Current Version of Sindarin manual: 250626

Site: www.nimloth.app

Youtube: www.youtube.com/@nimlothapp

Email: nimlothapp@gmail.com

Author: Guilherme Oliveira Siqueira

If you have a suggestion or want to report an issue, I would be glad to hear from you.

Summary

1 The Sindarin 9
1.1 Features 9
1.2 Why "Sindarin language"? 10
2 User Interface 11
2.1 Command Line Interface 11
2.2 Integration with Visual Studio Code 13
2.2.1 Text Highlighting 13
2.2.2 Sindarin Commands 14
2.2.3 See Diffractograms 14
2.2.4 Using Sindarin with VS Code 15
2.3 Nimloth - A Dedicate User Interface 16
2.4 Sindarin file 17
3 Sindarin Syntax 18
3.1 Data Type 18
3.1.1 Integer 18
3.1.2 Decimal 18
3.1.3 String 19
3.1.4 Boolean 19
3.1.5 Enumerator 19
3.1.6 Array 19
3.2 Objects 20
3.2.1 Parameters 21
3.2.2 Object Parenting 21
3.2.3 Indentation 21
3.2.4 Object's Block 21
3.2.5 Properties 22
3.2.6 Implicit Objects and default values of parameters and properties 23
3.2.7 Some details of children 23
3.3 Variables 23
3.3.1 Fitting 24
3.3.2 Restraints 25
3.3.3 Implicit Variables 25
3.3.4 Internal Variables 26

3.4 Comments 26
3.5 Case Insensitive 27
3.6 In what order should it be written? 27
3.7 Custom User Functions 28
4 Specific information 31
4.1 The Sindarin Interpreter 31
4.2 Sindarin Commands 31
4.2.1 Interpret command 31
4.2.2 Optimization commands 31
4.2.3 Results command 31
4.2.4 Reset command 31
4.3 Optimization Routine 31
4.4 Errors returned by the Interpreter 32
4.5 Experimental Data File 32
4.6 An overview of peak diffraction calculation 33
4.6.1 General equations for calculating the position and intensity of reflections 33
4.7 Refinement with Different structure details 34
4.7.1 Complete structure information (Rietveld Refinement) 34
4.7.2 Only lattice parameter information (Pawley method) 35
4.7.3 No structure information 35
4.8 Profiles 36
4.8.1 Profile level 36
4.8.2 Convolution of peak profiles 36
4.8.3 Fundamental Parameters Approach (FPA) 37
4.8.4 Whole powder pattern modelling (WPPM) 38
4.8.5 The pseudo-Voigt function 38
4.8.6 Peak asymmetry 40
4.9 Diffraction geometries 41
4.9.1 Diffraction Setup: Reflection, Symmetric, Flat Specimen, without Lorentz-Polarization correction 41
4.9.2 Diffraction Setup: Reflection, Symmetric, Flat Specimen 42
4.9.3 Diffraction Setup: Reflection, Symmetric, Flat Specimen, Polarized incident beam 42
4.9.4 Diffraction Setup: Reflection, Symmetric, Flat Specimen, Monochromator 42

4.9.5 Diffraction Setup: Reflection, Symmetric, Flat Specimen, Monochromator, Polarized incident beam 42
4.9.6 Diffraction Setup: Reflection, Symmetric, Thin Film 43
4.9.7 Diffraction Setup: Reflection, Asymmetric, Flat specimen 43
4.9.8 Diffraction Setup: Reflection, Asymmetric, Thin Film 43
4.9.9 Diffraction Setup: Transmission, Capillary 43
4.10 Radiation modeling 44
4.11 Crystallographic Information 45
4.12 Example of Sindarin text to Rietveld refinement 45
5 Tutorial 48
5.1 General information 48
5.2 Tutorial 0: First contact with Sindarin 49
5.3 Tutorial 1: Quantitative Phase Analysis 50
5.4 Tutorial 2: Lattice Parameters Analysis 54
6 Sindarin Dictionary 61
6.1 Words by object type 61
6.1.1 Calculation Control 61
6.1.2 Experimental data 61
6.1.3 Diffraction 61
6.1.4 Crystallographic 61
6.1.5 Background Function 62
6.1.6 Profile 62
6.1.7 Decimal Operators and Functions 63
6.1.8 Decimal Constants 65
6.1.9 Internal variables 65
6.2 Alphabetic Entries 66
6.2.1 AbsorptionFunction 66
6.2.2 AngularDependency 67
6.2.3 Atoms 68
6.2.4 AxialDivergence 68
6.2.5 BackgroundFile 69
6.2.6 Box 69
6.2.7 Calculation 70
6.2.8 CalculeProfileInPeak 70
6.2.9 Capillary 71
6.2.10 Cell 71

6.2.11 CenteredInMiddle 72
6.2.12 Chebyshev 72
6.2.13 Circle 73
6.2.14 ConvolutionSMax 74
6.2.15 ConvolutionType 74
6.2.16 ConvolutionXMax 75
6.2.17 CosineFourier 75
6.2.18 CutOff 76
6.2.19 Cycles 76
6.2.20 Data 76
6.2.21 DataSimulation 77
6.2.22 DebyeLike 77
6.2.23 Defocusing 78
6.2.24 DerivationBothSide 79
6.2.25 Direction 79
6.2.26 Displacement 80
6.2.27 DistributionType 80
6.2.28 Ellipsoidal 81
6.2.29 EnableIntensity 81
6.2.30 EnableShift 82
6.2.31 EquatorialDivergence 82
6.2.32 ExcludeRegion 83
6.2.33 Experimental 83
6.2.34 Exponential 84
6.2.35 FFTResolution 85
6.2.36 Function 85
6.2.37 FunctionPsVoigt 86
6.2.38 Gauss 86
6.2.39 GaussWavelength 87
6.2.40 IncidentAngle 88
6.2.41 Interpolation 88
6.2.42 Interval 89
6.2.43 Inverse 89
6.2.44 InverseProfile 90
6.2.45 Lorentz 90
6.2.46 LorentzWavelength 91

6.2.47 ManageHighCorrelation 91
6.2.48 MarchDollase 92
6.2.49 MeanStandardDeviation 92
6.2.50 MonochromatorTwoTheta 93
6.2.51 Multiplicity 93
6.2.52 PeakXI 94
6.2.53 Phase 94
6.2.54 Polarization 95
6.2.55 Polynomial 95
6.2.56 ProfileFunction 96
6.2.57 ProfileFunctionFT 96
6.2.58 ProfileXMax 97
6.2.59 psVoigt 97
6.2.60 psVoigtWavelength 98
6.2.61 Radiation 99
6.2.62 RapidYRecalcule 99
6.2.63 ReceivingSlitWidth 99
6.2.64 ReflectionHKL 100
6.2.65 ReflectionXI 100
6.2.66 Scale 101
6.2.67 ScaterringFactor 101
6.2.68 Shape 102
6.2.69 Shift 102
6.2.70 ShiftFT 103
6.2.71 Simulation 103
6.2.72 SizeDistribution 104
6.2.73 SpecimenThickness 105
6.2.74 Split 105
6.2.75 Stabilization 106
6.2.76 TargetWidth 106
6.2.77 Tau 107
6.2.78 Tilt 107
6.2.79 Transparency 108
6.2.80 UseAnomalousScattering 108
6.2.81 VariableSlit 109
6.2.82 Voigt 109

6.2.83 VoigtianSize 110
6.2.84 VoigtianStrain 111
6.2.85 VoigtWavelength 112
6.2.86 WeightPoint 113
6.2.87 Width 113
6.2.88 Z 114
6.2.89 Zero 114
7 Bibliography 115

1 THE SINDARIN

Sindarin is a program designed to perform various calculations for polycrystalline diffraction techniques, including but not limited to, Rietveld refinement, Fundamental Parameters Approach (FPA), and Whole Powder Pattern Modelling (WPPM). It is designed to be flexible and straightforward, utilizing a plain text file written in a proprietary syntax for calculations.

1.1 FEATURES

The main features of Sindarin include:

- Different structure details:
  - Rietveld (full structure information)
  - Pawley method (only cell information)
  - Without structure information.
- Background: Highly flexible
  - Combination of different background models
  - Several analytical functions
  - Read from file
  - Interpolation
  - Custom analytical function
- Profile: Comprehensive and flexible reflection profile calculation
  - Several analytical functions
  - Convolution of multiple profile functions
  - Profile asymmetry correction
  - Fundamental Parameters Approach (FPA)
  - Whole powder pattern modelling (WPPM)
  - Custom analytical function
- Reflection intensity corrections:
  - Monochromator correction
  - Absorptions
  - Preferred orientation
    - March and March-Dollase multi direction
    - Arbitrary and individual correction of reflection intensity
- Reflection position corrections:
  - Zero
  - Specimen Displacement
  - Shift: Arbitrary and individual correction of reflection position
- Different diffraction geometries:
  - Reflection
  - Transmission
  - Symmetric
  - Asymmetric
  - Flat specimen
  - Thin film

    - Capillary
    - Variable divergence slit
- Radiation modeling
    - Possibility to model the radiation flexibly
    - Any wavelength
    - Any relative intensity of discrete lines
    - Any number of discrete lines
    - Profile modeling of each discrete line separately
        - Voigtian profiles described in wavelength with physical variation of the FWHM with the diffraction angle
    - Possibility to fit the incident beam polarization
- Multiple data files for the same experiment
- Multiple experiments fitted together
    - Possibility to simultaneously refine multiple experiment data under different conditions, for example, different radiations

1.2 WHY "SINDARIN LANGUAGE"?

Sindarin uses a plain text file to store the data for calculations. An internal interpreter reads this file and performs the calculations. To make the file readable, it has to use correct words that the interpreter recognizes and follow a specific pattern of writing. This is why we call it the Sindarin language, with its own "dictionary" and "grammar" (syntax).

Don't worry if you think that Sindarin is like the old software that needed strict sequences of characters to work properly. We know that those software could be frustrating and time-consuming, especially when a small mistake like a space character could cause errors. Sindarin is different. It is flexible and user-friendly, and it doesn't have the rigid character sequence rules of the old software.

I chose this method because it is simple and flexible for creating theoretical models for diffraction. It lets you control the calculation and adjust the model to fit the needs of each experiment and data set.

You can edit the plain text file in any text editor you like. Nimloth is a good option because it is fully integrated with Sindarin. You can use it to edit texts, do calculations, and see results. It also has some tools that make your tasks easier. Another option is Visual Studio Code, which has a Sindarin extension that helps you use Sindarin with this editor. You can find more details in Section 2.

One last thing: when we say "language" in Sindarin, we mean the way of inputting data. We don't mean the same thing as in software development.

2 USER INTERFACE

To utilize Sindarin, users have several options available: a command line interface (CLI), integration with Visual Studio Code, and Nimloth, a dedicated user interface. All three options can be used on the three major operating systems (OS): Windows, macOS, and Linux. Nimloth will likely be the best choice for most users.

2.1 COMMAND LINE INTERFACE

The Command Line Interface (CLI) is Sindarin's basic interface and can be used in terminals of each OS. The structure of a Sindarin command is as follows

```
1| Sindarin [help] <sindarin file> [interpreter] [options]
```

Where [help], <sindarin file>, [interpreter] and [options] are the type of arguments. The [help] argument is used alone. To use a interpret command, the <sindarin file> is mandatory, and [interpreter] and [options] are optional.

If the terminal is not in Sindarin program folder, it will be necessary to write the full system path for executable program, not just "Sindarin" at the beginning of the command. The example below shows how to run the Sindarin when the program's folder is a folder called "Sindarin" in the drive C: on Windows OS:

```
1| C:\Sindarin\Sindarin
```

It is highly recommended to add the Sindarin program folder to the OS environment variable. This way, you can omit the path for the program folder and just use "Sindarin" at the beginning of the command, even if the terminal is not in Sindarin folder.

Executing the help command or the Sindarin without an argument returns a list of possible arguments, like this:

```
1| Sindarin [help] <sindarin file> [interpreter] [options]
2|     [help]
3|         -h or --help: show these command information
4|         -t or --test: run test Sindarin file: \Test\test.sin
5|         -u or --update: run Sindarin program update from web
6|         -v or --version: show the version of Sindarin program
7|     <sindarin file> the system path of Sindarin file to be
   interpreted
8|     [interpreter]
9|         -I or --interpret: interpret the file (default)
10|        -S or --step: step in optimization calculation
11|        -W or --walk: walk in optimization calculation
12|        -M or -- marathon: marathon in optimization calculation
13|    [options]
14|        -r or --results: get and write results to file
15|        -c or --charts: get and write charts to file
16|        -f or --format: format the Sindarin text before to
   interpret
17|        -n or --norewrite: Sindarin file will not change
18|        -z or --all: enable all options, except --norewrite
```

Some arguments are explicated with more details bellow.

--update (-u)

If the message bellow is written in terminal after executing a Sindarin command, there is a new version of Sindarin available.

```
1| There is a new version of Sindarin
2| Current version: ###
3| New version: ###
4| Run the update command: Sindarin --update
```

If the computer is connected to the internet, run the command to update:

```
1| Sindarin -u
```

<sindarin file>

This argument is the full or relative system path to Sindarin file that will be interpreted. The path must be written according of operation system that is running, the "\" character is used on Windows and "/" on Linux and macOS. Below is an example of the file "test.sin" that is in folder "Test" inside the folder "Sindarin" in drive c: on Windows OS.

```
1| Sindarin C:\Sindarin\Test\Test.sin
```

[interpreter]

These arguments select the Interpreter command that will run with the Sindarin file. If this argument is omitted, the default is --interpret (-I). To read about the Interpreter Command, see the Sindarin Commands section.

[options]

Some options when running an Interpreter command

--results(-r)

After executing the Interpreter command, the results will be written in a plain text file with the name "Sindarin file name - Results.txt" in the same folder as the Sindarin file.

--charts (-c)

After run Interpreter command, the experimental data, calculated diffraction, and background will be written in a plain text file named "Sindarin file name - Chart.txt" with a TSV (tab-separated values) format.

--format (-f)

Before running the Interpreter command, the text in the Sindarin file will be formatted. The indentation will be written with standard character (tab) and only one character per level of indentation. Consecutive white spaces will be removed to leave only one character.

--norewrite (-n)

The default behavior of Sindarin is to rewrite the Sindarin file replacing the values of refinable variables with new values after an Interpreter command. If this argument is used, the original text will not be modified.

2.2 INTEGRATION WITH VISUAL STUDIO CODE

Visual Studio Code (VSCode) is a robust text editor that utilizes extension packages to modify behavior and add functionality. It specializes in editing code text and can run external programs when properly configured.

Sindarin for VS Code is an extension that integrates VSCode with Sindarin. This extension enables functionalities that simplify the use of Sindarin, eliminating the need to use the command line interface to run Sindarin. Instead, Sindarin can be executed directly from VSCode by clicking on Sindarin buttons enabled by the extension.

The use of VS Code with Sindarin is highly recommended over the command line interface to enhance user experience and productivity.

2.2.1 Text Highlighting

Sindarin text in VSCode is highlighted, with words appearing in different colors according to their type, such as class name, string, number, enumerator, operator, and others (see image below). This feature enhances the visibility of the text. The color scheme aligns with the one configured in VSCode and can be changed. For instructions on how to change the color scheme, refer to the official site:

https://code.visualstudio.com/docs/getstarted/themes

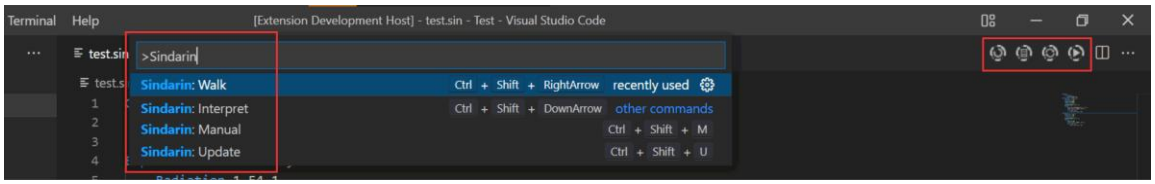
```
test.sin
1   Computer
2       Convolution
3           ProfilexMax 10
4   Experimental "teste.xye"
5       Radiation 1.54 1
6       Polynomial 100
7       CutOff 0.000001
8       //Polarization 0
9       IncidentAngle 4{0.03}
10      Phase
11          Scale @0.00099988725{±1.1E-07}
12          VoigtianSize 1000 1 0.5 0
13          Cell "230" 10
14              O 0.1 0.1 0.1 1 0.1
15  Simulation
16      Radiation 1.54 1
17      Polynomial 100
18      //Polarization 0
19      IncidentAngle 4{0.03}
20      Phase
21          Scale 0.001
22          VoigtianSize 1000 1 0.5 0
23          Cell "230" 10
24              O 0.1 0.1 0.1 1 0.1
```

2.2.2 Sindarin Commands

Several Sindarin commands are enabled in VS Code:

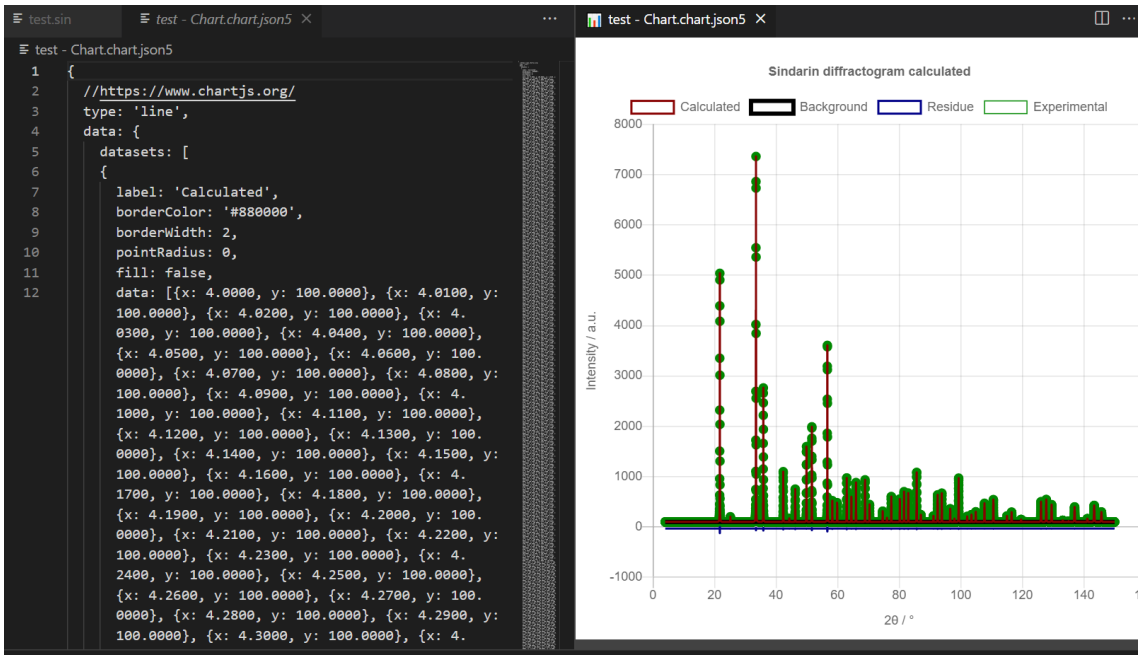
1. Interpret: Runs Sindarin to interpret for the current Sindarin file (Shortcut: Ctrl+Shift+Down Arrow)
2. Walk: Runs Sindarin to interpret and execute the optimization routine for the current sindarin file (Shortcut: Ctrl+Shift+Right Arrow)
3. Update: Runs Sindarin to update program from online repository, requires an internet connection (Shortcut: Ctrl+Shift+U)
4. Manual: Opens the Sindarin manual (Shortcut: Ctrl+Shift+M)

These commands can be accessed by Command Palette (Ctrl+Shift+P), shortcut keys, or dedicated icons in the editor title (see image below).



2.2.3 See Diffractograms

The Chart.js Preview extension is installed alongside the Sindarin extension. This allows you to view diffractograms within VS Code (see image below) from the ".chart.json5" file generated after executing the interpret or walk commands..



2.2.4 Using Sindarin with VS Code

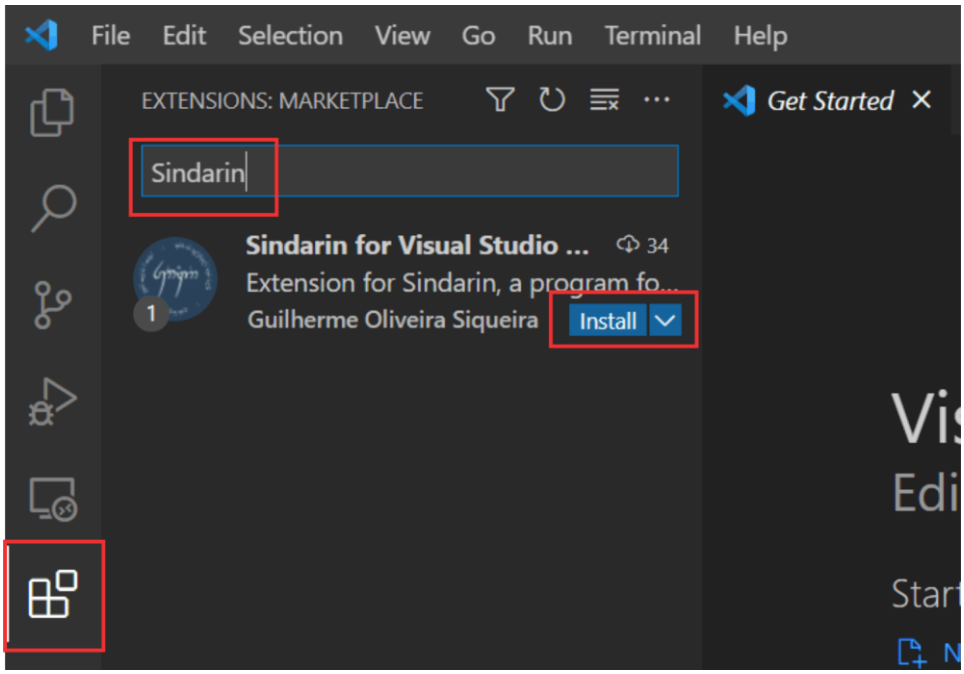
To use Sindarin with VS Code, follow these steps:

1. Download and unzip the file to default folder.

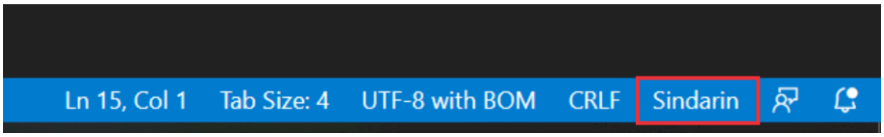
Default folders:

- Windows: C:\Sindarin
- Linux: ~/Sindarin
- macOS: ~/Sindarin

2. Download and install the VS Code from the official site:
   https://code.visualstudio.com/
3. Go to Extensions in left side (Activity Bar of VS Code) (shortcut Ctrl+Shift+X), type 'Sindarin' to search the Sindarin for VS Code extension, and click on install bottom, as shown in the image bellow.

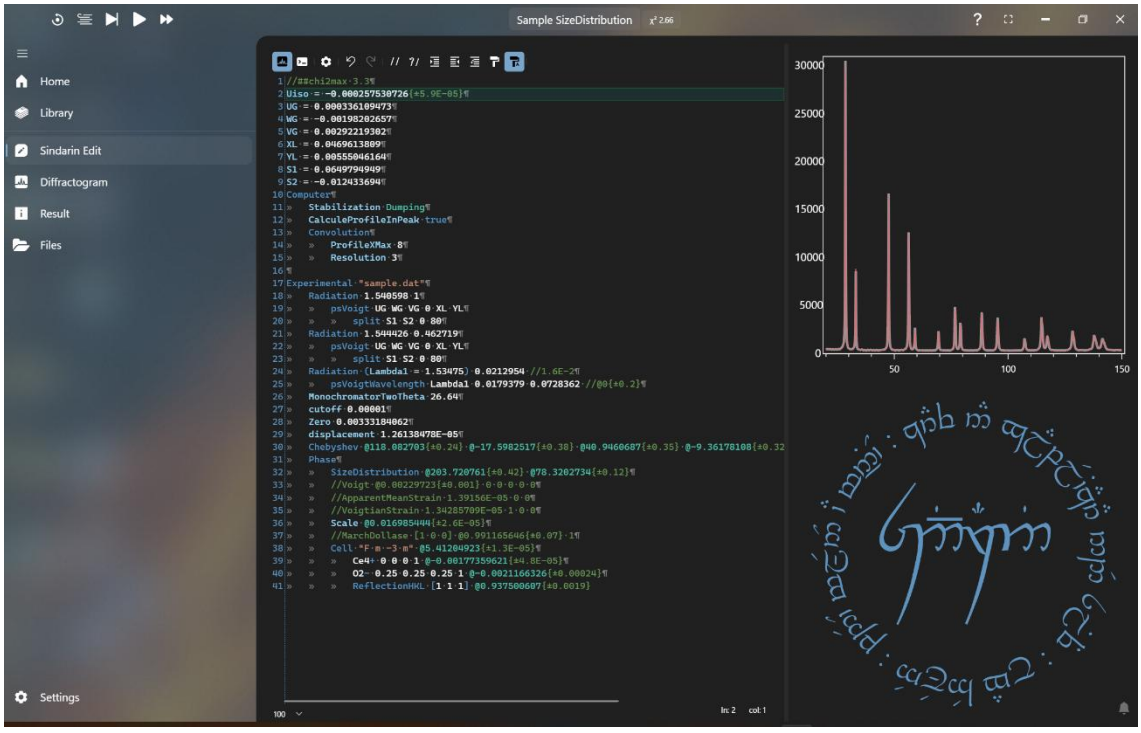


The extension will automatically activate when a text file with the ".sin" extension is opened, or if can be manually selected from the right side of the status bar (see image below)



2.3 NIMLOTH - A DEDICATE USER INTERFACE

Visit the Nimloth web page for more information and download: www.nimloth.app.

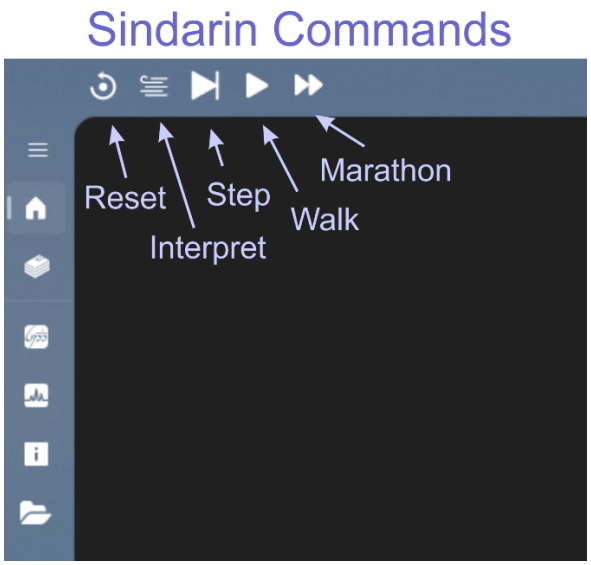


Nimloth is a program designed to simplify the use of Sindarin with several useful tools, making it the best choice for Sindarin users:

- Organize your diffraction works
- Keep all files and information about the diffraction experiment in one place
- Text editor
    - Text highlighting
    - Several tools to facilitate text typing
        - comment/uncomment
        - autocomplete
        - Sindarin text generators
- Results view
    - Detailed results
    - Export results to several file formats
    - Diffractogram functionality to enhance details visibility
- Feature-rich graphics for viewing, evaluating, correcting, and exporting
- Crystallographic tools
- Import crystallographic structure from CIF
    - Build a personal crystallographic database

The Nimloth is still in development, with a release coming soon, follow the official website.

The Sindarin commands can be accessed from the buttons in the top right corner of the application window. See the figure below.



2.4 SINDARIN FILE

A Sindarin file is a plain text file that contains the input information for the Sindarin Interpreter to do calculations. The text has to follow the Sindarin Syntax rules to be interpreted correctly. We recommend using the ".sin" extension for Sindarin files, because some user interfaces use this extension to activate some features or to sort files.

3 SINDARIN SYNTAX

You might be wondering why we use text as input for calculations in Sindarin. Or you might think that this is old-fashioned, boring or difficult. If so, I suggest you read the section on Why "Sindarin language"? I hope that after reading it and doing some tutorials, you will feel differently about this topic.

Sindarin is a language that aims to be simple, easy, flexible, intelligible and concise. It has some words and some rules that you need to follow to write correctly in Sindarin. You can find a comprehensive list of Sindarin words in the Sindarin Dictionary section. In the following sections, we will introduce the rules that form the Sindarin syntax.

3.1 DATA TYPE

Data is the term for specific information that can be used for calculations. It can either be involved in calculations directly or affect how objects behave. The interpreter recognizes six types of data: Integer, Decimal, String, Boolean, Enumerator and Array.

3.1.1 Integer

Integer data is a number that can represent positive and negative whole numbers, without decimal parts.

Examples:

```
1| 0
2| -1
3| 103
```

3.1.2 Decimal

The decimal data is a number, negative or positive, that can have fractional component, represented by digits after decimal point. The decimal data can alternatively be written in scientific notation, using the letter "e" (or "E") to indicate the exponential part of the number. Internally, the Sindarin uses a double-precision float number to handle decimal numbers to archive good precision in calculations.

Examples:

```
1| 3.14
2| 0.0015
3| -3E+5
4| 6.4e-7
```

The third and fourth examples use scientific notation, where -3E+5 means $-3x10^{5}$, and 6.4e-7 means $6.4x10^{-7}$.

Decimals are important for calculating crystal diffraction and have a special syntax in Sindarin. They can be used with math operations like addition (+), subtraction (-), multiplication (*), division (/), and functions like logarithm (log()), trigonometry (sin(), tan()), etc. Use parentheses to show the order of operations. See the Decimal Operators section for more details. The results of these operations and functions are decimals and can

be used wherever decimals are needed. A list of all functions and operations of decimals can be found in Decimal Operators (page 63).

Examples:

```
1| (1.2 + 3) * 2
2| 2 / log(21.5 - 3)
3| sin(3.141592)
```

3.1.3 String

A string is a sequence of characters written between quotes “”

Examples:

```
1| “datafile.xye”
2| “Something written between quotes!”
3| “I a -3 d”
```

3.1.4 Boolean

A boolean is a binary data type that can be either true or false.

Examples:

```
1| true
2| false
```

3.1.5 Enumerator

An enumeration is a data type that has a fixed set of values. These values are words made of alphanumeric characters only.

Examples:

The Stabilization property can have one of three values: Marquardt, Dumping or None:

```
1| Stabilization Marquardt
2| Stabilization Dumping
3| Stabilization None
```

3.1.6 Array

An array is a sequence of one or more values of the same data type. In Sindarin, arrays can be decimal or integer. The numbers are separated by white space and enclosed in brackets []. For example, an integer array of length 3 can represent a Miller index (h, k and l) of a crystallographic plane, and a decimal array of length 2 can represent a point (x and y).

Examples:

```
1| [2 0 -1]
2| [20.52 505.8]
```

The first example is an array of three integers: 2, 0 and -1. The second example is an array of two decimal numbers: 20.52 and 505.8.

3.2 OBJECTS

Objects store data and perform actions. The class of an object defines the types and number of data it stores and the actions it can do. Objects of different classes have different data and actions. One can have multiple objects of the same class, but each one has its own values for its data.

Experimental is an object class that one will use a lot to calculate diffraction. One data stored in an Experimental object is the file name that contains the experimental diffraction data, which is a string. One action of this object is to access the file and load the experimental data. Two objects of the Experimental class with different file names will load different experimental data. They belong to the same Experimental class, but they are different objects with different data. The object class can be thought of as the object type, similar to how there are data types (string, integer, etc.).

The syntax for declaring an object in Sindarin is: The first word of the line is the object class followed by zero or more data separated by white space. The Sindarin text below shows a generic declaration of an object:

```
1| Class Data1 Data2 Data3...
```

The object class defines the type, number and order of the data one writes. For example, the Experimental declaration is:

```
1| Experimental (String)filename[1+n]
```

The first word of the line is the Experimental object class and the data in this declaration is a string that must be written one or more times. The data name is filename and it is the name of a file that has experimental data to be loaded when the Interpret is run

To declare Experimental objects, use this syntax:

```
1| Experimental “data.xye”
2| Experimental “data2.xye” “data3.dat”
```

Here, two objects of the Experimental class are declared. The first one loads the experimental data from the file “data.xye” and the second one loads the experimental data from two files named “data2.xye” and “data3.xye”.

One class that is frequently used is the Cell. The syntax for declaring this class follows this pattern:

```
1| Cell (string)spaceGroup (decimal)latticeParameter[1,2,3,4,6]
```

The first word of the line is the Cell object class, the second word is a string named spaceGroup and the next word is a decimal named latticeParameter. The decimal can be

written one, two, three, four or six times. Here are two examples of Cell object declarations:

```
1| Cell “P 1” 6.2 4.3 3.4 89.7 121.2 80.6
2| Cell “I a -3 d” 5.232
```

The first Cell object has a string data that corresponds to P 1 Space Group, number 1 from the International Union of Crystallography (IUCr) table, and six decimal data that represent the six lattice parameters of triclinic lattice: the edges a, b and c and the angles alpha, beta and gamma. The second Cell object has a string data that corresponds to I a -3 d Space Group, number 230 from the IUCr table, and one decimal data that is the lattice parameter for Cubic lattice: the edge a. The number of decimal parameters of the Cell object is determined by the first string data, as it must match the number of lattice parameters for the unit cell.

3.2.1 Parameters

Data written on the same line as the object declaration are called parameters. The parameters must be separated by white space.

3.2.2 Object Parenting

Children are objects that belong to another object, which is called their parent. The class and number of children depend on the class of the parent. The declarative syntax is the same for any object, but the children are declared inside the parent’s block. See the Object’s Block section for more details. An object that has no parent is called a root object.

3.2.3 Indentation

Indents are spaces at the beginning of each line. In Sindarin, the default character for indentation is the tab, but the whitespace character can also be used. However, it is not possible to mix these two characters within an object block (see the Object’s Block section). The indentation level is determined by the number of space characters at the start of the line, and they can be compared. For example, if the first line has two tabs and the second line has one tab, then the first line has a greater indentation level than the second line.

3.2.4 Object’s Block

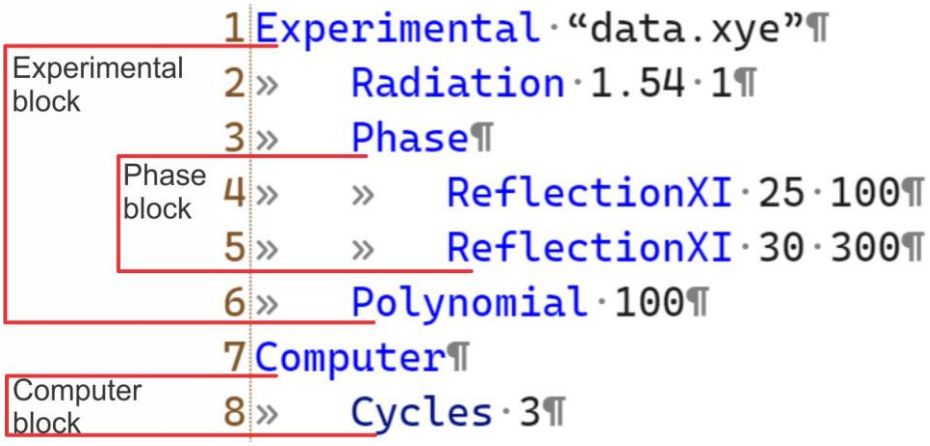
An object’s block consists of all the lines below the object’s declaration that have a higher indentation level (see the Indentationsection) than the declaration line. Consider the following Sindarin text:

```
1| Experimental “data.xye”
2|       Radiation 1.54 1
3|       Phase
4|             ReflectionXI 25 100
5|             ReflectionXI 30 300
6|       Polynomial 100
7| Calculation
8|       Cycles 3
```

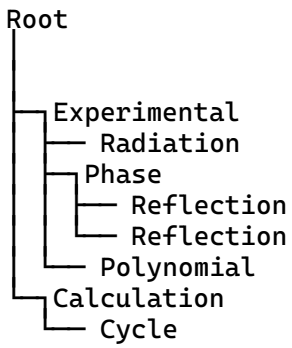
The declaration line of the Experimental object (line 1) has no space at the beginning and all the following lines that start with one or more tabs are inside the Experimental object’s block, so lines 2 to 6 belong to the Experimental object.

Experimental objects are root objects. The declaration line of the Radiation object (line 2) starts with one tab, so this Radiation object is a child of the Experimental object. For the same reason, the Phase object (line 3) is also a child of the Experimental object. The two declaration lines of the ReflectionXI objects (lines 4 and 5) start with two tabs, so they are inside the block of the Phase object and both are children of the Phase object. The next line (line 6) with the declaration of the Polynomial object starts with only one tab and therefore this line is not inside the Phase object's block but is inside the Experimental object's block. So, the Experimental object has three children: one Radiation object, one Phase object and one Polynomial object. And the Phase object has two children: two ReflectionXI objects.

The line that declares the Calculation object (line 7) has no space at the beginning, so this line is outside the Experimental block and the Calculation object is a root object. The last line (line 8) has a tab at the start, so this line is inside the Calculation object's block and is a property (Cycles is a property, not an object) of the Calculation object. The following image shows graphically the block of each object:



The hierarchy structure of the object from the example can be represented as follow:



Properties and variables declarations cannot have blocks.

3.2.5 Properties

Data written inside the object's block (see the Object's Block section) are called properties. The syntax for assigning properties is similar to the syntax for declaring objects,

except that the first word of the line is the property name (instead of the object class) followed by the property value. So, the syntax for assigning properties is:

```
1| PropertyName Data
```

The syntax for assigning the Displacement property to the Experimental object is:

```
1| Experimental “data.xye”
2|       Displacement 0.00578
```

3.2.6 Implicit Objects and default values of parameters and properties

Some objects or properties are created or assigned a value even if they are not declared in the text. Calculation is an example of an object that is always created by the interpreter with default values for its properties. Cutoff is a property of the Experimental object that always has a default value if it is not declared in the text. To see the default values of each property and parameter, consult the Sindarin dictionary entries for each object class.

3.2.7 Some details of children

Some classes can be declared only once per parent, others can be declared as many times as the user wants, and others cannot be declared together with other classes. Some classes cannot be children of any other objects; these are root objects, like the Experimental class. Other object classes can only be children of a specific object class, like Cell, which can only be a child of a Phase object. Some other classes can be children of different parents, like the profile classes, which can be children of Radiation, Experimental, Phase, ReflectionXI, ReflectionHKL or PeakXI. This information for each class is available in the specific entries in the Sindarin Dictionary.

3.3 VARIABLES

A variable is a special kind of object that stores a decimal value and can be used in any parameter or property of decimal type of any object. A variable has two parts: an identifier and a decimal value. An identifier is a sequence of characters that must be unique in the Sindarin text, no other variable can have the same identifier and it cannot match any reserved Sindarin word such as object class, property name, constant and so on. The allowed characters for an identifier are alphanumeric and “_”, but the first character cannot be a number. The basic syntax for declaring a variable is:

```
1| variableIdentifier = decimal
```

The character “=” is used to assign a decimal value to an identifier. When the interpreter reads this pattern, a variable is created in the memory. The decimal value can be a number or an expression.

Examples:

```
1| Var_1 = 2.0
2| IDENT = 4.2 + 3.1 * 0.25
```

The identifier of a declared variable can be used like any other decimal value, directly in properties or parameters or inside expressions. To use an identifier of a declared variable is the same as using its decimal value and it can be used anywhere that a decimal value is expected.

Examples:

```
1| Displacement Var_1
2| Cell “I a -3 d” IDENT
3| Var2_ = 2 * 0.4 + Var_1 + 1
```

In the first example (line 1), the variable Var_1 is directly assigned to the Displacement property. In the second example (line 2), the variable IDENT is assigned to a parameter of the Cell object. And in the third example (line 3), Var_1 is used within an expression to declare the variable Var2_. In these examples, it is assumed that two variables with identifiers Var_1 and IDENT are declared elsewhere in the Sindarin text. A variable can be declared in a single line, inside the block of an object or directly in an assignment property, parameter or expression. In the latter case, the variable must be enclosed in parentheses “()”.

Examples

```
1| Displacement (Var_1 = 2.0)
2| Cell “I a -3 d” 4.975
3|       Var_1 = 2.0
4| Var2_ = 2 * 0.4 + (Var_1 = 2.0) + 1
```

In the first example (line 1), the variable Var_1 is declared in the assignment of the Displacement property. In the second example (line 3), the variable Var_1 is declared within the block of a Cell object. And in the third example (line 4), Var_1 is declared inside the declaration of the variable Var2_.

When a value of a variable is written as an expression, after some Sindarin interpretation command is executed, the interpreter will write the calculated value of the variable after the identifier and between braces “{}”. For example, the variable Var2_, after an interpretation command, will appear in the text like this:

```
1| Var2_{3.8} = 2 * 0.4 + (Var_1 = 2.0) + 1
```

This means that the value of Var2_ is 3.8, which is the result of multiplying 2 by 0.4 and adding it to the value of Var_1, which is 2.0, and to 1

3.3.1 Fitting

Variables can be set to adjust their value in an optimization routine (see the Optimization commands section) to make the theoretical data closer to the experimental data. To mark a variable for adjustment, simply write the “@” character before the decimal number in the variable declaration, as shown in the following example:

```
1| Var_1 = @2.0
```

This means that the value of Var_1 can be changed by the optimization routine to find the best fit for the data.

The variable Var_1 is marked for adjustment and its value can be modified to reach the minimum residual value of the data when the optimization routine runs. After the optimization routine, the new values of all the variables marked for adjustment are written by the interpreter in the Sindarin text, replacing the old values. The standard deviation calculated for each adjusted variable value is also written by the interpreter after the numeric value, enclosed in braces "{}", like this:

```
1| Var_1 = @1.999998{±0.00005}
```

This means that the value of Var_1 after the optimization is 1.999998, with a standard deviation of 0.00005. You can use this information to evaluate the quality and uncertainty of your fit.

3.3.2 Restraints

It is possible to declare the minimum and maximum value that a variable can reach in an optimization routine. If the new value of the variable is outside the limits, the algorithm will return to the previous value and try to change it again in the next optimization cycle. If the user writes the value of a variable outside the declared limits, the interpreter will return an error and the calculation will not be performed. Minimum and maximum are properties of the variable object, but the syntax for assigning them is different from other object properties. To declare the minimum and maximum limits of a variable, just write the reserved words "min" or "max" and the decimal value at the end of the variable declaration. Some examples of declaring constraints are:

```
1| Var_1 = 2 min 1.1
2| Var_2 = 2 max 3.5
3| Var_3 = 2 min 1.1 max 3.5
4| Var_4 = 2 max 3.5 min 1.1
```

In these examples, the variable Var_1 has a minimum value of 1.1, Var_2 has a maximum value of 3.5 and Var_3 has both a minimum value of 1.1 and a maximum value of 3.5. Var_4 is identical to Var_3; the order of min and max is not important.

Some parameters and properties have internal limits. For example, the two parameters of the ReflectionXI object are the x-position in the diffractogram and the intensity. Both of them must be greater than zero, and the x-position also has a maximum limit: it must be less than 180. If a parameter or property has a value outside the internal bounds, the interpreter will do the same as for a constrained variable: it will return the variable (or variables) of that parameter or property to the previous value if it has been changed by the optimization routine, or it will return an error if the value was written by the user.

3.3.3 Implicit Variables

It is possible to declare a variable without an identifier; these are the implicit variables. You just need to write "@" before a decimal value. This way, the decimal value is transformed into an implicit variable and marked for adjustment in the optimization

routine. This is very useful for optimizing the value of a parameter or property, but its value is not useful elsewhere.

The implicit variable cannot have constraints. Some examples of implicit variables are:

```
1| Displacement @2.0
2| Cell “I a -3 d” @4.975
```

In these examples, the values of the Displacement property (@2.0) and the numeric parameter of the Cell (@4.975) are implicit variables that can be adjusted by the optimization routine.

Examples:

```
1| Displacement (1.9 + @0.1)
2| Cell “I a -3 d” (@4.970 * 1.001)
3| Var2_ = 2 * 0.4 + @2.0 + 1
```

In the first and second examples, @0.1 (line 1) and @4.970 (line 2) are implicit variables declared within the expressions of assignment of the Displacement property and a parameter of the Cell object, respectively. In the third example (line 3), @2.0 is an implicit variable declared within the expression of declaration of the variable Var_1.

3.3.4 Internal Variables

The Sindarin framework exposes several internal variables that can be utilized within decimal data in appropriate contexts. Unlike user-defined variables, internal variables are not refinable and may assume different values depending on the context. For instance, the internal variable xPeak may take on varied values for distinct reflections. Should an internal variable be employed within decimal data where it is prohibited, the interpreter returns an error. For example, the AbsorptionFunction cannot depend on the internal variable IPeak. In cases where an internal variable is used in permissible decimal data but its value cannot be deduced, the internal variable yields a NaN value. As an illustration, the Zero property may depend on dReflection; however, if the diffraction model does not utilize the Radiation object, dReflection cannot be calculated, resulting in Zero being NaN.

A complete list of the internal variables can be found in Internal variables at page 65.

3.4 Comments

Comments are parts of the text that are not interpreted by the interpreter and do not affect the calculation. They are user notes within the Sindarin text. Comments start with a double slash “//” and end at the end of the line, such as:

```
1| // some notes
```

Example:

```
2| // This is a comment
3| Experimental “data.xye” // This is another comment
```

```
4|       Displacement 0.005 // This is a comment after a property
   assignment
```

These comments are ignored by the interpreter and do not change the value of the Displacement property. You can use comments to document your Sindarin text, explain your choices, or remind yourself of something. Comments can make your Sindarin text more readable and understandable.

There is a special type of comment that the Interpreter uses to write some information about variables, such as calculated value and standard deviation, when applicable. These comments are only possible in specific places in the text, after identifiers or values of implicit variables. The user should not use this type of comment. They are enclosed in braces "{}". Some examples:

```
1| Displacement @0.0879371124{±0.00029}
2| Var_1{7.2} = 2.0 + 5.2
3| Var_2{±0.00029} = @2.0
```

The standard deviation (SD) calculated after the optimization algorithm have shown inside braces and start with "±".

3.5 Case Insensitive

The Sindarin language is fully case insensitive, which means that upper and lower case letters do not distinguish one word from another. So experimental, Experimental, EXPERIMENTAL and ExPeRiMeNtAl are the same word for the Interpreter. The same rule applies to string data, where "I a -3 d" is equivalent to "I A -3 D". The only exception is with strings that represent file names in Linux operating system, where the file name is case sensitive. Therefore, it is necessary to pay attention to the letter case of file names. For example, a file named Data.xye must have the string "Data.xye" because the string "data.xye" (or other string with different letter case) will return an error that the file was not found. For the other operating systems, Windows and macOS, it doesn't matter, because the same file can be found with strings like "DaTa.XyE" or any variety of letter case.

3.6 In what order should it be written?

The order of declaration of objects does not matter. The Interpreter will read the full text and instantiate all objects and variables and only then calculate each object in a specific internal order. Therefore, no matter which object, variable or property is written before or after in the text. The only logic that matters is the object block that impose the parentage relationship of objects.

Consider the examples 1 and 2, after executing the interpret command, they will return the same results.

Example 1:

```
1| Var1 = 2
2| Var2 = 1
3| DataSimulation
4|       Polynomial Var1 Var2
```

```
5|       Interval -10 10 0.1 “dataTest.xye”
6| Data “dataTest.xye”
7|       Polynomial Var1 Var2
```

Example 2:

```
1| Data “dataTest.xye”
2|       Polynomial Var1 Var2
3| Var2 = 1
4| DataSimulation
5|       Interval -10 10 0.1 “dataTest.xye”
6|       Polynomial Var1 Var2
7| Var1 = 2
```

3.7 CUSTOM USER FUNCTIONS

In Sindarin, there are objects and properties dedicated to customizing equations to describe peak shift, peak absorption, background function, and peak profile. These objects or properties are simple, each having just a decimal parameter. However, their great capability lies in the flexibility of decimal data with operators, functions, internal constants and especially the internal variables that Sindarin exposes to the user. For more information on this decimal flexibility, refer to the chapters: Decimal Operators and Functions (page 63), Decimal Constants (page 65) and Internal variables (page 65).

To correct peak positions, the Zero property of Experimental object can be used. Its value is directly added to the peak positions without other internal functions, as shown Equation 1 (page 34). To exemplify the use of Zero property as a custom user function to correct the peak position, the following Sindarin text uses the Zero property to correct the displacement of the sample relative to the goniometer center:

```
1|       disp = -0.00005
2|       Zero (deg(-2 * disp * cos(rad(xPeak / 2))))
```

The equation of decimal data in line 2 of the example uses the internal variable xPeak which is the value of the peak position calculated from the interplanar distance. This equation is the same as that used to calculate the displacement, and the text is similar to the next line:

```
1|       Displacement disp
```

The AbsorptionFunction property of Experimental object is dedicated to customizing functions to correct peak intensity. Its decimal value is directly multiplied by the peak intensity, as shown in Equation 2 (page 34). In the next example, the equation of the decimal value of this property is used to calculate the Lorenz factor and also uses the internal variable xPeak:

```
1| AbsortionFunction (1.0 + cos(rad(xPeak)) ^ 2 / 2)
```

For background description, the Function object can used with the internal variable x. The following example describes a polynomial function of order 3:

```
1| a0 = 4
2| a1 = 3
3| a2 = 2
4| a3 = 1
5| Function (a0 + a1 * x + a2 * x ^ 2 + a3 * x ^ 3)
```

The results are identical to using the following Polynomial object:

```
1| Polynomial a0 a1 a2 a3
```

The Function is calculated outside of phase and cell objects, so it is not possible to use some internal variables like xPeak and dReflection.

To describe peak profiles, there are two objects: ProfileFunction and ProfileFunctionFT. The ProfileFunction is used to describe the profile in the same space as the experimental data and usually uses two internal variables to describe the peak profile: x and xpeak. The following example describes a Gaussian function with an FWHM of 0.1 °2θ using ProfileFunction object:

```
1|       h = 0.1
2|       ProfileFunction ((0.9394 / h) * e ^ (-2.7725 * ((x - xpeak)
   / h) ^ 2))
```

That it is identical to using Gauss object with a parameter of 0.1:

```
1|       Gauss 0.1
```

Alternatively, it is possible to use the declarative syntax of the ProfieFucntion object with two parameters, where the second parameter is used to describe the internal variable fwhm:

```
1| ProfileFunction ((0.9394 / fwhm) * e ^ (-2.7725 * ((x - xpeak) /
   fwhm) ^ 2)) 0.1
```

The ProfileFunction can have a Split object as a child to introduce asymmetry effects in the profile.

It is possible to use the ProfileFunctionFT to describe the profile directly in Fourier space, using the internal variable xl instead of x. The next example the same gaussian function is described directly in Fourier space.

```
1| lamb = 1.540598
2| h = 0.1
3| hxl = 4 * cos(rad(xpeak / 2)) * sin(rad(h / 4)) / lamb
4| ProfileFunctionFT (e ^ (-3.5597 * (hxl * xl) ^ 2))
```

The first line declares a variable with the value of the wavelength of the radiation, which is necessary to computer Fourier space, and it is used to calculation the FWHM of the Gaussian in Fourier space units, stored in variable hxl (line 3).

The ProfileFunctionFT object has an alternative declarative syntax with two parameters, where the second parameter represents the imaginary portion of the complex number of the profile in Fourier space.

The use of equations or function in decimal parameters or properties is not restricted to specialized objects or properties for user customization, it can be applied to any decimal data. For example, the Gauss object has a declarative syntax with one parameter, which is the FWHM of the profile. This parameter's value is not used in any internal function but is directly applied as the FWHM in the Gaussian function. Therefore, to describe the modified Caglioti dependence of the FWHM on the peak position (see The pseudo-Voigt function section, page 38), the following can be used:

```
1| Gauss (sqrt(gu * tan(rad(xpeak / 2)) ^ 2 + gv * tan(rad(xpeak /
   2)) + gw + gz / cos(rad(xpeak / 2)) ^ 2))
```

The results will be the same as using the declarative syntax of Gauss object with four parameters:

```
1| Gauss gu gv gw gz
```

4 SPECIFIC INFORMATION

This section is dedicated to presenting some information and details about the various aspects of Sindarin, especially related to the calculations.

4.1 THE SINDARIN INTERPRETER

The Sindarin Interpreter is a part of the Sindarin program that receives the text from a Sindarin file and reads the source text, instantiates the appropriate objects and data (see Objects) in the memory, calculates the theoretical diffractogram, executes the optimization routine, and returns the results or related errors.

4.2 SINDARIN COMMANDS

The Sindarin is executed by some commands: Interpret, Step, Walk, Marathon, Result and Reset. To know how to execute these commands, see the User Interface section.

4.2.1 Interpret command

The interpret command performs the basic operations in Sindarin, the interpreter reads the text, instantiates all objects and calculates each of them in a specific internal order to calculate the theoretical diffractogram and write the value of each calculated variable (equations) in the text.

4.2.2 Optimization commands

The Step, Walk and Marathon commands are the optimization commands, they do the same as the interpret command and, in addition, perform the optimization routine (see the Optimization Routine section) and finally write the new values of the refined variables in the text. The difference between these three commands is the number of cycles of the optimization routine, Step is only one cycle, Walk is the number defined by the Cycles property of the Calculation object (the default is 3) and Marathon executes cycles until a convergence or divergence criterion is reached.

4.2.3 Results command

The results command can be executed after the interpret command or any optimization command to get the results of the calculation.

4.2.4 Reset command

The Reset command cleans all memory, all objects and variables are erased. All objects and variables will be instantiated and assigned in the next interpret command. This command has effect only on the Nimloth interface.

4.3 OPTIMIZATION ROUTINE

Sindarin uses the nonlinear least squares (nlls) approach to approximate the theoretical diffractogram to the experimental diffractogram by optimizing the values of the parameters of the mathematical model. The Gauss–Newton algorithm is used to estimate the new parameter values and the Marquardt modification (Marquardt, 1963) to stabilize the refinement process when it tends to diverge.

The user controls the choice of which parameters are optimized by marking/unmarking the variables to refine in the text (see the section 3.3.1).

Sindarin automatically selects the optimal cycle when the optimization routine diverges or encounters issues, and returns the results of this cycle at the end of refinement. If divergence occurs in the first cycle, all refinable variables revert to their initial values. This feature enhances the reliability and stability of Sindarin, ensuring that a set of poorly static or error-generating variables are never returned.

Furthermore, Sindarin's optimization routine possesses a unique characteristic: when variables exhibit high correlation within the covariance matrix, they are automatically addressed to prevent divergence while they continue to be refined. All other variables proceed with optimization, and only thereafter, the highly correlated variables are refined individually. This approach eliminates the need for manual alternation in optimizing these highly correlated variables.

4.4 Errors returned by the Interpreter

When the Interpreter cannot read the Sindarin text properly or cannot evaluate the theoretical diffractogram calculation correctly, it will return one or more errors with information about the impediment of the calculations. An error can occur for several reasons, some examples: syntax error, wrong number of parameters, unrecognized words, value of a parameter or property is outside the internal constraints, value of a variable is outside the constraints, some data calculated internally is wrong (like negative FWHM of a specific peak). The returned error has five parts of information that help the user to correct the text:

1. Error Code: A unique numeric code that corresponds to the type of error
2. Description: A brief description of the type of error
3. Object: The object that generated the error, if applicable.
4. Line: Text line of the object or property that generated the error, if applicable.
5. Message: More details about the error, e.g., the file name that cannot be found.

4.5 Experimental Data File

The experimental data file used in Sindarin is a plain text file with a series of experimental points. The data in the file can be of two different arrangements:

1) Each point is written on a line and can have three numbers: x, the position on the horizontal axis, y, the position on the vertical axis and e, the experimental error of measuring that point. On the line, they are separated by white space, comma or tab. Measurement error information is optional, so lines can only have two numbers. The points must be in ascending order and the step in x, the difference between the sequential x's, does not need to be constant for all points.

   Example:

```
1| 4.00  1023.45     3.52
2| 4.02  1020.91     6.1
3| 4.04  1019.38     4.93
```

```
4| 4.06  1021.44     2.04
```

2) The first line of the text file has three decimal data: x_i, x_{step} and x_f, where xi is the horizontal coordinate of the first point, x_{step} is the value of the difference between each sequential x and x_f is the horizontal coordinate of the last point in the series. The other lines must have only one decimal value, which is the value of the vertical coordinate (y) of each point in ascending order of x. Therefore, it is not possible to have experimental error information and the step in x is constant for all points. The number of lines of y must match the number of x generated from the first line.

   Example:

```
1| 4.00  0.02  4.06
2| 1023.45
3| 1020.91
4| 1019.38
5| 1021.44
```

The two examples above have the same data written in two different arrangements. The decimal data can also be written in scientific notation, so the y coordinate of the first point in the examples above can be written as 1.02345E3.

It is possible to write comments in data files with the same rules as comments in Sindarin texts (see the Comments section).

4.6 AN OVERVIEW OF PEAK DIFFRACTION CALCULATION

The diffraction peaks of the theoretical diffractogram are calculated by the PeakXI objects. Reflection objects (ReflectionHKL and ReflectionXI) always create PeakXI objects to perform the calculations. The number of PeakXI objects created by each reflection object is equal to the number of declared Radiation objects. If no Radiation object is declared, ReflectionHKL object cannot be used and ReflectionXI will only have one PeakXI object. The position of each PeakXI of a reflection object is calculated by the Bragg equation using the interplanar distance (d) of the reflection and the wavelength (λ) of each Radiation object.

Reflection objects are created by the user by declaring them directly in the Sindarin text or a list of ReflectionHKL objects is created automatically from the crystallographic information of the Cell object (see more details in section 4.7).

The diffraction peaks must have three pieces of information to be calculated: position (x_{ref}), intensity (I_{ref}) and profile. The next sections are devoted to explaining the various ways in which this information is provided to calculate the theoretical diffractogram. Sindarin was designed to be flexible, there are a lot of possibilities to model the theoretical diffractogram, this allows the user to obtain the desired information from the calculation.

4.6.1 General equations for calculating the position and intensity of reflections

The position of reflections in the diffractogram is calculated by the ReflectionHKL, ReflectionXI or PeakXI objects. The following equation shows the calculation of the position of reflections:

Equation 1 $$x_{Ref} = Bragg(d) + x_{obj} + Zero + Disp(x) + Shift$$

The *Bragg(d)* term is the Bragg function to calculate the diffraction angle from the interplanar distance (d) of the ReflectionHKL object and the x_{obj} term is the diffraction angle directly from the parameter of the ReflectionXI or PeakXI objects. The default value of all terms in Equation 1 is 0. For the ReflectionHKL object, the *Bragg(d)* is different from zero and *x* is zero, for the ReflectionXI or PeakXI objects, the *x* is different from zero and *Bragg(d)* is zero. The decimal terms *Zero* and *Displ(x)* are position corrections, properties of the Experimental object, and the *Shift* term is an arbitrary position correction, a property of the ReflectionHKL object. The calculation of Specimen Displacement depends on the diffraction geometry.

The intensity of reflections is calculated by the ReflectionHKL, ReflectionXI or PeakXI objects. The following equation shows the general equation for the intensity calculation:

Equation 2 $$I_{ref} = S|F(HKL)|^2 I_{obj} M_{HKL} L_P(x) P_O(HKL) Abs(x) I_{rad}$$

The $|F(HKL)|^2$ term is the structure factor calculated for each reflection with the crystallographic information of the Cell object when Atom objects are declared. The *M* term is the reflection multiplicity which depends on the Miller indices of the reflection and the Laue symmetry. The I_{obj} term is the intensity declared by the user as a parameter of the ReflectionHKL, ReflectionXI or PeakXI objects. The *S* term is the phase scale factor, a property of the Phase object. The $L_P(x)$ term contains the polarization and Lorentz factors, including the monochromator polarization correction. The $P_O(HKL)$ term is the preferred orientation correction when a preferred orientation object is declared. The *Abs(x)* term is the absorption correction term. The I_{rad} term is the relative intensity of the radiation line. The default value of all terms in Equation 2 is 1.

The next sections will use the terms of Equation 1 and Equation 2 to explain details about the calculation of the diffraction peak.

4.7 Refinement with Different Structure Details

The calculation of the theoretical diffractogram can be performed by providing different details about the crystal structure: Complete structure information (Rietveld Refinement), Only lattice parameter information (Pawley method) or No structure information. Th last two approach is also called Whole Powder Pattern Fitting (WPPF).

4.7.1 Complete structure information (Rietveld Refinement)

Classical Rietveld refinement is performed when the diffraction model has complete crystallographic information, with cell geometry and all atoms in the cell. In this case, a list of ReflectionHKL objects is created internally by the Cell object, which calculates the interplanar distance (d) for all ReflectionHKL using the lattice parameters. The structure factor ($|F(HKL)|^2$) is calculated using information about all Atoms objects and the multiplicity (M_{HKL}) is determined using Miller indices and the Laue symmetry. Therefore, the x_{ref} is calculated using the *Bragg(d)* function and the x_{obj} term is zero in Equation 1, and the I_{ref} is calculated using $|F(HKL)|^2$ and M_{HKL} and the I_{obj} term is 1 by default in Equation 2.

The crystal structure can be refined to obtain accurate crystallographic information using Rietveld refinement. In general, this refinement uses fewer refinable variables among the three different structure detailing models and has no problem with overlapping peaks.

Sindarin text example:

```
1|       Phase
2|             Scale 0.001
3|             Cell "F D -3 M" 5.432117
4|                   Si "Si1" 0 0 0 1 0.05
```

4.7.2 Only lattice parameter information (Pawley method)

It is possible to refine a theoretical diffraction model without atomic information, only with unit cell geometry; this is the Pawley method. The reflection position (x_{ref}) is calculated in the same way as in Rietveld Refinement, but the intensity of each reflection needs an arbitrary refinable scale factor. Instead of writing a list of Atom objects in the Cell object block, a list of ReflectionHKL objects with two parameters (Miller index and intensity scale factor (I)) should be written. The I_{ref} is calculated with I_{obj} and M_{HKL}; $|F(HKL)|^2$ term has the default value 1 in Equation 2. The S term can't be refined simultaneously with all ReflectionHKL intensity parameters because there is a high correlation and it makes the optimization routine unstable.

In this refinement, the cell geometry can be fitted and in general has more refinable variables than Rietveld refinement and less than in the refinement without any structure information. This method is useful for obtaining information from profiles and cell geometry or when intensities are used to solve the structure.

Different overlapping reflections, with the same position (x_{ref}) or very close, can't have their intensity individually fitted, as these intensities have a high correlation, which would make the optimization routine unstable.

Sindarin text example:

```
1|       Phase
2|             Cell "F D -3 M" 5.432117
3|                   ReflectionHKL [1 1 1] 1000
4|                   ReflectionHKL [2 0 2] 880
1|                   ReflectionHKL [1 1 3] 530
2|                   ReflectionHKL [4 0 0] 140
```

4.7.3 No structure information

It is still possible to have the refinement of the theoretical diffractogram without any crystal cell information. In this case, the cell object is not declared; it is necessary to write a list of ReflectionXI objects in the Phase object block with two refinable parameters: position (x_{obj}) and an intensity scale factor (I_{obj}). The x_{ref} is calculated with x_{obj} ($Bragg(d) = 0$) and the I_{ref} is calculated using I_{obj} ($|F(HKL)|^2 = 1$ and $M_{HKL} = 1$ by default).

No structure information is refined and, in general, this method has more refinable variables. This method is useful when only peak profile information is of interest or when the peak position information is used to find the lattice parameters and the space group.

The same observations about the *S* term and overlapping reflections of the Pawley method are valid for this refinement.

Example:

```
1|       Phase
2|             ReflectionXI 28.43 1000
3|             ReflectionXI 47.28 880
4|             ReflectionXI 56.10 530
5|             ReflectionXI 69.11 140
```

4.8 PROFILES

The profile is the shape that the peaks take on the diffractogram. A correct analysis of the diffractogram peak profiles can provide important information about the sample, such as crystal size and its mean and distribution, mean crystal shape, defects in terms of microstrain or more specific ones, such as dislocation and stacking fault. Many approaches have been proposed to fit the profile of diffraction peaks and Sindarin can perform several of them, allowing the user to choose the most suitable approach for their experimental data and samples. Some aspects of profiles calculation in Sindarin are explained in the next sections.

4.8.1 Profile level

PeakXI objects are used to calculate the profiles, which can be declared at different levels: Radiation, Experimental, Phase, Reflection/Peak. Each profile object will be assigned to the corresponding PeakXI objects based on the parent/children relationship. So:

- Experimental: a profile object declared on an Experimental block will be computed on all peaks of all its child Phases.
- Phase: a profile object declared on a Phase block will be computed only on its peaks.
- Reflections: if profile objects are declared in blocks of ReflectionXI or ReflectionHKL objects, they will be computed only on peaks derived from them.
- Peaks: if a profile object is declared in a PeakXI block, it will be computed only for that peak.
- Radiation: profile objects declared on a Radiation block will be computed on corresponding peaks of all phases that are children of the Experimental object.

4.8.2 Convolution of peak profiles

The profiles are calculated at the PeakXI object. If there are two or more profile objects, they are convoluted. The convolution can be calculated by three different mathematical methods: analytical convolution, FFT convolution and special numerical convolution.

- Analytical convolution

Analytical convolution is possible when there is a known function that represents the convolution of the initial functions. It is the fastest convolution method. Sindarin performs analytical convolutions for Gauss, Lorentz, Voigt and psVoigt functions. In Sindarin, these functions and others derived from them are called Voigtian functions.

- Fast Fourier Transform (FFT) Convolution

The convolution in the reciprocal FFT space is performed quickly because it is a simple operation of multiplying each point. The convoluted data in the reciprocal FFT space is transformed to the direct space by the Inverse Fast Fourier Transform (IFFT). This procedure is very fast, especially when the functions can be evaluated directly in the reciprocal FFT space.

- Special numeric convolutions

The conventional numerical convolution is very computationally intensive and time-consuming, but some functions can be convoluted with a special numerical method that is very fast, comparable to the FFT convolution. In Sindarin, the following functions and their derivatives can be convoluted by special numerical methods: Box, InverseProfile, Exponential and Circle.

- Microstructure Voigtians

The profile objects VoigtianSize and VoigtianStrain undergo a special process before they can be convoluted with other profile functions. If there are two or more VoigtianSize objects, they are transformed into a single VoigtianSize object with a weighted average size property related to that specific crystallographic direction of reflection. The same is done with VoigtianStrain objects. This operation will return an average apparent microstructure, which is useful for modeling a complex anisotropic microstructure, for example, when the sample has a mixture of different crystal shapes. It is important to take into account that this approach is only meaningful for the average apparent microstructure, but not for the real value of the microstructure.

4.8.3 Fundamental Parameters Approach (FPA)

Sindarin can compute the instrumental profile by the Fundamental Parameters Approach (FPA) (Cheary & Coelho, 1992). The principle is to model the profile of the diffracted peak by the optical and instrumental effects of the radiation in the diffraction experiment. This is accomplished by convoluting the original profile of the emitted radiation by the source with functions that describe these effects. FPA calculation can achieve a very accurate instrumental line profile of the diffracted peaks. The following corrections are available:

- Target Width: The width of the x-ray beam at the radiation source. See TargetWidth, page 106.
- Axial Divergence: The effect generated by the difference in the radiation path due to the length of the incident beam on the sample when viewed perpendicularly to the diffractometer axis (or parallel to the rotational axis of the goniometer). Currently, only the simple correction function is available, as shown in (Cheary & Coelho, 1992). More complex functions for correction of axial divergence will be included to describe more accurate instrumental settings. See AxialDivergence, page 68.
- Equatorial Divergence: The effect generated by the difference in radiation path due to the height of the incident beam on the sample when using a flat specimen. See EquatorialDivergence, page 82.

- Receiving Slit Width: The effect generated by the width of the receiving slit. See ReceivingSlitWidth, page 99.
- Transparency: When the sample has a small linear attenuation coefficient and/or small thickness, the diffracted radiation can come from below the center of the diffraction circle of the diffractometer. Transparency108
- Tilt: When the specimen surface is tilted with respect to the diffractometer axis. See Tilt, page 107.
- Defocusing: When the receiving slit is not on the focusing circle. See Defocusing, page 78.

The Sindarin also exposes the main base functions of FPA corrections: Box, also known as the top hat function (page 69), Circle (page 73), Exponential (page 84) and InverseProfile (page 90).

In addition, Sindarin has Voigtian functions with FWHM described in units of wavelength for use with Radiation objects to describe the profile of each discrete emission line radiation. The FWHM profile of these functions follows the optical dependence with the diffraction angle (Section Radiation modeling, page 44).

4.8.4 Whole powder pattern modelling (WPPM)

The Whole Powder Pattern Modeling (WPPM) is an approach to model the peak profiles of the diffractogram according to a physical model of the sample, without using biased analytical functions (Scardi & Leoni, 2002). Some characteristics of the sample can be modeled by this method:

- Crystal size distribution
- Surface relaxation (not implemented yet)
- Dislocation (not implemented yet)
- Stacking Fault (not implemented yet)

Although the WPPM method was initially developed to be used without crystallographic information, it can also be used for peak profile calculation in Rietveld refinement, combining the structural and microstructural analysis of the two methods.

4.8.5 The pseudo-Voigt function

Among the many functions that have been used to fit the diffraction peak profile, the pseudo-Voigt (psVoigt) has received great attention in the last decades. psVoigt can adequately describe the experimental diffraction peak profile in most cases, it is light to be computed, it has analytical convolution, it can be calculated directly in Fourier space and it has some approach to describe peak asymmetry. These characteristics make this function widely used, and it is implemented in many software for diffraction calculation. In Sindarin, it is a base function to describe the peak profile.

The psVoigt function (psV) is a linear combination of: Lorentzian (L(x)), and Gaussian (G(x)) functions.

Equation 3

$$psV = \eta L(x) + (1 - \eta)G(x)$$

The parameter η represents the linear combination coefficient for mixing the two functions, with its value ranging from 0 to 1.

The Lorentz function normalized by area is:

Equation 4

$$L(x) = \frac{2}{\pi} \frac{\frac{1}{H_L}}{1+4\left(\frac{x-x_{hkl}}{H_L}\right)^2}$$

Where x_{HKL} is the Bragg position of the diffracted peak and H_L is the Full Width at Half Maximum (FWHM) of the Lorentz's function.

The Gauss function normalized by area is:

Equation 5

$$G(x) = 2\sqrt{\frac{ln(2)}{\pi}} \frac{\frac{1}{H_G}}{e^{4\,ln(2)\left(\frac{x-x_{hkl}}{H_G}\right)^2}}$$

Where H_G is the FWHM of the Gauss`s function.

The psVoigt is a simplification of the Voigt function, that is a convolution of the Lorentzian and Gaussian with different FWHM. However, this convolution doesn't have an analytical result, and the numeric convolution is heavy to computer. The psVoigt can describe totally the Voigt profile, and is very light to computer. In the psVoigt, H_L and H_G are assumed to have the same FWHM as the FWHM of psVoigt (H_{psV}). The value of H_L and H_G of the equivalent Voigt, can be calculated by a semiempirical equation (Wertheim et al., 1974).

To define the psVoigt, two parameters are necessary. Some approaches use η and the H_{psV}, the parameters H_L and H_G of equivalent Voigt are a function of η and H_{psV}. On the other hand, some approaches define H_L and H_G of the equivalent Voigt and η and H_{psV} parameters are calculated as a function of H_L and H_G. In both cases, the two parameters can be parameterized as a function of x_{hkl} in Whole Profile Fit approach to significantly reduce the number of refined parameters. The parametrization of H_{psV} may adhere to the Caglioti equation[1] (Caglioti et al., 1958):

Equation 6

$$H_{psV}^2 = Utan^2\left(\frac{x_{hkl}}{2}\right) + V\,tan\left(\frac{x_{hkl}}{2}\right) + W + \frac{Z}{cos^2\left(\frac{x_{hkl}}{2}\right)}$$

Equation 7

$$\eta = \eta_0 + \eta_1 \frac{x_{hkl}}{2} + \eta_2 \left(\frac{x_{hkl}}{2}\right)^2$$

Where U, V, W and Z[2] are refinable parameters of H_{psV} and η_0, η_1 and η_2 are refinable parameters of η.

The parameterization of H_L and H_G of Thompson-Cox-Hastings psVoigt (TCHZ) (Thompson et al., 1987) are stable in optimization routine and therefore it is the more commonly used in most of software. The H_L and H_G are calculated following the equations:

[1] Historically, the parameters U and W in the Caglioti equation have been constrained to positive values due to underlying physical relationships. In contrast, the parameter V is permitted to assume negative values, as no physical relationship prescribes its sign. Nonetheless, given the empirical (or semi-empirical) nature of the equation, there are no inherent limitations on the individual values of these parameters.

[2] The Z parameter, while not present in the original publications, was subsequently introduced to establish a correlation with the Scherrer equation for determining crystal size (Young & Desai, 1989).

Equation 8

$$H_G^2 = Utan^2\left(\frac{x_{hkl}}{2}\right) + V\,tan\left(\frac{x_{hkl}}{2}\right) + W + \frac{Z}{cos^2\left(\frac{x_{hkl}}{2}\right)}$$

Equation 9

$$H_L = X\,tan\left(\frac{x_{hkl}}{2}\right) + \frac{Y}{cos\left(\frac{x_{hkl}}{2}\right)}$$

Where U, V, W and Z^2 are refinable parameters of H_G and X and Y are refinable parameters of H_L.

In Sindarin, it is possible to use different approaches. In the simplest one, without FWHM bias dependence with x_{hkl}, each peak profile is fitted independently without parameterization, and for this case, a psVoigt object with two decimal parameters is used in the block of the following objects: RefelctionHKL, ReflectionXI or PeakXI. A Sindarin text example is shown below:

```
1|       Phase
2|             Cell “Cubic” 5.4321
3|                   ReflectionHKL [1 0 0] 200
4|                         psVoigt 0.054 0.41
5|                   ReflectionHKL [1 0 1] 150
6|                         psVoigt 0.089 0.52
7|                   ReflectionHKL [2 1 0] 100
8|                         psVoigt 0.12 0.63
```

On the other hand, it is possible to use the parametrization approach to use fewer parameters and achieve stabilization in the optimization routine. The TCHZ approach, which uses six parameters in the object declaration (Equation 8 and Equation 9):

```
1| psVoigt 0.0003542 -0.00179 0.00257 0 0.04275 0.00842
```

Alternatively, using the parametrization of FHWM and eta, with seven parameters in object declaration (Equation 6 and Equation 7):

```
1| psVoigt 0.08044 0.001293 0.08138 -0.07885 0.3083 0.01257 -3.5208E-
   05
```

4.8.6 Peak asymmetry

The diffraction profile often assumes an asymmetrical shape, when one side of the profile is wider than the other. The asymmetry has different origins, both from geometry of the diffractometer and from the effects of the samples. There are two approaches for modeling the asymmetry in diffractogram analysis: a physic modeling of the effect from the instrument and the sample, or a simple and phenomenological modeling.

The first approach is made by the Fundamental Parameters Approach (FPA) and the Whole Powder Pattern Modeling (WPPM). FPA has functions that can model each effect of the geometry of the x-ray beam in the diffractometer, such equatorial and axial divergence, and WPPM has functions that can model the microstructure of the sample, such stacking-fault. These approaches can be very accurate and detailed. In Sindarin, three functions are available to address asymmetry through the FPA approach: AxialDivergence (page 68), EquatorialDivergence (page 82) and Transparency (page 108).

On the other hand, for general purposes, a simple and empirical function can be introduced to modify the profile function and describe the asymmetry. In Sindarin, Split (page 105) functions offer an alternative means to rectify asymmetry using a straightforward and empirical approach.

In Sindarin, the Circle (page 73) function constitutes the basis for axial divergence correction. However, it is possible to directly use the Circle object, whose parameter is calculated according to the diffraction angle by an empirical function. This delineates a compromise between the empirical approach's simplicity and the precision afforded by the rapid convolution algorithm intrinsic to the FPA method.

Symmetric Voigtian functions possess an analytical convolution. However, when a split function is employed to correct asymmetry, the convolution of this asymmetric Voigtian peak profile is assessed through FFT convolution.

4.9 DIFFRACTION GEOMETRIES

Sindarin can perform X-ray diffraction calculations with different experimental geometries: transmission, reflection, symmetric, asymmetric, with or without monochromator and different samples: flat specimen, capillary and thin film. Additionally, it is possible to set the polarization of the incident beam (useful for synchrotron radiation) and the variable receive slit. Each geometry has specific corrections for the intensity and position of the peaks.

The following sections show some Sindarin text examples for different diffraction setups. These Sindarin texts only cover the diffraction setup, and do not include other necessary declarations for the calculation, such as Radiation and background. It is useful to keep in mind that the default diffraction setup in Sindarin is reflection, symmetric, flat specimen, unpolarized incident beam and Lorentz factor not applied.

When either the MonochromatorTwoTheta or the Polarization property of the Experimental object is declared, the Lorentz factor is automatically applied to the intensity calculation. These properties will modify the $L_P(x)$ term of the intensity equation (Equation 2).

The SpecimenThickness and IncidentAngle properties of the Experimental object and the Capillary object will modify the *Abs(x)* term of the intensity equation. The Capillary object can also modify the *Displ(x)* term of the position equation (Equation 1).

To correct the shift of the diffraction reflections due to the displacement of the sample, there is the Displacement property of the Experimental object. Just declare it as: "Displacement -0.00123". Another available intensity correction (*Abs(x)* term of equation) is the variable divergence slit. Set the VariableSlit property of the Experimental object to True as: "VariableSlit True".

4.9.1 Diffraction Setup: Reflection, Symmetric, Flat Specimen, without Lorentz-Polarization correction

Sindarin text:

```
1| Experimental "dataFile.xye"
```

All the properties and objects related to the diffraction setup are set to default. The $L_P(x)$ and $Abs(x)$ terms are equal to 1 in Equation 2. In this setup, the intensity of the reflection objects, when declared by the user, is directly the value of the intensity observed in the diffractogram (I_{ref})

4.9.2 Diffraction Setup: Reflection, Symmetric, Flat Specimen

Sindarin text:

```
1| Experimental "dataFile.xye"
2|       Polarization 0 // to apply the Lorentz factor
```

For this setup, it is only necessary to declare the Polarization property. The other properties and objects related to the diffraction setup are set to default. When the Polarization property is declared, even if it is zero, the Lorentz factor is applied in the calculation. The $L_P(x)$ term is different from 1 and the $Abs(x)$ term is 1.

4.9.3 Diffraction Setup: Reflection, Symmetric, Flat Specimen, Polarized incident beam

Sindarin text:

```
1| Experimental "dataFile.xye"
2|       Polarization 0.95 //polarization fraction of incident beam
```

The code is very similar to the previous example, but the Polarization property is set to a value other than zero.

4.9.4 Diffraction Setup: Reflection, Symmetric, Flat Specimen, Monochromator

Sindarin text:

```
1| Experimental "dataFile.xye"
2|       MonochromatorTwoTheta 28.44 //2θ of monochromator
```

For this setup, it is only necessary to declare the MonochromatorTwoTheta property. The other properties and objects are set to default. When the MonochromatorTwoTheta property is declared, the Lorentz factor and the polarization of the monochromator are applied in the calculation. It is not necessary to declare "Polarization 0". The $L_P(x)$ term is different from 1 and the $Abs(x)$ term is 1.

4.9.5 Diffraction Setup: Reflection, Symmetric, Flat Specimen, Monochromator, Polarized incident beam

Sindarin text:

```
1| Experimental "dataFile.xye"
2|       Polarization 0.95 //polarization fraction of incident beam
3|       MonochromatorTwoTheta 28.44 //2θ of monochromator
```

The Lorentz factor is applied in the calculation. The $L_P(x)$ term is different from 1 and the $Abs(x)$ term is 1.

4.9.6 Diffraction Setup: Reflection, Symmetric, Thin Film

Sindarin text:

```
1| Experimental “dataFile.xye”
2|       Polarization 0 //to apply the Lorentz factor
3|       SpecimenThickness 0.1 //MuT
```

The value of the SpecimenThickness property is the linear absorption coefficient multiplied by the thin film thickness. The intensity correction in this case is applied by the *Abs(x)* term, which is different from 1.

If a polarized incident beam or a monochromator is used, just declare the appropriate property, as in the previous examples. The $L_P(x)$ term is different from 1.

4.9.7 Diffraction Setup: Reflection, Asymmetric, Flat specimen

Sindarin text:

```
1| Experimental “dataFile.xye”
2|       Polarization 0 //to apply the Lorentz factor
3|       IncidentAngle 4
```

The value of the IncidentAngle property is the fixed angle of the incident beam on the sample. The intensity correction is applied by the *Abs(x)* term, which is different from 1.

If a polarized incident beam or a monochromator is used, just declare the appropriate property, as in the previous examples. The $L_P(x)$ term is different from 1.

4.9.8 Diffraction Setup: Reflection, Asymmetric, Thin Film

Sindarin text:

```
1| Experimental “dataFile.xye”
2|       Polarization 0 //to apply the Lorentz factor
3|       SpecimenThickness 0.1 //MuT
4|       IncidentAngle 4
```

For this diffraction setup, just declare both the SpecimenThickness and IncidentAngle properties.

The *Abs(x)* and $L_P(x)$ terms are different from 1. If a polarized incident beam or a monochromator is used, just declare the appropriate property, as in the previous examples.

4.9.9 Diffraction Setup: Transmission, Capillary

Sindarin text:

```
1| Experimental “dataFile.xye”
2|       Polarization 0 //to apply the Lorentz factor
3|       Capillary 0.7 -0.31 0.047 //MuR, dispL, dipV
```

The three parameters of the Capillary object are: linear absorption coefficient multiplied by capillary radius, displacement in direction of incident beam and displacement in perpendicular direction of incident beam, respectively. The Capillary object will apply

intensity correction through the *Abs(x)* term (Equation 2) and position correction through the *Displ(x)* term (Equation 1).

If a polarized incident beam or a monochromator is used, just declare the appropriate property, as in the previous examples.

4.10 RADIATION MODELING

The modeling of radiation is very flexible in Sindarin. You can use any wavelength, any number of discrete lines, any relative intensity of each discrete line, incident beam polarization fitting and individual discrete line profile modeling with physical parameters. The incident beam polarization is an intensity correction described in section 4.9.3.

A Radiation object can be a child of Experimental and Simulation objects. The basic declaration syntax has two parameters: the wavelength and relative intensity:

```
1|       Radiation 1.540598 1
```

One can declare as many Radiation objects as needed. For example, to describe the two copper K_α lines:

```
1|       Radiation 1.540598 1
2|       Radiation 1.544426 @0.455
```

The intensity of the $K_{\alpha 2}$ Radiation is about half of the main discrete line, the $K\alpha_1$. In the Sindarin text example above, the intensity of the second Radiation object is a refinable variable, which means that this decimal data will be fitted in the optimization routine.

Each Radiation object will create one PeakXI object in each reflection object (see section 4.6).

The profile of each discrete radiation can be modeled by declaring Profile objects in each Radiation object block. There are several Voigtian profile objects with the FWHM parameter described in terms of wavelength unit and the dependence with the diffracted angle follows the diffraction optics. These peak profiles are very useful for modeling the physical profile of X-ray generated at the radiation source and are used in conjunction with FPA analyses.

The radiation generated by a Cu cathode X ray tube in a typical laboratory diffractometer can be described in detail by 5 discrete radiation lines as reported in (Cheary et al., 2004). The Sindarin text below shows how to use this radiation model in Sindarin:

```
1|       Radiation (Lambda1 = 1.53475) 0.0137
2|             LorentzWavelength Lambda1 3.686E-3
3|       Radiation (Lambda2 = 1.540591) 0.5710
4|             LorentzWavelength Lambda2 0.437E-3
5|       Radiation (Lambda3 = 1.541065) 0.0789
6|             LorentzWavelength Lambda3 0.643E-3
7|       Radiation (Lambda4 = 1.544399) 0.2328
8|             LorentzWavelength Lambda4 0.513E-3
9|       Radiation (Lambda5 = 1.544686) 0.1036
10|            LorentzWavelength Lambda5 0.687E-3
```

4.11 CRYSTALLOGRAPHIC INFORMATION

A Cell object stores all crystallographic information of a phase. The first parameter is a string and represents the symmetry space group. The next parameters are decimal numbers and are the lattice parameters. The number of decimal parameters depends on the lattice type. For instance, only one decimal parameter is required for a cubic lattice, and six decimal parameters are required for a triclinic lattice.

In the block of a Cell object, it is possible to declare two types of objects: ReflectionHKL and atoms. Atoms are declared using the symbol of the chemical element in the first word of the line followed by five decimal parameters: three for relative space position inside the cell (x, y, z), occupation of the crystallographic site (occ) and isotropic atomic displacement factor (uiso). ReflectionHKL is used mainly to describe diffraction reflection without atoms information (see Only lattice parameter information (Pawley method)).

The rutile is a typical structure of TiO_2 (COD 1530150) and the crystallographic information is:

- Cell type: tetragonal
- Lattice parameters: a = b = 4.594, c = 2.959 and $\alpha = \beta = \gamma = 90°$.
- Space Group (Hermann-Mauguin): P 4_2 / m n m (number 136)
- Oxygen atom position (x, y, z): 0.327, 0.327, 0
- Titanium atom position (x, y, z): 0, 0, 0
- Oxygen and Titanium site occupation: 1

So, the Sindarin text for this structure is:

```
1|              Cell "P 42/M N M" 4.59 2.96
2|                    O2- 0.327 0.327 0 1 0.0115
3|                    Ti4+ 0 0 0 1 0.00909
```

4.12 EXAMPLE OF SINDARIN TEXT TO RIETVELD REFINEMENT

This section presents a typical example of Sindarin text to Rietveld refinement. The experimental setup consists of a diffractometer with Bragg-Bretano geometry, copper x-ray tube and a graphite monochromator. The sample is a single-phase rutile (TiO_2):

```
1| Experimental "data.xye"
2|       Radiation 1.540560 1
3|       Radiation 1.544390 0.5
4|       MonochromatorTwoTheta 28.44
5|       Polynomial 227.2 -115.96 15.5 209.5 -106.58 -330.5 309.3
6|       psVoigt 0.013623 -0.00884 0.00283 0 0.0525 0.00489
7|             split -0.1968 0.0232 0
8|       Displacement -2.105E-5
9|       zero 0.0104
10|      Phase
11|            Scale 0.00173
12|            Cell "P 42/M N M" 4.59366447 2.95956998
13|                  O2- 0.327 0.327 0 1 0.09576
14|                  Ti4+ 0 0 0 1 0.0478
```

Line by line comments:

Line 1: Create an Experimental object with the data from the file data.xye. All other information below this line belongs to the block of this Experimental object. For more information, see section Experimental.

Line 2: Create a Radiation object for the $K\alpha_1$ line of copper: wavelength 1.54056Å and relative intensity of 1. For more information, see sections Radiation modeling and Radiation.

Line 3: Create a Radiation object for the $K\alpha_2$ line of copper: wavelength 1.54439Å and relative intensity of 0.5.

Line 4: The 2θ angle of the crystal monochromator used, to graphite is 28.44 °2θ. The Lorentz correction is applied when the monochromator is specified. For more information, see sections Diffraction Setup: Reflection, Symmetric, Flat Specimen, Monochromator and MonochromatorTwoTheta.

Line 5: Use a polynomial function with seven parameters to model the background. For more information, see section Polynomial.

Line 6: Use a pseudo-Voigt function with parametrized FWHM as function of 2θ to model the diffraction profile. Six parameters are informed: U, V, W and Z for Gaussian FWHM and X and Y for Lorenzian FWHM. Apply this profile to all peaks of calculated diffractogram. For more information, see sections Profile level, The pseudo-Voigt function and psVoigt.

Line 7: Create a Split object inside the psVoigt block that modifies the pseudo-Voigt function to model asymmetry of diffraction profile. For more information, see sections Peak asymmetry and Split.

Line 8: Correct the sample displacement from goniometer center. The decimal number is the ration of the displacement and the diffractometer radius. For more information, see sections General equations for calculating the position and intensity of reflections and Displacement.

Line 9: Correct the zero angle of the diffractometer. For more information, see sections General equations for calculating the position and intensity of reflections and Zero.

Line 10: Create a Phase object. For more information, see section Phase.

Line 11: Set the Scale factor of the phase. For more information, see sections General equations for calculating the position and intensity of reflections and Scale.

Line 12: Define the Cell object of the space group P 4_2 / m n m (number 136), with tetragonal lattice, a = 4.59366447Å and c = 2.95956998Å. For more information, see sections Crystallographic Information and Cell

Line 13: Define an atom object of the oxygen element with an oxidation number of -2. The relative position inside the cell is 0.327, 0.327 and 0 (x, y, z), the relative site occupation is 1 and the atomic displacement factor is 0.09576. For more information, see sections Crystallographic Information and Atoms

Line 14: Define an atom object of the titanium element with an oxidation number of +4. The relative position inside the cell is 0, 0 and 0 (x, y, z), the relative site occupation is 1 and the atomic displacement factor is 0.0478.

5 TUTORIAL

In this chapter, you will learn how to use Sindarin to calculate information on crystalline solid samples from X-ray diffraction experimental data. The data files that you need are located in the Sindarin folder (on Nimloth, a Work Library is created for these tutorials). These files are based on real experimental measurements, but they have been simulated for the purpose of these tutorials. The tutorials will guide you in writing the Sindarin text that sets up the calculations, adjusting the parameters of the theoretical model to fit the experimental data, running the Sindarin commands that perform the calculations, and checking the results that you obtain. The tutorials are designed to walk you through the calculations step by step. You will see different examples of calculations that cover various user needs and experimental measurement settings. The following are some examples: quantitative phase analysis, lattice parameter calculation, and microstructural analysis; measurements in a laboratory diffractometer with an X-ray tube using Bragg-Brentano geometry and measurements with synchrotron light; refinement with full crystallographic information and refinement with only lattice parameters and a reflection list.

5.1 GENERAL INFORMATION

All necessary files, including the data file, Sindarin file and crystallographic information files (CIF), required for following the tutorials are located in a folder named "Tutorials" inside of Sindarin program directory. These files are systematically organized into separate subfolders corresponding to each tutorial.

To edit the Sindarin files within the tutorial folder, one may utilize a preferred text editor (refer to "User Interface" chapter). It is to be noted that a Sindarin file is essentially a simple text file. Hency, any text file can be created, edited, and utilized to execute calculations in Sindarin using Sindarin commands. It is recommended that the file extension be ".sin" (refer to the "Sindarin file" chapter for details).

In instances where Nimloth is employed, a "Tutorials" entry is generated within the Works library upon the initial launch of the application. This entry contains all the tutorials present in this manual. One should select the desired tutorial Work to access and edit the corresponding Sindarin file and other files.

For instructions on executing Sindarin commands, consultation of the "User Interface" chapter is advised. Each user interface presents a unique method for execution.

Attention must be paid to the indentation of lines, that is the tab characters at the beginning of each line. The level of indentation delineates the hierarchical relationship among objects. For further details, please consult the chapters on "Indentation" and "Object's Block".

All text presented in these tutorials, formatted as code, may be copied and pasted into the Sindarin text file within the text editor. The indentation is appropriately structured. The following text serves as an example of the code format.

```
This line is in code format
```

Each tutorial encompasses the complete set of steps and explanations, as it is designed to be conducted independently. Consequently, there is a considerable amount of information and text that is reiterated across them. Feel free to omit certain paragraphs, not the steps, if the user has completed other tutorial beforehand.

All CIF files are obtained from Crystallography Open Database (COD).

5.2 TUTORIAL 0: FIRST CONTACT WITH SINDARIN

This tutorial provides an initial exposure to Sindarin text and the fundamental principles of Sindarin commands. The accompanying experimental data file contains a simulated dataset that delineates a polynomial function. This function is defined within the horizontal coordinate range of -10 to 10. The polynomial, characterized by the equation below, is of the third degree:

$$y = 3.2 + 2.5x + 1.75x^2 + 0.5x^3$$

Open the Sindarin file "sample.sin" located in the Tutorial 0 directory using a text editor. On Nimloth, access the corresponding Work and proceed to Sindarin Editor.

To commence the text, input the following line without any leading indentation. This instruction instantiates a Data object within the memory, which utilizes the file "sample.xye", containing the experimental data, as its parameter. The primary function of this object is to load the data from the aforementioned file:

```
Data "sample.xye"
```

Subsequent lines belong to the Experimental object block, where the indentation of these lines is greater than that of the Experimental line.

For the subsequent line, begin with a single tab character to denote indentation. This line initializes a Polynomial object that models a polynomial function. The parameters of this object correspond to the coefficients of the function. Given that there is only one parameter, the polynomial is of degree 0, which is represented as a horizontal line on the graph:

```
	Polynomial 0
```

Execute the interpret command to perform the calculations. At this point, the value of χ^2 is 48738, indicating substantial deviation of the calculated function from the experimental data. The graphical representation of the calculated function is a horizontal line at Y=0.

To illustrate the optimization command in decimal data, put a "@" on left side of the number 0 of Polynomial object. This line should look like this:

```
	Polynomial @0
```

Proceed by executing the Marathon command.

After finishing the optimization routine, the χ^2 is 45104. The original text is updated with the new optimized value of the first parameter of Polynomial, which is written with the estimated standard deviation enclosed in braces "{}":

```
Polynomial @61.9953451{±15}
```

At this stage, the function represents a horizontal line at Y=54.35. To refine the model further, add an additional parameter to the Polynomial object with a value of "0" and designate it for optimization by prefixing it with an "@" symbol:

```
Polynomial @61.9953451{±15} @0
```

Re-run the Marathon command. The χ^2 value is now reduced to 8749, and the text is updated accordingly:

```
Polynomial @62.116637{±6.6} @32.7909924{±1.1}
```

The theoretical function now represents an inclined line, indicative of a first-degree polynomial.

Include more two parameters with initial value 0, sign to refine each parameter at a time and execute the Marathon command:

```
   Polynomial @3.22876789{±5.5} @32.7989961{±0.94} @1.74914551{±0.12}
// Chi2 = 5975
   Polynomial @3.20000022{±4.2E-05} @2.50023116{±7.2E-06}
@1.74999999{±9.2E-07} @0.499996185{±1.1E-07} // Chi2 = 0
```

Ultimately, the function impeccably describes the experimental data, with a χ^2 is 0 (and the r^2 is 1.0). The final function is a third-degree polynomial, with the calculated parameters being: 3.20000(2), 2.500116(4), 1.7500000(5), 0.49999809(5). The numbers in parentheses following the value of each decimal number represent the ESD for the last significant digit.

5.3 Tutorial 1: Quantitative Phase Analysis

Tutorial 1 encompasses a standard quantitative phase analysis utilizing X-ray diffraction. The specimen comprises a binary mixture of titanium oxide phases: rutile and anatase. The analytical apparatus employed is a laboratory diffractometer equipped with an X-ray tube, featuring copper as the anode material. The diffractometer's arms are configured symmetrically, accommodating a flat specimen in reflection mode. Additionally, a graphite monochromator is situated preceding the point detector.

Open the Sindarin file "sample.sin" located in the Tutorial 1 directory using a text editor. On Nimloth, access the corresponding Work and proceed to Sindarin Editor.

To initiate the text, type the above line without indentation. This line instantiates the Experimental object in memory, utilizing the file "sample.xye" containing experimental data as a parameter. This object is responsible to load the data from the specified file:

```
Experimental "sample.xye"
```

Subsequent lines belong to the Experimental object block, where the indentation of these lines is greater than that of the Experimental line.

Input the next lines with a single tab character at the beginning for indentation. These lines generate two Radiation objects, each representing a Kα radiation line of the cooper element:

```
	Radiation 1.540598 1
	Radiation 1.544426 0.5
```

The ensuing command is employed to account for the monochromator's influence on the polarization of the diffracted X-ray beam. The specified value, 28.44, corresponds to the angle of the graphite analyzer crystal. Notably, the Lorentz correction is automatically applied upon assigning this property (for further information, refer to the "Diffraction Setup: Reflection, Symmetric, Flat Specimen, Monochromator" Chapter):

```
	MonochromatorTwoTheta 28.44
```

To adjust for background, input the following line. The decimal parameter of 100 server as an arbitrary starting point, approximating the minimum intensity observed in the experimental data:

```
	Polynomial 100
```

The following lines initializes the Phase object, within whose block the information about the phase, including crystallographic information, is specified. The string parameter is optional and serves merely as a identifier for this phase and does not influence the calculation:

```
	Phase "Anatase"
```

The indentation of the subsequent three lines, marked by two tab characters, signifies their inclusion within the Phase block.

The line below sets the scale factor of the phase:

```
		Scale 0.01
```

The next line instantiates a psVoigt object and associates it with the current phase. This pseudo-Voigt profile function applies to all peaks of this phase:

```
		psVoigt 0 0 0.01 0 0 0
```

The crystallographic information for the anatase phase is obtained from COD code 1530151. The correspondent CIF file is located in the tutorial directory.

The ensuing line generate a Cell object with space group I 4_1 / a m d and tetragonal lattice parameters of 3.78 and 9.51 Å:

```
Cell "I 41/A M D:1" 3.78 9.51
```

The cell encompasses two crystallographic sites: one occupied by Ti^{4+} at the origin, and the other by O^{2-} at relative coordinates x = 0, y = 0 and z = 0.1775. To declare an atom within the Cell block, the atom (or ion) symbol followed by an optional string label, three decimal values representing the atom's relative position within the cell, a decimal for the site occupation, and a final decimal for the atomic displacement parameter. The indentation of this lines uses three tab characters:

```
		Ti4+ "Ti1" 0 0 0 1 0.01
		O2- "O1" 0 0 0.1775 1 0.01
```

For the rutile phase, COD code 9004141 was utilized. The Sindarin text for this phase is as follows:

```
Phase "Rutile"
	Scale 0.01
	psVoigt 0 0 0.01 0 0 0
	Cell "P 42/M N M" 4.593 2.959
		Ti4+ “Ti1” 0 0 0 1 0.01
		O2- “O1” 0.3051 0.3051 0 1 0.01
```

The initial Sindarin text is complete and the refinement of the parameters could be started.

Execute the interpret command to perform the calculations. At this point, the χ^2 is 90.59 and the calculated diffractogram differs significantly from the experimental diffractogram, especially in the intensity and width of each peak. However, all peaks of the two phases and the background are present in calculated diffractogram.

To illustrate the optimization command in decimal data, put a "@" on left side of the number 100 of Polynomial object. This line should look like this:

```
Polynomial @100
```

Execute the walk command.

After finishing the optimization routine, the χ^2 is 89.17. The original text is updated with the new optimized value of the first parameter of Polynomial, which is written with the estimated standard deviation enclosed in braces "{}":

```
Polynomial @87.6222027{±1.2}
```

Next, refine the scale of the two phases, write the "@" on left side of the decimal data of Scale property of both phases and run the walk command. The χ^2 is now 33.19. The new values are written in the text:

Scale of the anatase phase after optimization:

```
        Scale @0.0109021647{±0.00017}
```

Scale of the rutile phase after optimization:

```
        Scale @0.0159739102{±0.0005}
```

Note that all decimal data marked with "@" are optimized, and the first parameter of Polynomial object was optimized and updated to 98.6615345{±0.73}.

Refine each psVoigt parameter, one by one, for each phase at a time. Be care with parameter Z of the gaussian (the fourth parameter of psVoigt), as it frequently has high correlation with others gaussian parameters. Therefore, if the model does not use the Z parameter to relate to a physical model, set it to zero and do not refine it.

The refined value of each parameter of psVoigt for the anatase phase is:

```
        psVoigt @0.0381318006{±0.0011} 0 0.01 0 0 0 // Chi2 = 27.58
        psVoigt @0.0489453603{±0.0048} @0.164330713{±0.0038} 0.01 0
0 0 // Chi2 = 19.63
        psVoigt @0.238433982{±0.035} @0.0607873592{±0.033}
@0.0429714632{±0.0064} 0 0 0 // Chi2 = 18.13
        psVoigt @105260.284 @-0.0227346743{±0.042} @105260.065 @-
105259.98 0 0 // Chi2 = 17.88
        psVoigt @0.0744250566{±0.035} @-0.0318636216{±0.021}
@0.0680701336{±0.0037} 0 @0.240396799{±0.0081} 0 // Chi2 = 17.29
        psVoigt @0.0965209583{±0.011} @-0.107362767{±0.0018}
@0.0299537624{±0.00077} 0 @0.23981764{±0.018} @0.107330071{±0.0041}
// Chi2 = 14.89
```

The refined value of each parameter of psVoigt for the rutile phase is:

```
        psVoigt @0.0193763101{±0.0012} 0 0.01 0 0 0 // Chi2 = 14.30
        psVoigt @0.0044609696{±0.0087} @0.124702209{±0.0061} 0.01 0
0 0 // Chi2 = 12.60
        psVoigt @0.611946509{±0.1} @-0.249795637{±0.092}
@0.10906531{±0.019} 0 0 0 // Chi2 = 12.01
        psVoigt @0.395158677{±0.13} @-0.126122406{±0.095}
@0.0899133819{±0.024} 0 @0.0587679881{±0.055} 0 // Chi2 = 11.97
        psVoigt @0.520128357{±0.19} @-0.143441323{±0.14}
@0.0209692495{±0.029} 0 @-0.202308611{±0.11} @0.198369517{±0.035} //
Chi2 = 11.77
```

Next, refine all the lattice parameters of each phase.

The results for the anatase phase are:

```
        Cell "I 41/A M D:1" @3.78742119{±8.7E-05}
@9.50876958{±0.00046} // Chi2 = 4.86
```

The result for the rutile phase is:

```
        Cell "P 42/M N M" @4.59571811{±0.00028}
@2.95885313{±0.00032} // Chi2 = 4.75
```

A XRD standard was not measured for this study and the alignment of the diffractometer arms (the Zero property of Experimental object) was not corrected, so the results of the lattice parameters are not accurate and should be used with caution.

At this point, increase the number of parameters of the Polynomial object by three, one by one:

```
   Polynomial @83.1878913{±0.3} @-8.28929491{±0.49} // Chi2 = 4.53
   Polynomial @66.8613854{±0.2} @-7.57916089{±0.37}
@50.4954739{±0.23} // Chi2 = 2.56
   Polynomial @65.9889957{±0.31} @7.31406166{±0.43}
@51.3298022{±0.68} @-24.8639752{±0.54} // Chi2 = 2.45
```

At this point, the refinement can be considered finished. The χ^2 is low and the theoretical diffractogram is very similar to the experimental diffractogram. Other parameters, such as zero, peak asymmetry, biso and position of each atom, can be refined, but they will not have significant impact on the main result, which is amount of each phase.

Execute the command to write the result to a file (or, if in Nimloth, go to the result panel). The relative amount of each phases is shown, the anatase phase is 78(2)% and the rutile is 22(2)%

5.4 TUTORIAL 2: LATTICE PARAMETERS ANALYSIS

Tutorial 2 encompasses a lattice parameters analysis utilizing X-ray diffraction. The specimen comprises a pure cerium oxide. The analytical apparatus employed is a laboratory diffractometer equipped with an X-ray tube, featuring copper as the anode material. The diffractometer's arms are configured symmetrically, accommodating a flat specimen in reflection mode. No monochromator is used.

For precise lattice parameters values, it is essential to measure the X-ray diffraction of a line position standard under identical diffractometer conditions as those used for the sample measurement. In this tutorial, the NIST standard SRM 640 (silicon) is employed for this purpose. The objective is to obtain the Zero correction for the position of the diffractometer arms. The diffractogram of the standard must be analyzed first.

The theoretical refinement model employed in this tutorial does not incorporate atom information. This is due to the fact that the lattice parameters, which are the sole focus of interest, are unrelated to the positions or types of atoms.

Open the Sindarin file "stand.sin" located in the Tutorial 2 directory using a text editor. On Nimloth, access the corresponding Work and proceed to Sindarin Editor.

To initiate the text, type the above line without indentation. This line instantiates the Experimental object in memory, utilizing the file "stand.xye" containing experimental data as a parameter. This object is responsible to load the data from the specified file:

```
Experimental "stand.xye"
```

Subsequent lines belong to the Experimental object block, where the indentation of these lines is greater than that of the Experimental line.

Input the next lines with a single tab character at the beginning for indentation. These lines generate two Radiation objects, each representing a Kα radiation line of the cooper element:

```
Radiation 1.540598 1
Radiation 1.544426 0.5
```

To adjust for background, input the following line. The decimal parameter of 100 server as an arbitrary starting point, approximating the minimum intensity observed in the experimental data:

```
Polynomial 100
```

The next line instantiates a psVoigt object and associates it with Experimental object. This pseudo-Voigt profile function is applicable to all peaks of all phases (in this instance, there is only one phase):

```
psVoigt 0 0 0.01 0 0 0
```

The following lines initializes the Phase object, within whose block the information about the phase, including crystallographic information, is specified. The string parameter is optional and serves merely as an identifier for this phase and does not influence the calculation:

```
Phase "Silicon"
```

The indentation of the subsequent lines, marked by two or more tab characters, signifies their inclusion within the Phase object block.

The ensuing line generate a Cell object with space group F d $\bar{3}$ m and tetragonal lattice parameter of 5.431179 Å, information obtained from standard NIST certificate:

```
Cell "F D -3 M:1" 5.431179
```

Within the Cell object block, the ReflectionHKL objects are defined. Each object contains two parameters: the first is an array of three integers representing the Miller indices of the reflection; the second parameter corresponds to the reflection's intensity. To circumvent refinement instability caused by a high correlation between the intensities of reflection (1 1 5) and (3 3 3), which have identical position, only reflection (3 3 3) is incorporated into this theoretical model:

```
ReflectionHKL [1 1 1] 10000
ReflectionHKL [2 0 2] 10000
ReflectionHKL [1 1 3] 10000
ReflectionHKL [4 0 0] 10000
ReflectionHKL [1 3 3] 10000
ReflectionHKL [2 2 4] 10000
ReflectionHKL [3 3 3] 10000
ReflectionHKL [4 0 4] 10000
ReflectionHKL [1 3 5] 10000
```

The initial Sindarin text has been finalized, allowing for the commencement of parameter refinement. It should be noted that polarization corrections are unnecessary, as the peak intensities are considered arbitrary parameters within the context of this study.

Execute the interpret command to perform the calculations. At this point, the χ^2 is 201.50 and the calculated diffractogram differs significantly from the experimental diffractogram. However, all peaks of the phase and the background are present in calculated diffractogram.

To illustrate the optimization command in decimal data, put a "@" on left side of the number 100 of Polynomial object. This line should look like this:

```
Polynomial @100
```

Execute the walk command.

After finishing the optimization routine, the χ^2 is 151.52. The original text is updated with the new optimized value of the first parameter of Polynomial, which is written with the estimated standard deviation enclosed in braces "{}":

```
Polynomial @205.47025{±2.6}
```

Next, refine the intensity of all ReflectionHKL objects, write the "@" on left side of the second parameter of the ReflectionHKL objects and run the walk command. The χ^2 is now 23.48. The new values are written in the text:

```
ReflectionHKL [1 1 1] @5995.32463{±1.1E+02}
ReflectionHKL [2 0 2] @8654.82694{±2E+02}
ReflectionHKL [1 1 3] @3576.26452{±1.1E+02}
ReflectionHKL [4 0 0] @6197.85158{±4.3E+02}
ReflectionHKL [1 3 3] @2577.52875{±1.5E+02}
ReflectionHKL [2 2 4] @3652.0958{±1.9E+02}
ReflectionHKL [3 3 3] @6004.72356{±4.7E+02}
ReflectionHKL [4 0 4] @1969.87625{±2.5E+02}
ReflectionHKL [1 3 5] @820.60419{±68}
```

Note that all decimal data marked with "@" are optimized, and the first parameter of Polynomial object was optimized and updated to 217.363{±1}.

Refine each psVoigt parameter, one by one, for each phase at a time, starting with the nonzero parameter. Be care with parameter Z of the gaussian (the fourth parameter of psVoigt), as it frequently has high correlation with others gaussian parameters. Therefore, if the model does not use the Z parameter to relate to a physical model, set it to zero and do not refine it.

The refined value of each parameter of psVoigt for the anatase phase is:

```
   psVoigt 0 0 @0.00570526344{±0.00033} 0 0 0 // Chi2 = 21.67
   psVoigt @0.000787828174{±0.00034} 0 @0.0051298818{±0.00018} 0 0 0
// Chi2 = 21.60
   psVoigt @-7.80473866E-05{±0.0014} @0.00177900855{±0.002}
@0.00439349088{±0.00062} 0 0 0 // Chi2 = 21.58
```

```
   psVoigt @-0.000360177851{±0.00028} @-0.000604181858{±0.00021}
@0.00431652518{±0.00011} 0 @0.0235936942{±0.0018} 0 // Chi2 = 21.40
   psVoigt @-0.00148008535{±0.00095} @0.0031518711{±0.0017}
@0.000921536495{±0.0006} 0 @-0.00667311968{±0.0075}
@0.0262950318{±0.0034} // Chi2 = 20.34
```

Increase the number of parameters of the Polynomial object by three, one by one:

```
   Polynomial @220.054177{±0.87} @-54.3038548{±1.5} // Chi2 = 16.06
   Polynomial @215.864116{±1.3} @-55.3302255{±1.5} @12.5163189{±2.9}
// Chi2 = 16.00
```

Include the following line in the Experimental object block (one tab character as indentation) to correct the Zero of the diffractometer:

```
   Zero @0
```

Upon assigning the decimal value for refine and executing the Walk command, the value of the Zero is -0.0185934561{±0.00068} with the χ^2 currently 8.17.

Subsequently, assign the Displacement property within the Experimental object block (one tab character as indentation):

```
   Displacement @0
```

After refining the decimal value and executing the walk command, the value of the Displacement is 0.000806775035{±6.2E-05} and χ^2 equals 5.95.

To rectify the asymmetry of the peaks, incorporate the Split object within the psVoigt object block:

```
         Split 0 0 0
```

Refine the decimal parameters of this object sequentially:

```
         Split @-0.281082231{±0.01} 0 0 // Chi2 = 4.93
         Split @0.222740225{±0.021} @-0.160045594{±0.006} 0 // Chi2 =
4.17
         Split @0.583183299{±0.085} @-0.412158248{±0.057}
@0.0386203397{±0.0088} // Chi2 = 4.13
```

Run walk command some time to have sure that the refinement reached the local minimum. The χ^2 is 4.11 now.

At this point, the refinement of the standard is complete, the requisite information for refining the sample diffractogram is the Zero parameter, recorded as 0.0103501339{±0.0041}. The psVoigt object, inclusive the Split object, of the standard refinement serves as an advantageous starting point for refining the sample peak profile and will therefore be utilized in the Sindarin file of the sample.

Open the "sample.sin" Sindarin file located in the Tutorial 2 directory using a text editor. On Nimloth, access the corresponding Work and proceed to Sindarin Editor.

To initiate the text, type the above line without indentation utilizing the file name "sample.xye" as parameter:

```
Experimental "sample.xye"
```

Several lines are analogous to those in the Sindarin file used for the standard refinement, hence many explanations are identical and are not presented here. The subsequent four lines are replicated from the initial Sindarin text of the standard:

```
Radiation 1.540598 1
Radiation 1.544426 0.5
Polynomial 100
```

The next three are derived from the final Sindarin file of the standard. It is important to note that none of the decimal parameters in these lines are designated for refinement. These values must remain fixed during the refinement of the sample:

```
   psVoigt 0.0118750657{±0.00066} -0.0197128428{±0.0009}
0.00975592006{±0.00029} 0 0.0250685294{±0.0032}
0.00490697953{±0.0013}
         Split 0.607769983{±0.084} -0.431540361{±0.057}
0.0412900759{±0.0088}
   Zero 0.0103501339{±0.0041}
```

The next lines are to instantiate the Phase and the Cell objects. The space group and the lattice parameter are obtained from the COD code 4343161:

```
Phase "Ce O2"
      Cell "F M -3 M" 5.40972
```

The objective of this tutorial is to obtain the lattice parameters accurately, so it is not necessary information of atoms. Instead, atoms objects in this case are utilized ReflectionsHKL objects within Cell object block. For this phase and inside the range of 2 of the experimental diffractogram the following ReflectionsHKL must be

```
ReflectionHKL [1 1 1] 100
ReflectionHKL [2 0 0] 100
ReflectionHKL [2 0 2] 100
ReflectionHKL [1 1 3] 100
ReflectionHKL [2 2 2] 100
ReflectionHKL [4 0 0] 100
ReflectionHKL [1 3 3] 100
ReflectionHKL [2 0 4] 100
ReflectionHKL [2 2 4] 100
ReflectionHKL [3 3 3] 100
```

The reflection (1 1 5) was excluded from the analysis, despite being within the experimental measurement's angular range, due to its diffraction angle being identical to that of the reflection (3 3 3), which would result in a significant correlation of intensity between these two reflections. While this does not affect the present study, for scenarios involving anisotropic functions reliant on the Miller index, employing Rietveld refinement

is advisable as it allows for the determination of each reflection's intensity based on the structure factor.

Execute the interpret command to perform the calculations. At this point, the χ^2 is 2010. Refine two parameters of the Polynomial, the χ^2 is 1912 and the parameters are:

```
Polynomial @290.842687{±13}
```

Now, refine the intensity of each reflection of the phase. All reflection intensity may be refined simultaneously because the peaks are not overlapped. The χ^2 is 50.36 and the text updated shows:

```
ReflectionHKL [1 1 1] @105.880983{±3.1}
ReflectionHKL [2 0 0] @46.9633986{±2.6}
ReflectionHKL [2 0 2] @34.5458605{±1.5}
ReflectionHKL [1 1 3] @12.6961116{±0.66}
ReflectionHKL [2 2 2] @7.85052572{±1.1}
ReflectionHKL [4 0 0] @9.46162236{±1.5}
ReflectionHKL [1 3 3] @5.83805556{±0.49}
ReflectionHKL [2 0 4] @4.27182168{±0.45}
ReflectionHKL [2 2 4] @6.02336315{±0.51}
ReflectionHKL [3 3 3] @19.3517603{±1.6}
```

Signed to refine the lattice parameter of the Cell object and run walk command. The χ^2 is 35.58 and the new text look like this:

```
Cell "F M -3 M" @5.41188004{±5.9E-05}
```

Now include a line with Displacement property inside the Experimental object block (indentantion with one tab character) with value 0:

```
Displacement @0
```

Signed the value of Displacement to refine and run the walk command. The χ^2 is 12.25 and the new value of this property is 0.000386001063{±5.7E-06}.

Introduce a new line for the psVoigt object within the Phase object block, ensuring it is indented with two tab character. This function models the sample profile, whereas the first psVoigt object, located in the Experimental object block, represents the instrumental profile. For the propose of this tutorial, the physical significance of the sample's width is not considered. The initial value is set to be minimal, yet non-zero:

```
psVoigt 0.00001 0 0 0 0 0 //Chi2 = 13.33
```

Refine one by one parameters of psVoigt of phase object:

UZXY

```
psVoigt @0.00488009921{±0.00015} 0 0 0 0 0 // Chi2 = 5.84
psVoigt @-0.0102314085{±0.0005} @0.0140221237{±0.00036} 0 0
0 0 // Chi2 = 3.14
```

```
        psVoigt @0.0105859687{±0.0014} @-0.0130335453{±0.0018}
@0.00770203257{±0.0005} 0 0 0 // Chi2 = 2.80
        psVoigt @0.00975037281{±0.0012} @-0.0156928683{±0.0014}
@0.00756024322{±0.0004} 0 @0.0211451598{±0.0012} 0 // Chi2 = 2.49
        psVoigt @0.0068569538{±0.0012} @-0.0110027286{±0.0015}
@0.0055735064{±0.00044} 0 @0.017010966{±0.00092}
@0.00415227817{±0.00057} // Chi2 = 2.45
```

Include more two parameters to Polynomial object and refine them. The χ^2 is 2.25 and the new values can be seen in the updated text:

```
   Polynomial @347.177776{±0.69} @-8.0171543{±0.79} @16.1335562{±1.5}
// Chi2 = 2.25
```

To conclude the refinement process, execute the walk command several times ascertain that the local minimal of the refinement has been attained.

The lattice parameter obtained from this process is 5.40890±2 Å, with the estimated standard deviation in fifth decimal place.

6 SINDARIN DICTIONARY

A Dictionary with all "words" used in Sindarin is presented. Here, word means object classes and property names.

6.1 WORDS BY OBJECT TYPE

In this section, terminology is organized according to the classification of object types, accompanied by an explanation of the general aspects pertinent to each category. Only objects, which are capable of containing a block, are classified herein; their properties are not delineated. Concluding this section is a comprehensive enumeration of operations, internal constants, internal variables and functions applicable to decimal data.

6.1.1 Calculation Control

Entry	**Short Description**	**Section**	**Page**
Calculation			

6.1.2 Experimental data

Entry	**Short Description**	**Section**	**Page**
Data			
DataSimulation			
ExcludeRegion			
Experimental			
Interval			
Simulation			

6.1.3 Diffraction

Entry	**Short Description**	**Section**	**Page**
ReflectionXI			
PeakXI			
Ellipsoidal			
MarchDollase			
Radiation			
Phase			

6.1.4 Crystallographic

Entry	**Short Description**	**Section**	**Page**
Cell			
Atoms			
ReflectionHKL			

6.1.5 Background Function

In Sindarin, a variety of functions are available to compute the intensity (denoted as y in the chart) at each experimental point. In instances where the background is irregular and challenging to correct, the amalgamation of certain functions can facilitate this task and simplify the accurate modeling of the diffractogram background. Sindarin allows for the combination of any number of functions; one simply needs to declare multiple background objects within the Sindarin text.

Entry	**Short Description**	**Section**	**Page**
BackgroundFile			
Chebyshev			
CosineFourier			
DebyeLike			
Function			
FunctionPsVoigt			
Interpolation			
Inverse			
Polynomial			

6.1.6 Profile

6.1.6.1 Basic Profile

Entry	**Short Description**	**Section**	**Page**
ProfileFunction			
ProfileFunctionFT			
Split			
ShiftFT			

6.1.6.2 Voigtian function

6.1.6.2.1 Basic Voigtian functions

Entry	**Short Description**	**Section**	**Page**
Gauss			
Lorentz			
psVoigt			
Voigt			

6.1.6.2.2 Voigtian in Wavelength

Entry	**Short Description**	**Section**	**Page**
GaussWavelength			
LorentzWavelength			
psVoigtWavelength			
VoigtWavelength			

6.1.6.2.3 Voigtian Micrstructure

Entry	Short Description	Section	Page
VoigtianSize			
VoigtianStrain			

6.1.6.3 FPA function

6.1.6.3.1 FPA base functions

Entry	Short Description	Section	Page
Box			
Circle			
Exponential			
ProfileInverse			

6.1.6.3.2 Specific aberration correction

Entry	Short Description	Section	Page
AxialDivergence			
Defocusing			
EquatorialDivergence			
ReceivingSlitWidth			
TargetWidth			
Tilt			
Transparency			

6.1.6.4 WPPM function

Entry	Short Description	Section	Page
SizeDistribution			

6.1.7 Decimal Operators and Functions

Decimal data can be utilized with mathematical operators and functions. Operators are placed between two decimals data, such as the addition operator:

```
1| 1 + 1
```

Functions are indicated by a specific term followed by parentheses. The decimal value(s) are placed inside the parentheses. For instance, the square root function:

```
1| Sqrt(4)
```

The subsequent section will enumerate all operators and functions in Sindarin, categorized by type.

6.1.7.1 Decimal operators

Entry	**Short Description**
Division (/)	Divides two decimal data
Multiplication (*)	Multiplies two decimal data
Parentheses ( )	Parentheses in mathematics, used to order operations
Power (^)	A number raised to a power
Subtraction (-)	Subtract two decimal data
Sum (+)	Add two decimal data

6.1.7.2 General functions

Entry	**Short Description**
Sqrt	Square Root
Abs	Absolute
ln	The natural logarithm
log	The common logarithm
rad	Calculates the angle in radians from the angle in degrees
deg	Calculates the angle in degreed from the angle in radians

6.1.7.3 Trigonometric functions

Entry	**Short Description**
Acos	Arccosine
Asin	Arcsine
Atan	Arctangent
Cos	Cosine
Sin	Sine
Tan	Tangent

6.1.7.4 Special functions

Entry	**Short Description**
psVoigtH	Calculation of psVoigt FWHM from the Gaussian and Lorentzian FWHM
psVoigtEta	Calculation of psVoigt Eta from the Gaussian and Lorentzian FWHM
psVoigtBeta	Calculation of psVoigt integral breadth from the Gaussian and Lorentzian FWHM
psVoigtHBeta	Calculation of psVoigt FWHM from integrated breadth and eta of the psVoigt
psVoigtHGaus	Calculation of Gaussian FWHM from integrated breadth and eta of the psVoigt
psVoigtHLorentz	Calculation of Lorentzian FWHM from integrated breadth and eta of the psVoigt
randon(mean sd)	This function generates a random value with equal probability, centered around a specified base value (mean) with a given standard deviation (sd)
randonGauss(mean sd)	This function generates a random value following a Gaussian

	distribution, centered around a specified base value (mean) with a given standard deviation (sd)

Special functions require two decimal parameters, separated by space, within the function parentheses. An example is:

```
1| psVoigtH(0.43 0.184)
```

6.1.8 Decimal Constants

Entry	**Short Description**
E	The Euler number
CLight	The speed of light in vacuum
NaN	Not a Number
PI	The π constant
hPlanck	The Planck constant
NAvog	The Avogadro constant

6.1.9 Internal variables

Entry	**Short Description**
x	A decimal data representing the variable independent of the diffractogram derived from the experimental data, utilized in functions.
xl	A decimal data representing a point in one-dimensional Fourier space, calculated from the experimental data and the wavelength of the radiation. The unit of measurement is the same as that of the wavelength, which is angstrom.
xmin	A decimal array of length two denotes the point with the minimum x-value in the experimental data; the first decimal represents the x-coordinate, and the second indicates the intensity.
xmax	A decimal array of length two denotes the point with the maximum x-value in the experimental data; the first decimal represents the x-coordinate, and the second indicates the intensity.
ymin	A decimal array of length two denotes the point with the minimum y-value in the experimental data; the first decimal represents the x-coordinate, and the second indicates the intensity.
ymax	A decimal array of length two denotes the point with the maximum y-value in the experimental data; the first decimal represents the x-coordinate, and the second indicates the intensity.
xpeak	A decimal value corresponds to the peak's position in 2θ (or another horizontal unit of the diffractogram).
ipeak	The decimal value signifies the peak's intensity in diffractogram units.
dreflection	The interplanar spacing of the reflection, denoted as d, is measured in Å units.
fwhm	A decimal data is employed to compute the Profile Function with two or more parameters.
hkl	An integer array of length three represents the Miller indices of a reflection.

When an internal variable constitutes an array, its value may be utilized either as an array or by accessing the value at a specific array position. For instance, hkl[0] retrieves the value of h in the Miller index, while xmax[1] obtains the intensity value of the point with the maximum x-coordinate in the experimental data.

6.2 ALPHABETIC ENTRIES

In this section the words are presented in alphabetic order with systematic information useful to use these entries in Sindarin document and to understanding of behavior or action of each object and property. Each entry in dictionary has the following information, when applied: Declarative Syntax; Parameters; Description; Equations; Parents or owners; Children; Properties; References; Specific information; Internal Variables dependences. The decimal function and operators, decimal constants and internal variables it is not present here, they are in 6.1.7, 6.1.8 and 6.1.9 sections.

Common information table:

Declarative Syntax	
Type	
Parent(s)/Owner(s)	
References	
Short description	

Object-specific information table:

Object classification	
Parameters	
Internal variables dependences for parameters	
Properties	
Children	
Specific information	

Property-specific information table:

Data type	
Internal variables dependences	
Specific information	
Default value	

6.2.1 AbsorptionFunction

Declarative Syntax	`AbsorptionFunction (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	A generic decimal data that is used in calculation of intensity reflection.

Data type	Decimal
Internal variables dependences	
Specific information	
Default value	0

6.2.2 AngularDependency

Declarative Syntax	`AngularDependency (enum)[SinLike, QuadraticLike, FPALike]`
Type	Property
Parent(s)/Owner(s)	Split 105 Circle 73
References	(Cheary & Coelho, 1992; Enzo et al., 1988; Toraya, 1986)
Short description	Defines the x-dependent function type with three parameters for asymmetry peak correction.

Data type	bool
Internal variables dependences	-
Specific information	The AngularDependency property can only be assigned when there are three or four object parameters of the asymmetry correction object.
Default value	SinLike

If the propertier is SinLike, the dependency is a Quadratic function:

$$asym(x) = \mu_0 + \frac{\mu_1}{\sin\left(\frac{x}{2}\right)} + \frac{\mu_2}{\sin^2\left(\frac{x}{2}\right)}$$

If it is QuadraticLike:

$$asym(x) = \mu_0 + \mu_1[x - (90 + \mu_2)]^2$$

And, if it is FPALike:

$$asym(x) = \frac{\mu_0}{\sin(x)} + \mu_1\sqrt{\tan\left(\frac{x}{2}\right)} + \mu_2\tan(x)$$

6.2.3 Atoms

Declarative Syntax	`AtomSymbol (decimal)x (decimal)y (decimal)z (decimal)occ (decimal)uiso`
	
	`AtomSymbol (string)label (decimal)x (decimal)y (decimal)z (decimal)occ (decimal)uiso`

Type	Object
Parent(s)/Owner(s)	Cell 71
References	
Short description	Object that represents an atom inside the cell
Object classification	Crystallographic 61
Parameters	The label is an optional string data utilized for atom identification. The initial three decimal values represent the atom's relative position within the cell (x, y and z). This is succeeded by a decimal value indicating the atom's site occupancy in the crystallographic cell (occ). Subsequently, a decimal value denotes the isotropic atomic displacement parameter (uiso) in Å^2.
Internal variables dependences for parameters	
Properties	-
Children	-
Specific information	First word must be the atom symbol, for example, for oxygen atom must write O, or for sodium cation must write Na1+

6.2.4 AxialDivergence

Declarative Syntax	`AxialDivergence (decimal)laxial`
Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	Profile correction due the axial divergence of X-ray beam on the sample

Object classification	Profile 62
Parameters	Axial length of the X-ray beam on the sample divide by diffractometer radius
Internal variables dependences for parameters	
Properties	ConvolutionType (page 74)
Children	-
Specific information	For more information, see the Fundamental Parameters Approach (FPA) section, page 37

6.2.5 BackgroundFile

Declarative Syntax	`BackgroundFile (string)file[1+n]`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	-
Short description	Load a data from file(s) and apply it(them) to background

Object classification	Background Function 62
Parameters	File names to be load
Internal variables dependences for parameters	
Properties	-
Children	-
Specific information	Multiple background objects can be used simultaneously

6.2.6 Box

Declarative syntax	`Box (decimal)wBox`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94

References	(Cheary & Coelho, 1992)
Short description	A box profile shape

Object classification	Profile 62
Parameters	The width of the box in 2θ units
Internal variables dependences for parameters	
Properties	ConvolutionType (page 74)
Children	-
Specific information	The Box function is the base of the Defocusing, ReceivingSlitWidth, TargetWidth and Tilt profile functions.

6.2.7 Calculation

Declarative Syntax	`Calculation`
Type	Object
Parent(s)/Owner(s)	Root
References	-
Short description	This object has several data that control the calculation of diffractogram and the optimization routine

Object classification	Calculation Control 61
Parameters	-
Internal variables dependences for parameters	
Properties	Cycles 76 DerivationBothSide 79 CalculeProfileInPeak 70 RapidYRecalcule 99 Stabilization 106
Children	-
Specific information	This object is aways instantiated. If the user does not attribute explicitly in the text, the object assumes default values of the properties

6.2.8 CalculeProfileInPeak

Declarative syntax	`CalculeProfileInPeak (bool)`
Type	Property
Parent(s)/Owner(s)	Calculation 70
References	-
Short description	Enable the calculation of the profiles in each peak in the diffractogram. Otherwise, the calculation is performed in each Reflection.

Data type	Boolean
Internal variables dependences	-
Specific information	-
Default value	false

6.2.9 Capillary

Declarative Syntax	`Capillary (decimal)mur`
	
	`Capillary (decimal)mur (decimal)displ (decimal)dispv`

Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	(Gozzo et al., 2010; Sabine et al., 1998)
Short description	Enable corrections due the capillary sample in transmission diffraction geometry. These corrections are applied to the intensity and position of reflections due sample absorption. Additionally, capillary displacement can also be corrected.

Object classification	Diffraction 61
Parameters	A decimal data that is the effective linear absorption of the sample multiplied by capillary radius (muR). The other two decimal data that is the displacement of the sample form the goniometry center in two directions divided by diffractometer radius (dispL and dispV).
Internal variables dependences for parameters	
Properties	EnableIntensity 81 EnableShift 82
Children	-
Specific information	-

6.2.10 Cell

Declarative Syntax	`Cell (string)spaceGroup (decimal)latticeParameter[1,2,3,4,6]`
Type	Object
Parent(s)/Owner(s)	Phase 94
References	-
Short description	Create an object that represents the crystallographic cell

Object classification	Crystallographic 61
Parameters	The string represents the space group symbol, which can be expressed in either Hermann-Mauguin notation or Hall notation, as established by the International Table (IUCR). Accompanying this, the lattice parameters are provided in units of angstroms (Å), the number of decimal parameters depends on the lattice system.
Internal variables dependences for parameters	
Properties	Z 114
Children	Atoms 68 ReflectionHKL 100
Specific information	-

6.2.11 CenteredInMiddle

Declarative Syntax	`CenteredInMiddle (bool)`
Type	Property
Parent(s)/Owner(s)	Polynomial 95
References	
Short description	Set if the polynomial is calculated centered in the middle of the x range of experimental data

Data type	Bool
Internal variables dependences	
Specific information	
Default value	True for Experimental and Simulation root objects False for Data and DataSimulation root objects

6.2.12 Chebyshev

Declarative Syntax	`Chebyshev (decimal)coefficient[1+n]`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	
Short description	Chebyshev polynomial of the first kind

Object classification	Background Function 62
Parameters	Coefficients of the Chebyshev polynomial function
Internal variables dependences for parameters	
Properties	-
Children	-
Specific information	Multiple background objects can be used simultaneously

6.2.13 Circle

Declarative Syntax	`Circle (decimal)cutcir`
	
	`Circle (decimal)mu0 (decimal)mu1 (decimal)mu2`
	
	`Circle (decimal)mu0 (decimal)mu1 (decimal)mu2 (decimal)xmaxasy`

Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	The Circle profile function

Object classification	Profile 62
Parameters	The cutCir parameter is the limit of the calculation of Circle function. The parameters mu0, mu1, mu2 are used to calculate the limit of the circle function as a function of the 2θ. The parameter xMaxAsy is the maximum angle in 2θ that the Circle function is applied.
Internal variables dependences for parameters	
Properties	ConvolutionType 74

	Erro! Fonte de referência não encontrada. Erro! Indicador não definido.
Children	-
Specific information	The Circle is the base function of the AxialDivergence profile function

6.2.14 ConvolutionSMax

Declarative Syntax	`ConvolutionSMax (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	The maximum distance from the peak center (in Å) at which the profile in Fourier space is calculated for use in convolution.

Data type	Decimal
Internal variables dependences	
Specific information	ConvolutionXMax and ConvolutionSMax can not be instantiated simultaneously. If ConvolutionXMax and ConvolutionSMax is not signed by the user, it is used the ProfileXMax to calculate the convolution space of the profile
Default value	-

6.2.15 ConvolutionType

Declarative Syntax	`ConvolutionType (enum)[FFT, Numerical]`
Type	Property
Parent(s)/Owner(s)	Profile 62
References	
Short description	Set the type of convolution that is performed for the profile function

Data type	Enumerator
Internal variables dependences	-
Specific information	Numerical convolution can only be applied for profiles that has this possibility, the FPA functions. All profile functions can be convoluted by FFT approach.
Default value	Each profile has a default value

6.2.16 ConvolutionXMax

Declarative Syntax	`ConvolutionXMax (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	The maximum distance from the peak center (in units of the x-axis) at which the profile in diffraction space is calculated for use in convolution.

Data type	Decimal
Internal variables dependences	
Specific information	ConvolutionXMax and ConvolutionSMax can not be instantiated simultaneously. If neither ConvolutionXMax nor ConvolutionSMax is specified by the user, ProfileXMax will be utilized to calculate the convolution space of the profile
Default value	-

6.2.17 CosineFourier

Declarative Syntax	`CosineFourier (decimal)coefficient`$_{[1+n]}$
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	
Short description	A cosine Fourier series

Object classification	Background Function 62
Parameters	Coefficients of each term in the cosine Fourier series
Internal variables dependences for parameters	
Properties	-
Children	-
Specific information	Multiple background objects can be used simultaneously

6.2.18 CutOff

Declarative Syntax	`CutOff (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	The relative intensity used as a threshold for profile calculation

Data type	Decimal
Internal variables dependences	
Specific information	In Sindarin, profile calculation is restricted by two main terms: the CutOff and the ProfileXMax, whichever is reached first
Default value	0.001

6.2.19 Cycles

Declarative Syntax	`Cycles (int)`
Type	Property
Parent(s)/Owner(s)	Calculation 70
References	
Short description	The number of steps of optimization routine will be performed for walk command

Data type	Integer
Internal variables dependences	-
Specific information	The number of steps of the others optimization command is not modified
Default value	3

6.2.20 Data

Declarative Syntax	`Data (string)file[1+n]`
	
	`Data (string)label (string)file[1+n]`

Type	Object
Parent(s)/Owner(s)	Root

References	
Short description	The Data object serves as the primary entity for proposed calculations in general terms. Unlike the Experimental object, Data object can only have function objects (background objects). The main responsibility of the Data object includes importing experimental data from files outlined in the parameters

Object classification	Experimental data 61
Parameters	The label is an optional string data utilized to identify the experimental data and calculations. Following this, the file name of the experimental data, which is to be imported for calculation, is provided as a string data
Internal variables dependences for parameters	
Properties	WeightPoint
Children	Background Function 62 ExcludeRegion 83
Specific information	

6.2.21 DataSimulation

Declarative Syntax	`DataSimulation`
Type	Object
Parent(s)/Owner(s)	Root
References	
Short description	Although the DataSimulation object bears resemblance to the Data object, its functionality diverges; it does not import data from files but instead exports computed data to files

Object classification	Experimental data 61
Parameters	-
Internal variables dependences for parameters	
Properties	-
Children	Background Function 62 Interval 89
Specific information	Simulation and DataSimultaion objects can not have any refinable variable. Relevant data file information is contained within the Interval object.

6.2.22 DebyeLike

Declarative Syntax	`DebyeLike (decimal)length (decimal)w`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	A function based in Debye equation to x ray diffraction of two scatters.

Object classification	Background Function 62
Parameters	The distance between the scatters (length) and an arbitrary weight of the function to contribute to the background (w).
Internal variables dependences for parameters	
Properties	-
Children	-
Specific information	This function necessitates the inclusion of a radiation object. Multiple background objects can be used simultaneously

The equation is:

$$Y_i = w \frac{\sin\left[4\pi \frac{\sin(x_i)}{\lambda} length\right]}{4\pi \frac{\sin(x_i)}{\lambda}}$$

6.2.23 Defocusing

Declarative Syntax	`Defocusing (decimal)delr`
Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	Correction profile for defocus aberration, which arises when the receiving slit is not positioned within the focusing circle.

Object classification	Profile 62
Parameters	The decimal parameter delr represents the divergent equatorial angle (α), in degrees, multiplied by the ratio of the distance of the receiving slit to the goniometer circle (ΔR) to the radius of the diffractometer

	(R).
Internal variables dependences for parameters	
Properties	ConvolutionType 74
Children	-
Specific information	For more information, see the Fundamental Parameters Approach (FPA) section, page 37

6.2.24 DerivationBothSide

Declarative Syntax	`DerivationBothSide (bool)`
Type	Property
Parent(s)/Owner(s)	Calculation 70
References	
Short description	This property enables the use of double-sided calculation (plus and minus a delta) in the numerical calculation of the derivative of the diffractogram calculated with respect to a variable. While this increases the computation time, it significantly enhances the accuracy of the calculation. Therefore, it is highly recommended to keep this property set to true.

Data type	Boolean
Internal variables dependences	
Specific information	
Default value	true

6.2.25 Direction

Declarative Syntax	`Direction (arrayInt[3])`
Type	Property
Parent(s)/Owner(s)	VoigtianSize 110 VoigtianStrain 111
References	
Short description	The index of the main crystallographic direction of an anisotropic function.

Data type	Array of Integer
Internal variables dependences	

Specific information	
Default value	-

6.2.26 Displacement

Declarative Syntax	`Displacement (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	Displacement of the sample from the diffractometer circle center. It is used to peak position correction. The decimal value is the displacement of sample divided by diffractometer radius.

Data type	Decimal
Internal variables dependences	
Specific information	
Default value	0

6.2.27 DistributionType

Declarative Syntax	`DistributionType (enum)[Lognormal, Gamma]`
Type	Property
Parent(s)/Owner(s)	SizeDistribution 104
References	
Short description	Set the distribution function for the microstructure.

Data type	Enumerator
Internal variables dependences	
Specific information	
Default value	Lognormal

6.2.28 Ellipsoidal

Declarative Syntax	`Ellipsoidal (arrayInt[3])uvw (decimal)tau (decimal)k`
Type	Object
Parent(s)/Owner(s)	Phase 94
References	(Pecharsky & Zavalij, 2009)
Short description	March function to correct preferred orientation of the sample

Object classification	Diffraction 61
Parameters	The uvw parameter denotes the crystallographic direction associated with the preferred orientation. The Tau parameter represents the factor within the function that delineates the correlation between the correction in the principal direction and the perpendicular direction. The k parameter serves as a weighting factor employed when multiple preferred orientation functions are utilized.
Properties	-
Children	-
Specific information	Take careful to use this function, especially in quantitative phase analyses (QPA). Use MarchDollase object for QPA. Multiple preferred orientation objects can be used simultaneously.

The ellipsoidal function:

$$T_{hkl} = \frac{1}{N}\sum_{i=1}^{N}\left[1 + (\tau^2 - 1)\cos^2\left(\phi_{hkl}^{i}\right)\right]^{-\frac{1}{2}}$$

Where N is the total number of equivalent reflections, τ is the ratio between the length (or scaling factor) along the principal axis of the ellipsoid and the length along the perpendicular direction ($\tau = \frac{T_{\parallel}}{T_{\perp}}$) and ϕ_{hkl}^{i} is the angle between the principal axis of the ellipsoid and the direction of the i-th reflection of the equivalent reflection.

6.2.29 EnableIntensity

Declarative Syntax	`EnableIntensity (bool)`
Type	Property
Parent(s)/Owner(s)	Capillary 71
References	
Short description	Enable absorption intensity correction of reflections for capillary specimens

Data type	Boolean
Internal variables dependences	
Specific information	
Default value	true

6.2.30 EnableShift

Declarative Syntax	`EnableShift (bool)`
Type	Property
Parent(s)/Owner(s)	Capillary 71
References	
Short description	Enable shift correction of reflections due to the absorption for capillary specimens

Data type	Boolean
Internal variables dependences	
Specific information	
Default value	true

6.2.31 EquatorialDivergence

Declarative Syntax	`EquatorialDivergence (decimal)alphaeqdiv`
Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	Profile function to correct the effect of equatorial divergence of flat specimens.

Object classification	Profile 62
Parameters	The alphaeqdiv parameter represents the equatorial angle divergence of the primary beam in degrees.

Internal variables dependences for parameters	
Properties	ConvolutionType (page 74)
Children	
Specific information	For more information, see the Fundamental Parameters Approach (FPA) section, page 37

6.2.32 ExcludeRegion

Declarative Syntax	`ExcludeRegion (arrayDecimal[2])range[1+n]`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	
Short description	Exclude experimental points in specific regions of the experimental data

Object classification	Experimental data 61
Parameters	The parameters (range) are decimal arrays of length 2, where the two numbers in the array define the limits on the x-axis within which the experimental points are excluded.
Internal variables dependences for parameters	
Properties	
Children	
Specific information	

6.2.33 Experimental

Declarative Syntax	`Experimental (string)file[1+n]`
	
	`Experimental (string)label (string)file[1+n]`

Type	Object
Parent(s)/Owner(s)	Root
References	
Short description	For diffraction computations, the pivotal entity is the Experimental object. It encapsulates all ancillary data and objects utilized in the theoretical model to calculate the diffractogram. This object's chief

	role encompasses the importation of experimental data from files outlined in the parameters

Object classification	Experimental data 61
Parameters	The label is an optional string data utilized to identify the experimental data and calculations. Following this, the file name of the experimental data, which is to be imported for calculation, is provided as a string data
Internal variables dependences for parameters	-
Properties	AbsorptionFunction 66 UseAnomalousScattering 108 ConvolutionSMax 74 ConvolutionXMax 75 CutOff 76 Displacement 80 FFTResolution 85 IncidentAngle 88 MonochromatorTwoTheta 93 Polarization 95 ProfileXMax 97 SpecimenThickness 105 UseAnomalousScattering 108 VariableSlit 109 WeightPoint 113 Zero 114
Children	Background Function 62 Capillary 71 ExcludeRegion 83 Phase 94 Profile 62 Radiation 99 ScaterringFactor 101
Specific information	

6.2.34 Exponential

Declarative Syntax	`Exponential (decimal)wexp`
	
	`Exponential (decimal)wexp (decimal)cutexp`

Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94

References	(Cheary & Coelho, 1992)
Short description	The exponential profile function

Object classification	Profile 62
Parameters	Falta escrever
Internal variables dependences for parameters	
Properties	ConvolutionType 74
Children	
Specific information	The Exponential is the base function of the Transparency profile function

6.2.35 FFTResolution

Declarative Syntax	`FFTResolution (int)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	This property pertains to the relative point density used in calculating profiles for convolution via the Fast Fourier Transform approach, in relation to experimental data point density. If this property is set to 1, the point density matches the experimental data. If set to 2, the point density is double that of the experimental data.

Data type	Integer
Internal variables dependences	
Specific information	
Default value	4

6.2.36 Function

Declarative Syntax	`Function (decimal)func`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	

Short description	A user-defined function for background description.

Object classification	Background Function 62
Parameters	Func is a decimal parameter used to define the function and can use the expression capabilities of decimal data such as operators, functions, internal variables and internal constants.
Internal variables dependences for parameters	
Properties	
Children	
Specific information	Multiple background objects can be used simultaneously

6.2.37 FunctionPsVoigt

Declarative Syntax	`FunctionPsVoigt (decimal)x0 (decimal)int (decimal)fwhm (decimal) eta`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	
Short description	The psVoigt function can be utilized without a Peak object, such as ReflectionHKL, ReflectionXI, or PeakXI. This function is computed across the entire x-space.

Object classification	Background Function 62
Parameters	x0: the position of the maximum of the function int: the intensity, proportional to the area fwhm: the full width at half maximum eta: the linear combination parameter of Lorentz and Gauss functions
Internal variables dependences for parameters	
Properties	
Children	
Specific information	Multiple background objects can be used simultaneously

6.2.38 Gauss

Declarative Syntax	`Gauss (decimal)fwhm`

	`Gauss (decimal)gu (decimal)gv (decimal)gw (decimal)gz`
Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	The Gaussian function for profile peak

Object classification	Profile 62
Parameters	fwhm: the full width at half maximum gu, gv, gw and gz: the Caglioti parameters to angular dependency of fwhm (See The pseudo-Voigt function, page 38)
Internal variables dependences for parameters	
Properties	ConvolutionType 74
Children	Split 105
Specific information	

6.2.39 GaussWavelength

Declarative Syntax	`GaussWavelength (decimal)lambda (decimal)fwhmLambda`
Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	The Gaussian function in terms of wavelength of the radiation

Object classification	Profile 62
Parameters	lambda: The wavelength of the radiation in Å unit. fwhmLambda: the full width at half maximum in Å unit.
Internal variables dependences for	

parameters	
Properties	ConvolutionType 74
Children	Split 105
Specific information	For more information on profiles in terms of wavelength, see the Radiation modeling chapter, page 44

6.2.40 IncidentAngle

Declarative Syntax	`IncidentAngle (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	Defines the angle of the incident beam in an asymmetric diffractometer geometry.

Data type	Decimal
Internal variables dependences	
Specific information	IncidentAngle can not be simultaneously declared with the Capillary object
Default value	-

6.2.41 Interpolation

Declarative Syntax	`Interpolation (arrayDecimal[1,2])points[2+n]`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	
Short description	Describe the background in terms of interpolation of points.

Object classification	Background Function 62
Parameters	points: arrays of one or two elements that describe points in the chart. When the array contains only one decimal data, it represents the x-coordinate, and the y-coordinate is obtained from experimental data. When the array contains two decimal data, they represent the x and y coordinates, respectively.
Internal variables dependences for parameters	
Properties	

Children	
Specific information	Multiple background objects can be used simultaneously. The Interpolation must have at least two parameters.

6.2.42 Interval

Declarative Syntax	`Interval (decimal)xi (decimal)xf (decimal)step`
	
	`Interval (decimal)xi (decimal)xf (decimal)step (string)filename`
	
	`Interval (decimal)xi (decimal)xf (decimal)step (string)fileName (decimal)noise`

Type	Object
Parent(s)/Owner(s)	Data 76 DataSimulation 77
References	
Short description	Describes the information for simulated data to be written to a file.

Object classification	Experimental data
Parameters	xi: the initial x-coordinate of the data xf: the final x-coordinate of the data step: the difference between two consecutive points in x-coordinate filename: the name of the file where the simulated data is written. noise: to include noise in simulated data. Its value is the standard deviation of the Gaussian distribution.
Internal variables dependences for parameters	
Properties	
Children	
Specific information	More the one Interval object can be declared

6.2.43 Inverse

Declarative Syntax	`Inverse (decimal)coeff`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	
Short description	An inverse function, proportional to 1/x, to describe the background.

Object classification	Background Function 62
Parameters	coeff: A multiplier of the function
Internal variables dependences for parameters	
Properties	
Children	
Specific information	Multiple background objects can be used simultaneously

The inverse function:

$$inverse(x) = coeff\frac{1}{x}$$

6.2.44 InverseProfile

Declarative Syntax	`InverseProfile (decimal)cutInv`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	The inverse profile function

Object classification	Profile 62
Parameters	cutInv: is the limit on x-coordinate to calculate the function
Properties	
Children	
Specific information	The Inverse is the base function of the EquatorialDivergence profile function.

6.2.45 Lorentz

Declarative Syntax	`Lorentz (decimal)fwhm`
	
	`Lorentz (decimal)lx (decimal)ly`

Type	Object
Parent(s)/Owner(s)	Radiation99

	Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	The Lorentzian function for profile peak

Object classification	Profile 62
Parameters	fwhm: the full width at half maximum lx and ly: the parameters to angular dependency of fwhm (See The pseudo-Voigt function, page 38
Properties	ConvolutionType 74
Children	Split 105
Specific information	

6.2.46 LorentzWavelength

Declarative Syntax	`LorentzWavelength (decimal)lambda (decimal)fwhmLamb`
Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	The Lorentzian function in terms of wavelength of the radiation.

Object classification	Profile 62
Parameters	lambda: The wavelength of the radiation in Å unit. fwhmLambda: the full width at half maximum in Å unit.
Properties	ConvolutionType 74
Children	Split 105
Specific information	For more information on profiles in terms of wavelength, see the Radiation modeling chapter, page 44

6.2.47 ManageHighCorrelation

Declarative Syntax	`ManageHighCorrelation (bool)`

Type	Property
Parent(s)/Owner(s)	Calculation 70
References	
Short description	Enable the automatic management of variables with high correlation.

Data type	Bool
Internal variables dependences	
Specific information	
Default value	true

6.2.48 MarchDollase

Declarative Syntax	`MarchDollase (arrayInt[3])uvw (decimal)tau (decimal)k`
Type	Object
Parent(s)/Owner(s)	Phase 94
References	(Dollase, 1986)
Short description	March-Dollase function to correct preferred orientation of the sample

Object classification	Diffraction 61
Parameters	The uvw parameter denotes the crystallographic direction associated with the preferred orientation. The Tau parameter represents the factor within the function that delineates the correlation between the correction in the principal direction and the perpendicular direction. The k parameter serves as a weighting factor employed when multiple preferred orientation functions are utilized.
Properties	-
Children	-
Specific information	Multiple preferred orientation objects can be used simultaneously

6.2.49 MeanStandardDeviation

Declarative Syntax	`MeanStandardDeviation (bool)`
Type	Property
Parent(s)/Owner(s)	SizeDistribution 104
References	
Short description	When this property is enabled, the two parameters of the SizeDistribution object directly represent the mean and the standard deviation of the distribution. When it is disabled, they correspond to the parameters used directly in the distribution equation. See the equations below for more details.

Data type	bool
Internal variables dependences	-
Specific information	
Default value	true

In the disabled case, the mean and standard deviation of the distribution can be calculated as follows:

For Lognormal distribution:

$$mean = e^{\mu+\frac{\sigma^2}{2}}$$

$$sd = \sqrt{e^{2\mu+\sigma^2}\left(e^{\sigma^2}-1\right)}$$

For Gamma distribution:

$$mean = \mu$$

$$sd = \frac{\mu}{\sigma}$$

6.2.50 MonochromatorTwoTheta

Declarative Syntax	`MonochromatorTwoTheta (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	The angle of the crystal monochromator corresponds to a specific parameter utilized in the calculation of peak intensity

Data type	Decimal
Internal variables dependences	
Specific information	Automatically enable the Lorentz correction
Default value	-

6.2.51 Multiplicity

Declarative Syntax	`Multiplicity (int)`
Type	Property
Parent(s)/Owner(s)	ReflectionHKL 100

References	
Short description	Manually define the multiplicity of a reflection.

Data type	Integer
Internal variables dependences	
Specific information	If the space group of the cell is specified, the multiplicity of the reflection is automatically calculated. However, this property allows the user to manually define the multiplicity.
Default value	-

6.2.52 PeakXI

Declarative Syntax	`PeakXI (decimal)x0 (decimal)inten`
Type	Object
Parent(s)/Owner(s)	Phase 94
References	
Short description	Represents a diffraction peak in the diffractogram.

Object classification	Diffraction 61
Parameters	x0: the position of the peak, measured in units along the horizontal axis of the diffractogram inten: intensity of the peak related to the area.
Properties	
Children	
Specific information	Unlike ReflectionXI, PeakXI ignores all information regarding radiation objects, such as the number of radiation lines, relative intensities, and profiles.

6.2.53 Phase

Declarative Syntax	`Phase`
	
	`Phase (string)label`

Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	Object that represents a phase in the sample

Object classification	Diffraction 61
Parameters	The optional string parameter is a label to identify the phase

Properties	Scale 101
Children	Cell 71 **Erro! Fonte de referência não encontrada. Erro! Indicador não definido.** MarchDollase 92 PeakXI 94 Profile 62 ReflectionXI 100
Specific information	

6.2.54 Polarization

Declarative Syntax	`Polarization (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	The degree of polarization of the incident beam is quantified on a scale where a value of 0 indicates no polarization, and a value of 1 denotes complete polarization. It is used in calculation of peak intensity

Data type	Decimal
Internal variables dependences	
Specific information	If Polarization is declared, the Lorentz correction is automatically enabled
Default value	-

6.2.55 Polynomial

Declarative Syntax	`Polynomial (decimal)coeff[1+n]`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103 Data 76 DataSimulation 77
References	
Short description	A polynomial function

Object classification	Background Function 62
Parameters	The decimal parameters are the coefficients of the polynomial.
Properties	CenteredInMiddle 72
Children	

Specific information	Multiple background objects can be used simultaneously

6.2.56 ProfileFunction

Declarative Syntax	`ProfileFunction (decimal)func`
	
	`ProfileFunction (decimal)func (decimal)fwhm`

Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	A user-defined function for peak profile

Object classification	Profile 62
Parameters	func: A decimal parameter used to define the function, utilizing the expression capabilities of decimal data, including operators, functions, variables, internal variables, and internal constants. fwhm: The Full Width at Half Maximum, which can be described in terms of a decimal function and referenced in the main function of the profile using the internal variable fwhm
Properties	
Children	Split 105
Specific information	

6.2.57 ProfileFunctionFT

Declarative Syntax	`ProfileFunctionFT (decimal)funcreal`
	
	`ProfileFunctionFT (decimal)funcreal (decimal)funcimag`

Type	Object
Parent(s)/Owner(s)	Radiation99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	A user-defined function for peak profile in fourier space.

Object classification	Profile 62
Parameters	funcreal and funcimag: Decimal parameters used to define the functions of real and imaginary parts. It possible to use the expression capabilities of decimal data, including operators, functions, variables, internal variables, and internal constants.
Properties	
Children	
Specific information	

6.2.58 ProfileXMax

Declarative Syntax	`ProfileXMax (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	The maximum distance from the peak center used as a threshold value for profile calculation

Data type	Decimal
Internal variables dependences	
Specific information	In Sindarin, profile calculation is restricted by two main terms: the CutOff and the ProfileXMax, whichever is reached first
Default value	4

6.2.59 psVoigt

Declarative Syntax	`psVoigt (decimal)fwhm (decimal)eta`
	
	`psVoigt (decimal)gu (decimal)gv (decimal)gw (decimal)gz (decimal)lx (decimal)ly`
	
	`psVoigt (decimal)psu (decimal)psv (decimal)psw (decimal)psz (decimal)eta0 (decimal)eta1 (decimal)eta2`

Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100

	ReflectionHKL 100 PeakXI 94
References	(Caglioti et al., 1958; Thompson et al., 1987; Wertheim et al., 1974)
Short description	The pseudo-Voigt function to profile modeling. It is a linear combination of Gaussian function and Lorentzian function to perform a numeric approximate of Voigt function.

Object classification	Profile 62
Parameters	The decimal parameter fwhm represents the Full Width at Half Maximum, while the decimal parameter eta signifies the linear combination factor of Gaussian and Lorentzian functions. The FWHM can be computed as a function of 2θ. In second declarative syntax, according to Thompson et al., 1987 (pseudo-Voigt TCHZ), the first four parameters calculate the FWHM of the Gaussian component, and the following two parameters determine the Lorentzian FWHM. The third declaration utilizes the relationship between the FWHM of the peak (the final FWHM, HpsV) and 2θ, as described by Caglioti et al., 1957, and the linear mixing factor η is a quadratic function of 2θ. Further details can be found in The pseudo-Voigt function chapter on page 38.
Properties	ConvolutionType 74
Children	Split 105
Specific information	

6.2.60 psVoigtWavelength

Declarative Syntax	`psVoigtWavelength (decimal)lambda (decimal)fwhmLamb (decimal)eta`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	The pseudo-Voigt function in terms of wavelength of the radiation

Object classification	Profile 62
Parameters	lambda: The wavelength of the radiation in Å unit. fwhmLambda: the full width at half maximum in Å unit. eta: the linear combination parameter of Lorentzian and Gaussian functions
Properties	ConvolutionType 74

Children	Split 105
Specific information	For more information on profiles in terms of wavelength, see the Radiation modeling chapter, page 44

6.2.61 Radiation

Declarative Syntax	`Radiation (decimal)lambda (decimal)intensity`
Type	Object
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	An X-ray radiation.

Object classification	Diffraction 61
Parameters	The lambda parameter signifies the wavelength of the radiation in decimal form. The intensity parameter quantifies the relative intensity of this spectral line, applicable when multiple radiations are employed.
Properties	
Children	Profile 62
Specific information	Multiples radiation objects can be instantiated per Experimental object

6.2.62 RapidYRecalcule

Declarative Syntax	`RapidyRecalcule (bool)`
Type	Property
Parent(s)/Owner(s)	Calculation 70
References	[ref]
Short description	Use numerical approximation calculation to estimate the theoretical diffractogram after a full calculation cycle.

Data type	Boolean
Internal variables dependences	
Specific information	This significantly increase the performance but decrease the accuracy, so use this with caution.
Default value	false

6.2.63 ReceivingSlitWidth

Declarative Syntax	`ReceivingSlitWidth (decimal)wslitrec`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	Function to correct the effect of the width of the receive slit on diffraction peak profile.

Object classification	Profile 62
Parameters	The wslitrec parameter is width of the receive slit divided by diffractometer radius.
Properties	ConvolutionType (page 74)
Children	
Specific information	For more information, see the Fundamental Parameters Approach (FPA) section, page 37

6.2.64 ReflectionHKL

Declarative Syntax	`ReflectionHKL (arrayInt[3])hkl (decimal)inten`
Type	Object
Parent(s)/Owner(s)	Cell 71
References	
Short description	Represent a reflection in a cell in terms of Miller index

Object classification	Crystallographic 61
Parameters	The hkl parameter is an array consisting of three integers that represent the Miller indices of the reflection. The inten parameter denotes an arbitrary value of the reflection's intensity.
Properties	Multiplicity 93 Shift 102
Children	Profile 62
Specific information	

6.2.65 ReflectionXI

Declarative Syntax	`ReflectionXI (decimal)x0 (decimal)inten`
Type	Object
Parent(s)/Owner(s)	Phase
References	
Short description	Represent a reflection in a phase in terms of diffraction angle.

Object classification	Diffraction 61
Parameters	The x0 parameter represents the position of the reflection in decimal form, measured in units along the horizontal axis of the diffractogram. The inten parameter indicates an arbitrary value corresponding to the intensity of the reflection related to the area.
Properties	Multiplicity 93
Children	Profile 62
Specific information	The horizontal position of the reflection in the diffractogram is determined in relation to the main radiation line. The peaks from other radiation lines of this reflection are calculated relative to this position. Unlike PeakXI, ReflectionXI uses all information regarding radiation objects, such as the number of radiation lines, relative intensities, and profiles.

6.2.66 Scale

Declarative Syntax	`Scale (decimal)`
Type	Property
Parent(s)/Owner(s)	Phase 94
References	
Short description	An arbitrary scale factor. Its value multiplies the intensity of all peaks in the phase.

Data type	Decimal
Internal variables dependences	
Specific information	
Default value	1

6.2.67 ScaterringFactor

Declarative Syntax	`ScaterringFactor (string)atom (decimal)fn1 (decimal)fn2`
Type	Object
Parent(s)/Owner(s)	Experimental 83

	Simulation 103
References	
Short description	User defined anomalous scattering factor for a specific atom

Object classification	Diffraction 61
Parameters	atom: a string to identify the chemistry element, must be the symbol of the chemical element. fn1 and fn2: atomic scattering factor components (dispersion coefficients).
Properties	
Children	
Specific information	The fn1 and fn2 coefficients are, by default, obtained from the data of (Henke et al., 1993) when not declared by the user.

6.2.68 Shape

Declarative Syntax	`Shape (enum)[Sphere, MarchDollase, Ellipsoidal, Cylinder] //for VoigtianSize`
	
	`Shape (enum)[Sphere, MarchDollase, Ellipsoidal] //for VoigtianStrain`
	
	`Shape (enum)[Sphere, Cube, Tetrahedron, Octahedron] //for SizeDistribution`

Type	Property
Parent(s)/Owner(s)	SizeDistribution 104 VoigtianSize 110 VoigtianStrain 111
References	
Short description	The shape of the crystals or of the spatial distribution of microstrain.

Data type	Enumerator
Internal variables dependences	
Specific information	
Default value	Sphere

6.2.69 Shift

Declarative Syntax	`Shift (decimal)`
Type	Property

Parent(s)/Owner(s)	ReflectionHKL
References	
Short description	An arbitrary shift of the reflection position in x-coordinate units.

Data type	Decimal
Internal variables dependences	
Specific information	
Default value	0

6.2.70 ShiftFT

Declarative Syntax	`ShiftFT (decimal)deltax`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Mendenhall et al., 2015)
Short description	A profile function in Fourier space to apply an arbitrary shift of the reflection position.

Object classification	Profile 62
Parameters	deltax: the shift in the position in x-coordinate units.
Properties	
Children	
Specific information	

6.2.71 Simulation

Declarative Syntax	`Simulation`
Type	Object
Parent(s)/Owner(s)	Root
References	
Short description	Although the Simulation object bears resemblance to the Experimental object, its functionality diverges; it does not import data from files but instead exports computed data to files

Object classification	Experimental data 61
Parameters	-
Properties	AbsorptionFunction 66 UseAnomalousScattering 108 ConvolutionSMax 74 ConvolutionXMax 75 CutOff 76 Displacement 80 FFTResolution 85 IncidentAngle 88 MonochromatorTwoTheta 93 Polarization 95 ProfileXMax 97 SpecimenThickness 105 UseAnomalousScattering 108 VariableSlit 109 Zero 114
Children	Background Function 62 Capillary 71 Interval 89 Phase 94 Profile 62 Radiation 99 ScaterringFactor 101
Specific information	Simulation and DataSimultaion objects can not have any refinable variable. Relevant data file information is contained within the Interval object.

6.2.72 SizeDistribution

Declarative Syntax	`SizeDistribution (decimal)mu (decimal)sigma`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Scardi & Leoni, 2002)
Short description	Peak profile function to model crystal size distribution

Object classification	Profile 62
Parameters	The mean cristal size (mu) in Å unit and the standard deviation of the distribution (sigma) in Å unit.
Properties	Shape 102 DistributionType 80
Children	-
Specific information	The crystal size is defined as the cube root of the volume of the geometric object.

6.2.73 SpecimenThickness

Declarative Syntax	`SpecimenThickness (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	It is the effective linear absorption coefficient multiplied by specimen thickness. It is used for intensity correction, specifically the absorption term, when the specimen is thin.

Data type	Decimal
Internal variables dependences	
Specific information	SpecimenThickness can not be simultaneously declared with the Capillary object
Default value	-

6.2.74 Split

Declarative Syntax	`Split (decimal)mu`
	
	`Split (decimal)mu (decimal)twoThetaMax`
	
	`Split (decimal)mu0 (decimal)mu1 (decimal)mu2`
	
	`Split (decimal)mu0 (decimal)mu1 (decimal)mu2 (decimal)twoThetaMax`

Type	Object
Parent(s)/Owner(s)	Voigtian function 62 ProfileFunction 96
References	
Short description	An empirical function used to modify the profile function and describe profile asymmetry

Object classification	Profile 62
Parameters	The Decimal parameter μ (mu) represents the asymmetry factor, which is defined as the relative difference between the Half Width at Half Maximum (HWHM) on the right side and left side of the peak. The three decimal parameters μ_0 (mu0), μ_1 (mu1) and μ_2 (mu2) are used to calculate the asymmetry factor as a function of the peak position. The type of peak position dependence function is determined by the QuadraticLike property. For more information, refer to the description of this property in the dictionary entry.

	The decimal parameter twoThetaMax is the maximum peak position at which the asymmetry is applied.
Properties	**Erro! Fonte de referência não encontrada. Erro! Indicador não definido.**
Children	-
Specific information	μ must be between -1 and 1, excluding.

Definition of the asymmetry factor μ:

$$\mu = \frac{HWHM_{right} - HWHM_{left}}{FWHM}$$

Where FWHM é the Full Width at Half Maximum of the peak and $HWHM_{rigth}$ and $HWHM_{left}$ are the Half Width at Half Maximum of each side of the peak. If μ = 0 the peak is symmetric.

6.2.75 Stabilization

Declarative Syntax	`Stabilization (enum)[Marquardt, Damping or None]`
Type	Property
Parent(s)/Owner(s)	Calculation 70
References	(Marquardt, 1963)
Short description	Definition of Stabilization Routines for Divergent Nonlinear Least Squares (NLLS) Procedures

Data type	Enumerator
Internal variables dependences	-
Specific information	Marquardt: the Marquardt routine is employed to try find new values for the fit variables within the current NLLS cycle. Damping: a successive reduction of the variable delta is performed to try find a new values for the fit variables within the current NLLS cycle. None: no stabilization routine is applied.
Default value	Marquardt

6.2.76 TargetWidth

Declarative Syntax	`TargetWidth (decimal)wtar`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103

	Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	Function to correct profile effect from x ray source with a specific width.

Object classification	Profile 62
Parameters	The wtar parameter represents the ratio between the X-ray tube target's projected focal-line width and the diffractometer radius.
Properties	ConvolutionType (page 74)
Children	
Specific information	For more information, see the Fundamental Parameters Approach (FPA) section, page 37

6.2.77 Tau

Declarative Syntax	`Tau (decimal)`
Type	Property
Parent(s)/Owner(s)	VoigtianSize 110 VoigtianStrain 111
References	
Short description	Tau is the ratio defined as the quotient of the value of microstructures in the direction perpendicular to a given reference direction to that in the reference direction itself. A ratio of 1 denotes isotropy within the microstructure. A ratio greater than 1 suggests that the spatial form assumes a disc-like shape, whereas a ratio less than 1 indicates a needle-like spatial characteristic.

Data type	Decimal
Internal variables dependences	
Specific information	
Default value	-

6.2.78 Tilt

Declarative Syntax	`Tilt (decimal)delt`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103

	Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	The function to correct profile effect due specimen tilt.

Object classification	Profile 62
Parameters	delt: represents the ratio between the variation in the height of the sample surface and the diffractometer radius.
Properties	ConvolutionType (page 74)
Children	
Specific information	For more information, see the Fundamental Parameters Approach (FPA) section, page 37

6.2.79 Transparency

Declarative Syntax	`Transparency (decimal)mu`
	
	`Transparency (decimal)mu (decimal)thickness`

Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Cheary & Coelho, 1992)
Short description	The function to correct profile effect due specimen transparency.

Object classification	Profile 62
Parameters	mu: represents the effective linear absorption coefficient multiplied by the diffractometer radius. thickness: represents the ratio between the thickness of the sample and the diffractometer radius. This parameter is used only if the sample is thin relative to the depth of the X-ray beam
Properties	ConvolutionType (page 74)
Children	
Specific information	For more information, see the Fundamental Parameters Approach (FPA) section, page 37

6.2.80 UseAnomalousScattering

Declarative Syntax	`UseAnomalousScattering (bool)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	Enable the anomalous atomic scattering to be applied in structure factor calculation

Data type	Boolean
Internal variables dependences	-
Specific information	
Default value	true

6.2.81 VariableSlit

Declarative Syntax	`VariableSlit (bool)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	(Bowden & Ryan, 1991; Kimmel, 1987)
Short description	When activated, a correction attributable to a variable divergence slit is implemented on the peak intensity. Within the Sindarin, this adjustment is incorporated into the absorption term of Equation 2 (page 34).

Data type	Boolean
Internal variables dependences	
Specific information	
Default value	false

6.2.82 Voigt

Declarative Syntax	`Voigt (decimal)fwhmG (decimal)fwhmL`

	`Voigt (decimal)gu (decimal)gv (decimal)gw (decimal)gz (decimal)lx (decimal)ly`
	
	`Voigt (decimal)vu (decimal)vv (decimal)vw (decimal)vz (decimal)eta0 (decimal)eta1 (decimal)eta2`

Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	The Voigt function as peak profile.

Object classification	Profile 62
Parameters	The decimal parameter fwhmG denotes the Full Width at Half Maximum (FWHM) of the Gaussian function. Similarly, the decimal parameter fwhmL indicates the FWHM of the Lorentzian function. The second and third declarations correspond to those outlined in the psVoigt entry (page 97). When employing the mixing function factor η, the FWHM values for both Gaussian and Lorentzian functions are computed in accordance with the methodology described by Thompson et al., 1987. However, the calculation involves a convolution of a Lorentzian and a Gaussian, rather than a linear combination.
Properties	ConvolutionType 74
Children	-
Specific information	

6.2.83 VoigtianSize

Declarative Syntax	`VoigtianSize (decimal)size (decimal)eta`
	
	`VoigtianSize (decimal)size (decimal)eta (decimal)k`

Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Langford, 1980)

Short description	A Voigtian function (Voigt or pseudo-Voigt) whose angular dependence follows the effect of crystal size for peak width. This model assumes that the effect of size on broadening can have both a Gaussian and a Lorenzian part.

Object classification	Profile 62
Parameters	The size decimal parameter is the mean crystal size or apparent mean crystal size, depend on Shape property value (see below), in Å unit. The decimal parameters η (eta) is the linear combination factor of Gaussian and Lorentzian functions. The decimal parameter k is used as weight when more the one VoigtianSize is used in the model.
Properties	ConvolutionType 74 Direction 79 Shape 102 Tau 107 Width 113
Children	Split 105
Specific information	The Tau and Width properties are mutually exclusive.

The size parameter is a measurement that depends on the shape property:

1. Sphere: Refers to cubic root of the volume with constant Scherrer $K_\beta = 1.0747$ calculated by integral breath method (Langford & Wilson, 1978).
2. Ellipsoidal or MarchDollase: Corresponds to the length of the principal axis of the ellipsoid, in terms of the apparent mean size.
3. Cylinder: Refers to the length of the principal axis, i.e., the height of the cylinder (Langford & Louër, 1982).

Note on anisotropy and equivalent reflections:

An observed reflection in a diffractogram corresponds to a set of equivalent reflections aligned along different crystallographic directions. When using an anisotropic model for microstructure - except in the case of symmetrized spherical harmonics - this can lead to the issue that reflections within the same family may exhibit varying microstructural values due to their directional differences.

Typically, the tridimensional object and its spatial orientation are chosen to conserve the symmetry between equivalents reflections in specific lattice system.

6.2.84 VoigtianStrain

Declarative Syntax	`VoigtianStrain (decimal)strain (decimal)eta`
	
	`VoigtianStrain (decimal)strain (decimal)eta (decimal)k`

Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	(Adler & Houska, 1979)
Short description	A Voigtian function (Voigt or pseudo-Voigt) whose angular dependence follows the effect of lattice microstrain for peak width. This model assumes that the effect of microstrain on broadening can have both a Gaussian and a Lorenzian part.

Object classification	Profile 62
Parameters	The strain decimal parameter is the maximum (upper-limit) microstrain (MMS). The decimal parameters η (eta) is the linear combination factor of Gaussian and Lorentzian functions. The decimal parameter k is used as weight when more the one VoigtianStrain is used in the model.
Properties	ConvolutionType 74 Direction 79 Shape 102 Tau 107 Width 113
Children	Split 105
Specific information	The Tau and Width properties are mutually exclusive.

See the note on anisotropy of VoigtianSize.

6.2.85 VoigtWavelength

Declarative Syntax	`VoigtWavelength (decimal)lambda (decimal)fwhmLamb (decimal)eta`
Type	Object
Parent(s)/Owner(s)	Radiation 99 Experimental 83 Simulation 103 Phase 94 ReflectionXI 100 ReflectionHKL 100 PeakXI 94
References	
Short description	The Voigt function in terms of wavelength of the radiation

Object classification	Profile 62
Parameters	lambda: The wavelength of the radiation in Å unit. fwhmLambda: the full width at half maximum in Å unit. eta: the linear mixture parameter of Lorentz and Gauss functions
Properties	ConvolutionType 74
Children	Split 105
Specific information	The eta parameter is used to calculate the FWHM of Lorentzian and Gaussian functions according to Thompson et al., 1987 (pseudo-Voigt TCHZ). However, the calculation involves a convolution of a Lorentzian and a Gaussian, rather than a linear combination. For more information on profiles in terms of wavelength, see the Radiation modeling chapter, page 44.

6.2.86 WeightPoint

Declarative Syntax	`WeightPoint (enum)[Unit, InverseY, InverseE, Weight]`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Data 76
References	
Short description	Specify the type of weight of each point in optimization routine.

Data type	Integer
Internal variables dependences	-
Specific information	
Default value	Assign by default 'Unit' weighting to the Data object and 'InverseY' weighting to the Experimental object. Should the experimental data error be available, the default setting is 'InverseE' for both objects.

6.2.87 Width

Declarative Syntax	`Width (decimal)`
Type	Property
Parent(s)/Owner(s)	VoigtianSize 110 VoigtianStrain 111
References	
Short description	The value of the microstructure in the perpendicular direction to the specified direction

Data type	Decimal
Internal variables	

dependences	
Specific information	
Default value	-

6.2.88 Z

Declarative Syntax	`Z (int)`
Type	Property
Parent(s)/Owner(s)	Cell
References	
Short description	The multiplicity of the chemical formula denoted by the total number of atoms contained within the unit cell.

Data type	Integer
Internal variables dependences	
Specific information	This property is not employed in the calculations of the diffractogram but is reserved for the interpretation of results.
Default value	1

6.2.89 Zero

Declarative Syntax	`Zero (decimal)`
Type	Property
Parent(s)/Owner(s)	Experimental 83 Simulation 103
References	
Short description	Correction of the angular position of the diffractometer arms.

Data type	Decimal
Internal variables dependences	
Specific information	
Default value	0

7 BIBLIOGRAPHY

Adler, T., & Houska, C. R. (1979). Simplifications in the x-ray line-shape analysis. *Journal of Applied Physics*, *50*(5), 3282–3287. https://doi.org/10.1063/1.326368

Bowden, M. E., & Ryan, M. J. (1991). Comparison of Intensities from Fixed and Variable Divergence X-Ray Diffraction Experiments. *Powder Diffraction*, *6*(2), 78–81. https://doi.org/10.1017/S0885715600017048

Caglioti, G., Paoletti, A., & Ricci, F. P. (1958). Choice of collimators for a crystal spectrometer for neutron diffraction. *Nuclear Instruments*, *3*(4), 223–228. https://doi.org/10.1016/0369-643X(58)90029-X

Cheary, R. W., & Coelho, A. (1992). A fundamental parameters approach to X-ray line-profile fitting. *Journal of Applied Crystallography*, *25*(2), 109–121. https://doi.org/10.1107/S0021889891010804

Cheary, R. W., Coelho, A. A., & Cline, J. P. (2004). Fundamental parameters line profile fitting in laboratory diffractometers. *Journal of Research of the National Institute of Standards and Technology*, *109*(1), 1. https://doi.org/10.6028/jres.109.002

Dollase, W. A. (1986). Correction of Intensities for Preferred Orientation in Powder Diffractometry: Application of the March Model. In *J. Appl. Cryst* (Vol. 19).

Enzo, S., Fagherazzi, G., Benedetti, A., & Polizzi, S. (1988). A Profile-Fitting Procedure for Analysis of Broadened X-ray Diffraction Peaks. I. Methodology. In *J. Appl. Cryst* (Vol. 21).

Gozzo, F., Cervellino, A., Leoni, M., Scardi, P., Bergamaschi, A., & Schmitt, B. (2010). Instrumental profile of MYTHEN detector in Debye-Scherrer geometry. *Zeitschrift Fur Kristallographie*, *225*(12), 616–624. https://doi.org/10.1524/zkri.2010.1345

Henke, B. L., Gullikson, E. M., & Davis, J. C. (1993). X-Ray Interactions: Photoabsorption, Scattering, Transmission, and Reflection at E = 50-30,000 eV, Z = 1-92. *Atomic Data and Nuclear Data Tables*, *54*(2), 181–342. https://doi.org/10.1006/adnd.1993.1013

Kimmel, G. (1987). High Quality X-Ray Diffraction Data Using an Adjustable Divergence Slit and Thin Samples. *Powder Diffraction*, *2*(1), 22–27. https://doi.org/10.1017/S0885715600012173

Langford, J. I. (1980). Accuracy of crystallite size and strain determined from the integral breadth of powder diffraction lines. *Accuracy in Powder Diffraction*, 255–269.

Langford, J. I., & Louër, D. (1982). Diffraction line profiles and Scherrer constants for materials with cylindrical crystallites. *Journal of Applied Crystallography*, *15*(1), 20–26. https://doi.org/10.1107/S0021889882011297

Langford, J. I., & Wilson, A. J. C. (1978). Scherrer after sixty years: A survey and some new results in the determination of crystallite size. *Journal of Applied Crystallography*, *11*(2), 102–113. https://doi.org/10.1107/S0021889878012844

Marquardt, D. W. (1963). AN ALGORITHM FOR LEAST-SQUARES ESTIMATION OF NONLINEAR PARAMETERS*. In *J. Soc. INDUST. APPL. MATH* (Vol. 11, Issue 2).

Mendenhall, M. H., Mullen, K., & Cline, J. P. (2015). An Implementation of the Fundamental Parameters Approach for Analysis of X-ray Powder Diffraction Line Profiles. *Journal of Research of the National Institute of Standards and Technology*, *120*, 223. https://doi.org/10.6028/jres.120.014

Pecharsky, V. K., & Zavalij, P. Y. (2009). Fundamentals of diffraction. In V. K. Pecharsky & P. Y. Zavalij (Eds.), *Fundamentals of Powder Diffraction and Structural Characterization of Materials* (2nd ed., pp. 99–260). Springer.

Sabine, T. M., Hunter, B. A., Sabine, W. R., & Ball, C. J. (1998). Analytical Expressions for the Transmission Factor and Peak Shift in Absorbing Cylindrical Specimens. In *J. Appl. Cryst* (Vol. 31).

Scardi, P., & Leoni, M. (2002). Whole powder pattern modelling. *Acta Crystallographica Section A Foundations of Crystallography*, *58*(2), 190–200. https://doi.org/10.1107/S0108767301021298

Thompson, P., Cox, D. E., & Hastings, J. B. (1987). Rietveld refinement of Debye–Scherrer synchrotron X-ray data from Al 2 O 3. *Journal of Applied Crystallography*, *20*(2), 79–83. https://doi.org/10.1107/S0021889887087090

Toraya, H. (1986). Whole-Powder-Pattern Fitting Without Reference to a Structural Model: Application to X-ray Powder Diffractometer Data. In *J. Appl. Cryst* (Vol. 19).

Wertheim, G. K., Butler, M. A., West, K. W., & Buchanan, D. N. E. (1974). Determination of the Gaussian and Lorentzian content of experimental line shapes. *Review of Scientific Instruments*, *45*(11), 1369–1371. https://doi.org/10.1063/1.1686503

Young, R. A., & Desai, P. (1989). Crystallite size and microstrain indicators in Rietveld refinement. *Archiwum Nauki o Materialach*, *10*, 71–90.