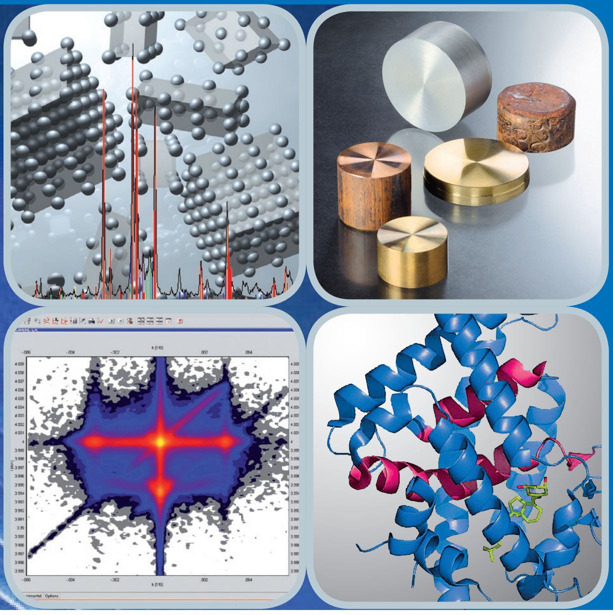
Bruker AXS GmbH



DIFFRAC^{plus} TOPAS

● TOPAS 4.2 Technical Reference

think forward

XRD

Order no. DOC-M88-EXX066 V4.2. Updated: Jan 15, 2009.

Printed in the Federal Republic of Germany.

Bruker AXS GmbH
Östliche Rheinbrückenstr. 49
76187 Karlsruhe, Germany
Tel. +49 (7 21) 5 95-28 88
Fax +49 (7 21) 5 95-45 87
info@bruker-axs.de
www.bruker-axs.com

CONTENTS

(intentionally left blank)

1 INTRODUCTION

This document describes all TOPAS functionality 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/modeling 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
• *hkl_Is_from_hkl4*	:	Structure factors $(F_{obs})^2$ for creating a powder pattern from single crystal data

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 7.

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 (ie. = a+b;) or constant (ie. 1.245) or a parameter name with a value (ie. 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 TOPAS vs. TOPAS P

TOPAS P is a TOPAS variant designed for profile analysis of powder data **without reference to a crystal structure model**, structure analysis is not available.

Fit objects supported by both variants are indicated in Table 1-1. TOPAS P users can ignore descriptions of all structure analysis related keywords (i.e. *str* and its dependents) in this manual.

Table 1-1: Fit objects supported by TOPAS and TOPAS P.

Fit Objects	TOPAS	TOPAS P
xo_Is, d_Is	✓	✓
hkl_Is	✓	✓
str	✓	✗

1.5 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:\TOPAS4 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* E] 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;
```

local can greatly simplify complex INP files.

2.3 Parameter constraints

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.

2.4 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.

2.5 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
```

2.6 Parameter errors and correlation matrix

When *do_errors* is defined parameter errors 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.

2.7 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*.

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

2.8 Reserved parameter names

2.8.1 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.

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 7.2.
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 equations (6-5a) and (6-5b), 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, $\text{Iobs\_no\_scale\_pks} = \text{Get}(scale) I_p \sum_x P_{x,p} \frac{Y_{obs,x}}{Y_{calc,x}}$ 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.

2.8.2 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;
```

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.

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 true if a = b
	a < b	Returns true if a < b
	a <= b	Returns true if a <= b
	a > b	Returns true if a > b
	a >= b	Returns true if a >= b
	And(a, b, ...)	Returns true if all arguments evaluates to true
	Or(a, b, ...)	Returns true if one arguments evaluates to true
Mathematical	Sin(x)	Returns the sine of x
	Cos(x)	Returns the cosine of x
	Tan(x)	Returns the tangent of x
	ArcSin(x)	Returns the arc sine of x (-1 <= x <= 1)
	ArcCos(x)	Returns the arc cos of x (-1 <= x <= 1)
	ArcTan(x)	Returns the arc tangent of x
	Exp(x)	Returns the exponential e to the x
	Ln(x)	Returns the natural logarithm of x
	Sqrt(x)	Returns the positive square root
Special	Sum(returns summation_eqn, initializer, conditional_test, increment_eqn)	
	If(conditional_test, return true_eqn, return false_eqn)	
	For(Mi = 0, Mi < M, Mi = Mi+1 ,....)	
	Get($keyword)	
Miscallenous	Min(a,b,c...)	Returns the min of all arguments
	Max(a,b,c...)	Returns the max of all arguments
	Abs(x)	Returns the absolute value of x
	Mod(x, y)	Returns the modulus of x/y. Mod(x, 0) returns 0
	Rand(x1, x2)	Returns a random value between x1 and x2

Classes:	Symbols / Functions:	Remarks:
	Sign(x)	Returns the sign of x, or zero if x = 0
	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);

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.

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 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 default non-linear least squares routine used is based on the Newton-Raphson method with the Marquardt (1963) method 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:

$$\chi^2 = \chi_1^2 + \chi_2^2 \tag{4-1}$$

where

$$\chi_1^2 = K \sum_{m=1}^{M} w_m \left(Y_{o,m} - Y_{c,m}\right)^2 \text{ and } \chi_2^2 = K\, K_1 \sum_{p=1}^{N_P} K_{2,p} P_p \tag{4-2}$$

where

$$K = 1 \Big/ \sum_{m=1}^{M} w_m\, Y_{o,m} \tag{4-3}$$

$Y_{o,m}$ 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, N_p the number of penalty functions and K_1 and $K_{2,p}$ are weightings applied to the penalty functions and are described below. K normalizes χ^2 such that the default *chi2_convergence_criteria* value of 0.001 is sufficient for routine refinements. Typical *chi2_convergence_criteria* values for structure determination range from 0.01 to 0.1. Penalty functions 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** ignoring second order terms. 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 resulting normal equations are:

$$\mathbf{A}\, \Delta\mathbf{p} = \mathbf{Y} \tag{4-4}$$

where $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$

$$A_{1,ij} = \sum_{m=1}^{M} w_m \frac{\partial Y_{c,m}}{\partial p_i} \frac{\partial Y_{c,m}}{\partial p_j}$$

$$A_{2,ij} = K_1\, K_{2,i}\, B_{2,ij} \quad \text{where } B_{2,ij} = \frac{1}{2} \sum_{p=1}^{N_P} \frac{\partial^2 P_P}{\partial p_i \partial p_j} \tag{4-5}$$

$$Y_i = \sum_{m=1}^{M} w_m \left(Y_{o,m} - Y_{c,m}\right) \frac{\partial Y_{c,m}}{\partial p_i} - \frac{K_1\, K_{2,i}}{2} \sum_{p=1}^{N_P} \frac{\partial P_P}{\partial p_i}$$

The Taylor coefficients $\Delta\mathbf{p}$ correspond to the changes in the parameters $\mathbf{p}$. Equation (4-4) represents a linear set of equations in $\Delta\mathbf{p}$ that are solved for each iteration of refinement. The calculation of the off diagonal terms in $\mathbf{A}_2$ (the second derivatives of the penalty functions) are tedious and preliminary investigations have indicated that their inclusion does not significantly improve convergence of χ^2; thus $A_{2,ij}$ for $i \neq j$ has been set to zero.

The penalty weighting $K_{2,i}$ is used to give equal weights to the sum of the inverse error terms in the parameters $\sigma_1(p_i)^2$ and $\sigma_2(p_i)^2$ calculated from χ_1^2 and χ_2^2 respectively. Neglecting the off diagonal terms gives $\sigma_1(p_i)^2 = 1/A_{1,ii}$ and $\sigma_2(p_i)^2 = 1/B_{2,ii}$ and thus $K_{2,i}$ is written as shown in equation (4-6).

$$K_{2,i} = \mathrm{Min}\left(\frac{1}{4} \sum_{i=1}^{N} A_{1,ii} \Big/ \sum_{i=1}^{N} B_{2,ii} \,, \quad A_{1,ii} / B_{2,ii} \right) \tag{4-6}$$

The penalty weighting K_1 determines the weight given to the penalties χ_2^2 relative to χ_1^2, typical values range from 0.2 to 2.

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:

$$\Delta p_i = Y_i \,/\, (A_{ii}\,(1 + \eta)) \tag{4-7}$$

The Marquardt method is used by default when the refinement comprises observed data Y_o. The keyword *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_1^2 and χ_2^2 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 foot prints 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 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 tutorial examples 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*.

4.6 Summary, Iteration and Refinement Cycle

Table 4-1 shows various keywords usages for typicaly 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 used if *line_min*. 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.

4.7 *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 simulated annealing, 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.

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. (4-8) is met for three consecutive iterations.

$$\Delta p_i < \left(0.01\,\mathrm{quick\_refine} \,/\, \left(K\,N\,Y_i\right)\right) \tag{4-8}$$

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 *quick_refine_remove* is not defined then all parameters are reinstated at the start of refinement cycles.

4.8 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. (4-9), where T is the current temperature and K is as defined in Eq. (4-3).

$$\Delta p_i = Q\,\mathrm{Sign}(\mathrm{Rand}(-1,1))\,\sqrt{0.02\,T/\left(K\,A_{ii}\right)} \tag{4-9}$$

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.

4.9 Criteria of fit

Criteria of fit used in TOPAS are shown in Table 4-2 (see Young 1993 for details).

Table 4-2: 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 *k*th 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)	$R_{\exp} = \sqrt{\frac{\sum M - P}{\sum w_m Y_{o,m}^2}}$	$R_{\exp}' = \sqrt{\frac{\sum M - P}{\sum w_m (Y_{o,m} - Bkg_m)^2}}$
"Goodness of fit", GOF	$GOF = chi^2 = \frac{R_{wp}}{R_{\exp}} = \sqrt{\frac{\sum w_m (Y_{o,m} - Y_{c,m})^2}{M - P}}$	
"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	$d = \frac{\sum_{m=2}^{M} (\Delta Y_m - \Delta Y_{m-1})}{\sum_{m=1}^{M} (\Delta Y_m)^2};$	$\Delta Y_m = Y_{o,m} - Y_{c,m}$

5 GENERATION OF PHASE PEAKS AND "PEAK_TYPE"

A number of analytical profile shapes can be convoluted with predefined or user-defined functions. Analytical convolutions are used where possible.

Convolution implies integration. A function analytically integrated is exact whereas numerical integration is an approximation with accuracy dependent on the step size used for integration. When analytical convolution is not possible then TOPAS allows for accurate numerical convolution making it possible to include complex functions in the generation of peak shapes.

Numerical convolution is important in regards to laboratory powder diffraction data as many of the instrument aberration functions cannot be convoluted analytically. The process of convolution from a fundamental parameters perspective is an approximation whereby second order effects and higher are typically neglected. These approximations are valid except for extreme cases that are unlikely to exist in practice, for example, axial divergence with Soller slits acceptance angles that are greater than about 12 degrees.

5.1 Source emission profiles

Generation of the emission profile is the first step in peak generation. It comprises EM 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 [mili-Å] that is convoluted into the emission profile line
- lg Gaussian HW of the emission profile line in [mili-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 from the following:

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

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_Is* phases is calculated from the lattice parameters.

The FWHW$_k$ in [° 2θ] for an EM$_k$ line is determined from the relations provided in Table 5-1.

Table 5-1: FWHW$_k$ in [° 2θ] for an EM$_k$ line for the different peak types.

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 s*pvii* peak types:**

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

4. **For *spv* peak types**

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

When the keyword *no_th_dependence* is defined then the calculation of 2θ$_k$ is determined from

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

The macro No_Th_Dependence can be used when refining on non-X-ray data or fitting to negative 2θ values.

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

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

where EMREF corresponds to the emission profile with the largest *la* value. The default for *ymin_on_ymax* is 0.001.

The emission profile data used in TOPAS, as appearing in the LAM directory, have been taken from Hölzer et al. (1997).

5.2 Peak generation and peak types

Phase peaks P are generated as follows:

P = Get(*scale*) Get(all_scale_pks) *I* EM(*peak_type*) ⊗ Convolutions (5-1)

where the emission profile (EM) is first generated with emission profile lines of type *peak_type*; the symbol ⊗ denotes convolution. Peaks are then convoluted with any defined convolutions, multiplied by the *scale* parameter, multiplied by any defined *scale_pks*, and then multiplied by an intensity parameter. For *xo_Is*, *d_Is* and *hkl_Is* phases the intensity is given by the *I* parameter. For *str* phases it 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 : Fundamental Parameters
- pv : Pseudo-Voigt
- spvii : Split-PearsonVII
- spv : Split-PseudoVoigt

The definitions of the pseudo-Voigt and PearsonVII functions are provided in Table 5-2 (symmetric functions) and Table 5-3 (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:

- *fwhm*1, *fwhm*2 : fwhm for the left and right composite function
- *m*1, *m*2 : Exponents for the composite functions
- $\eta 1$, $\eta 2$: PV mixing parameters for the composite functions

Table 5-2: 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)	$L_{UA}(x)=\left(\frac{l_1}{fwhm}\right)\Big/\left(\frac{1+l_2x^2}{fwhm^2}\right)$ where $l_1=2/\pi$, $l_2=4$
PseudoVoigt, PV_{UA}(x)	$PV=\eta\,L_{UA}(x)+(1-\eta)G_{UA}(x)$

Table 5-3: Unit area peak types for split functions.

Profile Function:	**Definition:**
Split PearsonVII, SPVII	$SPVII = PVII\_Left + PVII\_Right$ where $PVII\_Left = \left(1 + b_1 x^2\right)^{-m1} / a \quad \text{for} \left(-\infty < x < 0\right)$ $PVII\_Right = \left(1 + b_2 x^2\right)^{-m2} / a \quad \text{for} \left(0 < x < \infty\right)$ $a = (½)\Gamma(½)\left[\frac{\Gamma(m1 - ½)}{\Gamma(m1)\sqrt{b_1}} + \frac{\Gamma(m2 - ½)}{\Gamma(m2)\sqrt{b_2}}\right]$ $b_1 = \left(2^{-m1} - 1\right)/h1^2$, $b_2 = \left(2^{-m2} - 1\right)/h2^2$ $fwhm1 = 2\,h1$, $fwhm2 = 2\,h2$ $fwhm = h1 + h2$
Split PseudoVoigt, SPV	$SPV = 2\left(PV\_Left + a\,PV\_Right\right)/\left(1 + a\right)$ where $PV\_Left = PV\left(h1, \eta 1\right) \quad \text{for} \left(-\infty < x < 0\right)$ $PV\_Right = PV\left(h2, \eta 2\right) \quad \text{for} \left(0 < x < \infty\right)$ $a = (PV\_Left\left(x = 0\right)/(PV\_Right\left(x = 0\right)$ $fwhm1 = 2\,h1$, $fwhm2 = 2\,h2$ $fwhm = h1 + h2$

Lorentzian and Gaussian convolutions using *lor_fwhm* and *gauss_fwhm* equations are analytically convoluted with the FP and PV peak types and numerically convoluted with the SPVII and SPV peak types. These numerical convolutions have a high degree of accuracy as they comprise analytical Lorentzian and Gaussian functions convoluted with straight line segments.

For FP and PV peak types, the first defined *hat* convolution is analytically convoluted. Additional hat convolutions for all peak types are convoluted numerically.

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* $\tan\theta$ + *hc/*$\cos\theta$

η = *lora* + *lorb* $\tan\theta$ + *lorc/*$\cos\theta$

where

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

PVII_Peak_Type:

$fwhm1 = fwhm2 = ha + hb \tan\theta + hc/\cos\theta$

$m1 = m2 = 0.6 + ma + mb \tan\theta + mc/\cos\theta$

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):

$\eta = 1.36603\ q - 0.47719\ q^2 + 0.1116\ q^3$

where

- $q = \Gamma_L/\Gamma$
- $\Gamma = (\Gamma_G^5 + A\Gamma_G^4\Gamma_L + B\Gamma_G^3\Gamma_L^2 + C\Gamma_G^2\Gamma_L^3 + D\Gamma_G\Gamma_L^4 + \Gamma_L^5)^{0.2} = \text{fwhm}$
  $A = 2.69269$, $B = 2.42843$, $C = 4.47163$, $D = 0.07842$
- $\Gamma_G = (U \tan^2\theta + V \tan\theta + W + Z/\cos^2\theta)^{0.5}$
- $X \tan\theta + Y/\cos\theta$

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

5.3 Convolution and the peak generation stack

The emission profile of a peak P0 of a certain peak type (ie. 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 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 the peaks in the peaks buffer and then 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 5.1).

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 5-4.

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 5-4: 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 and *gauss_fwhm*	Peak_Calculation_Step for all *lor_fwhm* and *gauss_fwhm* defined.

6 MISCELLANOUS

6.1 Instrument and sample convolutions

Instrument and sample aberration functions used for peak profile synthesis are generated from generic convolutions. For example, the "simple" axial divergence model is described using the generic convolution *circles_conv* as defined in the Simple_Axial_Model macro. Table 6-1 lists some of the instrument convolutions supported. In addition the full axial divergence model as described in Cheary & Coelho (1998a, 1998b) is also supported.

Table 6-1: Instrument and sample aberration functions in terms of $\varepsilon=2\theta-2\theta_k$, where 2θ is the measured angle and $2\theta_k$ the Bragg angle. R_P and R_S correspond to the primary and secondary radius of the diffractometer respectively.

Aberrations:	Name:	Aberration function Fn(ε):
Instrument:		
Equatorial divergence (fixed divergence slits)	EDFA [°]	$Fn(\varepsilon)=(4\varepsilon_m\varepsilon)^{-½}$ for $\varepsilon=0$ to $\varepsilon_m=-(\pi/360)\cot(\theta_k)EDFA^2$ [°2θ]
Equatorial divergence (variable divergence slits)	EDFL [mm]	$Fn(\varepsilon)=(4\varepsilon_m\varepsilon)^{-½}$ for $\varepsilon=0$ to $\varepsilon_m=-EDFL^2\sin(2\theta_k)(180/\pi)/4\,{R_S}^2$ [°2θ]
Size of source in the equatorial plane	TA [mm]	$Fn(\varepsilon)$ = Hat Shape, for $-\varepsilon_m/2<\varepsilon<\varepsilon_m/2$ where $\varepsilon_m=(180/\pi)TA/R_S$ [°2θ]
Specimen tilt; thickness of sample surface as projected onto the equatorial plane	ST [mm]	$Fn(\varepsilon)$ = Hat Shape, for $-\varepsilon_m/2<\varepsilon<\varepsilon_m/2$ where $\varepsilon_m=(180/\pi)\cos(\theta_k)ST/R_S$ [°2θ]
Receiving slit length in the axial plane	SL [mm]	$Fn(\varepsilon)=(1/\varepsilon_m)\left(1-(\varepsilon_m/\varepsilon)^{½}\right)$ for $\varepsilon=0$ to $\varepsilon_m=-(90/\pi)(SL/R_S)^2\cot(2\theta_k)$ [°2θ]
Width of the receiving slit in the equatorial plane	SW [mm]	$Fn(\varepsilon)$ = Hat Shape, for $-\varepsilon_m/2<\varepsilon<\varepsilon_m/2$ where $\varepsilon_m=(180/\pi)SW/R_S$ [°2θ]
Sample:		
Linear absorption coefficient	AB [cm^{-1}]	$Fn(\varepsilon)=(1/\delta)\exp(-\varepsilon/\delta)$ for $\varepsilon<=0$ and $\delta=900\sin(2\theta_k)/(\pi\,AB\,R_S)$ [°2θ]

6.2 Microstructure convolutions

The Double-Voigt approach (e.g. Balzar, 1999) is supported for modeling of microstructure effects. Crystallite size and strain comprise Lorentzian and Gaussian component convolutions varying in 2θ as a function of 1/cos(θ) and tan(θ) respectively.

6.2.1 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 5-2.

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

  $G_{UAH} = G_{UA}(x=0) = g_1 / \text{fwhm}$
  $L_{UAH} = L_{UA}(x=0) = l_1 / \text{fwhm}$

- 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

  $G_H = A\ G_{UAH}$
  $L_H = A\ L_{UAH}$

- Integral breadth of Gaussian and Lorentzian respectively

  $\beta_G = A / G_H = 1 / G_{UAH} = \text{fwhm} / g_1$
  $\beta_L = A / L_H = 1 / L_{UAH} = \text{fwhm} / l_1$

- Unit area Pseudo Voigt, PV_{UA}

  $PV_{UAH} = \eta\ L_{UAH} + (1-\eta)\ G_{UAH}$
  $\beta_{PV} = 1 / PV_{UAH}$

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

$V = G(\text{fwhm}_G) \otimes L(\text{fwhm}_L)$

where "⊗" 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

$V(x) = G(\text{fwhm}_G) \otimes L(\text{fwhm}_L) = PV_{UA}(x, \text{fwhm}_{PV})$

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,

$\text{fwhm}(s) = \text{fwhm}(2\theta) \cos(\theta) / \lambda$
$IB(s) = IB(2\theta) \cos(\theta) / \lambda$

6.2.2 Crystallite size and strain

6.2.2.1 Crystallite size

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

fwhm(2θ) of Gaussian = $(180/\pi)\ \lambda$ / (cos(θ) CS_G)
fwhm(2θ) of Lorentzian= $(180/\pi)\ \lambda$ / (cos(θ) CS_L)
$\beta(2\theta)$ of Gaussian = $(180/\pi)\ \lambda$ / (cos(θ) CS_G g_1)
$\beta(2\theta)$ of Lorentzian = $(180/\pi)\ \lambda$ / (cos(θ) CS_L l_1)

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

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

The macros CS_L and CS_G (see section 8.3.13) 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.

6.2.2.2 Strain

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)
$\beta(2\theta)$ of Gaussian = Strain_G tan(θ) / g_1
$\beta(2\theta)$ 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
$\beta_{GD}(s)/s_0$ s = $\beta(s)$ of Gaussian = (Strain_G / g_1) s / 2
$\beta_{CD}(s)/s_0$ s = $\beta(s)$ of Lorentzian = (Strain_L / l_1) s / 2

The macros Strain_L and Strain_G (see section 8.3.13) are used for calculating the Strain_L and Strain_G parameters respectively.

From these equations we get:

$\beta_{GD}(s) = s_0$ Strain_G / (2 g_1)
$\beta_{CD}(s) = s_0$ Strain_L / (2 l_1)

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

$e = \beta_D(2\theta) / (4 \tan(\theta))$

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.

6.3 Calculation of structure factors

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

$F = \Sigma_s (A_S + i\, B_S)\, \Sigma_a (f_{o,a} + f_a^{'} + i\, f_a^{''})\, O_a$ (6-1)

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 $f_{o,a}$ corresponds 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:

$A_S = \Sigma_e T_{s,e} \cos(2\pi\, h\, re),\quad B_S = \Sigma_e T_{s,e} \sin(2\pi\, h\, re)$ (6-2)

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

$f_{o,s} = \Sigma_a f_{o,a} O_a,\quad f_s^{'} = \Sigma_a f_a^{'} O_a,\quad f_s^{''} = \Sigma_a f_a^{''} O_a$ (6-3)

and separating the real and imaginary components gives:

$F = \Sigma_s (A_s + i\, B_s)\, (f_{o,s} + f_s^{'} + i\, f_s^{''})$ (6-4)

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

or, $F = A + i\, B$

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

$F^2 = A^2 + B^2$ (6-5a)

or,

$F^2 = A_{01}^2 + B_{01}^2 + A_{11}^2 + B_{11}^2 + 2\, B_{01} A_{11} - 2\, A_{01} B_{11}$ (6-5b)

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.

Dispersion coefficients ($f_a^{'}$ and $f_a^{''}$) used are found in the SSF directory. For elements with $Z \leq 92$ they are by default those from http://www.cxro.lbl.gov/optical_constants/-asf.html, covering the energy range from 0.010 to 30 keV. Dispersion coefficients and mass attenuation coefficients for $Z > 92$ have been calculated using the FPrime software (Larson & Von Dreele, 2004), covering the energy range from 4 to 77 keV. 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 O2- from Hovestreydt (1983). These data are in discrete energy steps corresponding to wavelengths typically found in laboratory X-ray tubes.

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.

6.3.1 Friedel pairs

For centrosymmetric structures the intensities for a Friedel reflection pair are equivalent, or, $F^2(h\ k\ l) = F^2(-h\text{-}k\text{-}l)$. 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:

$$F = A_{01} + i\, A_{11} \quad \text{and} \quad F^2 = A_{01}^2 + A_{11}^2 \qquad (6\text{-}6)$$

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 $F^2(h\ k\ l) = F^2(-h\text{-}k\text{-}l)$. Or, for a single atomic species we have:

$$B_{01}\, A_{11} = (f0 + f^{'})\, (\Sigma_S\, B_S)\, f^{''}\, (\Sigma_S\, A_S), \quad A_{01}\, B_{11} = (f0 + f^{'})\, (\Sigma_S\, A_S)\, f^{''}\, (\Sigma_S\, B_S) \qquad (6\text{-}7)$$

or,

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

and thus from cancellation in equation (6-5b) we get

$$F^2(h\ k\ l) = F^2(-h\text{-}k\text{-}l) = A_{01}^2 + B_{01}^2 + A_{11}^2 + B_{11}^2 \qquad (6\text{-}8)$$

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

6.3.2 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 (6-5b) gives the correct intensity for unmerged Friedel pairs and thus it cannot be used for merged Friedel pairs. Using the fact that:

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

$B_{01}(h\ k\ l) = B_{01}(-h\ -k\ -l), \quad B_{11}(h\ k\ l) = B_{11}(-h\ -k\ -l)$ (6-9)

then F^2 from equation (6-5b) in terms of $B_{01}(h\ k\ l)$ and $B_{11}(h\ k\ l)$ evaluates to:

$F^2(h\ k\ l) \quad = Q_1 + Q_2$ (6-10)

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

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:

$F^2(h\ k\ l) + F^2(-h\ -k\ -l) = 2\ Q_1$ (6-11)

The factor 2 in equation (6-11) 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:

$F^2(h\ k\ l)_{merged} = Q_1$ (6-12)

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 8.3.10):

- 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.

6.3.3 Calculation of structure factors – single crystal data

SHELX HKL4 single crystal data comprise unmerged equivalent reflections and thus equation (6-5b) 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 (6-12) is used for calculating F^2. For non-centrosymmetric structures, merging excludes the merging of Friedel pairs and thus (6-5b) 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 (6-12) for calculating F^2. The reserved parameter name Mobs 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 pairs. As a consequence equation (6-12) is used for calculating F^2; any definitions of *dont_merge_equivalent_reflections*, *dont_merge_Friedel_pairs* and *ignore_differences_in_Friedel_pairs* are ignored.

6.3.4 The Flack parameter

For single crystal data and for non-centrosymmetric structures the Flack parameter (Flack, 1983) as implemented scales F2(h) and F2(-h):

F2(h k l) = Q1 + (1 – 2 Flack) Q2 (6-13)

F2(-h -k -l) = Q1 – (1 – 2 Flack) Q2

6.3.5 Single Crystal Output

The macro Out_Single_Crystal_Details, see below, outputs details for a single crystal refinement. 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.

```
macro Out_Single_Crystal_Details(file)
{
      phase_out file load out_record out_fmt out_eqn
      {
            "%4.0f" = H;
            "%4.0f" = K;
            "%4.0f" = L;
            "%4.0f" = Mobs;
            "%4.0f" = M;
            " %11.4f" = A01;
            " %11.4f" = A11;
            " %11.4f" = B01;
            " %11.4f" = B11;

            ' I_no_scale_pks
            '   = Get(scale) Mobs (A01-B11)^2 + (B01+A11)^2; when
            '     ignore_differences_in_Friedel_pairs is NOT defined.
            '   = Get(scale) Mobs (A01^2 + B01^2 + A11^2 + B11^2); when
            '     ignore_differences_in_Friedel_pairs IS defined
            ' If there are no scale_pks then:
            '   I_no_scale_pks = I_after_scale_pks = Ycalc

            " %11.4f" = I_no_scale_pks;
            " %11.4f" = I_after_scale_pks;
            " %11.4f" = Ycalc;
            " %11.4f" = Yobs;
            " %11.4f\n" = SigmaYobs;
      }
}
```

6.4 Lorentz-Polarisation

6.4.1 Predefined Lorentz-Polarisation macros

6.4.1.1 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

6.4.1.2 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 ));
}
```

6.4.2 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).

6.4.2.1 Lorentz-Polarisation Corrections in TOPAS

TOPAS uses:

$$LP\_Factor = \frac{1+\cos^2 2\theta \cos^2 2\theta_M}{\cos\theta \sin^2\theta} \qquad (6\text{-}14)$$

as defined in the LP_Factor macro. This comes from the Lorentz factor:

$$L = \frac{1}{\sin\theta \sin 2\theta} = \frac{1}{\cos\theta \sin^2\theta} \qquad (6\text{-}15)$$

as defined in the Lorentz_factor macro, and polarisation with a monochromator:

$$P = \frac{1-K+K\cos^2 2\theta \cos^2 2\theta_M}{2} \qquad (6\text{-}16)$$

where K is fractional polarisation of the beam.

For neutrons K = 0 and the LP expression becomes:

$$LP = \frac{1}{\cos\theta \sin^2\theta} \qquad (6\text{-}17)$$

In expression (6-14) $2\theta_M = 90$ reduces to (6-17). This is the same as Lorentz factor (6-15).

With no monochromator and unpolarised source K = 0.5 and the LP expression becomes:

$$LP = \frac{0.5 + 0.5\cos^2 2\theta}{2\cos\theta\sin^2\theta} \tag{6-18}$$

Give or take a scale factor using $2\theta_M = 0$ in (6-14) reduces to (6-18).

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 (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:

$$LP = \frac{1}{2\cos\theta\sin^2\theta}\frac{1 - pp + pp\cos^2 2\theta\cos^2 2\theta_M}{1 + pp\cos^2 2\theta_M} \tag{6-19}$$

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:

$$LP = c\left(\frac{0.5 + 0.5\cos^2 2\theta\cos^2 2\theta_M}{2\cos\theta\sin^2\theta}\right) \tag{6-20}$$

which is the same as (6-14), give or take a scale factor.

Use of pp = 0 for fully polarised synchrotron reduces to:

$$LP = \frac{1}{2\cos\theta\sin^2\theta} \tag{6-21}$$

which is again the same as $2\theta_M = 90$ in (6-14).

For a real synchrotron pp=0.05 and the expression becomes:

$$LP = c\left(\frac{0.95 + 0.05\cos^2 2\theta\cos^2 2\theta_M}{2\cos\theta\sin^2\theta}\right) \tag{6-22}$$

This is not significantly different from expression (6-17) in real situations.

6.4.2.2 Releationship to GSAS

In GSAS-speak there are three equations available:

IPOL = 0: $$\frac{Ph + (1 - Ph)\cos^2 2\theta}{2\sin^2\theta\cos\theta} \tag{6-23}$$

IPOL = 1: $$\frac{1 + Ph\cos^2 2\theta}{\sin^2\theta\cos\theta} \tag{6-24}$$

IPOL = 2: $$\frac{1 + Ph\cos^2 2\theta}{\left(1 + \cos^2 2\theta\right)\sin^2\theta\cos\theta} \tag{6-25}$$

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 (6-14) 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:

$$c\left(\frac{0.555+0.444\times\cos^2 2\theta}{2\cos\theta\sin^2\theta}\right) = \frac{c}{0.555}\left(\frac{1+0.8\times\cos^2 2\theta}{\cos\theta\sin^2\theta}\right) \tag{6-26}$$

which is the gsas IPOL = 1 equation.

6.4.2.3 Releationship to FullProf

Fullprof uses:

$$P = \frac{1-K+K\cos^2 2\theta \cos^2 2\theta_M}{2\sin^2\theta\cos\theta} \tag{6-27}$$

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.

6.5 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 of Eq. (5 5) 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 alloted 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.

6.6 Space groups, hkls and symmetry operator generation

The keyword space_group 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.

6.7 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 6-2 shows some *operate_on_points* strings and the corresponding sites identified for this particular example.

Table 6-2: 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

6.8 Occupancies and symmetry operators

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}. It is therefore necessary to enter fractions in the form of equations when entering fractional atomic coordinates that have recurring values such as 0.33333..., 0.666666... etc., for example, use

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

instead of

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

6.9 Pawley and Le Bail extraction using hkl_Is

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

```
hkl_Is
   space_group p-1
```

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

hkls are generated if there are no *hkl_m_d_th2* and *I* keywords defined. 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.

6.10 Anisotropic refinement models

Keywords that can be a function of H, K, L and M, as shown in Table 6-3, allow for the refinement of anisotropic models including preferred orientation, and peak broadening.

Table 6-3: 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 examples of anistropic refinement models are provided:

- Spherical harmonics
- Miscellaneous models using user-defined equations

6.10.1 Spherical harmonics

TOPAS implements a normalized symmetrized sperical harmonics function, see Järvinen (1993). 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.

The user determines how the series is used, typically usage is preferred orientation correction or description of anisotropic line shapes.

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 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 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 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);
```

6.10.2 Miscellaneous models using user-defined equations

Anisotropic peak broadening:

Anisotropic Gaussian convolution 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

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));
```

6.11 Rigid bodies and bond length restraints

6.11.1 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.

6.11.2 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.

6.11.3 Translating part 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,

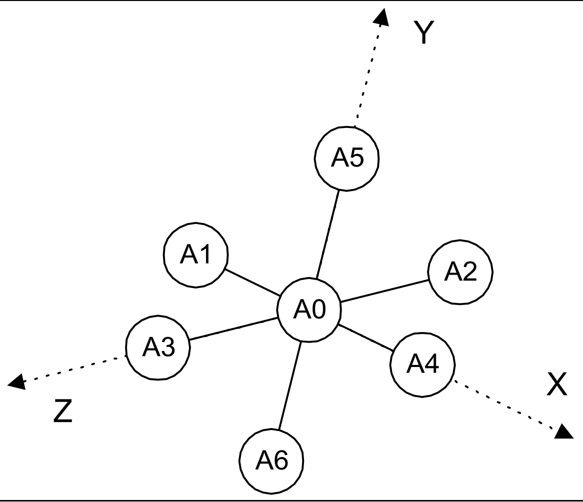
```
prm a 1.3 min 1.2 max 1.4
prm t1 0 min -.1 max .1
prm t2 0 min -.1 max .1
prm t3 0 min -.1 max .1
rigid
   point_for_site C1 ux =  a Sqrt(3) .5 + t1; uy =  a .5 + t2; uz = t3;
   point_for_site C2 ux =  a Sqrt(3) .5 + t1; uy = -a .5 + t2; uz = t3;
   point_for_site C3 ux = -a Sqrt(3) .5;      uy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

6.11.4 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. 6-1) 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. 6-1: Model of an ideal octahedron. A0: central atom A0; 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.

6.11.5 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. 6-2, 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.

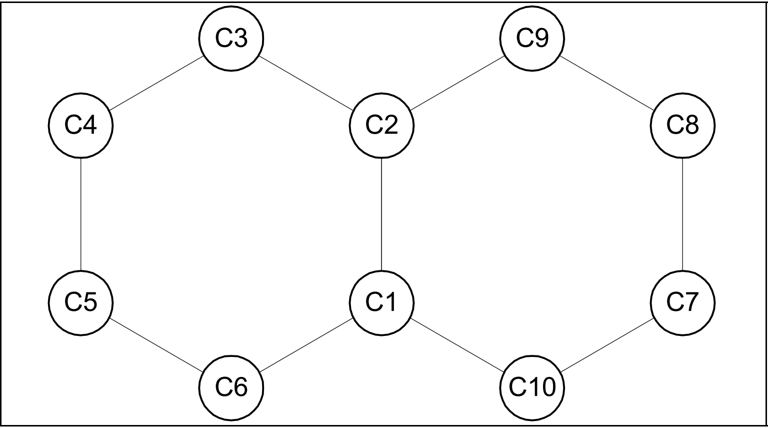


Fig. 6-2: Model of two connected Benzene rings

6.11.6 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. 6-3). 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 is typically used for Cyclopentadienyl and it 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 *z_matrix* keyword gives greater flexibility in defining rigid bodies.

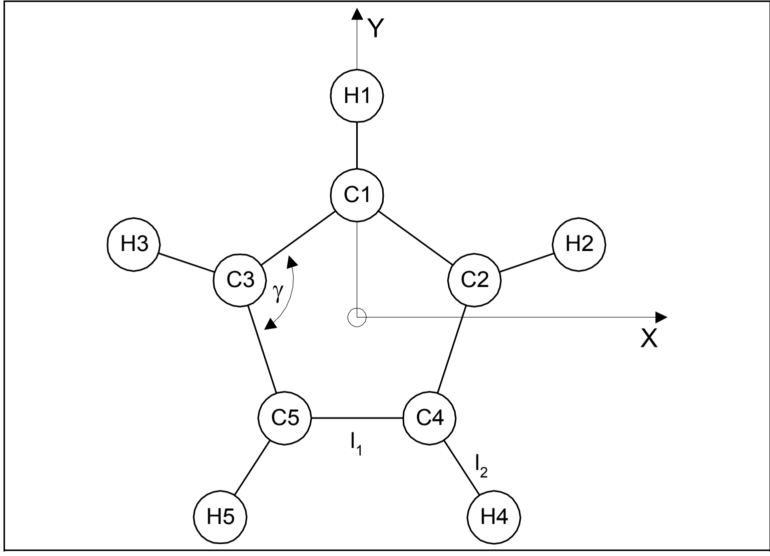


Fig. 6-3: Model of the idealized cyclopentadienyl anion (C_5H_5)..

6.11.7 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 rotates 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.

The Set_Length macro can instead 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 .

min/*max* can be used to constrain the distance between the two sites, for example,

```
Set_Length(A, B, @ 2 min 1.9 max 2.1, @, @, @, @ 30, @ 30)
```

Note, this macro defines the distance between the two sites as a parameter that can be refined.

6.11.8 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 6.1). *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"
```

6.12 Simulated annealing and structure determination

Defining *continue_after_convergence* and a temperature regime is analogous to defining a simulated annealing process. After convergence a new refinement cycle is initiated with parameter values changed according to any defined *val_on_continue* attributes and *rand_xyz* or *randomize_on_errors* processes. Thus simulated annealing is not specific to structure solution, see the tutorial examples ONLYPENA.INP and ROSENBROCK-10.INP

In regards to structure solution in real space, the need for increased computation efficiency is crucial. In many cases computation speed can be increased by up to a factor of 20 or more with the proper choice of keywords. Keywords that facilitate speed are as follows:

```
chi2_convergence_criteria !E
quick_refine !E
yobs_to_xo_posn_yobs !E
```

Another category is one that facilitate structure solution by changing the form of χ^2:

```
penalties_weighting_K1 !E
penalty...
occ_merge...
rigid...
```

Further keywords and processes typically used are:

```
file_name_for_best_solutions
seed
swap_sites...
temperature !E...
   move_to_the_next_temperature_regardless_of_the_change_in_rwp
   save_values_as_best_after_randomization
   use_best_values
   do_processes
xdd... or xdd_scr...
   str...
      site ... rand_xyz...
      break_if_been_there
      try_site_patterns...
```

6.12.1 Penalties used in structure determination

Introducing suitable penalty functions can reduce the number of local minima in χ^2 and correspondingly increase the chances of obtaining a global minimum. The structure factor for a reflection with Miller indices 10 0 0 for a two atom triclinic unit cell with fractional atomic coordinates of (0,0,0) and (*x*, 0,0) is given by 4 $\cos(\pi h x)^2$; here there are 10 local minima for $0<x<1$. If it was known that the bond length distance is half the distance of the *a* lattice parameter then a suitable penalty function would reduce the number of minima to one. In this trivial example it can be seen that the number of minima increases as the Miller indices increase. For non-trivial structures and for the important d spacing range near inter-atomic distances of 1 to 2Å the number of local minima is very large. Bragg reflections with large Miller indices that are heavily weighted are expected to contain many false minima; 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:

```
weighting = 1 / (Sin(X Deg / 2) Max(Yobs,1));
```

A more elaborate scheme which also works well for single crystal data is as follows:

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

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

$$AB_i = \begin{cases} \sum \left(r_{ij} - r_o\right)^2, & \text{for } r_{ij} < r_o \text{ and } i \neq j \\ 0, & \text{for } r_{ij} \geq r_o \end{cases} \tag{6-28}$$

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. 6-29 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 simulated annealing 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.

grs_interaction can be used to either calculate the Lennard-Jones or Born-Mayer potentials and it is suited to ionic atomic models. For a particular site i they comprise a Coulomb term C_i and a repulsive term R_i and is written as:

$$U_i = C_i + R_i \tag{6-29}$$

where

- $C_i = A \sum Q_i Q_j / r_{ij}$, $i \neq j$
- $R_i = \sum B_{ij} / r_{ij}^n$, for Lennard Jones and $i \neq j$
- $R_i = \sum c_{ij} \exp(-d\, r_{ij})$, for Born-Mayer and $i \neq j$

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 the *grs_interaction* is typically used to describe the repulsive terms.

6.12.2 Definition of bond length restraints

The following example defines a bondlength 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.

7 KEYWORDS

7.1 Data structures

Table 7-1 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.

Note that each the Charge Flipping and Indexing methods come with application specific sets of keywords; they are described in section 9 for Charge Flipping, and section 10 for Indexing.

Table 7-1: Keywords and keyword dependencies.

Keyword
Ttop
Tcomm_1
Tcomm_2
Tcharge_flipping
Tglobal
Tindexing
Ttop_xdd
Txdd
Txdd_scr
Ttop_xdd
[*convolution_step* #]
[*r_p* #] [*r_wp* #] [*r_exp* #] [*gof* #] [*r_p_dash* #] [*r_wp_dash* #] [*r_exp_dash* #]
[*Rp* !E] [*Rs* !E]
[*weighted_Durbin_Watson* #]
[*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_convergence_criteria* !E]
[*conserve_memory*]
[*continue_after_convergence*]
[*do_errors*]
[*file_name_for_best_solutions* $file]
[*iters* #]
[*line_min*] [*use_extrapolation*] [*no_normal_equations*] [*use_LU*]
[*marquardt_constant* !E]…
[*no_LIMIT_warnings*]

Keyword

```
[only_penalties]
[out_A_matrix $file]
        [A_matrix_prm_filter $filter]
[out_prm_vals_per_iteration $file]... | [out_prm_vals_on_convergence $file]...
        [out_prm_vals_filter $filter]
[out_rwp $file]
[penalties_weighting_K1 !E]
[percent_zeros_before_sparse_A #]
[process_times]
[[quick_refine !E]  [quick_refine_remove !E]]
[randomise_file_out_normal $file]
[seed]
[temperature !E]...
        [do_processes]
        [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
        [mixture_density_g_on_cm3 !E]
        [mixture_MAC !E]
        [weight_percent_amorphous !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]
        [str]...
                Tcomm_1_2_phase_1_2
                Thkl_lat
                Tmin_max_r
                Trigid
                Tspace_group
                Tstr_details
                [hkl_Is_from_hkl4]
                        Tcomm_1_2_phase_1_2
                        Thkl_lat
                        Tscr_1
                        Tspace_group
```

Keyword

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* [@] # # #...]

[*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]]...
[*neutron_data*]
[*randomize_file_out_normal*]
[*rebin_with_dx_*of !E]
[s*mooth* #]
[*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]...

Keyword

[*exp_conv_const* E [*exp_limit* E]]...
[*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
[*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]...

Tcomm_2

[*lam* [*ymin_on_ymax* #] [*no_th_dependence*] [*la* E *lo* E *lh* E *lg* E]...] [*Lam* !E] [*calculate_Lam*]]
[*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]
[*phase_MAC* !E]
[*phase_name* $phase_name]
[*phase_out* $file [*append*]]...
  Toutrecord
[*r_bragg* #]
[*scale* E]

Tphase_2

[*numerical_area* E]
[*peak_buffer_step* E [*report_on*]]
[*peak_type* $type]

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]...

Keyword

```
[break_if_been_there  $sites !E]
        [been_there_buffer #buffer_size]
        [been_there_clear_buffer !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]]…
[hkl_plane $hkl]…
[grs_interaction [from_N #] [to_N #] [no_self_interaction] $site_1 $site_2 qi # qj # N E]...
[normalize_FCs]
[occ_merge $sites [occ_merge_radius !E]]...
[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]
[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 # #]
[swap_sites $sites_1 $sites_2]…
[try_site_patterns $sites [num_patterns_at_a_time #] ]...
[view_structure]
```

Thkl_lat

```
[a E] [b E] [c E] [al E] [be E] [ga 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]
```

Keyword
[*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 7.3)

7.2 Alphabetical description of keywords

a, *b*, *c*, *al*, *be*, *ga*

Syntax	[*a* E] [*b* E] [*c* E] [*al* E] [*be* E] [*ga* E]
Description	Lattice parameters in Angstroms and lattice angles in degrees.

A_matrix, C_matrix, A_matrix_normalized, C_matrix_normalized

Syntax	[*A_matrix*][*C_matrix*][*A_matrix_normalized*][*C_matrix_normalized*]
Description	Generates the un-normalized and normalized A and correlation matrices. If *do_errors* is defined then *C_matrix_normalized* is automatically generated and appeneded to the OUT file.

aberration_range_change_allowed !E

Syntax	[*aberration_range_change_allowed* !E]
Description	*D*escribes the maximum change allowed in the x-axis extent of a convolution aberration before a new peak is calculated for the peaks buffer.

adps, u11, u22, u33, u12, u13, u23

Syntax [*adps*] [*u11* E] [*u22* E] [*u33* E] [*u12* E] [*u13* E] [*u23* E]

Description *adps* when used generates the *unn* atomic displacement parameters with considerations made for special positions. On termination of refinement the *adps* keword 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. The following INP segments are valid:

```
site C1 ... occ C 1 adps ADPs_Keep_PD
```

```
site C1 ... occ C 1 ADPs_Keep_PD adps
```

However the following is not valid as generation of ADPs are requested whilst unn parameter(s) are defined:

```
site C1 ... occ C 1 adps ADPs_Keep_PD u11 .1
```

amorphous_phase

Syntax	[amorphous_phase]
Description	Signals that the associated phase is amorphous and is e.g. used for calculating *degree_of_crystallinity*.

append_cartesian, append_fractional

Syntax [*append_cartesian*] [*append_fractional* [*in_str_format*]]

Description 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

Syntax [*append_bond_lengths* [*consider_lattice_parameters*]]

Description 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

approximate_A

Syntax [*approximate_A*]
[*A_matrix_memory_allowed_in_Mbytes* !E]
[*A_matrix_elements_tollerance* !E]
[*A_matrix_report_on*]

Description See section 4.3 for a 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_memory_allowed_in_Mbytes* 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.

atomic_interaction, ai_anti_bump

Syntax	[*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*]
Description	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.
Functions	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.

Hint	*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.

atom_out

Syntax	[atom_out $file [append]]… Toutrecord
Description	Used for writing *site* dependent details to a file. See the keyword *out* for a description of *out_record*.

axial_conv

Syntax	[*axial_conv*]... *filament_length* E *sample_length* E *receiving_slit_length* E [*primary_soller_angle* E] [*secondary_soller_angle* E] [*axial_n_beta* !E]
Description	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.
Hint	The macro Full_Axial_Model simplifies the use of *axial_conv*.

bkg

Syntax	[*bkg* [@] # # #...]
Description	Defines a Chebyshev polynomial where the number of coefficients are equal to the number of numeric values appearing after the keyword *bkg*.
Hint	The number of coefficients is not limited.

bootstrap_errors

Syntax	[*bootstrap_errors* !Ecycles] [*fraction_of_yobs_to_resample* !E] [*resample_from_current_ycalc* [*determine_values_from_samples*]]
Description	*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 Rwp 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.

box_interaction

Syntax	[*box_interaction* [*from_N* #] [*to_N* #] [*no_self_interaction*] $site_1 $site_2 N E]...
Description	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.

Example

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.

Hint

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.

break_if_been_there

Syntax

[*break_if_been_there* $sites !E]
[*been_there_buffer* #buffer_size]
[*been_there_clear_buffer* !E]

Description

Breaks the current refinement cycle if the sites identified in $sites are less than !E away to a previous configuration.

[*been_there_buffer*] determines the number of previous site configurations to keep.

[*been_there_clear_buffer*] If defined and if it evaluates to non-zero then the "been there buffer" is cleared.

break_if_been_there is typically not used with *randomize_on_errors* as the latter includes its own version of remembering parameter values and breaking a cycle early.

brindley_spherical_r_cm

Syntax	*brindley_spherical_r_cm* !E
Description	Used for applying the Brindley correction (Brindley, 1945) for spherical particles.
Example	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
```

capillary_diameter_mm

Syntax	[*capillary_diameter_mm* E]... [*capillary_u_cm_inv* E] [*capillary_parallel_beam*] [*capillary_divergent_beam*]
Description	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.
Hint	Both *capillary_parallel_beam* and *capillary_divergent_beam* assume that the capillary is fully illuminated by the beam in the equitorial plane.

cell_mass, *cell_volume*, *weight_percent*, *spiked_phase_measured_weight_percent*, *corrected_weight_percent*

Syntax [*cell_mass* !E] [*cell_volume* !E] [*weight_percent* !E] [*spiked_phase_measured_weight_percent* !E] [*corrected_weight_percent* !E]

Description 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

- N_p = Number of phases
- Q_p = $S_p M_p V_p / B_p$
- 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.

chi2_convergence_criteria

Syntax [*chi2_convergence_criteria* !E]

Description 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.

Example chi2_convergence_criteria = If(Cycle_Iter < 10, .001, .01);

circles_conv

Syntax [*circles_conv* E]...

Description Defines ε_m in the convolution function:

$$(1 - |\varepsilon_m / \varepsilon|^{½}) \qquad \text{for } \varepsilon = 0 \text{ to } \varepsilon_m$$

that is convoluted into phase peaks. ε_m can be greater than or less than zero.

Hint *circles_conv* is used for example by the Simple_Axial_Model macro.

cloud

Syntax [*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]

Description *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 of the GUI.

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 membes 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 Rwp 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.

conserve_memory

Syntax	[*conserve_memory*]
Description	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%.
Hint	Useful when there are many independent parameters.

continue_after_convergence

Syntax	[*continue_after_convergence*]
Description	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
Hint	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*.

convolution_step

Syntax	[*convolution_step* #]
Description	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.
Hint	*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*.

d_Is

Syntax	[*d_Is*]... [*d* E *I* E]...
Description	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.

d_spacing_to_energy_in_eV_for_f1_f11

Syntax [*d_spacing_to_energy_in_eV_for_f1_f11* !E]

Description 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;
```

degree_of_crystallinity

Syntax

Description 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.).

Example

```
xdd ...
      crystalline_area 0
      amorphous_area 0
      degree_of_crystallinity 0
      str...
            numerical_area 0
      hkl_I...
            numerical_area 0
      d_I...
            numerical_area 0
      str...
            numerical_area 0
            amorphous_phase
```

Hint 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.

default_I_attributes

Syntax [*default_I_attributes* E]

Description 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*.

do_errors

Syntax	[*do_errors*]
Description	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.

exclude

Syntax	[*exclude* #ex1 #ex2]...
Description	Excludes an x-axis region between #ex1 and #ex2.
Hint	The macro "Exclude" simplifies the use of *exclude*.

exp_conv_const

Syntax [*exp_conv_const* E [*exp_limit* E]]...

Description Defines ε_m in the convolution function:

$$\text{Exp}(\text{Ln}(0.001)\ \varepsilon / \varepsilon_m) \quad \text{for } \varepsilon = 0 \text{ to } exp\_limit$$

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.

extra_X_left, *extra_X_right*

Syntax	[*extra_X_left* !E] [*extra_X_right* !E]
Description	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.

file_name_for_best_solutions

Syntax [*file_name_for_best_solutions* $file]

Description 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	

fit_obj

Syntax	[*fit_obj* E [*min_X* !E] [*max_X* !E]]...
Description	*fit_obj*'s can be used to insert a user-defined function. *fit_obj*'s can be a function of X.
	[*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.

fourier_map

Syntax

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]]…

Description

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 E_h values and by the Structure Viewer window for picking atoms. For the tangent formula then relative quantities are important.

gauss_fwhm

Syntax	[*gauss_fwhm* E]...
Description	Defines the FWHM of a Gaussian function to be convoluted into phase peaks.
Hint	*gauss_fwhm* is for example used by the CS_G and Strain_G macros.

grs_interaction

Syntax	[*grs_interaction* [*from_N* #] [*to_N* #] [*no_self_interaction*] $site_1 $site_2 *qi* # *qj* # N E]...
Description	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.
Hint	*grs_interaction* is typically used for applying electrostatic constraints in inorganic materials. The GRS_Interaction macro simplifies the use of *grs_interaction*.

hat

Syntax	[*hat* E [*num_hats* #]]...
Description	Defines the X-axis size of an impulse function that is convoluted into phase peaks.
	[*num_hats*]: The number of hats to be convoluted.
Hint	*num_hats* is set to 1 by default.
	hat is used for example by the Slit_Width and Specimen_tilt macros.

hkl_Is

Syntax [*hkl_Is*]...
[*lp_search* !E]
[*I_parameter_names_have_hkl* $start_of_parameter_name]
[*hkl_m_d_th2* # # # # # # *I* E]...

Description 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 10.2.

[*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 Le Bail keyword is not defined then the intensity parameters are given the unique code of @ .

Example

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
          ...
         }
```

Hint

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*.

hkl_Is_from_hkl4

Syntax

[*xdd* $file.hkl]...
      [*hkl_Is_from_hkl4*]
            [*i_on_error_ratio_tolerance* #]
            [*num_highest_I_values_to_keep* #num]

Description

hkl_Is_from_hkl4 is used for generating a powder pattern from single crystal data in ShelX HKL4 format.

[*i_on_error_ratio_tolerance*]: Filters out hkl's that does 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.

hkl_plane $hkl

Syntax

[*hkl_plane* $hkl]...

Description

Used by the OpenGL viewer to display hkl planes. Here are some examples:

```
str…
   hkl_plane 1 1 1
   hkl_plane "2 -2 0"
```

iters

Syntax

[*iters* #]

Description

The maximum number of refinement iterations.

Hint

iters is set to 500000 by default

lam

Syntax	[*lam* [*ymin_on_ymax* #] [*no_th_dependence*] [*la* E *lo* E [*lh* E *lg* E]]...] [[*Lam* !E] [*calculate_Lam*]]
Description	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. Allows the use of non-X-ray data or fitting to negative 2θ values. [*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.
Hint	For more details about *lam* refer to section 5.

lebail

Syntax	[*lebail* #]
Description	A 1 for the *lebail* keyword flags the use of the Le Bail method for peak intensity extraction.

line_min, use_extrapolation, no_normal_equations, use_LU

Syntax	[*line_min*][*use_extrapolation*][*no_normal_equations*][*use_LU*][*approximate_A*]
Description	For a detailed description of the TOPAS minimization routines refer to section 4.

lor_fwhm

Syntax	[*lor_fwhm* E]...
Description	Defines the FWHM of a Lorentzian function that is convoluted into phase peaks.
Hint	*lor_fwhm* is for example used by the CS_L and Strain_L macros.

lpsd_th2_angular_range_degrees

Syntax	[*lpsd_th2_angular_range_degrees* E]... *lpsd_equitorial_divergence_degrees* E *lpsd_equitorial_sample_length_mm* E
Description	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.

marquardt_constant

Syntax	[*marquardt_constant* !E]
Description	Allows for changing the Marquardt constants.

min_r, *max_r*

Syntax	[*min_r* #] [*max_r* #]
Description	Defines the minimum and maximum radii for calculating bond lengths.
Hint	*min_r* and *max_r* are by default set to 0 and 3.2 Å respectively.

mixture_density_g_on_cm3

Syntax	[mixture_density_g_on_cm3 !E]
Description	Calculates the density of the mixture assuming a packing density of 1.

mixture_MAC

Syntax	[*mixture_MAC* !E]
Description	Calculates the mass absorption coefficient in cm^2/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*.

Example: 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.

Hint: Mass absorption coefficients obtained from NIST at http://physics.nist.gov/PhysRefData/XrayMassCoef are used to calculate *mixture_MAC* and *phase_MAC*.

neutron_data

Syntax: [*neutron_data*]

Description: 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)

Hint: 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));
```

no_LIMIT_warnings

Syntax: [*no_LIMIT_warnings*]

Description: Suppresses LIMIT_MIN and LIMIT_MAX warning messages.

normalize_FCs

Syntax	[*normalize_FCs*]
Description	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.

numerical_area

Syntax	[*numerical_area* E]
Description	Returns the area calculated numerically under the phase and is e.g. used for calculating *degree_of_crystallinity*.

occ_merge

Syntax [[*occ_merge* $sites] [*occ_merge_radius* !E]]...

Description *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.

omit_hkls

Syntax [*omit_hkls* !E]

Description 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;
```

one_on_x_conv

Syntax [*one_on_x_conv* E]...

Description Defines ε_m in the convolution function:

$$(4\ |\varepsilon_m\ \varepsilon|)^{-½} \qquad \text{for } \varepsilon = 0 \text{ to } \varepsilon_m$$

that is convoluted into phase peaks. ε_m can be greater than or less than zero.

Hint *one_on_x_conv* is used for example by the Divergence macro.

only_penalties

Syntax	[*only_penalties*]
Description	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

Syntax [*out_A_matrix* $file]
  [*A_matrix_prm_filter* $filter]

Description *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

Syntax [*out* $file [*append*]]...
  [*out_record*]
    [*out_eqn* !E]
    [*out_fmt* $c_fmt_string]
    [*out_fmt_err* $c_fmt_string]...

Description Used for writing parameter details to a file. The details are appended to $file when *append* is defined. *out_eqn* defines the equation or parameter to be written to $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 *eqn.*

Both *out_fmt* and *out_fmt_err* requires 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.

Example The following example illustrates the use of *out* using the Out macros.

```
xdd...
   out "sample output.txt" append
   str...
      CS_L(cs_l, 1000)
      Out_String("\tCrystallite Size Results:\n")
      Out_String("\t========================\n")
      Out(cs_l, "\tCrystallite Size (nm):\t%11.5f",
              "\tError in Crystallite Size:\t%11.5f\n")
```

Hint *out* provides a means of defining individual refinement result files.

out_rwp

Syntax	[*out_rwp* $file]
Description	Outputs a list of R_{wp} values encountered during refinement to the file $file.

out_prm_vals_per_iteration, *out_prm_vals_on_convergence*

Syntax	[*out_prm_vals_per_iteration* $file]... l [*out_prm_vals_on_convergence* $file]... [*out_prm_vals_filter* $filter]
Description	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 $fliter 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*.

peak_type

Syntax [*peak_type* $type]

Description Sets the peak type for a phase. The following peak types are available:

Peak type	**$type**	**Parameters**
Fundamental Parameters	fp	
PseudoVoigt	pv	[*pv_lor* E *pv_fwhm* E] *pv_lor*: Lorentzian fraction of the peak profile(s). *pv_fwhm*: FWHM of the peak profile(s).
Split-PearsonVII	spvii	[*h1* E *h2* E *m1* E *m2* E] *h1*, *h2*: The sum of *h1* and *h2* gives the full width at half maximum of the composite peak. *m1, m2*: 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] *spv_h1*, *spv_h2*: The sum of *spv_h1* and *spv_h2* gives the full width at half maximum of the composite peak. *spv_l1, spv_l2*: Lorentzian fractions of the left and right composite peak, respectively.

peak_buffer_step

Syntax [*peak_buffer_step* E [*report_on*]]

Description 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.

Hint The equation of *peak_buffer_step* is set to 500**Peak_Calculation_Step* by default.

penalty

Syntax	[*penalty* !E]...
Description	Defines a penalty function that can be a function of other parameters. Penalties are useful for stabilizing refinements as in for example their use in bond-length restraints.
Example	This example defines a bond length restraint using the GRS series (see *grs_interaction* below):

```
site C   x @ 0.2096  y @ 0.2798  z @ 0.5574  occ C 1  beq 3.35
site O1  x @ 0.0811  y @ 0.3515  z @ 0.5373  occ O 1  beq 3.35
site O2  x @ 0.2603  y @ 0.1558  z @ 0.6205  occ O 1  beq 3.35
site O3  x @ 0.3428  y @ 0.3689  z @ 0.5403  occ O 1  beq 3.35

Grs_Interaction(C, O1,  4, -2, !co1, 1.25, 5)  penalty = co1;
Grs_Interaction(C, O2,  4, -2, !co2, 1.25, 5)  penalty = co2;
Grs_Interaction(C, O3,  4, -2, !co3, 1.25, 5)  penalty = co3;
```

The next example applies a user-defined penalty function to lattice and crystallite size parameters, which are expected to be 5.41011 Å and 200 nm respectively:

```
str...
   Cubic(lp_ceo2 5.41011)
   penalty = (lp_ceo2-5.41011)^2;

   CS_L(cs_l, 200)
   penalty =(cs_l-200)^2;
```

Hint: TOPAS will minimize on penalty functions only when there are no observed data or when *only_penalties* is defined.

penalties_weighting_K1

Syntax	[*penalties_weighting_K1* !E]
Description	Defines the weighting K1 given to the penalty functions, see Eq. (4-2).
Hint	*penalties_weighting_K1* is set to 1 by default

percent_zeros_before_sparse_A

Syntax	[*percent_zeros_before_sparse_A* #]
Description	Defines the percentage of the A matrix than can be zero before sparse matrix methods are invoked. The default value is 60%.

phase_MAC

Syntax	*phase_MAC* !E
Description	Calculates the mass absorption coefficient in cm^2/g for the current phase. See description for *mixture_MAC*.

phase_name

Syntax	[*phase_name* $phase_name]
Description	The name given to a phase, used for reporting purposes.

phase_out

Syntax	[*phase_out* $file [*append*]]... [*out_record*] [*out_eqn* !E] [*out_fmt* $c_fmt_string] [*out_fmt_err* $c_fmt_string]...
Description	Used for writing phase hkl dependent details to file. See the keyword *out* for a description of *out_record*.
Hint	The Create_d_Ip_file macro is a good example of the use of *phase_out*.

phase_penalties

Syntax	[*phase_penalties* $sites N [*hkl_Re_Im* #h #k #l #Re #Im]...]… [*accumulate_phases_and_save_to_file* $file] [*accumulate_phases_when* !E]
Description	*phase_penalties* for a single hkl is defined as follows:

$$\mathrm{Pp}_{\mathrm{hkl}} = \begin{cases} 0, & \text{if the phase } \phi_{s,hkl}-45° < \phi_c < \phi_{s,hkl}+45° \\ d\, \mathrm{I}^2_{\mathrm{c,hkl}}(\phi_{s,hkl}-\phi_{c,hkl})^2, & \text{if the phase } \phi_c < \phi_{s,hkl}-45°, \text{or}, \phi_c > \phi_{s,hkl}+45° \end{cases}$$

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 particuar *penalty* equations. ϕ_c is calculated form 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.

pk_xo

Syntax	[*pk_xo* E]
Description	Provides a mechanism for transforming peak position to an x-axis position.
Example	The peak position for neutron time-of-flight data is typically calculated in time-of-flight space, tof, or: $tof = t0 + t1\ d_{hkl} + t2\ d_{hkl}^2$ where t0 and t1 and t2 are diffractometer constants. *pk_xo* can be used to refine TOF data.

process_times

Syntax	[*process_times*]
Description	On termination of refinement, process times are displayed.

prm, local

Syntax	[*prm* I *local* E [*min* !E] [*max* !E] [*del* !E] [*update* !E] [*stop_when* !E] [*val_on_continue* !E]]...
Description	Refer to section 2.

push_peak, bring_2nd_peak_to_top, add_pop_1st_2nd_peak, scale_top_peak

Syntax	[*push_peak*]... [*bring_2nd_peak_to_top*]... [*add_pop_1st_2nd_peak*]... [*scale_top_peak* E]…
Description	Provides for manipulation of the peak calculation stack; see section 5.3 for details.

quick_refine

Syntax [*quick_refine* !E]
        [*quick_refine_remove* !E]

Description

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_errros* 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 4.3 is used and parameters are not reinstated until the next refinement cycle.

Hint	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;
```

r_bragg

Syntax	[*r_bragg* #]
Description	Reports on the value for R-Bragg.
Hint	Note, R-Bragg is independent of hkl's and thus can be calculated for all phase types that contain phase peaks.

r_p, *r_p_dash*, *r_wp*, *r_wp_dash*, *r_exp*, *r_exp_dash*, *gof* *weighted_Durbin_Watson*

Syntax	[*r_p* #] [*r_p_dash* #] [*r_wp* #] [*r_wp_dash* #] [*r_exp* #] [*r_exp_dash* #] [*gof* #] [*weighted_Durbin_Watson* #]
Description	*xdd* dependent or global scope refinement indicators. Keywords ending in “_dash” correspond to background subtracted indicators.

rand_xyz

Syntax

Description: 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.

randomize_file_out_normal

Syntax	[*randomize_file_out_normal* $file]
Description	Instructs TOPAS to randomize the calculated pattern Y_c using a Normal distribution and writes Y_c to the file $file.

randomize_on_errors

Syntax	[*randomize_on_errors*]
Description	see Section 4.8
Hint	When val_on_continue is defined then the corresponding parameter is not randomized according to *randomize_on_errors*.

*rebin_with_dx*_of

Syntax	[*rebin_with_dx*_of !E]
Description	*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.

rigid

Syntax

```
[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]
```

Description

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* #displacement]: When *continue_after_convergence* is defined, *rand_xyz* processes are initiated after convergence. It introduces a random displacement to the translate fractional coordinates (*tx*, *ty*, *tz*) that are independent parameters. The size of the random displacement is given by the current *temperature* multiplied by #displacement where #displacement is in Å.
	[*start_values_from_site* $unique_site_name]: initializes the values *ta*, *tb*, *tc* with corresponding values taken from the site $unique_site_name.
Hint	See section 6.11 for a description of rigid bodies.

Rp, *Rs*

Syntax	[*Rp* #] [*Rs* #]
Description	Defines the primary and secondary radius of the diffractometer in mm.
Hint	The default for *Rp* and *Rs* is set to 217.5 mm.

scale

Syntax	[*scale* E]
Description	The Rietveld scale factor, can be applied to all phase types..

scale_pks

Syntax	[*scale_pks* E]...
Description	Used for applying intensity corrections to phase peaks.
Example	The following instruction defines a Lorentz-Polarisation correction:

```
scale_pks = (1+Cos(c Deg)^2 Cos(2 Th)^2)/(Sin(Th)^2 Cos(Th));
```

Hint	*scale_pks* is used for example in the LP_Factor, Preferred_Orientation and Absorption_With_Sample_Thickness_mm macros.

seed

Syntax	[*seed*]
Description	Initialises the random number generator with a different seed based on the computer clock.

site

Syntax [*site* $site_name [*x* E] [*y* E] [*z* E]]...
[*occ* $atom E [*beq* E]]...
[*num_posns* #] [*rand_xyz* !E] [*inter* !E #]

Description 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.

[*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.

[*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!]: When *continue_after_convergence* is defined, *rand_xyz* processes are initiated after convergence. It introduces a random displacement to the site fractional coordinates (*x*, *y*, *z*) that are independent parameters. The size of the random displacement in Å is given by the current *temperature* multiplied by the value returned by *rand_xyz.*

[*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.

Example 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
```

sites_distance, sites_angle, sites_flatten

Syntax [*sites_distance* N][*sites_angle* N][*sites_flatten* N [*sites_flatten_tol* !E]]...
[*site_to_restrain* $site [#ep [#n1 #n2 #n3]]]...

Description 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 follows:

$$sites\_flatten = \frac{6}{n(n-1)(n-2)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} (\left|b_i \times b_j \cdot b_k\right| - \text{tol})^2 \text{ , if } \left|b_i \times b_j \cdot b_k\right| > \text{tol}$$

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.

#eq, #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 constraints 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
       O1:0   0  0  0  1.52473 112.923 112.923
       O2:0   0  0  0  1.59001 105.749 105.749 105.749
```

Example constraints 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)
```

Hint — Note, for more than around 6 sites then *sites_flatten* becomes computationally expensive.

sites_geometry

Syntax — [*sites_geometry* N]...
[*site_to_restrain* $site [#ep [#n1 #n2 #n3]]]...

Description — *sites_geometry* defines a grouping of up to four sites; $Name is the name given to ths 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

Syntax — [*siv_s1_s2* # #]

Description — Defines the S_1 and S_2 integration limits for the spherical interaction volume of the GRS series.

smooth

Syntax — [smooth #num_pts_left_right]

Description — Performs a Savitzky-Golay smoothing of the observed data. The smoothing encompasses (2 * #num_pts_left_right + 1) points.

spherical_harmonics_hkl

Syntax	[*spherical_harmonics_hkl* $name]... [*sh_Cij_prm* $Yij E]... [*sh_alpha* !E] [*sh_order* #]
Description	Defines a hkl dependent symmetrized spherical harmonics series (see section 6.10.1). $name defines the name given to the series and when used in equations it returns the value of the spherical harmonics series. [*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.
Hint	The PO_Spherical_Harmonics macro simplifies the use of *spherical_harmonics_hkl*.

stacked_hats_conv

Syntax	[*stacked_hats_conv*]... [*whole_hat* E [*hat_height* E]]... [*half_hat* E [*hat_height* E]]...
Description	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.
Hint	*stacked_hats* is used for example to describe tube tails using the Tube_Tails macro.

start_X, finish_X

Syntax	[*start_X* #] [*finish_X* #]
Description	Defines the start and finish X region to fit to.

str

Syntax	[*str*]... *space_group* $symbol
Description	Defines the start of structure information. [*space_group* $symbol]: $symbol can be any space group symbol occurring in the ICSD, it can also be a space group number.
Hint	The macro "STR" simplifies the use of *str*.

str_hkl_angle

Syntax	[*str_hkl_angle* N h k l]...
Description	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.
Hint	*str_hkl_angle* is used for example in the Preferred_Orientation macro.

swap_sites

Syntax [*swap_sites* $sites_1 $sites_2]...

Description *Swap_sites* attempts to find a lower χ^2 by swapping sites defined in $sites_1 with those defined in $sites_2. The swapping continues until all swaps are performed. Outline of the algorithm:

```
While swap_site processes still to be processed {
        k = 0
        Do
                Randomize the cation configuration according to swap_sites.
                Find the local minimum of $\chi^2$
                k = k + 1
                If $\chi_1^2$ (k+1) > $\chi_1^2$ (k) then reset site positions
        While $\chi_1^2$ (k+1) > $\chi_1^2$ (k) or all swap possibility exhausted
}
```

Starting from a particular cation configuration, a single swap of two different cation species does not exhaust all possible cation configurations. For example, three sites A, B and C have the six possible configurations of ABC, ACB, BAC, CAB, BCA, CBA. Swapping any two from the starting configuration of ABC results in the possible configurations of BAC, CBA, ACB with CAB and BCA absent. The absent configurations can be obtained by a double swap where, for example, the swapping of A and C yields CBA and then the swapping of A and B yields CAB. Thus all possible configurations can be visited if in between swapping steps are realized by the simulated annealing process.

Swapping of two sites can be performed as follows:

```
swap_sites Ca Zr
```

If sites are e.g. named Ca1, Ca2, Ca3, Zr1, Zr2, Zr3 then the following can be used:

```
swap_sites Ca* Zr*
```

temperature

Syntax [*temperature* !E]...
    [*move_to_the_next_temperature_regardless_of_the_change_in_rwp*]
    [*save_values_as_best_after_randomization*]
    [*use_best_values*]
    [*do_processes*]

Description 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_1^2 increases relative to a previous cycle then the temperature is advanced to the next temperature. If χ_1^2 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”.

[*do_processes*]: Executes any *swap_sites* or *try_site_patterns* processes.

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_1^2 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
```

Hint The macro "Temperature_Regime" simplifies the use of *temperature*. The temperature regime as defined in the macro Auto_T is sufficient for most problems.

th2_offset

Syntax	[*th2_offset* E]...
Description	Used for applying 2θ corrections to phase peaks.
Example	The following instruction applies a sample displacement correction:

```
th2_offset = -2 Rad (c) Cos(Th) / Rs;
```

Hint	*th2_offset* is used for example in the Zero_Error and Specimen_Displacement macros.

try_site_patterns

Syntax	[*try_site_patterns* $sites [*num_patterns_at_a_time* #]]...
Description	*try_site_patterns* performs an exhaustive search of the possible cation configurations. *num_patterns_at_a_time* defines the number of patterns to process at any one time. Outline of the algorithm:

```
While try_site_patterns processes still to be processed {
        k = 0
        Do
                Change the cation configuration according to try_site_patterns.
                Find the local minimum of $\chi^2$
                k = k + 1
                If $\chi_1^2$ (k+1) > $\chi_1^2$ (k) then reset site positions
        While $\chi_1^2$ (k+1) > $\chi_1^2$ (k) or all swap possibility exhausted
}
```

The number of possible cation configurations determines the approximate magnitude of a structure determination problem. A structure consisting of N_A cation sites of species A and N_B cation sites of species B has, for a particular set of cation positions, a number of possible configurations N_{AB} calculated as follows:

$$N_{AB} = {}^{(N_A+N_B)}C_{N_B} \,/\, N_B!$$

where the notation ${}^{u}C_{v}$=(u-0)(u-1)(u-2)…(u-(v-1)) has been used. Thus for N_A=3 and N_B=4 we have N_{AB}=(7 6 5 4)/(4 3 2 1)=N_{BA}=(7 6 5)/(3 2 1)=35. The number of configurations N_{ABD} for and additional set of cation sites of species D becomes:

$$N_{ABD} = N_{AB} \left({}^{(N_A+N_B+ND_D)}C_{N_D} \,/\, N_D! \right)$$

An additional two D sites, or, N_A=3, N_B=4 and N_D=2 gives and N_{ABD}=1260. Thus the number of configurations quickly becomes prohibitive for an exhaustive search.

Here is an example of using *try_site_patterns* on three Ca sites and two Zr sites:

```
str...
      site Ca...
      site Ca...
      site Ca...
      site Zr...
      site Zr...
      try_site_patterns "Ca Zr" num_patterns_at_a_time 3
```

N_{CaZr}=10 and thus *try_site_patterns* will cycle through the 10 patterns searching for a reduction in χ^2.

use_tube_dispersion_coefficients

Syntax	[*use_tube_dispersion_coefficients*]
Description	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.

user_defined_convolution

Syntax	[*user_defined_convolution* E *min* E *max* E]...
Description	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.
Example	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
```

verbose

Syntax	[*verbose* #]...
Description	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.
Hint	The Simulated_Annealing_1 macro has *verbose* with a value of -1; this ensures that lengthly simulated annealing runs do not exhaust memory due to saving R_{WP} values and text output buffers.

view_structure

Syntax	*[view_structure]*
Description	Displays the crystal structure in the Structure Viewer window.

weight_percent_amorphous

Syntax	[*weight_percent_amorphous* !E]
Description	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.

weighting

Syntax [*weighting* !E [*recal_weighting_on_iter*]]

Description Used for calculating the *xdd* dependent weighting function in χ^2_1. 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.

x_calculation_step

Syntax [*x_calculation_step* !E]

Description 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* is as follows:

```
x_calculation_step .01
x_calculation_step = .02 (1 + Tan(Th));
x_calculation_step = Yobs_dx_at(Xo);
```

xdd

Syntax	[*xdd* $file [{ $data }] [*range* #] [*xye_format*] [*gsas_format*] [*fullprof_format*]]...
Description	Defines the start of *xdd* dependent keywords and the file containing the observed data. [{$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.

Example: The following instruction will refine on the first range in the data file pbso4.raw:

```
xdd pbso4.raw
```

To 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   ...
}
```

Hint: The macro "XDD" simplifies the use of *xdd*.

Supported file formats are described in section 11.

xdd_out

Syntax: [*xdd_out* $file [*append*]]...
  [*out_record*]
    [*out_eqn* !E]
    [*out_fmt* $c_fmt_string]
    [*out_fmt_err* $c_fmt_string]...

Description	Used for writing *xdd* dependent details to file. The *out_eqn* can contain the reserved parameter names of X, Yobs, Ycalc and SigmaYobs. See the keyword *out* for a description of *out_record*.
Hint	The Out_Yobs_Ycalc_and_Difference macro is a good example using *xdd_out*.

xdd_scr

Syntax

[*xdd_scr* $file] …
  [*dont_merge_equivalent_reflections*]
  [*dont_merge_Friedel_pairs*]
  [*ignore_differences_in_Friedel_pairs*]
  [*str*]
    [*auto_scale* !E]
    [*i_on_error_ratio_tolerance* #]
    [*num_highest_I_values_to_keep* #num]

Description

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 6.3.3.

[*dont_merge_Friedel_pairs*]: Prevents the merging of Friedel pairs, see also section 6.3.3.

[*ignore_differences_in_Friedel_pairs*]: Forces the use of equation (6-12) for calculating F^2, see also section 6.3.3.

[*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.

[*i_on_error_ratio_tolerance*]: Filters out hkl's that does 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.

Example

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
   i_on_error_ratio_tolerance 4
   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
   ...
```

Hint

The SCR format is white space delimited and consists of entries of h, k, l, m, d, 2θ, Fo^2 which is the format outputted by the Create_hklm_d_Th2_Ip_file macro.

xo_Is

Syntax

[*xo_Is*]...
  [*xo* E *I* E]...

Description

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.

yobs_eqn

Syntax	[*yobs_eqn* !N E]
Description	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.

yobs_out, ycalc_out, diff_out

Syntax	[*yobs_out* $file] [*ycalc_out* $file] [*diff_out* $file]
Description	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

Syntax [*yobs_to_xo_posn_yobs* !E]

Description 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 simulated annealing 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.

7.3 Keywords for simplified user input

7.3.1 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.

7.3.2 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.

7.3.3 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 the xdd's
      th2_offset ...
      ...
      str...                        first str
            scale ...               str dependent keywords, not common to the str's
            ...
xdd ...                             second xdd
      bkg ...                       xdd dependent keywords, not common to the xdd's
      th2_offset ...
      ...
      str...                        second str
            scale ...               str dependent keywords, not common to the str's
            ...
for xdds {
      Slit_Width(.2)                xdd dependent keywords, common to the xdd's
      CuKa2
      ...
      for strs {                    str dependent keywords, common to the xdd's

            ...

      }

      for strs 1 to 1 {

            space_group p1          str dependent keywords, specific to the first str
            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.

8 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

8.1 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 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.

8.1.1 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.

8.1.2 Directives invoked on macro expansion

#m_ifarg, #m_else, #m_endif 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 8-1.

Table 8-1 : #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 8-2.

Table 8-2 : 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.

8.2 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
```

In Table 8-3 an overview of the standard macros used by TOPAS is provided.

Table 8-3: Predefined TOPAS macros.

Macro
***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

No_Th_Dependence
CuKa1(yminymax)
CuK1sharp(yminymax)
CuKa2_analyt(yminymax)
CuKa2(yminymax)
CuKa4_Holzer(yminymax)

Macro

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)

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)
MVW(m_v, v_v, w_v)

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)

Macro
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)

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)

Macro
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)
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)

8.3 Detailed description of macros

8.3.1 xdd file input macros

DAT, RAW, XDD, XY, XYE, SCR, SHELX_HKL4, TOF_XYE, TOF_GSAS

Syntax	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)

Description	Import measured data in different file formats, see also section 11. [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.

8.3.2 Lattice parameters

Cubic, Tetragonal, Hexagonal, Rhombohedral

Syntax	Cubic(a_cv)
	Tetragonal(a_cv, c_cv)
	Hexagonal(a_cv, c_cv)
	Rhombohedral(a_cv, al_cv)

Description	Simplifies the definition of lattice parameters. [a_cv]: Lattice parameter a. [c_cv]: Lattice parameter c. [al_cv]: Lattice parameter alpha.

8.3.3 Emission profile macros

No_Th_Dependence

Syntax	No_Th_Dependence
Description	Defines an emission profile that is 2θ independent. Allows the use of non-X-ray data or fitting to negative 2θ values.

CuKa1, CuK1sharp, CuKa2_analyt, CuKa2, CuKa4_Holzer, CuKa5, CuKa5_Berger, CuKb4_Holzer

CoKa3, CoKa7_Holzer, CoKb6_Holzer

CrKa7_Holzer, CrKb5_Holzer

FeKa7_Holzer, FeKb4_Holzer

MnKa7_Holzer, MnKb5_Holzer

NiKa5_Holzer, NiKb4_Holzer

MoKa2

Syntax	CuKa1(yminymax) and so forth
Description	Defines a source emission profile. [yminymax]: Determines the x-axis extent to which an emission profile line is calculated.

8.3.4 Instrument convolutions

Radius

Syntax	Radius(rp, rs)
Description	Instrument radius. [rp, rs]: Primary and secondary instrument radii in [mm]. For most diffractometers rp equals rs.

Specimen_Tilt

Syntax	Specimen_Tilt(c, v)
Description	Specimen tilt.
	[c, v]: Parameter name, Specimen tilt in [mm].

Slit_Width, SW

Syntax	Slit_Width(c, v), SW(c, v)
Description	Aperture of the detector (= receiving) slit.
	[c, v]: Parameter name, detector slit aperture in [mm].

Divergence

Syntax	Divergence(c, v)
Description	Horizontal divergence of the beam for fixed slits.
	[c, v]: Parameter name, beam divergence in [°].

Variable_Divergence

Syntax	Variable_Divergence(c, v)
Description	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

Syntax	Variable_Divergence_Shape(c, v)
Description	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

Syntax	Variable_Divergence_Intensity
Description	Constant illuminated sample length for variable slits (i.e. variable beam divergence). This macro corrects intensity deviations inherent to variable slits.

Simple_Axial_Model

Syntax	Simple_Axial_Model(c, v)
Description	Simple model for describing peak asymmetry due to axial divergence of the beam. [c, v]: Parameter name, receiving slit length [mm].

Full_Axial_Model, Sollers

Syntax	Full_Axial_Model(filament_cv, sample_cv, detector_cv, psol_cv, ssol_cv)
Description	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

Syntax	Finger_et_al(s2, h2)
Description	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

Syntax	Tube_Tails(source_width_c, source_width_v, z1_c, z1_v, z2_c, z2_v, z1_z2_h_c, z1_z2_h_v)
Description	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

Syntax	UVW(u, uv, v, vv, w, wv)
Description	Cagliotti relation (Cagliotti et al., 1958). [u, v, w]: Parameter names. [uv, vv, wv]: Halfwidth parameters.

8.3.5 Phase peak_type's

PV_Peak_Type, PVII_Peak_Type, TCHZ_Peak_Type

Syntax	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)
Description	Pseudo-Voigt, TCHZ pseudo-Voigt and PearsonVII functions. For the definition of the functions and function parameters refer to section 5.2.

8.3.6 Quantitative analysis

Apply_Brindley_Spherical_R_PD

Syntax	Apply_Brindley_Spherical_R_PD(R, PD)
Description	Applies the Brindley correction for quantitative analysis (Brindley, 1945). [R, PD]: Radius of the particle in [cm], packing density.

MVW

Syntax	MVW(m_v, v_v, w_v)
Description	Returns cell mass, cell volume, and relative phase amount. [m_v, v_v, w_v]: Mass, volume, and weight parameters.

8.3.7 2θ corrections

Cylindrical_2Th_Correction

Syntax	Cylindrical_2Th_Correction(μR)
Description	Applies a 2θ 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.

Zero_Error, ZE

Syntax	Zero_Error(c, v), ZE(c, v)
Description	Zero point error. [c, v]: Parameter name, zero point error in [° 2θ].

Specimen_Displacement, SD

Syntax	Specimen_Displacement(c, v), SD(c, v)
Description	Specimen displacement error. [c, v]: Parameter name, sample displacement in [mm].

8.3.8 Intensity corrections

Cylindrical_I_Correction

Syntax	Cylindrical_I_Correction(μR)
Description	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,

Syntax	Lorentz_Factor LP_Factor(c, v)
Description	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, LP_Factor_Synchrotron_Simple

Syntax	LP_Factor_Synchrotron(pp, ppv, mono, monov) LP_Factor_Synchrotron_Simple
Description	Synchrotron-specific Lorentz-Polarisation factors, for details see section 6.4. 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, PO

Syntax	Preferred_Orientation(c, v, ang, hkl), PO(c, v, ang, hkl)
Description	Preferred orientation correction based on March (1932). [c, v]: Parameter name, March parameter value. [ang, hkl]: Parameter name, lattice plane.

PO_Two_Directions

Syntax	PO_Two_Directions(c1, v1, ang1, hkl1, c2, v2, ang2, hkl2, w1c, w1v)
Description	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

Syntax	PO_Spherical_Harmonics(sh, order)
Description	Preferred orientation correction based on spherical harmonics according to Järvinen (1993). [sh, order]: Parameter name, spherical harmonics order.

Surface_Roughness_Pitschke_et_al, Surface_Roughness_Suortti

Syntax	Surface_Roughness_Pitschke_et_al(a1c, a1v, a2c, a2v)
	Surface_Roughness_Suortti(a1c, a1v, a2c, a2v)

Description	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

8.3.9 Bondlength penalty functions

Anti_Bump, AI_Anti_Bump, AI_Anti_Bump

Syntax	Anti_Bump(ton, s1, s2, ro, wby)
	AI_Anti_Bump(s1, s2, ro, wby, num_cycle_iters)
	AI_Anti_Bump(s1, s2, ro, wby)

Description	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, see also section 2.8.1) is smaller than num_cycle_iters For more details refer to *box_interaction* and *ai_anti_bump*.

Parabola_N

Syntax	Parabola_N(n1, n2, s1, s2, ro, wby)
Description	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

Syntax	Grs_Interaction(s1, s2, wqi, wqj, c, ro, n)
Description	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

Syntax	Grs_No_Repulsion(s1, s2, wqi, wqj, c)
Description	Used for calculating the Madelung constants. [s1, s2]: Sites. [wqi, wqj]: Valence charge of the atoms. [c]: Name of the GRS.

Grs_BornMayer

Syntax	Grs_BornMayer(s1, s2, wqi, wqj, c, ro, b)
Description	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, Angle_Restrain, Flatten, Distance_Restrain_Keep_Within, Distance_Restrain_Keep_Out

Syntax	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)

Description	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 2.8.1) is smaller than num_cycle_iters

Keep_Atom_Within_Box

Syntax	Keep_Atom_Within_Box(size)
Description	Applies constraints such that the present site cannot more outside of a box with a length of 2*size.

8.3.10 Reporting macros

Create_2Th_Ip_file

Syntax	Create_2Th_Ip_file(file)
Description	Creates a file with positions (2θ) and intensities.
	[file]: Filename. Can include drive and path.

Create_d_Ip_file

Syntax	Create_d_Ip_file(file)
Description	Creates a file with positions (d) and intensities.
	[file]: Filename. Can include drive and path.

Create_hklm_d_Th2_Ip_file

Syntax	Create_hklm_d_Th2_Ip_file(file)
Description	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

Syntax	Out_Yobs_Ycalc_and_Difference(file)
Description	Outputs the x-axis, Yobs, Ycalc and difference.
	[file]: Filename. Can include drive and path.

Out_X_Yobs, Out_X_Ycalc, Out_X_Difference

Syntax	Out_X_Yobs(file), Out_X_Ycalc(file), Out_X_Difference(file)
Description	Outputs the x-axis, Yobs, Ycalc and difference to files.
	[file]: Filename. Can include drive and path.

Out_F2_Details, Out_A01_A11_B01_B11

Syntax	Out_F2_Details(file)
	Out_A01_A11_B01_B11(file)

Description	Outputs structure factor details, see section 6.3.2. [file]: Filename. Can include drive and path.

Out_FCF

Syntax	Out_FCF(file)
Description	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

Syntax	Out_CIF_STR(file)
Description	Outputs structure details in CIF format. [file]: Filename. Can include drive and path.

8.3.11 Rigid body macros

Set_Length

Syntax	Set_Length(s0, s1, r, xc, yc, zc, cva, cvb)
Description	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

Syntax	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)
Description	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

Syntax	Triangle(s1, s2, s3, r)
	Triangle(s0, s1, s2, s3, r)
	Triangle(s0, s1, s2, s3, r, xc, yc, zc, cva, cvb, cvc)

Description	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

Syntax	Tetrahedra(s0, s1, s2, s3, s4, r, xc, yc, zc, cva, cvb, cvc)
Description	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

Syntax	Octahedra(s0, s1, s2, s3, s4, s5, s6, r) Octahedra(s0, s1, s2, s3, s4, s5, s6, r, xc, yc, zc, cva, cvb, cvc)
Description	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 Hexagon_sitting_on_side_in_xy_plane

Syntax	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)

Description	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

Syntax	Point_for_site(site, cart_x, cart_y, cart_z)
Description	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

Syntax	Translate(acv, bcv, ccv) Translate(acv, bcv, ccv, ops)
Description	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

Syntax	Translate_with_site_start_values(s0, xc, yc, zc)
Description	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

Syntax	Rotate_about_points(cv, a, b) Rotate_about_points(cv, a, b, pts)
Description	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

Syntax	Rotate_about_these_points(cv, a, b, ops)
Description	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

Syntax	Rotate_about_axies(cva, cvb)
	Rotate_about_axies(cva, cvb, cvc)

Description	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

Syntax	Rotation_vector_from_points(a, b)
Description	Determines a rotation vector form two points. [a, b]: Rotation vector defined by the sites a and b.

8.3.12 Background functions using fit_objects

One_on_X

Syntax	One_on_X(c, v)
Description	1/X background function ideal to describe background intensity at low angles due to air scattering [c, v]: Parameter name and value.

Bkg_Diffuse

Syntax	Bkg_Diffuse(b, bv, bb, bbv)
Description	Defines a function to describe short range order effects. [b, bv]: Parameter name, refineable weight. [bb, bbv]: Parameter name, correlation shell radii.

PV

Syntax	PV(a, xo, fwhm, g) PV(a, xo, fwhm, g, av, xov, fwhmv, gv)
Description	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

Syntax	PV_Left_Right(a, xo, fwhm1, fwhm2, g) PV_Left_Right(a, xo, fwhm1, fwhm2, g, av, xov, fwhm1v, fwhm2v, gv)
Description	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.

8.3.13 Sample convolutions

Sample_Thickness

Syntax	Sample_Thickness(dc, dv)
Description	Describes the sample thickness in the direction of the scattering vector. [dc, dv]: Parameter name, sample thickness in [mm].

Absorption, AB

Syntax	Absorption(c, v), AB(c, v)
Description	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

Syntax	Absorption_With_Sample_Thickness_mm_Shape(u, uv, d, dv)
Description	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

Syntax	Absorption_With_Sample_Thickness_mm_Shape_Intensity(u, uv, d, dv)
Description	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, Crystallite_Size, CS

Syntax	CS_L(c, v), Crystallite_Size(c, v), CS(c, v)
Description	Applies a Lorentzian convolution with a FWHM that varies according to the relation lor_fwhm = c / Cos(Th). [c, v]: Parameter name, crystallite size in [nm].

CS_G

Syntax	CS_G(c, v)
Description	Applies a Gaussian convolution with a FWHM that varies according to the relation gauss_fwhm = c / Cos(Th). [c, v]: Parameter name, crystallite size in [nm].

Strain_L, Microstrain, MS

Syntax	Strain_L(c, v), Microstrain_L(c, v) MS(c, v)
Description	Applies a Lorentzian convolution with a FWHM that varies according to the relation lor_fwhm = c Tan(Th). [c, v]: Parameter name, strain.

Strain_G

Syntax	Strain_G(c, v)
Description	Applies a Gaussian convolution with a FWHM that varies according to the relation gauss_fwhm = c Tan(Th). [c, v]: Parameter name, strain.

LVol_FWHM_CS_G_L

Syntax	LVol_FWHM_CS_G_L(k, lvol, kf, lvolf, csgc, csgv, cslc, cslv)
Description	Calculates FWHM and IB (integral breadth) based volume-weighted column heights (LVol). For details refer to section 6.2. [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. [cslc, cslv]: Parameter name, Lorentzian component.

8.3.14 Neutron TOF

TOF_LAM

Syntax	TOF_LAM(w_ymin_on_ymax)
Description	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)

Syntax	TOF_x_axis_calibration(t0, t0v, t1, t1v, t2, t2v)
Description	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)

Syntax	TOF_Exponential(a0, a0v, a1, a1v, wexp, t1, lr)
Description	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, TOF_CS_G

Syntax	TOF_CS_L(c, v, t1), TOF_CS_G(c, v, t1)
Description	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

Syntax	TOF_PV(fwhm, fwhmv, lor, lorv, t1)
Description	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.

8.3.15 Miscellaneous

Auto_T

Syntax	Auto_T(t)
Description	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 4.8. [t]: Temperature. See the *temperature* keyword.

Temperature_Regime

Syntax	Temperature_Regime
Description	Defines a temperature regime and is defined as { load temperature }. See the *temperature* keyword as well as section 4.8.

STR

Syntax	STR(sg)
Description	Signals the start of structure information. [sg]: Space group symbol.

Exclude

Syntax	Exclude
Description	Defines excluded regions. See the *exclude* keyword.

ADPs_Keep_PD

Syntax	ADPs_Keep_PD
Description	Keeps the anisotropic temperature parameters matrix *unn* positive definite, see the *adps* keyword.

Decompose

Syntax	Decompose(diff_toll)
Description	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, Phase_LAC_1_on_cm and Phase_Density_g_on_cm3

Syntax	Mixture_LAC_1_on_cm (mlac), Phase_LAC_1_on_cm (u), Phase_Density_g_on_cm3 (pd)
Description	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/cm^3]

Gauss, Lorentzian

Syntax	Gauss(xo, fwhm), Lorentzian(xo, fwhm)
Description	Unit area Gaussian or Lorentzian functions. [xo]: Position [fwhm]: FWHM

9 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).

9.1 Charge Flipping items and keywords

Charge Flipping keywords are listed in Table 9-1. Equations appearing in Charge Flipping keywords can be functions of the items shown in Table 9-2. 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 9-1: Keywords that can be used in Charge Flipping.

Keyword	**Default**
Tcharge_flipping	
General	
a !E *b* !E *c* !E [*al* !E] [*be* !E] [*ga* !E]	*al* = *be* = *ga* = 90
[*cf_hkl_file* $file]	
[*cf_in_A_matrix* $file]	
[*scale_Aij* !E]	Get(Aij)^2
[*break_cycle_if_true* !E]	
[*delete_observed_reflections* !E]	
[*extend_calculated_sphere_to* !E]	
[*f_atom_type* $type *f_atom_quantity* !E]…	
[*find_origin* !E]	1
[*fraction_density_to_flip* !E]	0.75
[*fraction_reflections_weak* !E]	0
[*min_d* !E]	0
[*min_grid_spacing* !E]	
[*neutron_data*]	
[*space_group* $]	P1
[*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]	1
[*hm_covalent_fwhm* !E]	1

Keyword	Default
[*pick_atoms* $atoms]...	
[*activate* !E]	1
[*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]	0
[*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]	30
[*tangent_min_triplets_per_h* !E]	1
[*tangent_scale_difference_by* !E]	1
Miscellaneous	
[*apply_exp_scale* !E]	1
[*correct_for_atomic_scattering_factors* !E]	1
[*correct_for_temperature_effects* !E]	1
[*hkl_plane* $hkl]…	
[*randomize_initial_phases_by* !E]	Rand(-180,180)
[*scale_E* !E]	1
[*scale_F* !E]	1
[*scale_F000* !E]	0
[*scale_weak_reflections* !E]	
[*user_threshold* !E]	
[*verbose* #]	1
GUI related	
[*add_to_cloud_N* !E [*add_to_cloud_when* !E]]	
[*pick_atoms_when* !E]	
[*view_cloud* !E]	1

Table 9-2: Items that can be used in Charge Flipping equations.

Keyword	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	
Macros	
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*.	

9.2 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.

9.2.1 Perturbations

Perturbations can be categorized as being of either phase, structure factor intensity or electron density perturbations as shown in Table 9-1. 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:

$$\rho = \begin{cases} -\rho, & \text{for } \rho < \delta \\ \rho & \text{for } \rho \geq \delta \end{cases} \tag{9-1}$$

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*.

9.2.2 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.

9.2.3 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-spcaing edge of are often in error to a large extent; it is therefore best to remove these reflections using *delete_observed_reflections*, for example example:

```
delete_observed_reflections = D_spacing < 1.134;
```

9.2.4 The algorithm of Oszlányi & Süto (2005) and F000

Normalized structure factors enhances the chances of finding a solution (Oszlányi & Süto, 2006) 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.

9.3 Alphabetical description of keywords

add_to_cloud_N

Syntax	[*add_to_cloud_N* !E [*add_to_cloud_when* !E]]
Description	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.

add_to_phases_of_weak_reflections

Syntax	[*add_to_phases_of_weak_reflections* !E]
Description	Allows for modification to phases of weak reflections. For example, to add π/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*.

apply_exp_scale

Syntax	[*apply_exp_scale* !E]
Description	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.

cf_hkl_file

Syntax	[*cf_hkl_file* $file]
Description	Defines the input hkl file.

cf_in_A_matrix

Syntax	[*cf_in_A_matrix* $file [scale_Aij !E]]
Description	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.

break_cycle_if_true

Syntax	[*break_cycle_if_true* !E]
Description	Interrupts charge flipping to execute *randomize_phases_on_new_cycle_by*. Cycle_Iter is set to zero and Cycle is incremented.

correct_for_atomic_scattering_factors

Syntax	[*correct_for_atomic_scattering_factors* !E]
Description	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

Syntax	[*correct_for_temperature_effects* !E]
Description	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*.

delete_observed_reflections

Syntax	[*delete_observed_reflections* !E]
Description	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

Syntax [*extend_calculated_sphere_to* !E]

Description 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

Syntax [*f_atom_type* $type f_atom_quantity !E]…

Description 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

Syntax [*find_origin* !E]

Description 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.

flip_equation

Syntax [*flip_equation* !E]

Description Allows for a user defined flip; here’s an example:

```
flip_equation =
If(Get(density) < Get(threshold), -Get(density), Get(density));
```

flip_regime_2

Syntax [*flip_regime_2* !E]

Description The electron density is modified according to equation (9-1) and then further modified using:

$$\rho = \rho - \text{Get(flip\_regime\_2)}\, \rho^3 \big/ \rho_{\max}^2$$

flip_regime_2 is typically ramped from 1 to 0.

flip_regime_3

Syntax	[*flip_regime_3* !E]
Description	The electron density is modified according to equation (9-1) and then further modified using:

$$\rho = \begin{cases} \rho, & \text{for } \rho < \delta \\ \rho = \text{Min}(\rho, \rho_{\text{max}} \text{ Get(flip\_regime\_3)}) & \text{for } \rho \geq \delta \end{cases}$$

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.

fraction_density_to_flip

Syntax	[fraction_density_to_flip !E]
Description	The amount of charges flipped is fractionally based. A value of 0.6, for example, sets the threshold δ such that the sign of the lowest 60% of charge is changed. Get(threshold) can be used to retrieve δ.

fraction_reflections_weak

Syntax	[fraction_reflections_weak !E]
Description	Defines the fraction of observed reflections to flag as “weak”. 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.

histogram_match_scale_fwhm

Syntax	[*histogram_match_scale_fwhm* !E] [*hm_size_limit_in_fwhm* !E] [*hm_covalent_fwhm* !E]
Description	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.

min_d

Syntax	[*min_d* !E]
Description	Determines in Angstroms the resolution of observed reflections to work with; only observed reflections with a d-spacing above min_d are considered. *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

Syntax	[*min_grid_spacing* !E]
Description	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

Syntax	[*neutron_data*]
Description	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.

pick_atoms

Syntax	[*pick_atoms* $atoms]... [*activate* !E] [*choose_from* !E] [*choose_to* !E] [*choose_randomly* !E] [*omit* !E] [*displace* !E] [*insert* !E]

Description

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 attempts 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 thee 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 descreases 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. Additionaly weak reflections are also given more weighting.

pick_atoms_when

Syntax	[*pick_atoms_when* !E]
Description	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.

randomize_initial_phases_by

Syntax	[*randomize_initial_phases_by* !E]
Description	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)
```

randomize_phases_on_new_cycle_by

Syntax	[*randomize_phases_on_new_cycle_by* !E]
Description	Example:

```
randomize_phases_on_new_cycle_by = Rand(-180, 180);
```

scale_density_below_threshold

Syntax	[*scale_density_below_threshold* !E]
Description	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).

scale_E

Syntax [*scale_E* !E]

Description 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.

scale_F

Syntax [*scale_F* !E]

Description 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));
```

scale_F000

Syntax [*scale_F000* !E]

Description 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

Syntax [*scale_weak_reflections* !E]

Description 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.

set_initial_phases_to

Syntax [*set_initial_phases_to* $file]
 [*modify_initial_phases* !E]

Description 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).

space_group

Syntax	[*space_group* $]
Description	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.

symmetry_obey_0_to_1

Syntax	[*symmetry_obey_0_to_1* !E]
Description	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%.

tangent_num_h_read

Syntax [*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]

Description *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 minium number of triplets per reflection h. 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:

$$\phi_{h,new} = \phi_{h,cf} + S\,\alpha_h\left(\phi_{h,tf} - \phi_{h,cf}\right)$$

$$\tan(\phi_{h,tf}) = T_h / B_h$$

$$T_h = \sum_k E_h E_k E_{h-k}\ \sin\left(\phi_k + \phi_{h-k}\right)$$

$$B_h = \sum_k E_h E_k E_{h-k}\ \cos\left(\phi_k + \phi_{h-k}\right)$$

$$\alpha_h = M_h / M_{h,\max}\ ,\ M_h = \sqrt{T_h^2 + B_h^2}$$

user_threshold

Syntax	[*user_threshold* !E]
Description	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.

use_Fc

Syntax	[use_Fc]
Description	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.

verbose

Syntax	[*verbose* #]
Description	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.

view_cloud

Syntax	[*view_cloud* !E]
Description	Displays the electron densityin the Structure Viewer window. Here are some examples:

```
view_cloud 1 ' Update cloud every charge-flipping iteration
view_cloud = Mod(Cycle_Iter, 10) == 0;
```

10 INDEXING

10.1 LSI-Index

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

10.1.1 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
   ...
}
```

10.1.2 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 10-1). 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 10-1: 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

10.1.3 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.

10.1.4 Figure of Merit

The deWolff figure of merit M_N (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.

10.1.5 Keywords

10.1.5.1 Overview

Table 10-2 gives an overview of all keywords used by LSI-Index. Values for most keywords are automatically determined or have adequate default values that are sufficient for most difficult indexing problems.

Table 10-2: LSI-Index specific keywords and default values

Keyword	Default	
Tindexing		
[*seed*]		
[*index_zero_error*]		
[*index_max_zero_error #*]	0.2	
[*try_space_groups $symbol ...*] …		
[*X_scaler #*]	Cubic	0.99
	Hexagonal/Trigonal	0.95
	Tetragonal	0.95
	Orthorhombic	0.89
	Monoclinic	0.85
	Triclinic	0.72
[*X_angle_scaler #*]	0.1	
[*index_lam E*]	1.540596	
[*index_th2 E*]*... or* [*index_d E*]*…*		
[*index_I E* [*good*]]	1	
[*index_min_lp E*]	2.5	
[*index_max_lp E*]		
[*index_max_th2_error E*]	0.05	
[*index_max_Nc_on_No E*]	5	
[*Index_x0 E*]		
[*index_max_number_of_solutions*]	1000	
[*dummy*]		

10.1.5.2 Alphabetical description of keywords

Keywords are listed in alphabetical order.

dummy

Syntax	[*dummy*]
Description	Allows for the exclusion of columns in data
Example	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

Syntax	[*index_lam E*]
Description	Defines the wavelength in Å

index_max_Nc_on_No

Syntax	[*index_max_Nc_on_No E*]
Description	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_number_of_solutions

Syntax	[*index_max_number_of_solutions*]
Description	The number of best solutions to keep

index_max_th2_error

Syntax	[*index_max_th2_error E*]
Description	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.

index_max_zero_error

Syntax	[*index_max_zero_error #*]
Description	Excludes solutions with zero errors greater than *index_max_zero_error*

index_min_lp, index_max_lp

Syntax [*index_min_lp E*] [*index_max_lp E*]

Description Defines the minimum and maximum allowed lattice parameters. Typically the maximum is determined automatically.

index_th2, index_d

Syntax [*index_th2 E*]... *or* [*index_d E*]…
[*index_I E* [*good*]]

Description *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.

[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

Hint

Examples 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

Syntax [*Index_x0 E*]

Description 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 10.1.6.2.

index_zero_error

Syntax	[*index_zero_error*]
Description	Includes a zero error

seed

Syntax	[*seed*]
Description	Seeds the random number generator

try_space_groups

Syntax	[*try_space_groups $symbol ...*] … [*X_scaler #*] [*X_angle_scaler #*]
Description	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. [*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.
Hint	A series of macros described in section 10.1.6.1 simplifies the use of *try_space_groups*

10.1.6 Macros

All standard macros used by LSI-Index are defined in the file TOPAS.INC.

10.1.6.1 Space group macros

In Table 10-3 an overview of the space group macros is provided. For a reference about unique space group hkls in powder diffraction see Table 10-4.

Table 10-3: 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"

10.1.6.2 Miscellaneous macros

Index_x0_from_d, Index_x0_from_th2

Syntax	Index_x0_from_d(d, worder), Index_x0_from_th2(th2, worder)
Description	Allow to define x0 in the reciprocal lattice equation, see the *index_x0* keyword
	[d], [th2]: d and 2θ, respectively
	[worder]: reflection order

Indexing_Solutions, Indexing_Solutions_With_Zero_Error

Syntax	Indexing_Solutions, Indexing_Solutions_With_Zero_Error
Description	Allows obtaining a detailed summary of solutions by including the solution lines of the NDX file into the INP file.

10.1.7 Unique space group hkls in powder diffraction

Table 10-4: 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

Space group numbers with identical hkls	Space group symbols with identical hkls
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

Space group numbers with identical hkls	Space group symbols with identical hkls
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

10.2 LP-Search

LP-Search uses a new indexing algorithm that is independent of d-spacing extraction. It 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 avoids difficulties associated with extracting d-spacings from complex patterns comprising heavily overlapped lines; the primary difficulty being that of ascertaining the number of lines present.

10.2.1 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)
```

10.2.2 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.

11 FILE TYPES AND FORMATS

Table 11-1: 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 10.1.2 and 10.1.3. Same format as *.INP.
*.LOG	Log file. Useful for tracking input errors.
Measurement Data	
*.RAW	Bruker AXS binaries (DIFFRAC AT and DIFFRAC^{plus})
*.DAT	ASCII file formats, see Table 11-2
*.XDD, *.CAL	
*.XY, *XYE	
*.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 11-2: 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

12 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:

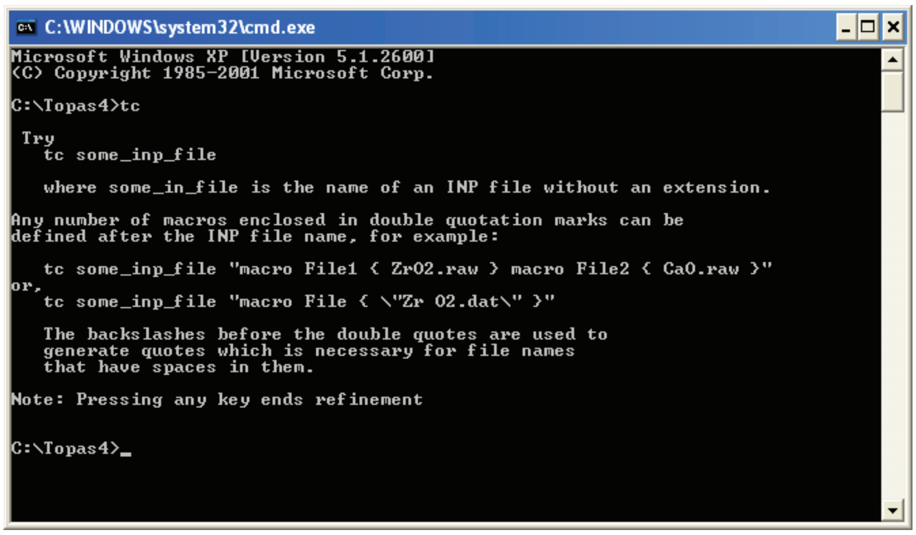
```
CuKa2(0.001)
bkg @ 0 0
```

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 horizontal divergence
axial_conv
   primary_soller_angle    2.3                      ' primary soller slit
   secondary_soller_angle  2.3                    ' secondary soller slit
```

13 AUTOMATED TOPAS OPERATION

TOPAS offers batch mode operation and user task file capabilities 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. 13-1.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Topas4>tc

 Try
   tc some_inp_file

   where some_in_file is the name of an INP file without an extension.

Any number of macros enclosed in double quotation marks can be
defined after the INP file name, for example:

   tc some_inp_file "macro File1 { ZrO2.raw } macro File2 { CaO.raw }"
or,
   tc some_inp_file "macro File { \"Zr 02.dat\" }"

   The backslashes before the double quotes are used to
   generate quotes which is necessary for file names
   that have spaces in them.

Note: Pressing any key ends refinement

C:\Topas4>_
```

Fig. 13-1: 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. With TC installed in the c:\topas4 directory, and pbso4-rietveld.inp installed in the c:\topas4\tutorial\pbso4 directory, the following command can be used:

```
c:\topas4\tc   c:\topas4\tutorial\pbso4\pbso4
```

When running TC as a user task file, typically the last measured RAW file is evaluated after measurement. In this case the RAW file name can be transferred to the INP file using a macro defined on the command line. Here's an example user task (UTF) file:

```
#BATCH_AFTER_MEASUREMENT
cmd /c cd /d c:\topas4 & tc c:\topas4\opc\opc "macro FileName { %RAW }"
```

where

- "cmd /c" starts the command processor, but hides the window
- "cd /d c:\topas4" changes the directory as required; the switch /d enforces also the change of the drive
- the "&" character concatenates several DOS commands; this is important as otherwise the individual commands would be executed in several command

processor instances (and therefore would have no effect as these instances are independent on each other)

Here TC will be run using the input file opc.inp located in the c:\topas4\opc directory. The macro *FileName* will transfer the name of the last measured RAW file ("%RAW" parameter) to opc.inp, in which the macro name *FileName* defines the RAW file to be evaluated, e.g.

```
      xdd FileName
```

Working examples for automated TOPAS operation in both batch as well as user task mode are provided in the Tutorial manual.

14 REFERENCESS

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.

Bergmann, J., Kleeberg, R., Haase, A. & Breidenstein, B. (2000): *Advanced Fundamental Parameters Model for Improved Profile Analysis.* - Mat. Sci. Forum, **347-349**, 303-308.

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. (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.

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

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.

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.

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.

Karle, J. & Hauptman, H. (1956): *A theory of phase determination for the four types of non-centrosymmetric space groups 1P222, 2P22, 3P$_1$2, 3P$_2$2.* - Acta Cryst., **9**, 635-651.

Hill, R.J. & Flack, H.D. (1987): *The Use of the Durbin-Watson d Statistic in Rietveld Analysis*. - J. Appl. Cryst., **20**, 356-361.

Hölzer, G., Fritsch, M., Deutsch, M., Härtwig, J. & Förster, E. (1997): *K$\alpha_{1,2}$ and K$\beta_{1,2}$ X-ray emission lines of the 3d transition metals.* - Physical Review A, **56**, 4554-4568.

Hovestreydt, E. (1983): *On the atomic scattering factor for O^{2-}*. Acta Cryst*.,* A**39**, 268-269.

Järvinen, M. (1993): *Application of symmetrized harmonics expansion to correction of the preferred orientation effect*. - J. Appl. Cryst., **26**, 525-531.

Evans, J.S.O. (2008). Personal communication.

Larson A.C. & Von Dreele, R.B. (2004): *General Structure Analysis System (GSAS).* - Los Alamos National Laboratory Report LAUR 86-748.

Madsen, I.C. (1999). Personal communication.

March, A. (1932): *Mathematische Theorie der Regelung nach der Korngestalt bei affiner Deformation.* - Z. Krist., **81**, 285-297.

Marquardt, D. W. (1963): *An algorythm for least-squares estimation of nonlinear parameters.* - J. Soc. Ind. Appl. Math., **11(2)**, 431-331.

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.

Pitschke, W., Mattern, N. & Hermann, H. (1993): *Incorporation of microabsorption corrections in Rietveld analysis.* - Powder Diffraction, **8(4)**, 223-228.

Sabine, T.M., Hunter, B.A., Sabine, W.R. and Ball, C.J. (1998): *Analytical Expressions for the Transmission Factor and Peak Shift in Absorbing Cylindrical Specimens.* - J. Appl. Cryst., **31**, 47-51.

Schneider, T.R. & Sheldrick, G.M. (2002): *Substructure solution with SHELXD.* - Acta Cryst., **D58**, 1772-1779.

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

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.

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.