeping the relative alignment correct.

5.1.3 Phases, Components, AtomRelations and UnitCellProperties

Phase objects contain all the information needed to calculate a one-dimensional X-ray diffraction pattern for a (mixed-layer) mineral. A *Phase* is built out of (i) a *Probability* object, (ii) an object describing the coherent scattering domain size (*CSDS*) and (iii) one or more *Component* objects which describe the different types of layers in the *Phase*. The *Probability* object describes how these layers are stacked using Markovian statistics and the Reichweite concept as detailed in chapter 4.8. The *CSDS* object describes what type of coherent scattering domain size distribution should be used and contains the necessary parameter values (e.g. average CSDS). Currently two types are implemented: a generic log-normal distribution and a log-normal distribution in which the average values published in Drits et al. (1997) are used and the average CSDS is the only variable. Each phase also has a σ* factor which allows to correct for incomplete preferred orientation (see chapter 4.6).

Component objects describe the size, structure, composition and (variation in) basal spacing for each layer type in that phase. A *Component* contains two lists of *Atom* objects. The first list contains atoms in the silicate lattice while the other list contains the variable interlayer ions. This way, the silicate structure can be shared between different phases (e.g. AD and EG states) while keeping the interlayer contents separate. It also allows to automatically adjust the positions of the interlayer content in function of the basal spacing, as the size of the silicate lattice can be determined and be used as a reference plane for scaling the interlayer atom positions.

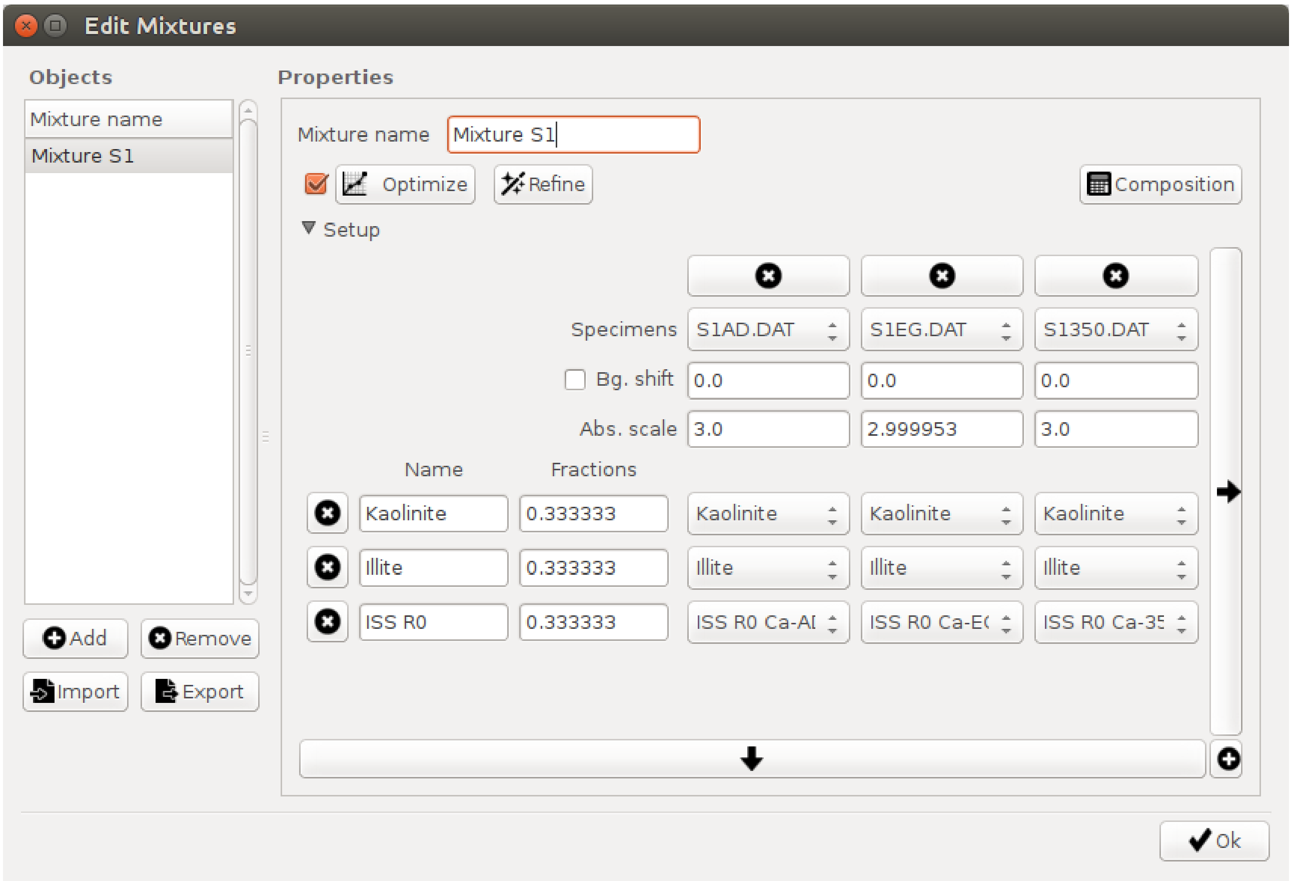
Inside *Components* one can also define several *AtomRelations*: these describe relations between atoms. One such relation are AtomRatio's, for example the octahedral composition of a dioctahedral clay mineral can be expressed as the ratio of iron atoms over the sum of iron and aluminium atoms in that octahedral position. The *AtomRatio* object will then make sure that the sum of both *Atoms* multiplicity is always 4, and that their relative amounts is controlled by a certain ratio set by the user. Both the value of the sum and the ratio can be adjusted. Another example is interlayer cation contents, which are controlled by an *AtomContents* object.

Another type of *AtomRelation* are *UnitCellProperties*. These objects describe the b and c axis length of the unti cell. This object is needed, since these properties are inter-dependent (i.e. the b-length can be calculated if the c-length is known and vice-versa) and dependent on the composition (e.g. the octahedral iron content). A *UnitCellProperty* object allows to define these simple mathematical relations, making the adjustment of the unit cell size automatic.

5.1.4 Specimens, Mixture and Projects

Specimen objects contain all the information regarding the experimental data (the actual measurements, sample size, etc.) and the *Goniometer* set up (radius, slit sizes, etc.). They also have a bunch of visual settings (e.g. linwidth, line color, wether to display phase profiles separately, …) and import/export functionality.
Maybe slightly counter-intuitive, they do not hold a direct reference to phases, but are linked with them using *Mixture* objects.

Mixture objects link phases and specimens together. To visualize this, in the user interface, a table is created with just as many rows as there are *Phases* in the *Mixture* and just as many columns as there are *Specimens* in the *Mixture*. At the column headers there are slots where the user can select the *Specimen* and in each cell of the grid there are slots where the user can select the corresponding *Phase*. This allows to select different states of smectite for an AD and an EG pattern loaded in a *Specimen* (Figure 3), while keeping unaffected *Phases,* like kaolinites and micas identical.



Mixture name: Mixture S1

Mixture name
Mixture S1

☑ Optimize

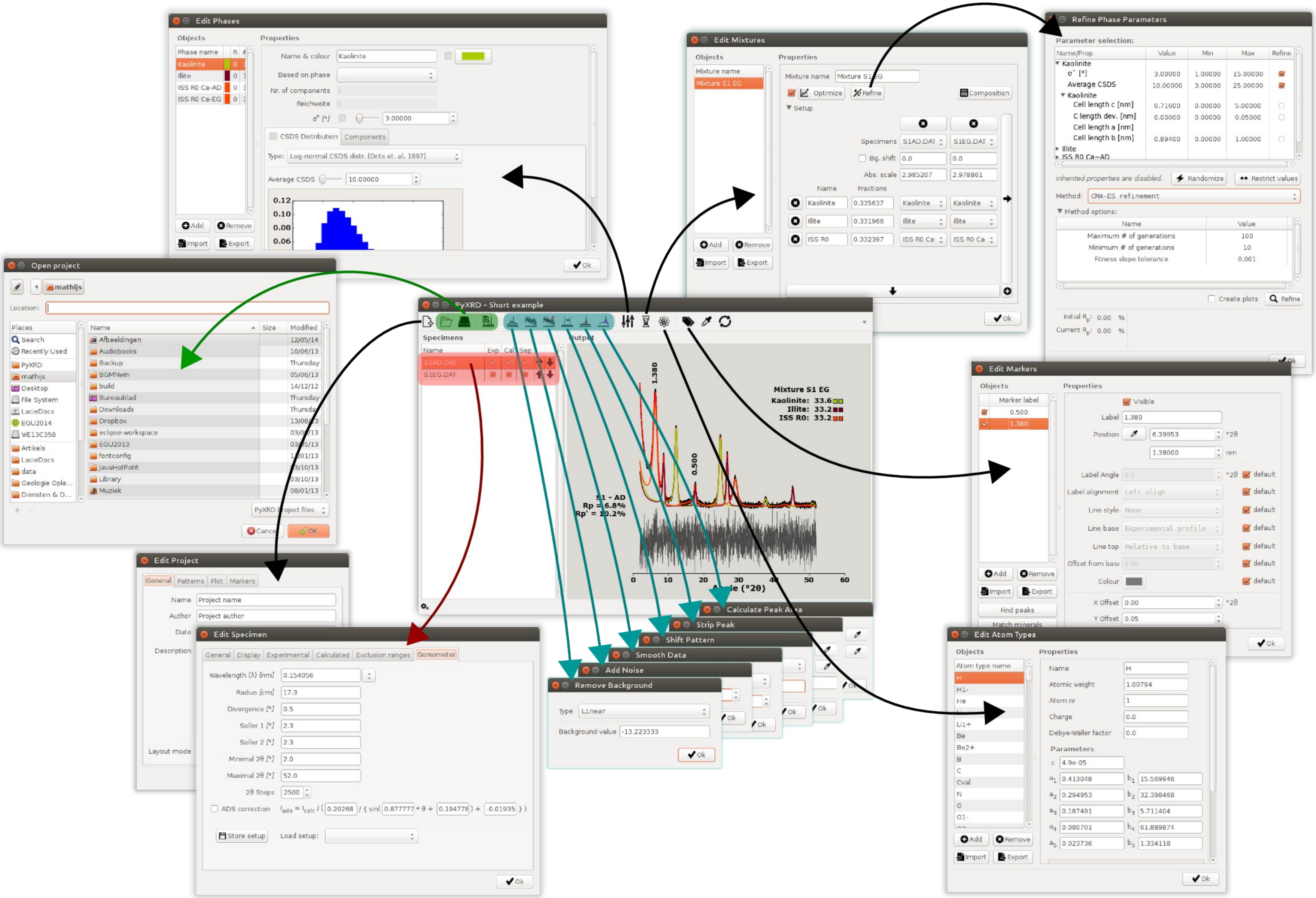
			S1AD.DAT	S1EG.DAT	S1350.DAT
		Specimens	S1AD.DAT	S1EG.DAT	S1350.DAT
		☐ Bg. shift	0.0	0.0	0.0
		Abs. scale	3.0	2.999953	3.0
Name	Fractions				
Kaolinite	0.333333		Kaolinite	Kaolinite	Kaolinite
Illite	0.333333		Illite	Illite	Illite
ISS R0	0.333333		ISS R0 Ca-AI	ISS R0 Ca-E(	ISS R0 Ca-35

Figure 3: screenshot showing the 'Edit Mixtures' dialog where a user can link different phases (Kaolinite, Illite, ISS R0 Ca-AD, ...) with the corresponding specimen (S1AD.dat, S1EG.dat).

Once a *Mixture* is created a number of parameters are available for automatic refinement (e.g. weight fractions from the *Probability* object, the average CSDS, etc.). In the refinement dialog, the user can select which parameters it would like to improve and in between which minimum and maximum values the ideal value should be searched for. A number of different refinement methods are available for this purpose.

As mentioned, Project objects group together all *Mixtures*, *Specimens, Phases* and *AtomTypes* and provides a way to store and (re)load these *Projects*.

6 GUI screenshot flowchart



Edit Phases

Name & colour: Kaolinite

Based on phase:

Nr. of components: 1

Reichweite: 0

σ* [°]: 3.00000

☐ CSDS Distribution | Components

Type: Log-normal CSDS distr. (Drits et. al, 1997)

Average CSDS: 10.00000

Edit Project

Name: Project name

Author: Project author

Edit Specimen (Goniometer)

Wavelength (λ) [nm]: 0.154056

Radius [cm]: 17.3

Divergence [°]: 0.5

Soller 1 [°]: 2.3

Soller 2 [°]: 2.3

Minimal 2θ [°]: 2.0

Maximal 2θ [°]: 52.0

2θ Steps: 2500

☐ ADS correction $I_{ads} = I_{calc} / (0.20268 / \{ \sin(0.877777 * \theta + 0.194778) + -0.01935 \})$

Remove Background

Type: Linear

Background value: -13.223333

Edit Mixtures

Mixture name: Mixture S1 EG

	S1AD.DAT	S1EG.DAT
Specimens	S1AD.DAT	S1EG.DAT
☐ Bg. shift	0.0	0.0
Abs. scale	2.985207	2.978861

Name	Fractions		
Kaolinite	0.335637	Kaolinite	Kaolinite
Illite	0.331966	Illite	Illite
ISS R0	0.332397	ISS R0 Ca	ISS R0 Ca

Refine Phase Parameters

Name/Prop	Value	Min	Max	Refine
Kaolinite				
σ* [°]	3.00000	1.00000	15.00000	☑
Average CSDS	10.00000	3.00000	25.00000	☑
Kaolinite				
Cell length c [nm]	0.71600	0.00000	5.00000	☐
C length dev. [nm]	0.00000	0.00000	0.05000	☐
Cell length a [nm]				
Cell length b [nm]	0.89400	0.00000	1.00000	☐
Illite				
ISS R0 Ca-AD				

Method: CMA-ES refinement

Name	Value
Maximum # of generations	100
Minimum # of generations	10
Fitness slope tolerance	0.001

Initial R_p: 0.00 %

Current R_p: 0.00 %

Edit Markers

Marker label
0.500
1.380

☑ Visible

Label: 1.380

Position: 6.39953 °2θ

1.38000 nm

Label Angle: 0.0 °2θ ☑ default

Label alignment: Left align ☑ default

Line style: None ☑ default

Line base: Experimental profile ☑ default

Line top: Relative to base ☑ default

Offset from base: 0.00 ☑ default

Colour: ☑ default

X Offset: 0.00 °2θ

Y Offset: 0.05

Edit Atom Types

Name: H

Atomic weight: 1.00794

Atom nr: 1

Charge: 0.0

Debye-Waller factor: 0.0

c: 4.9e-05

a_1	0.413048	b_1	15.569946
a_2	0.294953	b_2	32.398468
a_3	0.187491	b_3	5.711404
a_4	0.080701	b_4	61.889874
a_5	0.023736	b_5	1.334118

7 Guided example

7.1 Installing DEAP (optional)

DEAP is a python library PyXRD uses for its genetic refinement algorithms. These are much more robust then the default algorithms PyXRD provides.

To install DEAP in Windows:

- Open a command line as administrator (Start button → Search for “cmd” → Right click and select “Run as Administrator”)
- In the command line type the following:
  `C:\Python27\Scripts\easy_install.exe deap`

7.2 Instructions

This short example will guide through creating a simple project for phase refinement. It uses a calculated pattern as 'experimental data' so you can be sure everything has been set-up the way it should be.

The input files needed can be found online:

- http://users.ugent.be/~madumon/pyxrd/simple/S1AD.dat
- http://users.ugent.be/~madumon/pyxrd/simple/S1EG.dat

7.2.1 Creating a project and loading data

1. Get the two files above and store them somewhere you'll find them again.
2. Launch PyXRD
3. Create a new project by going to *Project → New Project* in the menu. An *Edit Project* dialog will appear.
4. By default, PyXRD runs in "View Mode" which means it hides all the user interface elements that allow quantitative analysis. Change the layout mode to 'Full' at the dropdown box at the bottom of the *Edit Project* dialog.
5. Import the 2 patterns you downloaded:
    - Right-click in the specimens list on the left of the main window (indicated in red in the screenshot flowchart)
    - Select *Import Specimens* from the drop-down menu
    - Browse to the 2 files you downloaded.
    - Select both files and click *Ok*

7.2.2 Goniometer settings

Now that we have loaded our data we still need to check if the Goniometer settings are correct. To do this:

1. Select the first specimen, right click it and select *Edit Specimen* from the pop-up menu.

2. In the *Edit Specimen* dialog select the *Goniometer* tab.
3. Load the default “Philips X'PERT PW3710 Cu” goniometer setyp by selecting the correct item in the '*Load Setup*' dropdown box at the bottom of the *Edit Specimen* dialog.

That's it, you've correctly setup the Goniometer for this example. For your own data you need to adjust the values to match your XRD goniometer setup. Also make sure you set the *'Sample length'* value in the *'General'* tab, for this example it is already correct.

7.2.3 Adding phases

Have a good look at those patterns and notice there are probably 3 phases in there: kaolinite, something like a mica and something like a smectite. Let's add these 3 phases and see if this works:

1. Open the *Edit Phases* dialog by clicking the corresponding icon in the toolbar or by going to *Data → Edit Phases* in the menu.
2. At the bottom of the phases list click the Add button.

   Note: If this is the first time you're adding Phases, no default Phases are available. However, they can be generated by clicking the reload button on the bottom right (having a "↻" sign on it).This takes a few minutes.
3. Select the following phases one-by-one, from the list of default phases:
   - Kaolinite
   - Illite
   - (dioctahedral) ISS R0 Ca (= randomly orderered Illite-Smectite)

   Default phases can be found in the dropdown box at the bottom.

If all went well you should have a list of 5 phases: the kaolinite, illite and 3 states of ISS R0: air-dry, ethylene-glycolated and heated.

7.2.4 Creating mixtures

The next step is to create a mixture where we link the 2 loaded patterns with the 3 added phases.

1. Click the Edit Mixtures button in the toolbar or go to Data → Edit Mixtures in the menu.
2. In the Edit Mixtures dialog, add a new mixture by clicking the 'Add' button at the bottom of the list on the left. Select the new mixture in the list.
3. After selecting your mixture, its properties should appear in the right pane of the 'Edit Mixtures' dialog:
   - Give it a proper name by editing the 'Mixture name' field at the top, e.g. "Mixture S1".
   - In the 'Setup' section, you can notice <u>three buttons</u>:
     - one at the bottom with a downwards-pointing arrow,
     - one on the right side with a rightwards-pointing arrow and
     - one in the bottom right corner with a "+" icon.

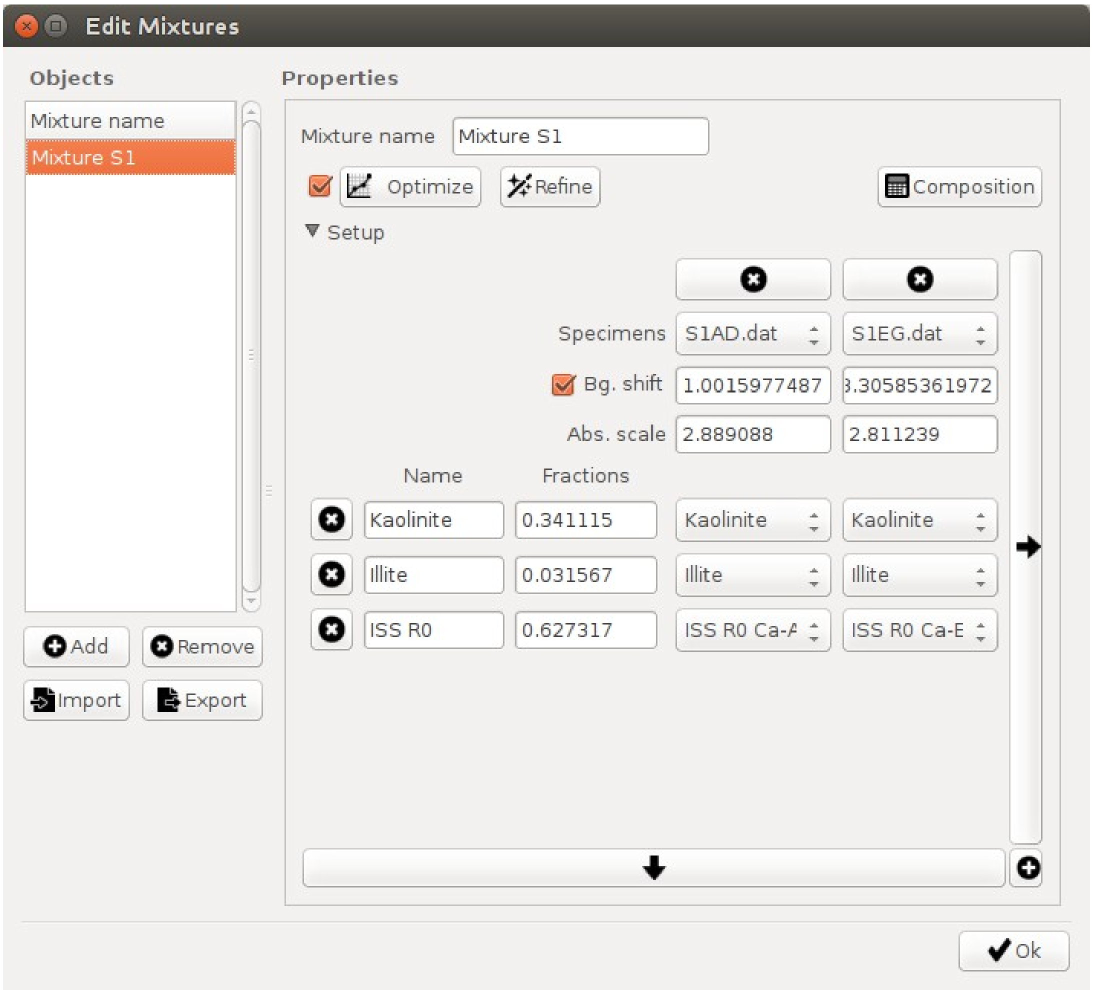
     These buttons can be used to add a phase, a specimen or both at the same time respectively, so:

- Click the phase button (bottom one) 3 times (once for each phase we want to add). You should immediately see a red line appearing on the plot. This is the calculated pattern, but since we have not selected our phases yet, it isn't calculating much yet.
- Then click the specimen button twice (once for each diffraction pattern or specimen).

4. On the line saying '*Specimens*', select the air-dry and glycolated patterns in the first and second dropdown box respectively.
5. For each '*New Phase'* line, change the name text input and dropdown boxes as follows:

Name input:	**Air-dry column phase dropdown:**	**Glycolated phase dropdown:**
Kaolinite	Kaolinite	Kaolinite
Illite	Illite	Illite
ISS R0	ISS R0 Ca AD	ISS R0 Ca EG

6. Finally, check the checkbox next to the *Optimize* button, so PyXRD will automatically adjust the contents of your phases to the best fit.
7. Your final *Edit Mixtures* dialog should look something like this:



Mixture S1

Mixture name: Mixture S1

☑ Optimize

Specimens: S1AD.dat, S1EG.dat

☑ Bg. shift: 1.0015977487, .30585361972

Abs. scale: 2.889088, 2.811239

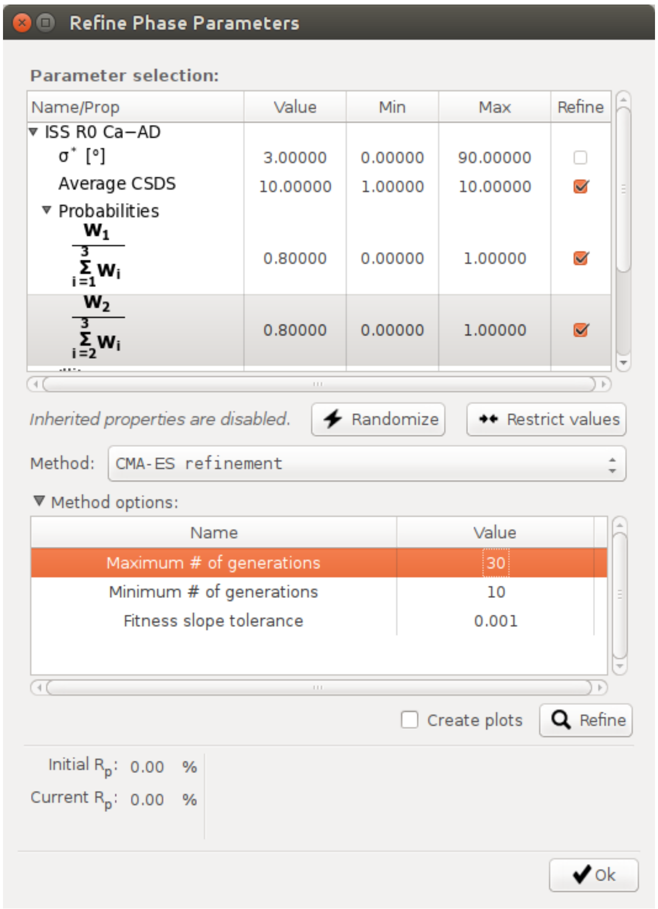
Name	Fractions		
Kaolinite	0.341115	Kaolinite	Kaolinite
Illite	0.031567	Illite	Illite
ISS R0	0.627317	ISS R0 Ca-A	ISS R0 Ca-E

8. To update your calculated patterns, press the *Optimize* button. Then make sure you have selected both specimens (in the main window) and check for the presence of a black line (experimental data) and a red line (calculated data) in the plot area.

7.2.5 Refine some parameters

As you will have noticed, the calculated pattern does not really fit the experimental pattern that well. We can try to improve the fit by adjusting some of the parameters:

1. If you've closed the *Edit Mixtures* dialog, re-open it
2. Click the 'Refine' button in the 'Edit Mixtures' dialog. A *Refine Phase Parameters* dialog will appear.
3. For this simple example, 3 parameters were changed in the ISS R0 phase. So we're going to set up our refinement strategy for these 3 parameters alone:
   - ○ Click the arrow next to the 'ISS R0 Ca-AD' entry to unfold the options for this phase. Then check the 'Refine' box for the following parameters:
     - Average CSDS
     - And under *Probabilities*:
       - ○ W1 / ∑1-3(Wi)
       - ○ W2 / ∑2-3(Wi)
   - ○ For the 'Average CSDS' it is also best to change the maximum value to 10.
4. Change the algorithm to *CMA-ES refinement* if you've installed DEAP, and change the number of generations to 30. Otherwise leave at L BFGS B.
5. After these changes your dialog should look something like this:



Parameter selection:

Name/Prop	Value	Min	Max	Refine
▼ ISS R0 Ca–AD				
σ^* [°]	3.00000	0.00000	90.00000	☐
Average CSDS	10.00000	1.00000	10.00000	☑
▼ Probabilities				
$\frac{W_1}{\sum_{i=1}^{3} W_i}$	0.80000	0.00000	1.00000	☑
$\frac{W_2}{\sum_{i=2}^{3} W_i}$	0.80000	0.00000	1.00000	☑

Inherited properties are disabled.

Method: `CMA-ES refinement`

▼ Method options:

Name	Value
Maximum # of generations	30
Minimum # of generations	10
Fitness slope tolerance	0.001

☐ Create plots

Initial R_p: 0.00 %
Current R_p: 0.00 %

6. Finally, click the *Refine* button and wait for the computer to finish its calculation. If successful, the refinement should end up close to these values:
    - Average CSDS = 5
    - Probabilities:
        - W1 / ∑1-3(Wi) = 0.1
        - W2 / ∑2-3(Wi) = 0.5

When closing the refinement dialog, you can also check if the phases are given weight percentages in a 1:1:1 ratio. Check this by closing the 'Refine Phase Parameters' dialog and clicking the 'Optimize' and 'Apply' buttons in the 'Edit Mixtures' dialog. After that the 'Fractions' column should have (near-to-)equal proportions for each phase.

That's it, you have used PyXRD succesfully for the first time!

8 References

Drits, V. and Tchoubar, C. (1990). X-Ray Diffraction by Disordered Lamellar Structures: Theory and Applications to Microdivided Silicates and Carbons. pp. 369. *Springer-Verlag,* Berlin, Germany.

Drits, V., Środon, J. and Eberl, D.D. (1997). XRD Measurement of mean crystallite thickness of illite and illite/smectite: reappraisal of the Kubler Index and the Scherrer Equation. *Clays and Clay Minerals*, **45**, 461-475.

Ergun, S. (1970). X-ray scattering by very defective lattices. *Phys. Rev. B,* **131**, 3371-3380.

Kakinoki, J. and Komura, Y. (1952) Intensity of X-ray diffraction by one dimensionally disordered crystals. I: General derivation in the case of the “Reichweite” S=0 and 1. *Journal of the Physical Society of Japan*, **7**, 30-35.

Kakinoki, J. and Komura, Y. (1954a) Intensity of X-ray diffraction by one-dimensionally disordered crystals. The close packed structure. *Journal of the Physical Society of Japan*, **9**, 177-183.

Kakinoki, J. and Komura, Y. (1954b) Intensity of X-ray diffraction by one dimensionally disordered crystals. II: General derivation in the case of the correlation range S≥ 2. *Journal of the Physical Society of Japan*, **9**, 169-176.

Kakinoki, J. and Komura, Y. (1965) Diffraction by a one-dimensionally disordered crystal. I: The intensity equation. *Acta Crystallographica*, **19**, 137-147.

Moore, D.M. and Reynolds, R. C. (1997) X-Ray Diffraction and the Identification and Analysis of Clay Minerals, 2nd edition. pp. 400. *Oxford University Press,* New York, USA.

Plançon A. (1981). Diffraction by layer structures containing different kinds of layers and stacking faults. *Journal of Applied Crystallography*, **14**, 300-304.

Plançon A. (2002) New modeling of X-ray diffraction by disordered lamellar structures, such as phyllosilicates. *American Mineralogist*, **87**, 1672-1677.

Plançon, A. and Tchoubar, C. (1976). Etude des fautes d’empilement dans les kaolinites partiellement désordonnées. *Journal of Applied Crystallography*, **9**, 279-285.

Reynolds, R.C. (1986). The Lorentz-Polarisation factor and preferred orientation in oriented clay aggregates. *Clays and Clay Minerals*, **34**, 359-367.

Sakharov, B.A., Naumov, A.S. and Drits, V.A. (1982a). X-ray diffraction by mixed-layer structures with random distribution of stacking faults. *Doklady Akademii Nauk SSSR*, **265**, 339-343 (in Russian).

Sakharov, B.A., Naumov, A.S. and Drits, V.A. (1982b) X-ray intensities scattered by layer structure with short range ordering parameters S ≥ 1 and G ≥ 1. *Doklady Akademii Nauk SSSR*, **265**, 871-874 (in Russian).

Sakharov, B.A., Lindgreen, H., Salyn, A.L. and Drits, V.A. (1999) Determination of illite- smectite structures using multispecimen X-ray diffraction profile fitting. *Clays and Clay Minerals*, **47**, 555-566.

Waasmaier D. and Kirfel A. (1995). New analytical scattering factor functions for free atoms and ions. *Acta Cryst. A*, **51**, 416-413