DIFFRAC.SUITE

User Manual

TOPAS 5 Technical Reference

Innovation with Integrity

XRD

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections are included in subsequent editions. Suggestions for improvement are welcome.

All configurations and specifications are subject to change without notice.

Order no. DOC-M88-EXX066 V5. Updated: Octr 18, 2014.

Printed in the Federal Republic of Germany.

Bruker AXS GmbH
Östliche Rheinbrückenstr. 49
76187 Karlsruhe, Germany
Tel. +49 721 50997-0
Fax +49 721 50997-5654
info@bruker-axs.de
www.bruker.com

Table of Contents

(intentionally left blank)

1 INTRODUCTION

This document describes all TOPAS functionality together with its mathematical background. Furthermore it details the Launch Mode (kernel) operation of TOPAS together with its macro language.

1.1 Definitions

TOPAS supports two modes of operation:

1. A Graphical User Interface mode for parameter input ("GUI Mode")
2. Direct editing of an input file ("Launch Mode")

Input to the TOPAS kernel is through a macro language consisting of readable "keywords" and "macros", the latter being groupings of keywords. In GUI Mode all data input / output is handled by the Parameters Window, the macro language is hidden from the user. In Launch Mode input is through an input file (*.INP).

The TOPAS data structures comprise a tree similar to an XML representation, an input file can be thought of as being an XML document but without the tags. The INP format can be described by an XML schema but it is closer to a scripting/modelling language.

The TOPAS kernel pre-processes an INP file expanding macros as required; the resulting pre-processed file (written to TOPAS.LOG) comprises keywords that are operated on by the kernel.

A calculated pattern Y_c is made up of a summation of so-called "fit objects" as follows:

- *bkg* : background
- *str...* : structure information for Rietveld refinement and structure determination
- *xo_Is...* : 2θ - I values for single line or whole powder pattern fitting
- *d_Is...* : d - I values for single line or whole powder pattern fitting
- *hkl_Is...* : Lattice information for LeBail or Pawley fitting
- *fit_obj...* : User-defined fit models

str, *xo_Is*, *d_Is* and *hkl_Is* are referred to as "phases" and the peaks of these phases as "phase peaks". A full listing of the data structures is given in section 12.

1.2 Conventions

The following conventions are used in this manual:

- Keywords are provided in *italics*
- Keywords enclosed in square brackets [ ] are optional.
- Keywords ending in ... indicate that multiple keywords of that type are allowed.
- Text beginning with the character # corresponds to a number.
- Text beginning with the character $ corresponds to a user-defined string.
- E, !E or N placed after keywords have the following meaning:
  - E: An equation (i.e. = a+b;) or constant (i.e. 1.245) or a parameter name with a value (i.e. lp 5.4013) that can be refined
  - !E: An equation or constant or a parameter name with a value that cannot be refined.
  - N: Corresponds to a parameter name.

To avoid input errors it is useful to differentiate between keywords, macros, parameter names, and reserved parameter names. The conventions followed so far are as follows:

- Keywords : all lower case
- Parameter names : first letter in lower case
- Macro names : first letter in upper case
- Reserved parameter names : first letter in upper case

1.3 Input file example

The following is an example input file for Rietveld refinement of a phase mixture of corundum and fluorite:

```
/*
Rietveld refinement comprising two phases
*/

xdd filename

   CuKa5(0.001)
   Radius(217.5)
   LP_Factor(26.4)
   Full_Axial_Model(12, 15, 12, 2.3, 2.3)
   Slit_Width(0.1)
   Divergence(1)
   ZE(@, 0)
   bkg @ 0 0 0 0 0 0

   STR(R-3C, "Corundum Al2O3")
      Trigonal(@ 4.759, @ 12.992)
      site Al x 0         y 0       z @ 0.3521  occ Al+3 1  beq @ 0.3
      site O  x @ 0.3062  y 0       z   0.25    occ O-2  1  beq @ 0.3
      scale @ 0.001
      CS_L(@, 100)
      r_bragg 0

   STR(Fm-3m, Fluorite)
      Cubic(@ 5.464)
      site Ca    x 0        y 0      z 0       occ Ca   1   beq @ 0.5
      site F     x 0.25     y 0.25   z 0.25    occ F    1   beq @ 0.5
      scale @ 0.001
      CS_L(@, 100)
      r_bragg 0
```

The format is free text, but case sensitive. Optional indentation can be used to show tree dependencies and to aid readability. Within a particular tree level placement is not important. For example, the keyword str signifies that all information (pertaining to *str*) occurring between this keyword and the next one of the same level (in this case *str*) applies to the first str.

All input text streams can have line and/or block comments. A line comment is indicated by the character " ' " and a block comment by an opening "/*" and closing "*/". Text from the line comment character to the end of the line is ignored. Text within block comments is also ignored. Block comments can be nested. Here are some examples:

```
1.    ' This is a line comment

2.    space_group C_2/c ' Text to the end of this line is ignored

3.    /*
      This is a block comment.
      A block comment can consist of any number of lines.
      */
```

On termination of refinement an output parameter file (*.OUT) similar to the input file is created with refined values updated.

1.4 Test examples

The TOPAS distribution includes many example files demonstrating the use of TOPAS and its macro language, and can act as templates for creating own INP files.

All files are found in the TOPAS installation directory (C:\TOPAS5\Tutorial\ by default) and are discussed in the Tutorial manual.

2 PARAMETERS

2.1 Refinement flags

A parameter is flagged for refinement by giving it a name. The first character of the name can be an upper or lower case letter; subsequent characters can additionally include the underscore character '_' and the numbers 0 through 9. For example:

```
site Zr x 0 y 0 z 0 occ Zr+4 1 beq b1 0.5
```

Here b1 is a name given to the *beq* parameter. No restrictions are placed on the length of parameter names.

The character ! placed before b1, as in !b1, signals that b1 is not to be refined:

```
site Zr x 0 y 0 z 0 occ Zr+4 1 beq !b1 0.5
```

A parameter can also be flagged for refinement using the character @; internally the parameter is given a unique name and treated as an independent parameter. For example:

```
site Zr x 0 y 0 z 0 occ Zr+4 1 beq @ 0.5
```

or,

```
site Zr x 0 y 0 z 0 occ Zr+4 1 beq @b1 0.5
```

The b1 text is ignored in the case of @b1.

2.2 User-defined parameters

2.2.1 The prm keyword

The *prm* keyword defines a new parameter. For example:

```
prm b1 0.2    ' b1 is the name given to this parameter
              ' 0.2 is the initial value

site Zr x 0 y 0 z 0 occ Zr+4 0.5  beq = 0.5 + b1;
                    occ Ti+4 0.5  beq = 0.3 + b1;
```

Here b1 is a new parameter that will be refined; this particular example demonstrates adding a constant to a set of *beq's*.

Note the use of the '=' sign after the *beq* keyword indicating that the parameter is not in the form of N #value but rather an equation. In the following example b1 is used but not refined:

```
prm !b1 .2
site Zr x 0 y 0 z 0 occ Zr+4 0.5  beq = 0.5 + b1;
                    occ Ti+4 0.5  beq = 0.3 + b1;
```

Parameters can be assigned the following attribute equations that can be functions of other parameters:

[*min* !E] [*max* !E] [*del* !E] [*update* !E] [*stop_when* !E] [*val_on_continue* !E]

The *min* and *max* keywords can be used to limit parameter values, for example:

```
prm b1 0.2 min 0 max = 10;
```

Here b1 is constrained to within the range 0 to 10. *min* and *max* can be equations and thus they can be functions of other parameters. Limits are very effective in refinement stabilization.

del is used for calculating numerical derivatives with respect to the calculated pattern. This occurs when analytical derivatives are not possible.

update is used to update the parameter value at the end of each iteration; this defaults to the following:

new_Val = old_Val + Change

When update is defined then the following is used:

new_Val = “update equation”

The *update* equation can be a function of the reserved parameter names Change and Val. The use of *update* does not negate *min* and *max*.

stop_when is a conditional statement used as a stopping criterion. In this case convergence is determined when *stop_when* evaluates to a non-zero value for all defined *stop_when* attributes for refined parameters and when the *chi2_convergence_criteria* condition has been met.

val_on_continue is evaluated when *continue_after_convergence* is defined. It supplies a means of changing the parameter value after the refinement has converged where:

new_Val = val_on_continue

Here are some example attribute equations as applied to the *x* parameter

```
x @ 0.1234
      min       = Val-.2;
      max       = Val+.2;
      update    = Val + Rand(0, 1) Change;
      stop_when = Abs(Change) < 0.000001;
```

2.2.2 The local keyword

The *local* keyword is used for defining named parameters as local to the top level, xdd level or phase level. For example, the code fragment

```
xdd...
      local a 1

xdd...
      local a 2
```

actually has two 'a' parameters; one that is dependent on the first *xdd* and the other dependent on the second *xdd*. Internally two independent parameters are generated,

one for each of the 'a' parameters; this is necessary as the parameters require separate positions in the A matrix for minimization, correlation matrix, errors etc.

In the code fragment

```
local a 1                    ' top level
xdd...
      gauss_fwhm = a;        ' 1st xdd
xdd...
      local a 2              ' xdd level
      gauss_fwhm = a;        ' 2nd xdd
```

the 1st xdd will be convoluted with a Gaussian with a FWHM of 1 and the 2nd with a Gaussian with a FWHM of 2. In other words the 1st *gauss_fwhm* equation uses the 'a' parameter from the top level and the second *gauss_fwhm* equation uses the 'a' parameter defined in the 2nd xdd. This is analogous, for example, to the scoping rules found in the c programming language.

The following is not valid as b1 is defined twice but in a different manner.

```
xdd...
      local a 1
      prm b1 = a;
xdd...
      local a 2
      prm b1 = a;
```

The following comprises 4 separate parameters and is valid:

```
xdd...
      local a 1
      local b1 = a;
xdd...
      local a 2
      local b1 = a;
```

The keyword *local* can greatly simplify complex INP files.

2.2.3 Modification of existing prm/local keywords

The keyword *existing_prm* allows for the modification of an existing *prm*/*local* parameter. The following:

```
local a 1
existing_prm a = 3 (a + 1);
prm = a; : 0
```

will give the result:

```
prm = a; : 6.00000
```

The operators of +=, -=, *-, /= and ^= are allowed.

2.3 Equations

Equations can be a function of parameter names providing a mechanism for introducing linear and non-linear constraints, for example:

```
site Zr x 0 y 0 z 0 occ Zr+4 zr 1      beq 0.5
                    occ Ti+4 = 1-zr;   beq 0.3
```

Here the parameter zr is used in the equation "= 1-zr;". This particular equation defines the Ti+4 site occupancy parameter. Note, equations start with an equal sign and end in a semicolon.

min/*max* keywords can be used to limit parameter values. For example:

```
site Zr x 0 y 0 z 0
     occ Zr+4 zr      1  min=0; max=1;  beq 0.5
     occ Ti+4 = 1-zr;                   beq 0.3
```

Here zr will be constrained to within 0 and 1. *min*/*max* are equations themselves und thus they can be in terms of other named parameters.

An example for constraining the lattice parameters *a*, *b*, *c* to the same value as required for a cubic lattice is as follows:

```
a lp 5.4031
b lp 5.4031
c lp 5.4031
```

Parameters with names that are the same must have the same value. An exception is thrown if the "lp" parameter values above were not all the same. Another means of constraining the three lattice parameters to the same value is by using equations with the parameter "lp" defined only once as follows:

```
a lp 5.4031
b = lp;
c = lp;
```

More general again is the use of the Get function as used in the Cubic macro as follows:

```
a @ 5.4031
b = Get(a);
c = Get(a);
```

Here the constraints are formulated without the need for a parameter name.

2.3.1 Reporting on equation values

When an equation is used in place of a parameter 'name' and 'value' as in

```
occ Ti+4 = 1-zr;
```

then it is possible to obtain the value of the equation by placing " : 0" after it as follows:

```
occ Ti+4 = 1-zr; : 0
```

After refinement the "0" is replaced by the value of the equation. The error associated with the equation is also reported when *do_errors* is defined.

2.3.2 Naming of equations

Equations can be given a parameter name, for example:

```
prm a1 = a2 + a3/2; : 0
```

The a1 parameter here represents the equation “a2 + a3/2”. If the value of the equation evaluates to a constant then a1 would be an independent parameter, otherwise a1 is treated as a dependent parameter. If the equation evaluates to a constant then a1 will be refined depending on whether the "!" character is placed before its name or not. After refinement the value and error associated with a1 is reported. This following equation is valid even though it does not have a parameter name; its value and error are also reported on termination of refinement.

```
prm = 2 a1^2 + 3; : 0
```

Equations are not evaluated sequentially, for example, the following

```
prm a2 = 2 a1; : 0
prm a1 = 3;
```

gives the following on termination of refinement:

```
prm a2 = 2 a1; : 6
prm a1 = 3;
```

Non-sequential evaluation of equations are possible as parameters cannot be defined more than once with different values or equations; the following examples lead to redefinition errors:

```
prm a1 = 2;     prm a1 = 3;  ' redefinition error
prm b1 = 2 b3;  prm b1 = b3; ' redefinition error
```

2.4 Parameter errors and correlation matrix

When *do_errors* is defined parameter errors (estimated standard deviations, esds) and the correlation matrix are generated at the end of refinement. The errors are reported following the parameter value as follows:

```
a lp 5.4031_0.0012
```

Here the error in the lp parameter is 0.0012.

The correlation matrix is identified by *C_matrix_normalized* and is appended to the OUT file if it does not already exist.

Esds are output as defined by Schwarzenbach et al. (1995), for example:

Value	Error	TOPAS Output
91.14934000	3.19837000	91(3)
86.96109000	4.51236000	87(5)
0.00000000	78.83123000	0(80)
139.03005000	0.10975000	139.03(11)
139.15734000	0.09783000	139.16(10)
138.18998000	0.03175000	138.19(3)
1388193.90000000	1063.40000000	1388200(1100)
-0.78838230	12.73400000	-1(13)
-0.00078838	0.12734000	-0.00(13)
180.62670000	1.49463800	180.6(15)
1806267.00000000	14946.38000000	1806000(15000)
18.06267000	1.49463800	18.1(15)
78.83823000	1273.40000000	0(1300)
18.06267000	14.94638000	18(15)
3.81175000	0.00103000	3.8118(10)
95.60080000	0.05050000	95.60(5)
95553.30000000	48.10000000	95550(50)
869610.90000000	45123.60000000	870000(50000)

2.5 Default parameter limits and LIMIT_MIN / LIMIT_MAX

Parameters with internal default *min*/*max* attributes are shown in Table 2.1. These limits avoid invalid numerical operations and equally important they stabilize refinement by directing the minimization routines towards lower χ^2 values. Hard limits are avoided where possible and instead parameter values are allowed to move within a range for a particular refinement iteration. Without limits refinements often fail in reaching a low χ^2. User-defined *min*/*max* limits override the defaults. *min*/*max* limits should be defined for parameters defined using the *prm* or *local* keywords.

Functionality of TOPAS is often realized through the use of the standard macros as defined in TOPAS.INC; this is an important file to view. Almost all of the *prm* keywords defined within this file have associated limits. For example, the CS_L macro defines a crystallite size parameter with a *min*/*max* of 0.3 and 10000 nanometers respectively.

On termination of refinement, independent parameters that refined close to their limits are identified by the text "_LIMIT_MIN_#" or "_LIMIT_MAX_#" appended to the parameter value. The '#' corresponds to the limiting value. These warning messages can be suppressed using the keyword *no_LIMIT_warnings*.

2.6 Reserved parameter names

2.6.1 Overview

Table 2.2 and Table 2.3 list reserved parameter names that are interpreted internally. An exception is thrown when a reserved parameter name is used for a user-defined parameter name.

An example use of reserved parameter names is as follows:

```
weighting = Abs(Yobs-Ycalc)/(Max(Yobs+Ycalc,1) Max(Yobs,1) Sin(X Deg/2));
```

Here the *weighting* keyword is written in terms of the reserved parameter names Yobs, Ycalc and X.

2.6.2 Val and Change reserved parameter names

Val is a reserved parameter name corresponding to the #value of a parameter during refinement. Change is a reserved parameter name which corresponds to the change in a refined parameter at the end of an iteration. It is determined as follows:

"Change" = "change as determined by non-linear least squares"

Val can only be used in the attribute equations *min*, *max*, *del*, *update*, *stop_when* and *val_on_continue*. Change can only be used in the attribute equations *update* and *stop_when*.

Here are some examples:

```
min 0.0001
max = 100;
max = 2 Val + .1;
del = Val .1 + .1;
update = Val + Rand(0,1) Change;
stop_when = Abs(Change) < 0.000001;
val_on_continue = Val + Rand(-Pi, Pi);
x @ 0.1234 update = Val + 0.1 ArcTan(Change 10); min=Val-.2; max=Val+.2;
```

Table 2.1: Default parameter limits.

Parameter	*min*	*max*
la	1e-5	2 Val + 0.1
lo	Max(0.01, Val-0.01)	Min(100, Val+0.01)
lh, lg	0.001	5
a, b, c	Max(1.5, 0.995 Val - 0.05)	1.005 Val + 0.05
al, be, ga	Max(1.5, Val - 0.2)	Val + 0.2
scale	1e-11	
sh_Cij_prm	-2 Abs(Val) - 0.1	2 Abs(Val) + 0.1
occ	0	2 Val + 1
beq	Max(-10, Val-10)	Min(20, Val+10)
pv_fwhm, h1, h2, spv_h1, spv_h2	1e-6	2 Val + 20 Peak_Calculation_Step
pv_lor, spv_l1, spv_l2	0	1
m1, m2	0.75	30
d	1e-6	
xo	Max(X1, Val - 40 Peak_Calculation_Step)	Min(X2, Val + 40 Peak_Calculation_Step)
I	1e-11	
z_matrix radius	Max(0.5, Val .5)	2 Val
z_matrix angles	Val - 90	Val + 90
rotate	Val - 180	Val + 180
x, ta, qa, ua	Val - 1/Get(a)	Val + 1/Get(a)
y, tb, qb,ub	Val - 1/Get(b)	Val + 1/Get(b)
z, tc, qc, uc	Val - 1/Get(c)	Val + 1/Get(c)
u11, u22, u33	Val If(Val < 0, 2, 0.5) - 0.05	Val If(Val < 0, 0.5, 2) + 0.05
u12, u13, u23	Val If(Val < 0, 2, 0.5) - 0.025	Val If(Val < 0, 0.5, 2) + 0.025
filament_length, sample_length, receiving_slit_length, primary_soller_angle, secondary_soller_angle	0.0001	2 Val + 1

Table 2.2: Reserved parameter names.

Name(s)	Description
A_star, B_star, C_star	Corresponds to the lengths of the reciprocal lattice vectors
Change	Returns the change in a parameter at the end of a refinement iteration. Change can only appear in the equations *update* and *stop_when*
D_spacing	Corresponds to the d-spacing of phase peaks in Å
H, K, L, M	hkl and multiplicity of phase peaks
Iter, Cycle, Cycle_Iter	Returns the current iteration, the current cycle and the current iteration within the current cycle respectively. Can be used in all equations.
Lam	Corresponds to the wavelength *lo* of the emission profile line with the largest *la* value
Lpa, Lpb, Lpc	Corresponds to the *a*, *b* and *c* lattice parameters respectively
Mi	An iterator used for multiplicities. See the PO macro of TOPAS.INC for an example of its use.
Mobs	Returns the number of observed reflections belonging to a particular family of reflections
Peak_Calculation_Step	Returns the calculation step for phase peaks, see *x_calculation_step*
QR_Removed, QR_Num_Times_Consecutively_Small	Can be used in the quick_refine_remove equation, see section 12.2
R, Ri	The distance between two sites R and an iterator Ri. Used in the equation part of the keywords *atomic_interaction*, *box_interaction* and *grs_interaction*.
Rp, Rs	Primary and secondary radius respectively of the diffractometer
T	Corresponds to the current *temperature*, can be used in all equations
Th	Corresponds to the Bragg angle (in radians) of hkl peaks
Val	Returns the value of the corresponding parameter. Val can only appear in the attribute equations of *min*, *max*, *del*, *update*, *stop_when* and *val_on_continue*.
Yobs, Ycalc, SigmaYobs	Yobs and Ycalc correspond to the observed and calculated data respectively. SigmaYobs corresponds to the estimated standard deviation of Yobs. Can be used only in the *weighting* equation.
X, X1, X2	Corresponds to the measured x-axis, the start and the end of the x-axis respectively. X is used in *fit_obj*'s equations and X1 and X2 can be used in any *xdd* dependent equation.
Xo	Corresponds to the current peak position

Table 2.3: Phase intensity reserved parameter names.

Name(s)	Description
A01, A11, B01, B11	Used for reporting structure factor details as defined in Eq. (9.5), see the macros Out_F2_Details and Out_A01_A11_B01_B11.
Iobs_no_scale_pks Iobs_no_scale_pks_err	Returns the observed integrated intensity of a phase peak and its associated error without any *scale_pks* applied. Iobs_no_scale_pks for a particular phase peak p is calculated using the Rietveld decomposition formulae, or, $\mathrm{Iobs\_no\_scale\_pks} = \mathrm{Get}(scale) I_p \sum_x P_{x,p} \frac{Y_{obs,x}}{Y_{obs,x}}$ where Px,p is the phase peak p calculated at the x-axis position x and Ip corresponds to the I parameter for *hkl_Is*, *xo_Is* and *d_Is* phases or (M Fobs2) for *str* phases. The summation Σx extends over the x-axis extent of the peak p. A good fit to the observed data results in an Iobs_no_scale_pks being approximately equal to I_no_scale_pks.
I_no_scale_pks	The Integrated intensity without scale_pks equations applied, or, I_no_scale_pks = Get(scale) I where I corresponds to the I parameter for *hkl_Is*, *xo_Is* and *d_Is* phases or (M Fobs2) for *str* phases.
I_after_scale_pks	The Integrated intensity with scale_pks equations applied. I_after_scale_pks = Get(*scale*) Get(*all_scale_pks*) I where I corresponds to the I parameter for *hkl_Is*, *xo_Is* and *d_Is* phases or (M Fobs2) for *str* phases. Get(*all_scale_pks*) returns the cumulative value of all *scale_pks* equations applied to a phase.

3 EQUATION OPERATORS AND FUNCTIONS

3.1 Overview

Table 3.1 lists the symbols and functions supported in equations (case sensitive). In addition equations can be functions of user-defined parameter names.

In addition the following functions are implemented:

- AB_Cyl_Corr(μR), AL_Cyl_Corr(μR):
  Returns A_B and A_L for the cylindrical sample intensity correction (Sabine et al., 1998). These functions are used in the macros Cylindrical_I_Correction and Cylindrical_2Th_Correction.
- Constant(expression)
  Evaluates "expression" only once and then replaces Constant(expression) with the corresponding numeric value. Very useful when the expected change in a parameter insignificantly affects the value of a dependent equation, see for example the TOF macros such as TOF_Exponential.
- Get_Prm_Error($prm_name)
  Returns the error of the parameter $prm_name.
- PV_Lor_from_GL(*gauss_FWHM*, *lorentzian_FWHM*):
  Returns the Lorentzian contribution of a pseudo-Voigt approximation to the Voight where *gauss_FWHM*, *lorentzian_FWHM* are the FWHMs of the Gaussian and Lorentzian convoluted to form the Voigt.
- Sites_Geometry_Distance($Name)
  Sites_Geometry_Angle($Name)
  Sites_Geometry_Dihedral_Angle($Name):
  See the *sites_geometry* keyword, section 7.
- Voigt_Integral_Breadth_GL(*gauss_FWHM*, *lorentzian_FWHM*):
  Returns the integral breadth resulting from the convolution of a Gaussian with a Lorentzian with FWHMs of *gauss_FWHM* and *lorentzian_FWHM* respectively.
- Voigt_FWHM_GL(*gauss_FWHM*, *lorentzian_FWHM*):
  Returns the Voigt FWHM resulting from the convolution of a Gaussian with a Lorentzian with FWHMs of *gauss_FWHM* and *lorentzian_FWHM* respectively.
- Yobs_dx_at(#x)
  Returns the step size in the observed data at the x-axis position #x; can be used in all sub *xdd* dependent equations. If the step size in the x-axis is equidistant then Yobs_dx_at is converted to a constant corresponding to the step size in the data.

Table 3.1: Equation operators and functions.

Classes:	Symbols / Functions:	Remarks:
Parentheses	()	
Arithmetic	+	
	-	
	*	The multiply sign is optional. (x*y = x y)
	/	
	^	x^y, Calculates x to the power of y. Precedence: x^y^z = (x^y)^z x^y*z = (x^y)*z x^y/z = (x^y)/z
Conditional	a == b	Returns 1 if a = b
	a < b	Returns 1 if a < b
	a <= b	Returns 1 if a <= b
	a > b	Returns 1 if a > b
	a >= b	Returns 1 if a >= b
	And(a, b, ...)	Returns 1 if all arguments are non-zero
	Or(a, b, ...)	Returns 1 if one arguments are non-zero
Mathematical	ArcCos(x)	Returns the arc cos of x (-1 <= x <= 1)
	ArcSin(x)	Returns the arc sine of x (-1 <= x <= 1)
	ArcTan(x)	Returns the arc tangent of x
	ArcTan2(y,x)	Returns the arc tangent of y/x
	Cos(x)	Returns the cosine of x
	Cosh(x)	Hyperbolic cosine
	Erf_Approx(x)	Error function
	Erfc_Approx	Complimentary error function
	Gamma_Approx(x)	Returns the Gamma of x
	Gamma_Ln_Approx(x)	Returns the natural logarithm of the gamma function
	Exp(x)	Returns the exponential e to the x
	Ln(x)	Returns the natural logarithm of x
	Sin(x)	Returns the sine of x
	Sinh(x)	Hyperbolic sine
	Sqrt(x)	Returns the positive square root
	Tan(x)	Returns the tangent of x
	Tanh(x)	Hyperbolic tangent
Special	For(Mi = 0, Mi < M, Mi = Mi+1 ,....)	
	Get($keyword)	
	If(conditional_test, return true_eqn, return false_eqn)	
	Sum(returns summation_eqn, initializer, conditional_test, increment_eqn)	
Miscallenous	Abs(x)	Returns the absolute value of x

Break	Can be used to terminate loops implied by the equations *atomic_interaction*, *box_interaction* and *grs_interaction*.
Break_Cycle	Can be used to terminate a refinement cycle. For example, if a particular penalty is greater than a particular value then the refinement cycle can be terminated as follows: *atomic_interaction* ai = (R-1.3)^2; … *penalty* = If(ai > 5, Break_Cycle, 0);
Error(p)	Returns associated error of parameter "p"
Ln_Normal_x_at_CD(u, s, v, toll)	Returns x value of a Ln normal distribution such that x is at the Cumulative distribution value of "cd" where "u" and "s" are the mean and standard deviation of the variable's natural logarithm. x is calculated with a tolerance in "cd" of "toll".
Max(a,b,c...)	Returns the max of all arguments
Min(a,b,c...)	Returns the min of all arguments
Mod(x, y)	Returns the modulus of x/y. Mod(x, 0) returns 0
Obj_There(a)	Returns 1 if object "a" exists within the current scope
Prm_There(a)	Returns 1 if prm/local "a" exists
Rand(x1, x2)	Returns a random value between x1 and x2
Rand_Normal(mean,sd)	Returns a random number with a normal distribution with a mean of "mean" and standard deviation of "sd"

Round(x)	Rounds x to an integer. Examples:
	`prm = Round(.1); : 0.00000`
	`prm = Round(.5); : 0.00000`
	`prm = Round(1.6); : 2.00000`
	`prm = Round(-.1); : 0.00000`
	`prm = Round(-.5); : 0.00000`
	`prm = Round(-1.6); : -2.00000`
	Note that 0.5 is rounded down to zero.

Sign(x)	Returns the sign of x, or zero if x = 0

3.2 'If' and nested 'if' statements

'Sum' and 'If' statements can be used in parameter equations, for example:

```
str...
      prm a .1
      prm b .1
      lor_fwhm = If(Mod(H, 2)==0, a Tan(Th), b Tan(Th));
```

Min and Max can also be used in parameter equations; for example the following is valid:

```
prm a .1
th2_offset = Min(Max(a, -.2), .2);
```

'If' should be preferred in non-attribute equations as analytical derivatives are possible; they can be nested, for example:

```
prm cs 200 update =
    If(Val < 10, 10,
       If(Val > 10000, 10000,
          Val
         )
      );
```

For those who are familiar with if/else statements then the IF THEN ELSE ENDIF macros as defined in TOPAS.INC can be used as follows:

```
IF a > b THEN
      return expression value
ELSE
      return expression value
ENDIF
```

3.3 User-defined functions

Functions can be defined using the *fn, def*, *return*, and *noinline* keywords; here's an example of a recursive function:

```
fn factorial(x) { return If(x == 1, 1, x factorial(x-1)); }
prm = factorial(5); : 120
```

There's also the simple form where the "return" statement is implied:

```
fn factorial(x) = If(x == 1, 1, x factorial(x-1));
```

The equation part of *prm* objects can have a function body, for example

```
prm = { def a = 2; return a; }
```

Most importantly functions can reference parameters defined with the *prm* keyword; this simplifies the writing of *prm* equations and additionally memory usage can be greatly reduced when the *noinline* keyword is used. Equations called *def* objects can be used and defined within non-simple functions. Here's an example:

```
fn gauss(a, x, f, g)
{
   def a1 = 2 Sqrt(Ln(2) / Pi) / f;
   def a2 = 4 Ln(2);
   def a3 = (x / f);
   return a1 Exp(-a2 a3^2);
}
```

A *def* object must be defined prior to its use. They can be assigned to other *def* objects but not to objects of *prm* type. In other words *prm* objects are write-protected within functions. The arguments to functions can be *def* or *prm* objects. c-style braces can be used to scope variables; the following will throw an exception due to the attempted use of an uninitialized 'def' object:

```
fn foo(x) { def a; { def a = x; } return a; }
prm = foo(3); : 0 ' Exception thrown
```

The following will not throw an exception as the simplification routines will recognize the "0":

```
fn a(x) = x undefined_name 0; prm = a(3); : 0
```

Functions can be nested; for example:

```
fn foo() {
   def a, b;
   a = 3; b = 4;
   fn nested(x, y) { return Sqrt(x^2 + y^2); }
   return nested(a, b);
}

prm = foo(); : 5
```

def and *prm* objects have scope and their scope determine the actual object used.

Here *def* "a" is returned:

```
fn a(a) { def a = 2; return a; } prm = a(1);: 2
```

Here *prm* "a" is returned:

```
prm a = 2; fn a() = a; prm = a(); : 2
```

Here the argument "a" is returned:

```
prm a = 2; fn a(a) = a; prm = a(3); : 3
```

Function specifics:

- Functions are a kernel operation and not a pre-processor operation
- Functions must be defined prior to their use
- Function arguments are optional but parentheses must be used in both the function definition and its use
- A function cannot be defined with a name of a previously defined function name
- Functions are inlined by default
- Non-nested functions can be prevented from being inlined with the *noinline* prefix
- Nested functions cannot be prefixed with *noinline*

Use of *noinline* can often be slower than not using *noinline* as a stack mechanism is used for the *fn* arguments as well as the global simplification routines cannot simplify what's inside a noinlined function. Functions are therefore "inlined" (= expanded) by default. A macro can be considered an inlined function and there's no difference in the how the following is finally processed:

```
fn my_max(a, b, c) = Max(a, b, c);

macro & my_max(& a, & b, & c) { Max(a, b, c) }
```

The macro by definition is inlined in the pre-processed INP file. In the case of *fn* the program will inline "my_max". Prefixing *fn* with *noinline* prevents inlining, for example:

```
noinline fn gauss(x, f)=(2 Sqrt(Ln(2)/Pi)/f) Exp(-4 Ln(2)((X-x)/f)^2);
```

Its best to inline small functions as it gives the simplification routines a chance to simplify what's inside the function in regards to its surroundings. Consider the following:

```
noinline fn a(b, c) = b^2 + c^2;
prm p1 1
prm !p2 1
prm p3 1
prm !p4 1
prm p5 = a(p1, p2) + a(p3, p4); : 0
```

Without inlining the simplification routines won't see that p2 and p4 are constants inside the "a" function and hence no simplification is performed; the "a" function will be called twice and the stack used twice. Note, stack here refers to the computer algebra stack. With inlining p5 after simplification reduces to:

```
prm p5 = p1^2 + p3^2 + 2; : 0
```

In the case of large functions then not inlining may increase performance as the signalling of equation nodes for recalculation will be reduced. Inlined functions have scope allowing the use of the Get() function, for example:

```
fn lat(h, k, l) = h Get(a) + k Get(b) + l Get(c);
str...
   lor_fwhm = lat(H, K, L) - lat(-H, -K, -L);
```

3.4 Floating point exceptions

An exception is thrown when an invalid floating point operation is encountered, examples are:

- Divide by zero.
- Sqrt(x) for x < 0
- Ln(x) for x <= 0
- ArcCos(x) for x < -1 or x > 1
- Exp(x) produces an overflow for x ~> 700
- (-x)^y for x > 0 and y not an integer
- Tan(x) evaluates to Infinity for x = n Pi/2, Abs(n) = 1, 3, 5,...

min/max equations or "If" statements can be used to avoid invalid floating point operations. Equations can also be manipulated to yield valid floating point operations, for example, Exp(-1000) can be used in place of 1/Exp(1000).

4 THE MINIMIZATION ROUTINES

4.1 Non-linear least squares

The Newton-Raphson non-linear least squares method is used by default with the Marquardt method (1963) included for stability. A Bound Constrained Conjugate Gradient (BCCG) method (Coelho, 2005) incorporating *min*/*max* limits is used for solving the normal equations. The objective function χ^2 is written as:

$$\chi^2 = \chi_0^2 + \chi_P^2 + \chi_R^2 \tag{4.1}$$

where

$$\chi_0^2 = K \sum_{m=1}^{M} w_m \left(Y_{o,m} - Y_{c,m}\right)^2$$

$$\chi_P^2 = K\, K_1 K_P \sum_{p=1}^{N_P} P_p \tag{4.2}$$

$$\chi_R^2 = K\, K_1 K_R \sum_{r=1}^{N_R} R_r^2$$

$$K = 1 \Big/ \sum_{m=1}^{M} w_m\, Y_{o,m}^2 \tag{4.3}$$

$Y_{o,m}$ and $Y_{c,m}$ are the observed and calculated data respectively at data point m, M the number of data points, w_m the weighting given to data point m which for counting statistics is given by $w_m=1/\sigma(Y_{o,m})^2$ where $\sigma(Y_{o,m})$ is the error in $Y_{o,m}$, P_p are penalty functions, defined using the keyword *penalty*, and N_p the number of p functions. R_r are restraints, defined using the keyword *restraint*, and N_r the number of restraints. K_P and K_R are weights applied to the penalty functions and restraints respectively. K_1 corresponds to the user defined *penalties_weighting_K1* (default value of 1), typical values range from 0.1 to 2. Penalty functions and restraints are minimized when there are no observed data Y_o.

The normal equations are generated by the usual expansion of $Y_{c,m}$ to a first order Taylor series around the parameter vector **p**. The size of **p** corresponds to the number of independent parameters N. The penalty functions are expanded to a second order Taylor series around the parameter vector **p**. The restraints are expanded to a first order Taylor series around the parameter vector **p**. The resulting normal equations are:

$$\mathbf{A}\, \Delta\mathbf{p} = \mathbf{Y} \tag{4.4}$$

where $\mathbf{A} = \mathbf{A_0} + \mathbf{A_P} + \mathbf{A_R}$ **and** $\mathbf{Y} = \mathbf{Y_0} + \mathbf{Y_P} + \mathbf{Y_R}$

$$
\begin{aligned}
A_{0,ij} &= \sum_{m=1}^{M} w_m \frac{\partial Y_{c,m}}{\partial p_i}\frac{\partial Y_{c,m}}{\partial p_j} \\
A_{P,ij} &= \frac{1}{2}\sum_{p=1}^{N_P}\frac{\partial^2 P_P}{\partial p_i \partial p_j} \\
A_{R,ij} &= K_R \sum_{r=1}^{N_R}\frac{\partial Y_{r,i}}{\partial p_i}\frac{\partial Y_{r,j}}{\partial p_j}
\end{aligned}
\qquad (4.5)
$$

$$
\begin{aligned}
Y_{0,i} &= \sum_{m=1}^{M} w_m \left(Y_{o,m} - Y_{c,m}\right)\frac{\partial Y_{c,m}}{\partial p_i} \\
Y_{P,i} &= -K_P \frac{1}{2}\sum_{p=1}^{N_P}\frac{\partial P_P}{\partial p_i} \\
Y_{R,i} &= -K_R \sum_{r=1}^{N_R} R_r \frac{\partial R_P}{\partial p_i}
\end{aligned}
$$

The Taylor coefficients $\Delta\mathbf{p}$ correspond to the changes in the parameters **p**. Equation (4.4) represents a linear set of equations in $\Delta\mathbf{p}$ that are solved for each iteration of refinement. Off diagonal terms in $\mathbf{A}_P$ are not calculated and are instead set to zero.

K_R and K_P are both set to 1 in the absence of χ_0^2. When χ_0^2 does exist then K_P is used to give approximate equal weights to the sum of the inverse error terms in the parameters $\sigma_0(p_i)^2$ and $\sigma_P(p_i)^2$ calculated from χ_0^2 and χ_P^2 respectively. Neglecting the off diagonal terms results in $\sigma_0(p_i)^2 = 1/A_{0,ii}$ and $\sigma_P(p_i)^2 = 1/A_{P,ii}$; however to avoid numerical stabilities KP is written as shown in equation (4.6).

$$
K_P = \sum_{k}^{N_k} \mathrm{If}\left(Y_{P,kk} < 10^{-14} A_{0,kk}, 0, 1.05 A_{0,kk} / \left(A_{P,kk} + A_{0,kk} Min\left(Y_{P,kk}/Y_{0,kk}, 0.05\right)\right)\right) \qquad (4.6)
$$

k corresponds to independent parameters that are a function of χ_P^2. Similarly for K_R we have:

$$
K_R = \sum_{k}^{N_k} \mathrm{If}\left(Y_{R,kk} < 10^{-14} A_{0,kk}, 0, 1.05 A_{0,kk} / \left(A_{R,kk} + A_{0,kk} Min\left(Y_{R,kk}/Y_{0,kk}, 0.05\right)\right)\right) \qquad (4.7)
$$

K_R and K_P can be modified using the keyword *pen_weight* and the macro Pen_Wt. Pen_Wt calls the macro Write_Pen_Wt which then has to be defined by the user. A definition that mimics the default is as follows:

```
macro Write_Pen_Wt(Aii, Ai, Pii, Pi)

{
pen_weight = If(Pii<1e-14 Aii, 0, 1.05 Aii/(Pii + Aii Min(Pi/Ai, 0.05)));
}
```

A_{ii} and A_i corresponds to $A_{0,ii}$ and $Y_{0,i}$ respectively. For K_P then P_{ii} and P_i correspond to $A_{P,ii}$ and $Y_{P,i}$. For K_R then P_{ii} and P_i correspond to $A_{R,ii}$ and $Y_{P,i}$.

To formulate ShelX type restraints the following could be used:

```
pen_weight = 1;
penalties_weighting_K1 = (Get(r_wp)/Get(r_exp))^2;
do_errors_include_restraints
save_best_chi2
restraint = Sqrt(w) (yt-y);
```

where Sqrt(w) is simply the square root of the restraint weight used by ShelX.

4.2 The Marquardt method

The Marquardt (1963) method applies a scaling factor to the diagonal elements of the A matrix when the solution to the normal equations of equation (4.4) fails to reduce χ^2, or,

$A_{ii,scaled} = A_{ii} (1+\eta)$

where η is the Marquardt constant. After applying the Marquardt constant the normal equations are again solved and χ^2 recalculated; this scaling process is repeated until χ^2 reduces. Repeated failure results in a very large Marquardt constant and taken to the limit the off diagonal terms can be ignored and the solution to the normal equations can be approximated as:

$$\Delta p_i = Y_i / (A_{ii} (1 + \eta)) \qquad (4.8)$$

The Marquardt method is used by default when the refinement comprises observed data Y_o. *no_normal_equations* prevents the use of the Marquardt method.

The Marquardt constant η is automatically determined each iteration. This determination is based on the actual change in χ^2 and the expected change in χ^2.

4.3 Approximating the A matrix - the BFGS method

The *approximate_A* keyword can be used to approximate the **A** matrix of Eq. (4.4) without the need for calculating the **A** matrix dot products. The approximation is based on the BFGS method (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). BCCG is used by default for solving the normal equations, alternatively, LU-decomposition can be used if *use_LU* is defined and the **A** matrix is not sparse. Note, that LU-decomposition requires the full **A** matrix and thus it may be too memory intensive for problems with 10s of thousands of parameters. LU-decomposition can also be too slow when the number of parameters is greater than about one thousand parameters.

Approximating **A** is useful when the calculation of the **A** matrix dot products is proving too expensive. When penalties dominates a refinement then the use of *approximate_A* may also improve convergence. *approximate_A* cannot be used with *line_min* or *use_extrapolation*.

When using *approximate_A* the **A** matrix can be made sparse by defining A_matrix_memory_allowed_in_Mbytes and/or A_matrix_elements_tollerance. This allows for the refinement of a very large number of parameters.

4.4 Line minimization and parameter extrapolation

Line minimization, better known as the steepest decent method, is invoked with the keyword *line_min*. It uses a direction in parameter space given by $\Delta p_i = Y_i/A_{ii}$ to minimize on χ^2 $(p+\lambda\Delta p)$ by adjusting λ.

Parameter Extrapolation, invoked with the keyword *use_extrapolation*, uses parabolic extrapolation of the parameters as a function of iteration, or, λ is adjusted such that χ^2 $(a\lambda^2+b\lambda+c)$ is minimized where for a particular parameter p_i at iteration k we have $a_i=(y_1-2y_2+y_3)/2$, $b_i=(y_3-y_1)/2$ and $c_i=y_2$ where $y_1=(p_{i,k-5}+p_{i,k-4})/2$, $y_2=(p_{i,k-3}+p_{i,k-2})/2$ and $y_3 = (p_{i,k-1}+p_{i,k-0})/2$. Parameter Extrapolation encompasses the last six sets of parameter values. In cases where both χ^2_0 and χ^2_P exists then Parameter Extrapolation reduces possible oscillatory behaviour in χ^2. Parameter extrapolation when used with Line Minimization can increase the rate of convergence when refining on penalties only.

Line minimization and Parameter Extrapolation have relatively small memory footprints and thus can be useful when the **A** matrix consumes too much memory. Alternatively the *approximate_A* keyword can be used.

Line minimization with the full A matrix calculation (no *approximate_A* defined) can increase the rate of convergence on problems like Pawley refinement.

4.5 The Conjugate Gradient Solution method

The bound constrained conjugate gradient method (Coelho, 2005) used for solving the normal equations greatly assists in convergence of the non-linear least squares process. min/max limits are dynamically recalculated during the solution process for min/max limits that are a function of independent parameters. For example, to constrain site occupancies on three sites to full occupancy with three atomic species each with occupancy of 1 the following could be defined:

```
site Ni x .11 y .22 z .33
   occ Ni ni1 0.20000 min 0 max 1
   occ Zr zr1 0.30000 min 0 max = 1 - ni1;
   occ Ca ca1 = 1 - ni1 - zr1; : 0.50000

site Zr x .21 y .32 z .43
   occ Ni ni2 0.40000 min 0 max = 1 - ni1;
   occ Zr zr2 0.50000 min 0 max = 1 - ni2;
   occ Ca ca2 = 1 - ni2 - zr2; : 0.10000

site Ca x .31 y .42 z .53
   occ Ni ni3 = 1 - ni1 - ni2; : 0.40000
   occ Zr zr3 = 1 - zr1 - zr2; : 0.20000
   occ Ca ca3 = 1 - ca1 - ca2; : 0.40000

' Occupancy on sites add up to 1
prm = ni1 + zr1 + ca1; : 1.00000
prm = ni2 + zr2 + ca2; : 1.00000
prm = ni3 + zr3 + ca3; : 1.00000

' Individual species add up to 1
prm = ni1 + ni2 + ni3; : 1.00000
prm = zr1 + zr2 + zr3; : 1.00000
prm = ca1 + ca2 + ca3; : 1.00000
```

4.6 Constraints, restraints and penalties

Additional observations can be very helpful in stabilizing least-squares refinements, wether they are included in the form of "constraints", "restraints" or "penalties". Constraints generally reduce the number of refineable parameters, while restraints and penalties increase the number of observations.

Constraints are exact mathematical conditions and are imposed rigorously. One or more least-squares variables can be expressed exactly in terms of other variables or constants, and hence eliminated. Typical constraints are symmetry related constraints placed on lattice parameters or atoms in special positions. Line profile shapes are usually defined as functions of 2θ.

Restraints and penalties are incorporated in the least-squares refinement as additional experimental observations, which are not exact but are subject to a probability distribution (the distinction between restraints and penalties is discussed in section 4.6.1). As such they are relationships which are imposed approximately as determined by a given finite weight, which has to be chosen with great care. Restraints / penalties are often used to control bond lengths and angles and are more suitable than constraints, as bond lengths and angles may be highly influenced by the chemical environment.

The advantage of using restraints / penalties instead of constraints is that if a restraint / penalty does not model the observed intensity data correctly, then the refinement may (mostly) ignore it (provided a suitable weight is applied), but if the information sought is not supported by the data, then the restraint / penalty will be imposed. Therefore the weight given to the restraint / penalty is critical, because a too large weight may enforce a preconceived model that is contrary to the observed intensity data. If used correctly, restraints / penalties should improve the agreement between the refined model and the actual structure properties, but without significantly increasing the agreement factors.

The tutorial example ALVO4-SDPD-CRP.INP[1] demonstrates how to define a rigid body for an Al-O6 octahedron, and how to link restraints and penalties to control the bondlengths:

```
prm !dist 1.9 min 1.85 max 1.95     ' Constraint
' prm dist 1.9 min 1.85 max 1.95    ' Free refinement within limits
' restraint = (dist - 1.9) / 0.05;  ' Restraint
' penalty =(dist - 1.9)^2;          ' Penalty

rigid
   point_for_site Al1
   point_for_site O1 ux =  dist;
   point_for_site O2 ux = -dist;
   point_for_site O3 uy =  dist;
   point_for_site O4 uy = -dist;
   point_for_site O5 uz =  dist;
   point_for_site O6 uz = -dist;
```

Note: In TOPAS, constraints, restraints and penalties are not specific to structure analysis, but can be applied to all refinement parameters.

1 ...\TUTORIAL\STRUCTURE DETERMINATION AND REFINEMENT\ALVO4\

4.6.1 Restraints versus penalties

From equation (4.2) a particular restraint can be reformulated into a penalty by squaring the restraint, for example

```
restraint = a (x - b);
```

is equivalent to

```
penalty = a^2 (x - b)^2;
```

In the case of the restraint the off-diagonal terms $A_{R,ij}$ are calculated when *approximate_A* (the BFGS method) is not defined. In the case of the penalty the off-diagonal terms $A_{P,ij}$ is set to zero. Restraints often converge faster than equivalent penalties due to the use of the off-diagonal terms. Penalties are useful for functions that are not to be squared; these include negative functions such as the GRS series atomic interaction.

For efficiency the A_R matrix is treated as a sparse matrix which is combined with A_0 (if it exists) where A_0 could be either sparse or dense. When *approximate_A* is used then the diagonal elements of A_0, A_P, and A_R are not calculated; instead they are approximated by the BFGS method. When *approximate_A* is used and both penalties and restraints are defined then this effectively means that the restraints are treated as penalties. The following for example:

```
approximate_A
prm p1 1 prm r1 1
penalty !P1 = 5^2 (p1 - 7)^2;
penalty !P2 = 6^2 (p1 - 8)^2;
restraint !R1 = 7 (r1 - 9);
restraint !R2 = 8 (r1 - 10);
```

will have similar but not identical convergence to the following:

```
prm p1 1 prm r1 1
penalty !P1 = 5^2 (p1 - 7)^2;
penalty !P2 = 6^2 (p1 - 8)^2;
penalty !P3 = 7^2 (r1 - 9)^2;
penalty !P4 = 8^2 (r1 - 10)^2;
```

In the former example the diagonal elements of the A matrices are:

$$\mathrm{A}_{\mathrm{P,p1p1}} = (1/2)\partial^2(P1 + P2)/\partial\, p1^2$$

$$\mathrm{A}_{\mathrm{R,r1r1}} = (\partial R1/\partial r1)^2 + (\partial R2/\partial r1)^2$$

In the latter example they are:

$$\mathrm{A}_{\mathrm{P,p1p1}} = (1/2)\partial^2(P1 + P2)/\partial\, p1^2$$

$$\mathrm{A}_{\mathrm{P,r1r1}} = (1/2)\partial^2(R1 + R2)/\partial\, r1^2$$

The difference in behavior between penalties and restraints can be seen by comparing the ROSENBROCK tutorial examples[1]. In 500,000 iterations we have:

- ROSENBROCK-10.INP: 71 iterations on average to convergence
- ROSENBROCK-10-RESTRAINT.INP: 47 iterations on average to convergence

1 ...\TUTORIAL\MISCELLANEOUS\MINIMIZATION

The restraints converge faster as the $A_{R,ij}$ elements are calculated. Approximating $A_{R,ij}$ by defining *approximate_A* in ROSENBROCK-10-RESTRAINT.INP gives the fastest convergence time wise:

- ROSENBROCK-10-RESTRAINT.INP: 71 iterations on average to convergence

Many penalties however cannot be formulated as a restraint, RASTRIGIN.INP for example, and in these cases penalties are mandatory.

4.6.2 Minimizing on penalties only

When there are no observed data or when *only_penalties* is defined then by default the BFGS method is used; see the Minimization tutorial examples[1] ROSENBROCK-10.INP and HOCK.INP. For penalties only the BFGS method typically converges faster than *line_min* / *use_extrapolation* however for penalties only it can be overridden with the use of *line_min*.

4.7 Saving refined values

Values saved on termination of refinement are determined as follows:

- If *continue_after_convergence* is NOT defined and *save_best_chi2* is NOT defined then values saved corresponds to those of the last iteration.
- If *continue_after_convergence* is NOT defined and *save_best_chi2* is defined then values saved corresponds to those that gave the best Chi^2.
- If *continue_after_convergence* is defined and *save_best_chi2* is NOT defined then values saved corresponds to those that gave the best R_{WP}.
- If *continue_after_convergence* is defined and *save_best_chi2* is defined then values saved corresponds to those that gave the best Chi^2.

When there are no penalties or restraints then the best Chi^2 corresponds to the best R_{WP}.

4.8 Error calculation

Errors are calculated for all independent and non-independent parameters that are single valued when any of the following is defined:

- *do_errors*: Errors calculated without the inclusion of penalties and restraints in the A matrix.
- *do_errors_include_penalties*: Errors calculated with the inclusion of penalties in the A matrix.
- *do_errors_include_restraints*: Errors calculated with the inclusion of restraints in the A matrix.

1 ...\TUTORIAL\MISCELLANEOUS\MINIMIZATION

4.9 Simulated annealing adaptive step size

The adaptive step size used in simulated annealing has been improved. In many case the complex temperature regime found in the macro Auto_T can be replaced with a single temperature. When using a very incorrect starting temperature the program quickly modifies the temperature to a more appropriate vale. Output lines such as:

```
Breaking - randomize on errors revisit
```

indicate that a particular parameter configuration has been revisited and the temperature will be internally adjusted. Note, with *randomize_on_errors*, relative temperature values are pertinent and not absolute values.

4.10 Refining on an arbitrary Chi^2

The *chi2* keyword allows for minimization of a user defined χ^2. It can be a function of the reserved parameter names X, Yobs, Ycalc and SigmaYobs. In addition the keyword *xdd_sum* is a parameter that can be a function of these reserved parameter names. To, for example, define a normal least squares refinement the following can be used:

```
xdd...
   xdd_sum denominator = Yobs;
   xdd_sum numerator = (Yobs - Ycalc)^2 / Max(Yobs,1);
   chi2 = 100 Sqrt(numerator / denominator);
```

In refining on an arbitrary *chi2* the first and second derivatives of *chi2* with respect to each independent parameter is required. To do this fast Ycalc within *chi2* is approximated with a first order Taylor approximation around the parameter vector p. This approximation for various formulations of *chi2* has yielded good convergence even for non-linear parameters. To summarize:

- *chi2* is treated as a penalty
- For each independent parameter, a definite minima in *chi2* is bracketed and inverse parabolic interpolation used to determine the minima of *chi2* with respect to that parameter. In the calculation of *chi2*, Ycalc is replaced with its first order Taylor approximation and thus the full Ycalc is only calculated once per refinement iteration and not 100s of times.
- Finding the minima and the curvature of *chi2* with respect to each parameter yields 1st and 2nd order derivatives of *chi2* with respect to each parameter.
- The BFGS method (*approximate_A*) is then used to solve the resulting linear equations with off diagonal terms approximated according to the BFGS method.
- The BCCG method incorporating the Marquardt method with automatic Marquardt constant determination is used to solve the matrix equations.

The tutorial example CHI2-CEO2.INP[1] demonstrates various scenarios.

1 ...\TUTORIAL\MISCELLANEOUS\MINIMIZATION

4.11 Informing of unrefined parameters

Parameters that do not take part in a refinement are now reported, for example, the following:

```
prm a 1
prm b 1
```

where a and b are not used in any equations that are part of refinement will result in the following output:

```
Number of independent parameters not taking part in refinement: 2
prm_10: a
prm_10: b
```

The *val_on_continue* attribute of unrefined parameters are executed at the end of convergence. It can be useful, for example,

```
prm a 1 val_on_continue = b = 2; ' this sets the parameter b to 2.
```

4.12 Summary, iteration and refinement cycle

Table 4.1 shows various keywords usages for typical refinement problems. The term “refinement cycle” is used to describe a single convergence. The reserved parameter Cycle returns the current refinement cycle with counting starting at zero. The reserved parameter Cycle_Iter returns the current iterations within a cycle with counting starting at zero.

Table 4.1: Keyword sequences for various refinement types.

Refinement type:	**Keywords to use:**	**Comments:**
Rietveld refinement No penalties		Marquardt refinement. **A** matrix calculation.
Rietveld refinement with a moderate number of penalties.	*line_min* (maybe)	Line minimization Marquardt refinement. **A** matrix calculation.
Rietveld refinement dominated by penalties	*approximate_A*	BFGS method of refinement. **A** matrix approximation.
Pawley refinement	*line_min*	Line minimization Marquardt refinement. **A** matrix calculation.
Penalties only		BFGS method of refinement. **A** matrix approximation.
Refinements with a large number of parameters	*approximate_A*	BFGS method of refinement. **A** matrix approximation.

4.13 quick_refine and computational issues

The computationally dominant factor of calculating Eq. (4.5) is problem dependent. For Rietveld refinement with a moderate number of parameters then the calculation of the peak parameter derivatives may well be the most expensive. On the other hand for Rietveld refinement with a large number of structural parameters and data points then the calculation of the $A_{1,ij}$ dot products would be the dominant factor, where, the number of operations scale by $M(N^2+N)/2$. Before the development of the BCCG routine (Coelho, 2005), the solution to the normal equations, Eq. (4.4), was also very expensive.

For structure solution from powder data by Global Optimization, the keyword *yobs_to_xo_posn_yobs* can be used to reduce the number of data points M; thus reducing the number of operations in the $A_{1,ij}$ dot products, see the macro *Decompose*.

The *quick_refine* keyword removes parameters during a refinement cycle thus shrinking the size of the **A** matrix by reducing N. Parameters are removed if the condition defined in Eq. (4.9) is met for three consecutive iterations.

$$\Delta p_i < \left(0.01\,\text{quick\_refine} \,/\, \left(K\,N\,Y_i\right)\right) \tag{4.9}$$

Alternatively, parameters can be removed or reinstated during a refinement cycle using *quick_refine_remove*. This keyword provides a means of performing block refining. If q*uick_refine_remove* is not defined then all parameters are reinstated at the start of refinement cycles.

4.14 Auto_T and randomize_on_errors

It is sometimes difficult to formulate optimum *val_on_continue* functions for simulated annealing. This is especially true in structure solution using rigid bodies where optimum randomization of the rigid body parameters is difficult to ascertain. *randomize_on_errors* is a means of automatically randomizing parameters based on the approximate errors in the parameters as given in Eq. (4.10), where T is the current temperature and K is as defined in Eq. (4.3).

$$\Delta p_i = Q\,\text{Sign}(\text{Rand}(-1,1))\sqrt{0.02\,T/\left(K\,A_{ii}\right)} \tag{4.10}$$

Q is a scaling factor determined such that convergence to a previous parameter configuration occurs 7.5% of the time on average. When *randomize_on_errors* is used the magnitude of the *temperature*(s) is not of significance but the relative variation in *temperature*(s) are.

The macro Auto_T includes *quick_refine*, *randomize_on_errors* and a *temperature* regime. It has shown to be adequate for a wide range of simulated annealing examples, see the tutorial examples for structure determination and refinement[1].

Note: When *val_on_continue* is defined then the corresponding parameter is not randomized according to *randomize_on_errors*.

1 ...\TUTORIAL\STRUCTURE DETERMINATION AND REFINEMENT\

5 PROFILE FITTING

The present chapter provides an overview about the origin and modelling of line profile shapes, with the focus on TOPAS' convolution based approach to X-ray and neutron powder data. The principles discussed are equally valid for all profile fitting methods supported by TOPAS including (i) single line fitting, (ii) whole powder pattern fitting, (iii) whole powder pattern decomposition according to Pawley (Pawley, 1981) and Le Bail (Le Bail et al., 1988), (iv) Rietveld refinement (Rietveld, 1967; Rietveld, 1969) and (v) ab-initio structure determination from step intensity data (Coelho, 2000).

Suggested reading:

- **Cheary, R.W. & Coelho, A.A. (1992):** *A Fundamental Parameters Approach of X-ray Line-Profile Fitting.* - J. Appl. Cryst. 25, 109.
- **Cheary, R.W., Coelho, A.A. & Cline, J.P. (2004):** *Fundamental Parameters Line Profile Fitting in Laboratory Diffractometers.* - J. Res. Natl. Inst. Stand. Technol., 109, 1-25.
- **Kern, A. (2008):** *Profile Analysis.* - Principles and Applications of Powder Diffraction. Editors: Clearfield, A., Bhuvanesh N. & Reibenspies J. Blackwell Publishers, 400 pages. ISBN: 978-1-405-16222-7
- **Kern, A., Cheary, R.W., Coelho, A.A. (2004):** *Convolution Based Profile Fitting.* - Diffraction Analysis of the Microstructure of Materials. Editors: Mittemeijer, E.J. & Scardi, P. Springer Series in Materials Science, Vol. 68, 552 pages, ISBN: 978-3-540-40519-1

5.1 Origin of line profile shapes

Observed line profile shapes are the result of the convolution of a series of instrument contributions, including the wavelength distribution (source emission profile) and geometrical effects, and specimen contributions to the diffraction process (Jones, 1938). Full exploitation of the diffraction information requires an accurate description of all instrument and specimen contributions to the observed line profile shapes. This is particularly true for microstructure analysis, requiring accurate knowledge of the instrument contribution.

5.1.1 Instrument contributions

In laboratory diffractometers, where typically only broad-band filters (such metal Kß filters or single-bounce monochromators) are employed, the source emission profile may contribute significantly to the observed line profile shape. Such "partly filtered" spectral shapes do not have simple descriptions and need particular consideration independent on geometrical instrument contributions (sections 5.1.1.1 and 5.1.1.2, respectively).

The situation is different for synchrotron and neutron diffractometers utilizing a white light source, where the wavelength distribution in the beam is determined by the instrument characteristics (predominantly monochromator assembly), rather than the source. In contrast to laboratory diffractometers, a separate treatment of the source emission profile and instrument contributions is normally not of interest. For most synchrotron and neutron diffractometers instrument functions are usually determined by measurements of reference materials, exhibiting only low levels of line broadening. Thorough characterizations of instrument functions from first principles have been published by e.g. David and Jorgensen (1993), Masson et al. (2003), and Gozzo et al. (2006), and are not discussed here.

5.1.1.1 Source emission profiles in laboratory diffractometers

In the absence of instrumental effects the source emission profile represents the highest possible resolution of a laboratory diffractometer. Above 90° 2θ the source emission profile is the dominant contribution to the line profile and conforms closely to the source emission profile, as the total breadth of the instrument contributions gets relatively small. The exact knowledge of the shape of the source emission profile is of central importance to the development of an accurate powder profile description of X-ray laboratory data. With the high quality data obtainable from modern diffractometers in terms of intensity, FWHM, and peak-to-background ratio, the characteristics of source emission profiles become remarkably evident.

The form of the Cu Kα source emission profile is shown in Fig. 5.1. As the natural shape of a source emission profile is Lorentzian, the tails can extend a considerable distance from the central peak. For Cu as well as all other transition element anodes used in laboratory X-ray diffraction, both the Kα_1 and the Kα_2 lines are asymmetric with extended high angle tails. In addition, the asymmetries and FWHM of the Kα_1 and the Kα_2 lines are different. The natural asymmetry of the source emission lines arises from the multiplet structure of the transitions, the Kα_1 and the Kα_2 lines are each doublets (Deutsch et al., 1995). Another feature of the Kα emission profile is the satellite multiplet structure in the high energy tail. For Cu these collectively have an intensity of $\approx$ 0.6% of the Kα_1 emission line, rising uniformly with decreasing atomic number up to 1.4% for Cr (Parrat, 1936).

Laboratory diffractometers normally operate with some form of monochromatization such as metal filters, crystal monochromators or mirrors; as a result the source emission profiles are modified to varying degrees (see Fig. 5.2 to Fig. 5.5).

For metal-filtered characteristic radiation there is only a small variation in the attenuation across the source emission profile as seen in Fig. 5.2. This variation is owed to the increase in the linear absorption coefficient with increasing wavelength, but does not significantly affect the line profile shape. However, absorption edges can lead to a significant step in the low-angle tail intensity, which may form a major portion of the least-squares residual due to profile mismatch.

The inclusion of a focusing wide-band monochromator (pyrolytic graphite) greatly reduces the range of wavelengths resulting in profile tails that diminish more rapidly than the natural emission profile (Fig. 5.2). Such a monochromator can also affect the relative intensity $I_{K\alpha 2}/I_{K\alpha 1}$ depending on the alignment and setting of the crystal.

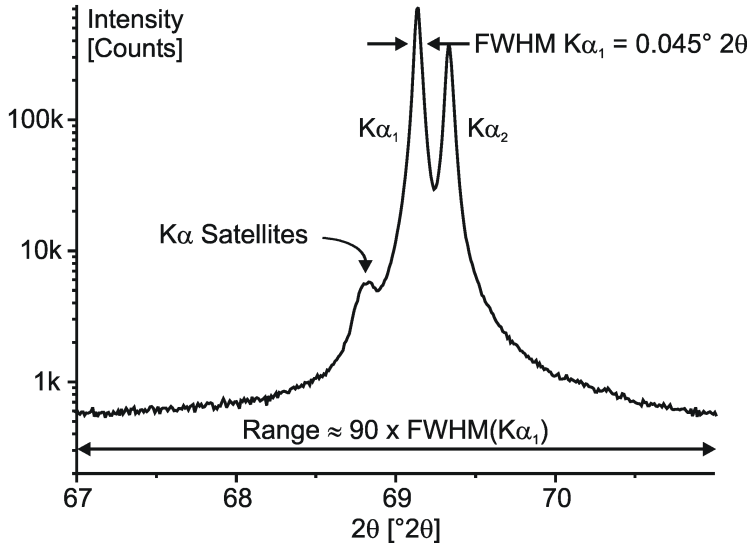


Fig. 5.1: Cu Kα emission profile showing the satellite group of lines and the extent of the tails from the $K\alpha_1$ and $K\alpha_2$ emission lines, covering a range of approximately 90 times the FWHM of $K\alpha_1$. This profile was recorded using the 400 line from a silicon single crystal wafer. From Cheary et al. (2004).

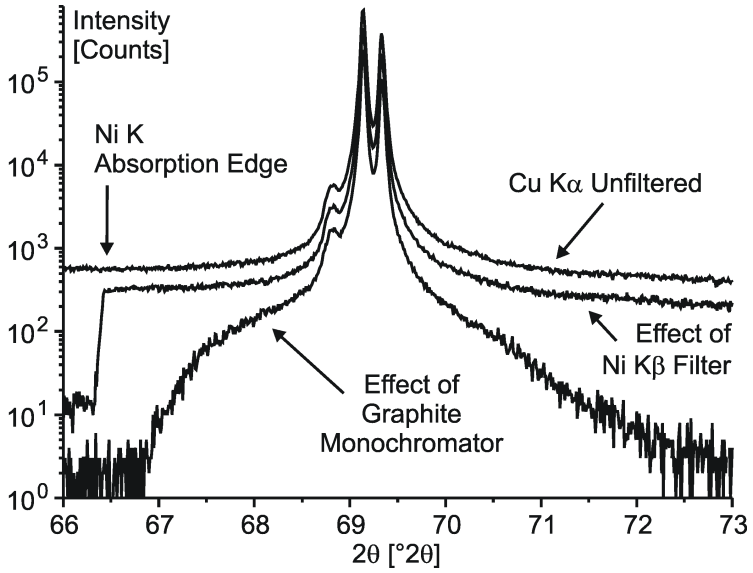


Fig. 5.2: Cu Kα emission profile obtained using the 400 line from a silicon single crystal wafer. Each pattern was recorded sequentially using the same sample, first with no filter or monochromator in the beamline, then with a Ni Kß filter, and finally with a standard curved graphite diffracted beam monochromator. From Cheary et al. (2004).

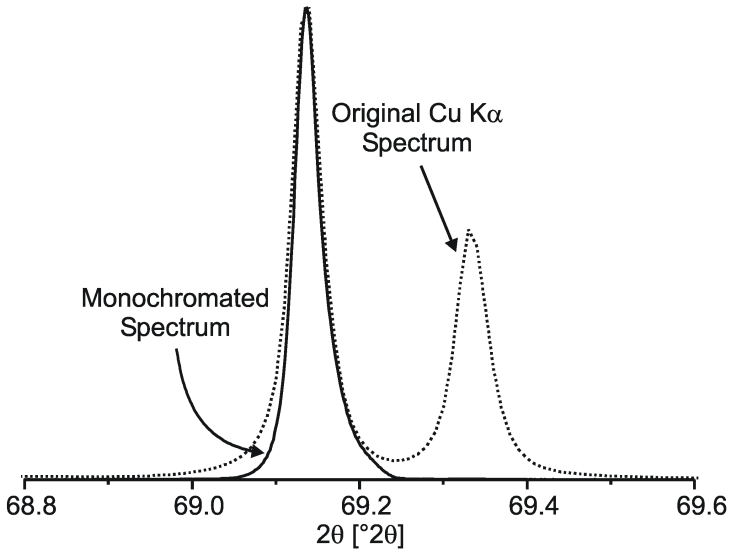


Fig. 5.3: Wavelength spectrum emerging from an asymmetrically cut Ge111 ground and bent incident beam monochromator (Johannson-type). From Cheary et al. (2004).

Johannson-type focusing monochromators are able to achieve near perfect focusing and to select a very narrow band of wavelengths, when correctly aligned. Fig. 5.3 shows the Cu Kα source emission profile from an asymmetrically cut, ground, and bent Ge crystal used as an incident beam monochromator. The wavelength band-pass of this monochromator type is narrow enough to remove 99.98% of the Kα_2 component and 100% of the Kα satellite group, and to almost completely eradicate the Lorentzian tails of the source emission profile.

Parabolic multilayer mirrors in the incident beam of a diffractometer can also distort the wavelength spectrum (Toraya and Hibino, 2000). As different wavelengths reflect off the mirror in slightly different directions as shown in Fig. 5.4 for the Kα_1 and Kα_2 components, an inhomogeneous spread in wavelength across the specimen in the equatorial plane will be obtained. For a mirror setup in the orientation shown in Fig. 5.4, the separation between Kα_1 and Kα_2 is smaller than the value expected from the natural Kα_1 and Kα_2 wavelengths by an amount corresponding to the difference $\Delta\psi$ in the directions of the two incident beams on to the specimen. As a result, a significant compression of the natural source emission profile may occur at high 2θ angles as shown in Fig. 5.5. The magnitude of $\Delta\psi$ depends on the bandwidth of the mirror, and increases with decreasing bandwidth. Therefore, mirrors with small bandwidths (usually intended for thin film analysis), will normally show stronger aberrations.

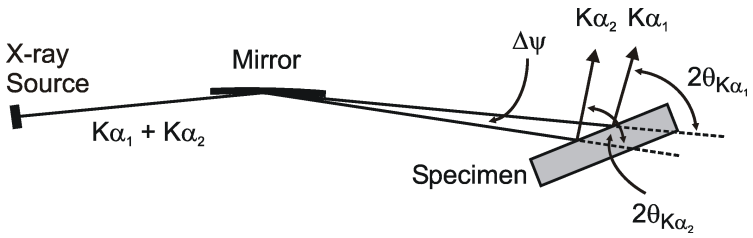


Fig. 5.4: Reflection of Kα_1 and Kα_2 wavelengths from a parabolic multilayer mirror diffracting off a powder specimen. From Cheary et al. (2004).

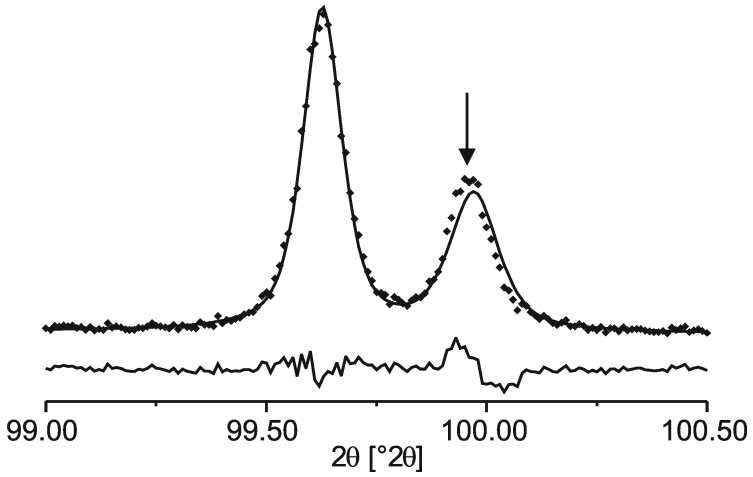


Fig. 5.5: Single line fit to the (041) / (322) line of LaB_6 (NIST SRM 660a) at about 99.6° 2θ using the natural Kα_1 and Kα_2 wavelengths. Parallel beam data have been taken with a D8 ADVANCE (Bruker AXS) equipped with a single Göbel mirror in the primary beam and Parrish-Hart analyser slits in the diffracted beam. The Kα_2 peak maximum (arrow) is significantly shifted towards the Kα_1 reflection.

From Fig. 5.1 to Fig. 5.5 it will be clear that the shape of the source emission profiles in laboratory diffractometers is much more complex than the simple Kα doublet / Kß singlet model used in traditional profile fitting software. As soon as the source emission profile starts to dominate the line profile shapes, such inappropriate modeling contributes significantly of the total misfit in profile fitting as is e.g. seen in Fig. 5.5.

5.1.1.2 Geometrical contributions in laboratory diffractometers

Geometrical instrument aberrations are inherent to the given instrument geometry and its configuration (instrument dimensions, choice of optical components). The most widely used laboratory diffractometers are operated either in Bragg-Brentano (Fig. 5.6) or Debye-Scherrer geometry (Fig. 5.7). All these configurations have an array of geometrical aberrations in common; for details see Cheary & Coelho (1992), Cheary et al. (2004), Kern et al. (2004), and Kern (2008).

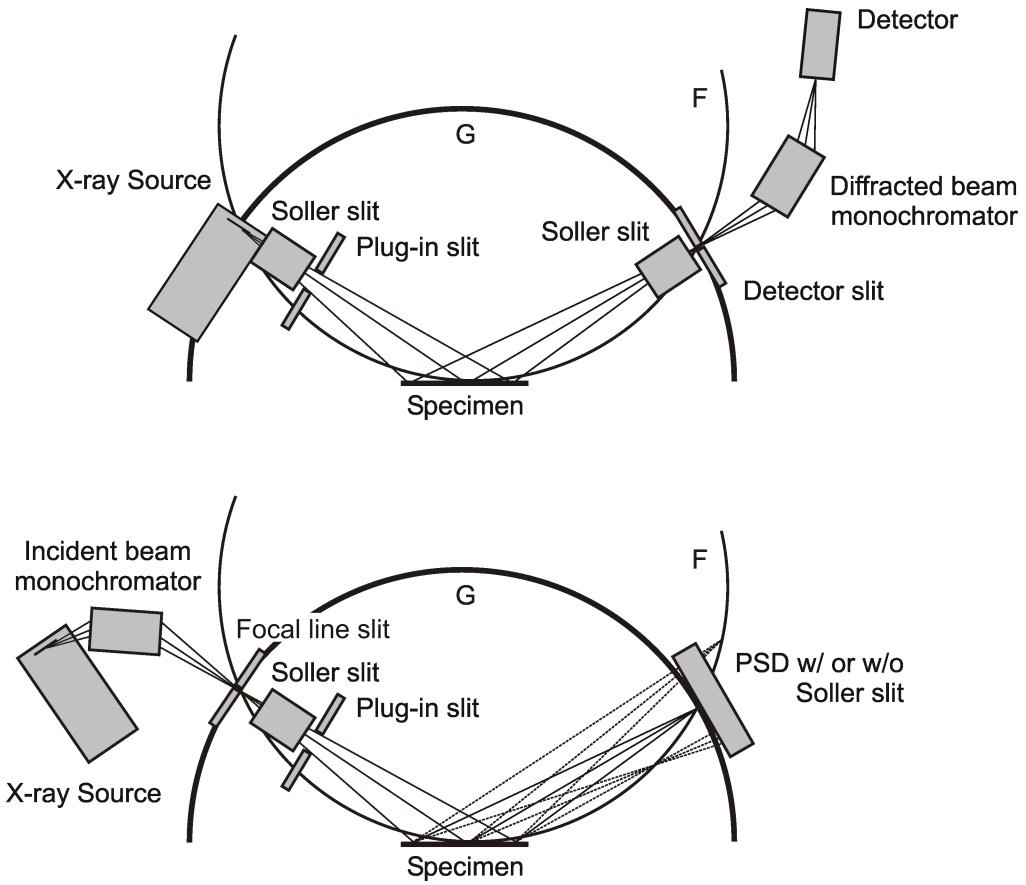


Fig. 5.6: Two common configurations of diffractometers operating in the Bragg-Brentano geometry, showing the principal optical components. G: goniometer circle; F: focusing circle.

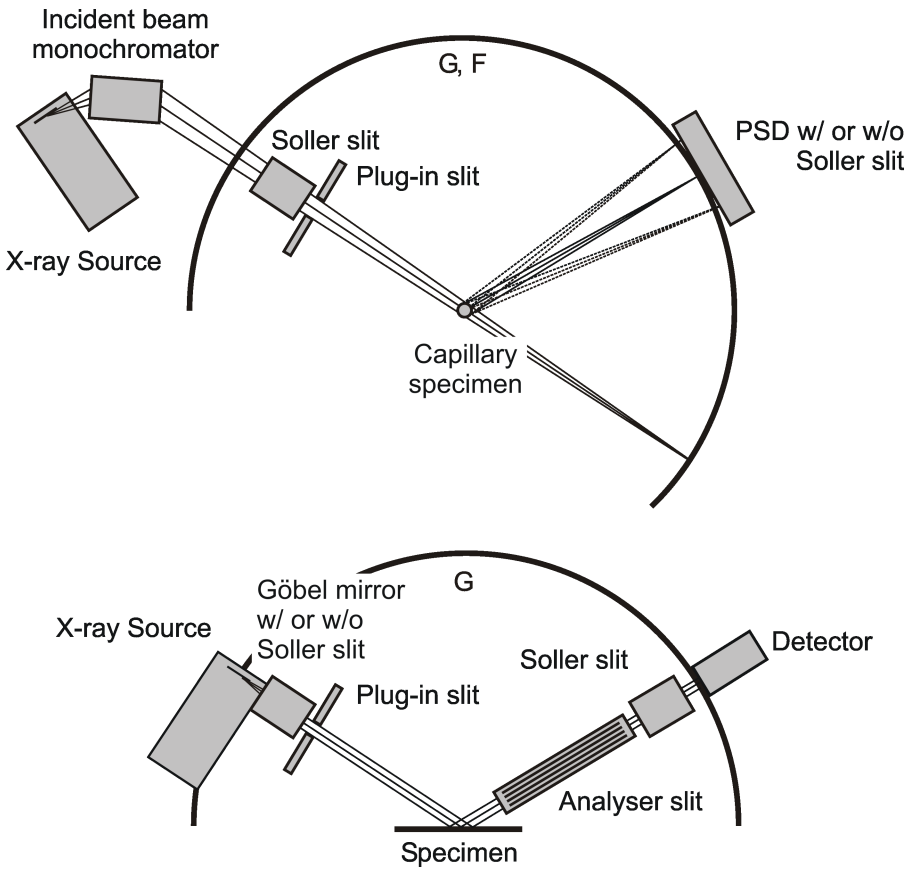


Fig. 5.7: Two common configurations of diffractometers operating in the Debye-Scherrer geometry showing the principal optical components. G: goniometer circle. For the focusing Debye-Scherrer geometry the focusing circle F coincides with the goniometer circle.

For a diffractometer operating in Bragg-Brentano geometry and equipped with a point detector, the major principal geometrical instrument aberrations contributing to a line profile, alongside the wavelength distribution, are:

- Finite width of the X-ray source
- Angular acceptance function of the Soller slit(s) controlling the axial beam divergence
- Angular acceptance function of the plug-in slit controlling the equatorial beam divergence (⇨ "flat specimen aberration")
- Angular acceptance function of the receiving slit

For linear position sensitive detector (PSD) systems, the receiving slit aberration is irrelevant and the flat specimen aberration is replaced by an aberration function that embraces three effects that are convoluted together,

- Defocusing due to asymmetric diffraction
- Discharge resolution of the detector
- Parallax error

For diffractometers operating in Debye-Scherrer geometry and equipped with a point detector, there are two major principal geometrical instrument aberrations:

- Angular acceptance function of the analyser slit
- Angular acceptance function of the Soller slit(s) controlling the axial beam divergence

Additional aberrations for linear PSD system are:

- Discharge resolution of the detector
- Parallax error for linear detectors

Geometrical instrument aberrations may broaden diffraction line profile shapes more or less asymmetrically and shift them from their theoretical 2θ position. Typical aberration functions are shown in Fig. 5.8 for the instrument contributions listed above. It must be borne in mind that these functions are approximations and that the degree of diffraction line profile broadening and shift is a function of 2θ for most aberations.

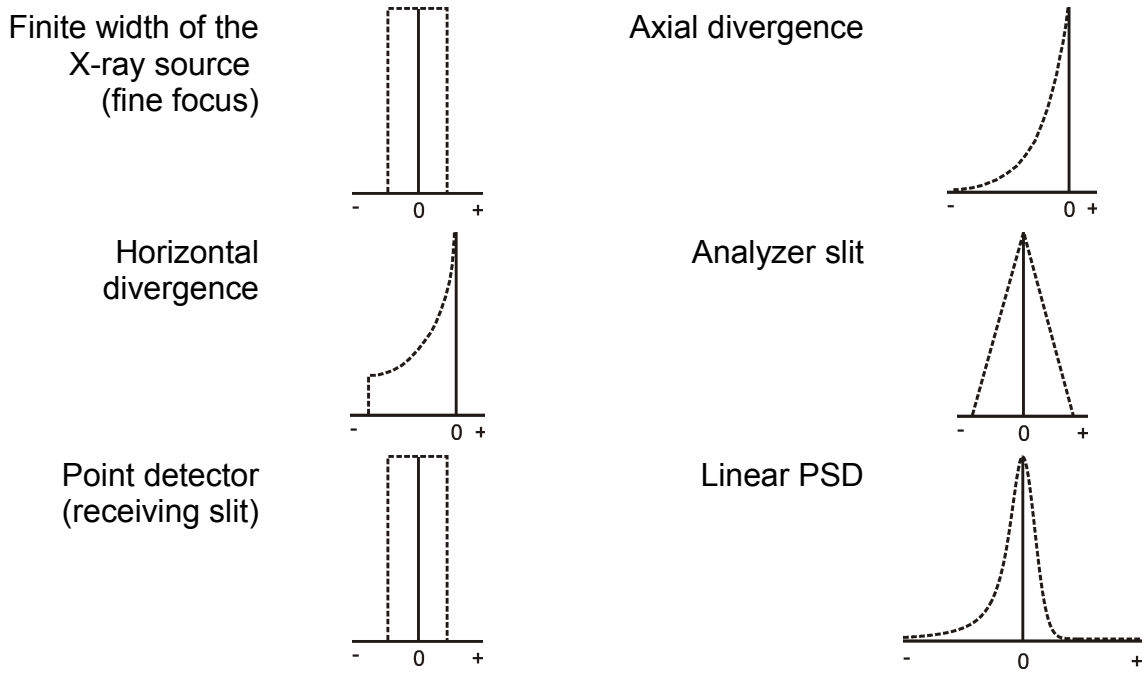


Fig. 5.8: Simplified aberation functions representing the major principal geometrical instrument contributions.

5.1.2 Specimen contributions

Specimen contributions affect the width and shape of diffraction-line profiles and may also shift them from their ideal position. Specimen contributions have to be categorized into 1) contributions form the specimen as an entity and 2) contributions from phase. The major principal specimen contributions are:

1. Specimen specific: Absorption
   - "Specimen transparency error" in Bragg-Brentano geometry related to beam penetration into the sample (Bragg-Brentano geometry)
   - Non-uniform absorption by a specimen mounted into a capillary (Debye-Scherrer geometry)
2. Phase specific: Microstructure broadening

Typical functions describing specimen contributions are shown in Fig. 5.9.

Specimen contributions related to absorption effects depend on both specimen properties (linear absorption coefficient) as well as specimens preparation (specimen thickness and packing density). Different specimens may contribute differently to the diffraction process and thus need particular consideration for accurate microstructure analysis.

The term microstructure encompasses a series of lattice imperfections leading to deviations from the ideal crystal structure and thus causing microstructure (= physical) broadening. In Fig. 5.9 a simplified representation of microstructure contributions is given in terms of size- and microstrain broadening. For a comprehensive discussion about microstructure analysis using TOPAS refer to section 6.

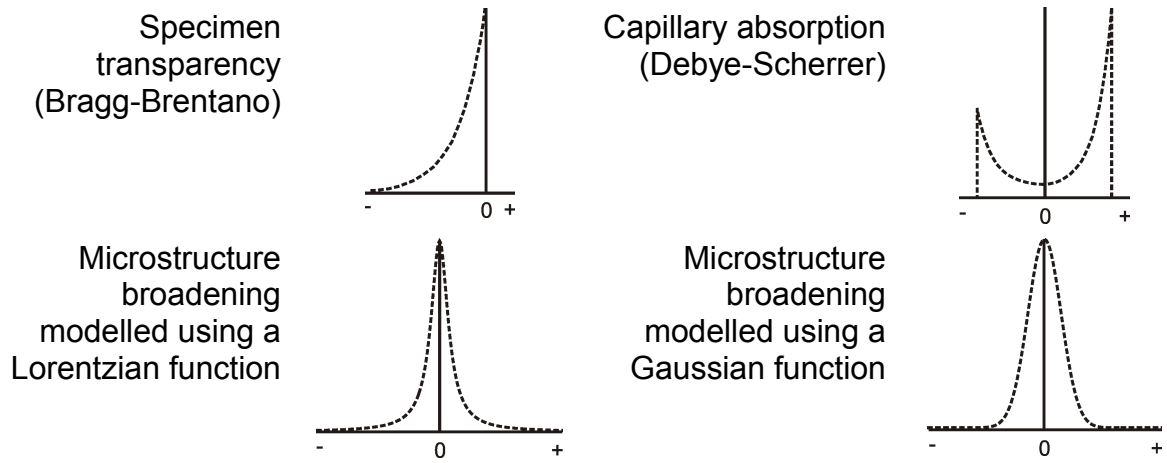


Fig. 5.9: Simplified functions representing the major principal specimen contributions.

5.2 Convolution based profile fitting

In convolution based profile fitting, profiles are modelled by convoluting appropriate functions to form the observed profile shapes. In direct convolution approaches, profile shape functions are fitted directly to observed diffraction-line profiles, in contrast to mere synthesis or deconvolution methods. Generally, any combination of appropriate functions may be used in this context for modelling profiles empirically. With functions representing both the aberration functions of the diffractometer as well as the various specimen contributions a fundamental parameters based synthesis of line profile shapes is achieved (fundamental parameters approach, FPA).

5.2.1 Convolution basics

The process of convolution is one in which the product of two functions $f(2\theta)$ and $h(2\theta)$ is integrated over all space,

$$y(2\theta) = f(2\theta) \otimes h(2\theta) = \int f(2\theta') \, h(2\theta - 2\theta') \, d(2\theta'), \qquad (5.1)$$

where

- $y(2\theta)$ is the convolution product,
- $2\theta'$ is the variable of integration in the same 2θ domain, and
- $\otimes$ denotes the convolution process.

In simple terms, convolution can be understood as "blending" one function with another, producing a kind of very general "moving average". The convoluted function is obtained by setting down the origin of the first function in every possible position of the second, multiplying the values of both functions in each position, and taking the sum of all operations.

TOPAS performs the operation of convolution in various ways by means of direct convolution and Fast Fourier Transform (FFT) convolution, for details see section 5.4.

5.2.2 Application areas

Generally, the application areas of convolution based profile fitting are,

1. Empirical profile fitting: The arbitrary parameterization of measured line profile shapes by convolution of any appropriate functions. Note that refined profile parameters have no physical meaning.
2. The explicit discrimination of instrument and specimen contributions: Instrument function approach. There are two cases, dependent on how the instrument function is determined, using,
   - measured instrument functions, or
   - calculated instrument functions

5.2.2.1 Empirical profile fitting

Convolution based profile fitting can be used for a fully empirical parameterization of diffraction-line profile shapes $Y(2\theta)$. For a convolution of n functions $F_i(2\theta)$, this process can be written as

$$Y(2\theta) = W \otimes F_1(2\theta) \otimes F_2(2\theta) \otimes ... \otimes F_i(2\theta) \otimes ... \otimes F_n(2\theta) \quad (5.2)$$

where

- $Y(2\theta)$ is the observed line profile shape, and
- W is the source emission profile

A schematic of which is shown in Fig. 5.10. The significance of this approach lies in its ability to construct PSFs with any shape dependence on angle and hkl direction, based on an appropriate choice of functions representing the observed line profile shape. In the TOPAS implementation, which also allows the use of user-supplied functions, a mixture of analytical and numerical convolutions is used, and the PSFs are fitted to the observed line profiles with all parameters refineable.

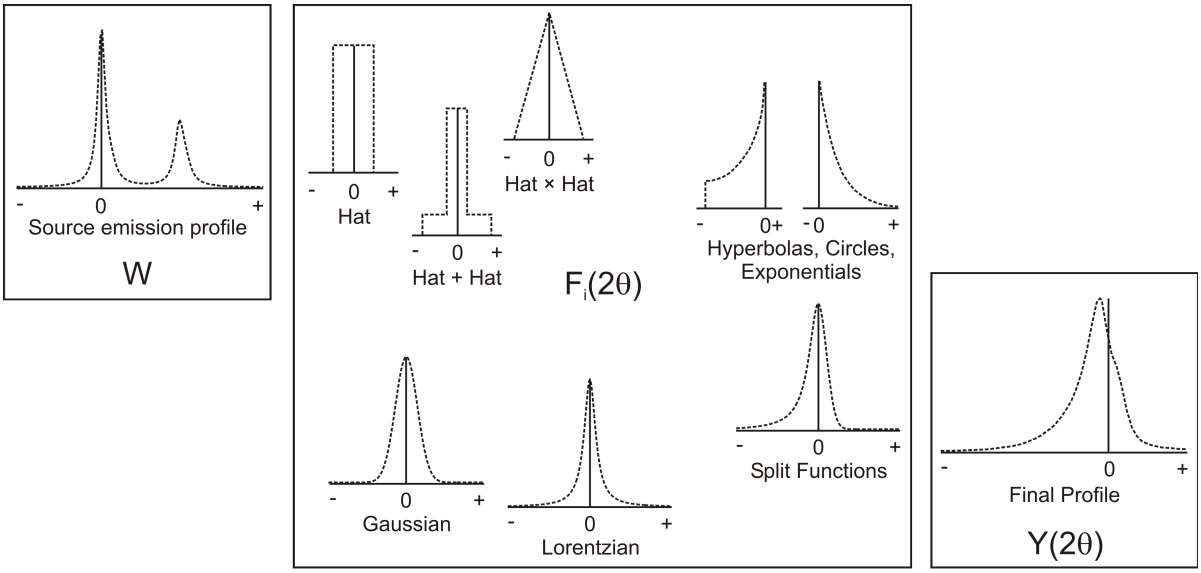
The notable flexibility of this approach results in a quality of fit, which is normally better than those obtained by other methods, or at least equal to them.

In practice,

- any appropriate functions $F_i(2\theta)$ may be convoluted to achieve a best fit, however,
- as always in fitting, the number of functions / function parameters needs to be minimized to minimize parameter correlation.

A wise choice of functions $F_i(2\theta)$ will usually require less refineable parameters than other profile fitting methods, but still provide a better quality of fit.

Note, that the refined profile parameters have no physical meaning, just as in any other empirical profile fitting techniques. Empirical convolution based profile fitting is therefore an excellent approach for all profile fit applications using any type of instrument and sample if micro-structure information is not of interest, and if peak overlap is moderate.



$$Y(2\theta) = W \otimes F_1(2\theta) \otimes F_2(2\theta) \otimes \ldots \otimes F_i(2\theta) \otimes \ldots \otimes F_n(2\theta)$$

Fig. 5.10: Schematic representation of the convolution approach as given in equation (5.2). The final profile $Y(2\theta)$ is described by the selection of appropriate functions $F_i(2\theta)$ and convoluting them on top of the emission profile W. Note the capabilities of the TOPAS implementation to add functions before convoluting them as shown for the sum of two hat functions as well as to convolute split-type functions.

5.2.2.2 Discrimination of instrument and specimen contributions

Convolution based profile fitting implicitly allows the discrimination of instrument and specimen contributions (Alexander, 1954; Wilson, 1963),

$$Y(2\theta) = W \otimes G(2\theta) \otimes S(2\theta), \qquad (5.3)$$

where G and S are geometric instrument and specimen contributions respectively, which can be modelled separately using appropriate $F_i(2\theta)$ functions according to equation (5.2). In general, for a particular instrumental setup, the entity $W \otimes G(2\theta)$ represents the instrument function $I(2\theta)$, which can be either measured or calculated.

The discrimination of instrument and specimen contributions is not just a prerequisite for microstructure analysis (see sections 5.3 and 6). Generally, an instrument function constraint is of great advantage for all powder diffraction applications adversely affected by peak overlap; this is particularly true for structure analysis and quantitative phase analysis. An instrument function constraint significantly

- improves the discrimination between profile and background intensity in complex peak overlaps, and
- greatly reduces the number of refineable profile parameters required to describe diffraction-line profile shapes compared to empirical profile fitting approaches.

Unless specimen broadening becomes the dominant contribution, uncertainties due to parameter correlation are reduced, which allows a more successful decomposition of peak overlaps at higher degrees of overlap.

5.2.2.2.1 Measured instrument function approach

Measured instrument functions are usually obtained from standard reference materials, which ideally do not contribute any specimen broadening to the diffraction process. In this case, the observed line profile shapes will directly represent the instrument function, i.e.

$$Y(2\theta) = I(2\theta), \tag{5.4}$$

which will be obtained by empirical profile fitting using equation (5.2).

In practice, the application of measured instrument functions for profile fitting represents a two stage approach:

1. Determination of the instrument function
   - Data acquisition using a suited standard reference material, e.g.:
     - Reflection geometry: LaB_6 (NIST SRM 660b)
     - Transmission geometry: Si (NIST SRM 640d)
   - Convolution of any number of appropriate functions $F_i(2\theta)$ to achieve a best fit via empirical profile fitting, see section 5.2.2.1.
   - Fixing of all refineable profile parameters; the obtained set of profile functions and related function parameters represents the instrument function sought

   Note, that measured instrument functions need redetermination after instrument alignment / change of configuration.
2. Modelling of the observed line profiles of the actual specimen
   - Convolution of appropriate functions $F_i(2\theta)$ on top of the instrument function to describe any specimen contributions.

The number of functions $F_i(2\theta)$ to obtain $I(2\theta)$ is irrelevant when determining an instrument function, provided that the background contribution is correctly described. For the analysis of actual sample diffraction patterns, all instrument function parameters are kept fixed and are therefore not included in the least-squares refinement process (instrument function constraint).

The measured instrument function approach is an excellent approach for all profile fit applications using any X-ray or neutron diffractometer:

- Microstructure information can be derived (see section 6)
- Higher degrees of peak overlap can be dealt with (with significant impact specifically on quantitative Rietveld analysis)
- A minimal number of profile parameters is used (as required to describe specimen contributions, see section 5.3)
- Specimen related profile parameters have a physical meaning when modelled appropriately (absorption, microstructure)

For accurate microstructure analysis a complication arises from the fact, that in practice measured instrument functions inevitably contain specimen contributions coming from the reference material. Even an ideally crystalline reference material will always contribute size broadening at a minimum. In addition, specimen transparency is another potential source of specimen broadening, whereby the measured instrument function is linked to specimen preparation (packing density). As a result, the broadening by the actual sample tends to be underestimated. This can get significant at smaller levels of microstructure broadening, leading to overestimated microstructure effects.

5.2.2.2.2 Calculated instrument function approach

The calculated instrument function approach (fundamental parameters approach, FPA) is characterized by functions $F_i(2\theta)$ representing both instrument as well as specimen contributions as illustrated in Fig. 5.11. In other words, FPA represents $Y(2\theta)$ in terms of the dimensions of the diffractometer and the physical properties of the specimen (section 5.1). When specimen effects include both absorption S_A and microstructure effects S_M, this can be written as,

$$Y(2\theta) = W \otimes G(2\theta) \otimes S_A(2\theta) \otimes S_M(2\theta), \quad (5.5)$$

whereby the instrument function in terms of the individual instrument aberration functions $G_i(2\theta)$ is given by,

$$I(2\theta) = W \otimes G_1(2\theta) \otimes G_2(2\theta) \otimes ... \otimes G_i(2\theta) \otimes ... \otimes G_n(2\theta). \quad (5.6)$$

From equations (5.2) throughout (5.6) it is seen that FPA is a special case of convolution based profile fitting, where all profile parameters have a physical meaning. From equation (5.5) it is also seen that FPA explicitly distinguishes between specimen transparency and microstructure, and therefore allows the independent treatment of the effective mean linear absorption coefficient or the thickness of a non-infinitely thick specimen (Kern and Coelho, 1998).

Important advantages are,

- FPA is an universal approach to profile fitting in which parameters fitted are physically identifiable and measurable. The validity of the fitted terms is therefore self-evident.
- FPA is ideal for the characterisation of standard reference materials.
- FPA automatically corrects peak shifts caused by geometrical instrument contributions and specimen transparency, significantly improving the accuracy of peak positions and derived lattice parameters. As a result, refined zero point and sample height errors no longer serve as the garbage can for misfits. This again makes FPA ideal for the characterisation of standard reference materials.
- Using suited standard reference materials it is possible to unambiguously identify whether or not a diffractometer is optimally aligned in terms of both 2θ accuracy and resolution for the used set-up.

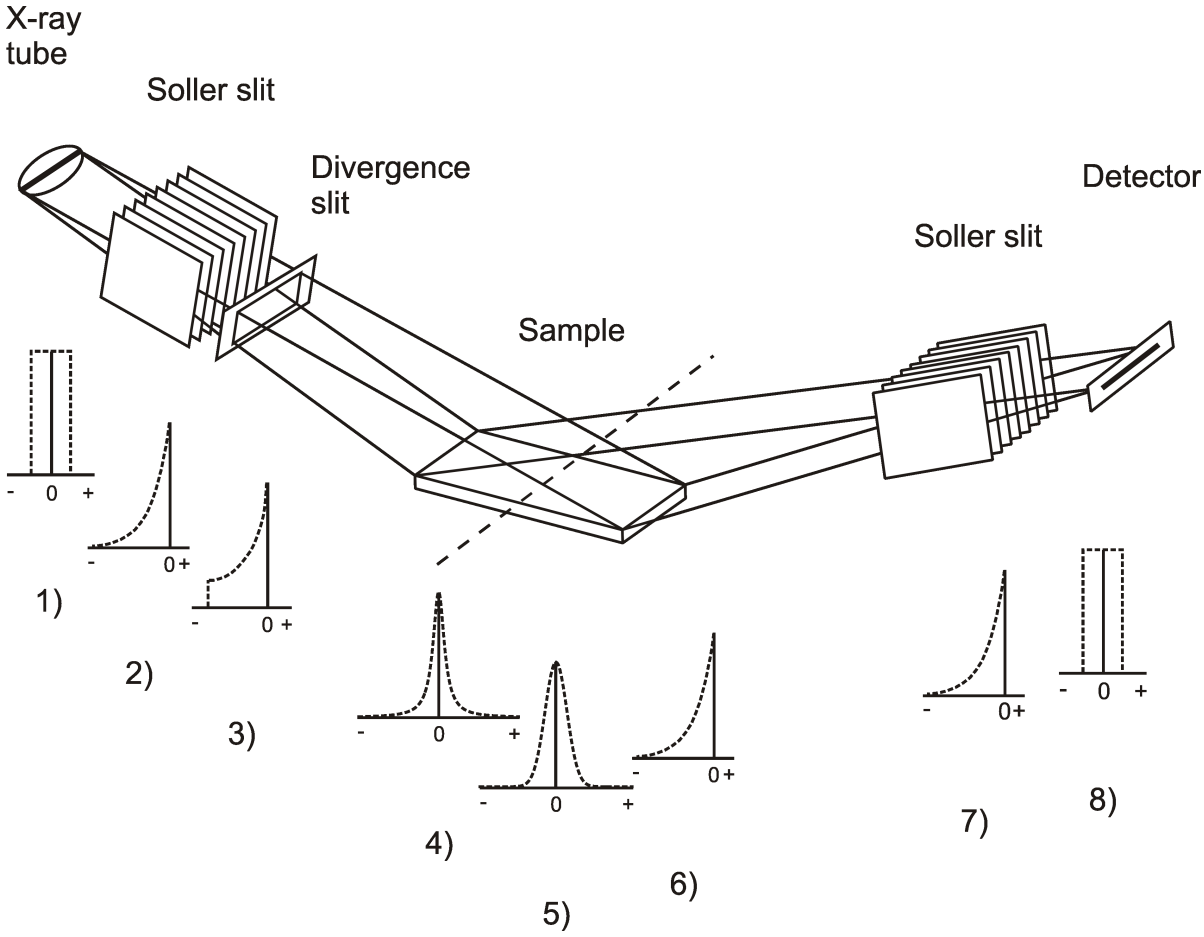


Fig. 5.11: Schematic representation of the fundamental parameters approach for a divergent beam diffractometer showing the principal optical components and the specimen together with their related aberation functions, compare section 5.1: 1) finite width of the X-ray source, 2) primary axial divergence, 3) equatorial divergence, 4), 5) crystallite size and microstrain broadening, 6) specimen transparency, 7) secondary axial divergence, 8) receiving slit width.

5.3 Implementation in TOPAS

TOPAS supports the whole range of convolution based profile fitting applications as discussed in section 5.2, i.e.

1. Empirical profile fitting (arbitrary parameterization of measured line profile shapes by convolution of any appropriate functions)
2. Instrument function approach (explicit discrimination of instrument and specimen contributions) using either
   - measured instrument functions, or
   - calculated instrument functions

The implementation of convolution based profile fitting in TOPAS has been extensively published in literature. The basics principles and the mathematical details

have been discussed by Cheary and Coelho (1992, 1994, 1998a,b,c) and Cheary et al. (2004). Two monographs providing an overview about convolution based profile fitting and its range of application have been published by Kern et al. (2004) and Kern (2008).

5.3.1 Peak generation and peak types

Phase peaks P are generated as follows:

$$P = EM(peak\_type) \otimes \text{Convolutions } scale\ all\_scale\_pks\ IP \quad (5.7)$$

where the emission profile (EM) is first generated with emission profile lines of type *peak_type*. Phase peaks are then convoluted with any defined convolutions, multiplied by the *scale* parameter (if present), multiplied by *all_scale_pks* (the cumulative value of all *scale_pks* equations applied to a phase, if present), and then multiplied by an intensity parameter IP.

For *xo_Is*, *d_Is* and *hkl_Is* phases the intensity is given by the *I* parameter of a phase peak. For *str* phases IP corresponds to the square of the structure factor F^2(hkl). Convolutions are normalized and thus do not change the area under a peak except for the *capillary_diameter_mm* and *lpsd_th2_angular_range_degrees* convolutions.

The area under the emission profile is determined by the sum of the *la* parameters; typically they add up to 1.

The following *peak_type*'s are available in TOPAS:

- *fp* : First Principles
- *pv* : Pseudo-Voigt
- *spvii* : Split-PearsonVII
- *spv* : Split-PseudoVoigt

The definitions of the pseudo-Voigt and PearsonVII functions are provided in Table 5.1 (symmetric functions) and Table 5.2 (split functions). The following terms are used:

Symmetric functions:

- *x* : (2θ-$2\theta_k$) where $2\theta_k$ is the position of the kth reflection
- *fwhm* : Full width at half maximum
- η : PV mixing parameter

Asymmetric functions:

- *fwhm1*, *fwhm2* : Fwhm for the left and right composite function
- *m1*, *m2* : Exponents for the composite functions
- $\eta 1$, $\eta 2$: PV mixing parameters for the composite functions

Table 5.1: Unit area peak types for symmetric functions.

Profile Function:	Definition:
Gaussian, $G_{UA}(x)$	$G_{UA}(x)=\left(\frac{g_1}{fwhm}\right)Exp\left(\frac{-g_2x^2}{fwhm^2}\right)$ where $g_1=2\sqrt{(Ln(2)/\pi)}$, $g_2=4\,Ln(2)$
Lorentzian, $L_{UA}(x)$	$L_{UA}(x)=\left(\frac{l_1}{fwhm}\right)\Big/\left(1+\frac{l_2x^2}{fwhm^2}\right)$ where $l_1=2/\pi$, $l_2=4$
PseudoVoigt, $PV_{UA}(x)$	$PV=\eta\,L_{UA}(x)+(1-\eta)G_{UA}(x)$

Table 5.2: Unit area peak types for split functions.

Profile Function:	Definition
Split PearsonVII, SPVII	$SPVII=PVII\_Left+PVII\_Right$ where $PVII\_Left=\left(1+b_1x^2\right)^{-m1}\Big/a \quad \text{for}\left(-\infty<x<0\right)$ $PVII\_Right=\left(1+b_2x^2\right)^{-m2}\Big/a \quad \text{for}\left(0<x<\infty\right)$ $a=(\tfrac{1}{2})\Gamma(\tfrac{1}{2})\left[\frac{\Gamma(m1-\tfrac{1}{2})}{\Gamma(m1)\sqrt{b_1}}+\frac{\Gamma(m2-\tfrac{1}{2})}{\Gamma(m2)\sqrt{b_2}}\right]$ $b_1=\left(2^{-m1}-1\right)\Big/h1^2$, $b_2=\left(2^{-m2}-1\right)\Big/h2^2$ $fwhm1=2\,h1$, $fwhm2=2\,h2$ $fwhm=h1+h2$
Split PseudoVoigt, SPV	$SPV=2\left(PV\_Left+a\,PV\_Right\right)/(1+a)$ where $PV\_Left=PV(h1,\eta1) \quad \text{for}\left(-\infty<x<0\right)$ $PV\_Right=PV(h2,\eta2) \quad \text{for}\left(0<x<\infty\right)$ $a=(PV\_Left(x=0)/(PV\_Right(x=0)$ $fwhm1=2\,h1$, $fwhm2=2\,h2$ $fwhm=h1+h2$

For classic analytical full pattern fitting the macros PV_Peak_Type, PVII_Peak_Type, TCHZ_Peak_Type can be used. These macros use the following relationships to describe profile width and shape as smooth functions of 2θ:

PV_Peak_Type:

fwhm = *ha* + *hb* tanθ + *hc*/cosθ

η = *lora* + *lorb* tanθ + *lorc*/cosθ

where *ha*, *hb*, *hc*, *lora*, *lorb*, *lorc* are refineable parameters

PVII_Peak_Type:

fwhm1 = *fwhm2* = *ha* + *hb* tanθ + *hc*/cosθ

m1 = *m2* = 0.6 + *ma* + *mb* tanθ + *mc*/cosθ

where *ha*, *hb*, *hc*, *ma*, *mb*, *mc* are refineable parameters

TCHZ_Peak_Type:

The modified Thompson-Cox-Hastings pseudo-Voigt "TCHZ" is defined as (e.g. Young, 1993):

η = 1.36603 q - 0.47719 q^2 + 0.1116 q^3

where

- q = Γ_L/Γ
- Γ = (Γ_G^5 + A$\Gamma_G^4\Gamma_L$ + B$\Gamma_G^3\Gamma_L^2$ + C$\Gamma_G^2\Gamma_L^3$ + D$\Gamma_G\Gamma_L^4$ + Γ_L^5)$^{0.2}$ = fwhm
  A = 2.69269, B = 2.42843, C = 4.47163, D = 0.07842
- Γ_G = (U tan2θ + V tanθ + W + Z/cos^{2}θ)$^{0.5}$
- Γ_L = X tanθ +Y/cosθ

with *U*, *V*, *W*, *X*, *Y*, *Z* as refineable parameters.

5.3.2 Source emission profiles

A source emission profile can comprise k emission profile lines, EM_k, each of which is a Voigt comprising the following parameters:

- *la*: Area under the emission profile line
- *lo*: Wavelength in [Å] of the emission profile line
- *lh*: Lorentzian HW of the emission profile line in [milli-Å] that is convoluted into the emission profile line
- *lg*: Gaussian HW of the emission profile line in [milli-Å] that is convoluted into the emission profile line

The reserved parameter name Lam is assigned the *lo* value of the EM_k line with the largest *la* value, this EM_k will be called EMREF. It is used to calculate d-spacings.

The interpretation of EM data is dependent on *peak_type*. For all peak types the position $2\theta_k$ calculated for a particular emission line for a particular Bragg position of 2θ is determined as follows:

$$2\theta_k = ArcSin\left(\frac{EM(k,lo)}{2d}\right)\frac{360}{\pi}$$

where

$$2d = EMREF(lo)/Sin(\theta)$$

2θ for *xo_Is* phases corresponds to the *xo* parameter. 2θ for *d_Is* phases is given by the Bragg equation 2θ = ArcSin(Lam / (2 d)) 360/Pi where d corresponds to the value

of the *d* keyword parameter. 2θ values for *str* and *hkl_ls* phases are calculated from the lattice parameters.

The $FWHW_k$ in [°2θ] for an EM_k line is determined from the following relations:

1. **For fp peak types:**

$$FWHM_k = \left(\frac{EM(k,lh)}{Lam}\right)\frac{Tan(\theta)360}{\pi}$$

2. **For pv peak types:**

$$FWHM_k = \frac{pv\_fwhm\,EM(k,lh)}{EMREF(lh)}$$

3. **For spvii peak types:**

$$FWHM_k = \frac{2h1\,EM(k,lh)}{EMREF(lh)} \qquad FWHM_k = \frac{2h2\,EM(k,lh)}{EMREF(lh)}$$

4. **For spv peak types:**

$$FWHM_k = \frac{2spv\_h1\,EM(k,lh)}{EMREF(lh)} \qquad FWHM_k = \frac{2spv\_h2\,EM(k,lh)}{EMREF(lh)}$$

The keyword *no_th_dependence* defines an emission profile that is 2θ independent (no broadening related to spectral dispersion) and allows to fit to any XY data. When *no_th_dependence* is defined then the calculation of $2\theta_k$ is determined from

$2\theta_k = 2\theta + EM(lo, i)$

The x-axis extent (x1, x2) to which an EM line is calculated is determined by:

$$\frac{\text{"Intensity of } EM(i, x = x1 = x2)\text{"}}{\text{"Intensity of } EMREF(x = 0)\text{"}} = \text{ymin\_on\_ymax}$$

The default for *ymin_on_ymax* is 0.001.

As discussed in section 5.1.1, unlike laboratory X-ray diffractometers, a separate treatment of the source emission profile and instrument contributions is normally not of interest for synchrotron and neutron diffractometers. Also, for energy dispersive diffraction, it is necessary to transform the measured energy scale into a d-spacing scale. Usage of source emission profiles thus depends on the type of the diffraction technique employed and is described below:

1. Angle-dispersive diffraction - laboratory X-ray diffractometers
2. Angle-dispersive diffraction - synchrotron and neutron diffractometers
3. Energy-dispersive diffraction - laboratory and synchrotron X-ray diffractometers
4. Energy-dispersive diffraction - time-of-flight neutron diffractometers

5.3.2.1 Angle-dispersive diffraction - laboratory X-ray diffractometers

For analysis of angle-dispersive laboratory X-ray diffraction data TOPAS comes with a selection of predefined Kα as well as Kß source emission profiles covering the most commonly used anode target materials for fine-focus tubes (*.LAM files, located

in C:\TOPAS5\LAM by default), see Table 5.3. The filenames represent the anode material, characteristic radiation, number of EM lines, and, partly, a literature reference to either Berger (1986) or Hölzer et al. (1997). Source emission profiles without reference have been defined by the TOPAS authors.

As an example, "CuKa5_Berger" represents the Cu Kα source emission profile consisting of 5 EM lines as published by Berger (1986), and is shown in Fig. 5.12. The following keyword sequence implements the "CuKa5_Berger" source emission profile, the numerical data for *la*, *lo* and *lh* are given in Table 5.4.

```
lam
ymin_on_ymax 0.0001
   la 0.0159  lo 1.534753  lh 3.6854
   la 0.5691  lo 1.540596  lh 0.437
   la 0.0762  lo 1.541058  lh 0.6
   la 0.2517  lo 1.54441   lh 0.52
   la 0.0871  lo 1.544721  lh 0.62
```

For all source emission profiles listed in Table 5.3 macros are available to simplify the use of *lam* (see also section 13.3.3).

The macro

```
CuKa5_Berger(0.0001)
```

is equivalent to the keyword sequence above, where the value 0.0001 defines *ymin_on_ymax*.

Table 5.3: Predefined source emission profiles for angle-dispersive laboratory X-ray diffraction data.

Wavelength		LAM File
Co	K$\alpha_{1,2}$	CoKa3*, CoKa7_Holzer
	Kβ	CoKb6_Holzer
Cr	K$\alpha_{1,2}$	CoKa3, CoKa7_Holzer
	Kβ	CoKb6_Holzer
Cu	Kα_1	CuKa1, CuK1sharp*
	K$\alpha_{1,2}$	CuKa2_analyt, CuKa2*, CuKa4_Holzer, CuKa5*, CuKa5_Berger
	Kβ	CuKb4_Holzer
Fe	K$\alpha_{1,2}$	FeKa7_Holzer,
	Kβ	FeKb4_Holzer
Mn	K$\alpha_{1,2}$	MnKa7_Holzer,
	Kβ	MnKb5_Holzer
Mo	K$\alpha_{1,2}$	MoKa2
	Kβ	---
Ni	K$\alpha_{1,2}$	NiKa5_Holzer,
	Kβ	NiKb4_Holzer

*) These source emission profiles are obsolete and will be removed from future TOPAS distributions.

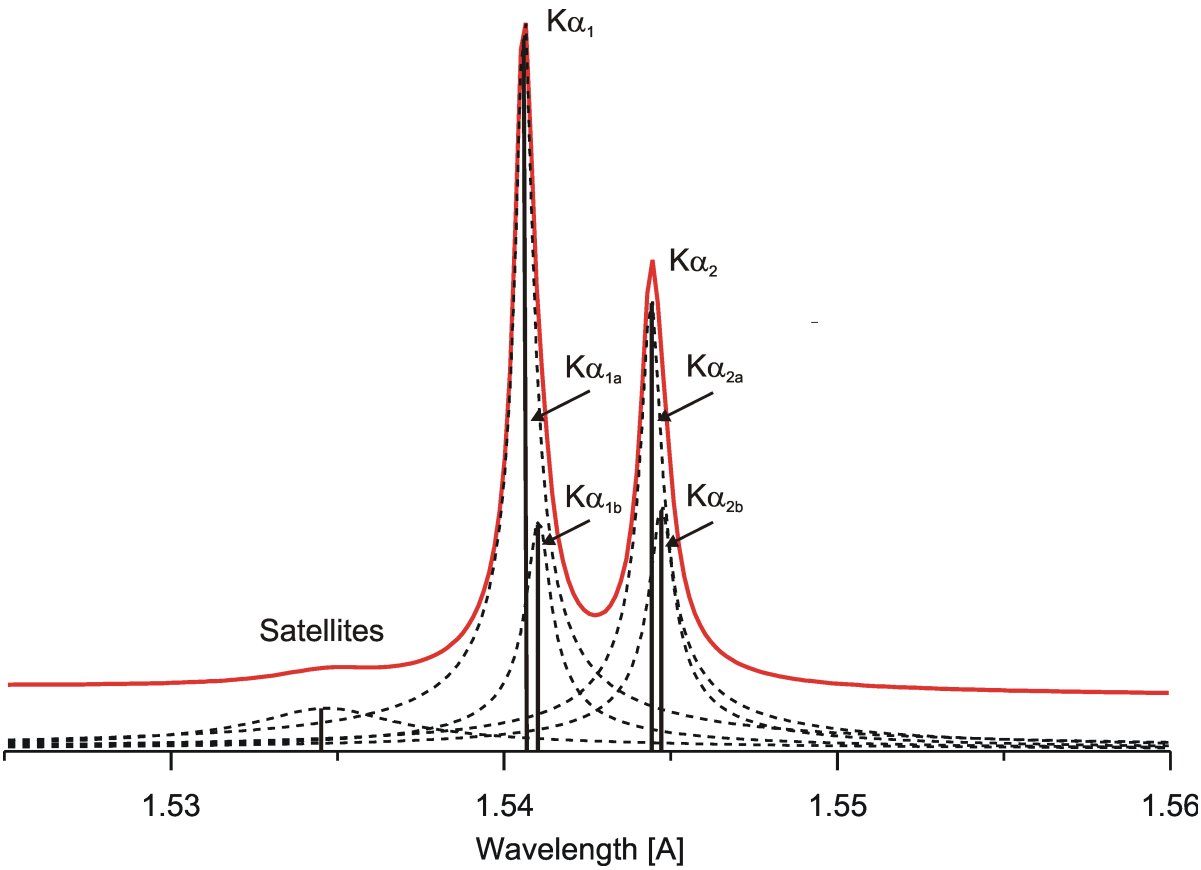


Fig. 5.12: CuKα emission profile (square-root scale). An optimum phenomenological representation is achieved by the sum of 5 Voigt functions (labelled "Satellites", "$K\alpha_{1a}$", "$K\alpha_{1a}$", "$K\alpha_{1a}$", "$K\alpha_{1a}$", see also Table 5.4).

Table 5.4: "CuKa5_Berger" source emission profile data, see text and Fig. 5.12.

Emission profile	**Emission lines**	***Io* [Å]**	***Ia* [%]**	***Ih* [mili-Å]**
CuKa5_Berger	Satellites	1.534753	1.59	3.6854
	$K\alpha_{1a}$	1.540596	56.91 (EMREF)	0.437
	$K\alpha_{1b}$	1.541058	7.62	0.6
	$K\alpha_{2a}$	1.544410	25.17	0.52
	$K\alpha_{2b}$	1.544721	8.71	0.62

Good practices require to check the suitability of any given source emission profile for a particular instrument.

For diffractometers exclusively equipped with metal Kß filters the Berger (1986) and Hölzer et al. (1997) source emission profile models can be used for accurate profile fitting. Treatment of "tube-tails" and absorption edge artefacts is described in sections 5.3.2.1.1 and 5.3.2.1.2, respectively. Generally, source emission profiles with a larger number of EM lines (Table 5.3) will perform better and should be preferred.

For laboratory diffractometers equipped with any monochromators or mirrors it is necessary to determine the actual source emission profile. Although first principles calculations of the wavelength transmission function through ideal monochromators and mirrors are possible, it is currently more practical to determine experimentally a “learned” spectrum. This can be done by modifying the phenomenological "sum of Lorentzians" representation in energy space or λ space to fit the spectrum entering the detector. In broad terms, monochromators reduce the width of the wavelength distribution and tend to truncate the tails of spectra (section 5.1.1.1). A number of approaches can be used to accommodate these changes:

- Modify the number of EM lines used and modify their relative intensities.
- If required, additionally represent the components of the wavelength distribution as Voigt or pseudo-Voigt functions rather than Lorentzians, to limit the extension of the profile tails.
- For mirror setups it may be necessary to additionally introduce wavelength dependent shifts to accommodate wavelength spreads by e.g. providing *lo* in the form of an equation.

The parameters of each representation can be obtained by analysing and fitting the high angle profiles from a reference line profile standard, such as LaB_6 SRM 660a, or a single crystal disc, such as a 111 wafer of silicon. The advantage of using measured source emission profiles is the inclusion of any aberrations introduced by monochromator or mirror misalignment into the model; any realignments, however, will require redetermination.

5.3.2.1.1 Source emission profiles with tube tails

For accurate line profile analysis it may be necessary to modify the simple hat function model for describing the finite width of the X-ray source (section 5.1.1.2) to accommodate for the so-called "tube tails" effect (Bergmann et al., 2000). Where "tube tails" are present, the source width aberration function can be better approximated by the sum of a sharp and a broad hat function as shown in Fig. 5.13 using the *stacked_hats_conv* keyword. The parameters introduced to describe tube tails are the extents of the high and low angle tails, Z_1 and Z_2, and the intensity of the tail f relative to the intensity at the tube focus. In most instances the intensity of the tails is ≤ 0.1% of the peak intensity and may be only significant when analyzing intense lines. The tails themselves are not necessarily symmetric with respect to the tube focus and can extend over a 2θ range up to 1° 2θ on both sides.

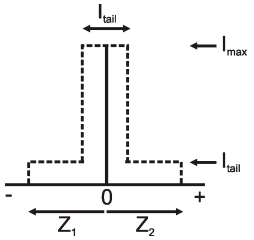


Fig. 5.13: "Tube tails" aberation model containing additional parameters $f = I_{tail}/I_{max}$ and angular widths Z_1 and Z_2 from the central maximum.

The Tube_Tails macro simplifies the use of *stacked_hats_conv*; its use is demonstrated in the tutorial example 660A53.PRO[1].

5.3.2.1.2 Source emission profiles with absorption edges

Metal Kß filters introduce absorption edges at the low energy side of diffraction peaks, dependent on the wavelength/filter material and its thickness. With position sensitive detectors (PSDs) absorption edges are much more pronounced due to the high intensity data typically collected; while for point detectors absorption edges usually are obscured by counting statistics, absorption edges are clearly visible in most patterns recorded using PSDs. Absorption edges frequently prevent the accurate description specifically of peaks tail regions, und thus often represent a major part of the remaining misfit to the data, specifically at low angles.

The Absorption_Edge_Correction macro allows an excellent fit to absorption edges with significantly lowered R_{WP}. This is achieved by

1. Extending source emission profiles to include transmitted Kß and Bremsstrahlung, adaptively damped by the transmission of the filter material
2. Blending of the resulting peak profile function with a filter function (Fig. 5.14) considering
   - the position of the absorption edge according to the respective filter material,
   - the change of transmission at the respective energy of the absorption edge,
   - the amplitude of the filter function according to the transmission (thickness) of the respective filter material, and
   - a broadening of the sharp transmission edge of the step function that is related to the real micro-structure of the sample

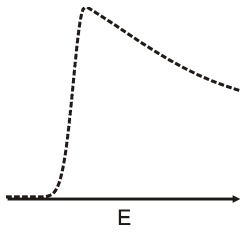


Fig. 5.14: Filter function used to describe absorption edges.

Tutorial files comprise several examples for fitting of absorption edges[2].

5.3.2.2 Angle-dispersive diffraction - synchrotron and neutron diffractometers

Data analysis involves definition of a source emission profile where the width of the EM line(s) (*lh* and / or *lg* parameters) is set significantly smaller than the widths of the observed line profile shapes, e.g.

1 ...\TUTORIAL\MISCELLANEOUS\MINIMIZATION\TUBE TAILS\
2 ...\TUTORIAL\MISCELLANEOUS\MINIMIZATION\ABSORPTION EDGES

```
lam
ymin_on_ymax 0.0001
   la 1  lo 0.6998  lh 0.0001
```

Note that the *lh* parameter in this example implies a delta function but has no physical meaning. The observed line profile shape is obtained by convolution of appropriate functions as discussed in section 5.2.2.1 for empirical profile fitting, and in section 5.2.2.2.1 for determination of a measured instrument function.

Tutorial files comprise several examples for fitting of angle-dispersive synchrotron and neutron diffraction data[1].

5.3.2.3 Energy-dispersive diffraction - laboratory and synchrotron X-ray diffractometers

The basis of diffraction is the satisfaction of Bragg's law, where energy is related to wavelength via

$$\lambda = 2d\sin\theta,$$

$$E_{keV} = \frac{hc}{\lambda}6.24\times10^{25} = \frac{12.4}{\lambda}$$

$$E_{keV} = 6.2/\left(d\sin\theta\right)$$

where E_{keV} is the energy of the incident radiation in keV, *h* is Planck's constant, *c* is the speed of light and λ is the wavelength associated with that energy in Å.

Accordingly, analysis of energy-dispersive X-ray diffraction data requires definition of a source emission profile as well as a transformation of the measured energy scale into a d-spacing scale using the *pk_xo* keyword, e.g.,

```
lam
   ymin_on_ymax 0.0001
   la 1 lo 0 lh 1
pk_xo = 6.2 / D_spacing Sin(ThB 2 Pi / 360);
```

where 2 ThB is the fixed Bragg diffraction angle. Use of *pk_xo* turns on *no_th_dependence* by default.

Note that both the *lo* and *lh* parameters do not have a physical meaning.

The observed line profile shape is obtained by convolution of appropriate functions as discussed in section 5.2.2.1 for empirical profile fitting, and in section 5.2.2.2.1 for determination of a measured instrument function.

Tutorial files comprise an example for fitting of energy-dispersive synchrotron data[2].

5.3.2.4 Energy-dispersive diffraction - time-of-flight neutron diffractometers

Similar to energy-dispersive X-ray diffraction (section 5.3.2.3), analysis of time-of-flight neutron diffraction data requires definition of a source emission profile as well as a transformation of the measured energy scale into a d-spacing scale using the *pk_xo* keyword, e.g.

1 ...\TUTORIAL\STRUCTURE DETERMINATION AND REFINEMENT\
2 ...\TUTORIAL\MISCELLANEOUS\EDXRD\

```
lam
   ymin_on_ymax 0.0001
   la 1 lo 0 lh 1
pk_xo = t0 + t1 D_spacing + t2 D_spacing^2;
```

Note that both the *lo* and *lh* parameters do not have a physical meaning. Use of *pk_xo* turns on *no_th_dependence* by default.

The three parameters t0, t1 and t2 are characteristic of a given counter bank on a TOF powder diffractometer. Accurate values for t0, t1 and t2 must be obtained by fitting to a powder diffraction pattern of a standard material and are typically provided by the instrument scientist.

The observed line profile shape is obtained by convolution of appropriate functions as discussed in section 5.2.2.1 for empirical profile fitting, and in section 5.2.2.2.1 for determination of a measured instrument function.

Tutorial files comprise an example for fitting of TOF neutron data[1].

5.3.3 Geometrical instrument contributions

Geometrical instrument contributions used for peak profile synthesis are generated from generic convolutions. Table 5.5 lists the contributions discussed in section 5.1.1.2 together with related generic convolutions. Also listed are macros simplifying the use of these convolutions. The mathematical background is described by Cheary and Coelho (1992; 1994; 1998a,b,c). The actual macro implementations can be viewed in the file TOPAS.INC.

The keyword *user_defined_convolution* provides for user-defined convolutions that are convoluted into phase peaks, see also section 5.4.

Table 5.5: Instrument contributions and related generic convolutions / macros.

Contribution	Generic convolution	Macro(s)
Finite width of the X-ray source (fine focus)	*hat*	Source_Width
	stacked_hats_conv	Tube_Tails
Horizontal divergence	*one_on_x_conv*	Divergence, Variable_Divergence
Axial divergence	*axial_conv*	Full_Axial_Model
Analyzer slit	*hat*	n.a.
Point detector (receiving slit)	*hat*	Slit_Width
Linear PSD	*lpsd_th2_angular_range_degrees*	n.a.
User-defined	*user_defined_convolution*	n.a.

1 ...\TUTORIAL\MISCELLANEOUS\TOF NEUTRON DATA\

5.3.4 Specimen contributions

Specimen contributions used for peak profile synthesis are generated from generic convolutions. Table 5.6 lists the contributions discussed in section 5.1.2 together with related generic convolutions. Also listed are macros simplifying the use of these convolutions. The mathematical background is described by Cheary and Coelho (1992; 1994; 1998a,b,c). The actual macro implementations can be viewed in the file TOPAS.INC.

The keywords *user_defined_convolution* and *ft_conv* provide for user-defined convolutions that are convoluted into phase peaks, see also section 5.4.

For a detailed discussion of microstructure analysis refer to section 6.

Table 5.6: Specimen contributions and related generic convolutions and macros.

Contribution	Generic convolution	Macro(s)
Specimen transparency (Bragg-Brentano)	*exp_conv_const*	Absorption
Capillary absorption (Debye-Scherrer)	*capillary_diameter_mm*	n.a.
Microstructure broadening modelled using a Lorentzian function	*lor_fwhm*	n.a
Microstructure broadening modelled using a Gaussian function	*gauss_fwhm*	n.a
User-defined	*user-defined-convolution, ft_conv*	n.a.

5.4 Convolution processes

5.4.1 Convolutions in general

TOPAS performs the operation of convolution in various ways by means of direct convolution and Fast Fourier Transform (FFT) convolution, where, by definition, a "response function" with N_r data points is convoluted with a "peak function" with N_p data points.

Typically convolutions are broken down into a number of double summations; each of these double summations can then be calculated either directly or by using a FFT. The program uses the method that is fastest and it does this by trying to determine the number of operations required by each method (see also section 5.4.2.2).

Response functions that are known to the program are treated analytically. Response functions that are not known to the program (such as user defined convolutions) are treated as straight line segments. Convolution therefore can be between two sets of straight line segments, one set of straight line segments and an analytical expression, or simply done analytically. The extra cost of the piece wise integration is small and the benefit is a high degree of accuracy.

Apart from *lor_fwhm* and *gauss_fwhm*, the convolutions described below have discontinuities in 2θ space; their associated Fourier transform therefore is difficult to describe and hence convolution is performed in 2θ space.

Response functions that are treated as straight line segments are:

`[user_defined_convolution]`

`[capillary_diameter_mm]`

`[lpsd_th2_angular_range_degrees]`

Response functions that are analytically convoluted with the straight line segments of the peak are:

`[exp_conv_const]`

`[hat]`

`[stacked_hats_conv]`

Response functions that comprise a mixture of analytical and straight line segments are:

`[axial_conv]`

`[one_on_x_conv]`

`[circles_conv]`

lor_fwhm and *gauss_fwhm* convolutions are treated analytically with the emission profile to form the base profile. Convolutions are calculated with a step size given by:

`Peak_Calculation_Step = x_calculation_step / convolution_step`

For efficiency *x_calculation_step* should not be defined for data with equal x-axis steps; instead *rebin_with_dx_of* should be used. The following response functions are calculated at smaller step sizes without changing Peak_Calculation_Step or N_r:

```
[axial_conv]                    '(Step = Peak_Calculation_Step / 2)
[lpsd_th2_angular_range_degrees]  '(Step = Peak_Calculation_Step / 3)
[capillary_diameter_mm]           '(Step = Peak_Calculation_Step / 1 to 3)
```

In this manner a high degree of accuracy is maintained for the little extra cost in calculating the extra response function points and with the benefit of not increasing N_p*N_r. Typically a laboratory diffraction pattern can be accurately synthesized with a Peak_Calculation_Step of 0.01 to 0.02 degrees 2θ. The next step to increasing accuracy would be to increase *convolution_step* to 2 and so on.

The following convolutions are always performed directly:

```
[exp_conv_const]
[hat]
[stacked_hats_conv]
```

Calculating derivatives of parameters that are a function of a convolution can be demanding. Most convolutions however that have multiple dependent parameters require only one recalculation of the convolution; exceptions are *ft_conv*, *WPPM_ft_conv* and *user_defined_convolution*. In the case of convolutions that comprise multiple convolution parameters, for example, *axial_conv* with its convolution parameters of *primary_soller_angle* etc., then a recalculation for each of the convolution parameters is required.

5.4.2 Fourier Transforms

5.4.2.1 The ft_conv keyword

The keyword *ft_conv* describes a Fourier Transform (FT) of a response function that is convoluted into phase peaks using a Fast Fourier Transform (FFT); for example, to convolute a Voigt into a phase the following can be used:

```
ft_conv = Exp(-(Pi FT_K gfwhm)^2 / (4 Ln(2)) - Pi FT_K lfwhm);
   ft_min = 1e-8; ' this is the default and is optional
   ft_x_axis_range = 40 lfwhm;
```

The convolution theorem is used here multiplying the FT of a Gaussian by the FT of a Lorentzian. Were the Fourier transforms separately defined then the program will internally use the convolution theorem.

FT_K is a reserved parameter name and it returns the transform k divided by the x-axis range of the peak; this range includes *ft_x_axis_range*.

ft_x_axis_range can be an equation that needs to be set such that the transform decays to near zero; peak tails will otherwise be incorrect. A Lorentzian for example needs a large *ft_x_axis_range* for accurate x-axis tails.

ft_min defines the smallest value to which the transform is calculated to. For example, an already broadened peak in x-axis space will have a relatively narrow transform; the calculation of the transform is therefore terminated when FT(k)/FT(k=0)<*ft_min*. Transform values for larger k are then set to zero. If(,,) constructs can instead be used within the transform equation for further control; for example:

```
ft_conv = If (FT_K > D, FT_Break, Sphere(FT_K, D));
```

Here the calculation of the FT is terminated when FT_K>D using FT_Break.

Get(ft_0) returns FT(k=0) and can be used within the *ft_conv* equation, for example,

```
ft_conv = {
   def a = Exp(-Pi FT_K lf);
   return If(a < 1e-6 Get(ft_0), FT_Break, a);
}
```

ft_conv integrates with convolutions that are performed in direct space. It can be used within peak stack operations and it can be a function of the reserved parameter names and keywords:

H, K, L, M, Th, Xo, D_spacing, FT_K and *spherical_harmonics_hkl*

Multiple *ft_conv*(s) can be defined at either the *xdd* or phase level. When defined at the *xdd* level the convolution is applied to all phases of that *xdd*.

For a typical refinement, an *ft_conv* used to describe a Voigt is nearly as fast as the analytical equivalent. The speed of the analytical convolution is greater not simply because describing the peak analytically is faster but because derivatives of multiple parameters for *lor_fwhm* (or *gauss_fwhm*) requires only one peak calculation; whereas for *ft_conv* the peak is recalculated for each independent parameter that it is a function of.

For high accuracy the range of the peak, as defined with *ft_x_axis_range*, needs to be large, up to 400 FWHM for a Lorentzian; in these cases the *ft_conv* is considerably slower.

5.4.2.2 FFT versus direct summation

If a response function is known in x-axis space then it is often best to perform the convolution in x-axis space rather than describing the FT of the response function using *ft_conv*. The keyword *user_defined_convolution* can be used to perform convolution in x-axis space and the speed at which it operates is as fast or faster than *ft_conv* depending on the x-axis range of the response function. For each peak *user_defined_convolution* estimates the computational effort required to perform the convolution directly and with a FFT and chooses the one with the least computational effort. The FT for functions with discontinuities often cannot be described analytically and hence the usefulness of *user_defined_convolution*.

Typically a FFT convolution for response functions that are treated as histograms is quoted as comprising $O(N \log_2 N)$ operations (Cooley–Tukey algorithm for example) versus a direct convolution that comprises $O(N^2)$ operations, and that direct convolution is only faster for N<128. See e.g.:
https://ccrma.stanford.edu/~jos/ReviewFourier/FFT_Convolution_vs_Direct.html.
However, in XRD work a direct convolution rather than an FFT working on real numbers is often faster for N ~< 256 to 512 as the comparison of the $O(N \log_2 N)$ versus $O(N^2)$ is invalid.

To see why consider a response function comprising 3 points and a peak comprising 5 points. A convolution can be pictured as the response function R moving along the peak P as follows:

```
P    0 0 0 1 1 1 1 1 0 0 0
R      - - x
R        - x x
R          x x x
R            x x x
R              x x x
R                x x x
R                  x x -
R                    x - -
```

In this representation each "x" can be considered a multiply and in direct convolution this makes a total of 15 multiplies (N_r*N_p) and not N^2 where $N/2 \leq (N_r+N_p) \leq N$. To perform such a convolution with an FFT the number of operations is approximately $4*16*\log_2 16=256$ multiplies where 16 is the closest power of 2 to N_r+N_p. Of course FFT routines typically also have special cases for small N; nonetheless N=256 to 512 is not small and many peaks in XRD work typically comprise less points and in particular many of the response functions have a small N_r; these include axial divergence, equatorial divergence, receiving slit width, capillary convolution, LPSD convolution and often sample penetration.

Another factor favoring direct convolution for modest N_r and N_p is the fact that modern processors such as the Intel i7 are very fast when data in cache memory are arranged sequentially and accessed sequentially. In fact operations on data on a non-sequential manner can be as much as 8 times slower than for the sequential case.

5.4.3 Convolution and the peak generation stack

The emission profile of a peak P0 of a certain peak type (i.e. FP, PV etc…) is first calculated and placed onto a "Peak calculation stack". P0 analytically includes *lor_fwhm* and *gauss_fwhm* convolutions for FP and PV peak types and additionally one *hat* convolution if defined; the *hat* convolution is included analytically only if its corresponding *num_hats* has a value of 1 and if it does not take part in stack operations. Further defined convolutions are convoluted with the top member of the stack. The last convolution should leave the stack with one entry representing the final peak shape. The following keywords allow for the manipulation of the "Peak calculation stack":

`[push_peak]…`

`[bring_2nd_peak_to_top]…`

`[add_pop_1st_2nd_peak]…`

`[scale_top_peak E]…`

push_peak duplicates the top entry of the stack; *bring_2nd_peak_to_top* brings the second most recent entry to the top of the stack and *add_pop_1st_2nd_peak* adds the top entry to the second most recent entry and then pops the stack. *scale_top_peak* scales the peak at the top of the stack.

As an example use of these keywords consider the generation of back-to-back exponentials as required by GSAS time of flight peak shape 3:

```
push_peak
   prm a0 481.71904 del = 0.05 Val + 2;
   prm a1 -241.87060 del = 0.05 Val + 2;
   exp_conv_const = a0 + a1 / D_spacing;
bring_2nd_peak_to_top
   prm b0 -3.62905 del = 0.05 Val + 2;
   prm b1 6.44536 del = 0.05 Val + 2;
   exp_conv_const = b0 + b1 / D_spacing^4;
add_pop_1st_2nd_peak
```

The first statement *push_peak* pushes P0 onto the stack leaving two peaks on the stack, or,

Stack = P0, P0

The top member is then convoluted by the first *exp_conv_const* convolution, or,

Stack = P0, P0 ⊗ *exp_conv_const*

where ⊗ denotes convolution. *bring_2nd_peak_to_top* results in the following:

Stack = P0 ⊗ *exp_conv_const*, P0

and the next convolution results in:

Stack = P0 ⊗ *exp_conv_const*, P0 ⊗ *exp_conv_const*

Thus the stack contains two peaks convoluted with exponentials. The last statement *add_pop_1st_2nd_peak* produces:

Stack = P0 ⊗ *exp_conv_const* + P0 ⊗ *exp_conv_const*

5.4.4 Speed / Accuracy and "peak_buffer_step"

For computational efficiency phase peaks are calculated at predefined 2θ intervals in a “peaks buffer“. In between peaks are determined by stretching and interpolating. Use of the peaks buffer dramatically reduces the number of peaks actually calculated. Typically no more than 50 to 100 peaks are necessary in order to accurately describe peaks across a whole diffraction pattern. The following parameters affect the accuracy of phase peaks:

```
[peak_buffer_step !E]
[convolution_step #]
[ymin_on_ymax #]
[aberration_range_change_allowed !E]
```

Default values for these are typically adequate. *peak_buffer_step* determines the maximum x-axis spacing between peaks in the peaks buffer, it has a default value of 500*Peak_Calculation_Step. A value of zero will force the calculation of a new peak in the peaks buffer for each peak of the phase. Note that peaks are not calculated for x-axis regions that are void of phase peaks.

convolution_step defines an integer corresponding to the number of calculated data points per measurement data point used to calculate the peaks in the peaks buffer,

see *x_calculation_step* as well. Increasing the value for *convolution_step* improves accuracy for data with large step sizes or for peaks that have less than 7 data points across the FWHM.

ymin_on_ymax determines the x-axis extents of a peak (see also section 5.3.2).

aberration_range_change_allowed describes the maximum allowed change in the x-axis extent of a convolution aberration before a new peak is calculated for the peaks buffer. For example, in the case of *axial_conv* the spacing between peaks in the peaks buffer should be small at low angles and large at high angles. *aberration_range_change_allowed* is a dependent of the peak type parameters and convolutions as shown in Table 5.7.

Small values for *aberration_range_change_allowed* reduce the spacing between peaks in the peaks buffer and subsequently increase the number of peaks in the peaks buffer.

Table 5.7: Default values for *aberration_range_change_allowed* of the following peak type parameters and convolutions.

	Default for *aberration_range_change_allowed*
m1, *m2*	0.05
h1, *h2*, *pv_fwhm*, *spv_h1*, *spv_h2*	Peak_Calculation_Step
pv_lor, *spv_l1*, *spv_l2*	0.01
hat, *whole_hat*, *half_hat*	Peak_Calculation_Step
axial_conv, *one_on_x_conv*, *exp_conv_const*, *circles_conv*	Peak_Calculation_Step
lor_fwhm, *gauss_fwhm*	Peak_Calculation_Step for all *lor_fwhm* and *gauss_fwhm* defined.

5.5 Criteria of fit

Criteria of fit used in TOPAS are shown in Table 5.8, see Young (1993) or Toby (2006) for details.

Table 5.8: Criteria of fit. $Y_{o,m}$ and $Y_{c,m}$ are the observed and calculated data respectively at data point m, Bkg_m the background at data point m, M the number of data points, P the number of parameters, w_m the weighting given to data point m which for counting statistics is given by $w_m=1/\sigma(Y_{o,m})^2$ where $\sigma(Y_{o,m})$ is the error in $Y_{o,m}$, and $I_{"o",k}$ and $I_{c,k}$ the "observed" and calculated intensities of the kth reflection.

Criteria of fit	Definition	
"R-pattern", R_p "R-pattern", R_p' (background corrected)	$R_p = \frac{\sum \lvert Y_{o,m} - Y_{c,m} \rvert}{\sum Y_{o,m}}$	$R_p' = \frac{\sum \lvert Y_{o,m} - Y_{c,m} \rvert}{\sum \lvert Y_{o,m} - Bkg_m \rvert}$
"R-weighted pattern", R_{WP} "R-weighted pattern", R_{WP}', (background corrected)	$R_{wp} = \sqrt{\frac{\sum w_m (Y_{o,m} - Y_{c,m})^2}{\sum w_m Y_{o,m}^2}}$	$R_{wp}' = \sqrt{\frac{\sum w_m (Y_{o,m} - Y_{c,m})^2}{\sum w_m (Y_{o,m} - Bkg_m)^2}}$
"R-expected", R_{exp} "R-expected", R_{exp}' (background corrected)	$R_{\exp} = \sqrt{\frac{M-P}{\sum w_m Y_{o,m}^2}}$	$R_{\exp}' = \sqrt{\frac{M-P}{\sum w_m (Y_{o,m} - Bkg_m)^2}}$
"Goodness of fit", GOF	$GOF = chi = \frac{R_{wp}}{R_{\exp}} = \sqrt{\frac{\sum w_m (Y_{o,m} - Y_{c,m})^2}{M-P}}$	
"R-Bragg", R_B	$R_B = \frac{\sum \lvert I_{"o",k} - I_{c,k} \rvert}{\sum I_{"o",k}}$	
"Durbin-Watson statistic", d Durbin & Watson, 1971; Hill & Flack, 1987	$d = \frac{\sum_{m=2}^{M} (\Delta Y_m - \Delta Y_{m-1})}{\sum_{m=1}^{M} (\Delta Y_m)^2}; \quad \Delta Y_m = Y_{o,m} - Y_{c,m}$	

6 MICROSTRUCTURE ANALYSIS

“The microstructure of materials […] is a notion that comprises all aspects of the atomic arrangement in the material that should be known in order to understand its properties"

"Each […] line profile […] in the diffraction pattern represents an average over the diffracting material; […]. This indicates the strength and, at the same time, the limitation of diffraction analysis: Average values for structure parameters (microstructure parameters) are obtained [...], but the atomic configuration around an individual, isolated defect cannot be revealed in this way."

Mittemeijer & Scardi (2004)

This chapter deals with the qualitative and quantitative modelling of physically broadened line profile shapes using TOPAS' convolution based approach to X-ray and neutron powder data as described in section 5.

Convolution based profile fitting currently represents the state of the art for quantitative microstructure materials. The main advantage over traditional line profile analysis methods is the ability to refine physically sound microstructure models directly to the diffraction data, rather than deriving microstructure information from (usually arbitrary) profile parameters, which have been obtained in a separate, independent evaluation step.

Suggested reading:

- **Mittemeijer, E.J. & Scardi, P. (2004):** *Diffraction Analysis of the Microstructure of Materials.* Edited by Mittemeijer, E.J. and Scardi, P. Springer Series in Materials Science, Vol. 68, 552 pages, ISBN: 978-3-540-40519-1

6.1 General considerations

In traditional line profile analysis, observed line profile shapes are firstly characterized in terms of peak width parameters (FWHM or Integral Breadth) or Fourier coefficients using arbitrary analytical profile functions such as Gaussian, Lorentzian, (pseudo-)Voigt, and PearsonVII functions. This first step involves the separation of microstructure contributions from specimen absorption effects and instrument contributions, e.g. via difference or deconvolution. In a second, independent step, microstructure information is obtained utilizing peak width based methods (e.g. Scherrer, 1918; Williamson & Hall, 1953) or Fourier methods (chiefly, Warren & Averbach, 1950).

In convolution based line profile analysis, profiles are modelled by convolution of appropriate functions representing both instrument and specimen contributions to form the observed profile shapes, or, according to section 5.2 and equations (5.2) throughout (5.6),

$$Y(2\theta) = I(2\theta) \otimes S_A(2\theta) \otimes S_{M1}(2\theta) \otimes S_{M2}(2\theta) \otimes ... \otimes S_{Mi}(2\theta) \otimes ... \otimes S_{Mn}(2\theta) \quad (6.1)$$

where the functions S_{Mi} represent a variety of microstructure contributions such as coherent scattering domain size / shape, lattice distortions (e.g. due to dislocations), faulting (e.g., twin and deformation faults), anti-phase domain boundaries, composition fluctuations, and grain surface relaxation.

Benefits of convolution based line profile analysis are manifold. Physically meaningful profile shape functions with any shape dependence on angle and hkl direction can be constructed, based on an appropriate choice of functions representing both instrument and specimen contributions. Generally, a smaller number of refineable profile parameters is required, which significantly reduces parameter correlations. This, in combination with the instrument function constraint, allows to deal with higher degrees of peak overlap and to better distinguish between line profile tails and background.

Currently, in both traditional and convolution based line profile analysis, the most common approaches for obtaining microstructural information are based on the determination of Gaussian and Lorentzian components, related to distinct physical effects such as domain size and microstrain. The advantages of this methodology lie in its simplicity, flexibility, and ease of implementation. It is suited to follow trends qualitatively and can also allow to draw conclusions about the origins of line broadening. A commonly used approach is the so-called Double-Voigt Approach (e.g. Balzar, 1999), where the Gaussian and Lorentzian components of two Voigt functions are refined representing domain size and microstrain, respectively; for details see section 6.3.

The major disadvantages of microstructure characterization via Gaussian and Lorentzian components are due to simplification: Apart from a few cases, there is no physical law imposing Gaussian- or Lorentzian-type diffraction line profile broadening related to microstructural effects. Higher sophisticated approaches avoid physical constraints inherent to arbitrarily selected line profile functions (such as Gaussian and Lorentzian functions). Instead physically sound models are used to describe microstructural broadening, e.g. by characterizing the effects of crystallites size and shape as well as lattice defect type, density and distribution from first principles. This so-called Whole Powder Pattern Modeling (WPPM) approach currently represents the state of the art for quantitative microstructural analysis of nanocrystalline materials (Scardi & Leoni, 2001, 2004; Scardi et al. 2010). WPPM can be implemented in TOPAS in several ways via fit objects, or user-defined convolutions, or Fourier transforms. For details refer to section 6.4.

Anisotropic line broadening can be modelled in many ways. In literature, several approaches have been proposed to deal with anisotropic line broadening, and many of these have been implemented in TOPAS, or can be easily implemented by the user. Examples include, but are not limited to, second rank tensors (e.g. Le Bail & Jouanneaux, 1997), spherical harmonics (e.g. Järvinen, 1993; Popa, 1998), multi-dimensional distribution of lattice metrics (Stephens, 1999), and are discussed in section 11.5.

In most applications, anisotropic line broadening is being modelled qualitatively. Here the primary objective is the determination of more accurate peak positions and intensities in order to improve indexing, structure determination and structure

refinement results, rather than obtaining physically meaningful microstructure results. Arbitrarily selected line profile functions (such as Gaussian and Lorentzian functions) raise similar issues as for isotropic line broadening. Obtaining quantitative information can be difficult, specifically when several microstructural effects contribute, and cannot be discussed here. Recently, Leineweber (2010, 2011) discussed the extraction of quantitative information related to anisotropic microstrain, which can be implemented in TOPAS e.g. via user-defined convolutions.

6.2 Size-microstrain broadening: Terminology

Origins of (physical) line broadening are numerous. In general any lattice imperfections will cause additional line broadening, which can be dependent and independent on the reflection order:

- If a crystal is broken into smaller incoherently diffracting domains by dislocation arrays (small-angle boundaries), stacking faults, twins, or other extended imperfections, then size broadening occurs.
- Dislocations, vacancies, interstitials, substitutionals, and similar defects lead to microstrain broadening.

From the nature of the lattice imperfections noted above follows, that both effects are interconnected. For example dislocations cause lattice strain but also arrange into boundaries between incoherently diffracting domains resulting in crystallite size broadening. This is one of the reasons why any interpretation of the underlaying physics of broadening is difficult.

6.2.1 Size broadening

For parallel planes of atoms, with a space d between the planes and for a given wavelength λ, Bragg's law defines the (Bragg) angle for constructive interference.

As discussed more in detail by Jenkins & Snyder (1996), if the incident beam angle theta is away from the Bragg angle so that the phase shift becomes 1.1λ between adjacent planes, then the phase shift originating of the 6th plane down the surface will be 5.5λ resulting in destructive interference (so-called Bragg extinction). The same is true for the 2nd and 7th planes etc., thus no net scattering will occur. If theta is set closer to the Bragg angle so that the phase shift becomes 1.0001λ between adjacent planes, then the scattering of the 1st plane will be cancelled by scattering of the 5.001th plane, with a phase shift of 5000.5λ.

If the crystal is too small, then planes needed to cancel such phase shifts are not present; this is the source of size broadening. As a result intensities at lower and higher angles than the Bragg angle are observed. Crystallites larger than $2\mu m$ typically have a sufficient number of planes to allow the diffraction peak to display its inherent Darwin width.

The quantity sought in powder diffraction is crystallite size, which can not be measured directly; this is discussed in the following. In addition there is an enormous confusion in literature concerning the definition of particle size, crystal size, crystallite size, and domain size; the following definitions are appropriate, see also Fig. 6.1:

- Particles ("secondary grains") consist of one or more crystals, Fig. 6.1a. The different crystals may be separated e.g. by large angle boundaries, amorphous or crystalline interfaces.
  - Note that particle size cannot be determined by powder diffraction!
- Crystals ("primary grains") are solid, anisotropic, and chemically homogeneous materials in which the constituent atoms are arranged in an infinite, 3-dimensional periodic lattice
  - The size of a crystal is in general equal or less than the particle size
  - A crystal surface can be considered a 2-dimensional defect
- Crystallites are small crystals that, held together through highly defective boundaries, constitute a polycrystalline solid, Fig. 6.1b.
  - The size of a crystallite is in general equal or less than the crystal size
  - Crystallite size can be indirectly (!) determined by powder diffraction
- Domains are coherently diffracting volumes which do not contain 2-dimensional defects, and even do not need to be connected. As an example in Fig. 6.1c a crystallite is shown, which is "broken" into two domains as a result of stacking fault twinning ABCBA (dashed line). For a reflection hkl represented by the diffraction vector "a" there are two domains, but only one crystallite. For a reflection hkl represented by the diffraction vector "b" there may be two domains and two crystallites.
  - The size of a domain is in general equal or less than the crystallite size
  - Domain size broadening is the actual origin of "size broadening", but cannot be determined by powder diffraction!

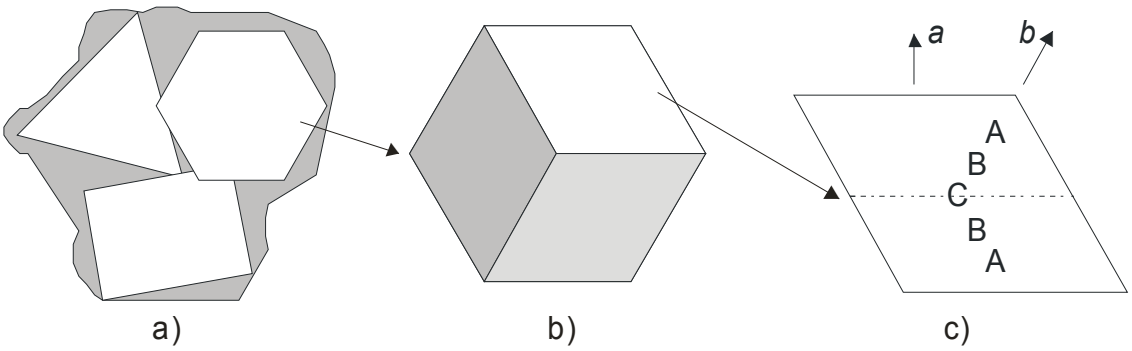
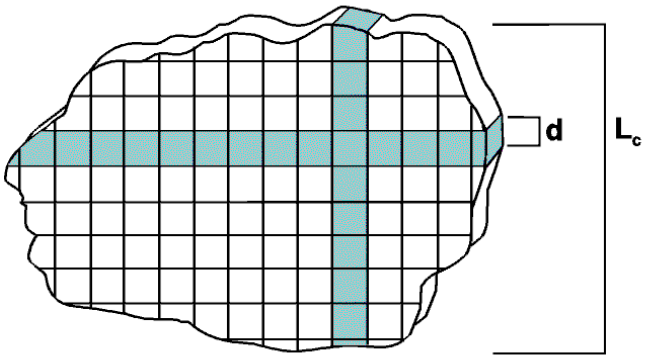


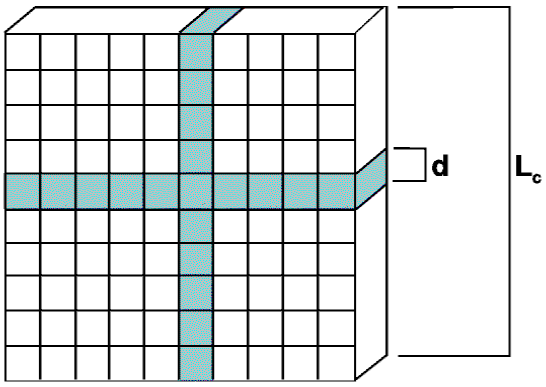
Fig. 6.1: Schematic representation of terms. a) particle and crystal size, b) crystallite size, and c) domain size.

Crystallite size and shape are no tensor properties and thus can be any; there is no relationship to crystal symmetry. Also in real polycrystal systems there are no uniform cystallite sizes and shapes but distributions. Bertaut (1949) proposed an elegant interpretation to consider size broadening without making assumptions on crystallite size and shape distributions: Crystallites are considered as made up of columns of cells along the scattering direction as illustrated in Fig. 6.2. Line broadening can then be treated in terms of a column length distribution, from which the mean column length can be related to the line profile width. As the scattering power of a column is dependent on its volume it makes sense to use volume weighted mean column

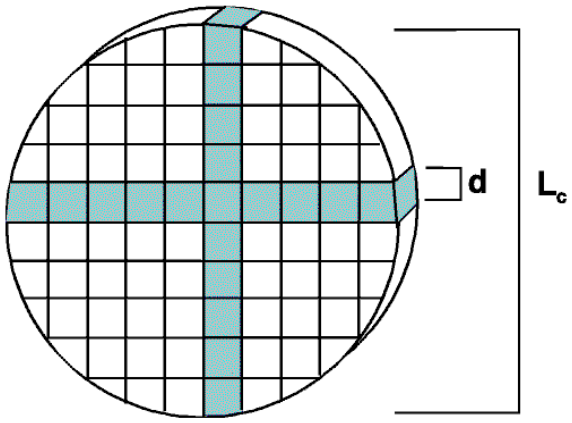
lengths, L_{Vol}. To obtain the true crystallite size, the true mean shape of the crystallites must be known in order to derive and apply a correction to the column height of each hkl, but this information is normally not available.



a) In most polycrystal systems the real crystallite shape is unknown, thus only L_{Vol} can be reported



b) For a cube and only for h00 reflections L_{Vol} is identical with the edge-length L_0 of the cube (i.e. the crystallite size for h00)



c) For a sphere a column height distribution is observed for any given hkl. L_{Vol} is not identical with the diameter of the sphere, but can be derived from the known crystallite shape: $L_0 = 4\ L_{Vol} / 3$

Fig. 6.2: Schematic representation of crystallites built up of columns perpendicular to the lattice planes (Haberkorn, 1999).

Scherrer (1918) noted that the line profile width is inversely proportional to crystallite size

$$\varepsilon = \frac{\lambda}{\beta_{FWHM(s)} \cdot \cos\theta} \tag{6.2}$$

where λ is the wavelength, θ the Bragg angle and $ß_{FWHM(s)}$ the full width at half maximum of the line profile component related to size broadening. ε has no direct physical interpretation but may be used as a measure for crystallite size.

Later the Scherrer constant K has been introduced to relate ε to L_{Vol}.

$$L_{vol} = \frac{K\lambda}{\beta_{FWHM(s)} \cdot \cos\theta} \tag{6.3}$$

Commonly (and indiscriminately!) used K values typically fall in the range of 0.87 - 1, e.g. 0.89 for spherical crystallites, and 0.94 for cubic crystallites. K depends on the how the line profile width is determined, the shape of the crystallites, and the size distribution, and is thus usually not known.

Laue (1926) realized that using the integral breadth (IB) rather than FWHM gives an evaluation that is approximately independent of the distribution in size and shape: K can be assumed to be 1 (IB is defined as the width of a rectangle with the same height and area as the line profile, obtained from dividing the line profile area by the line profile height).

$$L_{vol} = \frac{\lambda}{\beta_{IB} \cdot \cos\theta} \tag{6.4}$$

6.2.2 Microstrain broadening

Microstrain represents displacements of atoms from their ideal positions, produced by any lattice imperfection (dislocations, vacancies, interstitials, substitutionals, and similar defects) as illustrated in Fig. 6.3.

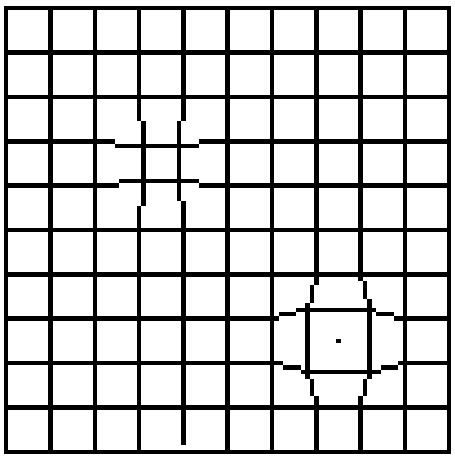


Fig. 6.3: Schematic representation of lattice strain (Haberkorn, 1999).

Microstrain can be conceived by considering two extreme values of the lattice spacing d, namely d + Δd and d - Δd, where ε_0 = Δd/d represents the "mean" deviation (more exactly: 50% probability of the undistorted state)

$$\varepsilon_0 = \frac{\Delta d}{d} = \frac{\beta_{FWHM\,(str)}}{4\tan\theta} \tag{6.5}$$

where $ß_{FWHM(str)}$ represents the full width at half maximum of the line profile component related to microstrain broadening.

6.3 Double-Voigt approach

The Double-Voigt approach (e.g. Balzar, 1999) is supported for modeling of microstructure effects and is also made available in the GUI. Crystallite size and microstrain comprise Lorentzian and Gaussian component convolutions varying in 2θ as a function of $1/\cos(\theta)$ and $\tan(\theta)$ respectively; the formalism used for description of size and microstrain broadening is outlined in sections 6.2.1 and 6.2.2.

The following size parameters are available:

- Cry Size L:
  Lorentzian type component convolution, required to determine the Voigt FWHM and IB parameters for LVol calculation. The Cry Size L parameter has no direct physical interpretation and should not be reported, see equation (6.2).
- Cry Size G:
  Gaussian type component convolution, required to determine the Voigt FWHM and IB parameters for LVol calculation. The Cry Size G parameter has no direct physical interpretation and should not be reported, see equation (6.2).
- LVol-FWHM:
  FWHM based LVol calculation using Lorentzian and Gaussian type component convolutions. The Scherrer constant (K) needs to be defined properly. This parameter has been included mainly to allow comparison to results obtained with traditional methods. As K is usually unknown, LVol-FWHM should not be reported.
- LVol-IB:
  Integral breadth based LVol calculation using Lorentzian and Gaussian type component convolutions (Cry Size L and Cry Size G). This parameter is physically sound and should be reported.

For microstrain the following parameters are available:

- Strain L:
  Lorentzian type component convolution, required to determine the Voigt FWHM parameter for ε_0 calculation. The Strain L parameter has no direct physical interpretation and should not be reported.
- Strain G:
  Gaussian type component convolution, required to determine the Voigt FWHM parameter for ε_0 calculation. The Strain G parameter has no direct physical interpretation and should not be reported.

- ε_0:
  FWHM based microstrain calculation using Lorentzian and Gaussian type component convolutions. This parameter is physically sound and should be reported.

The following sections describe the implementation of the Double-Voigt approach using the TOPAS macro language; it is suggested to inspect the macro definitions in TOPAS.INC.

6.3.1 Preliminary equations

The following preliminary equations are based on the unit area Gaussian, $G_{UA}(x)$, and Lorentzian, $L_{UA}(x)$, and pseudo-Voigt $PV_{UA}(x)$ functions as given in Table 5.1.

- Height of $G_{UA}(x)$ and $L_{UA}(x)$ respectively

  $G_{UA}H = G_{UA}(x=0) = g_1 / fwhm$
  $L_{UA}H = L_{UA}(x=0) = l_1 / fwhm$

- Gaussian and Lorentzian respectively with area A

  $G(x) = A\ G_{UA}(x)$
  $L(x) = A\ L_{UA}(x)$

- Height of G(x) and L(x) respectively

  $GH = A\ G_{UA}H$
  $LH = A\ L_{UA}H$

- Integral breadth of Gaussian and Lorentzian respectively

  $\beta G = A / GH = 1 / G_{UA}H = fwhm / g_1$
  $\beta L = A / LH = 1 / L_{UA}H = fwhm / l_1$

- Unit area Pseudo Voigt, PV_{UA}

  $PV_{UA}H = \eta\ L_{UA}H + (1-\eta)\ G_{UA}H$
  $\beta PV = 1 / PV_{UA}H$

A Voigt is the result of a Gaussian convoluted by a Lorentzian

$$V = G(fwhm_G) \otimes L(fwhm_L)$$

where "$\otimes$" denotes convolution and $fwhm_G$ and $fwhm_L$ are the FWHM of the Gaussian and Lorentzian components.

A Voigt can be approximated using a Pseudo Voigt. This is done numerically where

$$V(x) = G(fwhm_G) \otimes L(fwhm_L) = PV_{UA}(x, fwhm_{PV})$$

By changing units to s ($Å^{-1}$)

$$s = 1/d = 2 \sin(\theta) / \lambda$$

and differentiating and approximating $ds/d\theta = \Delta s / \Delta\theta$ we get

$$\Delta s = (2 \cos(\theta) / \lambda)\ \Delta\theta$$

thus,

fwhm(s) = fwhm(2θ) cos(θ) / λ

IB(s) = IB(2θ) cos(θ) / λ

6.3.2 Size broadening

For crystallite size in TOPAS the Gaussian and Lorentzian component convolutions are:

fwhm(2θ) of Gaussian = (180/π) λ / (cos(θ) CS_G)
fwhm(2θ) of Lorentzian= (180/π) λ / (cos(θ) CS_L)
β(2θ) of Gaussian = (180/π) λ / (cos(θ) CS_G g_1)
β(2θ) of Lorentzian = (180/π) λ / (cos(θ) CS_L l_1)

or, according to Balzar (1999), in terms of s, β_{GS} and β_{CS}

fwhm(s) of Gaussian = (180/π) / CS_G
fwhm(s) of Lorentzian = (180/π) / CS_L
β_{GS}(s) =β(s) of Gaussian = (180/π) / (CS_G g_1)
$\beta_{CS(}$s) =β(s) of Lorentzian = (180/π) / (CS_L l_1)

The macros CS_L and CS_G (see section 13.3.13) are used for calculating the CS_L and CS_G parameters respectively.

Determination of the volume weighted mean column height LVol, LVol-IB and LVol-FWHM is as follows:

LVol-IB = k / Voigt_Integral_Breadth_GL (1/CS_G, 1/CS_L)
LVol-FWHM = k / Voigt_FWHM(1/CS_G, 1/CS_L)

The macro LVol_FWHM_CS_G_L is used for calculating LVol-IB and LVol-FWHM.

6.3.3 Microstrain broadening

Strain_G and Strain_L parameters corresponds to the fwhm(2θ) of a Gaussian and a Lorentzian that is convoluted into the peak, or,

fwhm(2θ) of Gaussian = Strain_G tan(θ)
fwhm(2θ) of Lorentzian= Strain_L tan(θ)
β(2θ) of Gaussian = Strain_G tan(θ) / g_1
β(2θ) of Lorentzian = Strain_L tan(θ) / l_1

or, according to Balzar (1999), in terms of s, β_{CD} and β_{GD}

fwhm(s) of Gaussian = Strain_G sin(θ) / λ = Strain_G s / 2
fwhm(s) of Lorentzian = Strain_L sin(θ) / λ = Strain_L s / 2
β_{GD}(s)/s_0 s = β(s) of Gaussian = (Strain_G / g_1) s / 2
β_{CD}(s)/s_0 s = β(s) of Lorentzian = (Strain_L / l_1) s / 2

The macros Strain_L and Strain_G (see section 13.3.13) are used for calculating the Strain_L and Strain_G parameters respectively.

From these equations we get:

$\beta_{GD}(s)$ = s0 Strain_G / (2 g_1)
$\beta_{CD}(s)$ = s0 Strain_L / (2 l_1)

According to Balzar (1999), equation (34):

e = $\beta_D(2\theta)$ / (4 tan(θ))

where $\beta_D(2\theta)$ is the fwhm of a Voigt comprising a Gaussian with a fwhm = Strain_G Tan(θ) and a Lorentzian with a fwhm = Strain_L Tan(θ).

In TOPAS a value for e0 is given by:

4 e0 Tan(θ) = FWHM of the Voigt from Strain_L and Strain_G
= Voigt_FWHM(Strain_L, Strain_G) Tan(θ)

or,

e0 = Voigt_FWHM(Strain_L, Strain_G) / 4

The macro e0_from_Strain calculates e0 using the equation function Voigt_FWHM_GL.

6.4 WPPM

The WPPM approach for microstructure analysis (Scardi & Leoni, 2001, 2004; Scardi et al. 2010) is supported and can be applied using the following methodology:

1. WPPM using fit objects
2. WPPM using user defined convolutions
3. WPPM using Fourier transforms

The basics of WPPM cannot be discussed here, the interested user is referred to the original literature.

Suggested reading:

- **David, W.I.F., Leoni, M. and Scardi, P. (2010):** *Domain size analysis in the Rietveld method.* - Materials Science Forum, **651**, 187-200.
- **Scardi, P. & Leoni, M. (2001):** *Diffraction line profiles from polydisperse crystalline systems.* - Acta Cryst., **A 57**, 604-613.
- **Scardi, P & Leoni, M. (2004):** *Whole Powder Pattern Modelling: Theory and Applications.* - Diffraction Analysis of the Microstructure of Materials. Editors: Mittemeijer, E.J. & Scardi, P. Springer Series in Materials Science, Vol. 68, 552 pages, ISBN: 978-3-540-40519-1.
- **Scardi, P., Leoni, M. & Delhez, R. (2004):** *Line broadening analysis using integral breadth methods: a critical review.* - J. Appl. Cryst., **37**, 381-390.
- **Scardi, P., Ortolani, M. and Leoni, M. (2010):** *WPPM: Microstructural analysis beyond the Rietveld method.* - Materials Science Forum, **651**, 155-171.

6.4.1 WPPM using fit objects

Fit objects can be used to experiment with user-defined microstructure models using observed data. The following tutorial examples demonstrate the usage of fit_obj for WPPM analysis[1]:

- SPHERE-FIT-OBJ.INP demonstrates fitting of spheres with no distribution
- GAMMA-FIT-OBJ.INP demonstrates fitting of spheres with a gamma distribution
- SUPER-LORENTZIAN.INP is useful for answering the question: Can spheres with a gamma distribution describe a 1/(1+x^2)^m type function?
- COMPARE-1.INP is useful for answering the question: Can a Voigt fit to a particular case of spheres with a gamma distribution?

Fit objects should be only used for quantitative microstructure analysis in cases where instrument broadening is negligible. The example LEBAILBR-GAMMA- FIT-OBJ.INP[2] demonstrates fitting of spheres with a gamma distribution to nanocrystalline CeO_2.

Note that it is not possible to have fit objects as peak types and to apply any convolutions to it.

6.4.2 WPPM using user defined convolutions

User defined convolutions allow the implementation of WPPM operating in 2θ space. The example LEBAILBR-GAMMA-FIT-OBJ.INP[2] demonstrates fitting of spheres with a gamma distribution to nanocrystalline CeO_2.

6.4.3 WPPM using Fourier transforms

WPPM_ft_conv implements WPPM operating in 2θ space.

The following macros (courtesy by Matteo Leoni, University of Trento, Italy), as defined in TOPAS.INC, allow fitting of spheres, cubes, and octahedra using a log normal distribution:

```
WPPM_Sphere_Ln_Normal
WPPM_Cube_Ln_Normal
WPPM_Octahedron_Ln_Normal
```

Lattice parameters appearing within the macros are made constant using Constant; these convolutions are therefore made independent of lattice parameter changes and hence separate convolutions are not initiated whilst calculating lattice parameter derivatives.

The example LEBAILBR-PAWLEY-WPPM.INP demonstrates fitting of spheres with a normal distribution to nanocrystalline CeO_2[2].

1 \TUTORIAL\MISCELLANEOUS\SIZE-STRAIN\WPPM\FIT-OBJ\
2 \TUTORIAL\MISCELLANEOUS\SIZE-STRAIN\WPPM\CEO2\

The example LN-NORMAL-1.INP[1] can be used to visualise log normal distributions and to calculate the mean, mode and median values.

6.5 Interpretation of size values

The interpretation of size values is difficult and so is the comparison of results obtained from different methods, as different methods of analysis may report different quantities. These approaches can be broadly divided into two groups:

1. Model independent approaches
   Mainly identified with the Fourier-deconvolution correction for instrumental broadening (Stokes method), which is followed by the Warren-Averbach method for the separation of crystallite size and strain effects.
2. Model dependent approaches
   Mainly defined by profile fitting methods modelling instrument and specimen contributions as discussed in section 6.1).

Results obtained from model dependent and independent approaches are usually not comparable due to different parameter definitions (e.g. Klug & Alexander, 1974; Balzar, 1999), being an underlying source of disagreement between the results obtained from both methods:

- Warren-Averbach analysis usually gives area weighted column heights
- Profile fitting methods usually give volume weighted column heights

The disagreement between volume weighted and area weighted column heights as determined by both methods is demonstrated in Fig. 6.4, where a typical column height distribution is shown. The differences between volume weighted and area weighted column heights as determined by the different methods can be highly significant.

Things become even more complicated when keeping in mind, that any direct imaging techniques for crystallite size analysis (e.g. SEM, TEM, AFM) provide number weighted column heights!

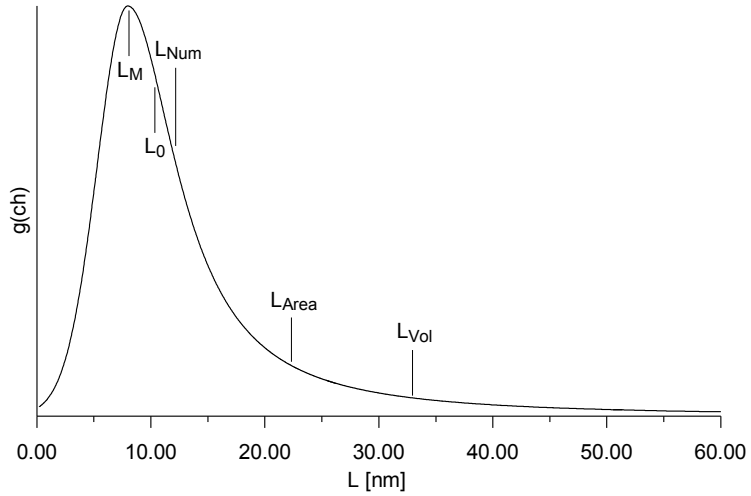


Fig. 6.4: Schematic representation of a column height distribution g(ch) showing the following characteristic points:
L_M : mode
L_0 : median
L_{Num} : number weighted mean
L_{Area} : area weighted mean
L_{Vol} : volume weighted mean

1 \TUTORIAL\MISCELLANEOUS\SIZE-STRAIN\WPPM\

The determination of crystallite size always requires the knowledge of the true mean crystallite shape (section 6.2.1). Even if several laboratories measure the same sample using the same evaluation method significantly different results can be obtained, if there is no agreement concerning the mean crystallite shape.

According to Balzar (2004), L_{Vol} (or D_V in the Balzar paper) can be related to the real dimensions (average radius) of the crystallites by the expression

$D_V=1.5*R(1+c)^3$

where R and c are the log normal distribution parameters (mean and standard deviation, respectively).

Obviously any report of microstructural results always requires information about the evaluation method employed.

7 INDEXING

7.1 LSI-Index

LSI-Index uses an indexing algorithm based on iterative use of least squares, refer to Coelho (2003).

7.1.1 Operation in Launch Mode

Indexing requires the setting up of an INP file with the relevant information. An indexing example is as follows:

```
index_zero_error
try_space_groups "2 75"
load index_d {
   8.912    good
   7.126
   4.296
   ...
}
```

Individual space groups can be tried as in this example or for simplicity all of the Bravais lattices can be tried by placing them in the INP file as follows:

```
index_zero_error
Bravais_Cubic_sgs
Bravais_Trigonal_Hexagonal_sgs
Bravais_Tetragonal_sgs
Bravais_Orthorhombic_sgs
Bravais_Monoclinic_sgs
Bravais_Triclinic_sgs
load index_d {
   8.912    good
   7.126
   4.296
   ...
}
```

7.1.2 NDX output files

On termination of indexing a *.NDX file is created with a name corresponding to the name of the INP file and placed in the same directory as the INP file. The *.NDX file contains solutions found as well as a detailed summary of the best 20 solutions. Here are two example output lines from an NDX file:

```
0) P42/nmc    3    0    1187.124    38.82    0.0000
   11.1904    11.1904    9.4799    90.000    90.000    90.000 ' === 24 19

1) P-421c     3    0    1187.124    35.67    0.0000
   11.1904    11.1904    9.4799    90.000    90.000    90.000 ' === 24 19
```

- The 1st column corresponds to the rank of the solution
- The 2nd corresponds to the space group
- The 3rd corresponds to the status of the solution:
  - Status 1: Weighting applied as defined in Coelho (2003)
  - Status 2: Zero error attempt applied
  - Status 3: Zero error attempt successful and impurity line(s) removal attempt successful
  - Status 4: Impurity line(s) removed
- The 4th column corresponds to the number of un-indexed lines
- The 5th column corresponds to the volume of the lattice
- The 6th corresponds to the figure of merit value, see section 10.1.4
- The 7th corresponds to the zero error if index_zero_error is included
- Columns 8 to 13 contain the lattice parameters

The last 2 columns contain the number of non-zero h^2+k^2+hk and l^2 values used in the indexed lines. These represent the hkl coefficient for X0 and X1 respectively for Trigonal/Hexagonal systems (an overview about all crystal systems is provided in Table 7.1). When one of these numbers is zero then the corresponding lattice parameters is not represented and the number is therefore displayed as the negative number of -999. This facility is particularly useful for identifying dominant zones. For example, if the smallest lattice parameter is 3Å and the smallest d-spacing is 4Å then it is impossible to determine the small lattice parameter. In these cases values of -999 will be obtained.

Table 7.1: hkl coefficients corresponding to the Xnn reciprocal lattice parameters for the seven crystal systems.

	X0	X1	X2	X3	X4	X5
Cubic	$h^2+k^2+l^2$					
Hexagonal	h^2+k^2+hk	l^2				
Trigonal	h^2+k^2+hk	l^2				
Tetragonal	h^2+k^2	l^2				
Orthorhombic	h^2	k^2	l^2			
Monoclinic	h^2	k^2	l^2	hl		
Triclinic	h^2	k^2	l^2	hk	hl	kl

7.1.3 Reprocessing solutions

Details of solutions can be obtained at a later stage by including solution lines found in the NDX file into the INP file. For example, supposing details of solutions 50 and 51 were sought then the following can be used:

```
index_lam 1.540596
index_zero_error
try_space_groups 2
Indexing_Solutions_With_Zero_Error {
   50) P-1    0  3203.030  8.42  0.0007  11.9955  17.2846  16.2351
       101.967  82.781  102.534 ' ===  21  21  19  19  16  16
   51) P-1  -10  2023.440  8.34  0.0102   7.2783   9.0246  33.7293
        99.192  75.243   76.294 ' ===  10  13  24   3  10  12
}
load index_d {
   15.83    good
    8.75
    7.91
   ...
   }
```

On termination of indexing a *.DET file containing details of the supplied solutions is created with a name corresponding to the name of the INP file and placed in the same directory as the INP file.

7.1.4 Figure of Merit

The deWolff figure of merit MN (deWolff, 1968) is used as a figure of merit. It is further scaled by $1/(N_{impurity} + 1)$ where $N_{impurity}$ corresponds to the number of unindexed lines.

7.1.5 Unique space group hkls in powder diffraction

Table 7.2 lists unique space group hkls in powder diffraction.

Table 7.2: Unique space group hkls in powder diffraction.

Space group numbers with identical hkls	Space group symbols with identical hkls
Triclinic	
1 2	P1 P-1
Monoclinic	
9 15	Cc C2/c
5 8 12	C2 Cm C2/m
14	P21/c
7 13	Pc P2/c
4 11	P21 P21/m
3 6 10	P2 Pm P2/m
Orthorhombic	
70	Fddd
43	Fdd2
22 42 69	F222 Fmm2 Fmmm
68	Ccca
73	Ibca
37 66	Ccc2 Cccm
45 72	Iba2 Ibam
41 64	Aba2 Cmca
46 74	Ima2 Imma
36 40 63	Cmc21 Ama2 Cmcm
39 67	Abm2 Cmma
20	C2221
23 24 44 71	I222 I212121 Imm2 Immm
21 35 38 65	C222 Cmm2 Amm2 Cmmm
52	Pnna
56	Pccn
60	Pbcn
61	Pbca
48	Pnnn
54	Pcca
50	Pban
33 62	Pna21 Pnma
34 58	Pnn2 Pnnm
32 55	Pba2 Pbam
30 53	Pnc2 Pmna
29 57	Pca21 Pbcm
27 49	Pcc2 Pccm

31 59	Pmn21 Pmmn
26 28 51	Pmc21 Pma2 Pmma
19	P212121
18	P21212
17	P2221
16 25 47	P222 Pmm2 Pmmm
Tetragonal	
142	I41/acd
110	I41cd
141	I41/amd
109 122	I41md I-42d
108 120 140	I4cm I-4c2 I4/mcm
88	I41/a
80 98	I41 I4122
79 82 87 97 107 119 121 139	I4 I-4 I4/m I422 I4mm I-4m2 I-42m I4/mmm
130	P4/ncc
126	P4/nnc
133	P42/nbc
103 124	P4cc P 4/mcc
104 128	P4nc P 4/mnc
106 135	P42bc P 42/mbc
137	P42/nmc
138	P42/ncm
134	P42/nnm
125	P4/nbm
114	P-421c
105 112 131	P42mc P-42c P42/mmc
102 118 136	P42nm P-4n2 P42/mnm
101 116 132	P42cm P-4c2 P42/mcm
100 117 127	P4bm P-4b2 P4/mbm
86	P42/n
85 129	P4/n P4/nmm
92 96	P41212 P43212
94	P42212
76 78 91 95	P41 P43 P4122 P4322
77 84 93	P42 P 42/m P4222
90 113	P4212 P-421m
75 81 83 89 99 111 115 123	P4 P-4 P4/m P422 P4mm P-42m P-4m2 P4/mmm

Trigonal & Hexagonal	
161 167	R3c R-3c
146 148 155 160 166	R3 R-3 R32 R3m R-3m
184 192	P6cc P6/mcc
159 163 186 190 194	P31c P-31c P63mc P-62c P63/mmc
158 165 185 188 193	P3c1 P-3c1 P63cm P-6c2 P63/mcm
169 170 178 179	P61 P65 P6122 P6522
144 145 151 152 153 154 171 172 180 181	P31 P32 P3112 P3121 P3212 P3221 P62 P64 P6222 P6422
173 176 182	P63 P63/m P6322
143 147 149 150 156 157 162 164 168 174 175 177 183 187 189 191	P3 P-3 P312 P321 P3m1 P31m P-31m P-3m1 P6 P-6 P6/m P622 P6mm P-6m2 P-62m P6/mmm
Cubic	
228	Fd-3c
219 226	F-43c Fm-3c
203 227	Fd-3 Fd-3m
210	F4132
196 202 209 216 225	F23 Fm-3 F432 F-43m Fm-3m
230	Ia-3d
220	I-43d
206	Ia-3
214	I4132
197 199 204 211 217 229	I23 I213 Im-3 I432 I-43m Im-3m
222	Pn-3n
218 223	P-43n Pm-3n
201 224	Pn-3 Pn-3m
205	Pa-3
212 213	P4332 P4132
198 208	P213 P4232
195 200 207 215 221	P23 Pm-3 P432 P-43m Pm-3m

7.2 LP-Search

LP-Search represents a new indexing algorithm based on Pawley / Le Bail fitting (section 8) using random lattice parameters (Coelho & Kern, 2005). LP-Search minimizes on a new figure of merit function that gives a measure of correctness for a particular set of lattice parameters. More specifically the figure of merit function assigns parts of the diffraction pattern to peaks and then sums the absolute values of the products of the diffraction intensities multiplied by the distance to the peaks:

$$\mathrm{FOM} = \sum_{\mathrm{j}} \sum_{\mathrm{i}} \mathrm{I}(2\theta_i) \left| 2\theta_i - 2\theta_{0,\mathrm{j}} \right|$$

where the summation in "j" is over the calculated Bragg positions $2\theta_0$ and the summation in "i" is over part of the diffraction pattern such that

$(2\theta_{0,\,j-1} + 2\theta_{0,\,j})/2 < 2\theta_i < (2\theta_{0,\,j+1} + 2\theta_{0,\,j})/2$

The method is independent of d-spacing extraction and line profile shape and therefore suited for indexing of poor quality powder data

7.2.1 The lp_search keyword

lp_search invokes LP-Search indexing by searching for the correct lattice parameters. The value for E! increases or decreases the search space.

Whilst LP-Search is running a log file (LP.LOG) containing the best lattice parameters is kept updated in the main TOPAS directory. The files is always appended to and not over written. The log file contains volume, lattice parameters and R_{WP} values.

7.2.2 Operation in Launch Mode

Indexing in Launch mode requires the setting up of an INP file with the relevant information. An indexing example is as follows:

```
iters 1000
continue_after_convergence

   hkl_Is
      lp_search 1
      space_group 16
      a @ 10 min =3; max =15;
      b @ 10 min =3; max =15;
      c @ 10 min =3; max =15;
      MVW( 0, 200 min =100; max =400;, 0)
```

8 PAWLEY AND LE BAIL REFINEMENTS

In whole powder pattern decomposition peak intensities are either refined (Pawley, 1981) or adjusted in an iterative procedure (Le Bail, 1988). The Le Bail method uses a two-step cyclic process, where peak intensities are not part of the least-squares refinement, but are determined independently using the Rietveld intensity partitioning formula (for details see e.g. David & Sivia, 2002). As a consequence, intensities determined using the Le Bail method are not affected by correlation problems, but convergence is usually slow.

For Pawley and Le Bail refinements the following input segments can be used:

```
hkl_Is
   space_group p-1
```

```
hkl_Is
   lebail 1
   space_group p-1
```

The default is Pawley refinement, the keyword *lebail* triggers a Le Bail refinement.

hkls are generated if there are no *hkl_m_d_th2* and *I* keywords defined. Intensity parameters are given an initial starting value of 1. If the *lebail* keyword is not defined then the intensity parameters are given the unique code of @. After refinement, the details for the generated hkl's are appended after the *space_group* keyword. For the Pawley method, once the hkl details are generated, parameter equations can be applied to the *I* parameters as usual.

For example, the following input segment:

```
xdd quartz.xdd
      ...
      hkl_Is
         Hexagonal(4.91459, 5.40603)
         space_group P_31_2_1
```

will generate the following OUT file:

```
xdd quartz.xdd
      ...
      hkl_Is
         Hexagonal(4.91459, 5.40603)
         space_group P_31_2_1
         load hkl_m_d_th2 I
         {
          1   0   0   6    4.25635   20.85324 @ 3147.83321
          1   0   1   6    3.34470   26.62997 @ 8559.23955
          1   0  -1   6    3.34470   26.62997 @ 8559.23955
          ...
         }
```

In GUI mode, the "Delete hkls on Refinement" option can be used to replace the current hkls by new hkls calculated for the actual spacegroup and data range, everytime a new refinement is started (refined intensities will be lost). This option should be used, if the spacegroup or the data range has been changed.

Traditionally, Pawley and Le Bail refinements are used for lattice parameter refinements and intensity extraction for structure determination and refinement. In general:

- Pawley refinements are characterized by better and faster convergence. The A matrix can be saved as a file and can be taken advantage of in subsequent structure determination using Charge Flipping (section 9.11.2).
- Le Bail refinements deliver intensities that are not affected by parameter correlation

In TOPAS, the functionality of Pawley and Le Bail refinements has been massively extended towards new applications.

- The LP-Search indexing method is based on Pawley / Le Bail refinements using random lattice parameters and is described in section 7.2.
- The PONKCS method (Scarlett & Madsen, 2006) provides for quantitative analysis of phases with Partially Or No Known Crystal Structures, and represents a highly significant extension to the Rietveld method for quantitative phase analysis. For details refer to section 10.2.4.

9 STRUCTURE ANALYSIS

9.1 Calculation of structure factors

The structure factor F for a particular reflection (h k l) is the complex quantity:

$$F = \Sigma_s (A_S + i B_S) \Sigma_a (f_{o,a} + f_a' + i f_a'') O_a \quad (9.1)$$

The summation Σ_s is over the sites of the unit cell and the summation Σ_a is over the atoms residing on site s. O_a and $f_{o,a}$ correspond to the site occupancy and the atomic scattering factor for atom 'a' respectively. f_a' and f_a'' are the anomalous dispersion coefficients for atom 'a'. A_S and B_S corresponds to the cosine and sine summations for site 's', or:

$$A_S = \Sigma_e T_{s,e} \cos(2\pi h re), \quad BS = \Sigma_e T_{s,e} \sin(2\pi h re) \quad (9.2)$$

where $T_{s,e}$ is the temperature factor and the summation Σe is over the equivalent positions of site 's' as dictated by the space group. Defining:

$$f_{o,s} = \Sigma_a f_{o,a} O_a, \quad f_s' = \Sigma_a f_a' O_a, \quad f_s'' = \Sigma_a f_a'' O_a \quad (9.3)$$

and separating the real and imaginary components gives:

$$F = \Sigma_s (A_s + i B_s) (f_{o,s} + f_s' + i f_s'') \quad (9.4)$$

$$F = \Sigma_s (A_s (f_{o,s} + f_s') - B_s f_s'') + i \Sigma_s (A_s f_s'' + B_s (f_{o,s} + f_s'))$$

or, $F = A + i B$

The observed intensity is proportional to the complex conjugate of the structure factor, or,

$$F^2 = A^2 + B^2 \quad (9.5)$$

or,

$$F^2 = A_{01}^2 + B_{01}^2 + A_{11}^2 + B_{11}^2 + 2 B_{01} A_{11} - 2 A_{01} B_{11}$$

where

$$A_{01} = \Sigma_s A_s (f_{o,s} + f_s'), \quad A_{11} = \Sigma_s A_s f_s'', \quad B_{01} = \Sigma_s B_s (f_{o,s} + f_s'), \quad B_{11} = \Sigma_s B_s f_s''$$

and

$$A = A_{01} - B_{11}, \quad B = B_{01} + A_{11}$$

Atomic scattering factors ($f_{o,a}$) used, are by default those from:
http://www.esrf.fr/computing/expg/subgroups/theory/DABAX/dabax.html;
these comprise 11 values per atom and are found in the file ATMSCAT_11.CPP. Correspondingly 9 values per atom, obtained from the International Tables, are found in the file ATMSCAT_9.CPP. Use of either 9 or 11 values can be invoked by running the batch files use_9f0 and use_11f0, respectively.

Dispersion coefficients (f_a' and f_a'') used are found in the \SSF directory. For elements with $Z \leq 92$ they are by default from
http://www.cxro.lbl.gov/optical_constants/asf.html,
These data cover the energy range from 10 to 30000 eV. The use of *use_tube_dispersion_coefficients* forces the use of dispersion coefficients from the International Tables for X-ray Crystallography (1995), Vol. C, 384-391 and 500-502, and for O^{2-} from Hovestreydt (1983). These data are in discrete energy steps corresponding to wavelengths typically found in laboratory X-ray sources.

For neutron diffraction data $f_a' = f_a'' = 0$ and $f_{o,a}$ is replaced by the bound coherent scattering length for atom "a" obtained from:
http://www.ccp14.ac.uk/ccp/web-mirrors/neutrons/n-scatter/n-lengths/LIST~1.HTM;
these data are found in the NEUTSCAT.CPP file.

9.1.1 Friedel pairs

For centrosymmetric structures the intensities for a Friedel reflection pair are equivalent, or, $F^2(h\ k\ l) = F^2(-h-k-l)$. This holds true regardless of the presence of anomalous scattering and regardless of the atomic species present in the unit cell. This equivalence in F^2 is due to the fact that $B_{01} = B_{11} = 0$ and thus:

$$F = A_{01} + i\,A_{11} \quad \text{and} \quad F^2 = A_{01}^2 + A_{11}^2 \tag{9.6}$$

For non-centrosymmetric structures and for the case of no anomalous scattering, or for the case where the unit cell comprises a single atomic species, then $F^2(h\ k\ l) = F^2(-h-k-l)$. Or, for a single atomic species we have:

$$B_{01}\,A_{11} = (f0 + f')\,(\Sigma_S\,B_S)\,f''\,(\Sigma_S\,A_S), \quad A_{01}\,B_{11} = (f0 + f')\,(\Sigma_S\,A_S)\,f''\,(\Sigma_S\,B_S) \tag{9.7}$$

or,

$$B_{01}\,A_{11} = A_{01}\,B_{11}$$

and thus from cancellation in equation (9.5) we get

$$F^2(h\ k\ l) = F^2(-h-k-l) = A_{01}^2 + B_{01}^2 + A_{11}^2 + B_{11}^2 \tag{9.8}$$

For non-centrosymmetric structures and for the case of anomalous scattering and for a structure comprising more than one atomic species then $F^2(h\ k\ l) \neq F^2(-h-k-l)$.

9.1.2 Calculation of structure factors – powder data

Friedel pairs are merged for powder diffraction data meaning that the multiplicities as determined by the hkl generator includes the reflections (h k l) and (-h –k –l); this merging of Friedel pairs improves computational efficiency. Equation (9.5) gives the correct intensity for unmerged Friedel pairs and thus it cannot be used for merged Friedel pairs. Using the fact that:

$$A_{01}(h\ k\ l) = A_{01}(-h\ -k\ -l), \quad A_{11}(h\ k\ l) = A_{11}(-h\ -k\ -l)$$

$$B_{01}(h\ k\ l) = B_{01}(-h\ -k\ -l), \quad B_{11}(h\ k\ l) = B_{11}(-h\ -k\ -l) \tag{9.9}$$

then F^2 from equation (9.5) in terms of $B_{01}(h\ k\ l)$ and $B_{11}(h\ k\ l)$ evaluates to:

$F^2(h\ k\ l) = Q_1 + Q_2$ (9.10)

$F^2(-h\ -k\ -l) = Q_1 - Q_2$

where $Q_1 = A_{01}^2 + B_{01}^2 + A_{11}^2 + B_{11}^2$

and $Q_2 = 2(B_{01}\ A_{11} - 2\ A_{01}\ B_{11})$

and for merged Friedel pairs we get:

$F^2(h\ k\ l) + F^2(-h\ -k\ -l) = 2\ Q_1$ (9.11)

The factor 2 in equation (9.11) is dropped due to the fact that the multiplicity as given by the hkl generator includes this factor. Thus the final equation describing F^2 for powder diffraction data for merged Friedel pairs is given by:

$F^2(h\ k\ l)_{merged} = Q_1$ (9.12)

The reserved parameter names of A01, A11, B01 and B11 can be used to obtain unmerged real, imaginary and F^2 components and the merged F^2. The following macros have been provided in TOPAS.INC (see also section 13.3.10):

- macro F_Real_positive { (A01-B11) }
- macro F_Real_negative { (A01+B11) }
- macro F_Imaginary_positive { (A11+B01) }
- macro F_Imaginary_negative { (A11-B01) }
- macro F2_positive { (F_Real_positive^2 + F_Imaginary_positive^2) }
- macro F2_negative { (F_Real_negative ^2 + F_Imaginary_negative^2) }
- macro F2_Merged { (A01^2 + B01^2 + A11^2 + B11^2) }

Note that F2_Merged = (F2_positive + F2_negative) / 2

The reserved parameters I_no_scale_pks and I_after_scale_pks for *str* phases are equivalent to the following:

- I_no_scale_pks = Get(*scale*) M F2_Merged
- I_after_scale_pks = Get(*all_scale_pks*) Get(*scale*) M F2_Merged

In addition the macros Out_F2_Details and Out_A01_A11_B01_B11 can be used to output F^2 details.

9.1.3 Calculation of structure factors – single crystal data

SHELX HKL4 single crystal data comprise unmerged equivalent reflections and thus equation (9.5) is used for calculating F^2. Equivalent reflections are merged by default and can be unmerged using the *dont_merge_equivalent_reflections* keyword. For centrosymmetric structures, merging includes the merging of Friedel pairs and thus equation (9.12) is used for calculating F^2. For non-centrosymmetric structures, merging excludes the merging of Friedel pairs and thus (9.5) is used for calculating F_2. The keyword *dont_merge_Friedel_pairs* prevents the merging of Friedel pairs. The *ignore_differences_in_Friedel_pairs* keyword forces the use of equation (9.12)

for calculating F^2. The reserved parameter name M_{obs} returns the number of observed reflections belonging to a particular family of reflections.

Merging of equivalent reflections reduces the computational effort and is useful in the initial stages of structure refinement. Only a single intensity is calculated for a set of equivalent reflections even in the absence of merging. Thus equivalent reflections and Friedel pairs are remembered and intensities appropriated as required.

*.SCR data are typically generated from a powder pattern and comprise merged equivalent reflections including merged Friedel pair. As a consequence, any definitions of *dont_merge_equivalent_reflections*, *dont_merge_Friedel_pairs* and *ignore_differences_in_Friedel_pairs* are ignored, and equation (9.12) is used for calculating F^2.

9.1.4 The Flack parameter

For single crystal data and for non-centrosymmetric structures the Flack parameter (Flack, 1983) as implemented scales $F^2(h)$ and $F^2(-h)$:

$$F^2(h\ k\ l) = Q_1 + (1 - 2\ \text{Flack})\ Q_2 \qquad (9.13)$$

$$F^2(-h\ -k\ -l) = Q_1 - (1 - 2\ \text{Flack})\ Q_2$$

9.1.5 Single Crystal Output

The macro Out_Single_Crystal_Details outputs details for single crystal refinements. Mobs corresponds to the number of observed reflections belonging to a particular family of planes. When Friedel Pairs are not merged then there will be a different Mobs for h and –h. Phase symmetry is considered in the values for A01, B01, A11 and B11.

9.2 Space groups, symmetry operators and user-defined rotational matrices

The keyword *space_group* $symbol is used to define the space group, where $symbol can be any space group symbol occurring in the file SGCOM5.CPP (case insensitive), it can also be a space group number; here are some examples:

```
space_group "I a -3"
space_group ia-3
space_group P_63_M_C
space_group I_41/A_M_D
space_group I_41/A_M_D:2      ' defines second setting of I_41/A_M_D
space_group 206
space_group 222:2             ' defines second setting of 222
```

Symmetry operators are generated by SGCOM6.EXE and placed into a \sg*.sg file with a name similar to the name of the space group. Space group names containing

the characters "/" or ":" are placed in files with names similar to the space group but with the characters replaced by "o" and "q" respectively. The reason for this is that file names containing these characters are not allowed on some operating systems. hkl generation uses information in the *.sg file.

User-defined rotational matrices can be added to the file sgrots3.cpp found in the main TA directory.

9.3 Site occupancies

Only unique positions are generated from symmetry operators. Fully occupied sites therefore require site occupancy values of 1.
A comparison of atomic positions is performed in the generation of the unique positions with a tolerance in fractional coordinates of 10^{-15}. When entering a fractional coordinate for a special position, such as 1/3, 1/6, etc., it is mandatory to enter a fraction in the form of an equation such as

```
x = 1/3; y = 1/3; z = 2/3;
```

instead of

```
x 0.33333 y 0.33333 z 0.66666
```

Not adhering to this convention may lead to severely wrong refinement results!

Note: The *Fit Window* drops a warning, if a coordinate is closer than 0.001Å to particular special positions. Increasing structure size there is an increasing probability that a general position may be close to a special position by chance; nevertheless a cross-check is always strongly recommended.

9.4 Site identifying strings

Keywords such as *operate_on_points* use a site identifying string; this string can contain the wild card character "*" and a negation character "!". The wild card character "*" used in "O*" means that sites with names starting with "O" are considered. In addition to using the wild card character, the site names can be explicitly written within double quotation marks. For example, consider the following segment:

```
str
   site Pb1...
   site S1 ...
   site O1 ...
   site O2 ...
   site O31 ...
   site O32 ...
   site O4 ...
```

Table 9.1 shows some *operate_on_points* strings and the corresponding sites identified for this particular example.

Table 9.1: Example *operate_on_points* strings and the corresponding sites identified.

***operate_on_points* $sites:**	**Sites identified:**
*	Pb1, S1, O1, O2, O31, O32, O4
Pb*	Pb1
"Pb1 S*"	Pb1, S1
O*	O1, O2, O31, O32, O4
"O* !O3*"	O1, O2, O4
"O* !O1 !O2"	O31, O32, O4

9.5 User defined rotational matrices

User defined rotational matrices for the space group generator can be added to the file sgrots3.cpp found in the TOPAS installation directory.

9.6 Atomic data files

Table 9.2 lists the sources used for atomic data. The files are found in the TOPAS installation directory. The references refer to the source of the data. In many cases the format of the data file corresponds to the original source format.

Table 9.2: Files and associated sources for atomic data.

File	Description
anomdisp.cpp	f' and f'' for Laboratory X-ray tubes. File is read if there are no associated SSF*.NFF file or if *use_tube_dispersion_coefficients* is defined.
atmscat.cpp	f0 or elastic photon-atom scattering, relativistic form factors; data from http://www.esrf.fr/computing/expg/subgroups/theory/DABAX/dabax.html
atom_colors.def	Red, Green, Blue (RGB) CPK atom colors from http://www.bio.cmu.edu/Courses/BiochemMols/Periodic/ElemList.htm. Used for assigning colors to atoms when displaying in OpenGL.
atom_radius.def	Atomic and covalent radii from http://www.esrf.fr/cgi-bin/periodic
isotopes.txt	Atomic Weights and Isotopic Compositions for all elements from http://physics.nist.gov/PhysRefData/Compositions/
magdata.dat	Data from GSAS data file via the International tables. Data correction for V entry made by Robert Von Dreele.
neutscat.cpp	Neutron scattering lengths from http://www.ccp14.ac.uk/ccp/web-mirrors/neutrons/n-scatter/n-lengths/LIST~1.HTM
no_polyhedra.def	Disables drawing of polyhedral for atoms listed
SSF*.NFF	Anomalous scattering factors f' and f'' for a range of wavelengths from http://www cxro.lbl.gov/optical_constants/asf.html The present data is in three columns "E(eV),f1,f2" where f'=f1–Z and f''= f2 and the conversion from wavelength to energy scale is E(eV)=10^5/ (8.065541*Lambda(Ang)).
MAC\Znn.html	X-Ray Mass Attenuation Coefficients from http://www.nist.gov/pml/data/xraycoef/index.cfm

9.7 Isotopes and Atom Names

Isotopes.txt is used for obtaining isotope weights. It's possible to have the following when refining either neutron (i.e. if the keyword *neutron_data* is defined) or X-ray data and to obtain the correct results without changing the INP str:

```
site ... occ Mg ...
site ... occ Mg+2 ...
site ... occ 24Mg ...
site ... occ 26Mg ...
site ... occ 26Mg+2 ...
```

In the cases of "Mg" and "Mg+2" the atomic weight used is the standard weight as defined in isotopes.txt.

In the cases of "26Mg" and "26Mg+2" the atomic weight used is the isotope weight as defined in isotopes.txt. Note the "+2" is dropped when searching that file.

The atomic weight for 24Mg is not the same as that for Mg. When 24Mg is used then the isotope weight for 24Mg is used. When Mg is defined then the standard weight is used. The standard weight corresponds to the mean weight of the naturally occurring Mg isotopes.

In the case of X-rays:

- Atomic scattering factors used (from file atmscat.cpp) for 26Mg and 26Mg+2 corresponds to those of Mg or Mg+2 respectively. The numbers occurring at the start of the symbol is dropped when searching atmscat.cpp.
- f' and f" corrections (files in ssf directory) corresponds to that for Mg. In other words the numbers occurring at the start of the symbol as well as the charge (i.e. "+2" in this case) is dropped.

In the case of neutrons:

- Scattering lengths used are from the neutscat.cpp file; the charge (i.e. "+2") is dropped when searching neutscat.cpp.
- Internally the program converts "D" and "T" to "2H" and "3H", respectively.

9.8 Refining on X-ray and neutron form factors

Default values for X-ray form factors (f0, f', f") and neutron form factors (scattering lengths) are being used as outlined in section 9.6).

X-ray and neutron form factors can be defined or refined independently using the *f0_f1_f11_atom* keyword and its parameters *f0*, *f1*, and *f11.* The *f0* parameter represents either the scattering factor f0 in the X-ray case, or scattering lengths in the neutron case. The *f1* and *f11* parameters apply to the f' and f" anomalous scattering factors, respectively.

The keyword *no_f11* instructs the program to ignore f11. This increases speed with little change in Ycalc.

The following keyword sequence defines f' and f" for e.g. Pb+2:

```
load f0_f1_f11_atom f1 f11 {
      Pb+2        -3.7275704        8.93529065
      }
```

f0, f', f" for e.g. Pb+2 can be refined as follows:

```
prm a1 25 min -50 max 50
load f0_f1_f11_atom f0 f1 f11 {
      Pb+2
      = a1         Exp(1.058874   (-0.25) / D_spacing^2) +
        16.496822  Exp(0.106305   (-0.25) / D_spacing^2) +
        19.984501  Exp(6.708123   (-0.25) / D_spacing^2) +
        6.813923   Exp(24.395554  (-0.25) / D_spacing^2) +
        5.233910   Exp(1.058874   (-0.25) / D_spacing^2) +
        4.065623;  ' this is f0 for Pb+2
      @ -3.7275704 ' this is f1 for Pb
      @ 8.93529065 ' this is f11 for Pb
}
report_on_str
```

The *f0* parameter can be a function of the reserved parameter D_spacing. The keyword *report_on_str* reports on f1 and f11 or neutron scattering lengths used. No values are reported when the keyword *d_spacing_to_energy_in_eV_for_f1_f11* is used.

To disable the effects of *f0*, *f1* and *f11*, for say CeO_2, then the following could be used:

```
load f0_f1_f11_atom f0 f1 f11
   {
   Ce+4 1 0 0
   O-2  1 0 0
   }
```

Scattering lengths of e.g. CeO_2 can be refined as follows:

```
load f0_f1_f11_atom f0 {
      Ce @  0.1 min -5 max 5
      O  @  0.1 min -5 max 5
}
report_on_str
```

9.9 Geometric constraints, restraints and penalties

Rigid bodies as well as anti-bumping and GRS penalties have been shown to particularily facilitate structure determination and refinement, and are discussed below.

9.9.1 Rigid bodies

TOPAS supports so-called "rigid bodies" based on Cartesian, fractional, and internal coordinates (z-matrix representation) with all parameters refinable (bond lengths, bond angles, torsion angles, translation, and rotation). Additionally, every rigid body parameter can be defined in the form of a restraint or a penalty (see also section 4.6). As a consequence, in the present TOPAS implementation, "rigid bodies" should be rather termed "soft bodies", but the term "rigid body" will be kept throughout this document as the currently established notation in literature.

9.9.1.1 Basic concepts

Rigid bodies comprise points in space defined using either the *z_matrix* or *point_for_site* keywords or both simultaneously. All or some of these points can then be operated on using the rotate and translate keywords.

Successful use of rigid bodies embodies

- Translating a rigid body or part of a rigid body.
- Rotating a rigid body or part of a rigid body around a point.
- Rotating a rigid body or part of a rigid body around a line.
- *ua*, *ub*, and *uc* of the *point_for_site* keyword, *ta*, *tb* and *tc* of the *translate* keyword, *qa*, *qb* and *qc* of the *rotate* keyword and the parameters of the z_*matrix* keyword are all refineable parameters. This means that parameter attributes such as *min*/*max* can be defined.

The directory RIGID contains many rigid body examples in *.RGD files. These files can be viewed and modified using the Rigid Body Editor of the GUI.

9.9.1.2 Fractional, Cartesian and Z-matrix coordinates

The most basic means of setting up a rigid body is by means of fractional or Cartesian coordinates. A Benzene ring for example without Hydrogens can be formulated as follows:

```
prm a 1.3 min 1.2 max 1.4
rigid
   point_for_site C1 ux =  a Sqrt(3) .5; uy =  a .5;
   point_for_site C2 ux =  a Sqrt(3) .5; uy = -a .5;
   point_for_site C3 ux = -a Sqrt(3) .5; uy =  a .5;
   point_for_site C4 ux = -a Sqrt(3) .5; uy = -a .5;
   point_for_site C5 uy =  a;
   point_for_site C6 uy = -a;

   ' rotate all previously defined points:

   Rotate_about_axies(@ 0, @ 0, @ 0)

   ' translate all previously defined points:

   Translate(@ .1, @ .2, @ .3)
```

The last two statements rotates and translates the rigid body as a whole and their inclusion are implied if absent in the following examples.

A formulation of any complexity can be obtained from a) databases of existing structures by simply using fractional or Cartesian coordinates of structure fragments or b) from sketch programs for drawing chemical structures.

A Z-matrix representation of a rigid body explicitly defines the rigid body in terms of bond lengths and angles. A Benzene ring can be formulated using two dummy atoms X1 and X2 as follows:

```
str...
   site X1... occ C 0
   site X2... occ C 0
   rigid
      load z_matrix {
         X1
         X2   X1  1.0
         C1   X2  1.3   X1  90
         C2   X2  1.3   X1  90  C1  60.0
         C3   X2  1.3   X1  90  C2  60.0
         C4   X2  1.3   X1  90  C3  60.0
         C5   X2  1.3   X1  90  C4  60.0
         C6   X2  1.3   X1  90  C5  60.0
      }
```

Atoms with occupancies fixed to zero (dummy atoms) do not take part in any structure factor calculations. The mixing of *point_for_site* and *z_matrix keywords* is possible as follows:

```
rigid
   point_for_site X1
   load z_matrix {
      X2   X1  1.0
      C1   X2  1.3   X1  90
      C2   X2  1.3   X1  90  C1  60.0
      C3   X2  1.3   X1  90  C2  60.0
      C4   X2  1.3   X1  90  C3  60.0
      C5   X2  1.3   X1  90  C4  60.0
      C6   X2  1.3   X1  90  C5  60.0
   }
```

Z-matrix parameters are like any other parameters; they can be equations and parameter attributes can be assigned. For example, refining on the 1.3 bond distance can be achieved as follows:

```
rigid
   point_for_site X1
   load z_matrix {
      X2   X1  1.0
      C1   X2  c1c2 1.3 min 1.2 max 1.4  X1  90
      C2   X2  =c1c2;   X1  90  C1  60.0
      C3   X2  =c1c2;   X1  90  C2  60.0
      C4   X2  =c1c2;   X1  90  C3  60.0
      C5   X2  =c1c2;   X1  90  C4  60.0
      C6   X2  =c1c2;   X1  90  C5  60.0
   }
```

This ability to constrain Z-matrix parameters through the use of equations allows for great flexibility. Example use of such equations could involve writing a particular Z-matrix bond length parameter in terms of other bond length parameters whereby the average bond length is maintained. Or, in cases where a bond length is expected to change as a function of site occupancy, an equation relating the bond length as a function of the site occupancy parameter can be formulated.

9.9.1.3 Translating parts of a rigid body

Once a starting rigid body model is defined, further translate and rotate statements can be included to represent deviations from the starting model. For example, if the C1 and C2 atoms are expected to shift by up to 0.1Å and as a unit then the following could be used:

```
rigid
   load z_matrix {
      X1
      X2   X1  1.0
      C1   X2  1.3   X1  90
      C2   X2  1.3   X1  90  C1  60.0
      C3   X2  1.3   X1  90  C2  60.0
      C4   X2  1.3   X1  90  C3  60.0
      C5   X2  1.3   X1  90  C4  60.0
      C6   X2  1.3   X1  90  C5  60.0
   }
   translate
      tx @ 0 min -.1 max .1
      ty @ 0 min -.1 max .1
      tz @ 0 min -.1 max .1
      operate_on_points "C1 C2"
```

Additional statements have been outlined in bold. The Cartesian coordinate representation allows an additional means of shifting the C1 and C2 atoms by refining on the *ux*, *uy* and *uz* coordinates directly, or,

```
prm a 1.3 min 1.2 max 1.4
prm t1 0 min -.1 max .1
prm t2 0 min -.1 max .1
prm t3 0 min -.1 max .1
rigid
   point_for_site C1 ux =  a Sqrt(3) .5 + t1; uy =  a .5 + t2; uz = t3;
   point_for_site C2 ux =  a Sqrt(3) .5 + t1; uy = -a .5 + t2; uz = t3;
   point_for_site C3 ux = -a Sqrt(3) .5;      uy =  a .5;
   point_for_site C4 ux = -a Sqrt(3) .5;      uy = -a .5;
   point_for_site C5                          uy =  a;
   point_for_site C6                          uy = -a;
```

9.9.1.4 Rotating part of a rigid body around a point

Many situations require the rotation of part of a rigid body around a point. An octahedra (Fig. 9.1) for example typically rotates around the central atom with three degrees of freedom. To implement such a rotation when the central atom is arbitrarily placed requires setting the origin at the central atom before rotation and then resetting the origin after rotation. This is achieved using the Translate_point_amount macro as follows:

```
prm r 2 min 1.8 max 2.2
   rigid
   ...
      point_for_site A0
      point_for_site A1 ux =  r;
      point_for_site A2 ux = -r;
      point_for_site A3 uy =  r;
      point_for_site A4 uy = -r;
      point_for_site A5 uz =  r;
      point_for_site A6 uz = -r;
      Translate_point_amount(A0, -) operate_on_points "A* !A0 "
      rotate @ 0 qa 1 operate_on_points "A* !A0 "
      rotate @ 0 qb 1 operate_on_points "A* !A0 "
      rotate @ 0 qc 1 operate_on_points "A* !A0 "
      Translate_point_amount(A0, +) operate_on_points "A* !A0 "
```

The *point_for_site* keywords could just as well be *z_matrix* keywords with the appropriate Z-matrix parameters. The first Translate_point_amount statement translates the specified points (A1 to A6) by an amount equivalent to the negative position of A0. This effectively sets the origin for these points to A0. The second Translate_point_amount resets the origin back to A0. If the A0 atom happens to be at Cartesian (0, 0, 0) then there would be no need for the Translate_point_amount statements.

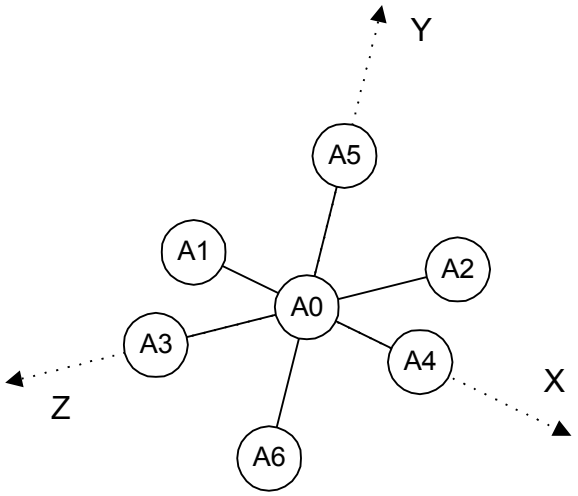


Fig. 9.1: Model of an ideal octahedron. A0: central atom; A1 to A6: outer atoms.

Further distortions are possible by refining on different bond-lengths between the central atom and selected outer atoms. For example, the following macro describes an orthorhombic bipyramid:

```
macro Orthorhombic_Bipyramide(s0, s1, s2, s3, s4, s5, s6, r1, r2)
{
   point_for_site s0
   point_for_site s1 ux   r1
   point_for_site s2 ux  -r1
   point_for_site s3 uy   r1
   point_for_site s4 uy  -r1
   point_for_site s5 uz   r2
   point_for_site s6 uz  -r2
}
```

Note the two different lengths r1 and r2; with r1 = r2 this macro would describe a regular octahedron.

9.9.1.5 Rotating part of a rigid body around a line

Rigid bodies can be created by using the rotate and translate keywords instead of explicitly entering fractional or Cartesian coordinates. For example, two connected Benzene rings, of which a schematic without Hydrogens is shown in Fig. 9.2, can be formulated as follows:

```
prm r 1.3 min 1.2 max 1.4
rigid
   point_for_site C1 ux = r;
   load point_for_site ux rotate qz operate_on_points {
      C2 =r; 60  1 C2
      C3 =r; 120 1 C3
      C4 =r; 180 1 C4
      C5 =r; 240 1 C5
      C6 =r; 300 1 C6
   }
   point_for_site C7 ux = r;
   load point_for_site ux rotate qz operate_on_points {
      C8  =r; 60  1 C8
      C9  =r; 120 1 C9
      C10 =r; 300 1 C10
   }
   translate tx = 1.5 r; ty = r Sin(60 Deg);
      operate_on_points "C7 C8 C9 C10"
```

The points of the second ring can be rotated around the line connecting C1 to C2 with the following:

```
Rotate_about_points(@ 50 min -60 max 60, C1, C2, "C7 C8 C9 C10")
```

The min/max statements limit the rotations to ±30 degrees.

C5 can be rotated around the line connecting C4 and C6 with the following:

```
Rotate_about_points(@ 40 min -50 max 50, C4, C6, C5)
```

Similar Rotate_about_points statements for each atom would allow for distortions of the Benzene rings without changing bond distances.

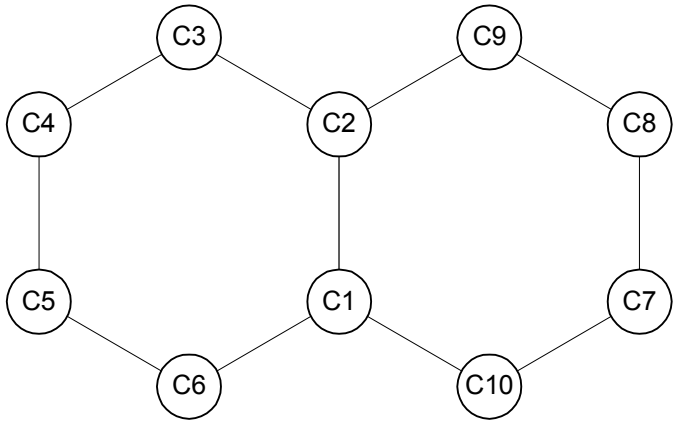


Fig. 9.2: Model of two connected Benzene rings.

9.9.1.6 Benefits of using Z-matrix together with rotate and translate

Cyclopentadienyl $(C_5H_5)^-$ is a well defined molecular fragment which shows slight deviation from a perfect five-fold ring (Fig. 9.3). The rigid body definition using point_for_site keywords is as follows:

```
prm r1 1.19
prm r2 2.24
rigid
   load point_for_site ux { C1 =r1; C2 =r1; C3 =r1; C4 =r1; C5 =r1; }
   load point_for_site ux { H1 =r2; H2 =r2; H3 =r2; H4 =r2; H5 =r2; }
   load rotate qz operate_on_points {  72 1 C2  144 1 C3
                                      216 1 C4  288 1 C5 }
   load rotate qz operate_on_points {  72 1 H2  144 1 H3
                                      216 1 H4  288 1 H5 }
```

and using a typical Z-matrix representation:

```
rigid
   load z_matrix {
      X1
      X2   X1 1
      C1   X2 1.19   X1  90
      C2   X2 1.19   X1  90   C1  72
      C3   X2 1.19   X1  90   C2  72
      C4   X2 1.19   X1  90   C3  72
      C5   X2 1.19   X1  90   C4  72
      X3   C1 1      X2  90   X1   0
      H1   C1 1.05   X3  90   X2 180
      H2   C2 1.05   C1 126   X2 180
      H3   C3 1.05   C2 126   X2 180
      H4   C4 1.05   C3 126   X2 180
      H5   C5 1.05   C4 126   X2 180
   }
```

This Z-matrix representation is one that allows for various torsion angles. It does not however directly allow for all possibilities, for example, no adjustment of a single parameter allows for displacement of the C1 atom without changing the C1-C2 and C1-C3 bond lengths. It is however possible to use the Rotate_about_points macro to achieve the desired result as follows:

```
Rotate_about_points(@ 0, C2, C3, "C1 H1")
```

Thus the ability to include rotate and translate statements together with the *z_matrix* keyword gives greater flexibility in defining rigid bodies.

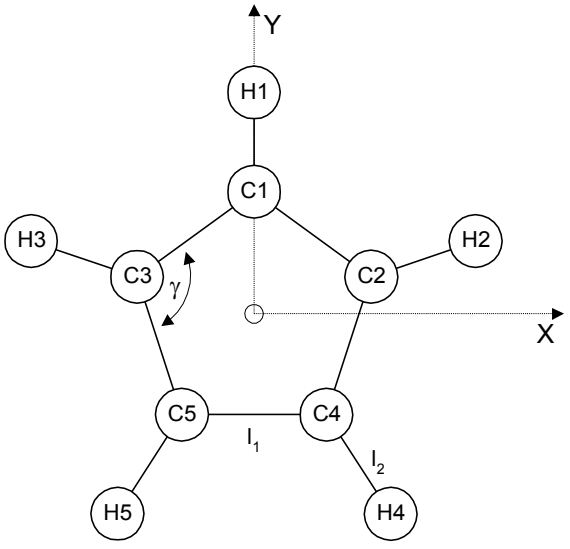


Fig. 9.3: Model of the idealized cyclopentadienyl anion (C_5H_5).

9.9.1.7 The simplest of rigid bodies

The simplest rigid body comprises an atom constrained to move within a sphere; for a radius of 1 then this can be achieved as follows:

```
rigid
   point_for_site Ca uz @ 0 min -1 max 1
   rotate r1 10 qx 1
   rotate r2 10 qx = Sin(Deg r1); qy = -Cos(Deg r1);
```

The coordinates are in fact spherical coordinates; this is preferred as the rotation parameters r1 and r2 are communicative. Constraining an atom to within a sphere is a very important constraint when the approximate atomic position is known.

Setting the distance between two sites, or, two sites A and B a distance 2Å apart can be formulated as:

In Z-matrix form

```
rigid
   z_matrix A                          ' line 1
   z_matrix B A 2                      ' line 2
   rotate @ 20 qa 1                    ' line 3
   rotate @ 20 qb 1                    ' line 4
   translate ta @ .1 tb @ .2 tb @ .3   ' line 5
```

In Cartesian form

```
rigid
   point_for_site A                    ' line 1
   point_for_site B uz 2               ' line 2
   rotate @ 20 qa 1                    ' line 3
   rotate @ 20 qb 1                    ' line 4
   translate ta @ .1 tb @ .2 tb @ .3   ' line 5
```

Lines 1 and 2 defines the two points (note that *ux*, *uy* and *uz* defaults to 0), line 3 and 4 rotate the two points around the a lattice vector and then the b lattice vector respectively and line 5 translates the two points to a position in fractional atomic coordinates of (.1, .2, .3). Lines 3 to 5 contain the five parameters associated with this rigid body.

Instead, the Set_Length macro can be used to set the distance between the two sites as follows:

```
Set_Length(A, B, 2, @, @, @, @ 30, @ 30)
```

where A and B are the site names, 2 is the distance in Å between the sites, arguments 4 to 6 are the names given to the translation parameters and arguments 7 and 8 are the rotational parameters. Set_Length is not supplied with the translate starting values; these are obtained from the A site with the use of the keyword *start_values_from_site* located in the Set_Length macro.

9.9.1.8 Generation of rigid bodies

A rigid body is constructed by the sequential processing of *z_matrix*, *point_for_site*, *rotate* and *translate* operations. The body is then converted to fractional atomic coordinates and then symmetry operations of the space group applied.

The conversion of Z-matrix coordinates to Cartesian is as follows:

- the first atom, if defined using the *z_matrix* keyword, is placed at the origin
- the second atom, if defined using the *z_matrix* keyword, is placed on the positive z-axis
- the third atom, if defined using the *z_matrix* keyword, is placed in the x-z plane

The conversion from Cartesian to fractional coordinates in terms of the lattice vectors a, b, and c is as follows:

- x-axis in the same direction as the a lattice parameter.
- y-axis in the a-b plane.
- z-axis in the direction defined by the cross product of a and b.

Rotation operations are not commutative and thus the rotation of a point A about the vector B-C and then about D-E is not the same as the rotation of A about D-E and then about B-C.

By default *rotate* and *translate* operate on all previously defined *point_for_site*'s; alternatively *point_for_site*'s can be explicitly defined using the *operate_on_points* keyword which identifies sites (see section 9.4). *operate_on_points* must refer to previously defined *point_for_site*'s and it can refer to many sites at once by enclosing the site names in quotes and using the wild card character "*" or the negation character "!", for example:

```
operate_on_points "Si* O* !O2"
```

9.9.1.9 Rigid body parameter errors propagated to fractional coordinates

When *do_errors* is defined, errors are propagated to the site fractional coordinates of sites defined as part of a rigid body.

9.9.1.10 Z-matrix collinear error information

The Z-matrix collinear points exception can be deciphered using information displayed on detection of the error. The collinear error is due to three atoms on a z-matrix line which are collinear. The information displayed includes a snap shot of the rigid body operations pertaining to the error. The following is an example of the information displayed:

```
DB_x_CB Zero dot product - Z-matrix possible collinear points at atoms
   O10
   C16 8.91631604e-016 1.0912987e-014 5.2
   C15 3.72315026e-016 1.0912987e-014 3.9
   C11 0 0 0
Partial z-matrix in error:
```

```
rigid
   z_matrix C11
   z_matrix C12 C11 1.3
   z_matrix C13 C12 1.3 C11 120
   z_matrix C14 C13 1.3 C12 120 C11 180
   z_matrix C15 C14 1.3 C13 120 C11 0
   z_matrix C16 C15 1.3 C14 120 C11 180
   z_matrix O10 C16 1 C15 108 C11 120
```

To investigate why the error is occurring the rigid body fragment can be copied to a Rigid Body Editor window. Looking at the O10 line (using the Rigid Body Editor window) it can be seen that atoms C16, C15, and C11 lie on a straight line; this is invalid as it becomes impossible to form the dihedral angle in a non-degenerate manner. The best way to think about a z-matrix line with 4 atoms A, B, C, D, ie.

```
z_matrix A B # C # D #
```

is to think of two triangles ABC and DBC hinged along the line BC. The angle between the triangles is the dihedral angle. If B, C, D are collinear then there's no triangle DBC and hence the dihedral angle cannot be formed. Thus for z-matrices both A,B,C and B,C,D must not be collinear. Dummy atoms can solve this problem. TOPAS tests for a zero dot product numerically with a tolerance of $1.0e^{-15}$.

9.9.2 Anti-Bump and GRS penalties

Two penalty functions that have shown to facilitate the determination of structures are the anti-bumping (AB) penalty and the GRS penalty (potential energy penalty U).

The anti-bumping penalty is written as:

$$\mathrm{AB_i} = \begin{cases} \sum \left(r_{ij} - r_o\right)^2, & \text{for } r_{ij} < r_o \text{ and } i \neq j \\ 0, & \text{for } r_{ij} \geq r_o \end{cases} \tag{9.14}$$

where r_0 is a bond length distance, r_{ij} the distance between atoms i and j including symmetry equivalent positions and the summation is over all atoms of type j. The *ai_anti_bump* and *box_interaction* keywords are used to implement the penalty of Eq. 9.14 using the AI_Anti_Bump and Anti_Bump macros respectively.

Typically anti-bump constraints applied only to heavy atoms is sufficient; an over use of such constraints can in fact hinder Global Optimization in finding the global minimum. Applying the constraint for the first few iterations of a refinement cycle only can also be beneficial; this is achieved in the AI_Anti_Bump macro by writing the penalty in terms of the reserved parameter Cycle_Iter.

The GRS penalty uses the *grs_interaction* keyword to either calculate the Lennard-Jones or Born-Mayer potentials and is suited to ionic atomic models. For a particular site i they comprise a Coulomb term Ci and a repulsive term Ri and is written as:

$$\mathrm{U_i} = \mathrm{C_i} + \mathrm{R_i} \tag{9.15}$$

where

- $C_i = A \sum Q_i Q_j / r_{ij}$, i≠j
- $R_i = \sum B_{ij} / r_{ij}^n$, for Lennard Jones and i≠j
- $R_i = \sum c_{ij} \exp(-d\, r_{ij})$, for Born-Mayer and i≠j

where $A = e^2/(4\pi\varepsilon_0)$ and ε_0 is the permittivity of free space, Q_i and Q_j are the ionic valences of atoms i and j, r_{ij} is the distance between atoms i and j and the summation is over all atoms to infinity. The repulsive constants B_{ij}, n, c_{ij} and d are characteristic of the atomic species and their potential surrounds. The equation part of *grs_interaction* is typically used to describe the repulsive terms.

The following example defines a bond length restraint using the GRS series between an Aluminum site and three Oxygen sites. Valence charges have been set to +3 and –2 for Aluminum and Oxygen, respectively. The expected bond length is 2 Angstroms between Oxygen sites and 1.5 Angstroms between Aluminum and Oxygen sites.

```
site Al  x @ 0.7491  y @ 0.6981  z @ 0.4069  occ Al+3 1  beq 0.25
site O1  x @ 0.6350  y @ 0.4873  z @ 0.2544  occ  O-2 1  beq 1
site O2  x @ 0.2574  y @ 0.4325  z @ 0.4313  occ  O-2 1  beq 1
site O3  x @ 0.0450  y @ 0.6935  z @ 0.4271  occ  O-2 1  beq 1

Grs_Interaction(O*, O*, -2, -2, oo,  2.0, 5)  penalty = oo;
Grs_Interaction(Al, O*,  4, -2, alo, 1.5, 5)  penalty = alo;
```

The following example defines a bondlength restraint using the AI_Anti_Bump macro between a Potassium site and three Carbon sites. The expected bond length is 4 Angstroms between Potassium sites and 1.3 Angstroms between Carbon sites.

```
site K   x @ 0.14305  y @ 0.21812  z @ 0.12167  occ K 1  beq 1
site C1  x @ 0.19191  y @ 0.40979  z @ 0.34583  occ C 1  beq 1
site C2  x @ 0.31926  y @ 0.35428  z @ 0.32606  occ C 1  beq 1
site C3  x @ 0.10935  y @ 0.30991  z @ 0.39733  occ C 1  beq 1

AI_Anti_Bump(K , K , 4  , 1)
AI_Anti_Bump(C*, C*, 1.3, 1)
```

Note, there's no explicit definition of a penalty function as in the first example. The AI_Anti_Bump macro already includes a predefined penalty function.

9.10 Stacking faults

A super cell approach to stacking faults has been implemented. The keyword *layer* identifies a site as belonging to a layer called $layer_name. The keyword *stack* applies a stacking vector (sx, sy, sz) to the named layer. Structures factors are generated in the usual manner; a shift corresponding to the stacking vector is then applied. stack operates in any space group. Sites that do not belong to a layer are treated as un-stacked and their structure factors are generated in the usual manner.

The example KAOLONITE.INP[1] demonstrates the use of *layer* and *stack*.

1 \TUTORIAL\MISCELLANEOUS\STACKING FAULTS\

9.11 Structure determination

TOPAS integrates 3 approaches to structure determination (Fig. 9.1), choosing the appropriate one is problem dependent:

1. Global optimization (Monte-Carlo, Simulated Annealing)
2. Charge Flipping
3. 3D Fourier analysis

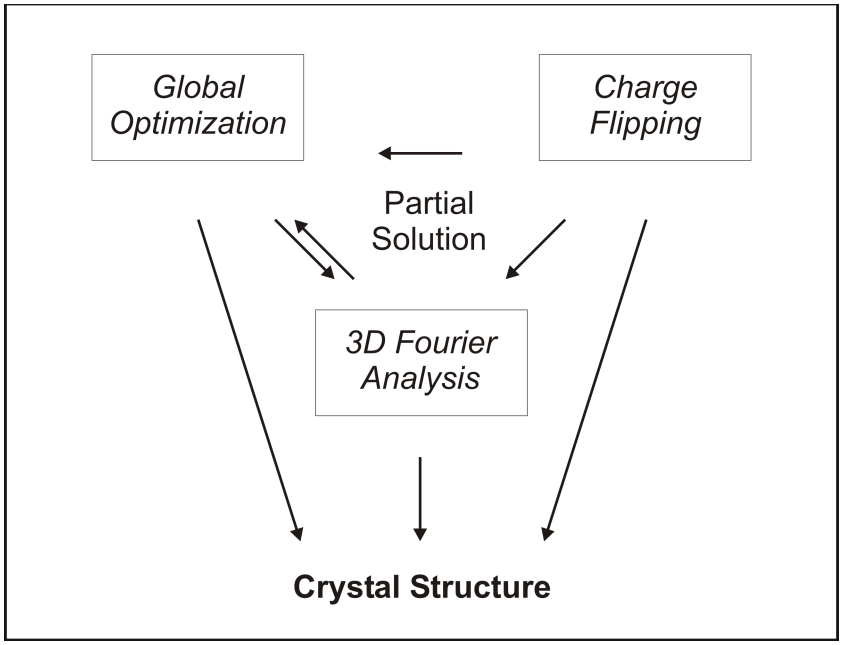


Fig. 9.1: Structure determination approaches in TOPAS.

All approaches are suited for single crystal as well as powder diffraction data, although the limit of structure size / complexity that can be solved is naturally smaller for powders. Both Global Optimization and Charge Flipping are equally capable in solving crystal structures ab-initio; distinct advantages of both methods are discussed below. 3D Fourier Analysis requires an a-priori (partial) structure model, and thus is a tool for structure completion and validation. Partial solutions as found by both Global Optimization and Charge Flipping can be completed by both Global Optimization or 3D Fourier Analysis as indicated by the arrows in Fig. 9.1. Note that Global Optimization and 3D Fourier Analysis can be used iteratively to work from a partial to the completed structure as indicated by the double arrow.

Diffraction data quality is of utmost importance for successful structure determination, and should fulfill the following requirements:

1. Maximum possible data resolution with data ideally ranging to d-values <1Å, that is equivalent to a data cutoff not before ~100° 2θ with Cu-radiation
2. Minimum peak overlap (use a high-resolution diffractometer)
3. High signal to noise ratio (use a variable counting time measurement strategy)

Throughout this section the term "high quality data" is used where all three requirements are fullfilled, while "poor quality data" denotes data where any single requirement is not fullfilled.

Suggested reading:

- **Coelho, A.A. (2000):** *Whole Profile Structure Solution from Powder Diffraction Data using Simulated Annealing.* - J. Appl. Cryst., **33**, 899-908.
- **Coelho, A.A. (2007):** *A Charge Flipping algorithm incorporating the tangent formula for solving difficult structures.* - Acta Cryst., **A36**, 400–406.
- **David, W.I.F., Shankland, K., McCusker, L.B. & Baerlocher, Ch. (2002):** *Structure Determination from Powder Diffraction Data.* - IUCr Book Series, Oxford University Press, 337 pages.

9.11.1 Global Optimization

Global Optimization is a direct space approach to structure determination. As such it is characterized by the ability to supplement the available diffraction information with prior chemical knowledge of the compound under study, e.g. connectivity in conjunction with tabulated bond lengths, bond angles and bond torsion angles. The relevant parameter space can be searched using a simulated annealing algorithm to minimize χ^2. Details about the implementation in TOPAS are given by Coelho (2000).

The general procedure can be summarized as follows:

1. Construct a trial crystal structure by randomly positioning and orienting individual atoms, molecular fragments or complete molecules taking into account (known or guessed) space group information
2. After calculating diffraction data and comparing it against the measured diffraction data, the variable parameters of the model are adjusted in order to maximise the level of agreement between the observed and calculated data (i.e. minimize χ^2)

In the traditional approaches to structure determination from powder data, this procedure is applied to observed structure factors. In a second (and independent) step the structure is refined using either the same observed structure factors (Two-Stage method; Will (1979, 2006)) or step intensity data (Rietveld method; Rietveld (1967, 1969)). In Fig. 9.2 both approaches are classified within the flow of processes required for structure determination and refinement.

Obviously, the ability to deal with observed structure factors allows structure determination and refinement from both single crystal and powder diffraction data.

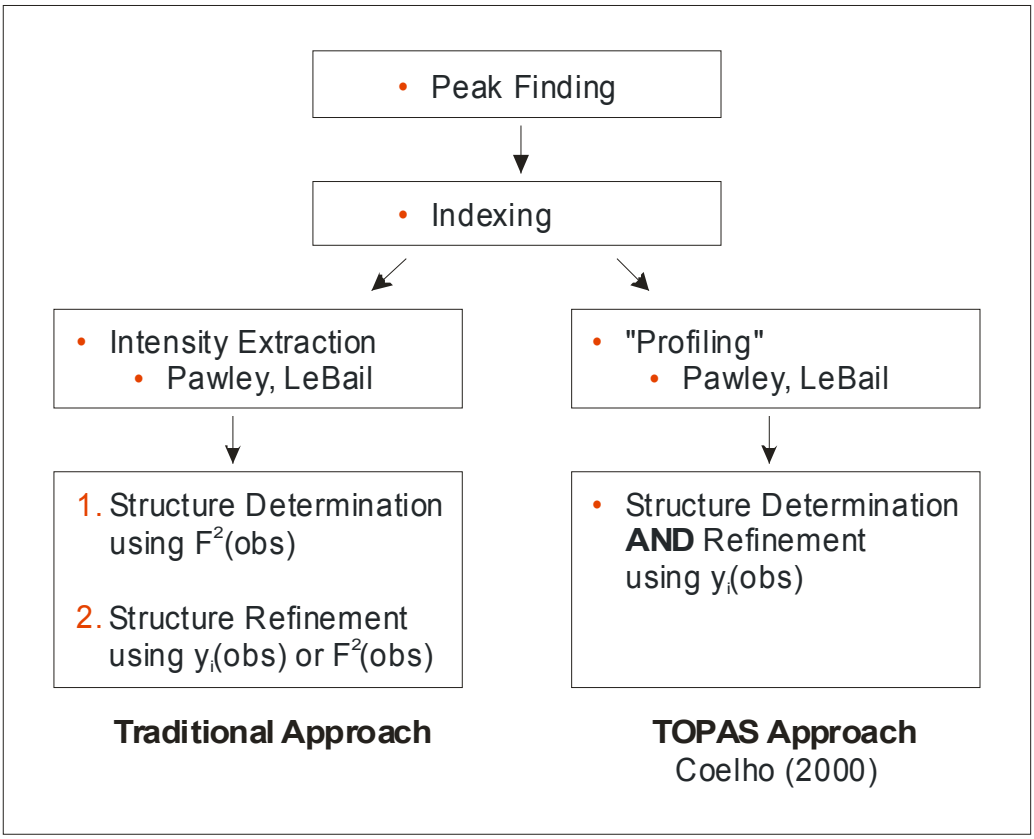


Fig. 9.2: Structure determination processes using Global Optimization supported by TOPAS. F^2(obs): observed structure factors, y_i(obs): observed step intensity data.

For powder diffraction data, usage of observed structure factors requires a preceeding intensity extraction (Pawley or Le Bail fit). High quality data is needed to avoid problems associated with peak overlap (intensity partitioning). Intensity extraction and Global Optimization runs have to be performed using the identical space group. Calculations are fast as the number of data points.

Most significantly, usage of step intensity data does not require preceeding intensity extraction. A single Pawley or Le Bail fit is needed to determine background and profile parameters; the space group for the structure determination step can be choosen arbitrarily. Calculations are slow as the number of data points (step intensities) is high. Often "peak maximum intensities" can be used instead; this data type is represented by non-equidistant step intensity data obtained by "decomposing" a powder pattern into a new diffraction pattern comprising at most one data point per hkl (Decompose macro). Calculations are fast as the number of data points (maximum intensities) is small.

Using step intensity data for structure determination bears the same advantages as for structure refinement using the Rietveld method. Each data point is an observation, no attempts are made to deconvolute overlapped peaks. This avoids problems associated with intensity partitioning and allows structure determination from poor quality powder data. Note that the possibility to use step intensity data and peak maximum intensities is unique to TOPAS.

9.11.1.1 Global Optimization usage

A Monte-Carlo process is initiated by defining *continue_after_convergence*. After convergence a new refinement is initiated with parameter values changed according to any defined *val_on_continue* attributes and *rand_xyz* or *randomize_on_errors* processes. Search of the relevant parameter space is improved by additionally defining a temperature regime using the *temperature* keyword: Simulated Annealing.

The macro Auto_T greatly simplifies the definition of Simulated Annealing runs and has be shown to be adequate for a wide range of examples.

By applying an appropriate weighting scheme to the diffraction data the search for the global minimum can be facilitated.

For powder data the default weighting scheme is:

```
weighting = If(Yobs <= 1, 1, 1 / Yobs);
```

For single crystal data the following, which is proportional to 1/d, works well but can be also tried for powder data:

```
weighting = 1 / (Sin(X Deg / 2) Max(Yobs,1));
```

A more elaborate scheme is as follows:

```
weighting = ( Abs(Yobs-Ycalc) / Abs(Yobs+Ycalc) +1) / Sin(X Deg / 2);
```

Note that Global Optimization is not specific to structure determination and can be also applied to any other application to assist finding the global minimum.

9.11.1.2 General guidelines

The following general guidelines for structure determination have been shown to be adequate in many cases:

- An atom in general position corresponds to 3 degrees of freedom (DoFs) resulting in a high number of DoFs for large structures. Introducing suitable geometric constraints / restraints / penalties can reduce both the number of DoFs as well as the number of local minima in χ^2 and correspondingly increase the chances of obtaining a global minimum. Irrespectively to the number of atoms, each molecular fragment or complete molecule, described as a rigid body, corresponds to 3 positional and 3 orientational DoFs. As a rule of thumb there should be not less than about 5 independent reflections per DoF.
- In general the calculation speed as well as the number of local minima in χ^2 correspond to the number of DoFs, so they should be kept small. However, an over use of constraints can in fact hinder the structure determination process in finding the global minimum, if e.g. the movement of atoms or rigid bodies becomes too restricted.
- For powder diffraction data, step intensity data should be tried first. This approach is most straightforward, and avoids problems associated with peak overlap (intensity partitioning). Maximum intensities can be used to speed up calculations, unless data are too noisy.

- Structure determination without or with partial knowledge of connectivity:
  - An unconstrained trial crystal structure (individual atoms only) should be tried in first place. Heavy atoms can often be located after some refinement cycles.
  - Appropriate constraints should be added step by step. The goal should be to introduce connectivity information and to reduce the number of DoFs. Examples are e.g. limiting the movement of heavy atoms located in the first step, or enforcement of bond distances and angles. The latter can range from general bondlength constraints for selected atom species up to the creation of rigid bodies, where connectivity is known. In general, over-use of constraints should be avoided, in particular if the data quality is good.
- Structure determination with knowledge of connectivity:
  - Involves creation of a rigid body and to refine its position and orientation in the cell. If heavy atoms are present, this process can be significantly simplified by locating them first using an unconstrained trial crystal structure as discussed above. Structure determination success will greatly depend on how accurately the true structure is represented by the rigid body model.
  - Difficulties typically arise in presence of unknown torsion angles. In many cases it can be sufficient to adjust these simultaneously, but each torsion angle adds one DoF to the problem. An alternative approach is to break up the rigid body into fragments (and even individual atoms) if data quality is good.

The structure determination process can be monitored using the *Structure Viewer* window (*view_structure* keyword). This is very useful to

- improve the temperature regime, if displacements of atoms, fragments or molecules are too small or too high
- identify over use of constraints, if movements of atoms, fragments or molecules is too restricted
- identify heavy atom positions, if atoms tend to move repeatedly to the same position

Note, that the use of the *Structure Viewer* window slows down the calculation speed. For maximum calculation speed turn graphics animation off.

9.11.2 Charge Flipping

The Charge Flipping method (Oszlányi & Süto, 2004) for structure determination is supported with a number of enhancements (Coelho, 2007), e.g. the inclusion of the tangent formula (Karle & Hauptman, 1956). Only cell parameters, reflection indices and intensities are required as input. The electron or scattering density is sampled on a discrete rectangular grid of pixels.

The Charge Flipping algorithm is illustrated in Fig. 9.3. Before starting iterations, a starting set of structure factors is created by combining the experimental structure-factor amplitudes $|Fhkl|_{obs}$ with random phases.

One iteration cycle involves four steps:

1. Calculation of a (trial) electron density
2. Flip all charges below a positive threshold δ by changing their sign (this is the name giving part). The value of δ is a fraction of a typical light atom peak and is the only parameter of the algorithm. Pixels above this value are accepted unchanged on the assumption that they belong to atomic peaks.
3. Calculation of temporary ("flipped") structure factors
4. Calculation of new structure factors by accepting the "flipped" phases and replacing the moduli by the experimental $|Fhkl|_{obs}$.

The algorithm seeks a Fourier map that is stable against repeated flipping of all density regions below δ. For poor quality data or for difficult structures the inclusion of the tangent formula can facilitate solution and sharpen electron densities. For details of the TOPAS implementation refer to Coelho (2007).

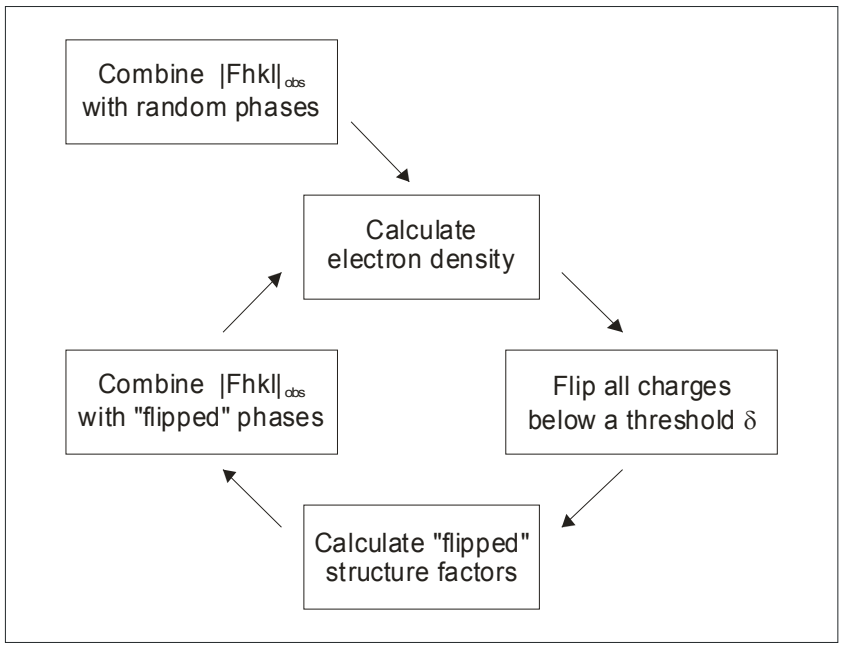


Fig. 9.3: Charge Flipping.

The method has several attractive properties that make it the method of choice in many cases as it does not require any information about atom types, chemical composition or any information on the space group symmetry. Apart from the routine solution of all types of organic and inorganic structures, charge flipping is the preferred approach for structure solution of structures with ambiguous space group or with unknown composition.

Charge Flipping uses integrated intensities from either single crystal or powder diffraction data, the latter require preceeding intensity extraction (Pawley or Le Bail fit). High quality data is needed to avoid problems associated with peak overlap (intensity partitioning).

9.11.2.1 Charge Flipping usage

Charge Flipping works particularily well on data at good resolution (<1Å resolution). For data at poor resolution or for difficult structures then inclusion of the tangent formula can facilitate solution and sharpen electron densities. Powder diffraction data usually fall under the poor resolution/data quality category and as such additional symmetry restraints using *symmetry_obey_0_to_1* can sharpen electron densities.

When using Charge Flipping the choice and amount of perturbation necessary for finding a solution are important considerations. Not enough perturbation leads to the system being trapped within a local parameter space; too much perturbation may lead to a solution not being found and in addition contrast in R-factors prior to and at convergence are diminished leading to difficult to identify solutions. The Ramp macro can be used to gradually vary control parameters, here are some examples:

```
fraction_density_to_flip = Ramp(0.85, 0.8, 100);
fraction_reflections_weak = Ramp(0.5, 0, 100);
flip_regime_2 = Ramp(1, 0, 200);
flip_regime_3 = Ramp(0.25, 0.5, 200);
symmetry_obey_0_to_1 = Ramp(0.5, 1, 100);
tangent_scale_difference_by = Ramp(0, 1, 100);
```

Choosing control parameters in this manner gradually decreases perturbation allowing for solutions to be found and identified. This is similar to a simulated annealing process where temperatures start at high values and then progressively lowered.

9.11.2.2 Perturbations

Perturbations can be categorized as being of either phase, structure factor intensity or electron density perturbations. There are two built in flipping regimes of *flip_regime_2* and *flip_regime_3* and one user defined regime *flip_equation*. Only one can be used and they all modify the electron density. In the absence of a flipping regime the following is used:

$$\rho = \begin{cases} -\rho, & \text{for } \rho < \delta \\ \rho & \text{for } \rho \geq \delta \end{cases} \tag{9.16}$$

where δ corresponds to the electron density threshold.

Using the tangent formula on either difficult structures or on data at poor resolution often leads to uranium atom solutions. Uranium atom solutions can be avoided by modifying the electron density using a flipping regime that dampens high electron densities or by using *pick_atoms*.

Using a large number of triplets per Eh value (a value for *tangent_max_triplets_per_h* greater than 100) reduces perturbation, increases occurrences of uranium atom solutions and increases the chances of finding a solution after an initial phase randomization. A large number of triplets would typically be used for poor resolution data; correspondingly a flipping regime that avoids uranium atom solutions should be chosen.

Perturbations mostly increase randomness in the system with the exceptions of the tangent formula, *scale_density_below_threshold* and *histogram_match_scale_fwhm*.

9.11.2.3 The Ewald sphere, weak reflections and Charge Flipping termination

By default Charge Flipping uses the minimum observed d spacing to define the Ewald sphere; alternatively *min_d* can be used. The Ewald sphere can be increased using *extend_calculated_sphere_to*; this inserts missing reflections and gives them the status of "weak" reflections. Weak reflections are also inserted for missing reflections within the Ewald sphere. Weak reflection phases and structure factors can be modified using *scale_weak_reflections* and *add_to_phases_of_weak_reflections*.

Reflections that have zero intensities according to the space group are not included in Charge Flipping; correspondingly the number of observed reflections removed are reported. Structure factor intensities within a family of reflections are determined by averaging the observed structure factors intensities. This averaging is also performed on calculated intensities each Charge Flipping iteration for weak reflections.

Changing the space group is possible; changing the space group to a higher symmetry from that as implied in the input hkl file often makes sense. Changing the space group to a lower symmetry implies less symmetry constraints and is useful for checking whether a significantly better R-factor is realized.

Typically a fraction of observed reflections are given the status of "weak" using *fraction_reflections_weak*. When a solution is found and Charge Flipping terminates a *.FC file is saved; this file comprises structures factors that produced the best R-factor. A new Charge Flipping process can be initiated with phase information saved in the *.FC file using the Restart_CF macro. To further complete the structure the new Charge Flipping process may for example reduce perturbations in order to sharpen the electron density.

9.11.2.4 Powder data considerations

For powder data it is usually best to maximize the number of constraints due to poor data quality; it is also best to use *.A files as generated by a Pawley refinement and to then use *cf_in_A_matrix*. No weak observed reflections within the observed Ewald sphere should be assigned by setting *fraction_reflections_weak* to zero. Instead weak reflections can be included by extending the Ewald sphere with something like:

```
extend_calculated_sphere_to 1
add_to_phases_of_weak_reflections = 90 Ramp(1, 0, 100);
```

If the Ewald sphere is extended such that the weak reflections are many then some of these weak reflections could well be of high intensity. Subsequently offsetting high intensity weak reflections by a constant could lead to too much perturbation and thus the following may be preferential:

```
extend_calculated_sphere_to 1
add_to_phases_of_weak_reflections = Rand(-180,180) Ramp(1, 0, 100);
```

In a Pawley refinement the calculated intensities at the low d-spacing edge of the pattern may be in error to a large extent; it is therefore best to remove these reflections using *delete_observed_reflections*, for example:

```
delete_observed_reflections = D_spacing < 1.1;
```

9.11.2.5 The algorithm of Oszlányi & Süto (2005) and F000

Normalized structure factors enhances the chances of finding a solution (Oszlányi & Süto, 2005) and are realized by inclusion of *f_atom_type*'s and when *correct_for_temperature_effects* is non-zero. In the original algorithm the amount of charge flipped is a function of the maximum electron density; this can be realized using:

```
user_threshold = 0.2 Get(max_density_at_cycle_iter_0);
```

Get(max_density_at_cycle_iter_0) is a different value at the start of each Charge Flipping process as phases are chosen randomly. An alternative means of defining the threshold is:

```
fraction_density_to_flip 0.83
```

The Charge Flipping process is sensitive to the threshold value. To overcome this sensitivity the fraction_density_to_flip parameter could be ramped as a function of iteration from a high value to a low value, or,

```
fraction_density_to_flip = Ramp(0.85, 0.8, 100);
```

F000 is allowed to float when scale_F000 is set to 1. In the Oszlányi & Süto (2005) algorithm a floating F000 produces the best results for some structures but not for others.

When the electron density is perturbed then a floating F000 often produces unfavourable oscillations in R-factors. In general the electron density is best left unperturbed when scale_F000 is non-zero.

9.11.3 3D Fourier Analysis

3D Fourier Analysis is a powerful tool for structure completion and validation allowing to calculate 3-dimensional F_{obs}, F_{calc}, and difference maps for X-ray (electron density) and neutron data (scattering density).

F_{obs} maps use experimental structure-factor amplitudes along with phases calculated from some structural model. In favourable cases these maps will show peaks that do not correspond to atoms in the current model, thus allowing location of missing atoms.

F_{calc} maps use experimental structure-factor amplitudes and phases calculated from some structural model. These maps only have peaks corresponding to atom positions in the model and thus are not very informative.

Difference maps give a representation of that part of the electron / scattering density that has not been correctly accounted for in the current atomic model, emphasizing the incorrect features of that model. Difference maps are obtained by combining the difference of experimental and calculated structure-factor amplitudes with calculated phases. The difference is usually calculated from (F_{obs} - F_{calc}) or (F_{obs} – $2F_{calc}$); different equations can be defined by the user to e.g. sharpen the densities. In difference maps peaks are seen where there are missing atoms, while valleys indicated atoms which have been incorrectly placed.

9.11.4 Concluding remarks

The following overview about the most important properties of Global Optimization, Charge Flipping and 3D Fourier analysis will allow to select the most promising method to start with for a given sample / data set:

Global Optimization:

- Requires an a-priori (trial) structure model, which can be partial or random
- Performs better on poor quality data. **Important advantage!**
- Performs well for structure completion
- Comparatively slow
- Simulated Annealing should be always applied to search parameter space

Charge Flipping

- No use of chemistry / a-priori structure models. **Important advantage!**
- Requires high quality data
- Very fast; with high quality data structures can be solved in seconds up to a few minutes
- Can quickly find partial structures, specifically heavy atom positions

3D Fourier analysis

- Requires an a-priori (partial) structure model
- Requires high quality data
- Performs well for structure completion

For ab-initio structure determination it is generally recommended to always start with Charge Flipping first, unless there is an a-priori (trial) structure model known to be close to the current structure, or if data quality is so poor that Charge Flipping is obviously no option. The time required to setup a Charge Flipping run is minimal (maybe a few minutes at maximum), and structures that can be solved with Charge Flipping will usually solve in seconds up to a few minutes. If Charge Flipping fails, then Global Optimization is the option of choice. Chances are very good though, that Charge Flipping may have already found a partial structure; this is often the case for heavy atoms. The probability for successful structure determination / completion with Global Optimization can be significantly increased by inclusion of a partial structure model found this way.

9.12 Magnetic Structure Refinement

Magnetic structure refinement is supported using the following keywords and macros:

```
str...
   [mag_only_for_mag_sites]
   [mag_space_group $symbol]
   site...
      [mlx E] [mly E] [mlz E] [mg E]
      [mag_only]

      ' site dependent macros
      MM_CrystalAxis_Display(mxc, myc, mzc)
      MM_CrystalAxis_Refine(mxc,mxv,myc,myv,mzc,mzv,mlx_v,mly_v,mlz_v)
      MM_Cartesian_Display(mxc, myc, mzc)
      MM_Cartesian_Refine(mxc,mxv,myc,myv,mzc,mzv,mlx_v,mly_v,mlz_v)
```

Magnetic intensity is given by:

Magnetic intensity = Fmagcperp . Fmagcperp* = |Fmagcperp|

where the superscript * denotes conjugate gradient and:

Fmagcperp = Fmagc - (Fmagc . Qhat) Qhat

Or in words, Fmagcperp is the component of the magnetic vector in the direction perpendicular to the scattering vector Q, where

$Q = (L^{-1})^T * h$
Qhat = Q / |Q|

where

- L is the Cartesian lattice parameters in 3x3 matrix form
- h is the Miller indices in vector form
- * denotes matrix multiplication
- Superscript $^{-1}$ denotes matrix inverse
- Superscript ^{T} denotes matrix transpose
- $(L^{-1})^T$ = reciprocal lattice parameters

Fmagc in terms of the Cartesian lattice parameters is:

Fmagc = L * Fmag

Fmag for the plane h for a single site is:

$Fmag = \sum_j (B_j * m) \exp(2\pi i Uj)$

where the summation is over the equivalent positions j and

- $U_j = h.R_j x + h.t_j$
- x = { x, y, z } = site fractional coordinates
- m = { mlx, mly, mlz } = magnetic moment
- R_j = rotation part of space group operator

- t_j = translational part of space group operator
- $d_j = s_j$ determinant(R_j) = s_j det(R_j)
- $B_j = s_j$ det(R_j) R_j = magnetic transformation matrix

The file MAGDATA.DAT (courtesy by Robert Von Dreele) comprises data for calculating magnetic form factors. The Lande splitting factor can be refined using the site dependent parameter *mg*; defaults for *mg* are obtained from MAGDATA.DAT. Shubnikov groups are obtained from the file SHUBNIKOVGROUPS.TXT.

mag_only: When defined the diffraction component to intensity for the site in question is ignored.

mag_only_for_mag_sites: When defined the diffraction component to intensity for all magnetic sites for the *str* in question is ignored.

Magnetic moments (Occupancy B_j*m) are displayed graphically when *view_structure* is defined. For the case where the atom balls are masking the display of the magnetic moment arrows the "Atom size" can be varied.

9.12.1 Magnetic refinement warnings / exceptions

The following two messages:

```
Warning: Magnetic moment mlx of site Fe has no contribution to Fmag

Magnetic moment mlx of site Fe cannot be refined
as it has no derivative
```

arise when for each group of equivalent positions of a special position the first row of the matrix $\sum_j B_j$*m is zero where the j's sum over the equivalent positions of a special position group. Similar messages for *mly* and *mlz* are given. Note the fact that *mlx*, *mly*, *mlz* may or may not be refined and their associated constraints are considered. Refinement terminates in the case of the latter message when *mlx* is being refined.

9.12.2 Decomposing Fmag for speed

When using magnetic space groups other than 1.1 equivalent positions of the space group are written in terms of other equivalent positions.

Let

$C_j = \cos(U_j)$
$S_j = \sin(U_j)$
$\text{Exp}(i\,U) = C_j + i\,S_j$ = Euler's formulae

1. For two equivalent positions of a special position we have

   $U_1 = U_2 = U$

   $$\begin{aligned} Fmag_1 + Fmag_2 &= s_1 \det(R_1)\, R_1\, m\, \text{Exp}(i\,U) + s_2 \det(R_2)\, R_2\, m\, \text{Exp}(i\,U) \\ &= (s_1 \det(R_1)\, R_1 + s_2 \det(R_2)\, R_2)\, m\, \text{Exp}(i\,U) \\ &= c\, m\, \text{Exp}(i\,U) \end{aligned}$$

   c is independent of x. Note, a particular special position could have many equivalent positions.

2. If $R_1 = -R_2$ and $t_1 = -t_2$ for two equivalent positions then

   $U_1 = -U_2 = U$
   $Fmag_1 + Fmag_2 = s_1 \det(R) R\ m\ Exp(i\ U) + s_2 \det(-R) (-R)\ m\ Exp(-i\ U)$

   Now,

   $\det(R) R = \det(-R) (-R)$

   or,

   $Fmag_1 + Fmag_2 = \det(R) R\ m\ (s_1\ Exp(i\ U) + s_2\ Exp(-i\ U))$

   For $s_1 = s_2$

   $Fmag_1 + Fmag_2 = s_1 \det(R) R\ m\ 2\ C$

   For $s_1 = -s_2$

   $Fmag_1 + Fmag_2 = s_1 \det(R) R\ m\ (2\ i\ S)$

3. If $R_1 = R_2$ for two equivalent positions then

   $Fmag_1 + Fmag_2 = s_1 \det(R) R\ m\ Exp(i\ h.\ R\ x)\ Exp(i\ h.t_1) + s_2 \det(R) R\ m\ Exp(i\ h.\ R\ x)\ Exp(i\ h.t_2)$

   $= \det(R) R\ m\ (s_1\ Exp(i\ h.t_1) + s_2\ Exp(i\ h.t_2))\ Exp(i\ h.\ R\ x)$

   $= c\ Exp(i\ h.\ R\ x)$

   c is independent of x and is calculated only once. Many R's can be the same for a particular space group with only the t's changing.

4. Calculating C and S:

   $Exp(i\ (h\ .\ R\ x + h.\ t)) = Exp(i\ h\ .\ R\ x)\ Exp(\ i\ h\ .\ t)$

   $Exp(i\ h\ .\ t)$ is constant for a particular h and is calculated only once.

   Only unique $Exp(i\ h\ .\ R\ x)$ are calculated.

   Trigonometric recurrence is used to calculate sines and cosines resulting in three cosine and three sine operations per unique equivalent r. In other words a sin and cos is not calculated for each h.

   Note a sin or cos function is equivalent to about 40 to 60 multiplies.

9.13 CIF

The following macros and Get's can be used to output data in CIF format:

```
Out_CIF_STR(file)
Out_CIF_ADPs(file)
Out_CIF_STR(file, with_id)
Out_CIF_Bonds_Angles(file)
Get(number_of_parameters)
Get(refine_ls_shift_on_su_max)
Get(weighting)
Xi = a reserved parameter name
```

_refine_ls_shift/su_max can be accessed using Get(refine_ls_shift_on_su_max) when *do_errors* is defined and when *continue_after_convergence* is NOT defined. A message similar to the following is displayed on calculation:

```
refine_ls_shift_on_su_max 0.409610469 corresponds to
parameter m501b939c_3 of object prm_10
```

Get(*weighting*) and Xi can be used as follows:

```
xdd_out file append load out_record out_fmt out_eqn
{
   " %9.0f" = Xi;
   " %11.5f" = X;
   " %11.5f" = Ycalc;
   " %11.5f" = Yobs;
   " %11.5f\n" = Get(weighting);
}
```

Get(*weighting*) returns the following masked with excluded regions:

```
1 / Max(1, Yobs), if SigmaYobs does not exist
```

```
1 / SigmaYobs^2, if SigmaYobs does exist
```

If *weighting* is a function of YCalc etc. then it returns the last weighting calculated depending on *recal_weighting_on_iter*.

10 QUANTITATIVE PHASE ANALYSIS

Quantitative phase analysis (QPA) of crystalline and amorphous phases by powder diffraction methods is the only analytical technique that is truly phase sensitive. The fundamental relationships used in QPA are well established and are based on the fact that each crystalline and amorphous component has an intensity contribution to the diffraction pattern which is proportional to its amount. Methods for QPA are mature and extensively covered in literature.

TOPAS offers a wide range of whole pattern methods for QPA of crystalline as well as amorphous phases:

- Traditional Rietveld method
- Internal standard method
- External standard method (O'Connor & Raven, 1988)
- PONKCS method (Scarlett & Madsen, 2006)
- Pattern scaling (Chipera & Bish, 2002)
- Degree of crystallinity method
- Calibration method

Most significantly, for a given multiphase sample, all these methods can be used simultaneously to describe individual phase contributions to the powder pattern. Accurate and reliable QPA is further facilitated by a series of associated features:

- Calibration of results obtained with any of the above methods
- Elemental composition calculation
- Constraints and restraints
  - Instrument function
  - Measured patterns (background, crystalline phases, amorphous phases)
  - Parametric refinements of the sample chemistry, e.g. via Vergard's law
  - Constrained / restrained phase composition
  - Constrained / restrained elemental composition
- Automatic removal of phases below user-defined threshold concentrations during refinement
- Brindley correction of microabsorption effects (Brindley, 1945)

For the determination of crystalline and / or amorphous material, both the analytical question as well as available prior knowledge about the sample will dictate the methodology used. Analysis of crystalline phases is comparatively straightforward, but can be difficult in the presence of amorphous material. In ideal cases, it is possible to detect, identify and quantify crystalline phases at concentrations of even less than 0.1%, and amorphous phases at concentrations of downto or even less than 1%.

Limitations are the same for quantitative analysis of crystalline and amorphous phases and are dictated by sample properties and the analytical technique used. A particular role in this is played by the "peak overlap" problem, that is difficulties related to the accurate distinction between individual intensity contributions from crystalline and amorphous components as well as "background intensity" (e.g. sample fluorescense and any contributions from the sample environment).

Suggested reading:

- **Kern, A. (2008):** *Profile Analysis.* - Principles and Applications of Powder Diffraction. Editors: Clearfield, A., Bhuvanesh N. & Reibenspies J. Blackwell Publishers, 400 pages. ISBN: 978-1-405-16222-7
- **Kern, A., Madsen, I.C. and Scarlett, N.V.Y. (2012):** *Quantifying amorphous phases.* - In "Uniting Electron Crystallography and Powder Diffraction". Editors: Kolb, U., Shankland, K., Meshi, L., Avilov, A. & David, W. Springer, 434 pages.
- **Madsen, I.C. and Scarlett, N.V.Y. (2008):** *Quantitative Phase Analysis.* - In "Powder Diffraction: Theory and Practice". Editors: Dinnebier, R.E. and Billinge, S.J.L. The Royal Society of Chemistry, Cambridge, UK, 582 pages.
- **Madsen, I.C., Scarlett, N.V.Y., Cranswick, L.M.D., and Lwin, T. (2001):** *Outcomes of the International Union of Crystallography Commission on Powder Diffraction Round Robin on Quantitative Phase Analysis: samples 1a to 1h.* - J. Appl. Cryst., **34**, 409-426
- **Madsen, I.C., Scarlett, N.V.Y. and Kern, A. (2011):** *Description and survey of methodologies for the determination of amorphous content via X-ray powder diffraction*. - Z. Krist., **226**, 944-955.
- **Madsen, I.C., Scarlett, N.V.Y., Riley, D.P. and Raven, M.D. (2012):** *Quantitative Phase Analysis using the Rietveld Method.* - In "Modern Diffraction Methods". Editors: Mittemeijer, E.J. and Welzel, U. Wiley-VCH, 554 pages.
- **Scarlett, N.V.Y., Madsen, I.C., Cranswick, L.M.D., Lwin, T., Groleau, E., Stephenson, G., Aylmoree, M. and Agron-Olshina, N. (2002):** *Outcomes of the International Union of Crystallography Commission on Powder Diffraction Round Robin on Quantitative Phase Analysis: samples 2, 3, 4, synthetic bauxite, natural granodiorite and pharmaceuticals*. - J. Appl. Cryst., **35**, 383-400

10.1 Methodology

Table 10.1 lists important properties of the methods supported by TOPAS for QPA.

Table 10.1: Important properties of the QPA methods supported.

Method	Calculation of amorphous content	Standardisation	Can correct for microabsorption errors	Can deal with more than one amorphous phase
Rietveld	Direct	Crystal structure for each phase	No	Yes
Internal standard	Indirect	Standard added to each sample	No	No
External standard	Indirect	External standard measured in regular intervals	No	No
PONKCS	Direct	One-off calibration per phase	Partly	Yes
Pattern scaling	Direct	Reference diffraction data for each phase	No	Yes
DOC	Direct	Case dependent	No	Yes
Calibration	Case dependent	Calibration suite	Yes	Yes

While QPA of well crystalline materials is comparatively straightforward, poorly crystalline or amorphous content can be difficult to quantify. More often than not, polycrystalline materials contain amorphous or poorly crystalline phases, whose intensity contributions to the diffraction pattern is not always evident, especially at low concentrations. In many cases, the presence of amorphous or poorly crystalline phases is undetected or simply ignored. While information about amorphous phase amounts is frequently not sought-after, this is also a result of the preferred (and sometimes indiscriminate) use of the traditional Rietveld method, as one of the most convenient methods for QPA available (e.g. Madsen et al., 2001; Scarlett et al., 2002). However, it is frequently overlooked, that the traditional Rietveld method only delivers relative phase amounts by default; in the presence of amorphous and/or any amount of unidentified crystalline phases, the analyzed crystalline weight fractions may be significantly overestimated. Therefore it is crucial to have other QPA methods at hand and to chose the most appropriate one(s). With respect to amorphous content, available evaluation methods can be grouped into two categories:

1. Indirect methods are based on the use of internal or external standards. Only crystalline components are analyzed and subsequently put on an absolute scale. The amorphous fraction is then calculated by difference, that is indirectly. Most importantly, unidentified and amorphous phases cannot be distinguished, only the sum of these phases can be estimated. Indirect methods are suited for

amorphous phases without clearly evident intensity contribution to the diffraction pattern.

2. Direct methods provide a direct measure of an amorphous compound by analyzing the amorphous intensity contribution to the diffraction pattern. Individual unidentified and amorphous phases may be distinguished, and some approaches allow the quantification of more than one amorphous phase if data quality is sufficient. Direct methods rely on the ability to observe the intensity contribution of amorphous phases to the diffraction pattern.

Most significantly, microabsoption effects can be only dealt with appropriately using calibration based methods. Specifically the use of the Brindley (1945) correction should be avoided due to the great risk of overcorrection (Scarlett et al., 2002).

Benefits and limitations of the individual analytical techniques are to be thoroughly considered when choosing the best method for a given mixture; some basic principles can be summarized as follows:

- Calibration based methods usually have the potential to achieve the highest accuracy, as most aberrations are included in the calibration function. In TOPAS it is possible to correct results obtained with any method when a calibration function is available.
    - Any method based on calibration suites or standards have issues with the availability of suitable calibration samples and standards. Where single samples are to be analyzed, it is generally not practical to make a calibration suite.
    - Any calibration sample and standard will contain amorphous materials which, if not accounted for, will decrease accuracy. This is also true for materials deemed to be perfectly crystalline, as any material possesses a non-diffracting surface layer with some degree of disorder due to relaxation of the crystal structure and inclusion of surface reaction products and adsorbed species (water, hydrocarbons, ...) Such a layer can easily account for a mass fraction of several percent in a finely divided solid.
- The traditional Rietveld method only delivers relative phase amounts by default; in the presence of amorphous and/or any amount of unidentified crystalline phases, the analyzed crystalline weight fractions may be significantly overestimated
    - Most phase abundances reported in literature, obtained via Rietveld analysis, are provided in a manner suggesting absolute values. Where no allowance of amorphous and/or unidentified phases has been made / reported, it is reasonable to assume relative phase abundances instead
- Intensity contributions of amorphous phases to the diffraction pattern are not always evident, especially at low concentrations. In this case amorphous bands will be difficult to model and indirect methods (Internal or External Standard Method) will usually perform better.
    - Any errors in structural models used for crystalline phases have the potential to decrease the overall accuracy. This is especially true when structure models for amorphous materials are selected.

- Where intensity contributions of amorphous phases are evident, any method based on modeling amorphous bands provides improved accuracy. These methods will usually require a sample of pure amorphous material, or a sample where the amorphous content is high, to establish an accurate model (single peak, PONKCS, DOC).

10.2 Methods for quantitative phase analysis

The following are items associated with QPA and the QPA methods supported by TOPAS:

```
xdd... / user_y...
      mixture_MAC
      mixture_density_g_on_cm3
      weight_percent_amorphous
      elemental_composition
      element_weight_percent...
      element_weight_percent_known...
      prm = Get(sum_smvs)...
      K_Factor_MAC_K                                ' macro
      Mixture_LAC_1_on_cm...                        ' macro
      str... / hkl_Is... / x0_Is... / d_Is...
            scale...
            weight_percent
            cell_mass
            cell_volume
            phase_MAC
            spiked_phase_measured_weight_percent
            corrected_weight_percent
            K_Factor_MAC_K                          ' macro
            prm = Get(sum_smvs)...
            prm = Get(smv)...
            prm = Get(sum_smvs_minus_this)...
            prm = Get_Element_Weight(atom)...
            Phase_LAC_1_on_cm                       ' macro
            Phase_Density_g_on_cm3                  ' macro
            Apply_Brindley_Spherical_R_PD           ' macro
```

Note the enormous flexibility provided by this architecture. *str*, *hkl_Is*, *x0_Is* and *d_Is* keywords have scale factors and weight percent associated to them thus allowing to quantify phases with and without known crystal structure simultanously.

Most significantly, quantitative phase amounts can be determined using several QPA methods simultaneously, thus enabling cross-validation of results.

It is also possible to write equivalent terms in the form of equations, for example:

```
' This is weight_percent
prm = 100 Get(smv) / Get(sum_smvs);: 0

prm q = spiked_phase_measured_weight_percent /
        spiked_phase_measured_weight_percent_wt; : 0

' This is corrected_weight_percent
prm = q Get(weight_percent); : 0

' This is weight_percent_amorphous
prm = 100 (1 - q); : 0
```

10.2.1 Rietveld method

In the Rietveld method, the weight fraction W_α of the crystalline phases present in each sample is estimated using Hill and Howard (1987) relationship:

$$W_\alpha = \frac{S_\alpha (ZMV)_\alpha}{\sum_{j=1}^{n} S_j (ZMV)_j} \tag{10.1}$$

where

- S_α is the Rietveld scale factor for phase α,
- ZM is the mass of the unit cell contents,
- V is the volume of the unit cell, and
- n is the number of phases in the analysis.

The Rietveld method relies on the assumption that all phases in the sample are crystalline and have been included in the analysis. Therefore, Equation (10.1) sums the analysed concentrations to unity. As a consequence, the Rietveld method can only calculates relative phase amounts. In the presence of any amorphous and/or unidentified crystalline phases, the analyzed crystalline weight fractions may be significantly overestimated.

Quantification of amorphous content using the Rietveld method is possible when a crystal structure can be found which adequately models the positions and relative intensities of the observable bands of a amorphous component in a diffraction pattern (e.g. Le Bail, 1995; Lutterotti et al., 1998). In this approach all components are treated as crystalline and thus can be included in the Rietveld analysis.

A typical input segment for quantitative Rietveld analysis is as follows:

```
xdd...
   str...
      scale @ 0.001
      MVW(0,0,0)        ' ZM - V - Weight
   str...
      scale @ 0.001
      MVW(0,0,0)        ' ZM - V - Weight
   ...
```

Note, that ZM is designated as "M" in the MVW macro.

10.2.2 Internal standard method

The presence of a known weight fraction of (usually) an internal standard material in the sample allows concentrations reported by the Rietveld method to be corrected proportionately according to:

$$Corr(W_\alpha) = W_\alpha \frac{STD_{known}}{STD_{measured}} \tag{10.2}$$

where

- Corr(W_α) is the corrected weight percent
- STD_{known} is the weighed concentration of the standard in the sample, and
- $STD_{measured}$ is the analysed concentration derived from Equation (10.1).

Once the corrected concentrations have been calculated, the amount of amorphous material $W_{amorphous}$ can then be derived from:-

$$W_{amorphous} = 1 - \sum_{j=1}^{n} Corr(W_j) \tag{10.3}$$

In this approach absolute amounts of the crystalline phases are first estimated and then the amount of amorphous content is determined by difference.

A typical input segment for quantitative Rietveld analysis using an internal standard is as follows:

```
xdd...
   weight_percent_amorphous 0
   str...
      scale @ 0.001
      spiked_phase_measured_weight_percent 30   ' known weight = 30%
      corrected_weight_percent 0
      MVW(0,0,0)
   str...
      scale @ 0.001
      corrected_weight_percent 0
      MVW(0,0,0)
   ...
```

Note, that only one phase can be flagged as an internal standard using the keyword spiked_phase_measured_weight_percent. Section 10.3.2.4 outlines how weight percents can be constrained if the concentration of more than one phase is known.

10.2.3 External standard method

The external standard method (O'Connor and Raven, 1988) is an indirect method that closely follows the approach in the Internal Standard Method in that it attempts to put the determined crystalline components on an absolute scale and derives the amorphous content by difference. An external standard - either a pure material or a mixture in which the chosen standard is present in known quantity - is used to determine a normalization constant K which allows the calculated weight percentages to be placed on an absolute scale,

$$W_\alpha = \frac{S_\alpha (ZMV)_\alpha \cdot \mu_m^*}{K} \tag{10.4}$$

where μ_m^* is the mass absorption coefficient of the entire sample. In contrast to the internal standard method the sample is not contaminated.

K depends only on the instrumental conditions and is independent of individual phase and overall sample-related parameters. Therefore, a single measurement is sufficient to determine K for a given instrumental configuration. As good laboratory practices

require regular instrument monitoring (alignment, tube aging) anyway, the choice of an appropriate standard enables to put all QPA results on an absolute basis and thus enables the calculation of the amorphous fraction, without any additional experimental effort.

A typical input segment for quantitative Rietveld analysis using an internal standard using the K_Factor_MAC_K and K_Factor_WP macros is as follows:

```
xdd...
   K_Factor_MAC_K(49.03, 168.355, 0)      ' µm*, K, amorphous content
   str...
      scale @ 0.001
      K_Factor_WP(0)                      ' corrected weight percent
   str...
      scale @ 0.001
      K_Factor_WP(0)                      ' corrected weight percent
   ...
```

10.2.4 PONKCS method

In the PONKCS method (Scarlett & Madsen, 2006), phases with Partial Or No Known Crystal Structure are characterized by measured rather than calculated structure factors. Most importantly, this overcomes the requirement of the traditional Rietveld method that the crystal structures of all phases must be known. Intensity contributions of crystalline as well as amorphous phases to the diffraction pattern may be characterized via single line or Pawley or Le Bail fitting methods.

The PONKCS method follows the same general form as that used in the Rietveld method, but now includes all crystalline and amorphous phases characterized by either calculated or empirical structure factors. However, for all such "PONKCS phases" calibration constants must be derived to substitute ZMV in Equation (10.1). This can be achieved by using a known mixture of the unknown (crystalline or amorphous) phase α and an internal standard s:

$$(ZM)_\alpha = \frac{W_\alpha}{W_s}\frac{S_s}{S_\alpha}\frac{(ZMV)_s}{V_\alpha} \tag{10.5}$$

Note that the obtained $(ZMV)_\alpha$ value has no physical meaning. Where Pawley or Le Bail fitting is used the unit cell volume V of the unknown is calculated during refinement. When V is not known then V requires an input of unity.

A typical input segment for a PONKCS refinement is as follows:

```
xdd...
   hkl_I...                          ' PONKCS phase 1, Pawley refinement
      hkl_m_d_th2 ... I 22.3         ' Peak list,
      hkl_m_d_th2 ... I 69.5         ' intensities must be kept fix
      hkl_m_d_th2 ... I 32.9
      ...
      scale @ 0.001                  ' scale must be refined
      MVW(10.2368, 254.7333024, 0)   ' empirical ZM - refined V - Weight

   hkl_I...                          ' PONKCS phase 2, Pawley refinement
      hkl_m_d_th2 ... I 17.1         ' Peak list,
      hkl_m_d_th2 ... I 78.4         ' intensities must be kept fix
      hkl_m_d_th2 ... I 78.4
      ...
      scale @ 0.001                  ' scale must be refined
      MVW(20.3905, 112.9737223, 0)   ' empirical ZM - refined V - Weight
   ...
```

A one time calibration per PONKCS phase with a single standard mixture is usually sufficient. When ZMVs have been calibrated for all phases, microabsorption can be reduced as this aberration is included in the calibration.

10.2.5 FULLPAT method

The FULLPAT method (Chipera & Bish, 2002) follows the same general form as that used in the PONKCS method, but uses observed diffraction data taken from reference samples. Functionality is achieved the pattern scaling method (section 11.6) and the *dummy_str* keyword.

Example usage:

```
dummy_str
   a = 1; b = 1; c = 1;                  ' dummy lattice parameters
   MVW( 298.116, 1.000, 0)
   scale cor_scale 0.1
   prm cor_offset_x 0                    ' zero point error, optional
   user_y cor_scan {#include SOME_FILE}
   fit_obj !f1 = cor_scale * (cor_scan);
      fo_transform_X = X + cor_offset_x;
```

dummy_str's are used to represent the quantitative amount of a phase, the number of *dummy_str*'s is not limited. M is to be determined in the same way as for PONKCS phases. Dummy lattice parameters are required because of technical reasons and can be arbitrary, but must be consistent to the value given for V. Lattice parameters and thus V will not be refined. It is suggested to always use the value 1 for both for the sake of simplicity.

The FULLPAT method should be regarded as a last resort approach only. Most significantly, potentially changing phase and sample properties such as lattice parameters, microstructural parameters, and preferred orientation cannot be refined. Furthermore, background subtraction is required for all reference data sets. The PONKCS method has none of these issues inherent to it and is thus recommended as the superior performer.

10.2.6 Degree of crystallinity method

The degree of crystallinity method is based on the estimation of the total intensity or area contributed to the overall diffraction pattern by each component in the analysis. The degree of crystallinity (in literature also referred to as "crystalline index"), DOC, is then calculated from the total areas under the defined crystalline and amorphous components from

$$\mathrm{DOC} = \frac{\text{Crystalline Area}}{\text{Crystalline Area} + \text{Amorphous Area}} \tag{10.6}$$

The weight fraction of the amorphous material, $W_{amorphous}$, can be calculated from

$$W_{amorphous} = 1 - \mathrm{DOC} \tag{10.7}$$

The method does not require application of any calibration constants, if the chemical composition of the crystalline phase(s) is identical to that of the whole sample. When this is not the case then an additional calibration step (see section 10.2.7) is required to obtain absolute phase amounts.

Although originally intended for quantitative analysis of amorphous content, the method can be also used to determine the ratio between 2 crystalline components.

Example usage:

```
xdd ...
      crystalline_area 0
      amorphous_area 0
      degree_of_crystallinity 0
      str...
            numerical_area 0
      hkl_I...
            numerical_area 0
      xo_I...
            numerical_area 0
            amorphous_phase
```

where the xo_I phase is flagged as the amorphous phase. More than one phase can be flagged as amorphous.

10.2.7 Calibration method

Calibration relies on the availability of a suite of standard samples from which a calibration curve can be determined. In TOPAS such a calibration curve can be used to obtain concentrations directly from any intensity estimates obtained via single line up to whole pattern fitting (traditional "single" peak approach), or be used to correct concentrations obtained by any of the QPA methods discussed above for systematic errors, most notably microabsorption.

The general procedure, which can be applied to one or more phases, is:

1. Preparation of a series of standards containing the phase of interest at known concentrations

2. Determination of a measure of the phase's intensity which may be related to its concentration
3. Generation of a suited calibration function, e.g.

$$W = A*I + B \tag{10.8}$$

where *W* is the fraction of the phase of interest, *I* is the measure of the intensity of this phase (usually in terms of a peak intensity value or a scale factor), and A and B are the coefficients of the calibration function.

The calibration method requires access to well-defined materials for preparation of standards, and are thus only applicable to mixtures similar to the calibration suite. Neither sample properties nor the instrument setup must change between the calibration and measurements of the unknown. Furthermore tube ageing must be monitored, otherwise the calibration function is invalidated.

Example usage:

```
' Rietveld method:
prm my_calibrated_weight_percent = Get(weight_percent) * ... ;:0
' Degree of crystallinity
prm my_calibrated_DOC = Get(degree_of_crystallinity) * ... ;:0
```

and so on, where "..." represents the calibration function to be applied.

10.3 Associated features

10.3.1 Elemental composition calculation

The *xdd* dependent keyword *elemental_composition* reports the total elemental composition over all *str* phases, for example:

Before Refinement:

```
xdd...
   elemental_composition
```

After Refinement:

```
xdd...
   elemental_composition
      {
                     Rietveld
      AL             0.875_0.021
      O             26.135_0.009
      SI             0.090_0.003
      Y              6.289_0.012
      ZR            66.612_0.029
      }
```

10.3.2 Constraints and restraints

Quantitative phase analysis is an application area, which is particularly characterized by (peak) overlap problems resulting in difficulties to distinguish between individaul crystalline and amorphous phase intensities, instrument and sample holder contributions, and sample fluorescense.

10.3.2.1 Convolution based profile fitting and instrument functions

Convolution based profile fitting and instrument functions are powerful tools to deal with overlapping peaks originating from well crystalline phases.

With convolution based profile fitting, the number of refinable profile parameters required is generally smaller compared to classic analytical profile fitting approaches. This is particularly the case, if an instrument function approach is used, as the instrument function constrains the instrumental part of the observed line profile shapes.

As a consequence many problems related to over-parameterization such as refinement of redundant parameters and parameter correlations can be effectively reduced, with a major impact on quantitative phase analysis of complex mixtures (Kern, 2008).

The use of an instrument function constraint drastically reduces the number of refineable profile parameters required, resulting in,

- improved refinement stability,
- reduced parameter correlations, and therefore
- more physically meaningful refinement results,

In all tutorial examples instrument function constraints are being used.

10.3.2.2 Measured patterns

The differentiation between intensities contributed by crystalline phase(s), amorphous phase(s), sample holder, and the instrument is even more difficult than just dealing with overlapping peaks from crystalline phases alone. An elegant solution to this problem is often possible, if the contributions from amorphous phase(s), and/or sample holder, and/or the instrument can be measured individually and used as a constraint. This can be done via pattern scaling (10.2.5) or the PONKCS method (10.2.4), whereby the latter is generally recommended.

10.3.2.3 Refinement of occupancy factors - Vegard's law

Refinement of accurate occupancy factors of a phase may be achieved by applying Vegard's law. This requires knowledge of the relation between the constituent elements and their associated lattice parameters and / or the unit cell volume (preferable for lower symmetric structures).

The simplest mathematical expression for Vegard's law for a binary solid solution A-B is:

$$a = a_0^{A}(1 - X) + a_0^{B}(X) \qquad (10.9)$$

where a is the lattice parameter, X is the mole fraction of component B and a_0^A and a_0^B are the lattice parameters of the pure components A and B, respectively.

As an example, according to Schulze (1984), the regression for the goethite b lattice parameter can be formulated as

$$\text{mole\% Al} = 1730 - 572\,b$$

and thus

```
str
   phase_name Goethite
   space_group 62
   a a_goethite 9.9000
   b b_goethite 2.9751
   c c_goethite 4.5624
   prm xal = (1730 - 572 * b_goethite) / 100;        ' Vegard's law
   site s1 x 0.3539 y 0.25 z 0.4523
        occ FE+3 =1-xal; beq 0.4
        occ AL+3 =xal;   beq 1
...
```

10.3.2.4 Constrained phase composition

To constrain a weight percent the following can be used:

```
str...
      Known_Weight_Percent(20)
      MVW( 0, 0, 0)
...
```

where the Known_Weight_Percent macro is defined as

```
macro Known_Weight_Percent(& w)
   {
   scale = (w/(100-w)) Get(sum_smvs_minus_this) /
           (Get(cell_mass) Get(cell_volume));
   }
```

Weight percents of more than one phase can be constrained.

10.3.2.5 Restrained phase composition

To restrain a weight percent the following can be used:

```
xdd...
   penalties_weighting_K1 .2
   restraint = (Cubic_Zirconia_wt_percent - 36); : 0
   str...
      MVW(0,0, !Cubic_Zirconia_wt_percent 0)
```

Weight percents of more than one phase can be restrained.

10.3.2.6 Constrained elemental composition

If for example an elemental weight percent was known and three phases of the mixture comprised this element then Get_Element_Weight can be used to get the weight of the element as a function of the structure; i.e.

```
prm !known_Zr 65
str...
   prm z1 = Get_Element_Weight(Zr);
   MVW(!m1 0, !v1 0,0)
str...
   scale s2 0.001
   prm z2 = Get_Element_Weight(Zr);
   MVW(0, !v2 0,0)
str...
   scale s3 0.001
   prm z3 = Get_Element_Weight(Zr);
   MVW(0, !v3 0,0)
```

Rearranging the formulae for element weight percent, the scale parameter of one of the phases, say the first one, can be written as follows:

```
scale = (0.01 known_Zr Get(sum_smvs_minus_this) - s2 v2 z2 - s3 v3 z3) /
        (v1 (z1 - 0.01 known_Zr m1));
```

Get(*sum_smvs_minus_this*) returns the sum of SMVs minus the phase where it is defined.

10.3.2.7 Restrained elemental composition

element_weight_percent $ELEMENT $NAME # is an xdd dependent keyword that returns the weight percent of an element within the corresponding str's of the xdd. Example usage:

Before Refinement:

```
penalties_weighting_K1 .1
xdd...
   element_weight_percent Zr+4 zr 0
   restraint = (zr - 65); :  0
```

After Refinement:

```
penalties_weighting_K1 .1
xdd...
   element_weight_percent Zr+4 zr 65.0275252`
   restraint = (zr - 65);: 0.0275251892`
```

In this example "zr" is the name given to the element Zr+4 and the restraint shows a known value of 65 (set for example by XRF results). The refinement obeys the restraint according to the value set for *penalties_weighting_K1*.

10.3.3 Removing Phases during a refinement

The *remove_phase* keyword, used by the Remove_Phase macro, allows for phase removal during refinement. Typical use is as follows:

```
str...
   remove_phase 0.3
```

This phase will be removed if its weight percent is below 0.3%.

```
for strs {
   Remove_Phase(0.3, 1)
}
```

Here all phases belonging to an *xdd* are removed if their weight percent is below 0.3% and if the error in the weight percent is greater than 1%.

The phase removal process is executed at the end of a Cycle. Text similar to the following is displayed on removal of a phase:

```
*** Deleting phase: Corundum ***
```

Refinement is terminated when no phase is removed during a cycle.

10.3.4 Brindley correction of microabsorption effects

The Brindley model for correction of microabsorption effects (Brindley, 1945) is supported via the Apply_Brindley_Spherical_R_PD macro and can be used to partially correct for microabsorption effects.

Note, that the Brindley correction has only limited applicability. Most significantly, the model assumes that all phases in a sample consist of spherical particles of uniform size, which is a totally unrealistic assumption. As a consequence, applying the correction will frequently decrease, rather than improve, accuracy.

Usage of the Brindley correction is strongly discouraged, instead data should be collected using a finely grinded powder with an appropriate wavelength to minimize microabsorption.

When microabsorption effects cannot be avoided, calibration based methods should be preferred over the Brindley model.

11 MISCELLANOUS

11.1 Calculated powder patterns

To calculate a powder pattern, the keyword *yobs_eqn* is used in place of the xdd keyword as follows:

```
iters 0
yobs_eqn !calc_pattern.xy = X;
      min 12
      max 160
      del .01
Out_X_Ycalc(My_Calculated_Pattern.XY)
```

The macro Out_X_Ycalc can be used to save the calculated pattern as a file.

11.2 Conditional loading of INP files

An "if" construct can be used for conditional loading of INP files operating on the pre-processed INP file. The syntax is as follows:

```
if expression {…}
else if expression {…}
else expression {…}
```

*"*expression*"* can be any valid TOPAS equation without the semicolon; in addition expression can contain the functions Prm_There(prm_name) and Obj_There(obj_name).

The following is equivalent to a /* */ block comment:

```
if 0 {
…
}
```

11.3 Large refinements with tens of 1000s of parameters

Refinements comprising many parameters and data points can be both slow and memory intensive. Computation speed is hindered by the **A** matrix dot products and in the case of dense matrices memory usage in forming the full **A** matrix can be prohibitive. The following keywords can be used to overcome these problems:

```
conserve_memory

bootstrap_errors 100

approximate_A
      A_matrix_memory_allowed_in_Mbytes 100
      A_matrix_elements_tollerance 0.00001
```

The *approximate_A* keyword avoids the calculation of the **A** matrix dot products. Typically more refinement iterations are required for convergence but in most large

problems the time to convergence is greatly decreased. Furthermore memory usage of the **A** matrix can be limited using *A_matrix_memory_allowed_in_Mbytes*; this produces a sparse matrix, dependening on allotted memory, by removing small A_{ij} values.

Typically the calculation of the covariance matrix is impractical and hence errors can instead be determined using the bootstrap method.

11.4 Lorentz-Polarisation

11.4.1 Predefined Lorentz-Polarisation macros

11.4.1.1 GUI and Launch Mode:

```
macro LP_Factor(v) { LP_Factor(,v) }


macro LP_Factor(c, v)
{
   #m_argu c
   If_Prm_Eqn_Rpt(c, v, min .0001 max 90)
      scale_pks = (1 + Cos(CeV(c,v) Deg)^2 Cos(2 Th)^2) /
                  (Sin(Th)^2 Cos(Th));
}
```

The LP_Factor macro is applied by the "LP factor" correction in GUI mode. The following polarisation values apply:

Synchrotron use	: 90
Neutron use	: 90
No monochromator use	: 0

Monochromator use (most common monochromators, Cu radiation):

Ge	: 27.3
Graphite	: 26.4
Quartz	: 26.6

11.4.1.2 Launch Mode:

```
macro Lorentz_Factor
{
   scale_pks = 1 / (Sin(Th)^2 Cos(Th));
}
```

```
macro LP_Factor_Synchrotron_Simple
{
   Lorentz_Factor
}


macro LP_Factor_Synchrotron(pp, ppv, mono, monov)
/*
   By Ian Madsen, CSIRO Minerals, Australia
   pp is the polarisation in the plane of the synchrotron
   pp = 1.0 for circularly polarised X-rays
        (i.e. laboratory X-ray tubes)
   pp = 0.0 for fully polarised X-rays
        (ideal synchrotron source)
         expect pp ~ 0.05 for 'real' synchrotron source
   mono = the 2theta diffraction angle of a
          crystal monochromator
   NOTE: pp and mono will be ~100% correlated -
         do not attempt to refine both together!
   Last modification - 12/02/2008
*/
{
   #m_argu pp
   #m_argu mono
   If_Prm_Eqn_Rpt(pp, ppv, min 0.0000001 max 1.0)
   If_Prm_Eqn_Rpt(mono, monov, min 0.0000001 max 90.0)
   scale_pks = (1/( Sin(Th)^2 Cos(Th))) * ((1 + CeV(pp,ppv) *
   Cos(CeV(mono,monov) Deg)^2 Cos(2 Th)^2)/(1 + CeV(pp,ppv) *
   Cos(CeV(mono,monov) Deg)^2 ));
}
```

11.4.2 Background information

In the following some mathematical background information as well as the relationship of the TOPAS polariation corrections to those in GSAS and FullProf is discussed (Evans, 2008).

11.4.2.1 Lorentz-Polarisation Corrections in TOPAS

TOPAS uses:

$$LP\_Factor = \frac{1+\cos^2 2\theta \cos^2 2\theta_M}{\cos\theta \sin^2 \theta} \tag{11.1}$$

as defined in the LP_Factor macro. This comes from the Lorentz factor:

$$L = \frac{1}{\sin\theta \sin 2\theta} = \frac{1}{\cos\theta \sin^2 \theta} \tag{11.2}$$

as defined in the Lorentz_factor macro, and polarisation with a monochromator:

$$P = \frac{1-K+K\cos^2 2\theta \cos^2 2\theta_M}{2} \tag{11.3}$$

where K is fractional polarisation of the beam.

For neutrons K = 0 and the LP expression becomes:

$$LP = \frac{1}{\cos\theta \sin^2\theta} \tag{11.4}$$

In expression (11.1) $2\theta_M = 90$ reduces to (11.4). This is the same as Lorentz factor (11.2).

With no monochromator and unpolarised source K = 0.5 and the LP expression becomes:

$$LP = \frac{0.5 + 0.5\cos^2 2\theta}{2\cos\theta \sin^2\theta} \tag{11.5}$$

Give or take a scale factor using $2\theta_M = 0$ in (11.1) reduces to (11.5).

For synchrotrons the radiation is typically assumed to be 100% plane polarised and that the analyser crystals have no effect on the vertical electric vector which means K=0 and one can therefore "pretend" to have the neutron situation and use $2\theta_M = 90$ or expression (11.4). This is an approximation of a "real" situation where K is typically a small number.

The macro LP_Factor_Synchrotron (Madsen, 2008) in TOPAS is:

$$LP = \frac{1}{2\cos\theta \sin^2\theta} \frac{1 - pp + pp\cos^2 2\theta \cos^2 2\theta_M}{1 + pp\cos^2 2\theta_M} \tag{11.6}$$

where pp = 0.5 for laboratory X-ray tubes with circularly polarised X-rays. The term on the bottom right is a constant and the equation reduces to:

$$LP = c\left(\frac{0.5 + 0.5\cos^2 2\theta \cos^2 2\theta_M}{2\cos\theta \sin^2\theta}\right) \tag{11.7}$$

which is the same as (11.1), give or take a scale factor.

Use of pp = 0 for fully polarised synchrotron reduces to:

$$LP = \frac{1}{2\cos\theta \sin^2\theta} \tag{11.8}$$

which is again the same as $2\theta_M = 90$ in (11.1).

For a real synchrotron pp=0.05 and the expression becomes:

$$LP = c\left(\frac{0.95 + 0.05\cos^2 2\theta \cos^2 2\theta_M}{2\cos\theta \sin^2\theta}\right) \tag{11.9}$$

This is not significantly different from expression (11.4) in real situations.

11.4.2.2 Relationship to GSAS

In GSAS-speak there are three equations available:

IPOL = 0: $\frac{Ph+(1-Ph)\cos^2 2\theta}{2\sin^2\theta\cos\theta}$ (11.10)

IPOL = 1: $\frac{1+Ph\cos^2 2\theta}{\sin^2\theta\cos\theta}$ (11.11)

IPOL = 2: $\frac{1+Ph\cos^2 2\theta}{\left(1+\cos^2 2\theta\right)\sin^2\theta\cos\theta}$ (11.12)

For laboratory diffractometers with 26.6 monochromator angle users typically use IPOL = 0 and Ph = 0.555 or IPOL = 1 and Ph = 0.8. Putting $2\theta_M$ = 26.6 into (11.1) gives:

$$LP\_Factor = \frac{1+\cos^2 2\theta\times 0.8}{\cos\theta\sin^2\theta} = \frac{0.5+\cos^2 2\theta\times 0.4}{2\cos\theta\sin^2\theta} = c\left(\frac{0.555+0.444\times\cos^2 2\theta}{2\cos\theta\sin^2\theta}\right)$$

i.e. the GSAS IPOL = 0 equation with Ph of 0.555. Or if one takes the last equation and divides through by 0.555 one gets:

$$c\left(\frac{0.555+0.444\times\cos^2 2\theta}{2\cos\theta\sin^2\theta}\right) = \frac{c}{0.555}\left(\frac{1+0.8\times\cos^2 2\theta}{\cos\theta\sin^2\theta}\right) \quad (11.13)$$

which is the gsas IPOL = 1 equation.

11.4.2.3 Relationship to FullProf

Fullprof uses:

$$P = \frac{1-K+K\cos^2 2\theta\cos^2 2\theta_M}{2\sin^2\theta\cos\theta} \quad (11.14)$$

For neutrons the manual says "K is ignored" but actually K = 0 is effectively used.

For characteristic X-rays (unpolarized beam) the formula is:

$$P = \frac{1+\cos^2 2\theta\cos^2 2\theta_M}{2\sin^2\theta\cos\theta}$$

i.e. K = 0.5 in the general formula multiplied by 2.

For synchrotrons K must be given and is ~ 0.1.

11.4.3 Alternate Lorentz Polarization factor definition

Traditionally, Lorentz-Polarisation corrections are applied to the integrated intensity of a phase peak at the calculated peak position. In TOPAS this is accomplished by the *scale_pks* keyword. Alternatively, for higher accuracy, the keyword *scale_phase_X* (section 11.7) can be used to scale phases point by point, e.g. by using the following macro:

```
macro LP_Factor_X(c, v)
{
   #m_argu c
   If_Prm_Eqn_Rpt(c, v, min .0001 max 90)
   local #m_unique th = X Pi / 360;
   scale_phase_X = (1 + Cos(c Deg)^2 Cos(2 th)^2)/(Sin(th)^2 Cos(th));
}
```

This correction has its largest effects for broad peaks and is thus relevant for accurate microstructure analysis.

11.5 Anisotropic refinement models

Keywords that can be a function of H, K, L and M, as shown in Table 11.1, allow for the refinement of anisotropic models to handle preferred orientation, anisotropic line broadening, anisotropic peak shifts, etc.

Table 11.1: Keywords that can be a function of H, K, L, M, Xo, Th and D_spacing.

lor_fwhm	*stacked_hats_conv*	*pv_lor, pv_fwhm*
gauss_fwhm	*user_defined_convolution*	*ymin_on_ymax*
hat	*th2_offset*	*la, lo, lh, lg*
one_on_x_conv	*scale_pks*	*phase_out*
exp_conv_const	*h1, h2, m1, m2*	*scale_top_peak*
circles_conv	*spv_h1, spv_h2, spv_l1, spv_l2*	*pk_xo*

An important consideration when dealing with hkls in equations is whether to work with hkls or whether to work with their multiplicities. The Multiplicities_Sum macro can be used when working with multiplicities, for example:

```
prm a 0
th2_offset = Multiplicities_Sum( If(Mod(L,2)==0, a Tan(Th), 0) );
```

L here corresponds to the L's of the multiplicities. Note, the preferred orientation macro PO uses the Multiplicities_Sum macro and Spherical Harmonics uses the hkls in the *.hkl file only.

A completely different viewpoint than to refine on half widths is to consider the distribution of lattice metric parameters within a sample. Each crystallite is regarded as having its own lattice parameters, with a multi-dimensional distribution throughout the powder sample. This can be achieved by adding the same structure several times to the input file.

In the following several selected anisotropic refinement models are discussed which have been shown to be adequately describe preferred orientation, anisotropic line profile broadening, and anisotropic peak shifts:

- Second rank tensors (Le Bail & Jouanneaux, 1997):
  - Anisotropic line broadening
- Stephens model (Stephens, 1999)
  - Anisotropic line broadening
- Spherical harmonics functions (Järvinen, 1993)
  - Preferred orientation, anisotropic line broadening
- March-Dollase model (March, 1932)
  - Preferred orientation, anisotropic line broadening
- Simple IF-THEN constructions
  - Anisotropic line broadening, anisotropic peak shifts

Users are encouraged to try these models also for other anisotropic quantities as listed in Table 11.1, or to develop different models.

11.5.1 Second rank tensors

A qualitative account for anisotropic line broadening in profile analysis has been given by Le Bail & Jouanneaux (1997) based on the Cagliotti relation (Cagliotti et al., 1958). A macro applying this model could look like the following:

```
macro LeBail_Jouanneaux_1997(
   uc11,uv11,uc22,uv22,
   uc33,uv33,uc23,uv23,
   uc13,uv13,uc12,uv12,
   vc11,vv11,...)
{
   prm u 0    min -1 max 2
   prm v 0    min -1 max 1
   prm w 0.01 min -1 max 2
   prm uc11 uv11   prm uc22 uv22   prm uc33 uv33
...
u = (
     H^2   A_star A_star uc11 +
...
v = (
     H^2   A_star A_star vc11 +
...
w = ...
gauss_fwhm = u Tan(Th)^2 + v Tan(Th) + w;      ' Cagliotti relation
}
```

When using this model the following symmetry conditions have to be considered:

Cubic	: 11=22=33, 12=13=23
Hexagonal Trigonal Tetragonal	: 11=22, 13=23
Orthorhombic Monoclinic Triclinic	: None

An analogous variation may also be applied to line profile shapes, so a maximum of 36 refineable parameters is obtained.

As the Cagliotti relation is a poor performer in describing half width dependence on 2θ for X-ray data, and as the extremely high number of parameters will not allow for stable and reliable refinements, the examples outlined in sections 11.5.2 throughout 11.5.5 should be preferred as a base for describing anisotropic peak broadening.

11.5.2 Stephens model

A model to describe anisotropic line broadening based on a multi-dimensional distribution of lattice metrics has been developed by Stephens (1999). This model requires a number of parameters dependent on symmetry which cannot be discussed here. Symmetry conditions have been derived for all crystal systems leading to the following macros:

```
Stephens_triclinic(...)
Stephens_monoclinic(...)
Stephens_orthorhombic(...)
Stephens_tetragonal_low(...)
Stephens_tetragonal_high(...)
Stephens_trigonal_low(...)
Stephens_trigonal_high(...)
Stephens_trigonal_high_2(...)
Stephens_hexagonal(...)
Stephens_cubic(...)
```

For details, specifically the macro parameters, refer to the macro definitions in TOPAS.INC and Stephens (1999).

The following sequence demonstrates an example implementation for the "tetragonal_high" case:

```
prm s400 0
prm s004 0
prm s220 0
prm s202 0
prm eta  0.5 min 0 max 1
prm mhkl = Abs(S400 (H^4 + K^4)+ S004 L^4 +
               S220 H^2 K^2 + S202 (H^2 L^2 + K^2 L^2));
prm pp = D_spacing^2 * Sqrt(Max(mhkl,0)) / 1000;
gauss_fwhm = 1.8/3.1415927 pp (1-eta) Tan(Th) + 0.0001;
lor_fwhm = 1.8/3.1415927 pp eta Tan(Th) + 0.0001;}
```

11.5.3 Spherical harmonics

TOPAS implements a normalized symmetrized spherical harmonics function, see Järvinen (1993).

In the case of correcting for preferred orientation as per Järvinen (1993) then the intensities of the reflections are multiplied by the series value. This is accomplished by first defining a series, e.g.:

```
str...
      spherical_harmonics_hkl sh
      sh_order 8
```

and then scaling the peak intensities:

```
      scale_pks = sh;
```

After refinement the INP file is updated with the coefficients.

Alternatively the predefined macro PO_Spherical_Harmonics can be used.

Typically the C00 coefficient is not refined as its series component Y00 is simply 1 and is 100% correlated with the scale parameter.

The series values can be written in a file as a function of hkl as follows:

```
scale_pks = sh;
phase_out sh.txt load out_record out_fmt out_eqn
{
      "%4.0f" = H;
      "%4.0f" = K;
      "%4.0f" = L;
      " %9g\n" = sh;
}
```

Note, that the value of the series can go negative resulting in negative peak intensities. This can happen, if the refinement model is simply inadequate, or due to parameter correlation if the order of the spherical harmonics is chosen too high. The series can be forced to be positive by for example using something like:

```
spherical_harmonics_hkl sh
      sh_order 8
      scale_peaks = Max(sh, 0);
```

The number of refined coefficients needs to be kept at a minimum. It is generally suggested to always start with a 2nd order spherical harmonics, and to only increase the number of orders to 4, then 6, and finally 8, if the quality of fit significantly improves (both visibly and in terms of R_{WP}). After each step it is mandatory to check all refinement parameters, specifically those parameters corrected by the spherical harmonics (intensities in case of a preferred orientation correction).

The following input sequence uses *spherical_harmonics_hkl* for describing anisotropic peak broadening applied to the Lorentzian half width:

```
str...
   prm p1 0.01 min 0.0001
   spherical_harmonics_hkl sh
      sh_order 6
      lor_fwhm = sh p1;
```

This input sequence uses *spherical_harmonics_hkl* for describing anisotropic peak asymmetry using the *exp_conv_const convolution*:

```
str...
   prm p1 0.01 min 0.0001
   spherical_harmonics_hkl   sh
      sh_order 8
   exp_conv_const = (sh-1) / Sin(Th);
```

11.5.4 March-Dollase model

To correct for preferred orientation the March (1932) relation and *str_hkl_angle* can be used, e.g.:

```
str...
   str_hkl_angle ang1 1 0 0
   prm p1 1    min 0.0001 max 2
   scale_pks = Multiplicities_Sum(((p1^2 Cos(ang1)^2 +
               Sin(ang1)^2 / p1)^(-1.5)));
```

In full analogy, the model can be also used to model anisotropic line broadening, e.g. using a Tan(θ) term to account for anisotropic microstrain broadening:

```
str...
   str_hkl_angle ang1 1 0 0
   prm p1 1    min 0.0001 max 2
   prm p2 0.01 min 0.0001 max 0.1
   lor_fwhm = p2 Tan(Th) Multiplicities_Sum(((p1^2 Cos(ang1)^2 +
              Sin(ang1)^2 / p1)^(-1.5)));
```

11.5.5 Simple IF-THEN constructions

Anisotropic line broadening as a function of L:

```
str...
   prm a 0.1 min 0.0001 max 5
   prm b 0.1 min 0.0001 max 5
   gauss_fwhm = If(L==0, a Tan(Th), b Tan(Th));
```

Anisotropic peak shifts as a function of L (*th2_offset*):

```
str...
   prm at 0.07 min 0.0001 max 1
   prm bt 0.07 min 0.0001 max 1
   th2_offset = If(L==0, at Tan(Th), bt Tan(Th));
```

11.6 Pattern scaling

The new keywords *user_y* and *fo_transform_X* provide a means to fit any kind of xy data to observed patterns such as other observed patterns or any (learnt) shapes. Typical application areas include but are not limited to fitting of reference data for quantitative phase analysis (section 10.2.5) and fitting of learnt shapes to describe background.

Example usage:

```
scale my_scale 0.0001
user_y my_shape {...}
prm my_offset_x 0
fit_obj !f1 = my_scale * (my_shape);
   fo_transform_X = X + my_offset_x;
Plot_Fit_Obj(my_shape, "My scan #")
```

The user defined my_shape corresponds to a parameter name given to the *user_y*; it can be used in all equations that can be a function of X, for example:

```
fit_obj = Exp(my_shape^2);
```

The *user_y* my_shape {…} usage allows shapes to be loaded or typed directly into the INP file, the format is defined using the "_x1_dx", "_xy", and "_xye tags". A triangle for example can be formulated in "_x1_dx" format as follows:

```
user_y my_shape
   {
   _x1_dx -1 1    ' the start x and step
   0 1 0          ' the shape data
   }
```

Alternatively *.XY and *.XYE data can be used using _xy and _xye tags; ie.

```
user_y my_shape
{
_xy
20.0000 11
20.0200 39
…
}
```

```
user_y my_shape
{
_xye
20.0000 11 1.234
20.0200 39 2.847
…
}
```

Usually it will be more desirable to read xy and xye data directly from a file; this can be achieved by, e.g.,

```
user_y my_shape { _xy #my_data.xy }
```

or

```
user_y my_shape { _xye #my_data.xye }
```

fo_transform_X is a dependent of *fit_obj* and it transforms the X used within the *fit_obj*. For example, my_scan could have an x-axis that does not match the x-axis of the Yobs pattern; *fo_transform_X provides* a means to transform the Yobs x-axis to the *user_y* x-axis.

More than one *user_y* can be defined and they can be used any number of times in equations that can be a function of X. Convergence is as fast as with any other other refinement.

11.7 Point by point scaling of calculated patterns

The *scale_phase_X* keyword scales calculated patterns (Ycalc) point by point. It can be used to define say alternate Lorentz Polarization factors. Some main points:

- Can be a function of X
- Multiple definitions are allowed and each is applied to the pattern.
- Can occur at the xdd or phase level.

Here's an example:

```
xdd...
   scale_phase_X...
   str...
      scale_phase_X...
   hkl_Is...
      scale_phase_X...
```

The first *str* is multiplied by the first and second *scale_phase_X*; the *hkl_Is* phase is multiplied by the first and third *scale_phase_X*.

11.8 Plotting

11.8.1 Plotting phases above background

By default phases are now plotted on top of background where background comprises *fit_obj*'s + *bkg*. The *xdd* dependent keyword *gui_add_bkg* and the *fit_obj* dependent *fit_obj_phase* can be used to change the defaults, for example,

```
xdd...
   gui_add_bkg !E
   fit_obj...
      fit_obj_phase !E
```

gui_add_bkg defaults to 1; if it's zero then phases are not plotted above background.

fit_obj_phase defaults to 1. If *gui_add_bkg* =1 then the following is added to phases:

bkg + (and any *fit_obj*'s that has *fit_obj_phase* =1)

11.8.2 Plotting fit_objs

fit_obj's can be plotted using the following macros:

```
macro Plot_Fit_Obj(p, name)
{
   dummy_str
   phase_name name
   scale = p;
}
```

```
macro Plot_Fit_Obj(name)
{
   dummy_str
   phase_name name
}
```

Here is an example:

```
xdd...
   fit_obj !f1 = ...
   Plot_Fit_Obj(f1, "Fit Obj")
```

Plotting is via a *dummy_str* and the scale parameter of the *dummy_str* is set to the name given to the *fit_obj*, which in this case is f1. At the plotting stage the *dummy_str* borrows the calculated pattern from the *fit_obj*.

The scale parameter of the *dummy_str* has some intelligence built into it such that if scale is not a function of a *fit_obj* name then it will search the place of the item it is a function of for a calculated pattern. For example, in the following:

```
xdd...
   Plot_Fit_Obj(a, "Fit Obj")
   fit_obj = a ...
   prm a ...
```

the "a" parameter lives locally to the *fit_obj* as it is defined after the *fit_obj*. Defining the scale parameter of the *dummy_str* in terms of "a" therefore allows the *dummy_str* to determine where to find the calculated pattern to display. In this way macros such as the PV macro can be used and plotted without having to define a name for the *fit_obj*.

Sometimes the *fit_obj* has no name and no parameter that belongs to it; instead of naming the *fit_obj* or rearranging *prm* definitions the second Plot_Fit_Obj macro can be used:

```
xdd...
   fit_obj = Plot_Fit_Obj("plot previously defined fit_obj")
```

Here the *fit_obj* defined prior to Plot_Fit_Obj is plotted.

12 KEYWORDS

12.1 Data structures

The following listing gives an overview of all keywords and keyword dependencies. Trailing "…" implies that more than one node of that type can be inserted under its parent. Items enclosed in square brackets are optional. Items beginning with a capital "T" corresponds to keyword groups analogous to complex types in XML.

Ttop

Ttop_xdd
Tglobal
Txdd
Txdd_scr
Tcomm_1
Tcomm_2
Miscellaneous
Tindexing
Tcharge_flipping

Ttop_xdd

[*convolution_step* #]
[*r_p* #][*r_p_dash* #][*r_wp* #][*r_wp_dash* #][*r_exp* #][*r_exp_dash* #][*gof* #][*weighted_Durbin_Watson* #]
[*Rp* !E][*Rs* !E]
[*x_calculation_step* !E]

Tglobal

[*A_matrix*][*C_matrix*][*A_matrix_normalized*][*C_matrix_normalized*]
[*approximate_A*]
    [*A_matrix_memory_allowed_in_Mbytes* #]
    [*A_matrix_elements_tollerance* #]
    [*A_matrix_report_on*]
[*bootstrap_errors* !Ecycles]
    [*fraction_of_yobs_to_resample* !E]
    [*resample_from_current_ycalc*]
    [*determine_values_from_samples*]
[*chi2* !E]
[*chi2_convergence_criteria* !E]
[*conserve_memory*]
[*continue_after_convergence*]
[*do_errors*]
[*do_errors_include_restraints*]
[*do_errors_include_penalties*]
[*file_name_for_best_solutions* $file]
[*iters* #]
[*line_min*][*use_extrapolation*][*no_normal_equations*][*use_LU*]
[*marquardt_constant* !E]…
[*no_LIMIT_warnings*]
[*only_penalties*]

[*out_A_matrix* $file]
    [*A_matrix_prm_filter* $filter]
[*out_prm_vals_per_iteration* $file]... I [*out_prm_vals_on_convergence* $file]...
    [*out_prm_vals_filter* $filter]
[*out_refinement_stats*]
[*out_rwp* $file]
[*penalties_weighting_K1* !E]
[*pen_weight* !E]
[*percent_zeros_before_sparse_A #*]
[*process_times*]
[[*quick_refine* !E][*quick_refine_remove* !E]]
[*randomize_on_errors*]
[*restraint* !E]
[*save_best_chi2*]
[*seed*]
[*temperature* !E]...
    [*move_to_the_next_temperature_regardless_of_the_change_in_rwp*]
    [*save_values_as_best_after_randomization*]
    [*use_best_values*]
[*use_tube_dispersion_coefficients*]
[*verbose #*]

Txdd

[*xdd* $file [{$data}][*range #*][*xye_format*][*gsas_format*][*fullprof_format*]]...
    Tcomm_1
    Tcomm_2
    Tmin_max_r
    Ttop_xdd
    Txdd_comm1
    [*elemental_composition*]
    [*element_weight_percent* $atom $Name #]…
    [*element_weight_percent_known* $atom #]…
    [*gui_add_bkg* !E]
    [*mixture_density_g_on_cm3* !E]
    [*mixture_MAC* !E]
    [*scale_phase_X* E]
    [*weight_percent_amorphous* !E]
    [*xdd_sum* !E]
    [*xo_Is*]...
        Tcomm_1_2_phase_1_2
        Tlebail
        [*xo* E I E]...
    [*d_Is*]...
        Tcomm_1_2_phase_1_2
        Tlebail
        [*d* E I E]...
    [*hkl_Is*]...
        Tcomm_1_2_phase_1_2
        Thkl_lat
        Tlebail
        Tspace_group
        [*hkl_m_d_th2* # # # # # # E]...
        [*I_parameter_names_have_hkl* $start_of_parameter_name]
        [*lp_search* !E]
        [*scale_phase_X* E]

[*str* | *dummy_str*]...
Tcomm_1_2_phase_1_2
Thkl_lat
Tmin_max_r
Trigid
Tspace_group
Tstr_details
[*scale_phase_X* E]
[*user_y* $Name { #include $Some_file }]…

Tcomm_1_2_phase_1_2

Tcomm_1
Tcomm_2
Tphase_1
Tphase_2

Txdd_scr

[*xdd_scr* $file] …
Tcomm_2
Tmin_max_r
Ttop_xdd
Txdd_comm_1
[*dont_merge_equivalent_reflections*]
[*dont_merge_Friedel_pairs*]
[*ignore_differences_in_Friedel_pairs*]
[*str*]...
Tcomm_2
Thkl_lat
Tmin_max_r
Tphase_1
Trigid
Tspace_group
Tscr_1
Tstr_details

Tscr_1

[*Flack* E]
[*i_on_error_ratio_tolerance* #]
[*num_highest_I_values_to_keep* #]

Txdd_comm_1

[*bkg* [@] # # #...]
[*degree_of_crystallinity* #]
[*crystalline_area* #] [*amorphous_area* #]
[*d_spacing_to_energy_in_eV_for_f1_f11* !E]
[*exclude* #ex1 #ex2]...
[*extra_X_left* !E][*extra_X_right* !E]
[*fit_obj* E [*min_X* !E][*max_X* !E]]...
[*fit_obj_phase* !E]
[*fo_transform_X* !E]
[*neutron_data*]
[*rebin_with_dx_of* !E]
[*smooth* #]
[*start_X* #][*finish_X* #]
[*weighting* !E [*recal_weighting_on_iter*]]
[*xdd_out* $file [*append*]]...
Toutrecord
[*yobs_eqn* !N E]
[*yobs_out* $file][*ycalc_out* $file][*diff_out* $file]
[*yobs_to_xo_posn_yobs* !E]

Tcomm_1

[*axial_conv*]...
    filament_length E *sample_length* E *receiving_slit_length* E
    [*primary_soller_angle* E][*secondary_soller_angle* E]
    [*axial_n_beta* !E]
[*capillary_diameter_mm* E]...
    [*capillary_u_cm_inv* E]
    [*capillary_parallel_beam*][*capillary_divergent_beam*]
[*circles_conv* E]...
[*exp_conv_const* E [*exp_limit* E]]...
[*ft_conv* !E]...
    [*ft_min* !E]
    [*ft_x_axis_range* !E]
    Get(ft_0)
    FT_Break
[*gauss_fwhm* E]...
[*h1* E *h2* E *m1* E *m2* E]
[*hat* E [*num_hats #*]]...
[*lor_fwhm* E]...
[*lpsd_th2_angular_range_degrees* E]...
    lpsd_equitorial_divergence_degrees E
    lpsd_equitorial_sample_length_mm E
        [*lpsd_beam_spill_correct_intensity* !E]
[*modify_peak*]
    [*modify_peak_eqn* !E]
    [*modify_peak_apply_before_convolutions*]
    [*current_peak_min_x* !E]
    [*current_peak_max_x* !E]
[*one_on_x_conv* E]...
[*pk_xo* E]
[*push_peak*]… [*bring_2nd_peak_to_top*]… [*add_pop_1st_2nd_peak*]… [*scale_top_peak* E]…
[*pv_lor* E *pv_fwhm* E]
[*spv_h1* E *spv_h2* E *spv_l1* E *spv_l2* E]
[*stacked_hats_conv*]...
    [*whole_hat* E [*hat_height* E]]...
    [*half_hat* E [*hat_height* E]]...
[*th2_offset* E]...
[*user_defined_convolution* E *min* E *max* E]...
[*WPPM_ft_conv* E]…
    WPPM_L_max E
    WPPM_th2_range E
    [*WPPM_correct_Is*]

Tcomm_2

[*f0_f1_f11_atom*]…
    [*f0* E] [*f1* E] [*f11* E]
[*existing_prm* E]
[*lam* [*ymin_on_ymax #*][*no_th_dependence*][*Lam* !E] [*calculate_Lam*]]
    [*la* E *lo* E *lh* E *lg* E]...]
[*scale_pks* E]...
[*prm* | *local* E [*min* !E][*max* !E][*del* !E][*update* !E][*stop_when* !E][*val_on_continue* !E]]...
[*penalty* !E]...
[*out* $file [*append*]]...
    Toutrecord

Tphase_1

[*atom_out* $file [*append*]]…
    Toutrecord
[*auto_scale* !E]
[*brindley_spherical_r_cm* !E]
[*cell_mass* !E][*cell_volume* !E][*weight_percent* !E]
    [*spiked_phase_measured_weight_percent* !E][*corrected_weight_percent* !E]
[*del_approx* !E]
[*phase_MAC* !E]
[*phase_name* $phase_name]
[*phase_out* $file [*append*]]...
    Toutrecord
[*r_bragg #*]
[*remove_phase* !E]
[*scale* E]

Tphase_2

[*amorphous_phase*]
[*numerical_area* E]
[*peak_buffer_step* E [*report_on*]]
[*peak_type* $type]

Tlebail

[*lebail #*]

Tstr_details

[*append_cartesian*][*append_fractional* [*in_str_format*]]
[*append_bond_lengths* [*consider_lattice_parameters*]]
[*atomic_interaction* N E] | [*ai_anti_bump* N]...
    ai_sites_1 $sites_1 *ai_sites_2* $sites_2
    [*ai_no_self_interation*]
    [*ai_closest_N* !E]
    [*ai_radius* !E]
    [*ai_exclude_eq_0*]
    [*ai_only_eq_0*]
[*box_interaction* [*from_N #*][*to_N #*][*no_self_interaction*] $site_1 $site_2 N E]...
[*cloud* $sites]...
    [*cloud_population* !E]
    [*cloud_save* $file]
    [*cloud_save_xyzs* $file]
    [*cloud_load_xyzs* $file]
        [*cloud_load_xyzs_omit_rwps* !E]
    [*cloud_formation_omit_rwps* !E]
    [*cloud_try_accept* !E]
    [*cloud_gauss_fwhm* !E]
    [*cloud_extract_and_save_xyzs* $file]
        [*cloud_number_to_extract* !E]
        [*cloud_atomic_separation* !E]
[*fourier_map* !E]
    [*fourier_map_formula* !E]
    [*extend_calculated_sphere_to* !E]
    [*min_grid_spacing* !E]
    [*correct_for_atomic_scattering_factors* !E]
    [*f_atom_type* $type [*f_atom_quantity* !E]]…
[*grs_interaction* [*from_N #*][*to_N #*][*no_self_interaction*] $site_1 $site_2 *qi #* *qj #* N E]...
[*hkl_plane* $hkl]…
[*mag_only_for_mag_sites*]
[*mag_space_group* $symbol]
[*no_f11*]

[*normalize_FCs*]
[*occ_merge* $sites [*occ_merge_radius* !E]]...
[*report_on_str*]
[*site* $site_name]...
    Tmin_r_max_r
    [*x* E][*y* E][*z* E]
    [*num_posns* #][*rand_xyz* !E][*inter* !E]
    [*occ* $atom E *beq* E]...
    [*adps*][*u11* E][*u22* E][*u33* E][*u12* E][*u13* E][*u23* E]
    [*layer* $layer_name]
    [*mlx* E] [*mly* E] [*mlz* E] [*mg* E]
    [*mag_only*]
    [*num_posns* #][*rand_xyz* !E][*inter* !E]
[*sites_distance* N] | [*sites_angle* N] | [*sites_flatten* N [*sites_flatten_tol* !E]]...
    [*site_to_restrain* $site [#ep [#n1 #n2 #n3]]]...
[*sites_geometry* N]...
    [*site_to_restrain* $site [#ep [#n1 #n2 #n3]]]...
[*siv_s1_s2* # #]
[*stack* $layer_name]...
    [*sx* E] [sy E] [sz E]
    [*generate_these* $sites]
        [*generate_name_append* $append_to_site_name]
[*view_structure*]

Thkl_lat

[*a* E][*b* E][*c* E][*al* E][*be* E][*ga* E]
[*normals_plot* !E]…
    [*normals_plot_min_d* !E]
[*phase_penalties* $sites N [*hkl_Re_Im* #h #k #l #Re #Im]...]…
    [*accumulate_phases_and_save_to_file* $file]
        [*accumulate_phases_when* !E]
[*omit_hkls* !E]
[*spherical_harmonics_hkl* $name]...
    [*sh_Cij_prm* $Yij E]...
    [*sh_order* #]
    [*sh_alpha* !E]
[*str_hkl_angle* N h k l]...

Trigid

[*rigid*]...
    [*point_for_site* $site_name [*ux*|*ua* E][*uy*|*ub* E][*uz*|*uc* E]]...
        [*in_cartesian*][*in_FC*]
    [*z_matrix* $atom_1 [$atom_2 E][$atom_3 E][$atom_4 E]] …
    [*rotate* E [*qx*|*qa* E] [*qy*|*qb* E] [*qz*|*qc* E]]...
        [*operate_on_points* $site_names]
        [*in_cartesian*][*in_FC*]
    [*translate* [*tx*|*ta* E][*ty*|*tb* E][*tz*|*tc* E]]...
        [*operate_on_points* $site_names]
        [*rand_xyz* !E]
        [*in_cartesian*][*in_FC*]
        [*start_values_from_site* $unique_site_name]

Tout_record

[*out_record*]
    [*out_eqn* !E]
    [*out_fmt* $c_fmt_string]
    [*out_fmt_err* $c_fmt_string]...

Tmin_r_max_r

[*min_r* #][*max_r* #]

Tspace_group

[*space_group* $symbol]

Miscellanous

[*aberration_range_change_allowed* !E]
[*default_I_attributes* !E]
for, *load*, *move_to* (see Section 12.3)

Tindexing

[*dummy*]
[*index_lam* E]
[*index_max_number_of_solutions*]
[*index_max_Nc_on_No* E]
[*index_max_th2_error* E]
[*index_max_zero_error #*]
[*index_min_lp* E] [*index_max_lp* E]
[*index_th2* E]... or [*index_d* E]...
    [*index_I* E [good]]
[*Index_x0* E]
[*index_zero_error*]
[*seed*]
[*try_space_groups* $symbol ...] …
    [*X_scaler #*]
    [*X_angle_scaler #*]

Tcharge_flipping

General
[*a* !E] [*b* !E] [*c* !E] [*al* !E][*be* !E][*ga* !E]
[*break_cycle_if_true* !E]
[*cf_hkl_file* $file]
[*cf_in_A_matrix* $file]
    [*scale_Aij* !E]
[*delete_observed_reflections* !E]
[*extend_calculated_sphere_to* !E]
[*f_atom_type* $type *f_atom_quantity* !E]…
[*find_origin* !E]
[*fraction_density_to_flip* !E]
[*fraction_reflections_weak* !E]
[*min_d* !E]
[*min_grid_spacing* !E]
[*neutron_data*]
[*space_group* $]
[*use_Fc*]

Electron density perturbations
[*flip_equation* !E]
[*flip_regime_2* !E]
[*flip_regime_3* !E]
[*histogram_match_scale_fwhm* !E]
    [*hm_size_limit_in_fwhm* !E]
    [*hm_covalent_fwhm* !E]
[*pick_atoms* $atoms]...
    [*activate* !E]
    [*choose_from* !E]
    [*choose_to* !E]
    [*choose_randomly* !E]
    [*omit* !E]
    [*displace* !E]
    [*insert* !E]

[*scale_density_below_threshold* !E]
[*symmetry_obey_0_to_1* !E]

Phase perturbations
[*add_to_phases_of_weak_reflections* !E]
[*randomize_phases_on_new_cycle_by* !E]
[*set_initial_phases_to* $file]
    [*modify_initial_phases* !E]
[*tangent_num_h_read* !E]
    [*tangent_num_k_read* !E]
    [*tangent_num_h_keep* !E]
    [*tangent_max_triplets_per_h* !E]
    [*tangent_min_triplets_per_h* !E]
    [*tangent_scale_difference_by* !E]

Miscellaneous
[*apply_exp_scale* !E]
[*correct_for_atomic_scattering_factors* !E]
[*correct_for_temperature_effects* !E]
[*hkl_plane* $hkl]…
[*randomize_initial_phases_by* !E]
[*scale_E* !E]
[*scale_F* !E]
[*scale_F000* !E]
[*scale_weak_reflections* !E]
[*user_threshold* !E]
[*verbose #*]

GUI related
[*add_to_cloud_N* !E [*add_to_cloud_when* !E]]
[*pick_atoms_when* !E]
[*view_cloud* !E]

12.2 Description of keywords

12.2.1 Ttop_xdd

[*convolution_step* #]

An integer defining the number of calculated data points per measured data point. It may be useful to increase this number when the measurement step is large.

convolution_step is set to 1 by default. Only when the measurement step is greater than about 1/8 to 1/5 of the observed peak FWHMs or when high precision is required is it necessary to increase *convolution_step*.

[*r_p* #] [*r_p_dash* #] [*r_wp* #] [*r_wp_dash* #] [*r_exp* #] *[r_exp_dash* #] [*gof* #] [*weighted_Durbin_Watson* #]

xdd dependent or global scope refinement indicators. Keywords ending in "_dash" correspond to background subtracted indicators. For details see section Error: Reference source not found.

[*Rp* #] [*Rs* #]

Defines the primary and secondary radius of the diffractometer in mm.

The default for *Rp* and *Rs* is set to 217.5 mm.

[*x_calculation_step* !E]

Calculation step used in the generation of phase peaks and *fit_obj*'s. Peak_Calculation_Step is the actual step size used, it is defined is as follows:

For and x-axis with equal steps and *x_calculation_step* not defined then

Peak_Calculation_Step = "Observed data step size" / *convolution_step*

otherwise

Peak_Calculation_Step = x_calculation_step / *convolution_step*

x_calculation_step can be a function of Xo and Th. In some situations it may be computationally efficient to write *x_calculation_step* in terms of the function Yobs_dx_at and the reserved parameter Xo. It is also mandatory to define *x_calculation_step* for data with unequal x-axis steps (*.xy or *.xye data files). Example uses of *x_calculation_step* are as follows:

```
x_calculation_step .01
x_calculation_step = .02 (1 + Tan(Th));
x_calculation_step = Yobs_dx_at(Xo);
```

12.2.2 Tglobal

[*A_matrix*] [*C_matrix*] [*A_matrix_normalized*] [*C_matrix_normalized*]

Generates the un-normalized and normalized A and correlation matrices. If *do_errors* is defined then C_matrix_normalized is automatically generated and appended to the OUT file.

[*approximate_A*]
[*A_matrix_memory_allowed_in_Mbytes* !E]
[*A_matrix_elements_tollerance* !E]
[*A_matrix_report_on*]

See section 4.3 for a detailed description of approximate_A.

[*A_matrix_memory_allowed_in_Mbytes* !E]: Limits the memory used by the A matrix to a maximum of *A_matrix_memory_allowed_in_Mbytes*. If the matrix requires less than *A_matrix_memory_allowed_in_Mbytes* then the full matrix is used otherwise the matrix is treated as a sparse matrix.

[*A_matrix_elements_tollerance* !E]: Removes elements in the A matrix with values less than *A_matrix_elements_tollerance*. The comparison is made againts normalized elements of A such that the diagonals have a values of 1. The A matrix is made sparse when *A_matrix_elements_tollerance* is defined. Typical values of *A_matrix_elements_tollerance* range from 0.0001 to 0.01. *A_matrix_memory_allowed_in_Mbytes* and *A_matrix_elements_tollerance* can be used simultanuously.

[*A_matrix_report_on*]: Displays the percentage of non-zero elements in the A matrix.

[*bootstrap_errors* !Ecycles]
[*fraction_of_yobs_to_resample* !E]
**[*resample_from_current_ycalc*
[*determine_values_from_samples*]

bootstrap_errors uses the bootstrap method of error determination (Efron & Tibshirani 1986, DiCiccio & Efron 1996, Chernick 1999). Bootstrapping comprises a series of refinements each with a fraction Yobs data modified to obtain a new bootstrap sample. The standard deviations of the refined values then become the bootstrap errors. !Ecycles corresponds to the number of refinement cycles to perform, it defaults to 200. The resulting bootstrap errors are written to the *.OUT file.

[*fraction_of_yobs_to_resample* !E]: Corresponds to the fraction of the observed data that is to be replaced each refinement cycle, it defaults to 0.37. Replacement data is by default obtained randomly from the calculated pattern obtained at the end of the first refinement cycle.

[*resample_from_current_ycalc*]: If defined then replacement data are obtained from the currently completed refinement cycle. The updated Yobs data is additionally modified such that the change in R_{WP} is unchanged in regards to the current Ycalc.

Parameter values used at the start of each refinement cycle are obtained from the end of the first refinement cycle. *val_on_continue* can additionlly be used to change parameter values at the start of a cycle.

[*determine_values_from_samples*]: If defined then parameter values at the end of bootstrapping are updated with values determined from the bootstrapping refinement cycles.

Parameter values obtained at the end of each bootstrap refinement cycle is written to disk in binary format. These values are then read and processed at the end of the bootstrap process without actually storing all of the values in memory. Thus the bootstrap process has a small memory footprint.

[*chi2* !E]

The *chi2* keyword allows for minimization of a user defined χ^2. It can be a function of the reserved parameter names X, Yobs, Ycalc and SigmaYobs. In addition the keyword *xdd_sum* is a parameter that can be a function of these reserved parameter names. For more details see section 4.10.

For example, to define a normal least squares refinement the following can be used:

```
xdd...
   xdd_sum denominator = Yobs;
   xdd_sum numerator = (Yobs - Ycalc)^2 / Max(Yobs,1);
   chi2 = 100 Sqrt(numerator / denominator);
```

[*chi2_convergence_criteria* !E]

Convergence of the minimization routine is determined when the change in χ^2 is less than *chi2_convergence_criteria* for three consecutive cycles and when all defined *stop_when* parameter attributes evaluate to true.

chi2_convergence_criteria = If(Cycle_Iter < 10, .001, .01);

[*conserve_memory*]

Deletes temporary arrays used in intermediate calculations; memory savings of up to 70% can be expected on some problems with subsequent lengthening of execution times by up to 40%. When *approximate_A* is used on dense matrices then *conserve_memory* can reduce memory usage by up to 90%.

Useful when there are many independent parameters.

[*continue_after_convergence*]

Refinement is continued after convergence. Before continuing the following actions are performed:

val_on_continue equations for independent parameters are evaluated

randomize_on_errors process is performed

rand_xyz processes are performed

The term "refinement cycle" is used to describe a single convergence. Also, when *val_on_continue* is defined then the corresponding parameter is not randomized according to *randomize_on_errors*.

[*do_errors*]

Errors for refined parameters (ESD's) and a correlation matrix are calculated at the end of refinement. The correlation matrix if defined using *C_matrix_normalized* is updated, if not defined then *C_matrix_normalized* is automatically defined and appended to the OUT file.

[*do_errors_include_restraints*]

Errors calculated with the inclusion of restraints in the A matrix.

[*do_errors_include_penalties*]

Errors calculated with the inclusion of penalties in the A matrix.

[*file_name_for_best_solutions* $file]

Appends INP file details to $file during refinement with independent parameter values updated. The operation is performed every time a particular convergence gives the best R_{WP}. For example, suppose that at convergence the following was obtained:

R_{WP}:

30	All prms appended to file in INP format
20	All prms appended to file in INP format
35	
40	
15	All prms appended to file in INP format
18	
10	All prms appended to file in INP format
15	

[iters #]

The maximum number of refinement iterations. *iters* is set to 500000 by default

[iters #]

The maximum number of refinement iterations. *iters* is set to 500000 by default

[*marquardt_constant* !E]

Allows for changing the Marquardt constants.

[*no_LIMIT_warnings*]

Suppresses LIMIT_MIN and LIMIT_MAX warning messages.

[*only_penalties*]

Instructs the minimization procedure to minimize on penalty functions only. The *only_penalties* switch is assumed when there are only penalties to minimize, i.e. when there are no observed data.

Note, parameters that are not functions of the penalties are not refined.

[*out_A_matrix* $file]
[*A_matrix_prm_filter* $filter]

out_A_matrix outputs the least squares A matrix to the file $file; used in the Out_for_cf macro. Output can be limited by using *A_matrix_prm_filter*, here's an example for outputting A matrix elements corresponding to parameters with names starting with "q":

```
out_A_matrix file.a
   A_matrix_prm_filter q*
```

[*out_prm_vals_per_iteration* $file]... I [*out_prm_vals_on_convergence* $file]...
[*out_prm_vals_filter* $filter]

Outputs refined parameter values per iteration or on convergence into the file $file. In the absence of *out_prm_vals_filter* then all parameters are outputted otherwise only parameters with names defined in *out_prm_vals_filter* are considered where $filter can contain the wild card character "*" and the negation character "!", for example:

```
out_prm_vals_per_iteration RESULTS.TXT
   out_prm_vals_filter "* !u*"
```

More than one *out_prm_vals_per_iteration* / *out_prm_vals_on_convergence* can be defined outputting different parameters into different files depending on the corresponding *out_prm_vals_filter*.

[*out_refinement_stats*]

Outputs some refinement statistics.

[*out_rwp* $file]

Outputs a list of R_{WP} values encountered during refinement to the file $file.

[*penalties_weighting_K1* !E]

Defines the weighting K1 given to the penalty functions, see Eq. (4.2).

penalties_weighting_K1 is set to 1 by default

[*pen_weight* !E]

Allows to modify the weights of penalties and restraints. For details refer to section 4.1.

[*percent_zeros_before_sparse_A #*]

Defines the percentage of the A matrix than can be zero before sparse matrix methods are invoked. The default value is 60%.

[*process_times*]

On termination of refinement, process times are displayed.

[*quick_refine* !E]
[*quick_refine_remove* !E]

quick_refine removes parameters that influence χ^2 in a small manner during a refinement cycle. Use of *quick_refine* speeds up simulated annealing, see for the macro Auto_T. All refined parameters are reinstated for refinement at the start of subsequent cycles. Large *quick_refine* values aggressively remove parameters and convergence to low χ^2 maybe hindered. A value of 0.1 usually works well.

quick_refine has the following consequences:

- If parameters are not reinstated using *quick_refine_remove* then χ^2 does not get to its lowest possible value for a particular refinement cycle.
- The degree of parameter randomization is increased with increasing values of *quick_refine*. Thus randomization should be reduced as *quick_refine* increases. Alternatively *randomize_on_errors* can be used which automatically determines the amount a parameter is randomized.

quick_refine_remove removes a parameter from refinement if it evaluates to non-zero or reinstates a parameter if it evaluates to zero. It can be a function of the reserved parameters QR_Removed or QR_Num_Times_Consecutively_Small and additionally global reserved parameters such as Cycle_Iter, Cycle and T. If *quick_refine_remove* is not defined then the removal scheme of section 4.13 is used and parameters are not reinstated until the next refinement cycle.

In most refinements the following will often produce close to the lowest χ^2 and in a shorter time period.

```
quick_refine .1
   quick_refine_remove =
      IF QR_Removed THEN
         0 ' reinstate the parameter
      ELSE
         IF QR_Num_Times_Consecutively_Small > 2 THEN
            1 ' remove the parameter
         ELSE
            0 ' dont remove the parameter
         ENDIF
      ENDIF;
```

[*randomize_file_out_normal* $file]

Instructs TOPAS to randomize the calculated pattern Yc using a Normal distribution and writes Yc to the file $file.

[*randomize_on_errors*]

See section 4.14.

When *val_on_continue* is defined then the corresponding parameter is not randomized according to *randomize_on_errors*.

[*restraint* !E]

Defines a penalty function that can be a function of other parameters. For details see sections 4.1 and 4.6.

[*save_best_chi2*]

Allows to save values on termination of refinement. For details see section 4.7.

[*seed*]

Initialises the random number generator with a different seed based on the computer clock.

[*temperature* !E]...
[*move_to_the_next_temperature_regardless_of_the_change_in_rwp*]
[*save_values_as_best_after_randomization*]
[*use_best_values*]

Defines temperatures for simulated annealing.

A temperature regime has no affect unless the reserved parameter name T is used in *val_on_continue* attributes, or, if the following temperature dependent keywords are used: *rand_xyz*, *randomize_on_errors*.

randomize_on_errors automatically determines parameter displacements without the need for *rand_xyz* or *val_on_continue*. It performs well on a wide range of problems.

The reserved parameter T returns the current temperature and it can be used in equations and in particular the *val_on_continue* attribute. The first temperature defined becomes the starting temperature; subsequent temperature(s) become the current temperature

If χ_0^2 increases relative to a previous cycle then the temperature is advanced to the next temperature.

If χ_0^2 decreases relative to previous temperatures of lesser values then the current temperature is rewound to a previous temperature such that its previous is of a greater value.

[*move_to_the_next_temperature_regardless_of_the_change_in_rwp*]: Forces the refinement to move to the next temperature regardless of the change in R_{WP} from the previous temperature.

[*save_values_as_best_after_randomization*]: Saves the current set of parameters and gives them the status of "best solution". Note, this does not change the global "best solution" which is saved at the end of refinement.

[*use_best_values*]: Replaces the current set of parameters with those marked as "best solution".

A typical temperature regime starts with a high value and then a series of annealing temperatures, for example:

```
temperature 2
   move_to_the_next_temperature_regardless_of_the_change_in_rwp
temperature 1
temperature 1
temperature 1
```

If the current temperature is the last one defined (the fourth one), and χ_0^2 decreased relative to the second and third temperatures, then the current temperature is set to the second temperature.

The current temperature can be used in all equations using the reserved parameter T, for example:

```
x @ 0.123 val_on_continue = Val + T Rand(-.1, .1)
```

The following temperature regime will allow parameters to randomly walk for the first temperature. At the second temperature the parameters are reset to those that gave the "best solution".

```
temperature 1
temperature 1   use_best_values
temperature 1
temperature 1   use_best_values
temperature 1
temperature 10
   save_values_as_best_after_randomization
   move_to_the_next_temperature_regardless_of_the_change_in_rwp
```

Note, that when a "best solution" is encountered the temperature is rewound to a position where the temperature decreased. For example, if the R_{WP} dropped at lines 2 to 5 then the next temperature will be set to "line 1".

The following will continuously use the "best solution" before randomisation. This particular temperature regime has a tendency to remain in a false minimum.

```
temperature 1 use_best_values
```

The macro "Temperature_Regime" simplifies the use of *temperature*. The temperature regime as defined in the macro Auto_T is sufficient for most problems.

[*use_tube_dispersion_coefficients*]

Instructs TOPAS to use laboratory tube anomalous dispersion coefficients instead of the more accurate data from http://www cxro.lbl.gov/optical_constants/asf.html.

[*verbose #*]...

A value of 1 (the default) instructs the kernel to output in a verbose manner.

A value of 0 reduces kernel output such that text output is only initiated at the end of a refinement cycle.

A value of -1 reduces the kernel output such that text output is initiated every second and only R_{WP} values at the end of a refinement cycle is kept.

The Simulated_Annealing_1 macro has verbose with a value of -1; this ensures that lengthy simulated annealing runs do not exhaust memory due to saving R_{WP} values and text output buffers.

12.2.3 Txdd

[*d_ls*]...
[*d* E *I* E]...

Defines a phase type that uses d-spacing values for generating peak positions. d corresponds to the peak position in d-space and I is the intensity parameter before applying any *scale_pks* equations.

[*elemental_composition*]

elemental_composition is an *xdd* dependent keyword that reports the total elemental composition over all *str* phases of the *xdd*, see also section 10.3.1.

[element_weight_percent $atom $Name #]…

element_weight_percent is an *xdd* dependent keyword that returns the weight percent of an element $atom within the corresponding *str*'s of the *xdd*. $Name is the name given to the element $atom. For more details see sections 10.3.2.6 and 10.3.2.7.

[*element_weight_percent_known* $atom #]…

Allows to constrain the weight percent of an element $atom. For more details see sections 10.3.2.6 and 10.3.2.7.

[*gui_add_bkg* !E]

[*fit_obj_phase* !E]: By default phases are now plotted on top of background where background comprises *fit_obj*'s + *bkg*. The *xdd* dependent keyword *gui_add_bkg* and the *fit_obj* dependent *fit_obj_phase* can be used to change the defaults. For details refer to section 11.8.1.

[*hkl_ls*]...
[*hkl_m_d_th2* # # # # # # I E]...
[*lp_search* !E]
[*I_parameter_names_have_hkl* $start_of_parameter_name]
[*scale_phase_X* E]

Defines a phase type that uses hkls for generating peak positions.

[*lp_search* !E]: *lp_search* uses a new indexing algorithm that is independent of d-spacing extraction. For more details see section 7.2.

[*I_parameter_names_have_hkl* $start_of_parameter_name]: Gives generated intensity parameters a name starting with $start_of_parameter_name and ending with the corresponding hkl.

[*hkl_m_d_th2* # # # # # # I E]: The numbers after the keyword *hkl_m_d_th2* define h k l m d and 2θ values, where

h, k, l	: Miller indicies
m	: multiplicity
d and th2	: d and 2θ values (not used by TOPAS)
I	: Peak intensity parameter before applying any *scale_pks*

If no *hkl_m_d_th2* keywords are defined then the hkls are generated using the space group; the generated *hkl_m_d_th2* details are appended at the end of the *space_group* keyword on refinement termination. Intensity parameters are given an initial starting value of 1. If the *lebail* keyword is not defined then the intensity parameters are given the unique code of @.

The Create_hklm_d_Th2_Ip_file macro creates an hkl file listing from structures in the same format as the "load hkl_m_d_th2 I" as shown above. Even though the structure would have no sites, the *weight_percent* keyword can still be used; it will use whatever value is defined by *cell_mass* in order to *calculate weight_percent*.

[scale_phase_X E]: Scales Ycalc point by point. For details see section 11.7.

[*mixture_density_g_on_cm3* !E]

Calculates the density of the mixture assuming a packing density of 1.

[*mixture_MAC* !E]

Calculates the mass absorption coefficient in cm2/g for a mixture as follows:

$$(\mu/\rho)_{mixture} = \sum_{i=1}^{N} (\mu/\rho)_i w_i$$

where w_i and $(\mu/\rho)_i$ is the weight percent and *phase_MAC* of phase i respectively. Errors are reported for *phase_MAC* and *mixture_MAC*.

The following example provides phase and mixture mass absorption coefficients.

```
xdd...
   mixture_MAC 0
   str...
      phase_MAC 0
```

The macros Mixture_LAC_1_on_cm, Phase_LAC_1_on_cm and Phase_Density_g_on_cm3 can calculate the mixture and phase linear absorption coefficients (for a packing density of 1) and phase density, for example:

```
xdd...
   Mixture_LAC_1_on_cm(0)
   str...
      Phase_Density_g_on_cm3(0)
      Phase_LAC_1_on_cm(0)
```

Errors for these quantities are also calculated.

Mass absorption coefficients obtained from http://physics.nist.gov/PhysRefData/XrayMassCoef are used to calculate *mixture_MAC* and *phase_MAC*.

[*scale_phase_X* E]

Scales Ycalc point by point. For details see section 11.7.

[*str* | *dummy_str*]...
[*scale_phase_X* E]

str defines the start of structure information. The macro "STR" simplifies the use of *str*.

dummy_str's can be used to represent quantitative results arising from non-str phases or to plot *fit_obj*'s representing any phases, see e.g. sections 10.2.5 and 11.8.2.

[scale_phase_X E]: Scales Ycalc point by point. For details see section 11.7.

[*user_y* $Name { #include $Some_file }]…

user_y provides a means to fit any kind of xy data to observed patterns such as other observed patterns or any (learnt) shapes. $Name corresponds to a parameter name given to the *user_y*; it can be used in all equations that can be a function of X. For details see section 11.6.

[*weight_percent_amorphous* !E]

Determines the amorphous content in a sample. The phase dependent keyword of *spiked_phase_measured_weight_percent* needs to be defined in order for *weight_percent_amorphous* to be calculated.

[*xdd* $file [{ $data }] [*range #*] [*xye_format*] [*gsas_format*] [*fullprof_format*]]...

Defines the start of *xdd* dependent keywords and the file containing the observed data.

[{$data}] allows the insertion of ASCII data directly into an input file.

[*range #*] applies to Bruker AXS *.RAW data files; in multi-range files it defines the range to be refined with the first range starting at 1. range is set to 1 by default.

[*xye_format*] [*gsas_format*] [*fullprof_format*]. *xye_format* signals the loading of columns of x, y and error values; additional columns are ignored. *gsas_format* and *fullprof_format* signals the loading of GSAS and FullProf file formats.

The following instruction will refine on the first range in the data file pbso4.raw:

```
xdd pbso4.raw
```

The following will refine on the third range:

```
xdd pbso4.raw range 3
```

To read data from an INP file, the following statements can be used:

```
xdd
{
1 1 10              ' start, step and finish (equidistant data)
1 2 3 4 5 6 7 8 9 10
}
xdd
{
_xy                  ' switch indicating x-y format
0.1 1   0.2 2   ...
}
```

The macro "XDD" simplifies the use of *xdd*. Supported file formats are described in section 14.

[*xdd_sum* !E]

The xdd_sum keyword can be e.g. used to define minimization of a user defined χ^2. It can be a function of the reserved parameter names X, Yobs, Ycalc and SigmaYobs. For more details see section 4.10.

[*xo_Is*]...
[*xo* E I E]...

Defines a phase type that uses x-axis space for generating peak positions.

[*xo* E I E]: *xo* corresponds to the peak position and I is the intensity parameter before applying any *scale_pks* equations.

12.2.4 Txdd_scr

[*xdd_scr* $file] …
[*dont_merge_equivalent_reflections*]
[*dont_merge_Friedel_pairs*]
[*ignore_differences_in_Friedel_pairs*]
[*str*]...

xdd_scr defines single crystal data from the file $file. The file can have extensions of *.HKL for ShelX HKL4 format or *.SCR for SCR format. All *xdd* and *str* keywords that are not dependent on powder data can be used by *xdd_scr* and *hkl_Is_from_hkl4*. Single crystal data is internally stored in 2θ versus F_o^2 format. This means that a *lam* definition is necessary and the keywords *start_X*, *finish_X* and *exclude* can be used with *xdd_scr*.

[*dont_merge_equivalent_reflections*]: Unmerges equivalent reflections, see also section 9.1.3.

[dont_merge_Friedel_pairs]: Prevents the merging of Friedel pairs, see also section 9.1.3.

[*ignore_differences_in_Friedel_pairs*]: Forces the use of equation (9.12) for calculating F^2, see also section 9.1.3.

An example input segment for single crystal data refinement is as follows:

```
xdd_scr ylidm.hkl
MoKa2(0.001)
finish_X 35
weighting = 1 / (Sin(X Deg / 2) Max(1, Yobs));
STR(P212121)
   a  5.9636
   b  9.0390
   c 18.3955
   scale @ 1.6039731906
   site S1  x @ 0.809  y @ 0.180  z @ 0.740  occ S 1  beq 2
   site O1  x @ 0.090  y @ 0.815  z @ 0.223  occ O 1  beq 2
   ...
```

The SCR format is white space delimited and consists of entries of h, k, l, m, d, 2θ, F_o^2 which is the format outputted by the Create_hklm_d_Th2_Ip_file macro.

12.2.5 Tscr

[*Flack* E]

Returns the Flack parameter for single crystal data and for non-centrosymmetric structures, for more details see section 9.1.4.

[*i_on_error_ratio_tolerance* #]

Filters out hkl's that do not meet the condition: |Fo| > *i_on_error_ratio_tolerance* |Sigma(Fo)|

[*num_highest_I_values_to_keep* #num]

Removes all hkl's except for #num hkl's with the highest Fo values.

12.2.6 Txdd_comm_1

[*bkg* [@] # # #...]

Defines a Chebyshev polynomial where the number of coefficients are equal to the number of numeric values appearing after the keyword *bkg*.

The number of coefficients is not limited.

[*degree_of_crystallinity* #]
[*crystalline_area* #]
[*amorphous_area* #]

The *degree_of_crystallinity* keyword reports the so-called degree of crystallinity of the sample in percent:

```
degree_of_crystallinity = 100 * crystalline_area /
                          (crystalline_area + amorphous_area)
```

crystalline_area comprises the sum of phase areas for phases that are not flagged as amorphous. *amorphous_area* comprises the sum of phase areas for phases that are flagged as amorphous using the *amorphous_phase* keyword

Areas are calculated numerically by *numerical_area* and include the scaling from *scale_pks* (i.e. Lorentz-Polarisation correction etc.).

```
xdd ...
      crystalline_area 0
      amorphous_area 0
      degree_of_crystallinity 0
      str...
            numerical_area 0
      hkl_I...
            numerical_area 0
      hkl_I...
            numerical_area 0
            amorphous_phase
      xo_I...
            numerical_area 0
            amorphous_phase
```

Note that the areas calculated by *numerical_area* therefore usually do not match peak areas obtained from profile fitting, unless the same corrections such as Lorentz-Polarisation are applied.

[*d_spacing_to_energy_in_eV_for_f1_f11* !E]

Can be a function of the reserved parameter D_spacing. Changes f' and f" to correspond to energies as given by *d_spacing_to_energy_in_eV_for_f1_f11*. Used for refining on energy dispersive data, for example,

```
' E(eV) = 10^5 / (8.065541 Lambda(A))
prm !detector_angle_in_rad = 7.77 Deg_on_2;
prm wavelength = 2 D_spacing Sin(detector_angle_in_rad);
prm energy_in_eV = 10^5 /  (8.065541 wavelength);
pk_xo = 10^-3 energy_in_eV + zero;
d_spacing_to_energy_in_eV_for_f1_f11 = energy_in_eV;
```

[*exclude* #ex1 #ex2]...

Excludes an x-axis region between #ex1 and #ex2.

The macro "Exclude" simplifies the use of *exclude*.

[extra_X_left !E] [extra_X_right !E]

Determines the extra range to which hkls are generated. For TOF data *extra_X_left* is typically used. For X-ray data then *extra_X_right* is typically used. Both default to 0.5.

[*fit_obj* E [*min_X* !E] [*max_X* !E]]...
[*fit_obj_phase* !E]
[*fo_transform_X* !E]

fit_obj's can be used to insert a user-defined function. *fit_obj*'s can be a function of X. Fit objects do not have any convolutions applied to them!

[*min_X* !E] [*max_X* !E]: These equations define the x-axis range of the *fit_obj*; if *min_X* is omitted then the *fit_obj* is calculated from the start of the x-axis; similarly if *max_X* is omitted then the *fit_obj* is calculated to the end of the x-axis.

[*fit_obj_phase* !E]: By default phases are now plotted on top of background where background comprises *fit_obj*'s + *bkg*. The *xdd* dependent keyword *gui_add_bkg* and the *fit_obj* dependent *fit_obj_phase* can be used to change the defaults. For details refer to section 11.8.1.

[*fo_transform_X* !E]: Provides a means to transform an observed data x-axis to the x-axis of a user_y object. For details see section 11.6.

[*neutron_data*]

Signals the use of neutron atomic scattering lengths. Scattering lengths for isotopes can be used, for example use the isotope name after "occ" as in:

```
occ 6Li 1
occ 36Ar 1
```

The scattering lengths data are contained in file neutscat.cpp, obtained from
www.ccp14.ac.uk/ccp/web-mirrors/neutrons/n-scatter/n-lengths/LIST~1.HTM

Constant wavelength neutron diffraction requires a Lorentz correction, e.g. using the Lorentz_Factor macro; it is defined as follows:

```
scale_pks = 1 / (Sin(Th)^2 Cos(Th));
```

[*rebin_with_dx_of* !E]

rebin_with_dx_of rebins the observed data and can be a function of the reserved parameter X. If *rebin_with_dx_of* evalutes to a constant then the observed data is re-binned to equal x-axis steps. For observed data that is of unequal x-axis steps then re-binning provides a means of converting it to equal x-axis steps.

[*smooth* #num_pts_left_right]

Performs a Savitzky-Golay smoothing of the observed data. The smoothing encompasses (2 * #num_pts_left_right + 1) points.

[*start_X* #] [*finish_X* #]

Defines the start and finish X region to fit to.

[*weighting* !E [*recal_weighting_on_iter*]]

Used for calculating the *xdd* dependent weighting function in χ_0^2. The reserved parameter names X, Yobs, Ycalc and SigmaYobs can be used in this equation, the default is as follows:

```
weighting = 1 / Max(Yobs, 1);
```

In cases where *weighting* is a function of Ycalc then *recal_weighting_on_iter* can be used to recalculate the weighting at the start of every refinement iterations. Otherwise the weighting is recalculated at the start of each refinement cycle.

[*xdd_out* $file [*append*]]...
Tout_record

Used for writing *xdd* dependent details to the file "$file". See Tout_record (section 12.2.15) for a description of *out_record*.

The Out_Yobs_Ycalc_and_Difference macro is a good example using *xdd_out*.

[*yobs_eqn* !N E]

The keyword *yobs_eqn* is used in place of the *xdd* keyword. It describes the observed data as an equation which is very useful for approximating functions. The name given to the equation !N is used for identifying the equation in the GUI.

Most significantly, *yobs_eqn* can be used to calculate powder patterns, see also section 11.1.

[*yobs_out* $file] [*ycalc_out* $file] [*diff_out* $file]

Outputs the observed, calculated and difference patterns respectively in a format as specified in the extension of $file. Extensions *.xy, *.scr, *.xdd are supported. Further formats are accommodated using any number of macros including Out_Yobs_Ycalc_and_Difference, Out_X_Yobs etc…

[*yobs_to_xo_posn_yobs* !E]

At the start of refinement *yobs_to_xo_posn_yobs* decomposes an X-ray diffraction pattern into a new diffraction pattern comprising at most one data point per hkl. Fitting to the decomposed pattern in a normal Rietveld refinement manner is then possible due to the ability to refine data of unequal x-axis step sizes. This normal Rietveld manner of fitting is important in structure solution from Global Optimization as the background can still be refined and the problem of peak overlap avoided. These new data points are not extracted intensites and thus the problem of peak overlap in intensity extraction is avoided. The much smaller number of data points in the new diffraction pattern can greatly improve speed in structure determination; in other words the calculation time in synthesising the diffraction pattern becomes close to that of when dealing with single crystal data.

If the distance between two hkls is less than the value of *yobs_to_xo_posn_yobs* then the proposed data point at one of these hkls is discarded. Thus the final decomposed pattern may in fact have less data points than hkls. A reasonable value for *yobs_to_xo_posn_yobs* is Peak_Calculation_Step, or,

yobs_to_xo_posn_yobs = Peak_Calculation_Step;

yobs_to_xo_posn_yobs can be a function of the reserved parameter X with X being the value of the x-axis at the hkl.

It is important to determine and then fix all peak shape, zero error and lattice parameters before using *yobs_to_xo_posn_yobs*. Also, if the original diffraction pattern contains a lot of noise then it may be best to smooth it using the *smooth* keyword or re-binned using *rebin_with_dx_of*. Alternatively, a calculated pattern could be used as input into the *yobs_to_xo_posn_yobs*

Note: The structure solution can be speeded up by preventing graphical output or by increasing in Graphics Response Time in the GUI.

12.2.7 Tcomm_1

[*axial_conv*]...
_filament_length_ E
_sample_length_ E
_receiving_slit_length_ E
[*primary_soller_angle* E]
[*secondary_soller_angle* E]
[*axial_n_beta* !E]

Defines the full axial divergence model (Cheary & Coelho, 1998b).

filament_length E: Length of the tube filament in the axial plane in mm.

sample_length E: Length of the sample in axial direction (perpendicular to the direction of the beam) in mm.

receiving_slit_length E: Length of the receiving slit in the axial plane in mm.

[*primary_soller_angle* E]: Angle of the primary Soller slit in degrees.

[*secondary_soller_angle* E]: Angle of the secondary Soller slit in degrees.

[*axial_n_beta* #20]: Define the number of rays emanating from a point X-ray source in the axial plane. Larger values for *axial_n_beta* increases both accuracy and calculation time.

The macro Full_Axial_Model simplifies the use of *axial_conv*.

[*capillary_diameter_mm* E]...
[*capillary_u_cm_inv* E]
[*capillary_parallel_beam*] [*capillary_divergent_beam*]

Calculates an aberration for capillary samples and convolutes it into phase peaks to correct for peak shapes, intensities and 2Th shifts. *capillary_diameter_mm* corresponds to the capillary diameter in mm.

[*capillary_u_cm_inv*]: The linear absorption coefficient of the sample in units of cm^{-1}.

[*capillary_parallel_beam*]: Results in a correction for a parallel primary beam.

[*capillary_divergent_beam*]: Results in a correction for a divergent primary beam.

Both *capillary_parallel_beam* and *capillary_divergent_beam* assume that the capillary is fully illuminated by the beam in the equitorial plane.

[*circles_conv* E]...

Defines ε_m in the convolution function:

$(1 - |\varepsilon_m / \varepsilon|^{½})$ for $\varepsilon = 0$ to ε_m

that is convoluted into phase peaks. ε_m can be greater than or less than zero.

circles_conv is used for example by the Simple_Axial_Model macro.

[*exp_conv_const* E [*exp_limit* E]]...

Defines ε_m in the convolution function:

$\text{Exp}(\text{Ln}(0.001)\ \varepsilon/\varepsilon_m)$ for $\varepsilon = 0$ to *exp_limit*

that is convoluted into phase peaks. *exp_conv_const* is used by the Absorption and Absorption_With_Sample_Thickness_mm macros. If *exp_limit* is not defined then it defaults to ε_m. ε_m can be greater than or less than zero.

[*ft_conv* !E]...
[*ft_min* !E]
[*ft_x_axis_range* !E]
Get(ft_0)
FT_Break

The keyword *ft_conv* describes a Fourier Transform (FT) of a response function that is convoluted into phase peaks using a Fast Fourier Transform (FFT). Refer to section 5.4.2.1 for a detailed description.

[*gauss_fwhm* E]...

Defines the FWHM of a Gaussian function to be convoluted into phase peaks.

gauss_fwhm is for example used by the CS_G and Strain_G macros.

[*h1* E *h2* E *m1* E *m2* E]

Split-PearsonVII profile parameters, for details see the *peak_type* keyword and section 5.

[*hat* E [*num_hats #*]]...

Defines the X-axis size of an impulse function that is convoluted into phase peaks.

[*num_hats*]: The number of hats to be convoluted.

num_hats is set to 1 by default.

hat is used for example by the Slit_Width and Specimen_tilt macros.

[*lor_fwhm* E]...

Defines the FWHM of a Lorentzian function that is convoluted into phase peaks.

lor_fwhm is for example used by the CS_L and Strain_L macros.

[*lpsd_th2_angular_range_degrees* E]...
_lpsd_equitorial_divergence_degrees_ E
_lpsd_equitorial_sample_length_mm_ E
[*lpsd_beam_spill_correct_intensity* !E]

Calculates a generic aberration for a linear position sensitive detector and convolutes it into phase peaks to correct for peak shapes, intensities and 2θ shifts. *lpsd_th2_angular_range_degrees* corresponds to the angular range of the LPSD in 2Th degrees.

lpsd_equitorial_divergence_degrees: Equatorial divergence in degrees of the primary beam.

lpsd_equitorial_sample_length_mm: Lengths of the sample in the equatorial plane.

lpsd_beam_spill_correct_intensity: Corrects intensity deviations inherent to variable slits when !E is set to 1.

[modify_peak]
[modify_peak_eqn !E]
[modify_peak_apply_before_convolutions]
[current_peak_min_x !E]
[current_peak_max_x !E]

Allows to apply a filter to peak profiles based on *modify_peak_eqn*. With *modify_peak_apply_before_convolutions* peak profiles are modified before convolutions, otherwise after. *current_peak_min_x* and *current_peak_max_x* determine the extent to which the peak profiles are calculated, where x refers to the wavelength. *modify_peak* functionality is realized by using the internal data objects of Get(*current_peak_x*) and Get(*current_peak*). These two objects return the x-axis wavelength being worked on by the program and the current calculated peak intensity at that x-axis position respectively.

The Absorption_Edge_Correction macro can be used to simplify the use of *modify_peak* for describing absorption edges (see section 5.3.2.1.2).

[*one_on_x_conv* E]...

Defines ε_m in the convolution function:

$(4 \mid \varepsilon_m \varepsilon \mid^{-½}$ for $\varepsilon = 0$ to ε_m

that is convoluted into phase peaks. ε_m can be greater than or less than zero.

one_on_x_conv is used for example by the Divergence macro.

[*pk_xo* E]

Provides a mechanism for transforming peak position to an x-axis position.

The peak position for neutron time-of-flight data is typically calculated in time-of-flight space, tof, or:

tof = t0 + t1 dhkl + t2 dhkl2

where t0 and t1 and t2 are diffractometer constants. *pk_xo* can be used to refine TOF data.

[*push_peak*]... [*bring_2nd_peak_to_top*]... [*add_pop_1st_2nd_peak*]...[*scale_top_peak* E]…

Provides for manipulation of the peak calculation stack; see section 5.4.3.

[*pv_lor* E *pv_fwhm* E]

Pseudo-Voigt profile parameters, for details see the *peak_type* keyword and section 5.

[*spv_h1* E *spv_h2* E *spv_l1* E *spv_l2* E]

Split-PseudoVoigt profile parameters, for details see the *peak_type* keyword and section 5.

[*stacked_hats_conv*]...
[*whole_hat* E [*hat_height* E]]...
[*half_hat* E [*hat_height* E]]...

Defines hat sizes for generating an aberration function comprising a summation of hat functions. whole_hat defines a hat with an X axis extent of ±*whole_hat*/2. *half_hat* defines a hat with an X-axis range of *half_hat* to zero if *half_hat*<0; or zero to *half_hat* if *half_hat*> 0. *hat_height* defines the height of the hat; it defaults to 1.

stacked_hats is used for example to describe tube tails using the Tube_Tails macro.

[*th2_offset* E]...

Used for applying 2θ corrections to phase peaks.

The following instruction applies a sample displacement correction:

```
th2_offset = -2 Rad (c) Cos(Th) / Rs;
```

th2_offset is used for example in the Zero_Error and Specimen_Displacement macros.

[*user_defined_convolution* E *min* E *max* E]...

Provides for user-defined convolutions that are convoluted into phase peaks and can be a function of X.

[*min* E] [*max* E]: The *min*/*max* equations are mandatory, they define the x-axis extents of the *user_defined_convolution* where *min* <= 0 and *max* >= 0.

E.g. a sinc function can be convoluted into phase peaks as follows:

```
str
   prm k 10 min 0.001 max 100
   user_defined_convolution =
      If(Abs(X) < 10^(-10), 1, (Sin(k X) /(k X))^2);
      min -3 max 3
```

[*WPPM_ft_conv* E]…
_____*WPPM_L_max* E
_____*WPPM_th2_range* E
_____[*WPPM_correct_Is*]

Describes a Fourier transform in s space and performs a convolution on phase peaks that have been interpolated to s space, for example:

```
WPPM_ft_conv = 1 - 1.5 WPPM_L / D + 0.5 (WPPM_L / D)^3;
   WPPM_L_max = D;
   WPPM_th2_range = 25 .1 Rad Lam / (D Cos(Th));
   WPPM_correct_Is
```

The result is then interpolated back to 2θ space. Interpolations are scaled such that I(s)ds =I(θ)dq when *WPPM_correct_Is* is defined; the effect of this scaling is typically small at low angles and becomes noticeable at very high angles reaching a maximum at 180 degrees 2θ where the derivative of Cos(θ) is at a maximum.

When multiple *WPPM_ft_conv*(s) are defined then the program will internally use the convolution theorem.

WPPL_L is a reserved parameter name that returns the transform parameter.

WPPM_L_max defines the maximum WPPL_L.

Get(ft_0) and FT_Break can both be used in *WPPM_ft_conv* in a manner similar to *ft_conv*.

The tails of WPPM peaks extend for almost the whole diffraction pattern; they can be shortened using *WPPM_th2_range*; in the above example this range has been written in terms of the fwhm as defined in the Scherrer equation.

WPPM_ft_conv can be a function of the following reserved parameter names and keywords:

H, K, L, M, Th, Xo, D_spacing, WPPM_L and *spherical_harmonics_hkl*

WPPM_Ln_k is a reserved parameter name that returns Ln of an integer and is used to calculate Ln(Kc WPPM_L) in a fast manner.

12.2.8 Tcomm_2

[*f0_f1_f11_atom*]…
[*f0* E] [*f1* E] [*f11* E]

Enables refinement of X-ray and neutron form factors.

[*f0* E]: Represents either the scattering factor f0 in the X-ray case, or scattering lengths in the neutron case.

[*f1* E] [*f11* E]: Apply to the f' and f" anomalous scattering factors, respectively.

[*existing_prm* E]

The keyword *existing_prm* allows for the modification of an existing *prm* or *local* parameter. For more details see section 2.2.3.

[*lam* [*ymin_on_ymax #*] [*no_th_dependence*] [*la* E *lo* E [*lh* E *lg* E]]...]
[[Lam !E] [*calculate_Lam*]]

Defines an emission profile where each "[*la* E *lo* E [*lh* E *lg* E]]" determines an emission profile line, where

la : Area under the emission profile line.

lo : Wavelength in Angstroms of the emission profile line.

lh : Lorentzian HW of the emission profile line in mili-Angstroms.

lg : Gaussian HW of the emission profile line in mili-Angstroms.

[*ymin_on_ymax #*]: Determines the x-axis extent to which an emission profile line is calculated. Set to 0.001 by default.

[*no_th_dependence*]: Defines an emission profile that is 2θ independent (no broadening related to spectral dispersion) and allows to fit to any XY data.

[Lam !E]: Defines the value to be used for the reserved parameter Lam. When Lam is not defined then the reserved parameter Lam is defined as the wavelength of the emission profile line with the largest *la* values. Note that Lam is used to determine the Bragg angle.

[*calculate_Lam*]: Calculates Lam such that it corresponds to the wavelength at the peak of the emission profile. Lam needs to be set to an approximate value corresponding to the peak of the emission profile.

[*out* $file [*append*]]...
Tout_record

Used for writing parameter details to a file. The details are appended to $file when append is defined. See Tout_record (section 12.2.15) for a description of the keyword *out_record*.

[*penalty* !E]...

Defines a penalty function that can be a function of other parameters. For details see sections 4.1 and 4.6.

[*prm* / *local* E
[*min* !E] [*max* !E] [*del* !E] [*update* !E] [*stop_when* !E] [*val_on_continue* !E]]...

Provides for the definition of user-defined parameters. For details refer to section 2.

[*scale_pks* E]...

Used for applying intensity corrections to phase peaks.

The following instruction defines a Lorentz-Polarisation correction:

```
scale_pks = (1+Cos(c Deg)^2 Cos(2 Th)^2)/(Sin(Th)^2 Cos(Th));
```

scale_pks is used for example in the LP_Factor, Preferred_Orientation and Absorption_With_Sample_Thickness_mm macros.

12.2.9 Tphase_1

[*atom_out* $file [*append*]]…
Toutrecord

Used for writing site dependent details to a file. See Tout_record (section 12.2.15) for a description of the keyword *out_record*.

[*auto_scale*]:

Rewrites the *scale* parameter in terms of F^2. This eliminates the need for the *scale* parameter. The value determined for *auto_scale* is updated at the end of refinement.

[*brindley_spherical_r_cm* !E]

Used for applying the Brindley correction (Brindley, 1945) for spherical particles.

The macro Apply_Brindley_Spherical_R_PD(R, PD), defined as:

```
macro Apply_Brindley_Spherical_R_PD(R, PD)
{
      brindley_spherical_r_cm = (R) (PD);
}
```

R is the radius of the particle in cm and PD is the packing density, a number that is not updated and not refined. Here is an example:

```
xdd...
      str
            Apply_Brindley_Spherical_R_PD(R, PD)
            MVW(0,0,0)
      str
            Apply_Brindley_Spherical_R_PD(R, PD)
            MVW(0,0,0)
```

Note, that PD is incorporated by way of an equation definition for *brindley_spherical_r_cm*. Also, *phase_MAC* or MVW need not be defined as they are created as needed; their definition however is necessary in order to obtain their respective values. The Brindley correction is not applied to phases without the *brindley_spherical_r_cm* defined.

The Brindley correction can be applied to all phases including *xo_Is*. In the case of phases that do not have lattice parameters or sites then the user would have to enter values for *volume*, *str_mass* and *phase_MAC* in order for the Brindley correction to work and for the weight percents to be obtained. This allows for the incorporation of non-structural phases in quantitative analysis. For example, the following works as the necessary information have been included.

```
xo_Is
      Apply_Brindley_Spherical_R_PD(.002, .6)
      MVW(654, 230, 0)
      phase_MAC 200
```

[*cell_mass* !E] [*cell_volume* !E] [*weight_percent* !E]
[*spiked_phase_measured_weight_percent* !E] [*corrected_weight_percent* !E]

Weight percent parameters.

[*cell_mass* !E]: Unit cell mass.

[*cell_volume* !E]: Unit cell volume.

[*weight_percent* !E]: Relative phase amount in a mixture.

[*spiked_phase_measured_weight_percent* !E]: Defines the weight percent of a spiked phase. Used by the xdd dependent keyword *weight_percent_amorphous* to determine amorphous weight percent. Only one phase per xdd is allowed to contain the keyword *spiked_phase_measured_weight_percent*

[*corrected_weight_percent* !E]: Weight percent after considering amorphous content as determined by *weight_percent_amorphous*.

The weight fraction w_p for phase "p" is calculated as follows:

$$w_p = \frac{Q_p}{\sum_{p=1}^{N_p} Q_p}$$

where

- $Q_p = S_p M_p V_p / B_p$
- N_p = Number of phases
- S_p = Rietveld scale factor for phase p.
- M_p = Unit cell mass for phase p.
- V_p = Unit cell volume for phase p.
- B_p = Brindley correction for phase p.

The Brindley correction (Brindley, 1945) is a function of *brindley_spherical_r_cm* and the phase and mixture linear absorption coefficients; the latter two are in turn functions of *phase_MAC* and *mixture_MAC respectively*, or,

B_p is function of: (LAC_{phase} - $MAC_{mixture}$) *brindley_spherical_r_cm*

where

- LAC_{phase} = linear absorption coefficients of phase p, packing density of 1
- $MAC_{mixture}$ = linear absorption coefficients of the mixture, packing density of 1

This makes B_p a function of the weight fractions w_p of all phases and thus w_p as written above cannot be solved analytically. Subsequently w_p is solved numerically through the use of iteration.

[*del_approx* !E]

del_approx groups peaks from the peaks buffer whilst summing peaks to Ycalc; the peaks are grouped such that their 2Th positions all lie within:

- *del_approx* Peak_Calculation_Step < 2Th < *del_approx* Peak_Calculation_Step

Once the group is found then only the two peaks with the smallest and largest 2Th is kept. The in-between peaks have their intensities appropriated to the kept peaks. A particular I(2Th) intensity has its intensity distributed to the two end peaks as follows:

$I(2Th_1) = I(2Th)\ f$

$I(2Th_2) = I(2Th)\ (1 - f)$

where,

$f = (\cos(\pi(2Th - 2\ Th1) / (2\ Th2 - 2\ Th1)) + 1) / 2$

del_approx increases computation speed at a relatively small cost to accuracy; a value between 1 and 3, dependent on Peak_Calculation_Step, is typically acceptable.

[*phase_MAC* !E]

Calculates the mass absorption coefficient in cm^2/g for the current phase. See description for *mixture_MAC*.

[*phase_name* $phase_name]

The name given to a phase, used for reporting purposes.

[*phase_out* $file [*append*]]...
Tout_record

Used for writing phase hkl dependent details to file. See Tout_record (section 12.2.15) for a description of the keyword *out_record*.

The Create_d_Ip_file macro is a good example of the use of *phase_out*.

[*r_bragg* #]

Reports on the value for R-Bragg.

Note, R-Bragg is independent of hkl's and thus can be calculated for all phase types that contain phase peaks.

[*remove_phase* !E]

The *remove_phase* keyword (used by the Remove_Phase macro) allows for phase removal during refinement. For more details see section 10.3.3.

[*scale* E]

The Rietveld scale factor, can be applied to all phase types.

12.2.10 Tphase_2

[*amorphous_phase*]

Signals that the associated phase is amorphous when calculating *degree_of_crystallinity*.

[*numerical_area* E]

Returns the area calculated numerically under the phase and is e.g. used for calculating *degree_of_crystallinity*.

[*peak_buffer_step* E [*report_on*]]

As the shapes of phase peaks do not change significantly over a short 2θ range, a new peak shape is calculated only if the position of the last peak shape calculated is more than the distance defined by *peak_buffer_step*. Various stretching and interpolation procedures are used in order to calculate in-between peaks.

The reserved parameter names of H, K, L, M or parameter names associated with the keywords *sh_Cij_prm* and *hkl_angle* when used in the peak convolution equations result in irregular peak shapes over short 2θ ranges and thus a separate peak shape is calculated for each peak.

When defined *report_on* causes the display of the number of peaks in the peaks buffer.

The equation of *peak_buffer_step* is set to 500*Peak_Calculation_Step by default.

[peak_type $type]

Sets the peak type for a phase. The following *peak_type*'s are available:

Peak type	$type	Parameters
First Principles	fp	
Pseudo-Voigt	pv	[*pv_lor* E *pv_fwhm* E] *pv_lor* is the Lorentzian fraction of the peak profile(s). *pv_fwhm* is the FWHM of the peak profile(s).
Split-PearsonVII	spvii	[*h1* E *h2* E *m1* E *m2* E] The sum of *h1* and *h2* gives the FWHM of the composite peak. *m1*, *m2* are the PearsonVII exponents of the left and right composite peak, respectively.
Split-PseudoVoigt	spv	[*spv_h1* E *spv_h2* E *spv_l1* E *spv_l2* E] The sum of *spv_h1* and *spv_h2* gives the full width at half maximum of the composite peak. *spv_l1*, *spv_l2*: the Lorentzian fractions of the left and right composite peak, respectively.

For more details about peak generation and peak types see section 5.

12.2.11 Tlebail

[*lebail #*]

A 1 for the *lebail* keyword flags the use of the Le Bail method for peak intensity extraction.

12.2.12 Tstr_details

[*append_cartesian*] [*append_fractional* [*in_str_format*]]

Appends site fractional coordinates in Cartesian coordinates or in fractional coordinates respectively to the end of the *.OUT file at the end of a refinement. For the case of *append_fractional*, the *in_str_format* keyword formats the output in INP format.

[*append_bond_lengths* [*consider_lattice_parameters*]]

Appends bond lengths to the end of the *.OUT file at the end of a refinement. A number corresponding to equivalent positions is appended to site names.

consider_lattice_parameters includes the effects of the lattice parameter errors in the calculation of bond length and bond angle errors.

An example of bond lengths output is as follows:

```
Y1:0     O1:0     2.23143
         O2:0     2.23143     88.083
         O3:0     2.28045     109.799     99.928
```

The first line gives the distance between the sites Y1 and O2. The first number in the second line gives the distance between sites Y1 and O2. The third number of 88.083 gives the angle between the vectors Y1 to O1 and Y1 to O2. The first number on the third line contains the distance between sites Y1 and O3. The second number in the third line contains the angle between the vectors Y1 to O3 and Y1 to O2. The third number in line three contains the angle between the vectors Y1 to O3 and Y1 to O1. Thus bond lengths correspond to the first number in each line and bond angles start from the second number. The numbers after the site name and after the ":" character corresponds to the site equivalent position as found in the *.SG space group files found in the SG directory

[*atomic_interaction* N E] | [*ai_anti_bump* N]...
** *ai_sites_1* $sites_1**
** *ai_sites_2* $sites_2**
** [*ai_no_self_interation*]**
** [*ai_closest_N* !E]**
** [*ai_radius* !E]**
** [*ai_exclude_eq_0*]**
** [*ai_only_eq_0*]**

Defines an atomic interaction with the name N between sites identified by $site_1 and $site_2. For *atomic_interaction* E is the site interaction equation that can be a function of the reserved parameters R and Ri. R returns the distance in Å between two atoms; these distances are updated when dependent fractional atomic coordinates are modified. The name of the *atomic_interaction* N can be used in equations and in particular penalty equations.

For *ai_anti_bump* an anti-bump interaction equation is internally generated. For anti-bumping only the *ai_anti_bump* is faster than using *atomic_interaction*. The macro AI_Anti_Bump uses *ai_anti_bump*.

[*no_self_interaction*]: Prevents any interactions between equivalent positions of a site. This is useful when a general position is used to describe a special position.

[*ai_closest_N* !E] When defined interactions between $sites_1 and $sites_2 are sorted by distance and only the first *ai_closest_N* number of interactions are considered.

[*ai_radius* !E] When defined, only the interactions between $sites_1 and $sites_2 that are within the distance ai_radius are considered.

When *ai_radius* and *ai_closest_N* are both defined then interactions from both sets of corresponding interaction are condidered.

[*ai_exclude_eq_0*] When defined only interactions that is not the first equivalent positions in $sites_2 are considered. For example, in the following:

```
atomic_interaction...
      ai_exclude_eq_0
      ai_sites_1 Pb
      ai_sites_1 O1 O2"
```

the following interactions are considered:

Pb:0 and O1:n (n ≠ 0)

Pb:0 and O2:n (n ≠ 0)

where the number after the ":" character corresponds to the equivalent positions of the sites.

[*ai_only_eq_0*] When defined only interactions between equivalent positions 0 are considered.

The *atomic_interaction* equation can be a function of the following functions:

AI_R(#ri): Returns the distance between the current site and the atom defined with Ri = #ri.

AI_R_CM: A function of no arguments that returns the geometric center of the current atom and the atoms defined in $sites_2.

AI_Flatten(#toll): A function that returns the sum of distances of the current atom and those defined in $sites_2 to an approximate plane of best fit. The plane of best fit is constructed such that the sum of the perpendicular distances to the current atom plus those defined in $sites_2 are a minimum

AI_Cos_Angle(#ri1, #ri2): Returns the Cos of the angle between the atom define as Ri=#ri1, the current atom and the atom defined as Ri=#ri2.

AI_Angle(#ri1, #ri2) : Similar to AI_Cos_Angle except that the value returned is the angle in degrees.

atomic_interaction's can be used to apply geometric restraints. For example, an anti-bump interaction between symmetry related molecules can be formulated as follows:

```
atomic_interaction ai1 =
IF R < 3 THEN
      (R-3)^2
ELSE
      0
ENDIF
;
ai_exclude_eq_0
ai_sites_1 C*
ai_sites_1 C*
ai_radius 3
penalty = If(Cycle_Iter < 10, ai1, 0);
```

This example demonstrates anti-bumping between molecules for the first ten iterations of a refinement cycle.

[*box_interaction* [*from_N #*] [*to_N #*] [*no_self_interaction*]
$site_1 $site_2 N E]...

Defines a site interaction with the name N between sites identified by $site_1 and $site_2. E represents the site interaction equation which can be a function of the reserved parameters R and Ri. R returns the distance in Å between two atoms; these distances are updated when dependent fractional atomic coordinates are modified. The name of the *box_interaction* N can be used in equations and in particular penalty equations.

[*from_N #*] [*to_N #*]: When either *from_N* or *to_N* are defined, the interactions between $site_1 and $site_2 are sorted by distance and only the interactions between the *from_N* and *to_N* are considered.

[*no_self_interaction*]: Prevents any interactions between equivalent positions of a site. This is useful when a general position is used to describe a special position.

For example, the following could be used to iterate from the nearest atom to the third atom from a site called Si1:

```
str
      site Si1...
      site O1...
      site O2...
      site O3...
      box_interaction Si1 O* to_N 2 !si1o = (R-2)^2;
      penalty = !si1o;
```

In this example the nearest three oxygen atoms are soft constrained to a distance of 2 Angstroms by the use of the penalty function. Counting starts at zero and thus *to_N* is set to 2 to iterate up to the third nearest atom.

The wild card character "*" used in "O*" means that sites with names starting with "O" are considered. In addition to using the wild card character, the site names can be explicitly written within double quotation marks, for example:

```
box_interaction Si1 "O1 O2 O3" to_N 3 etc...
```

It is important to realize that interactions between Si1 and the three oxygen atoms O1, O2, O3 may not all be included. For example, if Si1 had as its nearest neighbours the following:

Si1 -> O1,1 at a distance of 1.0 Angstroms

Si1 -> O2,3 at a distance of 1.1 Angstroms

Si1 -> O2,1 at a distance of 1.2 Angstroms

Si1 -> O1,2 at a distance of 1.3 Angstroms

then two equivalent positions of site O1 and two equivalent positions of O2 are included in the interaction equation; thus no interaction between Si1-O3 is considered. To ensure that each of the three oxygens had Si1 included in an interaction equation then the following could be used:

box_interaction "O1 O2 O3" Si1 to_N 0 etc…

Thus the order of $site_1 and $site_2 is important when either *from_N* or *to_N* is defined.

The reserved parameters Ri and Break can also be used in interaction equations when either *from_N* or *to_N* is defined. Ri returns the index of the current interaction being operated on with the first interaction starting at Ri=0.

box_interaction is used for example in the Anti_Bump macro.

[*cloud* $sites]...
[*cloud_population* !E]
[*cloud_save* $file]
[*cloud_save_xyzs* $file]
[*cloud_load_xyzs* $file]
[*cloud_load_xyzs_omit*_rwps !E]
[*cloud_formation_omit_rwps* !E]
[*cloud_try_accept* !E]
[*cloud_gauss_fwhm* !E]
[*cloud_extract_and_save_xyzs* $file]
[*cloud_number_to_extract* !E]
[*cloud_atomic_separation* !E]

cloud allows for the tracking of atoms defined in $sites in three dimensions. It can be useful for determining the average positions of heavy atoms or rigid bodies during refinement cycles.

[*cloud_population* !E]: The maximum number of population members. Each population member comprises the fractional coordinates of $sites and an associated R_{WP} value.

[*cloud_save* $file]: On termination of refinement a CLD file is saved; it can be viewed using the Rigid Body Editor.

For example, a dummy atom, “site X1” say, can be placed at the center of a benzene ring and then tracked as follows:

```
continue_after_convergence
...
cloud "X1"
   cloud_population 100
   cloud_save SOME_FILE.CLD
```

[*cloud_save_xyzs* $file]: Saves a cloud populations to a file.

[*cloud_load_xyzs* $file]: Loads and reuses previously saved populations. *cloud_load_xyzs_omit_rwps* can be used to exclude population members whilst loading; it can be a function of Get(Cloud_Rwp) where Cloud_Rwp is the associated R_{WP} of a population member.

[*cloud_formation_omit_rwps* !E]: Can be used to limit population members in the formation of CLD files; it can be a function of Get(Cloud_Rwp).

[*cloud_try_accept* !E]: Accepts population members if it evaluates to non-zero and if the best R_{WP} since the last acceptance is less than a present population member or if the number of members is less than *cloud_population*. If the number of population members equals *cloud_population* then the population member with the lowest R_{WP} is discarded. cloud_try_accept is evaluated at the end of each refinement cycle; its default value is true. Here’s are some examples:

```
cloud_try_accept = And(Cycle, Mod(Cycle, 50);
```

```
cloud_try_accept = T == 10;
```

[*cloud_gauss_fwhm* !E]: The full width at half maximum of a three dimensional Gaussian that is used to fill the cloud.

[*cloud_extract_and_save_xyzs* $file]: Searches the three dimensional cloud for high densities and extracts xyz positions; these are then saved to $file. *cloud_number_to_extract* defines the number of positions to extract and *cloud_atomic_separation* limits the atomic separation during the extraction. The actual number of positions extracted may be less than *cloud_number_to_extract* depending on the cloud.

***fourier_map* !E]**
[*fourier_map_formula* !E]
[*extend_calculated_sphere_to* !E]
[*min_grid_spacing* !E]
[*correct_for_atomic_scattering_factors* !E]
[*f_atom_type* $type [*f_atom_quantity* !E]]…

If *fourier_map* is non-zero then a Fourier map is calculated on refinement termination and shown in the OpenGL window; maps can be calculated for X-ray or neutron single crystal or powder data.

[*fourier_map_formula* !E]: Determines the type of map and can be a function of the reserved parameter names Fcalc, Fobs and D_spacing; here are some examples:

```
fourier_map_formula = Fobs;                ' The default
```

```
fourier_map_formula = 2 Fobs - Fcalc;
```

Fobs correspond to the observed structure moduli; in the powder data case Fobs is calculated from the Rietveld decomposition formula. Phases are determined from Fcalc.

Reflections that are missing from within the Ewald sphere are included with Fobs set to Fcalc.

[*extend_calculated_sphere_to !E*]: If defined then the Ewald sphere is extended.

scale_pks definitions are removed from Fobs. In the event that *scale_pks* evaluates to zero for a particular reflection then Fobs is set to Fcalc; the number of Fobs reflections set to Fcalc is reported on.

[*min_grid_spacing* !E]: If defined then the grid spacing used is set to the smaller of *min_d*/2 and *min_grid_spacing*; useful for obtaining many grid points for graphical purposes.

[*correct_for_atomic_scattering_factors* !E]: Structure factors are normalized when non-zero and when *f_atom_type*'s are defined. By default structure factors are normalized.

[*f_atom_type* $type [*f_atom_quantity* !E]]… : Defines atom types and number of atoms within the unit cell; used by the tangent formula in determining Eh values and by the Structure Viewer window for picking atoms. For the tangent formula then relative quantities are important.

[*grs_interaction* [*from_N #*] [*to_N #*] [*no_self_interaction*]
$site_1 $site_2 *qi* # *qj* # N E]...

Defines a GRS interaction with a name of N between sites identified by site_1 and site_2. E represents the GRS interaction equation that can be a function of the reserved parameter R, which returns the distance in Angstroms between two atoms; these distances are updated when dependent fractional atomic coordinates are modified. The name of the *grs_interaction* N can be used in equations and in particular penalty equations.

[*from_N #*] [*to_N #*]: When either *from_N* or *to_N* are defined, the interactions between $site_1 and $site_2 are sorted by distance and only the interactions between the *from_N* and *to_N* are considered.

[*no_self_interaction*]: Prevents any interactions between equivalent positions of a site. This is useful when a general position is used to describe a special position.

qi and *qj* corresponds to the valence charges used to calculate the Coulomb sum for the sites $site_1 and $site_2 respectively.

grs_interaction is typically used for applying electrostatic constraints in inorganic materials. The GRS_Interaction macro simplifies the use of *grs_interaction*.

[*hkl_plane* $hkl]...

Used by the OpenGL viewer to display hkl planes. Here are some examples:

```
str…
   hkl_plane 1 1 1
   hkl_plane "2 -2 0"
```

[*mag_only_for_mag_sites*]

When defined the X-ray component to intensity for all magnetic sites for the str in question is ignored. For more details see section 9.12.

[*mag_space_group* $symbol]

Defines the magnetic space group. For more details see section 9.12.

[*no_f11*]

Instructs the program to ignore *f11*. For more details see section 9.8.

[*normalize_FCs*]

If defined then site fractional coordinates are normalized. Normalization does not occur if a fractional coordinate has *min*/*max* limits, is part of a rigid body or part of site restraint of any kind.

[[*occ_merge* $sites] [*occ_merge_radius* !E]]...

occ_merge rewrites the site occupancy of the sites defined in $sites in terms of their fractional atomic coordinates (Favre-Nicolin & Cerny, 2004). This is useful during structure solution for merging octahedra or other types of defined rigid bodies. *occ_merge* is also useful for identifying special positions.

In the present implementation $sites are thought of as spheres with a radius *occ_merge_radius*. When two atoms approach within a distance less than the sum of their respective *occ_merge_radius*'s then the spheres intersect. The occupancies of the sites, occ_xyz, thus become:

occ_xyz = 1 / (1 + Intersection fractional volumes)

In this way any number of sites can be merged when their distances are less than 2 r.

Sites appearing in $sites cannot have their occupancy parameter refined. On termination of refinement the occ parameter values are updated with their corresponding occ_xyz.

[*report_on_str*]

The keyword *report_on_str* reports on f1 and f11 or neutron scattering lengths used. No values are reported when the keyword *d_spacing_to_energy_in_eV_for_f1_f11* is used.

[*site* $site_name
[*x* E] [*y* E] [*z* E]]...
[*num_posns #*] [*rand_xyz* !E] [*inter* !E #]
[*occ* $atom E [*beq* E]]...
[*adps*][*u11* E][*u22* E][*u33* E][*u12* E][*u13* E][*u23* E]
[*layer* $layer_name]
[*mlx* E] [*mly* E] [*mlz* E] [*mg* E]
[*mag_only*]

Defines a site where $site_name is a user-defined string used to identify the site.

[*x* E] [*y* E] [*z* E]: Fractional atomic coordinates.

[*num_posns* #]: Corresponds to the number of unique equivalent position generated from the space group; *num_posns* is updated on termination of refinement.

[*rand_xyz* !E]: If *continue_after_convergence* is defined then *rand_xyz* is executed at the end of a refinement cycle. It adds to the site fractional coordinate a vector u, the direction of which is random and the magnitude in Å is:

|u| = T *rand_xyz*

where T is the current *temperature*. Thus to add a shift to an atom between 0 and 1 Å the following could be used:

```
temperature 1
site ... occ 1 C beq 1  rand_xyz = Rand(0,1);
```

Note that only fractional coordinates (x, y, z) that are independent parameters are randomized.

[*inter* !E #]: Corresponds to the sum of all GRS interactions which are a function of the site. The value of *inter* can represent site electrostatic potentials depending on the type of GRS interactions defined.

[*occ* $atom E [*beq* E]]: Defines the site occupancy factor and the equivalent isotropic temperature factor Beq. $atom corresponds to a valid atom symbol or isotope the data of which is contained in the file ATMSCAT.CPP for X-ray data and NEUTSCATT.CPP for neutron data.

Definition of a site fully occupied by aluminium:

```
site Al1  x 0      y 0     z 0.352  occ Al+3 1    beq 0.3
```

Definition of a site occupied by two different cations:

```
site Fe2  x 0.928  y 0.25  z 0.953  occ Fe+3 0.5  beq 0.25
                                    occ Al+3 0.5  beq 0.25
```

[*adps*] [*u11* E] [*u22* E] [*u33* E] [*u12* E] [*u13* E] [*u23* E]: *adps* generates the *unn* atomic displacement parameters with considerations made for special positions. On termination of refinement the *adps* keyword is replaced with the *unn* parameters. Instead of using the *adps* keyword the *unn* parameters can be manually entered.

The *unn* matrix can be kept positive definite with the site dependent macro ADPs_Keep_PD; this can stabilize refinement. Examples:

```
site C1 ... occ C 1 adps ADPs_Keep_PD
```

```
site C1 ... occ C 1 ADPs_Keep_PD adps
```

[*layer* $layer_name]: *layer* identifies a site as belonging to a layer called $layer_name. The keyword *stack* can be used to apply a stacking vector (sx, sy, sz) to the named *layer*.

[mlx E] [mly E] [mlz E] [mg E]: *mlx*, *mly*, and *mlz* define the magnetic moment for a given site. The Lande splitting factor can be refined using the parameter *mg*.

[mag_only]: When defined the X-ray component to intensity for the site in question is ignored. For more details see section 9.12.

[*sites_distance* N][*sites_angle* N][*sites_flatten* N [*sites_flatten_tol* !E]]...
[*site_to_restrain* $site [#ep [#n1 #n2 #n3]]]...

When used in equations the name N of *sites_distance* and *sites_angle* returns the distance in Å between 2 sites and angle in degrees between 3 sites respectively. The sites are defined by *site_to_restrain*. In particular N can be used in penalty equations to restrain bond lengths.

N of *sites_flatten* returns a restraint term that decreases as the sites become coplanar; it is defined as

$$sites\_flatten = \frac{6}{n(n-1)(n-2)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} (\left|b_i \times b_j \cdot b_k\right| - tol)^2 \text{ , if } \left|b_i \times b_j \cdot b_k\right| > tol$$

where tol corresponds to *sites_flatten_tol*, n corresponds to the number of sites defined with the *site_to_restrain* keyword, b are Cartesian unit length vectors between the sites and the geometric centre of the sites.

#ep, #n1, #n2 and #n3 correspond to the site equivalent position and fractional offsets to add to the sites. This is useful if the structure is already known and restraints are required, for example, in the bond length output (*see append_bond_lengths*):

```
Zr1:0  O1:20   0  0 -1  2.08772
       O1:7    0 -1  0  2.08772  89.658
       O1:10   0  0 -1  2.08772  90.342  90.342
       O1:15  -1  0  0  2.08772 180.000  89.658  89.658
       O1:18  -1  0  0  2.08772  90.342  89.658 180.000  90.342
P1:0   O1:4    0  0  0  1.52473
       O1:8    0  0  0  1.52473 112.923
       O2:0    0  0  0  1.59001 105.749 105.749 105.749
```

Example restraints could look like the following:

```
Angle_Restrain(O1 P1 O1 8, 112, 112.92311, 0, 0.001)
```

```
Distance_Restrain(Zr1 O1 20 0 0 -1, 2.08, 2.08772, 0, 1)
```

Note, for more than around 6 sites then *sites_flatten* becomes computationally expensive.

[*sites_geometry* N]...
[*site_to_restrain* $site [#ep [#n1 #n2 #n3]]]...

sites_geometry defines a grouping of up to four sites; $Name is the name given to this grouping. The sites that are part of the group is defined using *site_to_restrain*, for example:

```
sites_geometry some_name
load site_to_restrain { C1 C2 C3 C4 }
```

Three functions, Sites_Geometry_Distance($Name), Sites_Geometry_Angle($Name) and Sites_Geometry_Dihedral_Angle($Name) can be used in equations to obtain the distance between sites C1 and C2, the angle between C1-C2-C3 and the dihedral angle formed between the planes C1-C2-C3 and C2-C3-C4. The convention used is the same as for z-matrices.

If $Name contains only two sites then only Sites_Geometry_Distance($Name) can be used. Three sites defined additionally allows the use of Sites_Geometry_Angle($Name) and four sites defined additionally allows the use of Sites_Geometry_Dihedral_Angle($Name).

[*siv_s1_s2 # #*]

Defines the S1 and S2 integration limits for the spherical interaction volume of the GRS series.

[*stack* $layer_name]...
[*sx* E] [*sy* E] [*sz* E]
[*generate_these* $sites]
[*generate_name_append* $append_to_site_name]

[*stack* $layer_name]...: Applies a stacking vector (sx, sy, sz) to a layer called $layer_name as defined using the keyword *layer* (see the sites keyword). Structure factors are generated in the usual manner; a shift corresponding to the stacking vector is then applied. *stack* operates in any space group. Sites that do not belong to a layer are treated as un-stacked and their structure factors are generated in the usual manner.

generate_these generates the sites found in $sites for the stack with coordinates that reflect original $sites positions plus the stacking vector. *generate_name_append* appends $append_to_site_name to the generated site. The generated sites have occupancies set to zero which signals a dummy site. Dummy sites do not take part in structure factor calculations and hence speed is not hindered. The dummy sites allow for graphical display of the layers.

Importantly penalties operate on dummy sites which allow restraints such as Distance_Restrain. For example,

```
space_group P1
site O1... layer A
site O2... layer A
stack A
   sx...
   generate_these O1
      generate_name_append _1
append_fractional
   in_str_format
```

will output for append_fractional the following:

```
site O1 …
site O2...
site O1_1 ... occ O 0
```

[*view_structure*]

Displays the crystal structure in the *Structure Viewer* window.

12.2.13 Thkl_lat

[*a* E] [*b* E] [*c* E] [*al* E] [*be* E] [*ga* E]

Lattice parameters in Angstroms and lattice angles in degrees.

[*normals_plot* !E]…
[*normals_plot_min_d* !E]

The keyword *normals_plot* opens a plot window to display lattice plane normals with the lengths of the normals defined by the keyword *normals_plot*. A typical application example is plotting of spherical harmonics coefficients, for example:

```
str...
   spherical_harmonics_hkl sh
   sh_order 8
   scale_pks = sh;                          ' Preferred orientation
   normals_plot = Abs(H * K + L^2) + 1;
      normals_plot_min_d .3
```

normals_plot_min_d is optional; smaller values than .3 increase the quality of the plot but can lead to extreme calculation times.

[*phase_penalties* $sites N [*hkl_Re_Im* #h #k #l #Re #Im]...]…
[*accumulate_phases_and_save_to_file* $file]
[*accumulate_phases_when* !E]

phase_penalties for a single hkl is defined as follows:

$$\mathrm{Pp_{hkl}} = \begin{cases} 0, & \text{if the phase } \phi_{s,hkl}-45° < \phi_c < \phi_{s,hkl}+45° \\ d\ \mathrm{I^2_{c,hkl}}(\phi_{s,hkl}-\phi_{c,hkl})^2, & \text{if the phase } \phi_c < \phi_{s,hkl}-45°, \text{or}, \phi_c > \phi_{s,hkl}+45° \end{cases}$$

where ϕ_s = assigned phase, ϕ_c = calculated phase, I_c = calculated intensity and d is the reflection d-spacing. The name *N* returns the sum of the *phase_penalties* and it can be used in equations and in particular *penalty* equations. ϕ_c is calculated from sites identified in $sites.

#h, #k, #l are user-defined hkls; they are used for formulating the phase penalties. #Re and #Im are the real and imaginary parts of ϕ_s. An example use of phase penalties is as follows:

```
penalty = pp1;
phase_penalties * pp1
load hkl_Re_Im
{
   0   1   2   1   0
   1   0  -2   1   0
   1  -2  -1   1   0
}
```

hkls chosen for phase penalties should comprise those that are of high intensity, large d-spacing and isolated from other peaks to avoid peak overlap. Origin defining hkls are typically chosen.

accumulate_phases_and_save_to_file saves the average phases collected to $file. Phases are collected when *accumulate_phases_when* evaluates to true; *accumulate_phases_when* defaults to true. Here's an example use:

```
temperature 1
temperature 1
temperature 1
temperature 1
temperature 10
...move_to_the_next_temperature_regardless_of_the_change_in_rwp
accumulate_phases_and_save_to_file RESULTS.TXT
accumulate_phases_when = T == 10;
```

Here phases with the best R_{WP} since the last accumulation are accumulated when the current temperature is 10.

[*omit_hkls* !E]

Allows for the filtering of hkls using the reserved parameter names of H, K, L and D_spacing. More than one omit_hkls can be defined, for example:

```
omit_hkls = If(And(H==0, K==0), 1, 0);
omit_hkls = And(H==0, K==1);
omit_hkls = D_spacing < 1;
```

[*spherical_harmonics_hkl* $name]...
[*sh_Cij_prm* $Yij E]...
[*sh_alpha* !E]
[*sh_order* #]

Defines a hkl dependent symmetrized spherical harmonics series (Järvinen, 1993). $name defines the name given to the series and when used in equations it returns the value of the spherical harmonics series. The expansion is simply a series that is a function hkl values. The series is normalized such that the maximum value of each component is 1. The normalized components are:

Y00 = 1
Y20 = (3.0 Cos(t)^2 - 1.0)* 0.5
Y21p = (Cos(p)*Cos(t)*Sin(t))* 2
Y21m = (Sin(p)*Cos(t)*Sin(t))* 2
Y22p = (Cos(2*p)*Sin(t)^2)
Y22m = (Sin(2*p)*Sin(t)^2)
Y40 = (3 - 30*Cos(t)^2 + 35*Cos(t)^4) *.1250000000
Y41p = (Cos(p)*Cos(t)*(7*Cos(t)^2-3)*Sin(t)) *.9469461818
Y41m = (Sin(p)*Cos(t)*(7*Cos(t)^2-3)*Sin(t)) *.9469461818
Y42p = (Cos(2*p)*(-1 + 7*Cos(t)^2)*Sin(t)^2) *.7777777778
Y42m = (Sin(2*p)*(-1 + 7*Cos(t)^2)*Sin(t)^2) *.7777777778
Y43p = (Cos(3*p)*Cos(t)*Sin(t)^3) *3.0792014358
Y43m = (Sin(3*p)*Cos(t)*Sin(t)^3) *3.0792014358
Y44p = (Cos(4*p)*Sin(t)^4)
Y44m = (Sin(4*p)*Sin(t)^4)
Y60 = (-5 + 105*Cos(t)^2 - 315*Cos(t)^4 + 231*Cos(t)^6) *.62500.0000
Y61p = (Cos(p)*(-5 + 30*Cos(t)^2 - 33*Cos(t)^4)*Sin(t)*Cos(t)) *.6913999628
Y61m = (Sin(p)*(-5 + 30*Cos(t)^2 - 33*Cos(t)^4)*Sin(t)*Cos(t)) *.6913999628
Y62p = (Cos(2*p)*(1 - 18*Cos(t)^2 + 33*Cos(t)^4)*Sin(t)^2) *.6454926483
Y62m = (Sin(2*p)*(1 - 18*Cos(t)^2 + 33*Cos(t)^4)*Sin(t)^2) *.6454926483
Y63p = (Cos(3*p)*(3- 11*Cos(t)^2)*Cos(t)*Sin(t)^3) *1.4168477165
Y63m = (Sin(3*p)*(3- 11*Cos(t)^2)*Cos(t)*Sin(t)^3) *1.4168477165
Y64p = (Cos(4*p)*(-1 + 11*Cos(t)^2)*Sin(t)^4) *.8167500000
Y64m = (Sin(4*p)*(-1 + 11*Cos(t)^2)*Sin(t)^4) *.8167500000
Y65p = (Cos(5*p)*Cos(t)*Sin(t)^5) *3.8639254683
Y65m = (Sin(5*p)*Cos(t)*Sin(t)^5) *3.8639254683
Y66p = (Cos(6*p)*Sin(t)^6)
Y66m = (Cos(6*p)*Sin(t)^6)
Y80 = (35 - 1260*Cos(t)^2 + 6930*Cos(t)^4 - 12012*Cos(t)^6 +6435*Cos(t)^8)* .0078125000

Y81p = (Cos(p)*(35*Cos(t) - 385*Cos(t)^3 + 1001*Cos(t)^5 -715*Cos(t)^7)*Sin(t))* .1134799545
Y81m = (Sin(p)*(35*Cos(t) - 385*Cos(t)^3 + 1001*Cos(t)^5 -715*Cos(t)^7)*Sin(t))* .1134799545
Y82p = (Cos(2*p)*(-1 + 33*Cos(t)^2 - 143*Cos(t)^4 + 143*Cos(t)^6)*Sin(t)^2)*.5637178511
Y82m = (Sin(2*p)*(-1 + 33*Cos(t)^2 - 143*Cos(t)^4 + 143*Cos(t)^6)*Sin(t)^2)*.5637178512
Y83p = (Cos(3*p)*(-3*Cos(t) + 26*Cos(t)^3 - 39*Cos(t)^5)*Sin(t)^3)*1.6913068375
Y83m = (Sin(3*p)*(-3*Cos(t) + 26*Cos(t)^3 - 39*Cos(t)^5)*Sin(t)^3)*1.6913068375
Y84p = (Cos(4*p)*(1 - 26*Cos(t)^2 + 65*Cos(t)^4)*Sin(t)^4)* .7011002983
Y84m = (Sin(4*p)*(1 - 26*Cos(t)^2 + 65*Cos(t)^4)*Sin(t)^4)* .7011002983
Y85p = (Cos(5*p)*(Cos(t) - 5*Cos(t)^3)*Sin(t)^5)* 5.2833000817
Y85m = (Sin(5*p)*(Cos(t) - 5*Cos(t)^3)*Sin(t)^5)* 5.2833000775
Y86p = (Cos(6*p)*(-1 + 15*Cos(t)^2)*Sin(t)^6)* .8329862557
Y86m = (Sin(6*p)*(-1 + 15*Cos(t)^2)*Sin(t)^6)* .8329862557
Y87p = (Cos(7*p)*Cos(t)*Sin(t)^7)* 4.5135349314
Y87m = (Sin(7*p)*Cos(t)*Sin(t)^7)* 4.5135349313
Y88p = (Cos(8*p)*Sin(t)^8)
Y88m = (Sin(8*p)*Sin(t)^8)

where t = theta and p = phi, representing the spherical coordinates of the normal to the hkl plane.

[*sh_Cij_prm* $Yij E]: Spherical harmonics coefficients, which can be defined by the user; alternatively if there are no coefficients defined then the *sh_Cij_prm* parameters are generated; only the coefficients allowed by the selection rules of the point group are generated (Järvinen, 1993). At the end of refinement the generated *sh_Cij_prm* parameters are appended to *sh_order*. This allows full control over the *sh_Cij_prm* parameters in subsequent refinements. $Yij corresponds to valid symmetrized harmonics that has survived after symmetrization. It is internally generated when there are no *sh_Cij_prm* parameters defined by the user.

[*sh_alpha* !E]: Corresponds to the angle in degrees between the polar axis and the scattering vector; *sh_alpha* defaults to zero degrees which is required for symmetric reflection as is the case for Bragg-Brentano geometry.

[*sh_order* #]: *sh_order* corresponds to the order of the spherical harmonic series which are even numbers ranging from 2 to 8 for non-cubic and from 2 to 10 for cubic systems.

The PO_Spherical_Harmonics macro simplifies the use of *spherical_harmonics_hkl*.

[*str_hkl_angle* N h k l]...

Defines a parameter name and a vector normal to the plane defined by h, k and l. When the parameter name is used in an equation, it returns angles (in radians) between itself and the normal to the planes defined by hkls.

str_hkl_angle is used for example in the Preferred_Orientation macro.

12.2.14 Trigid

[*rigid*]...
[*point_for_site* $site_name [*ux*|*ua* E] [*uy*|*ub* E] [*uz*|*uc* E]]...
[*in_cartesian*] [*in_FC*]
[*z_matrix* $atom_1 [$atom_2 E] [$atom_3 E] [$atom_4 E]] …
[*rotate* E [*qx*|*qa* E] [*qy*|*qb* E] [*qz*|*qc* E]]...
[*operate_on_points* $sites]
[*in_cartesian*] [*in_FC*]
[*translate* [*tx*|*ta* E] [*ty*|*tb* E] [*tz*|*tc* E]]...
[*operate_on_points* $sites]
[*rand_xyz* #displacement]
[*in_cartesian*] [*in_FC*]
[*start_values_from_site* $unique_site_name]

rigid defines a rigid body and associated translation and rotation operations. These operations are capable of creating and manipulating rigid bodies with hinges (torsion angles).

[*point_for_site* $site_name [*ux*|*ua* E] [*uy*|*ub* E] [*uz*|*uc* E]]... : Defines a point in space with Cartesian coordinates given by the parameters *ux*, *uy*, *uz*. Fractional equivalents can be defined using *ua*, *ub* and *uc*. $site_name is the site that the *point_for_site* represents.

[*z_matrix* $atom_1 [$atom_2 E] [$atom_3 E] [$atom_4 E]]… : Defines a point in space with coordinates given in Z-matrix format. The Z-matrix format is as follows:

- E can be an equation, constant or a parameter name with a value.
- $atom_1 specifies the site that the new Z-matrix point represents.
- [$atom_2 E]: E specifies the distance in Å between $atom_2 and $atom_1. $atom_2 must exist if $atom_1 is preceded by at least one point.
- [$atom_3 E]: E specifies the angle in degrees between $atom_3-$atom_2- $atom_1. $atom_3 must exist if $atom_1 is preceded by at least two points.
- [$atom_4 E]: E specifies the dihedral angle in degrees between the plane formed by $atom_3-$atom_2-$atom_1 and the plane formed by $atom_4-$atom_3-$atom_2. This angle is drawn using the right hand rule with the thumb pointing in the direction $atom_3 to $atom_2. $atom_4 must exist if $atom_1 is preceded by at least three sites of the rigid body.
- If $atom_1 is the first point of the rigid body then it is placed at Cartesian (0, 0, 0). If $atom_1 is the second point of the rigid body then it is placed on the positive z-axis at Cartesian (0, 0, E) where E corresponds to the E in [$atom_2 E]. If $atom_1 is the third point of the rigid body then it is placed in the x-y plane.

[*rotate* E [*qx*|*qa* E] [*qy*|*qb* E] [*qz*|*qc* E]]... : Rotates *point_for_site*'s an amount as defined by the rotate E equation around the vector defined by the Cartesian vector *qx*, *qy*, *qz*. The vector can be defined in fractional coordinates using *qa*, *qb* and *qc* instead.

[*translate* [*tx*|*ta* E] [*ty*|*tb* E] [*tz*|*tc* E]]... : Performs a translation of *point_for_site*'s an amount in Cartesian equal to *tx*, *ty*, *tz*. The amount can be defined in fractional coordinates using *ta*, *tb* and *tc* instead.

[*in_cartesian*] [*in_FC*]: If *in_cartesian* or *in_FC* is defined then the coordinates are interpreted as Cartesian or fractional atomic coordinates, respectively.

[*operate_on_points* $sites]: By default rotate and translate operates on any previously defined *point_for_site*'s; alternatively, *point_for_site*'s operated on can be identified using the *operate_on_points* keyword. The *operate_on_points* keyword must refer to previously defined *point_for_site*'s.

[*rand_xyz* !E]: If *continue_after_convergence* is defined then *rand_xyz* is executed at the end of a refinement cycle. It introduces a random displacement to the translate fractional coordinates (tx, ty, tz)

that are independent parameters. The displacement is given by a vector u, the direction of which is random and the magnitude in Å is:

|u| = T *rand_xyz*

where T is the current *temperature*. Thus to add a shift to a rigid body between 0 and 1 Å the following could be used:

```
temperature 1
rigid ...
rand_xyz = Rand(0,1);
```

[*start_values_from_site* $unique_site_name]: initializes the values *ta*, *tb*, *tc* with corresponding values taken from the site $unique_site_name.

See section 9.9.1 for a description of rigid bodies.

12.2.15 Tout_record

[*out_record*]
[*out_eqn* !E]
[*out_fmt* $c_fmt_string]
[*out_fmt_err* $c_fmt_string]...

out_record provides a means of defining individual (result) files.

out_eqn defines the equation or parameter to be written to a file using the *out_fmt*. $c_fmt_string describes a format string in c syntax containing a single format specified for a double precision number. *out_fmt_err* defines the $c_fmt_string used for formatting the error of *out_eqn*.

Both *out_fmt* and *out_fmt_err* require an *out_eqn* definition. *out_fmt* can be used without *out_eqn* for writing strings. The order of *out_fmt* and *out_fmt_err* determines which is written to file first; thus if *out_fmt_err* is defined first then it will be operated on first.

Keywords using *out_record* are *out*, *phase_out*, *atom_out*, and *xdd_out*; TOPAS.INC contains numerous macros demonstrating the usage of these keywords, for examples macros section 13.3.10.

12.2.16 Tmin_r_max_r

[*min_r* #] [*max_r* #]

Defines the minimum and maximum radii for calculating bond lengths.

min_r and *max_r* are by default set to 0 and 3.2 Å respectively.

12.2.17 Tspace_group

[space_group $symbol]

space_group $symbol is used to define the space group. For details see section 9.2.

12.2.18 Miscellanous

[*aberration_range_change_allowed* !E]

Describes the maximum change allowed in the x-axis extent of a convolution aberration before a new peak is calculated for the peaks buffer.

[*default_I_attributes* E]

Changes the attributes of the I parameter, for example,

```
xo_Is
   default_I_attributes 0 min 0.001 val_on_continue 1
```

Useful when randomising lattice parameters during refinements using the Le Bail method with *continue_after_convergence*.

12.2.19 Tindexing

[dummy]

Allows for the exclusion of columns in data

In this example all columns except "2Th" and "Area" will be ignored:

```
load index_th2 dummy dummy index_I dummy  {
'  2Th      d (A)       Height      Area        FWHM
   1.724    26.50645    2758.3      23303.7     0.0450
   2.646    17.27733    150393.8    747063.6    0.0250
   3.235    14.13204    98668.8     493153.7    0.0250
...}
```

[*index_lam* E]

Defines the wavelength in Å

[**index_max_number_of_solutions**]

The number of best solutions to keep, the default is set to 1000.

[*index_max_Nc_on_No* E]

Determines the maximum ratio of the number of calculated to observed lines. The default value of 5 allows for up to 80% of missing lines. *index_max_Nc_on_No* may need to be increased for extreme dominant zone cases.

[*index_max_th2_error* E]

Used for determining impurity lines (unindexed lines UNI in *.NDX files). Large values, 1 for example, forces the consideration of more observed input lines. For example if it is known that there are none or maybe just one impurity line then a large value for *index_max_th2_error* will speed up the indexing procedure. The default is set to 0.05.

[*index_max_zero_error #*]

Excludes solutions with zero errors greater than *index_max_zero_error*. The default is set to 0.2.

[*index_min_lp* E] [*index_max_lp* E]

Defines the minimum and maximum allowed lattice parameters. Typically the maximum is determined automatically. The *index_min_lp* default is set to 2.5.

[*index_th2* E]... or [*index_d* E]…
[*index_I* E [*good*]]

index_th2 or *index_d* defines a reflection entry in 2θ degrees or d-spacing in Å.

[*index_I*]: Typically set to the area under the peak; it is used to weight the reflection. The default is set to 1.

[*good*]: Indicates that the corresponding d-spacing is not an impurity line. A single use of good on a high d-spacing decreases the number of possible solutions and hence speeds up the indexing process

The "*load* { }" keyword can be used to simplify the use of *index_th2* or *index_d*, e.g.

```
load index_d {
   8.912    good
   7.126
   4.296
   ...
}
```

or

```
load index_th2 index_I {
   8.8656   6.672077
   9.7922   15.5885     good
   10.5663  28.61461
   ...
}
```

[*Index_x0* E]

Defines x0 in the reciprocal lattice equation:

$$(X_{hh}\,\mathrm{h}^2+X_{kk}\,\mathrm{k}^2+X_{ll}\,\mathrm{l}^2+X_{hk}\,\mathrm{hk}+X_{hl}\,\mathrm{h\,l}+X_{kl}\,\mathrm{k\,l} + Ze\,(\pi/360)(4/\lambda^2)\sin(2\theta))\,\mathrm{W_{hkl}} = \mathrm{W_{hkl}}/d_{\mathrm{o}}^2$$

In a triclinic lattice the highest d-spacing can probably be indexed as 100 or 200 etc. Thus

```
index_x0 = 1/(dmax)^2;
```

significantly speeds up the indexing process (if the first line can be indexed as 100) and additionally the chances of finding the correct solution is greatly enhanced. Note that the data is in 2Th degrees then the following can be used:

```
index_x0 = (2 Sin(2Thmin Pi/360) / wavelength))^2;
```

The two macros Index_x0_from_d and Index_x0_from_th2 simplify the use of *index_x0*, see section 13.3.16.2.

[*index_zero_error*]

Includes a zero error

[*seed*]

Seeds the random number generator

[*try_space_groups* $symbol ...] …
[*X_scaler #*]
[*X_angle_scaler #*]

Defines the space groups to be searched. At the end of a run higher symmetry space groups for the Bravais lattices corresponding to the 10 best solutions is subsequently searched.

[*X_scaler #*]: A scaling factor used for determining the number of steps to search in parameter space. *X_scaler* needs to be less than 1. Increasing *X_scaler* searches more of parameter space. Defaults are:

Cubic	0.99
Hexagonal/Trigonal	0.95
Tetragonal	0.95
Orthorhombic	0.89
Monoclinic	0.85
Triclinic	0.72

[*X_angle_scaler #*]: A scaling factor for determining the number of angular steps for monoclinic and triclinic space groups. Small values, 0.05 for example, increases the number of angular steps. The default value of 0.1 is usually sufficient.

A series of macros described in section 13.3.16.1 simplifies the use of *try_space_groups*

12.2.20 Tcharge_flipping

Equations appearing in Charge Flipping keywords can be functions of the items shown in Table 12.1. At the end of a charge flipping process a file with the same name as that given by *cf_hkl_file* is created but with a *.FC extension. Almost all of the Charge Flipping keywords can be equations allowing for great flexibility in regards to changing resolution etc. on the fly.

Table 12.1: Items that can be used in Charge Flipping equations.

Item	Remarks
Functions	
Get(Aij) Get(alpha_sum) Get(density) Get(cycles_since_last_best) Get(d_squared_inverse) Get(initial_phase) Get(iters_since_last_best) Get(F000) Get(max_density) Get(max_density_at_cycle_iter_0) Get(num_reflections_above_d_min) Get(phase_difference) Get(r_factor_1), Get(r_factor_2) Get(threshold)	These are updated internally each charge-flipping iteration or cycle or when needed.
Reserved parameters names	
Cycle_Iter, Cycle, Iter, D_spacing	

12.2.20.1 General

[*a* E] [*b* E] [*c* E] [*al* E] [*be* E] [*ga* E]

Lattice parameters in Angstroms and lattice angles in degrees. The defaults are *al* = *be* = *ga* = 90.

[*break_cycle_if_true* !E]

Interrupts charge flipping to execute *randomize_phases_on_new_cycle_by*. Cycle_Iter is set to zero and Cycle is incremented.

[*cf_hkl_file* $file]

Defines the input hkl file.

[*cf_in_A_matrix* $file [*scale_Aij* !E]]

Data input is from a file created using *out_A_matrix* from a previous Pawley refinement. The correlations in $file are used to partition intensities during each iteration of charge-flipping. This partitioning is applied to structure factors as used by CF and as used by the tangent formula.

scale_Aij can be used to modify the A matrix off-diagonal coefficients, here are some examples:

```
scale_Aij = Get(Aij);

scale_Aij = Get(Aij)^2;    ' the default

scale_Aij = 0;             ' Equivalent to using a Pawley
                           ' generated hkl file
```

CF on powder data can also be initiated using standard hkl files.

[*delete_observed_reflections* !E]

Reflections are deleted before entering Charge Flipping according to *delete_observed_reflections*; it can be a function of D_spacing, for example:

```
delete_observed_reflections = D_spacing < 1.1;
```

Once deleted, observed reflections cannot be reinstated by changing *min_d*.

[*extend_calculated_sphere_to* !E]

Used to sharpen electron density clouds by filling in missing reflections; added reflections are given the status of "weak". *extend_calculated_sphere_to* can be used in conjunction with *scale_weak_reflections* and *add_to_phases_of_weak_reflections* to modify "weak" reflection magnitudes and phases respectively; here's an example:

```
extend_calculated_sphere_to 1
add_to_phases_of_weak_reflections = If(Rand(0, 1) < .3, 90, 0);
```

[*f_atom_type* $type *f_atom_quantity* !E]…

Defines atom types and number of atoms within the unit cell; used by the tangent formula in determining Eh values and by the OpenGL display for picking atoms. For the tangent formula then relative quantities are important.

[*find_origin* !E]

If defined and non-zero then origin finding is turned ON. *symmetry_obey_0_to_1* defines *find_origin* by default. *symmetry_obey_0_to_1* can be used without *find_origin* by defining and setting *find_origin* to zero.

[fraction_density_to_flip !E]

The amount of charges flipped is fractionally based. The default value of 0.75, for example, sets the threshold δ such that the sign of the lowest 75% of charge is changed. Get(threshold) can be used to retrieve δ.

[*fraction_reflections_weak* !E]

Defines the fraction of observed reflections to flag as "weak". The default is set to zero. When *scale_weak_reflections*, *add_to_phases_of_weak_reflections* and *extend_calculated_sphere_to* are all not defined then intensities of weak reflections are set to zero effectively removing them from the charge flipping process. Otherwise intensities of weak reflections are not set to zero; instead they are left untouched prior to *scale_weak_reflections* and *add_to_phases_of_weak_reflections* and space group family averaging.

[*min_d* !E]

Determines in Angstroms the resolution of observed reflections to work with; only observed reflections with a d-spacing above *min_d* are considered. The default is set to zero. *min_d* is evaluated each CF iteration. Get(*num_observed_reflections_above_d_min*) is updated when a change in *min_d* is detected. See also *extend_calculated_sphere_to* and *delete_observed_reflections*.

[*min_grid_spacing* !E]

If defined then the grid spacing used is set to the smaller of *min_d*/2 and *min_grid_spacing*; useful for obtaining many grid points for graphical purposes.

[*neutron_data*]

Signals that the input data is of neutron type. Used in the picking of atoms and additionally Eh values are not corrected from any defined *f_atom_type* and *f_atom_quantity keywords*.

[*space_group* $]

If defined then the *cf_hkl_file* is assumed to comprise merged hkls corresponding to the defined space group; otherwise the *cf_hkl_file* is assumed to be of space group type P1.

[*use_Fc*]

Sets initial phases to those saved in a previous *.FC file. The FC file used corresponds to the same name as the data file, defined using *cf_hkl_file* or *cf_in_A_matrix*, but with a FC extension. *use_Fc* is similar to *set_initial_phases_to* except that the file used is implied.

12.2.20.2 Electron density perturbations

[*flip_equation* !E]

Allows for a user defined flip; here's an example:

```
flip_equation =
If(Get(density) < Get(threshold), -Get(density), Get(density));
```

[*flip_regime_2* !E]

The electron density is modified according to equation (9.16) and then further modified using:

$$\rho = \rho - \text{Get(flip\_regime\_2)}\,\rho^3 / \rho_{\max}^2$$

flip_regime_2 is typically ramped from 1 to 0.

[*flip_regime_3* !E]

The electron density is modified according to equation (9.16) and then further modified using:

$$\rho = \begin{cases} \rho, & \text{for } \rho < \delta \\ \rho = \text{Min}(\rho, \rho_{\max}\ \text{Get(flip\_regime\_3)}) & \text{for } \rho \geq \delta \end{cases}$$

A value of 0.5 for *flip_regime_3* introduces little perturbation whilst reducing the occurance of uranium atom solutions. It is recommended that *flip_regime_3* be used in cases where *flip_regime_2* produces uranium atom solutions. An additional perturbation, such as

```
add_to_phases_of_weak_reflections=90;
```

may be necessary.

[*histogram_match_scale_fwhm* !E]
[*hm_size_limit_in_fwhm* !E]
[*hm_covalent_fwhm* !E]

An implementation of Histogram Matching (Baerlocher et al., 2007b) where the distribution of pixels within the unit cell is restrained to one that mathes Gaussian atoms with intensities corresponding to the atoms defined by *f_atom_type*'s. The Histogram Matching operation is performed when *histogram_match_scale_fwhm* evaluates to a non-zero value. Subsequently the FWHM of the Gaussians (obtained from the file ATOM_RADIUS.DEF) is scaled by *histogram_match_scale_fwhm*. *hm_size_limit_in_fwhm* corresponds to the extent to which the Gaussians are calculated in units of FWHM. Covalent radii is used if *hm_covalent_fwhm evaluates* to a non-zero value otherwise ionic radii is used. An example use is as follows:

```
histogram_match_scale_fwhm = If(Mod(Cycle_Iter, 3), 0, 1);
   hm_size_limit_in_fwhm 1
   hm_covalent_fwhm 1
```

Reported on is the fraction of pixels modified; values of 1 for both *histogram_match_scale_fwhm* and *hm_size_limit_in_fwhm* seem optimal where typically ~15 to 20% of pixels are modified. Use of histogram matching should produce R-factors at convergence that are equal to or than less R-factors produced when not using histogram matching. Histogram matching sharpens the electron density cloud for data at poor resolution.

[*pick_atoms* $atoms]...
[*activate* !E]
[*choose_from* !E] [*choose_to* !E]
[*choose_randomly* !E]
[*omit* !E]
[*displace* !E]
[*insert* !E]

pick_atoms modifies the electron density based on picked atoms. $atom corresponds to the atom types to be operated on; it can contain the wild card character '*' and the negation character '!'. The operations of *pick_atoms* are invoked when *activate* evaluates to a non-zero value; for example,

```
pick_atoms "O C"
activate = Mod(Cycle_Iter, 20) == 0;
```

The picking routine will attempt to locate the atom types found in $atom based on the intensities of picked atoms and the scattering power of the atoms defined in *f_atom_type*. For example,

```
load f_atom_type f_atom_quantity { Ca 2 O 10 C 12 }
pick_atoms "O C"
```

Here 2 Ca atoms are first picked and then 10 O atoms and then 12 C atoms. The picked atoms operated on will be the O and C atoms with the Ca atoms ignored.

choose_from and *choose_to* can be used to limit the number of atoms operated on. Note, that picked atoms within a particular *pick_atoms* are sorted in decreasing intensity order. For example, to not operate on the first three O atoms and the last 2 C atoms then the following could be used:

```
choose_from 4
choose_to 20
```

choose_randomly further reduces the atoms operated on and is executed after *choose_from* and *choose_to*.

omit removes operated on atoms from the electron density. Atoms can be partially removed by setting *omit* to values less than 1. Values greater than 1 can also be used, the effect is to change the sign of the electron density. *omit* operating on a few of the highest intensity atoms is an extremely effective means of preventing the occurance of uranium atom solutions; for example:

```
pick_atoms *
   choose_to 5
   omit = Rand(1, 1.1);
```

Omitting atoms randomly is a technique referred to as "random omit maps" in ShelXD, (Schneider and Sheldrick, 2002).

insert inserts operated on atoms; a value of 1 inserts the atoms with an intensity that is equal to the average of the picked atoms. Values of less than 1 decrease the intensity of the inserted atoms. When insert is defined then *omit* is internally defined if it does not already exist. Thus, atoms are removed before insertion by default.

displace (in Angstroms) displaces atom positions from their picked positions; it is evaluated before *insert*. For example, to randomly displace atoms by 0.3 Angstroms then the following could be used:

```
displace = Rand(0.4, 0.6);
insert 1
```

There can be more than one occurance of *pick_atoms*, for example to limit uranium atom solutions then the follow can be used:

```
pick_atoms *
choose_to 5
insert = Rand(-.1, 1);
```

To further randomly remove ~33% of atoms then the following could additionally be used:

```
break_cycle_if_true = Get(iters_since_last_best) > 10;
pick_atoms *
activate = Cycle_Iter == 0;
insert = If(Rand(0, 1) > 0.33, 10, 0);
```

Note that in this example atoms are inserted at ten times the average picked intensity; this simply gives more weight to picked atoms relative to electron density noise. Additionally weak reflections are also given more weighting.

[*scale_density_below_threshold* !E]

Electron density pixels that are less than the threshold value are scaled by *scale_density_below_threshold*. Values for *scale_density_below_threshold* that are less than 1 tend to sharpen the electron density and to reduce large oscillations in R-factors; the latter occurs for poor quality data. A value of zero for *scale_density_below_threshold* results in "low density elimination" simlar to that of Shiono & Woolfson (1992).

[*symmetry_obey_0_to_1* !E]

If a space group is defined then symmetry is adhered to according to *symmetry_obey_0_to_1*. *symmetry_obey_0_to_1* can have values ranging between 0 and 1. If 1 then symmetry is obeyed 100%.

12.2.20.3 Phase perturbations

[*add_to_phases_of_weak_reflections* !E]

Allows for modification to phases of weak reflections. For example, to add $\pi/2$ to the phases of weak reflections then the following could be used:

```
add_to_phases_of_weak_reflections 90´
```

When *add_to_phases_of_weak_reflections* is defined then the intensities of weak reflections are not set to zero; instead they are left untouched meaning that their intensities are set to the values as determined by the inverse Fourier transform. See also *scale_weak_reflections*.

[*randomize_phases_on_new_cycle_by* !E]

Example:

```
randomize_phases_on_new_cycle_by = Rand(-180, 180); the default is set to
zero.
```

[*set_initial_phases_to* $file]
[*modify_initial_phases* !E]

Sets initial phases to those appearing in $file. Typically $file corresponds to a *.FC file saved in a previous charge-flipping process. *modify_initial_phases* is executed each iteration of Charge Flipping; it can be used to restrain the phases of $file. For example,

```
modify_initial_phases =
   Get(initial_phase) + Min(Abs(Get(phase_difference)), 45);
```

where *phase_difference* corresponds to the difference between the current phase and the initial phase; it has a value between ±90º. *modify_initial_phases* can be used to constrain phases to those as determined by HRTEM (Baerlocher et al., 2007a).

[*tangent_num_h_read* !E]
[*tangent_num_k_read* !E]
[*tangent_num_h_keep* !E]
[*tangent_max_triplets_per_h* !E]
[*tangent_min_triplets_per_h* !E]
[*tangent_scale_difference_by* !E]

tangent_num_h_read and *tangent_num_k_read* defines the number of highest h and highest k reflections to read in determining triplets.

tangent_num_h_keep defines the number of highest h reflections to include for tangent formula updating.

tangent_max_triplets_per_h and *tangent_min_triplets_per_h* defines the maximum and minimum number of triplets per reflection h; defaults are set to 30 and 1, respectively. Reflections with less that *tangent_min_triplets_per_h* are not included for tangent formula updating.

tangent_scale_difference_by corresponds to S in the following:

$$\phi_{h,new} = \phi_{h,cf} + S\,\alpha_h\left(\phi_{h,tf} - \phi_{h,cf}\right)$$

$$\tan(\phi_{h,tf}) = T_h / B_h$$

$$T_h = \sum_k E_h E_k E_{h-k} \sin\left(\phi_k + \phi_{h-k}\right)$$

$$B_h = \sum_k E_h E_k E_{h-k} \cos\left(\phi_k + \phi_{h-k}\right)$$

$$\alpha_h = M_h / M_{h,\max}\ ,\ M_h = \sqrt{T_h^2 + B_h^2}$$

The default is set to 1.

12.2.20.4 Miscellaneous

[*apply_exp_scale* !E]

Determines a and b each CF iteration such that the following is a minimum:

R-factor = ∑| a Exp(b / D_spacing^2) Fc – Fo |

where Fc and Fo are the calculated and observed moduli respectively. Use of *apply_exp_scale* corrects R-factors in case of an incorrect temperature factor correction as applied when normalizing structure factors. Use of *apply_exp_scale* typically increases the difference between R-factors prior to and at convergence. *apply_exp_scale* is used by default, setting it to zero prevents its use.

[*correct_for_atomic_scattering_factors* !E]

Structure factors are normalized when non-zero and when *f_atom_type*'s are defined. By default structure factors are normalized.

[*correct_for_temperature_effects* !E]

Attempts to remove isotropic temperature effects from the structure factors. *correct_for_temperature_effects* is ON by default, setting it to zero will prevent the correction. Normalized structure factors are realized when *correct_for_temperature_effects* is ON and the unit cell contents is defined using *f_atom_type* and *f_atom_quantity*.

[hkl_plane $hkl] ...

Used by the OpenGL viewer to display hkl planes, see the CeO2.STR file in the "Rigid" directory. Here are some examples:

```
str…
   hkl_plane 1 1 1
   hkl_plane "2 -2 0"
```

[*randomize_initial_phases_by* !E]

Initializes phases. To start a process with already saved phase information in a file file.fc then the following could be used:

```
set_initial_phases_to file.fc
randomize_initial_phases_by 0   ' this has a default of Rand(-180,180)
```

[*scale_E* !E]

Normalized structure factors (Eh values) are a function of *correct_for_temperature_effects* and unit cell contents. *scale_E* allows for an additional scaling of Eh values. The default is set to 1.

[*scale_F* !E]

Charge Flipping works with normalized structure factors by default. *scale_F* is an additional scaling of structure factors. The default *scale_F* broadens electron density peaks to avoid pixilation effects and is given by:

```
scale_F = Exp(-0.2 Get(d_squared_inverse));
```

The default is set to 1.

[*scale_F000* !E]

Scale should be set to 1 for compliance with the algorithm of Oszlányi & Süto (2004). When *scale_F000* is non_zero then modifications to the electron density produces unfavourable effects.

[*scale_weak_reflections* !E]

By default weak reflection structure factors are set to zero; however when either *scale_weak_reflections* or *add_to_phases_of_weak_reflections* is defined then weak reflections structure factors are instead modified accordingly, for example:

```
scale_weak_reflections = Rand(-0.2, 0.4);
```

scale_weak_reflections or *add_to_phases_of_weak_reflections* can be a function of D_spacing.

[*user_threshold* !E]

By default Get(threshold) is determined using *fraction_density_to_flip*. When defined then *user_threshold* overrides *fraction_density_to_flip*. Electron density pixels are normalized to have a maximum value of 1, thus typical values for *user_threshold* range between 0 and 0.1.

[*verbose #*]

A value of 1 outputs text in a verbose manner. A value of 0 outputs text only when a R-factor less that a previous value is encountered within a particular Cycle.

12.2.20.5 GUI related

[*add_to_cloud_N* !E [*add_to_cloud_when* !E]]

The current cloud is added to the GUI cloud creating a running average cloud for display purposes. *add_to_cloud_N* corresponds to the number of most recent clouds to include in the running average. *add_to_cloud_when* determines when the current cloud is to be included in the running average; here's an example:

```
add_to_cloud_N 10
   add_to_cloud_when = Mod(Cycle_Iter, 2);
```

Averaged clouds eliminate noise and is effective if the cloud remains stationery which is generally the case. Note that *add_to_phases_of_weak_reflections* can produce translations of the cloud and should not be included when averaging clouds.

[*pick_atoms_when* !E]

Atoms are picked in the Structure Viewer window when *pick_atoms_when* evaluates to a non-zero value; here's an example:

```
pick_atoms_when = Mod(Cycle_Iter + 1, 10) == 0;
```

Note that picking can be manually initiated from the Cloud dialog of the Structure Viewer. A text description of picked atoms can be obtained by opening the "Temporary output" text window of the Structure Viewer window.

[*view_cloud* !E]

Displays the electron density in the Structure Viewer window. Here are some examples:

```
view_cloud 1 ' Update cloud every charge-flipping iteration
view_cloud = Mod(Cycle_Iter, 10) == 0;
```

12.3 Keywords for simplified user input

12.3.1 The "load { }" keyword and attribute equations

The "*load* { }" keyword can be used to simplify user input. It allows the loading of keywords of the same type by typing the keywords once, for example, the *exclude* keyword in the following input segment:

```
xdd...
      exclude 20 22
      exclude 32 35
      exclude 45 47
```

can be rewritten using "*load* { }" as follows:

```
xdd...
      load exclude { 20 22 32 35 45 47 }
```

This input can be further simplified using the Exclude macro:

```
Exclude { 20 22 32 35 45 47 }
```

In some cases attribute equations are loaded by the parameter itself. For example, in the following:

```
prm t 0.01  val_on_continue = Rand(-Pi, Pi); min 0.4 max 0.5
```

the *prm* will actually load the attribute. But, in the following:

```
load prm val_on_continue min max { t 0.01 = Rand(-Pi, Pi); 0.4 0.5 }
```

the *load* keyword will load the attributes.

The following is valid:

```
load sh_Cij_prm
{
   y00 !sh_c00 1
   y20  sh_c20 0.26202642  min 0 max 1
   y40  sh_c40 0.06823548
   ...
}
```

In this case the *load* keyword does not contain *min*/*max* and the parameter will load its attributes. The load keyword however can contain attributes, for example:

```
load sh_Cij_prm min max
{
   y00 !sh_c00 1           0 1
   y20  sh_c20 0.26202642  0 1
   y40  sh_c40 0.06823548  0 1
   ...
}
```

In this case the attributes must be entered for all load entries.

12.3.2 The "move_to $keyword" keyword

The *move_to* keyword provides a means of entering parameter attributes without having to first load the parameter name and value, see for example the Keep_Atom_Within_Box macro. The site dependent ADPs_Keep_PD macro defines *min*/*max* limits; here's part of that macro:

```
move_to u12 min = -Sqrt(Get(u11) Get(u22)); max = Sqrt(Get(u11) Get(u22));
```

The $keyword of *move_to* can be any keyword and not just a parameter keyword.

12.3.3 The "for xdds { }" and "for strs { }" constructs

The for "xdds { }" and "for strs { }" constructs simplify the construction of input files when there are multiple diffraction patterns with similar structures. For example, two diffraction patterns of the same structures can be composed as follows:

```
xdd ...                          first xdd
      bkg ...                          xdd dependent keywords, not common to all xdd's
      th2_offset ...
      ...
      str...                           first str
            scale ...                        str dependent keywords, not common to all
            ...                              str's
```

```
xdd ...                          second xdd
      bkg ...                          xdd dependent keywords, not common to all xdd's
      th2_offset ...
      ...
      str...                           second str
            scale ...                        str dependent keywords, not common to
            ...                              all str's
```

```
for xdds {
      Slit_Width(.2)                   xdd dependent keywords, common to all xdd's
      CuKa2
      ...
      for strs {                       str dependent keywords, common to all str's in
            ...                        all xdd's
            }
      for strs 1 to 1 {                str dependent keywords, specific to the first str
            space_group p1
            site ...
      }
}
```

For obvious reasons, parameters should not be given the @ name within *for* constructs. The effect this would have is to give unique parameter names to the parameters iterated over; the output file would contain the parameter value corresponding to the last “for xdds” or “for strs” iteration.

13 MACROS AND INCLUDE FILES

INP files are pre-processed. The pre-processor comprises directives beginning with the character # and macros. Macros are used for grouping keywords. The directives are of two types, global directives and directives that are invoked on macro expansion, they are:

- Directives with global scope
  - macro $user_defined_macro_name { … }
  - #include $user_defined_macro_file_name
  - #delete_macros { $macros_to_be_deleted }
  - #define, #undef, #ifdef, #else, #endif
- Directives invoked on macro expansion
  - #m_ifarg, #m_else, #m_endif
  - #m_code, #m_eqn, #m_code_refine, #m_one_word
  - #m_argu, #m_first_word, #m_unique_not_refine

13.1 The macro directive

Macros are defined using the macro directive; here's an example:

```
macro Cubic(cv)
{
   a   cv
   b = Get(a);
   c = Get(a);
}
```

Macros can have multiple arguments or none at all; the Cubic macro above has one argument; here are some examples of the use of Cubic:

```
Cubic(4.50671)

Cubic(a_lp 4.50671  min 4.49671  max 4.52671)

Cubic(!a_lp 4.50671)
```

The first instance defines the a, b and c lattice parameters without a parameter name. The second defines the lattice parameters with a name indicating refinement of the a_lp parameter. In the third example, the a_lp parameter is preceded by the character "!". This indicates that the a_lp parameter is not to be refined; it can however be used in equations. The definition of macros need not precede its use. For example, in the segment:

```
xdd...
   Emission_Profile ' this is expanded
   macro Emission_Profile { CuKa2(0.001) }
```

Emission_Profile is expanded to "CuKa2(0.001)" even though Emission_Profile has been defined after its use.

Macro names do need not be unique; in cases where more than one macro have the same name then the actual macro expanded is determined by the number of arguments. For example, if the macro Slit_Width(.1) is used then the Slit_Width(v) macro is expanded. On the other hand if the macro Slit_Width(sw, .1) is used then the Slit_Width(c, v) macro is expanded.

Macros can also expand to other macro names. For example, the macro Crystallite_Size expands to CS and since CS is a macro then the CS macro will be expanded.

13.1.1 Directives with global scope

#include $user_defined_macro_file_name

Include files are used to group macros. The file TOPAS.INC contains standard macros; a good place to view examples. Text within include files are inserted at the position of the #include directive, thus the following:

```
#include "my include file.inc"
```

inserts the text within "my include file.inc" at the position of the #include directive. The standard macro file TOPAS.INC is always included by default.

#delete_macros { $macros_to_be_deleted }

Macros can be deleted using the #delete_macros keyword, for example,

```
#delete_macros { LP_Factor SW ZE }
```

will delete all macros previously defined with the names LP_Factor, SW and ZE including macros of the same name but with different arguments.

#define, #undef, #ifdef, #else, #endif

The #define and #undef directives works similar to the c pre-processor directives of the same name. #define and #undef is typically used with #ifdef, #else, #endif directives to control macro expansion in INP files. For example, the following INP segment,

```
#ifdef STANDARD_MACROS
       xdd...
#endif
```

will expand to contain the xdd keyword if STANDARD_MACROS has been previously defined using a #define directive. The following will also expand to contain the xdd keyword if STANDARD_MACROS has not been defined using a #define directive,

```
#ifdef !STANDARD_MACROS
   #define STANDARD_MACROS
   xdd...
#endif
```

or,

```
#ifndef !STANDARD_MACROS
   #define STANDARD_MACROS
   xdd...
#endif
```

Note the use of the '!' character placed before STANDARD_MACROS which means if STANDARD_MACROS is not defined.

13.1.2 Pre-processor equations and #prm, #if, #elseif, #out

Pre-processor parameters, called hash parameters, can be defined by placing a # before the text "prm". #prm's can be a function of other #prm's and they can be used in #if, #elseif, #m_if and #m_elseif pre-processor statements. #prm's are only evaluated at the pre-processor stage of loading INP files (see test_examples\hash_prm.inp); they are therefore unknown to the kernel and are totally separate to parameters defined using *prm*. Pre-processed output can be found in the TOPAS.LOG file when running TOPAS.EXE or TC.LOG when running TC.EXE.

The #out and #m_out allows pre-processor #prm's values, which can be strings or numbers, to be placed into the pre-processed text. For example:

```
#prm a = Constant(Rand(0,1));
#out a
```

will output a random number between 0 and 1 into the pre-processed file at the position of #out. INP files can therefore be manipulated with #prm's and #if statements with a means of identifying the manipulation carried out.

The following:

```
macro Ex1(a)
{
   #m_if a == "b";
      Yes b
   #m_elseif a == "c";
      Yes c
   #m_endif
}
Ex1("b")
```

expands to:

```
Yes b
```

In the following:

```
#prm ran = Constant(Rand(0,1));
#if ran < 0.5;
   view_structure
#endif
#if ran < 0.5;
   view_structure
#endif
#if ran < 0.5;
   view_structure
#endif
```

each call to "ran" in the #if statements would return the same value because of the use of Constant.

More complicated INP file manipulation is shown in the following:

```
#prm space_group_number = 4;
#if And(space_group_number >= 75, space_group_number <= 142);
…
#elseif And(space_group_number >= 16, space_group_number <= 74);
…
#endif
```

13.1.3 Directives invoked on macro expansion

#m_if, #m_ifarg, #m_elseif, #m_else, #m_endif, #m_if, #m_out are conditional directives that are invoked on macro expansion. #m_ifarg operates on two statements immediately following its use; the first must refer to a macro argument and the second can be any of the following: #m_code, #m_eqn, #m_code_refine, #m_one_word and “some string”. #m_ifarg evaluates to true according to the rules of Table 13.1.

Table 13.1: #m_ifarg syntax and meaning.

	Evaluates to true if the following is true:
#m_ifarg c #m_code	If the macro argument c has a letter or the character ! as the first character and if it is not an equation.
#m_ifarg c #m_eqn	If the macro argument c is an equation.
#m_ifarg c #m_code_refine	If the macro argument c has a letter as the first character and if it is not an equation.
#m_ifarg c “some_string”	If the macro argument c == “some_string”.
#m_ifarg v #m_one_word.	If the macro argument v consists of one word.

The directives #m_argu, #m_first_word, #m_unique_not_refine all operate on one macro argument with the intention of changing the value of the argument according to the rules of Table 13.2.

Table 13.2: Directives that operate and maybe change the value of a macro argument.

	Meaning:
#m_argu c	Change the macro argument c to a unique parameter name if it has @ as the first character.
#m_unique_not_refine c	Change the macro argument c to a unique parameter name that is not to be refined.
#m_first_word v	Replace the macro argument v with the first word in the macro argument v.

13.1.4 Defining unique parameters within macros

#m_unique $string assigns a unique parameter name to $string within a macro. This allows new unique parameters to be defined within macros without the worry of name clashes. In the example:

```
macro Some_macro(b) { prm #m_unique a = Cos(Th); }
```

"a" is assigned a unique parameter name and it has the scope of the macro body text. The Robust_Refinement and TCHZ_Peak_Type macros are good examples of its use.

13.1.5 Superfluous parentheses and the "&" type for macros and its arguments

The pre-processor is an un-typed language meaning that it knows nothing about the type of text passed to macro arguments. This has great flexibility but there can be drawbacks; for example, the following:

```
macro divide(a, b) { a / b }
prm e = divide(a+b, c-d);
```

expands to the unintended result of:

```
prm e = a + b / c - d;
```

The writer of the macro could solve this problem by rewriting the macro with parentheses:

```
macro divide(a, b) { (a) /( b) }
```

Alternatively the "&" type can be used for macros that expect equation type arguments. Defining the macro with "&" before the arguments as in:

```
macro divide(& a, & b) { a / b }
prm e = divide(a+b, c-d);
```

instructs the pre-processor that the argument is of an equation type and a check is made to determine whether the argument needs parentheses. This results in the correct expansion of:

```
prm e = (a+b) / (c-d);
```

Even with "&" types used for arguments, the following:

```
macro divide(& a, & b) { a / b }
prm e = divide(a+b, c-d)^2;
```

expands to the unintended:

```
prm e = (a + b) / (c - d)^2;
```

The writer of the macro could again rewrite the macro to include more parentheses:

```
macro divide(a, b) {  ((a) / (b))  }
```

Or, define the expansion of the macro itself to have an "&" type by placing the "&" character before the macro name itself as follows:

```
macro & divide(& a, & b) {  a / b  }
```

Expansion of prm e = divide(a+b, c-d)^2 now becomes the intended:

```
   prm e = ((a + b) / (c - d))^2;
```

With the use of the "&" type, macros such as Ramp defined in the previous TOPAS V4 as:

```
macro Ramp(x1, x2, n)
{
   ((x1) + ((x2)-(x1)) Mod(Cycle_Iter, (n)) / ((n)-1))
}
```

can now be written with less parentheses as follows:

```
macro & Ramp(& x1,& x2,& n)
{
   x1 + (x2-x1) Mod(Cycle_Iter, n) / (n-1)
}
```

13.2 Overview

The file TOPAS.INC is included into INP files by default; it contains commonly used standard macros. The meaning of the macro arguments in TOPAS.INC can be readily determined from the following conventions:

- Arguments called "c" correspond to a parameter name.
- Arguments called "v" correspond to a parameter value.
- Arguments called "cv" correspond to a parameter name and/or value.

For example, the macro Cubic(cv) requires a value and/or a parameter name as an argument. Some examples are:

```
Cubic(a_lp 10.604)
Cubic(10.604)
Cubic(@ 10.604  min 10.59 max 10.61)
```

Here are some more examples for the Slit_Width macro:

```
SW(@, .1)
SW(sw, .1  min = Val-.02; max = Val+.02;)
SW((ap+bp)/cp, 0)     ' where ap, bp anc cp are parameters defined elsewhere
```

The following listing gives an overview of the standard macros used by TOPAS:

xdd file input macros
DAT(path)
RAW(path)
RAW(path, range_num)
XDD(path)
XY(path, calc_step)
XYE(path_ext)
SCR(path)
SHELX_HKL4(path)
TOF_XYE(path, calc_step)
TOF_GSAS(path, calc_step)

Lattice parameters
Cubic(cv)
Tetragonal(a_cv, c_cv)
Hexagonal(a_cv, c_cv)
Rhombohedral(a_cv, al_cv)

Emission profile macros
CuKa1(yminymax)
CuK1sharp(yminymax)
CuKa2_analyt(yminymax)
CuKa2(yminymax)
CuKa4_Holzer(yminymax)
CuKa5(yminymax)
CuKa5_Berger(yminymax)
CoKa3(yminymax)
CoKa7_Holzer(yminymax)
CrKa7_Holzer(yminymax)
FeKa7_Holzer(yminymax)
MnKa7_Holzer(yminymax)
NiKa5_Holzer(yminymax)
MoKa2(yminymax)
CuKb4_Holzer(yminymax)
CoKb6_Holzer(yminymax)
CrKb5_Holzer(yminymax)
FeKb4_Holzer(yminymax)
MnKb5_Holzer(yminymax)
NiKb4_Holzer(yminymax)
No_Th_Dependence (yminymax)
Absorption_Edge_Correction (...)

Instrument convolutions
Radius(rp, rs)
Specimen_Tilt(c, v)
Slit_Width(c, v) or SW(c, v)
Divergence(c, v)
Variable_Divergence(c, v)
Variable_Divergence_Shape(c, v)
Variable_Divergence_Intensity
Simple_Axial_Model(c, v)
Full_Axial_Model(filament_cv, sample_cv, detector_cv, psol_cv, ssol_cv)
Finger_et_al(s2, h2)
Tube_Tails(source_width_c, source_width_v, z1_c, z1_v, z2_c, z2_v, z1_z2_h_c, z1_z2_h_v)
UVW(u, uv, v, vv, w, wv)

Phase peak_type's
PV_Peak_Type(ha, hav, hb, hbv, hc, hcv, lora, lorav, lorb, lorbv, lorc, lorcv)
TCHZ_Peak_Type(u, uv, v, vv, w, wv, x, xv, y, yv, z, zv)
PVII_Peak_Type(ha, hav, hb, hbv, hc, hcv, ma, mav, mb, mbv, mc, mcv)

Quantitative Analysis
Apply_Brindley_Spherical_R_PD(R, PD)
Known_Weight_Percent(& w)
MVW(m_v, v_v, w_v)
Remove_Phase

2θ Corrections
Cylindrical_2Th_Correction(μR)
Zero_Error or ZE(c, v)
Specimen_Displacement(c, v) or SD(c, v)

Intensity Corrections
Cylindrical_I_Correction(μR)
Lorentz_Factor, LP_Factor(c, v)
LP_Factor_Synchrotron, LP_Factor_Synchrotron_Simple
Preferred_Orientation(c, v, ang, hkl) or PO(c, v, ang, hkl)
PO_Two_Directions(c1, v1, ang1, hkl1, c2, v2, ang2, hkl2, w1c, w1v)
PO_Spherical_Harmonics(sh, order)
Surface_Roughness_Pitschke_et_al(a1c, a1v, a2c, a2v)
Surface_Roughness_Suortti(a1c, a1v, a2c, a2v)

Bondlength penalty functions
Anti_Bump(ton, s1, s2, ro, wby)
AI_Anti_Bump(s1, s2, ro, wby, num_cycle_iters), AI_Anti_Bump(s1, s2, ro, wby)
Parabola_N(n1, n2, s1, s2, ro, wby)
Grs_Interaction(s1, s2, wqi, wqj, c, ro, n)
Grs_No_Repulsion(s1, s2, wqi, wqj, c)
Grs_BornMayer(s1, s2, wqi, wqj, c, ro, b)
Distance_Restrain(sites, t, t_calc, tol, wscale)
Angle_Restrain(sites, t, t_calc, tol, wscale)
Flatten(sites, t_calc, tol, wscale)
Distance_Restrain_Keep_Within(sites, r, wby, num_cycle_iters)
Distance_Restrain_Keep_Out(sites, r, wby, num_cycle_iters)
Keep_Atom_Within_Box(size)

Reporting macros
Create_2Th_Ip_file(file)
Create_d_Ip_file(file)
Create_hklm_d_Th2_Ip_file(file)
Out_Yobs_Ycalc_and_Difference(file)
Out_X_Yobs(file)
Out_X_Ycalc(file)
Out_X_Difference(file)
Out_F2_Details(file)
Out_A01_A11_B01_B11(file)
Out_FCF(file)
Out_CIF_STR(file)
Out_Single_Crystal_Details(file)

Rigid body macros
Set_Length(s0, s1, r, xc, yc, zc, cva, cvb)
Set_Lengths(s0, s1, s2, r, xc, yc, zc, cva1, cvb1, cva2, cvb2)
Set_Lengths(s0, s1, s2, s3, r, xcv, ycv, zcv, cva1, cvb1, cva2, cvb2, cva3, cvb3)
Triangle(s1, s2, s3, r)
Triangle(s0, s1, s2, s3, r)

Triangle(s0, s1, s2, s3, r, xc, yc, zc, cva, cvb, cvc)
Tetrahedra(s0, s1, s2, s3, s4, r, xc, yc, zc, cva, cvb, cvc)
Octahedra(s0, s1, s2, s3, s4, s5, s6, r)
Octahedra(s0, s1, s2, s3, s4, s5, s6, r, xc, yc, zc, cva, cvb, cvc)
Hexagon_sitting_on_point_in_xy_plane(s1, s2, s3, s4, s5, s6, a)
Hexagon_sitting_on_side_in_xy_plane(s1, s2, s3, s4, s5, s6, a)
Point_for_site(site, cart_x, cart_y, cart_z)
Translate(acv, bcv, ccv)
Translate(acv, bcv, ccv, ops)
Translate_with_site_start_values(s0, xc, yc, zc)
Rotate_about_points(cv, a, b)
Rotate_about_points(cv, a, b, pts)
Rotate_about_these_points(cv, a, b, ops)
Rotate_about_axies(cva, cvb)
Rotate_about_axies(cva, cvb, cvc)
Rotation_vector_from_points(a, b)

Background functions using fit_objects

One_on_X(c, v)
Bkg_Diffuse(b, bv, bb, bbv)
PV(a, xo, fwhm, g)
PV(a, xo, fwhm, g, av, xov, fwhmv, gv)
PV_Left_Right(a, xo, fwhm1, fwhm2, g)
PV_Left_Right(a, xo, fwhm1, fwhm2, g, av, xov, fwhm1v, fwhm2v, gv)

Sample convolutions

Sample_Thickness(dc, dv)
Absorption or AB
Absorption_With_Sample_Thickness_mm_Shape(u, uv, d, dv)
Absorption_With_Sample_Thickness_mm_Shape_Intensity(u, uv, d, dv)
CS_L(c,v) or Crystallite_Size(c, v) or CS(c, v)
CS_G(c, v)
Strain_L(c, v) or Microstrain(c, v) or MS(c, v)
Strain_G(c, v)
LVol_FWHM_CS_G_L(k, lvol, kf, lvolf, csgc, csgv, cslc, cslv)
e0_from_Strain(e0, sgc, sgv, slc, slv)
WPPM_Cube_Ln_Normal(muc, muv, sigc, sigv)
WPPM_Sphere_Ln_Normal(muc, muv, sigc, sigv)
WPPM_Octahedron_Ln_Normal(muc, muv, sigc, sigv)
Stephens_triclinic(...)
Stephens_monoclinic(...)
Stephens_orthorhombic(...)
Stephens_tetragonal_low(...)
Stephens_tetragonal_high(...)
Stephens_trigonal_low(...)
Stephens_trigonal_high(...)
Stephens_trigonal_high_2(...)
Stephens_hexagonal(...)
Stephens_cubic(...)

Neutron TOF

TOF_LAM(w_ymin_on_ymax)
TOF_x_axis_calibration(t0, t0v, t1, t1v, t2, t2v)
TOF_Exponential(a0, a0v, a1, a1v, wexp, t1, lr)
TOF_CS_L(c, v, t1)
TOF_CS_G(c, v, t1)
TOF_PV(fwhm, fwhmv, lor, lorv, t1)

Miscalleneous
Auto_T
Temperature_Regime
STR(sg)
Exclude
Decompose(diff_toll)
ADPs_Keep_PD
Mixture_LAC_1_on_cm(mlac)
Phase_Density_g_on_cm3(pd)
Phase_LAC_1_on_cm(u)
Gauss(xo, fwhm), Lorentzian(xo, fwhm)
Robust_Refinement
CV(c, v)
CeV(c, v)
CeV_or_0(c, v)

Indexing
Predefined space group macros
Miscellaneous macros for indexing

Charge Flipping
Ramp, Ramp_Clamp, Cycle_Ramp, Tangent, Restart_CF, Pick, Pick_Best
Out_for_cf(file)

13.3 Detailed description of macros

13.3.1 xdd file input macros

RAW(path) **RAW(path, range_num)**
DAT(path) **XDD(path)**
XY(path, calc_step) **XYE(path_ext)**
SCR(path) **SHELX_HKL4(path)**
TOF_XYE(path, calc_step) **TOF_GSAS(path, calc_step)**

Import measured data in different file formats, see also section 14.

[path]: Filename. Can include drive and path but no file extension.

[range_num]: The range number to be loaded (RAW files only).

[calc_step]: Step size used in calculations.

13.3.2 Lattice parameters

Cubic(a_cv) **Tetragonal(a_cv, c_cv)**
Hexagonal(a_cv, c_cv) **Rhombohedral(a_cv, al_cv)**

Simplifies the definition of lattice parameters.

[a_cv]: Lattice parameter a.

[c_cv]: Lattice parameter c.

[al_cv]: Lattice parameter alpha.

13.3.3 Emission profile macros

**CuKa1(yminymax), CuK1sharp(yminymax),
CuKa2_analyt(yminymax), CuKa2(yminymax),
CuKa4_Holzer(yminymax),
CuKa5(yminymax), CuKa5_Berger(yminymax),
CuKb4_Holzer(yminymax)**

**CoKa3(yminymax), CoKa7_Holzer(yminymax),
CoKb6_Holzer(yminymax)**

**CrKa7_Holzer(yminymax),
CrKb5_Holzer(yminymax)**

**FeKa7_Holzer(yminymax),
FeKb4_Holzer(yminymax)**

**MnKa7_Holzer(yminymax),
MnKb5_Holzer(yminymax)**

**NiKa5_Holzer(yminymax),
NiKb4_Holzer(yminymax)**

MoKa2(yminymax)

Defines a source emission profile.

[yminymax]: Determines the x-axis extent to which an emission profile line is calculated.

**Absorption_Edge_Correction (& max_lam, cedge, vedge,
ca_white, va_white, cb_white, vb_white,
ca_erf, va_erf, cedge_extra, vedge_extra)**

Allows description of absorption edges. For details see section 5.3.2.1.2.

[& max_lam]: Maximum wavelength to apply the correction to.

[cedge, vedge]: Parameter name, position of the absorption edge (wavelength)

[ca_white, va_white]: Parameter name, remnant Bremsstrahlung curvature

[cb_white, vb_white]: Parameter name, remnant Bremsstrahlung curvature

[ca_erf, va_erf]: Parameter name, width of the absorption edge

[cedge_extra, vedge_extra]: Parameter name, height of the absorption edge

No_Th_Dependence

Defines an emission profile that is 2θ independent (no broadening related to spectral dispersion) and allows to fit to any XY data.

13.3.4 Instrument convolutions

Radius(rp, rs)

Instrument radius.

[rp, rs]: Primary and secondary instrument radii in [mm]. For most diffractometers rp equals rs.

Specimen_Tilt(c, v)

Specimen tilt.

[c, v]: Parameter name, Specimen tilt in [mm].

Slit_Width(c, v), SW(c, v)

Aperture of the detector (= receiving) slit.

[c, v]: Parameter name, detector slit aperture in [mm].

Divergence(c, v)

Equatorial divergence of the beam for fixed slits.

[c, v]: Parameter name, beam divergence in [°].

Variable_Divergence(c, v)

Constant illuminated sample length for variable slits (i.e. variable beam divergence). This macro considers the peak shape and corrects intensity deviations inherent to variable slits.

[c, v]: Parameter name, illuminated sample length in [mm].

Variable_Divergence_Shape(c, v)

Constant illuminated sample length for variable slits (i.e. variable beam divergence). This macro considers the peak shape inherent to variable slits.

[c, v]: Parameter name, illuminated sample length in [mm].

Variable_Divergence_Intensity

Constant illuminated sample length for variable slits (i.e. variable beam divergence). This macro corrects intensity deviations inherent to variable slits.

Simple_Axial_Model(c, v)

Simple model for describing peak asymmetry due to axial divergence of the beam.

[c, v]: Parameter name, receiving slit length [mm].

Full_Axial_Model(filament_cv, sample_cv, detector_cv, psol_cv, ssol_cv)

Accurate model for describing peak asymmetry due to axial divergence of the beam.

[filament_cv]: Tube filament length in [mm].

[sample_cv]: Sample length in axial direction in [mm].

[detector_cv]: Length of the detector (= receiving) slit in [mm].

[psol_cv, ssol_cv]: Aperture of the primary and secondary Soller slit in [°].

Finger_et_al(s2, h2)

Simple model for describing peak asymmetry due to axial divergence of the beam according to Finger et al., 1994.

[s2, h2]: Sample length, receiving slit length.

Tube_Tails(source_width_c, source_width_v, z1_c, z1_v, z2_c, z2_v, z1_z2_h_c, z1_z2_h_v)

Model for description of tube tails (Bergmann, 2000).

[source_width_c, source_width_v]: Parameter name, tube filament width in [mm].

[z1_c, z1_v]: Parameter name, effective width of tube tails in the equatorial plane perpendicular to the X-ray beam - negative z-direction [mm].

[z2_c, z2_v]: Parameter name, effective width of tube tails in the equatorial plane perpendicular to the X-ray beam - positive z-direction [mm].

[z1_z2_h_c, z1_z2_h_v]: Parameter name, fractional height of the tube tails relative to the main beam.

UVW(u, uv, v, vv, w, wv)

Cagliotti relation (Cagliotti et al., 1958).

[u, v, w]: Parameter names.

[uv, vv, wv]: Halfwidth parameters.

13.3.5 Phase peak_type's

PV_Peak_Type(ha, hav, hb, hbv, hc, hcv, lora, lorav, lorb, lorbv, lorc, lorcv),
TCHZ_Peak_Type(u, uv, v, vv, w, wv, x, xv, y, yv, z, zv)
PVII_Peak_Type(ha, hav, hb, hbv, hc, hcv, ma, mav, mb, mbv, mc, mcv)

Pseudo-Voigt, TCHZ pseudo-Voigt and PearsonVII functions.

For the definition of the functions and function parameters refer to section 5.3.2.1.2.

13.3.6 Quantitative analysis

Apply_Brindley_Spherical_R_PD(R, PD)

Applies the Brindley correction for quantitative analysis (Brindley, 1945).

[R, PD]: Radius of the particle in [cm], packing density.

Known_Weight_Percent(& w)

Constrains weight percents to a known value (& w).

MVW(m_v, v_v, w_v)

Returns cell mass, cell volume, and relative phase amount.

[m_v, v_v, w_v]: Mass, volume, and weight parameters.

Remove_Phase(& wt, & fr)

Allows for phase removal during refinement. For more details see section 10.3.3.

[& wt, & fr]: weight percent and error in the weight percent.

13.3.7 2θ corrections

Cylindrical_2Th_Correction(μR)

Applies a 2θ correction for use with capillary samples (Sabine et al., 1998).

[μR]: μ is the linear absorption coefficient and R is the diameter of the capillary.

Zero_Error(c, v), ZE(c, v)

Zero point error.

[c, v]: Parameter name, zero point error in [° 2θ].

Specimen_Displacement(c, v), SD(c, v)

Specimen displacement error.

[c, v]: Parameter name, sample displacement in [mm].

13.3.8 Intensity corrections

Cylindrical_I_Correction(μR)

Applies an intensity correction for use with capillary samples (Sabine et al., 1998).

[μR]: μ is the inear absorption coefficient and R is the diameter of the capillary.

Lorentz_Factor, LP_Factor(c, v)

Lorentz and Lorentz-Polarisation factor.

[c, v]: Parameter name, monochromator angle in [°2θ].

For unpolarized radiation v is 90 (e.g. X-ray diffractometers without any monochromator), for fully polarized radiation v is 0 (e.g. synchrotron radiation).

Values for most common monochromators (Cu radiation) are:

Ge : 27.3

Graphite : 26.4

Quartz : 26.6

There is no polarization factor for neutron data and thus the angle for Lorentz Polarization should be set to 90; this gives the Lorentz only part. Alternatively the Lorentz_Factor macro can be used for fixed wavelength neutron data.

LP_Factor_Synchrotron(pp, ppv, mono, monov), LP_Factor_Synchrotron_Simple

Synchrotron-specific Lorentz-Polarisation factors, for details see section 11.4. Note: LP_Factor_Synchrotron_Simple is equivalent to Lorentz_Factor.

[pp, ppv]: Parameter name (polarisation in the plane of the synchrotron), value.

[mono, monov]: Parameter name, monochromator angle in [°2θ].

Preferred_Orientation(c, v, ang, hkl), PO(c, v, ang, hkl)

Preferred orientation correction based on March (1932).

[c, v]: Parameter name, March parameter value.

[ang, hkl]: Parameter name, lattice plane.

PO_Two_Directions(c1, v1, ang1, hkl1, c2, v2, ang2, hkl2, w1c, w1v)

Preferred orientation correction based on March (1932) considering two preferred orientation directions.

[c1, v1]: Parameter name and March parameter value for the first preferred orientation direction.

[ang1, hkl1]: Parameter name and lattice plane for the first preferred orientation direction.

[c2, v2]: Parameter name and March parameter value for the second preferred orientation direction.

[ang2, hkl2]: Parameter name and lattice plane for the second preferred orientation direction.

[w1c, w1v]: Parameter name, fraction of crystals oriented into first preferred orientation direction.

PO_Spherical_Harmonics(sh, order)

Preferred orientation correction based on spherical harmonics according to Järvinen (1993).

[sh, order]: Parameter name, spherical harmonics order.

Surface_Roughness_Pitschke_et_al(a1c, a1v, a2c, a2v),
Surface_Roughness_Suortti(a1c, a1v, a2c, a2v)

Suortti (1972) and Pitschke et al. (1993) intensity corrections each with two parameters a1 and a2.

[a1c, a2c] Parameter names

[a1v, a2v] Surface roughness parameters

13.3.9 Bondlength penalty functions

Anti_Bump(ton, s1, s2, ro, wby)
Al_Anti_Bump(s1, s2, ro, wby, num_cycle_iters)
Al_Anti_Bump(s1, s2, ro, wby)

Applies a penalty function as a function of the distance between atoms. The closer the atoms are the higher the penalty is.

[ton]: Sets to_N of the *box_interaction* keyword.

[s1, s2]: Sites.

[ro]: Distance.

[wby]: Relative weighting given to the penalty function.

[num_cycle_iters]: Penalty is applied while Cycle_Iter (current iteration within the current cycle) is smaller than *num_cycle_iters*

For more details refer to *box_interaction* and *ai_anti_bump*.

Parabola_N(n1, n2, s1, s2, ro, wby)

Applies a penalty function as a function of the distance between atoms. The closer the atoms are the higher the penalty is.

[n1]: The closest n1 number of atoms of type s2 is soft constrained to a distance ro away from s1 .

[n2]: The closest n2 number of atoms of type s2 (excluding the closest n1 number of atoms of type s2) is repelled from s1, if the distance between s1 and s2 is less than ro .

[s1, s2]: Sites.

[ro]: Distance.

[wby]: Relative weighting given to the penalty function.

Grs_Interaction(s1, s2, wqi, wqj, c, ro, n)

Penalty function applying the GRS series according to Coelho & Cheary (1997).

[s1, s2]: Sites.

[wqi, wqj]: Valence charge of the atoms.

[c]: Name of the GRS.

[ro]: Distance.

[n]: The exponent of the repulsion part of the Lenard-Jones potential.

For more details refer to *grs_interaction*.

Grs_No_Repulsion(s1, s2, wqi, wqj, c)

Used for calculating the Madelung constants.

[s1, s2]: Sites.

[wqi, wqj]: Valence charge of the atoms.

[c]: Name of the GRS.

Grs_BornMayer(s1, s2, wqi, wqj, c, ro, b)

Uses the GRS series with a Born-Mayer equation for the repulsion term.

[s1, s2]: Sites.

[wqi, wqj]: Valence charge of the atoms.

[c]: Name of the GRS.

[ro]: Mean distance.

[b]: b-constant for the repulsion part of the Born-Mayer potential.

Distance_Restrain(sites, t, t_calc, tol, wscale),
Angle_Restrain(sites, t, t_calc, tol, wscale),
Flatten(sites, t_calc, tol, wscale),
Distance_Restrain_Keep_Within(sites, r, wby, num_cycle_iters)
Distance_Restrain_Keep_Out(sites, r, wby, num_cycle_iters)

Applies penalties restraining distances and angles between sites. 'sites' must comprise two sites for the distance restraints and three for the angle restraints. For Flatten 'sites' must contain more than three sites.

[sites]: Site names

[t]: Desired value for distances and angles respectively

[tcalc]: Calculated value for distances and angles respectively

[tol]: Angle or distance value constrained within the range t-tol < t < t+tol

[wscale], [wby]: Scales the penalty; useful when more than one penalty is used and one needs to be applied more forcefully than another

[r]: Distance in Angstroms

[num_cycle_iters]: Penalty is applied while Cycle_Iter (current iteration within the current cycle, see also section Error: Reference source not found) is smaller than *num_cycle_iters*

Keep_Atom_Within_Box(size)

Applies constraints such that the present site cannot more outside of a box with a length of 2*size.

13.3.10 Reporting macros

Create_2Th_lp_file(file)

Creates a file with positions (2θ) and intensities.

[file]: Filename. Can include drive and path.

Create_d_lp_file(file)

Creates a file with positions (d) and intensities.

[file]: Filename. Can include drive and path.

Create_hklm_d_Th2_lp_file(file)

Creates a file with the following information for each peak: h, k, l, multiplicity, positions d and 2θ and intensities.

[file]: Filename. Can include drive and path.

Out_Yobs_Ycalc_and_Difference(file)

Outputs the x-axis, Yobs, Ycalc and difference.

[file]: Filename. Can include drive and path.

Out_X_Yobs(file), Out_X_Ycalc(file), Out_X_Difference(file)

Outputs the x-axis, Yobs, Ycalc and difference to files.

[file]: Filename. Can include drive and path.

Out_F2_Details(file), Out_A01_A11_B01_B11(file)

Outputs structure factor details, see section 9.1.2.

[file]: Filename. Can include drive and path.

Out_FCF(file)

Outputs a CIF file representation of structure factor details suitable for generating Fourier maps using ShelX.

[file]: Filename. Can include drive and path.

Out_CIF_STR(file)

Outputs structure details in CIF format.

[file]: Filename. Can include drive and path.

Out_Single_Crystal_Details(file)

Outputs details for single crystal refinements

[file]: Filename. Can include drive and path.

13.3.11 Rigid body macros

Set_Length(s0, s1, r, xc, yc, zc, cva, cvb)

Fixes the distance between two sites.

[s0, s1]: Site names.

[r]: Distance in Angstroms.

[xc, yc, zc]: The parameter names for the coordinates of s0.

[cva, cvb]: Parameter names and values for rotations about the x and y axes

Set_Lengths(s0, s1, s2, r, xc, yc, zc, cva1, cvb1, cva2, cvb2)
Set_Lengths(s0, s1, s2, s3, r, xcv, ycv, zcv, cva1, cvb1, cva2, cvb2, cva3, cvb3)

Fixes the distance between two and three sites, respectively.

Set_Lengths(s0, s1, s2, r, xc, yc, zc, cva1, cvb1, cva2, cvb2) is defined as

```
macro Set_Lengths(s0, s1, s2, r, xc, yc, zc, cva1, cvb1, cva2, cvb2)
{
   Set_Length(s0, s1, r, xc, yc, zc, cva1, cvb1)
   Set_Length(s0, s2, r, xc, yc, zc, cva2, cvb2)
}
```

and so on.

Triangle(s1, s2, s3, r)
Triangle(s0, s1, s2, s3, r)
Triangle(s0, s1, s2, s3, r, xc, yc, zc, cva, cvb, cvc)

Defines a regular triangle without and with a central atom (s0).

[s0, s1, s2, s3]: Site names. s0 is the central atom of the triangle.

[r]: Distance in Angstroms.

[xc, yc, zc]: Parameter names for the fractional coordinates of the geometric center of the triangle.

[cva, cvb, cvc]: Parameter names and values for rotations about the x, y and z axes.

Tetrahedra(s0, s1, s2, s3, s4, r, xc, yc, zc, cva, cvb, cvc)

Defines a tetrahedra with a central atom.

[s0, s1, s2, s3, s4]: Site names. s0 is the central atom of the tetrahedra.

[r]: Distance in Angstroms.

[xc, yc, zc]: Parameter names for the fractional coordinates of the geometric center of the tetrahedra.

[cva, cvb, cvc]: Parameter names and values for rotations about the x, y and z axes.

Octahedra(s0, s1, s2, s3, s4, s5, s6, r)
Octahedra(s0, s1, s2, s3, s4, s5, s6, r, xc, yc, zc, cva, cvb, cvc)

Defines an octahedra with a central atom.

[s0, s1, s2, s3, s4, s5, s6]: Site names. s0 is the central atom of the octahedra.

[r]: Distance in Angstroms.

[xc, yc, zc]: Parameter names for the fractional coordinates of the geometric center of the octahedra.

[cva, cvb, cvc]: Parameter names and values for rotations about the x, y and z axes.

Hexagon_sitting_on_point_in_xy_plane(s1, s2, s3, s4, s5, s6, a)
Hexagon_sitting_on_side_in_xy_plane(s1, s2, s3, s4, s5, s6, a)

Defines a regular hexagon, where the hexagon is sitting on a point or on a side in the x-y plane, respectively.

[s1, s2, s3, s4, s5, s6]: Site names.

[a]: Distance in Angstroms.

Point_for_site(site, cart_x, cart_y, cart_z)

Transforms the Cartesian coordinates of a site into fractional coordinates.

[site]: Site name.

[cart_x, cart_y, cart_z)]: Cartesian coordinates of the site.

Translate(acv, bcv, ccv) ,
Translate(acv, bcv, ccv, ops)

Performs a translation of the rigid body.

[acv, bcv, ccv]: Amount of the translation in fractional coordinates.

[ops]: Operates on previously defined sites in “ops”.

Translate_with_site_start_values(s0, xc, yc, zc)

Performs a translation using the coordinates of s0 as start values.

[s0]: Site name.

[xc, yc, zc]: Parameter names for the coordinates of s0.

Rotate_about_points(cv, a, b)
Rotate_about_points(cv, a, b, pts)

Performs a rotation about a rotation vector specified by two sites.

[cv]: Amount the rigid body is rotated about the specified rotation vector in degrees.

[a, b]: Rotation vector defined by the sites a and b.

[pts]: Operates on previously defined point_for_site(s).

Note: Do not include points rotated about in the “operate on points” list of the Rotate_about_points macro. For example, in

```
Rotate_about_points(@ 1 0, C1, C2, " C3 C4 C5 C6 ")
```

the points C1 and C2 are not included in the “points operated on” list. Note also that Rotate_about_points without a “points operated on” list will operate on all previously defined point_for_site(s). Therefore, when an “operate on points” list is not defined then it is necessary to place the “points rotated about” after the Rotate_about_points macro. It is best to specify an “operate on points” list when in doubt.

Rotate_about_these_points(cv, a, b, ops)

Performs a rotation about a rotation vector specified by two sites.

[cv]: Amount the rigid body is rotated about the specified rotation vector in degrees.

[a, b]: Rotation vector defined by the sites a and b.

[ops]: Operates on previously defined *point_for_site*(s).

Rotate_about_axies(cva, cvb)
Rotate_about_axies(cva, cvb, cvc)

Performs a rotation about the axis of the rigid body.

[cva, cvb, cvc]: Parameter names and values for rotations about the x, y and z axes.

Rotation_vector_from_points(a, b)

Determines a rotation vector from two points.

[a, b]: Rotation vector defined by the sites a and b.

13.3.12 Background functions using fit_objects

One_on_X(c, v)

1/X background function ideal to describe background intensity at low angles due to air scattering

[c, v]: Parameter name and value.

Bkg_Diffuse(b, bv, bb, bbv)

Defines a function to describe short range order effects.

[b, bv]: Parameter name, refineable weight.

[bb, bbv]: Parameter name, correlation shell radii.

PV(a, xo, fwhm, g)
PV(a, xo, fwhm, g, av, xov, fwhmv, gv)

Defines a pseudo-Voigt function.

[a, av]: Parameter name, area.

[xo, xov]: Parameter name, Position in [° 2θ].

[fwhm, fwhmv]: Parameter name, full width at half maximum in [° 2θ].

[g, gv]: Parameter name, pseudo-Voigt mixing parameter.

PV_Left_Right(a, xo, fwhm1, fwhm2, g)
PV_Left_Right(a, xo, fwhm1, fwhm2, g, av, xov, fwhm1v, fwhm2v, gv)

Defines a split-pseudo-Voigt function.

[a, av]: Parameter name, area.

[xo, xov]: Parameter name, Position in [° 2θ].

[fwhm1, fwhm1v]: Parameter name, full width at half maximum in [° 2θ] for the left composite function.

[fwhm2, fwhm2v]: Parameter name, full width at half maximum in [° 2θ] for the right composite function.

[g, gv]: Parameter name, pseudo-Voigt mixing parameter.

13.3.13 Sample convolutions

Sample_Thickness(dc, dv)

Describes the sample thickness in the direction of the scattering vector.

[dc, dv]: Parameter name, sample thickness in [mm].

Absorption(c, v), AB(c, v)

Linear absorption coefficient used for adjusting the peak shape.

[c, v]: Parameter name, linear absorption coefficient in cm^{-1}.

Absorption_With_Sample_Thickness_mm_Shape(u, uv, d, dv)

Corrects the peak shape for absorption effects.

[u, uv]: Parameter name, linear absorption coefficient in cm^{-1}.

[d, dv]: Parameter name, sample thickness in [mm].

Absorption_With_Sample_Thickness_mm_Shape_Intensity(u, uv, d, dv)

Corrects the peak intensity for absorption effects.

[u, uv]: Parameter name, absorption coefficient in cm-1.

[d, dv]: Parameter name, sample thickness in [mm].

CS_L(c, v), Crystallite_Size(c, v), CS(c, v)

Applies a Lorentzian convolution with a FWHM that varies according to the relation:

`lor_fwhm = c / Cos(Th).`

For details refer to section 6.

[c, v]: Parameter name, Lorentzian component related to size broadening.

CS_G(c, v)

Applies a Gaussian convolution with a FWHM that varies according to the relation:

`gauss_fwhm = c / Cos(Th).`

For details refer to section 6.

[c, v]: Parameter name, Gaussian component related to size broadening.

Strain_L(c, v), Microstrain_L(c, v) MS(c, v)

Applies a Lorentzian convolution with a FWHM that varies according to the relation:

`lor_fwhm = c Tan(Th).`

For details refer to section 6.

[c, v]: Parameter name, Lorentzian component related to microstrain broadening.

Strain_G(c, v)

Applies a Gaussian convolution with a FWHM that varies according to the relation:

`gauss_fwhm = c Tan(Th).`

For details refer to section 6.

[c, v]: Parameter name, Gaussian component related to microstrain broadening.

LVol_FWHM_CS_G_L(k, lvol, kf, lvolf, csgc, csgv, cslc, cslv)

Calculates FWHM and IB (integral breadth) based volume-weighted column heights (LVol). For details refer to section 6.

[k, lvol]: shape factor (fixed to 1), integral breadth based LVol.

[kf, lvolf]: shape factor (defaults to 0.89), FWHM based LVol.

[csgc, csgv]: Parameter name, Gaussian component related to size broadening.

[cslc, cslv]: Parameter name, Lorentzian component related to size broadening.

e0_from_Strain(e0, sgc, sgv, slc, slv)

Calculates the e0 microstrain parameter. For details refer to section 6.

WPPM_Sphere_Ln_Normal(muc, muv, sigc, sigv)
WPPM_Cube_Ln_Normal(muc, muv, sigc, sigv)
WPPM_Octahedron_Ln_Normal(muc, muv, sigc, sigv)

WPPM based crystallite size analysis of spheres, cubes, and octahedra using a log normal distribution. For details see section 6.4.3.

[muc, muv]: Parameter name, mean of the log normal distribution

[sigc, sigv]: Parameter name, standard deviation of the log normal distribution

Stephens_triclinic(...)
Stephens_monoclinic(...)
Stephens_orthorhombic(...)
Stephens_tetragonal_low(...)
Stephens_tetragonal_high(...)
Stephens_trigonal_low(...)
Stephens_trigonal_high(...)
Stephens_trigonal_high_2(...)
Stephens_hexagonal(...)
Stephens_cubic(...)

See section 11.5.2.

13.3.14 Neutron TOF

TOF_LAM(w_ymin_on_ymax)

Defines a simple emission profile suitable for TOF data.

[w_ymin_on_ymax]: Determines the x-axis extent to which an emission profile line is calculated.

TOF_x_axis_calibration(t0, t0v, t1, t1v, t2, t2v)

Writes the *pk_xo* equation in terms of the three calibration constants t0, t1, t2 converting d-spacing to x-axis space.

[t0, t1, t2]: Calibration constants

[t0v, t1v, t2v]: Calibration constants values

TOF_Exponential(a0, a0v, a1, a1v, wexp, t1, lr)

An exponential convolution defined as.

```
exp_conv_const = lr Constant(t1) /
                 (CeV(a0,a0v) + CeV(a1,a1v) / D_spacing^(wexp));
```

[t1]: TOF calibration constant, see the TOF_x_axis_calibration macro.

[lr]: Defines the sign of the function in terms of "+" or "-"

TOF_CS_L(c, v, t1), TOF_CS_G(c, v, t1)

Lorentzian and Gaussian components for crystallite size.

[c, v]: Parameter name, crystallite size in [nm].

[t1]: TOF calibration constant, see the TOF_x_axis_calibration macro.

TOF_PV(fwhm, fwhmv, lor, lorv, t1)

Defines a pseudo-Voigt function suited for TOF data.

[fwhm, fwhmv]: Parameter name, full width at half maximum.

[lor, lorv]: Parameter name, pseudo-Voigt mixing parameter.

[t1]: TOF calibration constant, see the TOF_x_axis_calibration macro.

13.3.15 Miscellaneous

Auto_T(t)

Auto_T includes *quick_refine*, *randomize_on_errors* and a temperature regime and is adequate for a wide range of simulated annealing examples.

Note, that with Auto_T there is no need to determine a temperature regime or use *val_on_continue* or *rand_xyz* keywords, see also section 4.14.

[t]: Temperature. See the *temperature* keyword.

Temperature_Regime

Defines a temperature regime and is defined as *{ load temperature }*. See the *temperature* keyword as well as section 4.14.

STR(sg)

Signals the start of structure information.

[sg]: Space group symbol.

Exclude

Defines excluded regions. See the *exclude* keyword.

ADPs_Keep_PD

Keeps the anisotropic temperature parameters matrix unn positive definite, see the *adps* keyword.

Decompose(diff_toll)

Decomposes a diffraction pattern into data points at peak positions only.

[diff_toll]: Data points closer than diff_toll to another data point is not included. Decompose also sets *x_calculation_step* to the value of diff_toll.

Mixture_LAC_1_on_cm (mlac),
Phase_LAC_1_on_cm (u),
Phase_Density_g_on_cm3 (pd)

Mixture_LAC_1_on_cm, Phase_LAC_1_on_cm and Phase_Density_g_on_cm3 can calculate the mixture and phase linear absorption coefficients (for a packing density of 1) and phase density, see the *mixture_MAC keyword*.

[mlac]: linear absorption coefficient [1/cm] of the mixture, packing density of 1

[u]: Linear absorption coefficient [1/cm], packing density of 1

[pd]: Phase density [g/cm3]

Gauss(xo, fwhm), Lorentzian(xo, fwhm)

Unit area Gaussian or Lorentzian functions.

[xo]: Position

[fwhm]: FWHM

Robust_Refinement

Rescales peaks according to robust refinement algorithm by Stone et. al. (2009).

CV(c, v)

Inserts c if c is not blank; otherwise inserts v.

CeV(c, v)

Same as CV(c, v) but surrounds the result with brackets

CeV_or_0(c, v)

Same as CeV(c, v) but if v is blank then inserts a 0.

13.3.16 Indexing

13.3.16.1 Space group macros

In Table 13.3 an overview of the space group macros is provided.

Table 13.3: Predefined space group macros.

Macro	*try_space_groups*
All unique space groups individually	
Unique_Cubic_sgs	"228 219 203 210 196 230 220 206 214 197 222 218 201 205 212 198 195"
Unique_Trigonal_Hexagonal_sgs	"161 146 184 159 158 169 144 173 143"
Unique_Tetragonal_sgs	"142 110 141 109 108 88 80 79 130 126 133 103 104 106 137 138 134 125 114 105 102 101 100 86 85 92 94 76 77 90 75"
Unique_Orthorhombic_sgs	"70 43 22 68 73 37 45 41 46 36 39 20 23 21 52 56 60 61 48 54 50 33 34 32 30 29 27 31 26 19 18 17 16"
Unique_Monoclinic_sgs	"9 5 14 7 4 3"
Unique_Triclinic_sgs	"2"
Bravais lattices	
Bravais_Cubic_sgs	"196 197 195"
Bravais_Trigonal_Hexagonal_sgs	"146 143"
Bravais_Tetragonal_sgs	"79 75"
Bravais_Orthorhombic_sgs	"22 23 21 16"
Bravais_Monoclinic_sgs	"5 3"
Bravais_Triclinic_sgs	"2"
All Bravais lattices individually	
Cubic_F	"196"
Cubic_I	"197"
Cubic_P	"195"
Trigonal_Hexagonal_R	"146"
Trigonal_Hexagonal_P	"143"
Tetragonal_I	"79"
Tetragonal_P	"75"
Orthorhombic_F	"22"
Orthorhombic_I	"23"
Orthorhombic_C	"21"
Orthorhombic_P	"16"
Monoclinic_C	"5"
Monoclinic_P	"3"
Triclinic_P	"2"

13.3.16.2 Miscellaneous macros

Index_x0_from_d(d, worder), Index_x0_from_th2(th2, worder)

Allow to define x0 in the reciprocal lattice equation, see the *index_x*0 keyword

[d], [th2]: d and 2θ, respectively

[worder]: reflection order

Indexing_Solutions, Indexing_Solutions_With_Zero_Error

Allows obtaining a detailed summary of solutions by including the solution lines of the NDX file into the INP file.

13.3.17 Charge Flipping

Ramp, Ramp_Clamp, Cycle_Ramp, Tangent, Restart_CF, Pick, Pick_Best

See TOPAS.INC for details

Out_for_cf(file)

Outputs the A matrix from a Pawley refinement for use in charge flipping; uses *cf_in_A_matrix*

14 FILE TYPES AND FORMATS

Table 14.1: File types used in TOPAS.

File Type	Comments
TOPAS files	
*.PRO	Project files
*.INP	Input file
*.OUT	Output file created on termination of refinement. Same format as *.INP.
*.STR	Structure data. Same format as *.INP.
*.RGD	Rigid body data. Same format as *.INP.
*.CLD	Cloud file, see the keyword cloud
*.A	A-matrix file, contains the least squares A matrix
*.FC	Structure factor file created by Charge Flipping
*.PAR	Instrument parameters. Same format as *.INP.
*.LAM	Source emission profile data. Same format as *.INP.
*.DEF	Program defaults. Same format as *.INP.
*.NDX, *.DET	Indexing result files, see sections 7.1.2 and 7.1.3. Same format as *.INP.
*.LOG	Log file. Useful for tracking input errors.
Measurement Data	
*.RAW	Bruker AXS binaries (DIFFRAC AT and DIFFRAC*plus* V1-4)
*.BRML	Bruker AXS binaries (DIFFRAC.SUITE V3 and higher)
.DAT.XDD, *.CAL*.XY, *XYE	ASCII file formats, see Table 14.2
*.HKL	ShelX HKL4 format.
*.SCR	ASCII file format. Consists of lines comprising h, k, l, m, d, 2θ, and Fo.
Peak Profile Parameter Data	
*.DIF	Bruker DIF binaries. Can be used to import d-I data from the PDF (ICDD).
*.UXD	Bruker ASCII file format. Can be used to import d-I data from the PDF (ICDD).
Structure and structure factor data	
*.CIF	Crystallographic Information File; International Union for Crystallography (IUCr).
*.FCF	CIF file representation of structure factor details suitable for generating Fourier maps using ShelX

Table 14.2: ASCII input data file formats. *.XY, *.XYE. *.XDD and *.CAL are delimited by white space characters and can contain line and block comments.

File Type:	Format:	Explanation:
***.DAT**		
• **LHPM/RIET7/CSRIET**		
	Line 1-4	Comments
	Line 5	Start angle, step width, finish angle
	Line 6 onwards	Observed XRD data points (any number of rows)
• **GSAS** ("std - const", "alt - ralf"), use keyword *gsas_format*		
	Line 1	Legend
	Line 2	Item 3: Number of data points
	Line 3 onwards	Depending of item10 and item5
		For item10 = "STD" and item5 = "CONST" xmin = item6/div step =item7/div read(10(i2,F6.0) iww(i),y(i) i=1, npts sigma(i)=sqr(y(i)/iww(i)) i=1, npts For item10 = "ALT" and item5 = "RALF" xmin = item6/32 step = item7/32 read(4(F8.0,F7.4,F5.4) x(i), y(i), sigma(i) i=1, npts x(i) = x(i)/32 i=1, npts do i = 1, npts-1 div = x(i+1)-x(i) y(i) =1000 * y(i)/div sigma(i) = 1000 * sigma(i)/div end do rk (constant wavelength data): div = 100 rk (time of flight data): div = 1
• **FullProf** (INSTRM = 0: free format file), use keyword *fullprof_format*		
	Line 1	Start angle, step width, finish angle, comments
	Line 2 onwards	Observed XRD data points (any number of rows)
***.XDD / *.CAL**	Line 1	Optional line for comments
	Number 2	Start angle
	Number 3	Step width
	Number 4	Finish angle
	Number 5	Counting time
	Number 6	Unused
	Number 7	Unused
	Number 8 onwards	Observed XRD data points
***.XY**		2θ and intensity data values
***.XYE**		2θ, intensity and intensity error values

15 PROGRAM DEFAULTS

TOPAS has been designed to minimise user input by implementation of a general defaults mechanism using ".DEF" files stored in the TOPAS program directory:

- **STARTUP.DEF**
  This file is loaded when the GUI is started. Its contains the global display defaults which all are set from within TOPAS, therefore there is no need to edit this file directly.
- **RANGE.DEF**
  This file is loaded and attached to ranges as they are created in the GUI. All range dependent GUI keywords/macros that can appear in INP files can also appear in this DEF file.
- **STR.DEF**
  This file is loaded and attached to structures as they are created in the GUI. All structure dependent GUI keywords/macros that can appear in INP files can also appear in this DEF file.
- **HKLI.DEF**
  This file is loaded and attached to hkl_Is phases as they are created in the GUI. All hkl_Is dependent GUI keywords/macros that can appear in INP files can also appear in this DEF file.
- **CIF.DEF**
  This file is loaded when a CIF file is attached to structure phases. All structure dependent GUI keywords/macros that can appear in INP files can also appear in this DEF file.

As an example the RANGE.DEF file installed by default has the following content:

```
CuKa2(0.001)
bkg @ 0 0
```

This forces TOPAS to always load the CuKa2 emission profile and to refine on a 1st order background polynomial by default. If always the same instrument is used to generate diffraction data, it may make sense to include the instrument parameters into RANGE.DEF, which then could look like this:

```
CuKa2(0.001)
bkg @ 0 0
Rp 217.5                        ' primary instrument radius
Rs 217.5                        ' secondary instrument radius
Slit_Width(0.2)                 ' detector slit width
Divergence(1)                   ' fixed equatorial divergence
axial_conv
   primary_soller_angle    2.3  ' primary soller slit
   secondary_soller_angle  2.3  ' secondary soller slit
```

16 AUTOMATED TOPAS OPERATION

TOPAS offers batch mode operation with its command line version TC.EXE installed in the TOPAS directory. Running TC.EXE at the DOS command prompt without a command line parameter will display the help screen shown in Fig. 16.1.

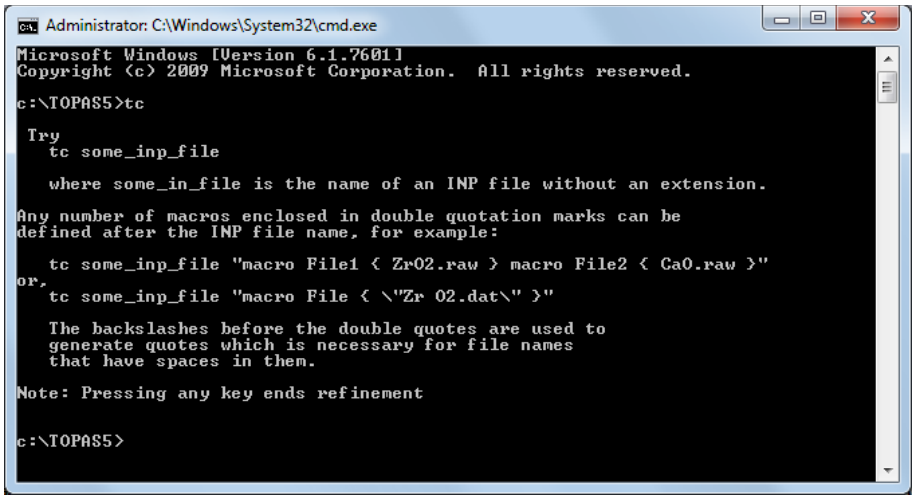


Fig. 16.1: Help screen of TC.EXE.

Any INP file can be run with TC as follows:

tc filename

where the extension INP is not included.

It is recommended to always use full file names including path, enclosed in double quotation marks (double quotation marks are mandatory when using file or path names including space characters!).

The examples BATCH.BAT[1] demonstrates batch operation of TOPAS on a series of eight data sets.

1 \TUTORIAL\MISCELLANEOUS\TOPAS Batch Operation\

17 REFERENCES

Alexander, L.E. (1954): *The synthesis of x-ray spectrometer line profiles with application to crystallite size measurements.* - J. Appl. Phys., **25**, 155-161

Baerlocher, C., Gramm, F., Massüger, L., McCusker, L.B., He, Z., Hovmöller, S. and Zou, X. (2007a): *Structure of the polycrystalline zeolite catalyst IM-5 solved by enhanced charge flipping.* - Science, **315**, 1113-1116

Baerlocher, C., McCusker, L.B., & Palatinus, L. (2007b): *Charge flipping combined with histogram matching to solve complex crystal structures from powder diffraction data.* - Z. Krist., **222**, 47-53

Balzar, D. (1999): *Voigt-function model in diffraction line broadening analysis.* - Microstructure Analysis from Diffraction, edited by R. L. Snyder, H. J. Bunge, and J. Fiala, International Union of Crystallography, 1999.

Berger, H. (1986): *Study of the Kalpha emission spectrum of copper.* X-ray Spectrometry, **15**, 241-243.

Bergmann, J., Kleeberg, R., Haase, A. & Breidenstein, B. (2000): *Advanced Fundamental Parameters Model for Improved Profile Analysis.* - Mat. Sci. Forum, **347-349**, 303-308.

Bertaut, E. F. (1949): *Etude aux rayons X de la repartition des dimensions des cristallites dans une poudre cristalline.* - C. R. Acad. Sci. Paris, **228**, 492-494.

Brindley, G.W. (1945): *The effect of grain or particle size on X-ray reflections from mixed powders and alloys, considered in relation to the quantitative determination of crystalline substances by X-ray methods.* - Phil. Mag., **36**, 347-369.

Broyden, C.G. (1970): *The Convergence of a Class of Double-rank Minimization Algorithms.* - J. Inst. Maths. Applics., **6**, 76-90.

Cagliotti, G., Paoletti, A. & Ricci, F.P (1958): *Choice of collimators for a crystal spectrometer for neutron diffraction.* - Nucl. Inst., **3**, 223-228.

Cheary, R.W. & Coelho, A.A. (1992): *A Fundamental Parameters Approach of X-ray Line-Profile Fitting.* - J. Appl. Cryst. 25, 109.

Cheary, R.W. & Coelho, A.A. (1994): *Synthesising and Fitting Linear Position-Sensitive Detector Step-Scanned Line Profiles.* - J Appl. Cryst., **27**, 673 - 681.

Cheary, R.W. & Coelho, A.A. (1998a): *Axial divergence in a conventional X ray powder diffractometer I. Theoretical foundations.* - J. Appl. Cryst., **31**, 851-861.

Cheary, R.W. & Coelho, A.A. (1998b): *Axial divergence in a conventional X ray powder diffractometer II. Implementation and comparison with experiment.* - J. Appl. Cryst., **31**, 862-868.

Cheary, R.W. & Coelho, A.A. (1998c): *An experimental investigation of the effects of axial divergence on diffraction line profiles.* - Powder Diffraction, **13**, 100-106.

Cheary, R.W., Coelho, A.A. & Cline, J.P. (2004): *Fundamental Parameters Line Profile Fitting in Laboratory Diffractometers.* - J. Res. Natl. Inst. Stand. Technol., **109**, 1-25.

Chipera, S.J. and Bish, D.L. (2002): *FULLPAT: a full-pattern quantitative analysis program for X-ray powder diffraction using measured and calculated patterns.* - J. Appl. Cryst., **35**, 744-749.

Coelho, A.A. (2000): *Whole Profile Structure Solution from Powder Diffraction Data using Simulated Annealing.* - J. Appl. Cryst., **33**, 899-908.

Coelho, A.A. (2003): *Indexing of powder diffraction patterns by iterative use of singular value decomposition.* - J. Appl. Cryst, **36**, 86–95.

Coelho, A. A. (2005): *A bound constrained conjugate gradient solution method as applied to crystallographic refinement problems.* - J. Appl. Cryst., **38**, 455-461.

Coelho, A. A. (2007): *A Charge Flipping algorithm incorporating the tangent formula for solving difficult structures.* - Acta Cryst., **A36**, 400–406.

Coelho, A. A. & Cheary, R. W. (1997): *A fast and simple method for calculating electrostatic potentials.* - Computer Physics Communications, **104**, 15-22

Coelho, A. A. & Kern A. (2005): *Discussion of the indexing algorithms within TOPAS.* - CPD Newsletter No. 32, 43-45

Chernick, M.R. (1999): *Bootstrap Methods, A Practitioner's Guide.* - Wiley, New York.

DiCiccio, T.J. & Efron, B. (1996): *Bootstrap confidence intervals.* - Statist. Sci., **11**, 189–228.

David, W.I.F. & Jorgensen, J.D. (1993): *Rietveld refinement with time-of-flight powder diffraction data from pulsed neutron sources.* - The Rietveld Method, edited by R.A. Young, IUCr Book Series, Oxford University Press 1993, 197-226.

David, W.I.F., Leoni, M. and Scardi, P. (2010): *Domain size analysis in the Rietveld method.* - Materials Science Forum, **651**, 187-200.

David, W.I.F. & Sivia, D.S. (2002): *Extracting integrated intensities from powder diffraction patterns.* - Structure Determination from Powder Diffraction Data, edited by David, W.I.F., Shankland, K., McCusker, L.B. & Baerlocher, Ch., IUCr Book Series, Oxford University Press, 337 pages.

David, W.I.F., Shankland, K., McCusker, L.B. & Baerlocher, Ch. (2002): *Structure Determination from Powder Diffraction Data.* - IUCr Book Series, Oxford University Press, 337 pages.

Durbin, J. & Watson, G.S. (1971): *Testing for Serial Correlation in Least Square Regression, III.* - Biometrika, **58**, 1-19.

Efron, B. & Tibshirani, R. (1986): *Bootstrap methods for standard errors, confidence intervals and other measures of statistical accuracy.* - Statist. Sci., **1**, 54–77.

Evans, J.S.O. (2008). Personal communication.

Favre-Nicolin, V. & Cerny. R. (2004): *Fox: Modular Approach to Crystal Structure Determination from Powder Diffraction.* - Material Science Forum, **443-444**, 35-38.

Finger, L.W., Cox, D.E & Jephcoat, A.P. (1994): *A correction for powder diffraction peak asymmetry due to axial divergence.* - J. Appl. Cryst., **27**, 892-900.

Flack, H. D. (1983): *On enantiomorph-polarity estimation.* - Acta Cryst. **A39**, 876-881

Fletcher, R. (1970): *A New Approach to Variable Metric Algorithms.* - Computer Journal, **13**, 317-322.

Goldfarb, D. (1970): *A Family of Variable Metric Updates Derived by Variational Means.* - Mathematics of Computing, **24**, 23-26.

Gozzo, F., De Caro, L., Giannini, C., Guagliardi, A., Schmitt, B. & Prodia, A. (2006): *The instrumental resolution function of synchrotron radiation powder diffractometers in the presence of focusing optics.* - J. Appl. Cryst., **39**, 347–357

Haberkorn, R. (1999). Personal communication.

Hill, R.J. & Flack, H.D. (1987): *The Use of the Durbin-Watson d Statistic in Rietveld Analysis.* - J. Appl. Cryst., **20**, 356-361.

Hill, R.J. and Howard, C.J. (1987): *Quantitative Phase Analysis from Neutron Powder Diffraction Data using the Rietveld Method.* - J. Appl. Cryst., **20**, 467-474.

Hölzer, G., Fritsch, M., Deutsch, M., Härtwig, J. & Förster, E. (1997): *$K\alpha_{1,2}$ and $K\beta_{1,3}$ X-ray emission lines of the 3d transition metals.* - Physical Review A, **56**, 4554-4568.

Hovestreydt, E. (1983): *On the atomic scattering factor for $O2^{-}$.* Acta Cryst., **A39**, 268-269.

Järvinen, M. (1993): *Application of symmetrized harmonics expansion to correction of the preferred orientation effect.* - J. Appl. Cryst., **26**, 525-531.

Jenkins, R. & Snyder, R.L. (1996): *Introduction to X-Ray Powder Diffractometry.* - John Wiley & Sons, New York, 391 pages.

Karle, J. & Hauptman, H. (1956): *A theory of phase determination for the four types of non-centrosymmetric space groups 1P222, 2P22, 3P12, 3P22.* - Acta Cryst., **9**, 635-651.

Kern, A., Cheary, R.W., Coelho, A.A. (2004): *Convolution Based Profile Fitting.* - Diffraction Analysis of the Microstructure of Materials. Editors: Mittemeijer, E.J. & Scardi, P. Springer Series in Materials Science, Vol. 68, 552 pages.

Kern, A. (2008): *Convolution Based Profile Fitting.* - Principles and Applications of Powder Diffraction. Editors: Clearfield, A., Bhuvanesh N. & Reibenspies J. Blackwell Publishers, 400 pages.

Kern, A., Madsen, I.C. and Scarlett, N.V.Y. (2012): *Quantifying amorphous phases.* - In "Uniting Electron Crystallography and Powder Diffraction". Editors: Kolb, U., Shankland, K., Meshi, L., Avilov, A. & David, W. Springer, 434 pages.

Klug, H. P. & Alexander, L. E. (1974): *X-ray diffraction procedures.* - 2nd Edition, J. Wiley and Sons Inc., New York, 996 pp.

Larson, A.C. & Von Dreele, R.B. (2004): *General Structure Analysis System (GSAS).* - Los Alamos National Laboratory Report LAUR 86-748.

Laue, M. (1926): *Lorentzfaktor und Intensitätverteilung in Debye-Scherrer Ringen*. - Z. Krist., **64**, 115-142.

Le Bail, A., Duroy, H. & Fourquet, J. L. (1988): *Ab-initio structure determination of $LiSbWO_6$ by X-ray powder diffraction.* - Materials Research Bulletin, **23**, 447-452.

Le Bail, A. (1995): *Modelling the silica glass structure by the Rietveld method.* - J. Non-Cryst. Solids, **183**, 39-42.

Leineweber, A. (2010): *Composition-induced microstrain broadening: from pattern decomposition to whole powder pattern modelling procedures.* - Materials Science Forum, **651**, 131-153.

Leineweber, A. (2011): *Understanding anisotropic microstrain broadening in Rietveld refinement.* - Z. Krist., **226**, 905–923.

Lutterotti, L., Ceccato, R., Dal Maschio, R. and Pagani, E. (1998): *Quantitative analysis of silicate glass in ceramic materials by the Rietveld method.* - Mater. Sci. Forum, **Vols. 278-281**, 87-92, 1998.

Madsen, I.C. (1999). Personal communication.

Madsen, I.C. and Scarlett, N.V.Y. (2008): *Quantitative Phase Analysis.* - In "Powder Diffraction: Theory and Practice". Editors: Dinnebier, R.E. and Billinge, S.J.L. The Royal Society of Chemistry, Cambridge, UK, 582 pages.

Madsen, I.C. and Scarlett, N.V.Y., Riley, D.P. and Raven, M.D. (2012): *Quantitative Phase Analysis using the Rietveld Method.* - In "Modern Diffraction Methods". Editors: Mittemeijer, E.J. and Welzel, U. Wiley-VCH, 554 pages.

Madsen, I.C., Scarlett, N.V.Y. and Kern, A. (2011): *Description and survey of methodologies for the determination of amorphous content via X-ray powder diffraction.* - Z. Krist., **226**, 944-955.

March, A. (1932): *Mathematische Theorie der Regelung nach der Korngestalt bei affiner Deformation.* - Z. Krist., **81**, 285-297.

Marquardt, D. W. (1963): *An algorithm for least-squares estimation of nonlinear parameters.* - J. Soc. Ind. Appl. Math., **11(2)**, 431-331.

Masson, O., Dooryhée, E. and Fitch, A. N. (2003): *Instrument line-profile synthesis in high-resolution synchrotron powder diffraction.* - J. Appl. Cryst., **36**, 286-294

Mittemeijer, E.J. & Scardi, P. (2004): *Diffraction Analysis of the Microstructure of Materials.* - Springer Series in Materials Science, Vol. 68, 552 pages.

O'Connor, B.H. & Raven, M.D. (1988): *Application of the Rietveld Refinement Procedure in Assaying Powdered Mixtures*. - Powder Diffraction, **3**, 2-6.

Oszlányi, G. & Süto A. (2004): *Ab initio structure solution by charge flipping.* - Acta Cryst., **A60**, 134-141.

Oszlányi, G. & Süto A. (2005): *Ab initio structure solution by charge flipping. II. Use of weak reflections.* - Acta Cryst., **A61**, 147-152.

Parrat, L. G. (1936): *Kα Satellite Lines.* - Physical Review, **50**, 1-15.

Pawley, G. S. (1981): *Unit-cell refinement from powder diffraction scans.* J. Appl. Cryst, **14**, 357-361.

Pitschke, W., Mattern, N. & Hermann, H. (1993): *Incorporation of microabsorption corrections in Rietveld analysis.* - Powder Diffraction, **8(4)**, 223-228.

Popa, N.C. (1998): *The (hkl) Dependence of Diffraction-Line Broadening Caused by Strain and Size for all Laue Groups in Rietveld Refinement.* - J. Appl. Cryst., **31**, 176-180.

Rietveld, H.M. (1967): *Line profiles of neutron powder-diffraction peaks for structure refinement.* - Acta Cryst., **22**, 151-152.

Rietveld, H.M. (1969): A Profile Refinement Method for Nuclear and Magnetic Structures. - Journal of Applied Crystallography, **2**, 65-71.

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.* - J. Appl. Cryst., **31**, 47-51.

Scardi, P. & Leoni, M. (2001): *Diffraction line profiles from polydisperse crystalline systems.* - Acta Cryst., **A 57**, 604-613.

Scardi, P and Leoni, M. (2004): *Whole Powder Pattern Modelling: Theory and Applications.* - Diffraction Analysis of the Microstructure of Materials. Editors: Mittemeijer, E.J. & Scardi, P. Springer Series in Materials Science, Vol. 68, 552 pages.

Scardi, P., Leoni, M. & Delhez, R. (2004): *Line broadening analysis using integral breadth methods: a critical review.* - J. Appl. Cryst., **37**, 381-390.

Scardi, P., Ortolani, M. and Leoni, M. (2010): *WPPM: Microstructural analysis beyond the Rietveld method.* - Materials Science Forum, **651**, 155-171.

Scarlett, N.V.Y. & Madsen, I.C. (2006): *Quantification of phases with partial or no known crystal structure.* - Powder Diffraction, **21(4)**, 278-284.

Scherrer, P. (1918): *Bestimmung der Grösse und der inneren Struktur von Kolloidteilchen mittels Röntgenstrahlen.* - Nachr. Ges. Wiss. Göttingen, 26, 98-100.

Schneider, T.R. & Sheldrick, G.M. (2002): *Substructure solution with SHELXD.* - Acta Cryst., **D58**, 1772-1779.

Schulze, D.G. (1984): *The influence of aluminum on iron oxides. VIII. Unit-cell dimensions of Al-substituted goethites and estimation of Al from them.* - Clays and Clay Minerals, **32**, 36-44.

Schwarzenbach, D., Abrahams, S.C., Flack, H.D., Prince, E. & Wilson, A.J.C. (1995): *Statistical Descriptors in Crystallography. II. - Report of a Working Group on Expression of Uncertainty in Measurement. - Acta Cryst.*, **A51**, 565-569.

Shanno, D.F. (1970): *Conditioning of Quasi-Newton Methods for Function Minimization.* - Mathematics of Computing, **24**, 647-656.

Shiono, M. & Woolfson, M. M. (1992): *Direct-space methods in phase extension and phase determination. I. Low-density elimination.* - Acta Cryst., **A48**, 451-456

Stephens, P. W. (1999): *Phenomenological model of anisotropic peak broadening in powder diffraction.* - J. Appl. Cryst., **32**, 281-289.

Stone, K. H., Lapidus, S. H & Stephens P. W. (2009): *Implementation and use of robust refinement in powder diffraction in the presence of impurities.* - J. Appl. Cryst., **42**, 385-391

Suortti, P. (1972): *Effects of porosity and surface roughness on the x-ray intensity reflected from a powder specimen.* - J. Appl. Cryst., **5**, 325-331.

Toby, B.H. (2006): *R factors in Rietveld analysis: How good is good enough?* - Powder Diffraction, **21**, 67-70.

Williamson, G. K. & Hall, W. H.(1953): *X-ray line broadening from filed aluminium and wolfram*. - Acta Metall., **1**, 22-31.

Warren, B. E. & Averbach, B. L. (1950): *The effect of cold-work distortion on X-ray patterns.* - J. Appl. Phys., **21**, 595-599.

Will, G. (1979): *POWLS: A powder least-squares program.* - J. App. Cryst, **12**, 483-485.

Will, G. (2006): *Powder Diffraction: The Rietveld Method and the Two-Stage Method.* - Springer, 224 pages.

Wilson, A.J.C. (1963): *Mathematical Theory of X-ray Powder Diffractometry,* pp. 1-53. New York: Gordon & Breach

Wolff, P.M. de (1968): *A simplified criterion for the reliability of a powder pattern indexing.* - J. Appl. Cryst., **1**, 108-113.

Young, R.A. (1993): *Introduction to the Rietveld method.* - The Rietveld Method, edited by R.A. Young, IUCr Book Series, Oxford University Press 1993, 1-39.