DISCUS

Users Guide

Version 6.20

written by

Reinhard Neder
Email: reinhard.neder@fau.de

Thomas Proffen
Email: tproffen@ornl.gov

`http://tproffen.github.io/DiffuseCode`

Document created: December 7, 2025

Preface

Disclaimer

The DISCUS software described in this guide is provided without warranty of any kind. No liability is taken for any loss or damages, direct or indirect, that may result through the use of DISCUS. No warranty is made with respect to this manual, or the program and functions therein. There are no warranties that the programs are free of error, or that they are consistent with any standard, or that they will meet the requirement for a particular application. The programs and the manual have been thoroughly checked. Nevertheless, it can not be guaranteed that the manual is correct and up-to-date in every detail. This manual and the DISCUS program may be changed without notice.
DISCUS is intended as a public domain program. It may be used free of charge. Any commercial use is, however, not allowed without the explicit written permission of the authors.

Acknowledgments

The parameters for the calculation of bond valence sums were compiled by I.D. Brown, McMaster University, Hamilton, ON, Canada.

Using DISCUS

Publication of results totally or partially obtained using the program DISCUS should state that DISCUS was used and contain the following reference:

> PROFFEN, TH. & NEDER, R.B. (1997) "DISCUS, a Program for Diffuse Scattering and Defect Structure Simulations" *J. Appl. Cryst.*, **30**, 171-175

More information

This users guide can only provide program specific details. A broader discussion of simulation techniques and many DISCUS examples and macro files can be found in our book

> NEDER, R.B. & PROFFEN, TH. "Diffuse Scattering and Defect Structure Simulations - A cook book using the programs DISCUS", *IUCr Texts on Crystallography*, Oxford University Press, 2008.

The macro files used in the cookbook can also be downloaded to the DISCUS homepage at `https://github.com/tproffen/DiscusCookBookExamples`.

Contents

Chapter 1

Introduction

1.1 What is DISCUS ?

DISCUS (Proffen and Neder, 1997) is intended as a versatile tool to simulate crystal structures and the corresponding intensity distribution in reciprocal space. The program offers several features that enable the user to easily generate a structure and to introduce various defects. The main intend is to simulate defect structures, the program, however, is not limited in that respect. Ideal structures can be simulated as well and usually will form the basis from which the defect structure is developed.

The program can read a whole crystal or the asymmetric unit of a unit cell. The latter is expanded to the whole unit cell by use of the space group symbol. The definition of rigid molecules is supported by DISCUS. The structure can be stored as structure file or certain layers or projections of the crystal can be saved for graphical display. The program DISCUS is part of the diffuse program package. Several tools are available to modify single atoms or molecules within the structure or to alter the complete crystal. These tools include thermal displacements, waves, domains, stacking faults and generalized symmetry operations. One feature of DISCUS is the possibility to introduce correlated defects and distortions using Monte Carlo (MC) simulations. Various tools allow the analysis of a given defect structure including the calculation of the PDF of a given structure.

The Fourier transform segment of the program allows the user to calculate the intensity distribution along a line, arbitrarily oriented plane or a three dimensional volume through reciprocal space. The resolution in reciprocal space can be chosen by the user. The resulting intensity maps can be written in several output formats, to be displayed by standard visualization programs, the program *KUPLOT* or be printed directly. The list of output formats includes the NeXus format. Neutron- as well as X-ray and electron scattering can be calculated. DISCUS supports the subtraction of the average structure factor and the calculation of the diffuse intensities as average over many small crystal volumes to create smooth noise free diffraction pattern. A detailed discussion is given in chapter 3. The direct analysis of measured diffuse scattering is possible using the Reverse Monte Carlo segment of the program. The program allows the refinement of the scattering intensity directly as well as the refinement of the PDF.

The program uses a command language to interact with the user. No predefined requirement exists for a given sequence of commands. The commands can be typed at the DISCUS prompt, or read from a macro file. The command language includes a FORTRAN style interpreter that

allows the user to program loops and logical structures. Several real and integer variables as well as structural variables can be used to design the intended defect structure. On line help is provided which gives the user information on any of the DISCUS commands as well as examples for typical DISCUS sessions. More detailed information how to get further help is given in section 1.2.

1.2 Getting started

The DISCUS section is reached from the main DISCUS SUITE by typing the command `discus` at the the prompt of the DISCUS SUITE.

The program uses a command language to interact with the user. The command `exit` terminates the program and returns control to the shell. All commands of DISCUS consist of a command verb, optionally followed by one or more parameters. All parameters must be separated from one another by a comma ",". There is no predefined need for any specific sequence of commands. DISCUS is case sensitive, all commands and alphabetic parameters MUST be typed in lower case letters. Basic line editing and recall of commands is possible. For more information refer to the reference manual or check the online help using ('help command input'). Names of input or output files are to be typed as they will be expected by the local operating system. If necessary include a path to the file. All commands may be abbreviated to the shortest unique possibility. At least a single space is needed between the command verb and the first parameter. No comma is to precede the first parameter. iAll text to the right of a hash tag "#" or exclamation mark "!" is considered a comment. iThis does not include these signs within a character string that is enclosed by single or double quotation marks.

The symbols used throughout this manual to describe commands, command parameters, or explicit text used by the program DISCUS are listed in Table 1.1. There are several sources of information, first DISCUS has a build in online help, which can be accessed by entering the command `help` or if help for a particular command <cmd> is wanted by `help <cmd>`. This manual describes background and principle functions of DISCUS and should give some insight in the ways to use this program.

Symbol	Description
"text"	Text given in double quotes is to be understood as typed.
<text>	Text given in angled brackets is to be replaced by an appropriate value, if the corresponding line is used in DISCUS . It could, for example be the actual name of a file, or a numerical value.
text	Text in single quotes exclusively refers to DISCUS commands.
[text]	Text in square brackets describes an optional parameter or command. If omitted, a default value is used, else the complete text given in the square brackets is to be typed.
{text | text}	Text given in curly brackets is a list of alternative parameters. A vertical line separates two alternative, mutually exclusive parameters.

Table 1.1: Used symbols

As we have mentioned before, DISCUS is controlled by a powerful command language which

is common to all programs in the DISCUS package. A detailed description of the command language can be found in the DISCUS Reference Guide which is included in the distribution. Variables specific to the program DISCUS are shown in Tables 1.2 and 1.3. DISCUS also provides a number of structure related functions which are listed in Tab. 1.4 and 1.5.

Variable	Description
n[1]	Number of atoms within the crystal
n[2]	Number of different scattering types, i.e. atoms
n[3]	Number of atoms within the unit cell
n[4]	Number of molecules within the crystal
n[5]	Number of different molecule types
n[6]	Number of molecules within the unit cell
n[7]	Number of non-void atoms in the crystal
cdim[i,1]	Lowest coordinate of any atom (i=1,2,3 for x,y,z)
cdim[i,2]	Highest coordinate of any atom (i=1,2,3 for x,y,z)
env[i]	Index of neighboring atoms after 'find' command
menv[i]	Index of neighboring molecules after 'find' command
lat[i]	Lattice parameters (i=1..6 for $a, b, c, \alpha, \beta, \gamma$
vol[1]	Unit cell volume
rvol[1]	Reciprocal unit cell volume
sym_n[1]	Space group number of the structure

Table 1.2: Crystal related variables

1.3   Technical notes

Most of the time you don not have to worry about the interior technical design of DISCUS. Most of the time the program will allocate its required memory and will not need user intervention. As the command language is designe for utmost flexibility, the program will not know if you still need a piece of the allocated memory or not. Again, most of the time this is not an issue, as few section of the program may require huge amount of memory. These sections are the Fourier menus (fourier and powder) and particularly the pdf menu. The latter can be very memory hungry if you deal with many different atom types and many different molecule types. It has been design to release temporary memory requirements. As such it should not really need help with the memory issues.

The Fourier menus might take up quite a bit of memory, if large (3-dimensional) sections of reciprocal space are calculated. DISCUS cannot know if you are done with the final Fourier map, After all, you might want to save it in the output menu in different formats. This prevents DISCUS from a fully automatic release of the allocated memory.

To free all memory, you can use the `reset` command within any menu or at the DISCUS top level. This command will free all allocated arrays and furthermore set all menu settings back to the default values at program start.

Variable	Description
m[i]	Number of scattering type (i.e. atom type) for atom i
b[j]	isotropic thermal factor B for atom **type** j
occ[j]	occupancy parameter for atom **type** j
at_name[i]	Atom name for atom i
at_type[j]	Atom name for atom **type** j
in_mole[i]	Molecule in which atom i is
prop [i]	Property of atom i
surf[i,k]	Surface vector for atom i, components k=1,2,3
at_type[i]	Atom name for atom type i
x[i]	fractional x position of atom i
y[i]	fractional y position of atom i
z[i]	fractional z position of atom i
mol_cont[i,j]	Index of atom j in molecule i
mol_len[i]	Number of atoms in molecule i
mol_biso[i]	Isotropic B-value for molecule i
mol_clin[i]	Linear correlation term for molecule i
mol_cquad[i]	Quadratic correlation term for molecule i
mol_dens[i]	Density of object i
mol_type[i]	Type of object / molecule / domain i
pdf_dens[i]	Number density for pdf calculations
pdf_scal[i]	Scale factor for pdf calculations

Table 1.3: Variables related to individual atoms and molecules

Type	Name	Description
real	bang(u1,u2,u3, v1,v2,v3 [,w1,w2,w3])	Returns the bond angle in degrees between **u** and **v** at site **w**. If **w** is omitted, the angle between direct space vectors **u** and **v** is returned.
real	blen(u1,u2,u3 [,v1,v2,v3])	Returns the length of the real space vector **v-u**. The vector **v** defaults to zero.
real	dstar(h1,h2,h3 [,k1,k2,k3])	Returns the length of reciprocal vector **k** - **h** in Å^{-1}. Vector **k** defaults to zero.
real	qstar(h1,h2,h3 [,k1,k2,k3])	Same as dstar, except that the result is Q = 2π d*. Returns the length of reciprocal vector **k** - **h** in Å^{-1}. Vector **k** defaults to zero.
real	rang(h1,h2,h3, k1,k2,k3 [,l1,l2,l3])	Returns the angle between reciprocal vectors **k** - **h** and **k** - **l** at site **k**. If **l** is omitted, the angle between reciprocal vectors **h** and **k** is returned.
real	scalpro(u1,u2,u3,v1,v2,v3 [,"dd"|"rr"|"dr"|"rd"])	Returns the scalar product between the two vectors **u** and **v**. Both vectors may be given in direct or real space coordinates, flagged in parameter no 7. "d" means real space, "r" reciprocal space.
real	vprod <u1>,<v1>,<w1>, <u2>,<v2>,<w2> [,<flag>]	This is not a function but rather a command that calculates the vector product **1** X **2**. The <flag> is a string of three characters that indicates whether the input vectors or the output vector are given in direct or reciprocal space coordinates. The three characters correspond to: first input vector, second input vector resulting vector "drd" means: vector one is given in direct space coordinates, vector 2 is in reciprocal space coordinates (hkl) and the resulting vector product is to be given in direct space coordinates Any combination of "d" and "r" is allowed.
real	bragg dval:<P_d>, theta:<P_theta> , lambda:<P_lam>	Again not a function but a command. You should provide any two of the three parameters in $\lambda = 2dsin(\Theta)$ and the command will calculate the other one. Each of the three parameters can be entered in various forms, for example dvalue as d, d* or Q, the wavlength as symbol or as energy, the angle as Theta or 2Theta. See the online help for full details.

Table 1.4: Crystallographic functions

Type	Name	Description
logical	isprop(atom [,[and:]property] [,or:property])	Tests if an atom has the property / properties specified by property. The test can check a single property, if the property is given by its name as in : 'normal', 'molecule', 'domain', 'outside', 'external', 'internal', 'ligand'. Alternatively several properties can be checked simultaneously if the property is a list of capital and/or small letters as in 'NMDOEIL', 'nmdoeil'. A capital letter signifies that the property must be present, a small letter that the property must be absent. The properties in the "and:" parameter must ALL be present, i.e. they are tested with a logical "and". The properties in the "or:" parameter are tested with a logical "or", i.e. the function is true if any of the listed properties fulfills the condition (capital letter==present, small letter==absent). At least one of the two parameters must be present. If only one parameter is given and the "and:" is omitted, the default is to test with the "and:" condition.

Table 1.5: Crystallographic functions continued

Chapter 2

Creating structures

The first steps towards any simulation are to create the desired structure and possibly to modify it. DISCUS offers two different ways to perform this task, the structure can either be generated from the contents of a asymmetric unit of a unit cell or be completely read from a file. For each atom the program stores its type, its fractional coordinates (x,y,z) an isotropic thermal coefficient (B) and a set of property flags. If the crystal is generated from an asymmetric unit, the program uses the space group symbol read from the data file to generate the unit cell from the asymmetric unit. Prior to this generation, the space group symbol is checked for consistency with the lattice constants. An error message flags any inconsistencies. The unit cell is copied to generate a crystal of desired (rectangular) dimensions.

The following sections describe the structure file formats and how to group atoms to (rigid) molecules. Once a crystal (which could be as small as one atom) is generated, several tools are provided to modify the crystal. These tools can be grouped into those that affect individual atoms and those that affect the whole crystal. The first group includes the following modifications of an individual atom: moving, replacing, removing and inserting new atoms. These are discussed in chapter 5. The second group includes at present the following modifications of the crystal at large, for more details see chapter 6: thermal displacement, waves, stacking faults and domains, generalized symmetry operations (chapter 4), Monte-Carlo modelling (chapter 8), and Reverse-Monte-Carlo modelling (chapter 10).

2.1   Reading structure files

DISCUS uses its own file format for crystal structures. In part this is a legacy feature since the *CIF* file format was not around at the time DISCUS was started. A second aspect concerns the fact that a *CIF* file is restricted to the content of a single unit cell. As DISCUS often builds large blocks of unit cells with reduced periodicity, the DISCUS file format still bears many merits.

For the use of initial input a *CIF* file is fine and can be read directly within the `read` menu. Alternatively the *CIF* file can be imported into the DISCUS file format. Further file formats can be imported into the DISCUS format.

As the standard choice the DISCUS specific format is described in this section.

A unit cell (actually the asymmetric unit within the unit cell) or a whole structure is read from file by the command `read`. The format of these two file types is identical. If a unit cell is read,

the contents of the file is regarded as the asymmetric unit of the unit cell. The space group information is used to generate the whole unit cell. If, on the other hand, a structure is read, the contents of the file is taken as it is.

A keyword controlled structure file format is used allowing more flexibility, e.g. the use of molecules. However, DISCUS is capable to read both the very old and the new structure files automatically recognizing the correct format. Furthermore the command `format nokey` in the `save` segment allows the user to save a structure using the old format (see section 2.4). As the non keyword file format lacks many of the storage features its use is highly discouraged.

2.1.1 Structure file format

The standard DISCUS structure file is a simple text file starting with a section with keywords setting parameters like lattice constants or space group followed by a section with the actual atoms. The sequence of keywords is arbitrary with two exceptions, the first line **must** contain the keyword `title` setting a title and the last keyword **must** be `atoms` followed by a list of atoms and/or molecule definitions (see section 2.1.2). Each atom within the asymmetric unit or complete structure is defined at least by its name (e.g. ZR), the fractional coordinates (x, y, z) and an isotropic temperature factor B. The property flag is optional and defines if the atom is a normal atom, inside a molecule, inside or outside the crystal, within a domain. Starting with version 5.18.0 one can in addition specify the molecule number to which the atom belongs and an occupancy for the atom.

A list of valid keywords within a structure file is given in Table 2.1.

Keywords are processed the same way as normal commands and parameters have to be separated by commas. An example input file for cubic zirconia (ZrO_2) is shown below.

```
title Structure of cubic ZrO2
spcgr Fm-3m
cell  5.14, 5.14, 5.14, 90., 90., 90.
atoms
ZR    0.00000, 0.00000, 0.00000,  0.5
O     0.25000, 0.25000, 1./4.  ,  1.0
```

A full information input file as of version 6.18.0 has the form:

```
title Structure of cubic ZrO2
spcgr F m -3 m, setting:abc, origin:1
cell  5.14, 5.14, 5.14, 90., 90., 90.
scat  Zr , O
occ   1.00, 1.00
adp   1.01, 2.01
anis  type:1, values:[0.006332]
anis  type:2, values:[0.012665]
atoms x,    y,    z,     Biso, Property, MoleNo, MoleAt, Occ, St,  Sh,  Sk,  Sl
ZR    0.00, 0.00, 0.00, 1.01, 1,         0,       0,      1.0, _,   0,   0,   0
O     0.25, 0.25, 1./4., 2.01, 1,        0,       0,      1.0, _,   0,   0,   0
```

The first line is the required `title` line that describes the structure. The next line specifies the space group $Fm\overline{3}m$. The symbols used should be the Hermann-Mauguin symbols used in International Tables for Crystallography Vol. A Aroyo et al. (2016). A center of inversion as in this example should be given as – sign immediately preceding the axis. Lattice types need to be given as capital characters, mirror planes as small characters. Monoclinic cell choices 2,3 or

---

Keyword	Description
#	Allows comments in the keyword section of the structure file..
adp	A list of isotropic B values for all atom types. If present, these values will override the parameter specified for an atom upon reading a structure.
anis	A List of the anisotropic atomic displacement parameters.
atoms	Marks the start of the atom list and must be the **last** keyword except for 'molecule'. Parameters are the names of the columns that are present for each atom. Valid columns are: x, y, z, Biso: These four columns must always be present. Property: The property flag MoleNo, MoleAt: Molecule number and entry number for the atom. These two columns are written by DISCUS during a save. To build a structure with molecules it is easier for you to use the `molecule` keyword. Occ: Occupancy for this atom type The sequence of the columns is arbitrary, yet the recommended sequence is: x,y,z, Biso, Property, MoleNo, MoleAt, Occ
cell	Parameters $a, b, c, \alpha, \beta, \gamma$ define the lattice constants of the crystal. Note that **all** six parameters must always be given.
generator	Allows to define generators additional to the ones defined by the space group. The 12 parameters define the symmetry operation.
molecule	Defines rigid molecules. The keyword must stand in the atoms section of the structure file (see section 2.1.2).
ncells	Values are `ncellx, ncelly, ncellz, nc, natoms, ntypes nummole, nummoletype numnoleatom`. Parameters `ncellx, ncelly, ncellz` are the number of unit cells of the crystal, along the three dimensions, `nc` are the number of atoms within a unit cell. `natoms` is the total number of atoms in the crystal, `nscat` the number of atom types. The last three value describe the number of molecules, the number of molecule types and the total number of atoms within all molecules. This information is needed if a complete structure is read in order to allow DISCUS to determine the crystals size. Upon input of an asymmetric unit this information is replaced by the contant of the actual cell file.
occ	A list of atom occupancy values
scat	A list of atom types. This will ensure that the atom types have a fixed sequence.
spcgr	Sets the space group for the crystal.
symmetry	Defines an additional symmetry operation defined by the 12 parameters given.
title	Sets a title for the structure file. This **must** be the first keyword in the file.

Table 2.1: List of keywords for structure files

unique c-axis will be assumed if the corresponding non standard Hermann-Mauguin symbol is used. Note that the space group symbal my be presented either as `Fm-3m` or with blanks between the symmetry elements along the different directions as in `F m -3 m`, or for a more involved symbol like `P 42/n b c`.

The optional parameter `setting:` allows to use a non standard setting, especially for the orthorhombic space groups. You can specify either the standard space group name and an alternative setting or the non standard space group name. Another optional parameter `1` or `2` is used to differentiate origin settings one and two. To ease readability, this optional parameter should be used as `origin:1,` respectively `origin:2.`

DISCUS checks the given space group symbol for contradictions between the space group symbal and the lattice constants and in case of an error the unit cell is not read. A complete list of valid space group symbols is part of the online help and can be accessed via 'help space_group' from the DISCUS command line.

The next line in the example above gives the lattice constants of $a = b = c = 5.14$Å and $\alpha = \beta = \gamma = 90$ degrees. Note that DISCUS requires all six values to be given. The keyword 'atoms' in the structure file header must be the last keyword. It starts the section with the list of atoms. Here zirconium occupies site 4(a) on (0,0,0) and oxygen is on 8(f) at ($\frac{1}{4}, \frac{1}{4}, \frac{1}{4}$). The isotropic temperature factors for Zr and O are set to 0.5 Å^2 and 1.0 Å^2 respectively.

If the lattice parameters or the atom coordinates are separated by comma, you can use simple expressions like 1./2. or even variables instead of explicit numerical values.

The full example listed above is the current recommended style. The header lines :

```
scat  Zr  ,  O
occ   1.00,  1.00
adp   1.01,  2.01
anis  type:1, values:[0.006332]
anis  type:2, values:[0.012665]
```

specify all atom types via the `scat` line. Corresponding `occ` and `adp` lines are required. The side effect of this line is that atom types will be numbered as they appear on the `scat` line, even if the first atom in the actual atom list does not correspond to the first atom on the `scat` line.

Note that the parameters on the `adp` line do not have to reflect actual values of `B` $= 8\pi^2 \langle u^2 \rangle$. Instead it is usually best to specify a numbering scheme, where the whole number reflects the atom type, and the decimal number enumerates identical chemical elements with individual atomic displacemnt parameters. The `Biso` parameter for each atom in the structure must correspond to one of th numbers on the `adp` line.

The actual ADPs are defined by the `anis` lines. One line should be present for each atom type, specified by the `type:` parameter. The actual values are given as `values:[ U11, U22, U33, U32, U13, U11]`. For atoms with isotropic ADP specify this as: `values:[ UISO ]`.

Additional generators can be defined through the optional `generator` keyword. These generators act identical to the generators defined through the space group symbol. All previously generated copies of the atoms in the asymmetric unit are copied by this generator, and will in turn be copied by any generators following later. Since these additional generators are applied after the space group generators, you can use these generators to create non-standard groups or to create a set of symmetries that does not from a group. The syntax of the 'generator' keyword is as follows:

```
generator g11,g12,g13,g14, g21,g22,g23,g24, g31,g32,g33,g34
```

Copies of an atom at (x,y,z) will be calculated using the following equation:

$$
\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} g_{14} \\ g_{24} \\ g_{34} \end{pmatrix}
\tag{2.1}
$$

Additional symmetry operations can be defined through the optional `symmetry` keyword. These symmetry operations act different than the generators described above which are defined through the space group symbol or listed as additional generators. The symmetry operations copy only those atoms created by the generators. In contrast to generators they do not act on copies of the atoms created by previous symmetry operations. Both keywords `generator` and `symmetry` define the symmetry operation in a similar way using 12 parameters as shown in equation 2.1

The following example shall illustrate the difference between generators and additional symmetry operations. The following two generators

```
generator 1,0,0,0.5, 0,1,0,0.5, 0,0,1,0.0
generator 1,0,0,0.5, 0,1,0,0.0, 0,0,1,0.5
```

would create the following copies of an atom at (0,0,0): $(\frac{1}{2}, \frac{1}{2}, 0)$, $(\frac{1}{2}, 0, \frac{1}{2})$ and $(0, \frac{1}{2}, \frac{1}{2})$. In contrast similar symmetry operations

```
symmetry 1,0,0,0.5, 0,1,0,0.5, 0,0,1,0.0
symmetry 1,0,0,0.5, 0,1,0,0.0, 0,0,1,0.5
```

will only generate the following two copies of an atom at (0,0,0): $(\frac{1}{2}, \frac{1}{2}, 0)$ and $(\frac{1}{2}, 0, \frac{1}{2})$ since the symmetry operations will not act on previously generated copies of the atom at (0,0,0). The second symmetry operation copies only the atom at (0,0,0), not the atom at $(\frac{1}{2}, 0, \frac{1}{2})$, since this atom was created by the previous symmetry operation.

2.1.2 Using molecules

The keyword controlled structure file format of DISCUS allows the definition of molecules using the `molecule` keyword. This keyword is allowed anywhere between the atoms of the unit cell file. It marks the beginning of a group of atoms that are grouped to form a molecule. The individual atoms are listed in the usual way (see section 2.1.1). The keyword `molecule end` signals the end of a molecule. All atoms still listed in the unit cell file are treated as individual atoms. The molecule related keywords are listed in table 2.2.

Starting with DISCUS 3.6, the internal symmetry of a molecule is automatically determined from the Wyckoff position of the first atom. It is mandatory that this first atom is at a Wyckoff position, whose local site symmetry corresponds to the molecule symmetry. If the molecules does not posses an atom at this site, insert a "VOID" at the site.

The internal symmetry of the molecule can be specified using the 'generator' and 'symmetry' sub-keywords. The generators are internal symmetry operations of the molecule. DISCUS compares the lists of atoms created by the space group and by the molecule generators. Identical sections are linked to one molecule. Atoms created by other symmetry operations, e.g. lattice

---

Keyword	Description
molecule	Defines the start of a molecule.
molecule atoms	Lists atom numbers belonging to current molecule.
molecule biso	Defines an overall B-value for the current molecule.
molecule character	Defines the character of a molecule.
molecule content	Defines the start of particular molecule type.
molecule density	Defines the density of molecules with "object" character.
molecule end	Defines end of a molecule atom list.
	Starting with DISCUS 3.6 the following keywords have become obsolete.
molecule generator	Defines generators for the internal symmetry of the molecule.
molecule symmetry	Defines internal symmetry operations for the molecule.

Table 2.2: List of keywords molecules

centering will form a new molecule of the same type. The generators of the molecule symmetry should be the generators that create the site symmetry. See the section on site symmetry in the International Tables A Aroyo et al. (2016) for further details. As in the previous section, symmetry operations will only act on the *original* atoms of the molecule whereas generators will operate on previously generated copies of atoms as well. There is one **important restriction** how molecules are defined in DISCUS: The first atom of any molecule defines the origin of the molecule used by various subsequent commands. In case the origin lies on a symmetry element of the space group it must be located at the point of highest symmetry of the molecule. If the structure does not have an atom at this site you must include a *void* on this site. This could be the case e.g. if you have an empty triangle on a threefold axis.

Alternatively to defining a molecule as discussed above, the command `molecule content` and `molecule atoms` might be used to define molecule types and the corresponding list of atom indices belonging to that molecule. This procedure is used to be able to save structures containing molecules since the order of the atoms required by various DISCUS functions might prevent storing atoms in groups belonging to a particular molecule. Check section 2.4 for more details about saving structures.

2.1.3   Using occupancies

Occupancies are a concept that is not at the heart of DISCUS, as the program is intended to simulate an actual crystal structure. In a real crystal structure, a site in a given unit cell is occupied by an actual atom or a void. Averaging all unit cells will result in an occupancy that may deviate from 1.0.

The usual style for DISCUS to deal with partial occupancies or with mixed occupancies by two or more atom types on a given site is to start with a fully ordered structure. In a second step atoms can be replaced by another atom type with the build in command `replace`. A random replacement will be performed by this command. For a small crystal the actual occupancy after a `replace` command may deviate considerably from the intended occupancy. At the extreme, it is impossible to have a real occupancy other than 1.0 or 0.0 within a single unit cell at least for a site with multiplicity of one. Even for a site with higher multiplicity the actual distribution of

the two or more atom types onto the site will in most cases severely change the symmetry.

To deal with structures that do not posses disorder other than randomly distributed deviations from full occupancy, an occupancy parameter was added to DISCUS in version 5.18.0.

The treatment of an occupancy parameter in the structure file depends on the input mode.

`stru` If a structure is read with the `stru` command, each atom type retains its occupancy value. The effect on a Fourier, powder pattern and PDF is the same as if the actual structure would have been modified with the `replace` command. If you plot the structure no difference to a fully occupied structure will be recognized.

`cell` The default behavior is to discard occupancies and to reset all values to 1.0. To change this behavior DISCUS offers an optional parameter to the `cell` command. This parameter allows to instruct DISCUS to `apply` the occupancies, i.e. the atom type in question will be (partially) substituted by voids. Alternatively the occupancy parameter can be `kept` to give the same result as if the structure were read with the `stru` command.

2.2 Freely generated structure

As an alternative, DISCUS can create a structure freely by the `free` command. This alternative is helpful to avoid disk I/O.

```
1  read
2  free    5.14,  5.14,  5.14,  90.00, 90.00, 90.00, Fm-3m
3  insert  Zr, 0.0000, 0.0000,  0.0000,  0.1
4  insert  O , 0.2500, 0.2500,  0.2500,  0.1
```

This example illustrates the alternative style to create the asymmetric unit of cubic zirconia. The last parameter in line 2 instructs DISCUS that the freely read structure should belong to space group $Fm\overline{3}m$. This parameter is optional and defaults to P1. After these four commands DISCUS has created just the atoms within the asymmetric unit. To build a full crystal, save this asymmetric unit (optionally save it as an internal file) and read it again with the `cell` command. See section 2.4 for more information on the save procedure.

2.3 Internal storage of a structure

All atoms are stored sequentially in an array. If a complete structure is read using the `stru` command, the atoms are stored in the same sequence as found on the input file. If several unit cells have been generated by the command `cell` in the `read` sub level, the contents of all unit cells are stored consecutively. The `cell` command always generates a rectangular block of nx, ny, nz unit cells along the direct axes. The innermost, fastest loop is over x, the outermost, slowest loop is over z.

Within each unit cell, the atoms are stored in the sequence that they have been read from file. Each atom is multiplied by the symmetry generators of the respective space group. The sequence of generators has been chosen identical to that in the International Tables for Crystallography Aroyo et al. (2016), Vol. A. The sequence of symmetrically equivalent atoms is therefore identical to that of the International Tables.

The generators of a space group are applied in the same order as given in Vol A of the International Tables. Thus the generated atoms will be in the same sequence as printed for each Wyckoff position of each space group.

If the space group is centered, the generators that correspond to symmetry operations are applied first and then each block is repeated by application of the centering generators. This is illustrated by the following example:

> *Space group C2 No.5, Wycoff position 4(c) x,y,z*
> *The atoms on this site are*
> $(0, 0, 0; \frac{1}{2}, \frac{1}{2}, 0)+$
> $x, y, z; \overline{x}, y, \overline{z}$

DISCUS stores these four atoms on position 4(c) in the following sequence:

> x, y, z
> $\overline{x}, y, \overline{z}$
> $x + \frac{1}{2}, y + \frac{1}{2}, z$
> $\overline{x} + \frac{1}{2}, y + \frac{1}{2}, \overline{z}$

Prior to DISCUS 3.6, in accordance with the header lines in the International Tables, the generators required for centered space groups were applied first. The atoms generated by these generators immediately followed each position generated by the ordinary symmetry operators as illustrated for the following example:

> x, y, z
> $x + \frac{1}{2}, y + \frac{1}{2}, z$
> $\overline{x}, y, \overline{z}$
> $\overline{x} + \frac{1}{2}, y + \frac{1}{2}, \overline{z}$

The old sequence caused problems for molecules in rhombohedrally centered space groups.

In order to provide backward compatibility, the `define` command allows you to switch between the newer storage sequence and the older one.

The origin within the model crystal is taken as the center of the simulated structure, i.e. the coordinates of an atom on site (0,0,0) of a crystal 20x20x20 unit cells in size would range in x,y and z from -10.0 to 9.0.

If the chemistry or position of an atom is modified, its position within the structure array remains the same. If it is deleted, it is replaced by atom type VOID. The position of all other atoms remains unchanged. This feature enables the user to calculate the position of an atom in the structure array, irrespective of any changes that took place in the structure. All new atoms are added at the end of the structure array. Their sequence is entirely based on the sequence of generation.

The command `trans` in the `chem` sub level of DISCUS allows to transform between the index of an atom and its unit cell and site number. Likewise the general DISCUS command `index2cell` can be used for this calculation.

Alternatively the atom index l' for an atom on site l in unit cell i, j, k can be calculated using the following equation:

$$l' = \{(k-1) \cdot nx \cdot ny + (j-1) \cdot nx + (i-1)\} \cdot n[3] + l \tag{2.2}$$

Here nx, ny, nz are the crystals dimensions in unit cells and $n[3]$ (see table 1.2 in section 1.2) contains the number of atoms per unit cell. Some calculations of DISCUS (e.g. usage of lots in Fourier sub level, quick mode for bond length calculations, ..) require this particular order of storage of the atoms. However, in order to save CPU time for the Fourier transform and disk space when saving a structure, the `purge` command may be applied which will delete all empty sites in the crystal and the given relation between atom index and unit cell/site is invalid. The authors generally do **not** recommend the usage of the 'purge' command. The `purge` command will be useful as last step prior to a lengthy calculation of a powder diffraction pattern via the Debye-Scattering-Equation.

2.4 Saving structures

A particular structure stored in DISCUS can be saved to a file using the `save` command. In cases where `save` is followed by a filename, the structure is directly saved using the currently set options.

If `save` is entered alone, DISCUS will enter a `save` sub level which allows the user to select various options and select all or only particular atoms to be saved. Note that the settings are valid for all subsequent calls of `save filename`. Generally the user can choose between the old structure file format (see section 2.1) and the new keyword controlled structure file format, which is the recommended format to be used. Beside the required keywords the user is able to select which additional keyword shall be written to the file. Note that in cases where a structure containing molecules is saved without selecting the `molecule` keyword, the molecule information of the crystal will be lost. Check the online help for `save` for a complete list of options.

Starting with DISCUS Version 5.0 you can save a structure into an internal memory location within the CPU instead of onto the disk. Simply prepend the filename with the characters "internal". DISCUS will allocate memory to store the structure. To recover the structure read it with `read cell` or `stru`. The filename must of course start again with "internal". This internal storage is highly recommended for temporary or intermediate structures that you do not really need. This holds especially during a refinement on a high performance computer cluster.

2.5 Exporting structures

DISCUS can export the current crystal structure for two purposes. The first purpose serves to use the structure with a different program, the second serves to display the crystal structure.

2.5.1 Exporting to other programs

Currently DISCUS can export the structure for the programs listed in Tables 2.3 and 2.4.

Format	Description
CIF	A minimum content CIF file
	export cif, <filename>.cif
	DISCUS writes a minimal CIF file. All atoms are transformed into a single unit cell. Along each of the axes the new unit cell dimensions are an integer multiple of the original unit cell. The factors are chosen to the samllest possible integer value. as default, the space group is set to P1i, alternatively use the optional parameter **spcgr**:original to write the current actual space group.
	Information on molecules is lost.
RMCPROFILE	Crystal structure storage format as used by RMCProfile
	export rmcprofile, <filename>{.rmc6f}, version:6
	export rmcprofile, <filename>{.rmc6f}, version:7
	DISCUS writes a RMC6F file. the extension ".rmc6f" is optional and will be appended to the filename automatically if absent.
	The optional **version**: parameter allows to switch between version 6 and 7. For version 7, a Deuterium atom is writen as "2H". Furthermore, for version 7 the optional string [type_number] is appended to the atom type. This type number allows to distinguish atoms of identical chemical nature that are on different sites in the asymmetric unit.
	The unit cell parameters are multiplied by the smallest integer number to include the whole crystal in one unit cell.
	The origin of the DISCUS structure is shifted such that all atom coordinates are positive.
	Information on molecules is lost.
VASP	Writes a POSCAR file as used by VASP
	export vasp, <filename>
	export poscar, <filename>
	The filename is optional and defaults to "POSCAR".
	Atoms are written as fractional coordinates of the crystal structure.
	The origin of the DISCUS structure is shifted such that all atom coordinates are positive.

Table 2.3: List of supported export file formats

2.5.2 Exporting to visualization programs

The structure, or parts thereof can be exported to file in formats suitable for display by corresponding programs. Currently supported programs are KUPLOT which is part of the DISCUS package and a number of structure plotting packages such as *ATOMS* and *DIAMOND*. A simplified *CIF* file can be read by most crystallographic packages. Check the online help for an up-to-date list of supported programs.

Different export option are available in the `plot` sub level of DISCUS. Individual atom or molecule types to be included in the output might be selected. The extend of the crystal to be exported as well as an optional limitation to a given slice within the crystal allow the user to

Format	Description
SHELX	Write an ins file as used by SHELX programs
	export shelx, <filename>.ins
	A regular SHELX instruction file is written.
	Identical atom types on different sites are written as "O", "O001", , "O002", etc.
	Information on molecules is retained.
	Optional parameters are :
	cycle:<number> Define the number of least squares cycle on the "L.S." instruction
	lambda:<value> Define the wave length numerical value to be placed on the "CELL" instruction
	lambda:<symbol> Define the wave length via a symbol like CUA1 to be placed on the "CELL" instruction
	origin:yes Tells DISCUS to shift the origin of a centrosymmetric structure to the location of the center of symmetry.
	An alternative value to "yes" is "shift" or "no". The latter does not shift the origin (Default)
	force:yes DISCUS will check if the structure is restricted to a single unit cell to avoid placing an extended structure into the ins file. A large molecule may have coordinates beyond a range of one, with this option DISCUS will write such a molecule.

Table 2.4: List of supported export file formats, continued

select specific regions within the crystal. Such a slice is defined by a point **v** in real space its thickness and its normal. For each atom in the crystal the vector from the point **v** to the atom is projected onto the real space normal to the slice. If the length of this projection is less than the thickness, the atom is plotted, else not. DISCUS simply writes the coordinates of selected atoms. The sequence of indices in the output file can be defined by the user as any permutation of x,y,z; x,z,y;... The output for KUPLOT will result in a projection of the structure along one of the crystallographic axes or a projection along the normal of the slice onto the plane, while the *GNUPLOT* format can be used for a three dimensional viewing. The file format for KUPLOT includes marker type, color and size for the individual atom or molecule representation. These properties can be altered using the set command within the plot sub level. Assuming the structure has been written to file *atoms.xy*, the following KUPLOT macro would be appropriate:

```
load cr,atoms.xy    ! loading the data file
plot                ! plot it
```

For more information about the program KUPLOT refer to the KUPLOT manual or online help. To import into a CIF file viewer export with the program cif option:

```
plot
  program cif
  sel     all
  outfile your_name.cif
```

```
    run
  exit
```

Several of the CIF file viewer programs will automatically create a copy of an atom whose factional coordinate along any axis is zero. This atom will then be displayed with coordinates "one" along the axis. To prevent this behavior, DISCUS shifts the structure such that the origin is a little "left" of all fractional coordinates.

As a CIF file contains atoms within a single unit cell only, DISCUS transforms the original unit cell to a supercell to enclose the full crystal in a single cell. To avoid issues with atoms whose fraction coordinates are "zero" or "one", the origin is shifted. The new unit cell is set to the smallest integer multiple of the original cell plus a margin of one extra unit cells.

If the program is set to "`jmol`", DISCUS will still write a CIF file to disk. In addition you can use the optional parameter `plot:inter` on the `run` command to start the Jmol program with an interactive plot of the structure, see section 2.7 for fulöl details.

2.6 Importing structures

DISCUS can import crystal structures from a few external file formats, as listed in Table 2.5. The relevant commands are found in the `import` menu. DISCUS will read the import structure and automatically save a DISCUS type file, whose extension has been replaced by **stru**.

Format	Description
cif	The Crystallographic Information File format, as described in the International Tables Aroyo et al. (2016) Vol. F. As default the atom symbols are used as atom names, Use **names**:cif to use the atom labels.
rmcprofile	Crystal structure storage format as used by RMCProfile As default the atom sequence is kept in the RMCprofile sequence. Use **sort**:discus to sort the atoms into the DISCUS sequence, unit cell by unit cell
shelx	Import a res or ins file as used by SHELX programs As default the atom symbols from the "SFAC" instruction are used as atom names, Use **names**:shelx to use the atom labels. Two options can be used to set up a group of macros for a refinement: **diffev**:[parameter list] **refine**:[parameter list] See the DIFFEV and REFINE manuals for a full description.
crystalmaker	Crystal structure storage format as used by CrystalMaker

Table 2.5: List of supported import file formats

2.7   Plotting / Visualization

The main task of the DISCUS section certainly is to create a wide range of structure types, to modify these and to calculate the diffraction pattern respectively the Pair Distribution Function. It is of course helpful to visualize a structure. The `plot` menu in DISCUS offers these possibilities.

Predominantly one will write an output file that can be interpreted by appropriate crystal structure visualization programs. While DISCUS offers several output formats, probably the most widely accepted file format is the `CIF` file format. As the `CIF` file format is restricted to a single unit cell, DISCUS will transform the structure into a large unit cell that is an integer multiple of the current unit cell dimensions. To deal with programs that like to wrap around atom positions, the structure is shifted to avoid atom coordinates that are equal to zero or one.

Starting with version `5.22` DISCUS offers the possibility to open JMOL, if JAVA and JMOL have been installed on your computer. On computers with Linux or MacOS, you need to install JAVA and JMOL. The Windows version includes a JMOL executable. You still need to install the Java Run time Environment JRE.

As of versions `6.00` and later, the JAVA installation is done automatically.

To start a JMOL window, set the program to "jmol" and include on the `run` command line the optional parameter `"plot:inter"` . DISCUS will still write a `CIF` file to the hard disk and pass a short script to JMOL to load this `CIF` file.

```
1  discus
2  read
3    cell quartz.cif, 2, 2, 2
4  plot
5    program   jmol
6    outf      quartz_plot.cif
7    extend    all
8    select    all
9    bond      Si, O, 1.2, 1.8, 0.1, 0.0, 0.0, 0.0
10    polyhedra Si, O, nmin:4,nmax:4, dmin:1.2, dmax:1.8, face:flat, hue:solid, color:auto
11    background 230, 230, 230
12    run plot:inter, kill:yes
13  exit
14  exit  ! back to suite
```

In this example, DISCUS reads a `CIF` file of Quartz and expands the structure to 2x2x2 unit cells. An output `CIF` file is written as `quartz_plot.cif`. In order not to overwrite the input `CIF` file, the name is modified to `quartz_plot.cif`. The `bond` command defines bonds from a central Silicon atom to neighboring Oxygen atoms in a distance range from 1.2 to 1.8 Å. The bonds will have a thickness of 0.a Å. The RGB color triplet is currently ignored, as JMOL has a sensible standard color scheme.

The `polyhedra` command tells JMOL to plot polyhedra with the following (optional) properties:

nmin/nmax	Minimum/Maximum number of neighbors an atom must have to be inside a polyhedron.
dmin/dmax	Minimum/Maximum distance to the neighbors. Only neighbors in this range are taken into account to build a polyhedron.
face "flat"|"collapsed"	JMOL can draw a polyhedron with flat surfaces or with indented surfaces.
hue "solid"|"trans"	A transparent polyhedron allows to see the central atom.
color	With "auto" JMOL will take the color of the central atom as template for the surfaces.

In this example the bond distances specified by the `bond` and the `polyhedra` command coincide. As JMOL takes only a single polyhedron definition you can use the additional `bond` command to restrict more clearly, which polyhedra will be drawn.

Finally the background can be set as an RGB triplet.

The optional parameter `plot:inter` initiates the JMOL plot. Omit this or set it to `plot:none` to skip the JMOL plot.

Every time you use the `plot:inter` parameter, a new JMOL window will be started. The optional parameter `kill:yes` allows to close all previous JMOL windows.

2.8 Properties

Within a crystal you will typically have a lot of different atom types. In addition to the type (Fe, Mn, Cl ...) atoms may be distinguished by further properties. An atom might be within a molecule or be a separate atom. DISCUS labels these different properties as logical flags and updates these automatically if any changes occur. Many of the modification and output tool can utilize these properties to include or reject a specific atom. The properties are listed in table 2.6:

Keyword	Description
Normal	The atom is a normal atom or a void.
Molecule	The atom is part of a molecule
Domain	The atom is within a domain.
Outside	The atom is outside the crystal, i.e. on the other side of a limiting surface. Usually these atoms are voids as well.
External	The atom is within a user defined distance to an external boundary, either on the inside or outside.
Internal	The atom is within a user defined distance to an internal boundary, either on the inside or outside. Internal boundaries arise through domains.
Ligand	The Atom is part of a ligand molecule that has been placed on top of the crystal.

Table 2.6: List of property flags

In addition you can define further property like behaviour. This behaviour is based on connectivities 2.9 that have been defined for an atom. These property like featureas allow to restrict

---

the selection of atoms to specific types that have coordination numbers within a user defined range or whose coordination numbers are outside an excluded range of coordination numbers.

2.9 Connectivity

Many of the tools in DISCUS need to reference neighboring atoms around a given atom. It is very time consuming to search throughout the entire crystal if an atom fulfills the desired neighborhood conditions. To avoid these lengthy loops DISCUS provides two styles to reference a neighboring atom.

The first style is the fastest but it depends on the initial storage order of the atoms, which was described in section 2.3. As long as this sequence is maintained DISCUS can reference a neighbor by a vector relationship:

```
1   set vec, 1,  1, 1,   1, 0, 0
2   set vec, 2,  3, 4,   1,-2, 4
```

The first parameter after the `vec` keyword numbers the vector relationships. The next two parameters describe which atom sites with a set of unit cells are related. The last three parameters describe the relative shift of the unit cell in which the second site, the neighbor, is found. Line (1) would thus create a relationship between any atom on site one in a unit cell to the atom on site 1 in a unit cell shifted by [1,0,0]. Line (2) creates a relationship between an atom on site 3 in a unit cell and an atom on site 4 in a unit cell shifted by [1,$\overline{2}$,4] unit cells. These relationships hold for all pairs throughout the crystal. In order to quickly calculate which atom in our internal sequence is referenced, the atoms must still be in their original order.

If you add a lot of atoms, or if you shape the crystal with the boundary commands (see section 6.5) and remove unnecessary outside voids, the internal sequence of atoms my change dramatically. Under these circumstances the vector relationships to not work efficiently any longer. For that reason you will find several warnings with respect to the `purge` command. To allow a fast reference to neighboring atoms DISCUS offers the `connectivity` menu. This menu allows you to define relationships between any pair of atoms through distance intervals:

```
 1   connectivity
 2      reset
 3      add  Zn, O , 1.90, 2.10, zn_o_short
 4      add  Zn, Zn, 3.10, 3.30, zn_zn_second
 5      add  O , Zn, 1.90, 2.10, o_o_short
 6      add  O , O , 3.10, 3.30, o_o_second
 7      create
 8      show
 9   exit
10   ...
11   connectivity
12      recreate Zn, zn_zn_second
13   exit
```

In this example we clear all previous connectivity lists in line (2) and add two connectivities. The first (line 3) relates Zn and O atoms that are within the interval 1.90 to 2.10Å and provides a name `zn_o_short` for this relationship. This relationship is uni directional, Zn is always at the center of the connectivity polyhedron. Thus to group all pairs, lines 5 and 6 add the opposite relationships with the oxygen as central atom.

The `add` command simply creates the definitions, the actual lists are generated with the `create` command. As the program has to perform a double loop over all atom pairs this may take a bit of time, but you will easily regain this time during a Monte-Carlo simulation.

In line 12 we recreate the connectivity `zn_zn_second` around Zinc atoms. This might be necessary, if atoms have been moved around.

Starting with version 5.14 two new optional features modify the build up of a connectivity.

The `first` optional parameter `first:<no>` allows you to limit the number of neighbors that will be included in a connectivity to the value <no>. This will include only the first <no> neighbors that are closest in distance to the central atom. The string "first" is used to indicate that this is an optional parameter.

This option might be needed, if you want to constrain the neighbors around a central atom to a specific shape. You would choose for example `first:4` to limit the connectivity to the four closest neighbors.

The optional parameter `molescope:` allows to restrict the connectivity for atoms that are located within a molecule. Possible options are: `ignore`, `within` and `outside`. With `ignore` the molecular status of the central atom and its neighbors is ignored. Any pair of atoms may be connected based on the distance shell only. With the `within` option, the central atom has to be within a molecule. Only those atoms within the distance shell will be included in the connectivity that are within the same molecule as the central atom. All other atoms are ignored, even if they are within the distance specified by <rmin> and <rmax>. This is useful to constrain a covalent distance or bond angle in `mmc` while ignoring other atoms that may be close by. The final option `outside` acts in an opposite sense. The central atom has to be within a molecule, while the neighbors have to be outside this molecule. A typical use might be to build a connectivity between a Hydrogen atom within a molecule to Oxygen atoms outside the molecule in order to establish a Hydrogen bond.

Chapter 3

Fourier transform

3.1 Calculating scattering intensities

Basically DISCUS calculates the Fourier transform (neutron, X-ray, or electron) according to the standard formula for kinematic scattering given in equation 3.1 or 3.2 in the case of anisoropic atomic displacement parameters.

$$F(\mathbf{h}) = \sum_{j=1}^{N} f_j(\mathbf{h}) e^{2\pi i \mathbf{h} \mathbf{r}_j} \cdot e^{-\frac{B_j |\mathbf{h}|^2}{4}} \tag{3.1}$$

$$F(\mathbf{h}) = \sum_{j=1}^{N} f_j(\mathbf{h}) e^{2\pi i \mathbf{h} \mathbf{r}_j} \cdot e^{-2\pi^2 [h^2 (a^*)^2 U_{11} + k^2 (b^*)^2 U_{22} + ... + 2hka^* b^* U_{12}]} \tag{3.2}$$

The sum is over all N atoms in the crystal, where f_i is the atomic form factor (or scattering length in case of neutrons), $\mathbf{r}_i$ the fractional coordinate of the atom. The sum is calculated at all points $\mathbf{h}$ in reciprocal space. The form factors are tabulated and calculated once at each $\mathbf{h}$ for all species present in the crystal. Optionally the Debye-Waller factor is calculated. Isotropic B, respectivela anisotropic atomic displacement parameters U_{ij} are used for the representation of thermal disorder.

DISCUS offers two algorithms to calculate eqns 3.1 or 3.2. The first algorithm is based on the program *DIFFUSE* (Butler and Welberry, 1992), the second uses the FastFourierTransform based on the concept of a non-uniform FFT Fourmont (1999).

The first realization of the actual code to calculate the Fourier transform is based on the program *DIFFUSE* (Butler and Welberry, 1992). By limiting the calculation to an equidistant grid and splitting the sum into sums over equal atom types, the computing time required dropped by a factor of 4 to 6 (depending on compiler and hardware) compared to calculating the sum given in equation 3.1 in a straight forward way. More details about the algorithm used can be found Butler and Welberry (1992). There are two considerations when calculating the scattering intensities: Unwanted scattering contributions due to the finite size of the model crystal and the fact that the complete model scatters coherently, which leads to unwanted high frequency oscillations in the calculated pattern. Finite size effects are avoided by using a grid of

n/N where N is the number of unit cells. Coherence might be limited by using the `lots` command in the `four` module (see online help). A detailed discussion of these topics can be found in Neder and Proffen (2007).

The Non-uniform FFT algorithm in DISCUS uses the `FINUFFT` library Barnett et al. (2019). For large structures and a dense calculation in reciprocal space the NUFFT algorithm is faster by orders of magnitude.

Let us consider a simple example. Assuming a structure has been read or created, the scattering pattern can be calculated using these commands:

```
 1  four
 2    xray
 3    wvle moa1
 4    ll 0.0,0.0,0.0
 5    lr 2.0,0.0,0.0
 6    ul 0.0,2.0,0.0
 7    na 101
 8    no 101
 9    set aver,0.0
10    lots off
11    set technique:turbo
12    run
13  exit
```

After entering the Fourier segment of DISCUS (line 1), $MoK\alpha$ radiation is selected in lines 2 and 3. Next the lower left, lower right and upper left corner of the desired plane in reciprocal space are specified (lines 4-6). The number of grid points in both directions are set in lines 7 and 8. Finally the subtraction of the average scattering intensity (Bragg peaks) $\langle F \rangle$ is disabled (line 9) and the lot option is switched off (line 10). In this example the Fourier technique has been set to the `turbo` mode rather than the `nufft` mode (line 11). Finally the Fourier transform is calculated (line 12). Note that one needs to save the result in the `output` segment of DISCUS as we will discuss later in this chapter.

Starting with DISCUS 5.0 you can also calculate a 3-dimensional volume in reciprocal space. A third axis is added to the values defined in the previous macro:

```
 1  four
 2    xray
 3    wvle moa1
 4    ll 0.0, 0.0, 0.0
 5    lr 2.0, 0.0, 0.0
 6    ul 0.0, 2.0, 0.0
 7    tl 0.0, 0.0, 2.0
 8    na 101
 9    no 101
10    nt 101
11    set aver,0.0
12    lots off
13    set technique:turbo
14    run
15  exit
```

This third axis defines the `top left` corner of the reciprocal space volume. The corresponding data can be saves as a series of slices parallel to the layer defined by the corners `ll`, `lr`, `ul` or as a 3D file in the NeXus file format or as a series of 2D slices.

Starting with DISCUS 6.06 three new features were added to the Fourier calculation.

The first new feature allows to calculate and merge several diffraction pattern. This allows you to simulate smaller crystals whose individual diffraction might be rather noisy but can be calculated rapidly. The average of all the respective diffraction pattern will give you a smoother final diffraction pattern. Consider the following macro:

```
1  fourier
2    ...                  ! Define all relevant Fourier settings
3    set mode:init        ! Initialize the sum of Fourier calculations
4  exit
5  do LOOP=1, 5           ! A loop over five simulations
6    @build_a_nice_crystal.mac   ! Details omitted, just some nice
7                                 ! simulation with local disorder
8    fourier
9      set mode:accumulate       ! Accumulate all individual calculations
10     run
11   exit
12 enddo
13 fourier
14    set mode:finish            ! We are done with the accumulation
15 exit
```

The command `set mode:init` initializes the accumulation of several Fourier calculations. All prior calculations will be lost. To go back to the single mode use the Fourier command `set mode:single`. Within the loop the individual Fourier transforms are accumulated into temporary storage as indicated by the `set mode:accumulate` command. Once the loop is finished, all pattern are merged with the final Fourrier command `set mode:finish`. The merged pattern is now ready to be written using the `output` menu.

The second new feature is the option to convolute the Fourier transform with a Gaussian resolution function.

```
1  fourier
2    ...                  ! Define all relevant Fourier settings
3    run  [ sigabs:[0.01, 1, 0, 0], sigord:[0.01, 0,1,0], sigtop:[0.01, 0,0,1]
4  exit
```

DISCUS allows to define a general triaxial ellipsoid. In the first example above, the sigma is 0.01Å^{-1} along all three axes. With:

```
1  fourier
2    ...                   ! Define all relevant Fourier settings
3    run  [ sigabs:[0.005, 1.0, 1.0, 0], sigord:[0.02, -1.0,1.0,0], sigtop:[0.01, 0,0,1]
4  exit
```

three different half axes have been specified. Along reciprocal space direction [110] a sigma of 0.005Å^{-1}, along [$\bar{1}$10] a sigma of 0.02Å^{-1} and finally along [001] a sigma of 0.01Å^{-1} is used. As pointed out in this second example, the three half axes of the ellipsoid may point in arbitrary reciprocal space directions. They do not even have to be at mutual $90°$ to each other.

The third enhancement involves symmetry averaging. If requested, the crystals point group symmetry is used to average the calculated diffraction pattern. Currently the abscissa, ordinate and top axes must be parallel to a*, b* and c* respectively. The center need not be at the origin of reciprocal space. The average will work most evenly, however, if the center of the calculated data is at the origin.

3.2   Powder diffraction

In addition to the calculation of single crystal scattering intensities, DISCUS can also computer a powder diffraction pattern of a given structure. There are two ways DISCUS can calculate powder intensities: One is a complete integration of reciprocal space which is then mapped onto the desired range between $2\Theta_{min}$ and $2\Theta_{max}$. The second algorithm uses the Debye scattering equation to calculate the powder pattern. More details as well as examples are given in our book (Neder and Proffen, 2007). Information about the related commands are found in the online help using the command `help powder`.

The next two sections give a quick overview of the two different calculation strategies, followed by general information on the usual Lorentz, Polarisation corrections and the profile functions. The last subsections give examples.

3.2.1   Complete integration

A complete integration relies on the calculation of the complete reciprocal space and subsequent spherical averaging onto a linear Q-scale. This essentially mimics the actual experiment. In the experiment you actually measure teh spherically averaged single crystal diffraction pattern of many individual small crystals. Averaging is performed by the orientational average of all individual grains. This may be assisted by rotating the sample around one or two axes.

Computationally this intewgration will be an expensive task. Its application will likely be restricted to two important cases:

- Materials with stacking faults. Diffuse scattering is usually limited to a discrete set of rods with continuous intensity along the reciprocal space direction normal to the individual layers.

- Materials with one dimensional disorder. For these materials diffuse scattering is limited to 2D layers in reciprocal space.

3.2.2   Debye-Scattering-Equation

The Debye-Scattering-Equation results if one performs a spherical average of the equation for the intensity. It assumes a perfectly random orientation distribution of the individual finite sized objects.

The intensity is nominally calculated as:

$$I(h) = \sum_{i=1}^{N} \sum_{j=1}^{N} f_i f_j \frac{\sin(2\pi h r_{ij})}{2\pi h r_{ij}} \tag{3.3}$$

The double sum is performed over all N atoms in the sample. Here f_i and f_j are the atomic form factors for x-rays or electrons respectively the scattering lengths for neutrons. `h` is the length of the reciprocal space vector:

$$h = 1/d^* = Q/2\pi = \frac{2 sin(\Theta)}{\lambda} \tag{3.4}$$

and r_{ij} is the length of the vector between atoms i and j.
As this equation stands, it is extremely time consuming. Within DISCUS several measures have been performed to reduce the calculation time.

3.2.3   General powder corrections

DISCUS modifies the calculated intensities by a Lorentz and Polarisation correction as well as a convolution by a profile function.

$$I_{obs} = (L \cdot P \cdot I_{calc}) \otimes Profile \tag{3.5}$$

The **Lorentz** corrections that are available are:
Bragg-Brentano Geometry:

$$L = \frac{1}{sin(\Theta)sin(2\Theta)} \tag{3.6}$$

Neutron Time-of-Flight instrument:

$$L = d^4 sin(\Theta) \tag{3.7}$$

where Θ is half the 2-Theta angle of the detector bank central point.

The **Polarisation** corrections that are available for X-ray diffraction data are:
Bragg-Brentano Geometry:

$$P = 1 + \frac{1}{2}cos^2(2\Theta)cos^2(2\Theta_M) \tag{3.8}$$

If the instrument does not use a Monochromator or analysor, set the monochromator angle to zero.A
Synchrotron radiation

$$P = K + (1 - K)cos^2(2\Theta)cos(2\Theta_M) \tag{3.9}$$

If required, the data can be convoluted with a profile function. At the moment the available profile functions are :
Pseudo-Voigt function:

$$Profile(Q) = (1 - \eta)L + \eta G \tag{3.10}$$

DISCUS allows to modify η as function of Q, respectively 2Theta:

$$\eta_{(Q)} = \eta_0 + \eta_1 \cdot Q + \eta_2 \cdot Q^2 \tag{3.11}$$

The common FWHM can be modified as well, using the Cagliotti formalism:

$$FWHM_Q^2 = u \cdot tan^2(2\Theta) + v \cdot tan(2\Theta) + w \tag{3.12}$$

If the data are calculated on a Q-scale (the standard mode), DISCUS will convert the uvw values accordingly.

A purely empirical form to calculate the FWHM on the Q scale uses a polynomial of Q:

$$FWHM_Q^2 = u \cdot Q^2 + v \cdot Q + w \tag{3.13}$$

As an alternative the FWHM can be calculated as

$$FWHM_Q^2 = u \cdot \cos^4(2\Theta) + v \cdot \cos^2(2\Theta) + w \tag{3.14}$$

This equation was derived by Chernyshov et al. (2021) for powder data collected with an area detector.

An asymmetry parameter allows to use 2Theta dependent asymmetric profiles. It is calculated as in Fullprof, except that all asymmetry parameters may have a 2Theta, respectively Q dependence.

The user can set these within the powder menu via the `set profile` commands as illustrated in the following macro section:

```
set profile, pseudo
set profile, eta, eta, P_eta, P_eta_l, P_eta_q
set profile, uvw, P_u, P_v, P_w                  ! Defaults to polynomial on Q
set profile, uvw, P_u, P_v, P_w, fwhm:cagliotti  ! Use Cagliotty equation
set profile, uvw, P_u, P_v, P_w, fwhm:poly       ! Use polynomial on Q
set profile, uvw, P_u, P_v, P_w, fwhm:area       ! Use area detector equation
set profile, asym,    P_a1,    P_b1,    P_a1,    P_b2
set profile, asym_l,  P_a1_l,  P_b1_l,  P_a1_l,  P_b2_l
set profile, asym_q,  P_a1_q,  P_b1_q,  P_a1_q,  P_b2_q
set profile, asym_i1, P_a1_i1, P_b1_i1, P_a1_i1, P_b2_i1
set profile, width, 20
```

In the macro above, all terms P_* refer to a user defined real valued variable. The variables ending in $\_l$ refer to the linear Q dependence, the ones $\_q$ to a square Q dependence and the $\_i1$ to a dependence on Q^{-1}.

Pearson type VII function:

$$Profile(Q) = \frac{1}{\alpha B\left(m - \frac{1}{2}, \frac{1}{2}\right)}\left[1 + \left(\frac{Q - Q_0}{\alpha}\right)^2\right]^{-m} \tag{3.15}$$

Here m is the shape control parameter, and should be larger than 1.

Parameter α is calculated from the user provided FWHM as :

$$\alpha = \frac{FWHM}{2\sqrt{2^{1/m} - 1}} \tag{3.16}$$

B is the Beta function:

$$B(z_1, z_2) = \frac{\Gamma(z_1)\Gamma(z_2)}{\Gamma(z_1 + z_2)} \tag{3.17}$$

with Γ the `Gamma` function :

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt \tag{3.18}$$

where the real part of the complex number z is larger than zero.

The Q dependencies of m and FWHM are calculated as for the Pseudo-Voigt profile function. Use eta as place holder for m.

Neutron Time-of-Flight function:

This profile function is calculated as in GSAS-II Toby and von Dreele (2013). The main profile equation is:

$$
\begin{aligned}
Profile(\Delta T) \quad = \quad & (1-\eta)N\left[e^u erfc(y) + e^v erfc(z)\right] - \\
& \tfrac{2N\eta}{\pi}\left\{Im\left[e^p E_1(p)\right] + Im\left[e^q E_1(q)\right]\right\}
\end{aligned}
\tag{3.19}
$$

Here *erfc* is the complementary error function, and $Im\left[e^p E_1(p)\right]$ is the imaginary part of the complex exponential of p times the exponential integral of p. The four coefficients u,y and v, z are defined as :

$$
\begin{aligned}
u \quad &= \quad \tfrac{\alpha}{2}\left(\tfrac{\alpha\Gamma^2}{8ln2} + 2\Delta T\right) \\
v \quad &= \quad \tfrac{\beta}{2}\left(\tfrac{\beta\Gamma^2}{8ln2} - 2\Delta T\right) \\
y \quad &= \quad \frac{\frac{\alpha\Gamma^2}{8ln2} + \Delta T}{\sqrt{2\sigma^2}} \\
z \quad &= \quad \frac{\frac{\beta\Gamma^2}{8ln2} - \Delta T}{\sqrt{2\sigma^2}}
\end{aligned}
\tag{3.20}
$$

Parameters p and q are complex numbers:

$$
\begin{aligned}
p \quad &= \quad -\alpha\Delta T + \tfrac{i\alpha\gamma}{2} \\
q \quad &= \quad -\beta\Delta T + \tfrac{i\beta\gamma}{2}
\end{aligned}
\tag{3.21}
$$

The internal parameters α, β, σ in turn are:

$$
\begin{aligned}
\alpha \quad &= \quad \alpha_0 + \tfrac{\alpha_1}{d} \\
\beta \quad &= \quad \beta_0 + \tfrac{\beta_1}{d^4} + \tfrac{\beta_Q}{d^2} \\
\sigma \quad &= \quad \sigma_0^2 + \sigma_1^2 d^2 + \sigma_2^2 d^4 + \sigma_Q d \\
\gamma \quad &= \quad Z + Yd + Xd^2 + S_{gamma} + M_{gamma} \\
S_{gamma} \quad &= \quad 1.0 \cdot 10^{-4} \cdot DIFC \cdot d^2/size \\
M_{gamma} \quad &= \quad 1.0 \cdot 10^{-6} \cdot DIFC \cdot d \cdot strain
\end{aligned}
\tag{3.22}
$$

Note that the terms Z, Y, X are also referred to as $\gamma_0, \gamma_1, \gamma_2$.

The mixing parameter η is calculated by the following set of equations:

$$\eta \quad = \quad 1.36603(\gamma/\Gamma) - 0.47719(\gamma/\Gamma)^2 + 0.11116(\gamma/\Gamma)^3$$

$$\Gamma \quad = \quad \sqrt[5]{\Gamma_g^5 + 2.69269\Gamma_g^4\gamma + 2.42843\Gamma_g^3\gamma^2 + 4.47163\Gamma_g^2\gamma^3 + 0.07842\Gamma_g\gamma^4 + \gamma^5} \quad (3.23)$$

$$\Gamma_g \quad = \quad \sqrt{(8ln2)\sigma^2}$$

The normalization factor N is:

$$N = \frac{\alpha\beta}{2(\alpha + \beta} \tag{3.24}$$

The relationship between the d-spacing and the time-of-flight is:

$$T = DIFC \cdot d + DIFA \cdot d^2 + ZERO + DIFFB/d \tag{3.25}$$

The time is used in units of μs.

This finally gives the user parameters in table 3.1:

The user can set these within the powder menu via the `set profile` commands as illustrated in the following macro section:

```
set profile, tof
set profile, bank, difc:5608.82, difa:-0.960, difb:0.0000, tzero:3.3800
set profile, tab, a0:0.0, a1:0.99198, b0:3.340098, b1:10.0e-5, bq:0.0
set profile, tsig, s0:0.0, s1:620.716, s2:43.864, sq:0.0
set profile, tgam, z:0.0, y:22.2684, x:0.0, size:10.0, strain:0.0
```

3.2.4 Powder examples

In this section two examples to calculate powder pattern are presented. The first example uses the complete integration, the second the Debye-scattering-equation.

This first example illustrates the calculation of a powder pattern for a perfect crytalline sample. Since we want to simulate a powder pattern of a perfect crystal, it is actually sufficient to read just a single unit cell, see lines 3 and 4 in the macro.

Within the powder menu (line 6) the usual DISCUS philosophy is used to define all the necessary parameters and then launch the calculation with `run` (line 27). After a quick `reset` to ensure that we are back at standard conditions, the radiation is chosen with `xray`. Alternatively `neutron` or `electron` are available.

The next command is essentially obsolete, DISCUS always runs the internal calculation on a Q-scale (line 9).

The next series of `set` commands defines the required parameters. We start off by choosing the "`complete` integration mode, see the next example for the `deby` mode.

In this example the limits and step width are set on a Q-scale (lines 11-13), alternatively 2Θ limits could be used equivalently.

Reciprocal space will be empty except for the Bragg reflections thus we can limit the integration to these discrete points (lines 14-16).

Name	Bank 3	Bank 4	Bank 5	Units
α_0	0.00	0.00	0.00	μs^{-1}
α_1	0.99198	0.122511	0.094346	$\text{Å}\mu s^{-1}$
β_0	3.340098	0.05864	0.020882	μs^{-1}
β_1	$1.0 \cdot 10^{-5}$	0.095311	0.034961	$\text{Å}^4 \mu s^{-1}$
β_Q	0.00	0.00	0.00	$\text{Å}^2 \mu s^{-1}$
σ_0^2	0.00	0.00	2.97185	μs^2
σ_1^2	620.716	43.864	10.00	$\text{Å}^{-2} \mu s^2$
σ_2^2	176.425	122.734	12.9386	$\text{Å}^{-4} \mu s^2$
σ_Q	0.00	0.00	0.00	$\text{Å}^{-1} \mu s^2$
Z	0.00	1.81264	0.00	μs^2
Y	22.2684	21.4362	9.03932	$\text{Å}^{-1} \mu s^2$
X	0.00	4.32645	4.84055	$\text{Å}^{-2} \mu s^2$
size	10.00	0.00	0.00	μm
strain	0.00	0.00	0.00	
DIFC	5608.82	9041.78	9892.73	$\text{Å}^{-1} \mu s$
DIFA	-0.960	-1.350	-5.890	$\text{Å}^{-2} \mu s$
ZERO	3.380	-0.860	6.970	μs
DIFB	0.00	0.00	0.00	$\text{Å}\mu s$

Table 3.1: User parameters for the neutron TOF profile. Numerical values are rough examples for NO-MAD at SNS

Lines 17 to 21 define the profile function. The recommended style is the `set profile` command. The `set delta` command should no longer be used except with 0.00. The simple Gaus-

sian profile function with constant FWHM can be equivalently created with the `set profile` and $\eta = 0.0$. Here we use a Pseudo-Voigt function with constant FWHM.

This simple example does not use anomalous dispersion (line 22). The temperature factor is used, as should be default (line 23).

Insted of the wave length in Ångstrom, element names like `Cu` could be used for laboratory xray sources. Alternatively the energy can be specified with `set energy`, `<value>`, with `<value>` to be replaced by the appropriate numerical value or expression.

If the crystal is build using the stacking fault menu, the calculation of a diffraction pattern can be speed up, see the stacking fault chapter for details. Here we use the regulr Fourier mode (line 25).

As decribed in 3.2.3 Lorentz/Polarisation corrections are applied (line 26).

The `show` command, common to most DISCUS menus is omitted and the calculation is started with the usual `run` command.

The calculated powder pattern is written to disks within the `output` menu, lines(30 to 39). The `value` command specifies the value to be written. See table 3.4 for a full list of available choices. The `form` must be `powder`, with a choice of Q-scale (line 33) or 2Θ scale (line 37). For both forms, the range can optionally be limited.
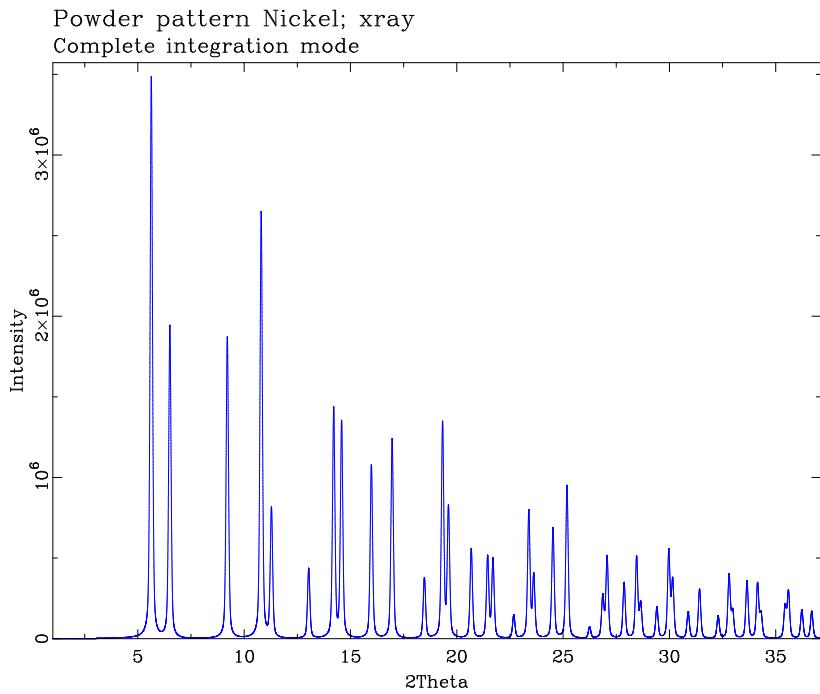
```
 1 discus
 2 #
 3 read
 4   cell nickel.cell
 5 #
 6 powder                        # Switch to powder menu
 7   reset
 8   xray                        # Select X-rays
 9   set axis,q                  # Perform calculation on equaly spaced Q grid
10   set calc,complete           # Use complete integration algorithm-algorithm
11   set qmin,0.550              # Starting value for Q
12   set qmax,20.00              # Final value for Q
13   set dq,  0.005              # Step size for Q
14   set dh, 1.0
15   set dk, 1.0
16   set dl, 1.0
17   set delta,0.0               # Set simple convolution by Gaussian off
18   set profile, pseudo         # Use Pseudovoigt (instead of Gaussian)
19   set profile, uvw, 0.000, 0.000, 0.005, fwhm:cagliotti # Cagliotti u,v,w values
20   set profile, eta, 0.500     # Mixing parameter 1=Lorenzian 0=Gaussian
21   set profile, asym, 0.00, 0.00, 0.00, 0.00 # Asymmetry parameters
22   set disp,off                # Switch anomalous dispersion off
23   set temp,use                # Use the Atomic displacement parameters
24   set wvle,0.20000            # Set the wavelength
25   set four,four               # Just for stacking fault mode, set to normal
26   set lpcor,bragg, 0.00       # Define Lorentz-Pol to BraggBrentano Diffractometer
27   run                         # Do the actual calculation
28 exit                         # Go back to main DISCUS menu
29 #
30 output                        # Switch to output menu
31   outf  complete.inte         # Define output file name
32   value inte                  # Select intensity as output value
33   form  powder,q, 0.,20.0,0.001 # Write output as powder data, limit Q-range
34   run                         # Perform the actual output
35   outf  complete.tth          # Define output file name
36   value inte                  # Select intensity as output value
37   form  powder,tth            # Write output as powder data
38   run                         # Perform the actual output
39 exit                         # Go back to main DISCUS menu
```

```
40 exit                           # Go back to the SUITE
```

Fig 3.1 shows the calculated powder pattern on a 2Θ scale.

Figure 3.1: Example of a powder pattern

For the example using the Debye-Scattering-Equation we will create a simple spherical finite sized crystal and calculate its powder pattern.

The macro expands a Nickel unit cell to 12x12x12 unit cells (lines 3 and 4), sufficiently large for the 35 Ångstrom sphere. A more sophisticated algorithm would calculate teh necessary number of unit cells from the intended diameter, something like `n = diameter/lattice_parameter`. The surface menu (lines 5 to 7) uses the `boundary` command to create this `sphere` with 17.5 Ångstrom radius. Since the DSE is not super fast, it is a good idea to clean away the `Voids` created by this command (line 8).

The remainder of the macro is identical except for line 15, which sets the calculation mode to `debye`.

```
1 discus
2 #
3 read
4   cell nickel.cell, 12, 12, 12
5 surface
6   boundary sphere, 17.5
7 exit
8 purge
9 #
0 #
1 powder                          # Switch to powder menu
2   reset
3   xray                          # Select X-rays
4   set axis,q                    # Perform calculation on equally spaced Q grid
```

```
15   set calc,debye                # Use Debye-Scattering-Equation
16   set qmin,0.550                # Starting value for Q
17   set qmax,20.00                # Final value for Q
18   set dq,  0.005                # Step size for Q
19   set dh, 1.0
20   set dk, 1.0
21   set dl, 1.0
22   set delta,0.0                 # Set simple convolution by Gaussian off
23   set profile, pseudo           # Use Pseudovoigt (instead of Gaussian)
24   set profile, uvw, 0.000, 0.000, 0.005, fwhm:cagliotti # Cagliotti u,v,w values
25   set profile, eta, 0.500       # Mixing parameter 1=Lorenzian 0=Gaussian
26   set profile, asym, 0.00, 0.00, 0.00, 0.00 # Asymmetry parameters
27   set disp,off                  # Switch anomalous dispersion off
28   set temp,use                  # Use the Atomic displacement parameters
29   set wvle,0.20000              # Set the wavelength
30   set four,four                 # Just for stacking fault mode, set to normal
31   set lpcor,bragg, 0.00         # Define Lorentz-Pol to BraggBrentano Diffractometer
32   run                           # Do the actual calculation
33 exit                            # Go back to main DISCUS menu
34 #
35 output                          # Switch to output menu
36   outf  debye.inte              # Define output file name
37   value inte                    # Select intensity as output value
38   form  powder,q, 0.,20.0,0.001 # Write output as powder data, limit Q-range
39   run                           # Perform the actual output
40   outf  debye.tth               # Define output file name
41   value inte                    # Select intensity as output value
42   form  powder,tth              # Write output as powder data
43   run                           # Perform the actual output
44 exit                            # Go back to main DISCUS menu
45 exit                            # Go back to the SUITE
```

Fig 3.3 shows the calculated powder pattern on a 2Θ scale. Note the small angle scattering below 3°. Since the pattern was calculated for a single perfect sphere, the finite size oscillations are clearly visible within the small angle region.

3.3   Fourier methods

Additional to the Fourier transform of a real space structure, three other Fourier transforms are available in the current version of DISCUS : *difference Fourier, inverse Fourier* and calculation of the *Patterson* (Fourier transform of scattering intensities). All three of these Fourier transforms use the following equation:

$$\rho(\mathbf{r}) = \frac{1}{V} \sum_{i=1}^{N} F_i(\mathbf{h}) e^{-2\pi i \mathbf{h} \mathbf{r}_i} \tag{3.26}$$

The resulting density function $\rho(\mathbf{r})$ and the Fourier coefficient $F(\mathbf{h})$ take different meanings depending on the intended inverse Fourier transformation listed in table 3.2. The Fourier coefficient are in general complex numbers. Therefore, DISCUS requires in most cases two input files for the inverse Fourier transforms. The allowed combinations are listed in the help file and the command reference.

Powder diffraction pattern Nickel; xray
Debye-Scatering-Equation; 35Å sphere
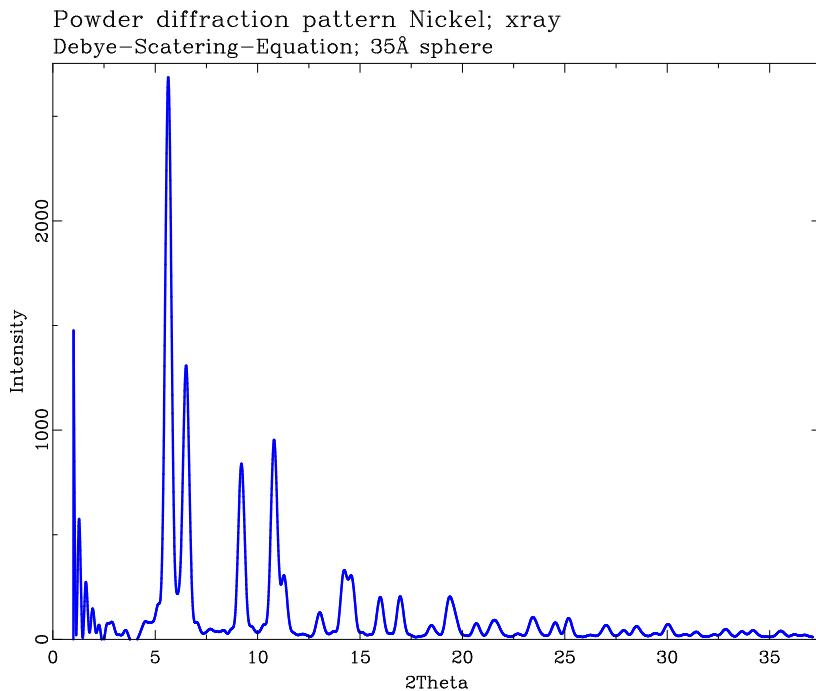
Figure 3.2: Example of a powder pattern

Command	F(h)	ρ(r)
diff	$F_{obs} - F_{calc}$	difference scattering density
inv	F_{obs}	scattering density
patt	I_{obs}	Patterson density

Table 3.2: List of available Fourier methods

3.4 Output file formats

All results of the Fourier transforms are written at the `output` segment of DISCUS . The following values can be saved after you have calculated a single crystal diffraction pattern:

intensity, amplitude, real part, imaginary part, phase angle (in degrees) and the 3D PDF

A macro to write such a single crystal pattern could be :

```
output
  form stan
  value inte
  outf  pattern.inte
  run
exit
```

Especially for large 3D volumes the HDF5 file format might be advisable.
Following a powder diffraction pattern calculation, you can write:

intensity, S(Q), F(Q), $< f^2 >$, $< f >^2$ and the powder PDF

A simple macro to write such a powder diffraction pattern could be :

```
output
  form powder, q
  value inte
  outf  powder_pattern.inte
  run
exit
```

3.4.1   Single crystal formats

The single crystal Fourier transform calculates the real and imaginary part, all other values
are calculated at the time of output. If the average structure factor $\langle F \rangle$ was subtracted during
the calculation of the Fourier transform, the corresponding values of $\langle F \rangle$ can be saved as well.
Note that if `lots` were used to calculate the Fourier transform, only the intensity values can
be saved to a file. There is no need to calculate the Fourier again, if several of the values
listed are to be written to file. Just define a new output value and output file name and run the
output again. The desired output format is selected using the `format` command. The available
output formats are listed in the table 3.3. The standard and gnu output formats write the actual
numbers calculated by the Fourier transform. The KUPLOT or standard file format is defined
as:

```
1       na,no
2       xmin,xmax,ymin,ymax
3ff     z,z,z,z,z
```

The values 'na' and 'no' given in the first line define the size of the section and are identical
to the values given for the 'na' and 'no' commands at the Fourier segment. KUPLOT uses the
coordinates along the abscissa and ordinate to scale the resulting picture. Since only two pairs
of coordinates are read, the user has to define the necessary indices. 'xmin' and 'ymin' are the is
the 'x' and 'y' coordinates of the lower left corner in reciprocal space, 'xmax' the 'x' coordinate
of the lower right corner and 'ymax' the 'y' coordinate of the upper left corner. Which of the
indices, h,k or l is interpreted as 'x' and 'y' coordinate depends on the values given for the
'abs' and 'ord' commands at Fourier sub level. If for example the $(hkl) * [1\bar{1}0] = 0.0$ layer is
calculated, suitable values would be 'abs h' and 'ord l'. Now the h index of the lower left corner
is written as 'xmin', and the l index as 'ymin'. The values are written row by row, each row
consisting of the values along the abscissa. An empty line separates the rows in the output file.
The it HDF5 output is a binary file written with the help of the HDF5 library. Currently the
format is compatible to the data read by **Yell**, the 3D-PDF program by Arkadiy Simonov. The
KUPLOT section can read these files and step through the 3D-data.
Two file formats are intended to visualize 3D data with external programs such as Vesta and
Chimera. For the Vesta program, DISCUS writes two files, the `base.vesta` and the `base.grd`
files. The first contains display setting read by Vesta, while the second file is an ASCII 3D
bitap of the scattering data. Likewise, the `base.mrc` data are 3D data which can be read and
displayed for example with Chimera.

Output format	Description
`stan`	Output is saved as real numbers in a format suitable for KUPLOT which is part of the DISCUS program package.
`hdf5`	Create output in an HDF5 file format.
`vesta`	Create output in an ASCII file format suitable for input into the VESTA programm.
`mrc`	Create output in the MRC file format.
`nexus`	Output is saved in the NeXus file format. See nexus.org for information on the NeXus file format and file viewers.
`gnu`	Real numbers suitable for the program *GNUPLOT* which can be obtained on most software archive sites.
`pgm, ppm`	Integer bitmaps in PNM format as defined by Jef Poskanzer (gray scale and colour). Various programs are capable of reading PNM files and the *pnmplus-package* is a freely available collection of tools and conversion programs from PNM to virtually any other graphics format.
`post`	Creates a Postscript bitmap suitable for direct printing or to be imported by other programs.

Table 3.3: Output formats for single crystal scattering intensities

As bitmaps for 3D-PDF calculations can be large, DISCUS offers the option to write only a reduced section of the output into these bitmaps. The `range` command has several optional parameters to place and restrict the output range.

```
1  output
2    ...
3    range full     ! write full range of the bitmap
4    range center:middle, pixel:[41, 51, 61]
5    range center:[101,131,51], pixel:[41, 51, 61]
6    range quad:llr
```

With line 3, the full range of the bitmap would be written into the output file. Line 4 defines the center of the section at the middle of the complete data set and writes a bitmap of full width with [41, 51, 61] pixels along the three axes. With a command like in line 6 the center of the data is placed at the specified coordinate. Finally the option presented in line 6 will write a quadrant/octant of the data. These will always include the center of the complete data set and one of the four/eight possible quadrants/octants. In the example along the first two axes the

left half is chosen and along the third axis the right half.

The *GNUPLOT* output is written row by row, each data point within each row in its own line of output. The format of one such row is: "$h_1, h_2, z - value, h_3$". Again the sequence of indices h,k and l depends on the values given for the commands `abs` and `ord` at Fourier sub menu. For the bitmap output the calculated value is scaled linearly to values between 0 and 255. All values less than a definable threshold using the 'thresh' command are set to zero, all values above a maximum threshold are set to 255. Again, the bitmap is written row by row. A color map included in file *color.map* is used to attribute colors to the output values.

3.4.2 Single crystal output with transformations

To avoid finite size Fourier ripples, it is usually advisable to calculate the diffraction pattern along axes that stand normal to the terminating facets. The easiest way to achieve this is to build a block shaped crystal that consists of MxNxO unit cells. Single crystal diffraction should then be calculated along a*, b* and c*.

To compare this block in reciprocal space with experimental data, or to analyze the distribution of diffuse intensity it might often be desirable to view the data along a different axis. This task can be achieved with the optional `trans:yes` parameter on the `output run` command line. If this optional parameter is provided, a new transformation menu is entered. Within this menu you can define the new view direction.

Three different combinations seem the likeliest to define a new viewing direction:

- Define a zone axis and an abscissa

- Define new abscissa, ordinate and top axes

- Define new corners in reciprocal space

These cases will be illustrated by examples. Each example assumes that reciprocal space has been calculated along a*, b* and c* as for example:

```
 1  fourier
 2  ll -3.0,-3.0, -3.0
 3  lr  3.0,-3.0, -3.0
 4  ul -3.0, 3.0, -3.0
 5  tl -3.0,-3.0,  3.0
 6  na  241
 7  no  241
 8  nt  241
 9  abs  h
10  ord  k
11  top  l
12  temp use
13  disp off
14  run
15  exit
```

All examples will set a viewing direction along the [111] axis.

- Define a zone axis and an abscissa
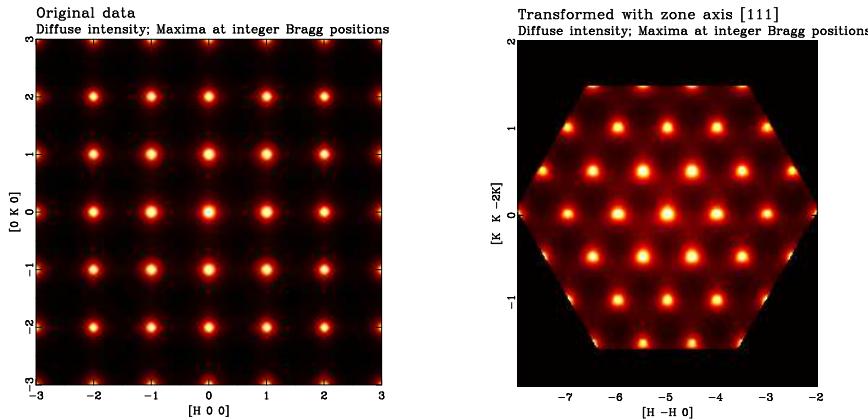
```
 1   output
 2   form hdf5
 3   outfile   example.h5
 4   run trans:yes
 5     reset
 6     set abs:[0.04, -0.04, 0.00], zone:[1,1,1]
 7     set npoints:[251, 251, 251]
 8     run
 9     exit  ! Leave transformation menu; back to output menu
10   exit     ! Leave output menu
```

The `run trans:yes` command in line 4 instructs DISCUS to enter the transformation menu. After a `reset` to ensure clear conditions, line 6 defines a zone axis, here [uvw]=[111] and an increment vector along the abscissa. The abscissa in this example is parallel to the reciprocal vector [hkl]=[110].

If line 7 in the macro is omitted, The overall size in reciprocal space is determined automatically and encompasses the original block in reciprocal space. Data points outside the original block will be set to zero.

To restrict the new reciprocal volume to a smaller or to expand it to a larger volume, the optional parameter `npoints:[251, 251, 251]` can be used. Within the transformation menu you can place this optional parameter either with other optional parameters onto a single `set` line or onto a `set npoints:[251,251,251]` line by itself. The numbers `251` are, of course just an example, feel free to vary as needed.

Figure 3.3: Example of an output transformation. Left original with [uvw]*[hkl]=[001]*[hkl]=0; right [111]*[hkl]=0. To emphasize the surounding zeros the plot has been drawn at slightly lower contour levels.

- Define new abscissa, ordinate and top axes

Essentially the identical transformation is achieved with:

```
 1   output
 2   form hdf5
 3   outfile   example.h5
 4   run trans:yes
 5     reset
 6     set abs:[0.04, -0.04, 0.00], ord:[0.04, 0.04,-0.08], top:[0.04, 0.04, 0.04]
```

```
 7    set npoints:[251, 251, 251]
 8    run
 9    exit  ! Leave transformation menu; back to output menu
10  exit    ! Leave output menu
```

Instead of a zone axis, explicit increment vectors along the abscissa, ordinate and top axis are defined.

- Define new corners in reciprocal space

Finally, if needed four corners in reciprocal space can be defined, in a similar fashion as in the `fourier` menu:

```
 1  variable real, corners,dim:[3,4]
 2  #              left          lower        bottom
 3  corners[1,1] = -0.04*125  -  0.02*125  -  0.04*125
 4  corners[2,1] = +0.04*125  -  0.02*125  -  0.04*125
 5  corners[3,1] =  0.00*125  +  0.04*125  -  0.04*125
 6  #
 7  #              right         lower        bottom
 8  corners[1,2] = +0.04*125  -  0.02*125  -  0.04*125
 9  corners[2,2] = -0.04*125  -  0.02*125  -  0.04*125
10  corners[3,2] =  0.00*125  +  0.04*125  -  0.04*125
11  #
12  #              left          upper        bottom
13  corners[1,3] = -0.04*125  +  0.02*125  -  0.04*125
14  corners[2,3] = +0.04*125  +  0.02*125  -  0.04*125
15  corners[3,3] =  0.00*125  -  0.04*125  -  0.04*125
16  #
17  #              left          lower        top
18  corners[1,4] = -0.04*125  -  0.02*125  +  0.04*125
19  corners[2,4] = +0.04*125  -  0.02*125  +  0.04*125
20  corners[3,4] =  0.00*125  +  0.04*125  +  0.04*125
21  #
22  #  left  lower  bottom  corner :   -12.5000    -2.5000     0.0000
23  #  right lower  bottom  corner :    -2.5000    12.5000     0.0000
24  #  left  upper  bottom  corner :    -7.5000     2.5000   -10.0000
25  #  left  lower  top     corner :    -2.5000     7.5000    10.0000
26  #
27  echo " ll  %5.1f  %5.1f  %5.1f ", corners[1,1], corners[2,1], corners[3,1]
28  echo " lr  %5.1f  %5.1f  %5.1f ", corners[1,2], corners[2,2], corners[3,2]
29  echo " ul  %5.1f  %5.1f  %5.1f ", corners[1,3], corners[2,3], corners[3,3]
30  echo " tl  %5.1f  %5.1f  %5.1f ", corners[1,4], corners[2,4], corners[3,4]
31  output
32  form hdf5
33  outfile  example.h5
34  run trans:yes
35  reset
36  set ll:[corners[1,1], corners[2,1], corners[3,1]], lr:[corners[1,2], corners[2,2], corners[3,2]]
37  set ul:[corners[1,3], corners[2,3], corners[3,3]], tl:[corners[1,4], corners[2,4], corners[3,4]]
38  set npoints:[251,251, 251]
39  run
40  exit
41  exit    ! Leave output menu
```

As the macro examplifies, in a general view direction it might be easier to define the corners as vector sum of the abscissa, ordinate and top vectors. Lines 22 to 25 summarize the effective corners that are set in this example.

3.4.3 Powder diffraction formats

Table 3.5 shows the output options for powder diffraction data. Possible values were listed at the beginning of this section.

Output format	Description
`powder`	Output is saved as powder diffraction data in a simple two column ASCII file. Data can be written on a 2Theta of Q scale as defined by the first parameter. Optionally the range of data along the chosen scale can be limited. These settings are independent of the scale on which the data were calculated.
`pdf`	The powder diffraction data are converted into the 1D-powder PDF. DISCUS takes care of the scaling and normalization.

Table 3.4: Output values for powder diffraction data.

For a powder pattern you can write the regular intensity, normalized intensities as listed in table 3.5. Note that the output values `I(Q), S(Q), F(Q)` must be typed as listed, with capital letters and including the parentheses.

Powder output values	Description
`inte`	Regular powder pattern intensity.
`I(Q)`	Normalized powder pattern intensity.
`S(Q)`	Normalized powder scattering function.
`F(Q)`	Reduced normalized powder scattering function.
`faver2`	Average atomic form factors, squared. $\langle f \rangle^2$
`f2aver`	Average squared atomic form factors. $\langle f^2 \rangle$

Table 3.5: Output formats for powder diffraction data.

A simple macro to write a 1D-powder PDF pattern could be :

```
output
  form pdf, r, 0.01, 100.0, 0.01
  value PDF
  outf  calculated.grobs
  run
exit
```

This macro will instruct DISCUS to convert the calculated powder pattern into the PDF. The PDF will be written in the distance range 0.0Å, to 100.0Å in steps of 0.01Å. The parameters `pdf, r` are mandatory, the distance range is optional and defaults to the range as given in this example.

Chapter 4

Crystallography

In this chapter crystallographic calculations such as conversion from direct to reciprocal space and *vice versa* are discussed as well as symmetry operations and unit cell transformations. The mathematical background of the described calculations can be found e.g. in Sands (1995).

4.1 Crystallographic calculations

In order to determine the proper place of a new atom, or to check existing atoms several functions perform calculations that are based on the metric of the crystal. Several intrinsic functions are provided by DISCUS. They are listed in Table 1.4 in section 1.2 of this manual. Those function allow the user to calculate e.g. bond length and bond angles within the current crystal.

Command	Description
d2r	Conversion from direct (real) space to reciprocal space
r2d	Conversion from reciprocal space to direct space
vprod	Vector product in real or reciprocal space
proj	Calculates the projection from one vector on another in real or reciprocal space

Table 4.1: Commands for crystallographic calculations

The crystallographic calculation commands provided by DISCUS are shown in Table 4.1. All commands display the results on the screen and store them in the variable res[i] (see section Variables in the package manual), which allows further use of the results in macro files.

The commands d2r and r2d calculate the components of a given vector **u** in the complementary space. The vector product $\mathbf{w} = \mathbf{u} * \mathbf{v}$ gives a vector **w** that is normal to the two vectors **u** and **v**. The length of the resulting vector **w** is given by $|\mathbf{u}| \cdot |\mathbf{v}| \cdot \sin(\angle(\mathbf{u}, \mathbf{v}))$ which is the area of the parallelogram spanned by the vectors **u** and **v**. This vector product is calculated by the

command `vprod` and any of the three vectors **u**, **v** and **w** may be in direct or reciprocal space. The command `proj` computes the projection of a vector **u** onto the vector **v** or on the plane normal to vector **v**. The length of the projected vector is given by $\mathbf{u} \cdot \mathbf{v}/|\mathbf{v}|$. Again all input or resulting vectors can be in direct or reciprocal space. More details about these commands can be found in the online help of DISCUS. This version of DISCUS does not offer a special command for the scalar product of two vectors. You can readily calculate the scalar product by keeping in mind its definition: $|\mathbf{u}| \cdot |\mathbf{v}| \cdot \cos(\angle(\mathbf{u}, \mathbf{v}))$. You can calculate this value by the expression: `blen(`**u**`)*blen(`**v**`)*cosd (bang(`**u, v**`))`.

4.2   Generalized symmetry operations

Lets assume that you want to rotate the oxygen atoms of a SiO_4 tetrahedra around one of the Si-O bonds. Or you want to create the twinned structure of a triclinic crystal. The operations require the application of a symmetry element whose axis is not parallel to any of the base vectors and/or whose rotation angle is different from 60, 90, 120, or 180°. Needless to say, the resulting symmetry matrix will not contain just one's and zero's. A general symmetry operation can be described by a 4x4 matrix of the following form:

$$
S = \begin{pmatrix}
w_{11} & w_{12} & w_{13} & t_1 \\
w_{21} & w_{22} & w_{23} & t_2 \\
w_{31} & w_{32} & w_{33} & t_3 \\
0 & 0 & 0 & 1
\end{pmatrix}
\tag{4.1}
$$

Here the components w_{ij} define the rotation and t_i the translational part of the symmetry operation. A detailed description for the derivation of this general symmetry operation is found in Sands (1995).

Additionally DISCUS allows one to select the atom or molecule types to be included in the symmetry operation using the `sele/dese` and `msel/mdes` commands. The user can specify whether the atom or molecule created by the symmetry operation replaces its original or is inserted as a new atom or molecule in the crystal. The direction of the symmetry axis can either be given in direct or reciprocal space. A pure mirror operation is performed by an improper rotation by 180°.

4.3   Generalized shear operations

A similar operation is required if you want to distort the structure or part of the structure. Such a distortion may occur upon a orthorhombic to monoclinic transformation. If the whole crystal is sheared under this transformation it is, of course, easier to simply change the lattice constants. If, however, only part of the structure is to be deformed to produce for example a fish bone like pattern a local shear operation may be required.

The `shear` menu in DISCUS allows to perform a shear operation:

$$
S = \begin{pmatrix}
a_{11} & a_{12} & a_{13} & t_1 \\
a_{21} & a_{22} & a_{23} & t_2 \\
a_{31} & a_{32} & a_{33} & t_3 \\
0 & 0 & 0 & 1
\end{pmatrix}
\tag{4.2}
$$

Here the matrix elements may take up any value in order to perform the desired distortion. DISCUS allows several ways to set up the shear operation, see the on-line help for details. You can specify the full matrix, define the Eigenvectors of a shear or define the shear by a plane parallel to which a shear operation is performed.

Additionally DISCUS allows one to select the atom or molecule types to be included in the shear operation using the `sele/dese` and `msel/mdes` commands.

4.4 Unit cell transformations

A common task in crystallography is to transform the unit cell of a crystal, i.e. use an alternative setting for some specific reason. Subsequently one needs to transform the coordinates of the atoms within the crystal to the new set of basis vectors. The `trans` segment of DISCUS gives various options to perform this type of task. The transformation can either be defined as new unit cell in terms of the old unit cell or in terms of the new atom coordinates with respect to the old ones and *vice versa*. Independent of this choice, the origin can be shifted by a user defined amount. DISCUS allows the user to select individual atoms or to transform the complete crystal to the new system. If all atoms in the crystal are transformed to the new base vectors, then the unit cell dimensions and the metric tensor are transformed as well.

Let us assume we have a cubic crystal defined by $a = b = c = 5$Å. The base vectors for the old system are $\mathbf{a}, \mathbf{b}, \mathbf{c}$. The new set of basis vectors defining the new unit cell shall given by $\mathbf{a}', \mathbf{b}', \mathbf{c}'$ which are given by the relations $\mathbf{a}' = \mathbf{a} + \mathbf{b}$, $\mathbf{b}' = \mathbf{a} - \mathbf{b}$ and $\mathbf{c}' = \mathbf{c}$. After entering the `trans` segment of the program DISCUS , these definitions could be entered as follows:

```
anew 1.0, 1.0, 0.0
bnew 1.0,-1.0, 0.0
cnew 0.0, 0.0, 1.0
```

Note that the relation between the old and the new system could also be defined with respect to the atom coordinates. All those relationships are calculated by DISCUS and can be displayed using the `show` command. Part of the output for our example can be found below:

```
( a(new) ) = (   1.00000,  1.00000,   .00000 )   ( a(old) )
( b(new) ) = (   1.00000, -1.00000,   .00000 ) * ( b(old) )
( c(new) ) = (    .00000,   .00000,  1.00000 )   ( c(old) )

( a(old) ) = (    .50000,   .50000,   .00000 )   ( a(new) )
( b(old) ) = (    .50000,  -.50000,   .00000 ) * ( b(new) )
( c(old) ) = (    .00000,   .00000,  1.00000 )   ( c(new) )
```

```
( x(new) ) = (    .50000,    .50000,    .00000 )    ( x(old) )   (    .00000)
( y(new) ) = (    .50000,   -.50000,    .00000 ) * ( y(old) ) + (    .00000)
( z(new) ) = (    .00000,    .00000,   1.00000 )    ( z(old) )   (    .00000)

( x(old) ) = (   1.00000,   1.00000,    .00000 )    ( x(new) )   (    .00000)
( y(old) ) = (   1.00000,  -1.00000,    .00000 ) * ( y(new) ) + (    .00000)
( z(old) ) = (    .00000,    .00000,   1.00000 )    ( z(new) )   (    .00000)
```

This particular example might seem very simple but as soon as e.g. a unit cell transformation in a triclinic system is needed, this part of DISCUS becomes the most popular feature.

Chapter 5

Simple crystal modifications

Chapter 2 discussed techniques to create a perfect model crystal from the content of the asymmetric unit. This chapter will give an overview of simple modifications of single atoms within a model crystal. Section 6 will describe DISCUS tools to modify a complete crystal.
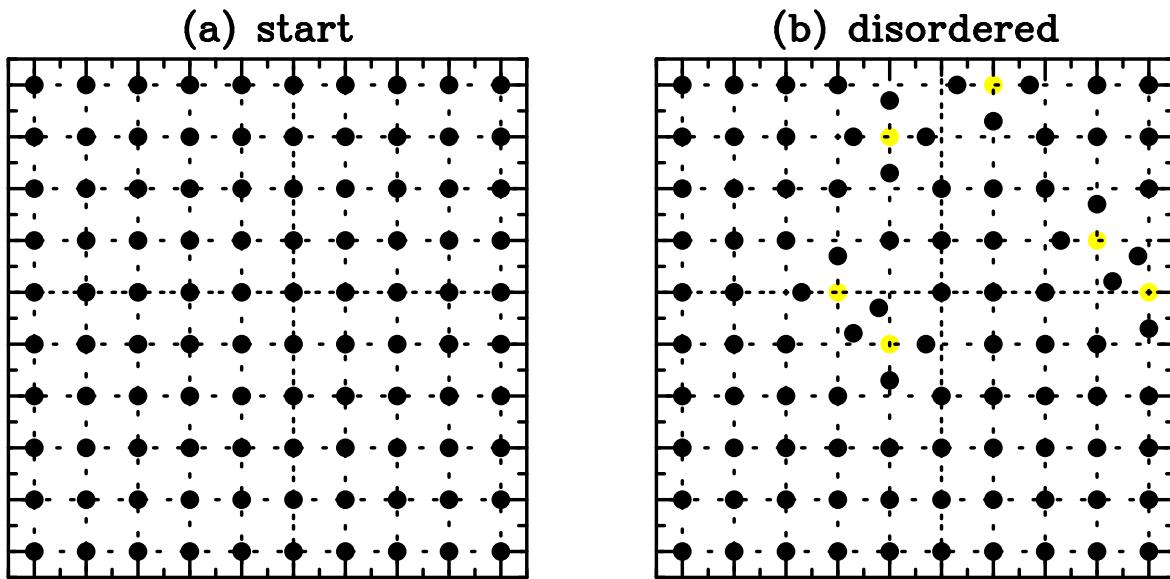
5.1   Modifications using variables

Each atom $<i>$ within the model crystal stored by the program DISCUS is associated with a set of variables x[$<i>$], y[$<i>$] and z[$<i>$] that describe its position and the variable m[$<i>$] that contains the atom type (see also section 1.2). In principle every desired defect structure might be realized using these variables and the FORTRAN style loops and conditional statements (see chapter FORTRAN style interpreter in the package manual). One should be aware that this procedure may be very slow for complex problems and large model crystals. Since the variables are addressed using the atom index $<i>$, a knowledge about the internal storage as described in section 2.3 is very important. The command `trans` in the 'chem' segment of DISCUS allows one to convert between atom index and the corresponding unit cell and site. To illustrate the use of variables, the following type of defect should be constructed using DISCUS. The starting structure is a 10x10x1 unit cell square symmetric crystal with Zr on (0,0,0) and a lattice constant of a=5Å. The defects consist of randomly introduced vacancies on Zr sites and nearest neighbors surrounding the vacancy should be relaxed towards the vacant site. The perfect starting structure and the resulting disordered structure are shown in figure 5.1. Here is the macro used to create the described defects. Again the line numbers are only shown for convenience and not part of the macro itself. To achieve a high degree of flexibility values like crystal size or the number of defects to be created are stored in variables at the beginning of the macro. The desired crystal size is stores in variables i[1], i[2] and i[3] (lines 1-3). The variable i[4] (line 5) gives the number of defects to be created and r[1] (line 6) specifies the relaxation (here 30 %) of the surrounding neighbors towards the vacant site.

```
1  i[1] = 10
2  i[2] = 10
3  i[3] = 1
4  #
5  i[4] = 6
6  r[1] = 0.30
7  #
```

(a) start   (b) disordered

Figure 5.1: Structures created by crystal modification example

Next we read the unit cell from file *cell.cll* and expand it to the desired size (lines 8-9). The macro *plot.mac* (line 11) saves the starting structure suitable for plotting with *KUPLOT* as seen in figure 5.1a).

```
 8  read
 9  cell cell.cll,i[1],i[2],i[3]
10  #
11  @plot before.plot
12  #
```

The next part (lines 13-15) is needed to switch off periodic crystal boundaries for the command `find` (line 24), otherwise our simple way of relaxing the neighbors would not work.

```
13  chem
14  set mode,quick,noperiodic
15  exit
```

Now the creation of the disordered structure starts with a loop over the number of defects to be created (line 19). Next a crystal site is chosen at random (line 20). Note that the function `ran(0)` produces a random number between 0 and 1 which is multiplied with the number of atoms within the crystal (n[3] contains the number of atoms per unit cell, see table 1.2 in section 1.2). If an occupied site was picked (line 22) the atom is removed (line 23). The command `find` (line 24) returns all atoms of the type Zr around the position of the selected site within a radius of 5.5Å. The atom indices of these nearest neighbors are stored in the variables `env[<i>]` and `env[0]` contains the number of found neighbors. Finally the positions of all neighboring atoms returned by `find` are moved towards the vacant site (lines 27-29).

```
16  #
17  # Loop number of wanted defects
18  #
19  do i[5]=1,i[4]
```

```
20    i[6]=int(ran(0)*i[1]*i[2]*i[3]*n[3])+1
21  #
22    if(m[i[6]].eq.1) then
23      m[i[6]]=0
24      find env,zr,x[i[6]],y[i[6]],z[i[6]],5.5
25  #
26      do i[7]=1,env[0]
27        x[env[i[7]]]=x[env[i[7]]]-r[1]*(x[env[i[7]]]-x[i[6]])
28        y[env[i[7]]]=y[env[i[7]]]-r[1]*(y[env[i[7]]]-y[i[6]])
29        z[env[i[7]]]=z[env[i[7]]]-r[1]*(z[env[i[7]]]-z[i[6]])
30      enddo
31  #
32    endif
33  #
34  enddo
35  #
36  @plot after.plot
```

The resulting structure can be seen in figure 5.1b. The introduced vacancies are shown as yellow circles (or light grey). Since this simple macro does not check whether neighboring atoms are already displaced, the clustered defects in the upper right quadrant of figure 5.1b have a different local environment compared to the isolated defects.

5.2   Build in functions

This section will give an overview of those DISCUS functions modifying single atoms or molecules. Some of these functions can be realized using variables as well, others cannot like the command `insert` to insert a new atom in the model crystal. Table 5.1 summarizes the available commands. Basically there are three commands to insert a new atom in the structure. The command `insert` just creates a new atom at a specified position without taking notice of its environment. Thus the new atom might be on top of an existing one. In order to avoid this problem, DISCUS provides two other commands `append` which will not insert the atom if there are other atoms within a given distance and `kick` which will remove those atoms close by before inserting the new one. The command `copy` allows the user to copy an existing atom to a new position that can be given in absolute coordinated or relative to the position of the atom to be copied. Note, that these commands are limited to atoms. New molecule types can be created and molecules copied using the generalized symmetry segment of DISCUS described in chapter 4.2.

The commands `remove` and `switch` work for atoms as well as molecules. Not surprisingly, `remove` removes an atom or molecule and `switch` swaps two atoms or molecules with respect to their location within the crystal.

The commands mentioned in this section so far operate only on a single atom or molecule. The command `replace` can replace a single atom or molecule by a given type or replace more until a given concentration is reached. Although this can be easily realized using the FORTRAN interpreter and variables associated with the crystal, the internal function is much faster, especially for large crystal sizes. The command `purge` will remove all vacancies (VOID) within the crystal. Note that if you remove an atom or a molecule by the `remove` command, DISCUS will not actually remove atoms but rather replace their atom type with VOID. Thus when saving the structure a large number of VOID's might appear in the number. The use of `purge` actually removes those VOID atoms. However, the use of the `purge` command is **not recommended**

Command	Description
append	Appends an atom at a given position within the crystal if no other atoms are present in a given distance.
copy	Copies an atom to a different position given absolute or relative to the other position.
insert	Inserts a new atom at the given position without condition.
kick	Inserts a new atom at a given position and removes all other atoms within a given distance from the new atom.
remove	Removes an atom or molecule from the crystal.
switch	Swaps two given atoms or molecules within the crystal.
purge	Removes vacancies (VOID) from the crystal (**use with care**)
replace	Replaces one or more atoms or molecules with the given type.

Table 5.1: DISCUS commands for single atoms or molecules

since many DISCUS function require the crystal to be build in a given order, i.e. having the same number of atom sites within every unit cell either occupied by an atom or a VOID.

5.3   Inserting extended objects

In order to insert extended objects for small angle scattering or domains 6.3 DISCUS offers an insert menu. In contrast to the insert command for individual atoms, the menu is evoked by the `insert` command followed by the keyword `domain` or `object`. A corresponding version for molecules is under construction.

5.3.1   Inserting domains

Within the DISCUS formalism, a domain (see section 6.3 for a full explanation) is essentially a molecule which will be interpreted differently. The domain allows you to place an extended defect into a host structure. The `insert domain` menu allows you to specify the character and location of the domain to be inserted.

5.4   Building a molecule

To build a molecule or to extend an existing molecule, DISCUS offers the `molecularize` command.

```
1 molecularize conn, <central> [, <exclude> ...]
2 molecularize range, <from>, <to>, <molecule_type> , <molecule_Biso>
```

The first command version builds molecules for the central atom number <central>. Prior to this command a connectivity list must have been created with the help of the connectivity menu. This connectivity list is searched to find all neighbors to the central atom type and the resulting further neighbors in turn. The list of excluded atom number can be used to prevent these excluded atom to become part of the molecule. This allows you to build a partial molecule, which could be helpful if a section of a molecule needs to be turned with respect to the rest.

The second command form combines all atoms from number <from> to number <to> into a singel molecule. DISCUSdoes not check if these atoms are close to each other.

5.5   Dissolving a molecule

Under rare cirumstances it might be advantageous to *dissolve* a molecule. The idea is to remove all molecule info, while retaining the atoms that are in the molecule at their original place with all other but molecule related properties to remain the same. The instructions for this task are bundled in the "demolecularize" menu.

```
1 demolecularize
2   reset
3   select moletype:all,atomtype:none
4   include molerange:all, atomrange:none
5   property {"ignore"|"present"|"absent"} , <property> [, <property> ...]
6   show
7   run
8 exit
```

The "select" and "include" commands on line 3 and 4 are the two main instructions for this menu. "select" allows you to chose, which molecule types are to be dissolved. Possible values for the parameter are:

Parameter	Description
none	No molecule type will be dissolved. Default
all	All molecule types will be dissolved.
<n1>	Only molecule of type <n1> will be dissolved.
(<n1>,<n2>,...)	All molecule of the specified types will be dissolved. The list needs to be enclosed in round parentheses.

The "include" command defines which numerical range of molecules shall be dissolved.

Parameter	Description
none	No range of molecules will be dissolved. Default
all	All molecule numbers will be dissolved.
<n1>	Only the molecule number <n1> will be dissolved.
(<n1>,<n2>)	All molecules in the range from <n1> to <n2> will be dissolved. If <n1> is equal to -1 or "last", all molecules from number <n1> to the last molecule will be dissolved. The list needs to be enclosed in round parentheses.

The two commands work together as a logical AND. Only molecules that are in the list of molecule types and that fall in the range of molecule numbers will be dissolved.

To specify even more details, the selection and inclusion can be more fine tuned with the "atomtype" and "atomrange" parameters.

The atomtype accepts the following parameters:

Parameter	Description
none	The atoms in the molecule do not have to be of a specific type for the molecule to be dissolved. A molecule may consist of any atom type(s). Default
all	A molecule must contain all atom types to be dissolved. If a molecule is missing any atom type it will not be dissolved.
<n1>	A molecule must contain at least one atom of type <n1> to be dissolved. Other atom types may be present as well and do not affect the decision.
(<n1>,<n2>,...)	A molecule must contain at least one atom of each of the specified types to be be dissolved. The list needs to be enclosed in round parentheses.

The "atomrange" command accepts the following parameters:

Parameter	Description
none	No specific range of atoms needs to be present in a molecule for it to be dissolved. Default
all	Any molecule must contain all atoms from number 1 to the last to be dissolved.
<n1>	The molecule must contain exactly the atom number <n1> in order to be dissolved.
(<n1>,<n2>)	All atoms in the molecule must fall into the range from <n1> to <n2> for the molecule to be dissolved. It is not necessary that the entire range is covered. The condition states that if any atom in the molecule is outside the range from <n1> to <n2> the molecule will not be dissolved. If <n1> is equal to -1 or "last", all molecules from number <n1> to the last molecule will be dissolved. The list needs to be enclosed in round parentheses.

The local properties set with the "property" command define, which properties all atoms in a given molecule must have for this molecule to be demolecularized. The properties default to "ignore", which means that an atom may have any property, without affecting the choice for the given molecule.

Both, the "atomtype" and "atomrange" and the property selection impose conditions that must be met, they are interpreted as a logical AND with the "moletype" and "molerange" parameters.

5.6 Creating a periodic structure

Under several circumstances, the current structure might not be periodic or at least might not correspond to the sequence of atoms and unit cells as generated upon the expansion of an asymmetric unit, see chapter 2. A simple example might be the structure after you purged vacancies. Other more involved examples might result from the introduction of domains, see 6.3, or if the crystal was build using stacking faults, see 6.4. Finally, the sequence of atoms generated for other programs like RMCprofile, molecular dynamics programs or DFT like programs will likely be different from the sequence use by DISCUS.

For fast calculations within DISCUS it is often desireable to have the atoms arranged in the strict sequence as generated in chapter 2. This sequence allow DISCUS to find neighboring atoms much faster as the search is restricted to a few unit cells. These searches are perfomed at many parts of the program. It is thus desireable to be able to sort the atoms of a given structure in this standard sequence.

The `perioditize` menu has been designed to perform this task. Under opportune circumstances it will do the job completely automatically, but it will most likely to be helped by providing information such as the number of unit cells, the number of atoms per unit cell and the sites within the unit cell.

A simple macro to perform the task might be:

```
1  perioditize
2    set ncell, 3,4,5
3    set natoms, 8
4    run
5  exit
```

The `ncell` command defines that the crystal consists of the given number of unit cells along the three axes. The `natoms` command defines the number of atoms per unit cell. As no `set pos` command is given, the program will attempt to automatically establish the actual positions within the average unit cell.

The task is much simpler if the user specifies the actual sites within the average unit cell. this can be done within the `perioditize` menu with

```
1  set site, atoms:[names], pos:[x,y,z], number:value
```

Both, the `atoms:` and the `pos:` parameters are required. The `atoms:` parameter defines which atom types may occupy the given site. Atom names can be spcifies as list of current atom names or alternatively as the string `all` to imply all atom types. The parameters to the `pos` parameter must specify fractional coordinates in the range [0,1]. The `number` parameter is optional and mainly serves to correct previous settings.

In case of a structure with many symmetrically equivalent sites, it will be more comfortable to use the equivalent command:

```
1  set wyck, atoms:[names], pos:[x,y,z]
```

In contrast to `set site` this version creates all symmetrically equivalent Wyckoff sites for the position [x,y,z]. It uses the current space group. As several sites will be generated, `set wyck` does not use the `number:` parameter, and the new sites will be placed into the next free spaces. There are many cases under which a structure cannot be rearranged in a periodic fashion since the underlying actual structure simply is not periodic or differs too widely for the current algorithm to cope. Examples include:

Example	Description
amorphous	A truly amorphous structure such as a glass or liquid cannot be expressed as a periodif structure.
composite	A structure that consists of blocks with different internal lattice parameters. This could be an inclusion compound or a layered structure.
core/shell	Nanaoparticles whose core differs in its structure from the shell.
stacks	The result of stacking faults will in most cases yield an inherently non-periodic structure. This holds in particular if the translation vectors are not simple rational numbers.
finite	For a finite object created with the surface menu large fractions might be missing. These can only be reconstructed if the full site, unit cell number and atoms per unit cell is given.

Chapter 6

Build in defect models

The following built in defect models modify the whole structure. They offer extended defects that include modification of many atoms throughout the whole crystal. While such modifications could be achieved as well by the user through extended use of the FORTRAN interpreter, this would require extremely lengthy and necessarily slow macros. Defect models built in the source code of the program on the other hand are calculated in much shorter time. The defect models were implemented with as many general features as seemed necessary. They use separate sub menus to define the intended defect structure.

6.1 Thermal displacements

The command `therm` can be used to randomly displace all atoms or rigid molecules of the crystal as to be expected by purely random thermal disorder. The direction of the displacement is distributed with a uniform random spherical distribution. The amplitude of the displacement is distributed by a Gaussian distribution. The mean of the distribution is zero, its sigma is calculated from the isotropic thermal coefficient, B, of each atom as:

$$\langle u^2 \rangle = \frac{B}{8\pi^2} \tag{6.1}$$

After the command `therm` is executed, a summary of theoretical and achieved displacement averages is displayed on the screen. An example is shown below:

```
Thermal displacements summary :

              Input              Achieved              Maximum displacement
Atom        B    <u**2>      <ux**2> <uy**2> <uz**2>       x       y       z
-----------------------------------------------------------------------------
LA(1)      0.34  0.0043      0.0043  0.0043  0.0044      0.2299  0.2450  0.2487
MN(2)      0.21  0.0027      0.0027  0.0027  0.0026      0.1980  0.1868  0.2184
O(3)       0.50  0.0063      0.0061  0.0063  0.0064      0.2827  0.3518  0.3011
O(4)       0.43  0.0054      0.0054  0.0055  0.0055      0.2745  0.3036  0.2855
```

The first column lists the name of the atom type followed by the B and corresponding $\langle u^2 \rangle$ value. The next three columns show the achieved $\langle u_i^2 \rangle$ values for the x-, y- and z-direction. The last three columns give the maximum displacements in the three directions that were introduced. The values are given in Å^2 and Å.

The parameter `mol` will displace rigid molecules according to the B value of the atom at the origin of the molecule. Furthermore the parameter `2d` allows to restrict the displacements to directions with more than one unit cell extension of the crystal. This might be used when working with 2-dimensional model crystals where a displacement in the third direction is not wanted.

6.2 Modulations

The given structure can be modulated using the `wave` segment of the program DISCUS . Three different types of waves are available, density waves modulating the occupation of sites within the crystal, displacements waves modulating the position of atoms or molecules and rotational waves which modulate the orientation of molecules by rotating them around a user defined axis. First common features shall be described followed by details about the different wave types in separate sections.

DISCUS offers three different wave functions $w(\mathbf{r})$, sinusoidal, square and saw tooth defined in equations 6.2, 6.3 and 6.4, respectively. The parameter δ for the box shaped wave functions determines its asymmetry. The symmetric case is given by $\delta = 0.5$. The computed value of $w(\mathbf{r})$ modulates density, position or orientation as a function of the position $\mathbf{r}$ within the crystal depending on the wave type selected.

$$w(\mathbf{r}) = A \cos(2\pi \left[\frac{\mathbf{kr}}{\lambda} + \psi\right]) + A_0 \tag{6.2}$$

$$w(\mathbf{r}) = \begin{cases} A + A_0 & \text{for } \frac{\delta}{2} \leq \left|\frac{\mathbf{kr}}{\lambda} + \frac{\psi}{360°}\right| < 1 - \frac{\delta}{2} \\ \\ A_0 & \text{otherwise} \end{cases} \tag{6.3}$$

$$w(\mathbf{r}) = A \left[\frac{\mathbf{kr}}{\lambda} + \frac{\psi}{360°}\right] + A_0 \tag{6.4}$$

The following list describes the different properties common to all wave types that can be defined by the user.

- *Wave vector* **k***:* The wave vector **k** defines the *traveling* direction of the wave. The vector is defined using the `vect` command in multiples of the base vectors of the crystal, i.e. in unit cell units.

- *Wave length* λ*:* The wave length λ of the modulation wave is entered using the `len` command. Note that λ must be given in Å.

- *Amplitude A:* The amplitude A defines the upper limit of displacements (in Å) or rotation angle (in degrees). In case of density waves the probability of replacing an atom or molecule oscillates between an upper and lower limit (see section 6.2). The command to set the amplitude A is `amp`.

- *Constant offset* A_0*:* A constant displacement or rotation angle can be added to all selected atoms or molecules using the command `amp0`.

- *Phase ψ:* The phase ψ of the atom or molecule at (0,0,0) can be altered via the command `phase`.

The argument $[\mathbf{kr}/\lambda + \psi/360°]$ of the wave functions is limited to the range of -1 to 1 for the box and saw tooth function and to a range of $\pm 2\pi$ for the sinusoidal wave function. The origin of the vector $\mathbf{r}$ is the crystals origin. DISCUS allows the user to select which of the atoms in the crystal are to be modified by the wave function.

Density waves

Displacement waves are selected using the `dens` command. A density wave replaces an existing atom or molecule by another atom or molecule type or alternatively removes the atom or molecule. The direction of the density wave is along the wave vector. A lower and upper probability P_{low} and P_{high} for replacing an atom or molecule must be provided using the command `plow` and `phigh`. The probability for the current atom or molecule to remain in the structure oscillates between these two values. The parameters A and A_0 are calculated from these two probabilities. In case of a sinusoidal wave the parameters A and A_0 are defined by:

$$
\begin{aligned}
A &= \frac{1}{2}(P_{high} - P_{low}) \\
A_0 &= \frac{1}{2}(P_{high} + P_{low})
\end{aligned}
\tag{6.5}
$$

In case of a box or saw tooth shaped wave function, the parameters A and A_0 are determined by the following equations:

$$
\begin{aligned}
A &= P_{high} - P_{low} \\
A_0 &= P_{low}
\end{aligned}
\tag{6.6}
$$
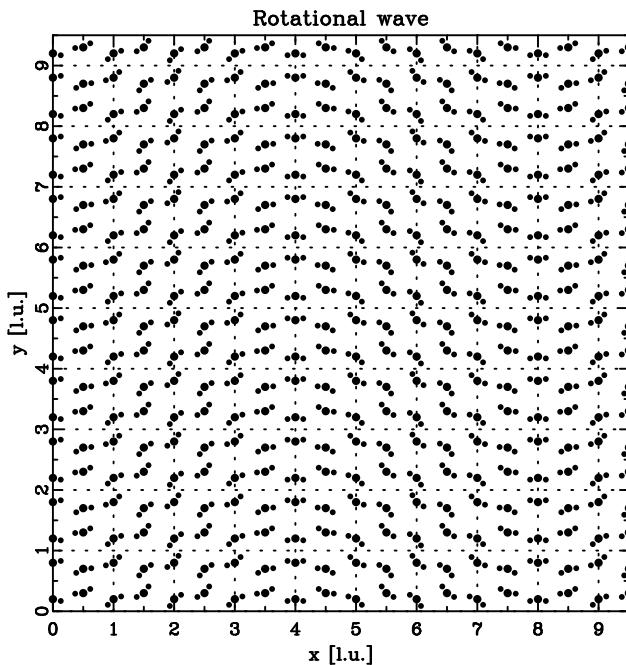
A random number is calculated every time an atom is encountered that had been selected. If this random number is less than $w(\mathbf{r})$, the amplitude calculated by the wave function, the atom or molecule is retained as was, otherwise it is deleted or replaced by another atom or molecule. Its position is not changed.

Displacement waves

Displacement waves are selected using the `trans` command. For displacement waves, the wave function $w(\mathbf{r})$ determines the displacement of an atom or molecule along a given direction of oscillation. The command `osci` defines the oscillation direction in lattice units. DISCUS allows any direction of that vector relative to the propagation direction of the wave. In cases where propagation and oscillation vector are parallel, we speak of longitudinal waves. In cases where propagation and oscillation are normal to each other we speak of a transversal wave.

The default displacement mode is set to acoustic, i.e. all atoms are displaced in the same direction. An (admittedly crude) optical mode will displace all atoms that are identified as negative ions in the opposite direction to all others. Negative ions are recognized through their respective name, e.g. `CL1-`. As a side effect, if charged ions are used, the Fourier transform will use the corresponding scattering curve. If this is to be avoided, calculate the desired wave twice, once with positive amplitude and once with negative, while selecting the corresponding atoms.

Rotational waves

Another feature of DISCUS are rotational waves limited to be used with molecules. Obviously rotating an atom around itself makes no sense. Rotational wave are selected with the `rot` command followed by parameters defining the rotation axis relative to the molecules origin in lattice units. The wave function $w(\mathbf{r})$ then modulates the rotation angle ϕ around this axis. Additionally an optional offset relative to the molecule origin for the rotation axis can be specified by the `rot` command. An example of such a rotational wave can be seen in figure 6.1.

Figure 6.1: Example of a rotational wave

Here a cubic crystal with a size of 20x20x1 unit cells (a = 10Å) was used, each containing four H_2O molecules. The rotation axis is normal to the drawing plane, i.e. in z-direction. The wave vector is parallel to x and the wave length λ is eight unit cells (80 Å). The amplitude A was set to 45.0, i.e. the rotation angle ϕ oscillates between ± 45 degrees.

6.3   Domains

The `domain` module of DISCUS has replaced the microdomain module. A domain within DISCUS is understood as a place holder for an extended defect. This defect might be as small as an individual atom or be a region that extends over many unit cells.
Note, that in our book (Neder and Proffen, 2007) we present examples using this domain module and discuss domains at length.
As usual, the commands related to this modules are accessed after entering the module via the command `domain`. Let us look at a simple example how to use this module:

```
 1  domain
 2    rese
 3    mode    pseudo
 4    input   dom.cube.pseudo
 5    assign char,si,sphere
 6    assign fuzzy,si,0.5
 7    assign cont, si,dom.guest.cube.stru
 8    assign shape ,si,1,  1. , 0. , 0. ,  0.
 9    assign shape ,si,2,  0. , 1. , 0. ,  0.
10    assign shape ,si,3,  0. , 0. , 1. ,  0.
11    assign orient,si,1,  1. , 0. , 0. ,  0.
12    assign orient,si,2,  0. , 1. , 0. ,  0.
13    assign orient,si,3,  0. , 0. , 1. ,  0.
14    show
15    run
16  exit
```

After resetting the module (line 2), mode `pseudo` is selected. In this mode, a DISCUS structure file is used to define the distribution of domains and each *pseudo atom* in this file is later replaced by the corresponding domain. In our case, the atom Si is replaced by the defined spherical domain (line 5). The boundary is created by removing all atoms that are within 0.5Å of the domain (line 6). Next the domain dimension is define using the matrix defined in lines 8–10. These values are in terms of unit cells of the host structure. The second matrix (lines 11-13) defines the orientation of the domain. Finally the settings are listed (line 14) and the domain is finally created (line 15). There is an alternative extension of a structure file and the keyword `domain` allows to readily define the domain structure, its shape and orientation (see Neder and Proffen (2007)).

6.4 Stacking faults

All crystals whose structures can be described by layers are prone to stacking faults. A stacking fault is any defect that alters the periodic sequence of layers. These defects may be a wrong layer inserted into the sequence, a change of the layer sequence or a different translation between two subsequent layers. These defects may affect the whole crystal or a finite region if e.g. an additional layer is present between an otherwise perfect sequence of layers. DISCUS contains a tool to create layered crystal structures and to introduce stacking faults into these crystals. The crystals are formed in a two step procedure. First, the origin and type of each layer is determined and second, the atoms corresponding to each layer are introduced into the crystal. The user can define each layer type, the translation vectors between consecutive layers and the correlation between neighboring layers. A new feature of the stacking fault part of DISCUS is the addition of rotational disorder for the layer sequences.

The stacking fault sequence is defined by several parameters that can be set in the `stack` sub level of DISCUS:

- *Type of layers :* The positions of all atoms within a layer are read from a DISCUS type structure file. These layer files have to be created for each layer type involved beforehand using the various DISCUS tools.

- *Translations :* A translation vector between neighboring layers of each type must be provided. Thus N different layer types result in a N*N matrix of translation vectors. An example for translation vectors in a cubic face centered structure is given in table 6.1.

- *Uncertainties for translation vectors:* In some materials small deviations in the translation vectors might occur. This behavior can be simulated in DISCUS by setting a standard deviation σ to each of the elements of the translation vector matrix. DISCUS will calculate the actual translation vector as sum of the 'ideal' vector plus a Gaussian distributed part defined by the value of σ.

- *Correlations :* A correlation matrix is used to define the probabilities of two layer types to be nearest neighbors. No further correlations are taken into account. To create correlations with *Reichweite* larger than 1, you can build a one dimensional crystal of two or more atom types. The types and positions of these atoms can be interpreted as layer types and origins.

- *Crystal shape :* The resulting crystal can be generated using two different modes: First, the crystal continuously grows in one direction as given by the translation vector(s). Secondly one or two coordinates can be constrained to a finite range, which results in a zig-zag shaped crystal. If any of the parameters is not equal to zero, the corresponding coordinate of the origin is taken modulo this parameter. Note, that DISCUS does not check whether the modulo vectors are translation vectors of the current space group.

Layer type A	Layer type B	Translation vector
A	A	$(1,1,1)$
A	B	$\frac{1}{2}(1,1,0)$
A	C	$\frac{1}{2}(1,0,1)$
B	A	$\frac{1}{2}(\bar{1},\bar{1},0)$
B	B	$(1,1,1)$
B	C	$\frac{1}{2}(0,1,1)$
C	A	$\frac{1}{2}(\bar{1},0,\bar{1})$
C	B	$\frac{1}{2}(0,\bar{1},\bar{1})$
C	C	$(1,1,1)$

Table 6.1: Translation vectors for stacking faults in a cubic face centered structure

The command `create` in the `stack` segment of DISCUS creates the list of layer origins and `run` actually generates the corresponding crystal by decorating the origins with the individual layer types. In order to speed up the calculation of the Fourier transform, rather than using the resulting complete structure, the scattering intensity is calculated in the following way. The

scattering density $\rho(\mathbf{r})$ of a layered structure can be expressed as the scattering density of the individual layer types convoluted with the layer origin distribution.

$$\rho(\mathbf{r}) = \sum_{i=1}^{nl} \left\{ \sum_{j=1}^{no} o_{ij}(\mathbf{r}) \right\} \star l_i(\mathbf{r}) \tag{6.7}$$

The outer sum runs over all nl layer types and the inner sum runs over all origins o_{ij} of layer type i. The variable l_i is the scattering density of layer type i. Using the convolution theorem, the Fourier transform of this expression becomes

$$\mathcal{F}\{\rho(\mathbf{r})\} = \sum_{i=1}^{nl} \mathcal{F}\left\{ \sum_{j=1}^{no} o_{ij}(\mathbf{r}) \right\} \cdot \mathcal{F}\{l_i(\mathbf{r})\} \tag{6.8}$$

Here $\mathcal{F}$ denotes the Fourier transform. This procedure not only speeds up the calculation but it also allows the usage of much larger crystal sizes since the actual structure does not have to be created in order to calculate the Fourier transform.

6.5 Surfaces

An initial model crystal will be a block of IxJxK unit cells. This shape is ideal to avoid finite size effects when you want to calculate diffuse scattering for an infinite large crystal. Nanoparticles or domains are of course explicitly finite objects, whose shape you might want to control. DIS-CUS offers the `surface menu` to shape the crystal into a finite sized object. The menu replaces the `boundary` command that is still available for backward compatibility.
Within the `surface` menu, the `boundary` command allows you to shape the crystal by any combination of:

- an individual lattice plane `HKL`

- a form, i.e. a set of symmetrically equivalent lattice planes `HKL`

- a spherical surface

- a triaxial ellipsoidal surface

- a cylindrical surface

A plane will remove all atoms to one side of this plane:

```
boundary hkl, 1, 1, -1, 15.0, inside
```

In this example a $11\overline{1}$ plane is placed at 15.0Åfrom the origin. All atoms remain that are inside the boundary plane i.e. at the same side of the plane as is the origin 0.0, 0.0, 0.0. As a plane extends to infinity parallel to its surface, DISCUS cannot create an object with an indented surface using individual planes. Such an object can be build using a
tt form,

tt sphere,
tt ellipsoid or
tt cylinder.
A complete form of symmetrically equivalent faces can be created via:

```
boundary form, 1, 1,  1, 15.0, inside
```

This will create an object that is limited by the complete set of all symmetrically equivalent {111} faces. Keep in mind that this set of faces will depend on the point group symmetry of your crystal!
A spherical object with radius R is created by the command:

```
boundary sphere, 35.0, inside
```

A triaxial ellipsoid with diameters DX=20, DY=30, DZ=40 Å is created by the command:

```
boundary ellipsoid, 20, 30,40, inside
```

Admittedly it is a bit of an inconsistency that the "ellipsoidal" surface takes diameters instead of radii. Independent of the crystal system, the three axes of the ellipsoid are always orthogonal to each other. The ellipsoid is oriented in a standard fashion. This orientation is chosen such that the ellipsoid "Y"-axis is parallel to the crystallographic b-axis, the "Z" axis parallel to the c∗ axis and the "X" axis forms a right handed orthogonal system. Table 6.2 lists the resulting orientations for the seven crystal systems:

system	X-axis	Y-axis	Z-axis
cubic	$\vec{a} = [100]$	$\vec{b} = [010]$	$\vec{c} = [001]$
hexagonal	$\vec{a^*} = [210]$	$\vec{b} = [010]$	$\vec{c} = [001]$
trigonal	$\vec{a^*} = [210]$	$\vec{b} = [010]$	$\vec{c} = [001]$
rhombohedral	$\vec{b} \otimes \vec{c^*}$	$\vec{b} = [010]$	$\vec{c^*}$
tetragonal	$\vec{a} = [100]$	$\vec{b} = [010]$	$\vec{c} = [001]$
orthorhombic	$\vec{a} = [100]$	$\vec{b} = [010]$	$\vec{c} = [001]$
monoclinic	$\vec{b} \otimes \vec{c^*} = [100]$	$\vec{b} = [010]$	$\vec{c^*}$
triclinic	$\vec{b} \otimes \vec{c^*} = [100]$	$\vec{b} = [010]$	$\vec{c^*}$

Table 6.2: Orientation of an ellipsoidal surface for the seven crystal systems.

If the last parameter on the `boundary` command is `outside`, all those atoms are retained that are on the outside of the given surface. For a single plane, only those atoms remain, that are

on the outside of the surface. For the closed shapes `form`, `sphere`, `ellipsoid`, `cylinder` a cavity inside your crystal will be the result. Keep in mind, that the `form` will not necessarily result in a true closed shape. Whether a closed shape results depends on the actual Miller indices and the point group.

In order to create closed interior hollow spaces in low-symmetry structures, the shapes: `sphere`, `ellipsoid`, `cylinder` can be used in any crystal system. If the shape shall consist of flat surfaces you will likely need a combination of several symmetrically non-equivalent faces to generate a closed shape. DISCUS provides optional parameters to the `boundary command` for this purpose:

```
boundary  form, 1, 0,  1, 15.0,outside, accum:init, exec:hold
boundary  form, 1, 0, -1, 15.0,outside, accum:add , exec:hold
boundary  hkl , 0, 1,  0, 15.0,outside, accum:add , exec:hold
boundary  hkl , 0,-1,  0, 15.0,outside, accum:add , exec:run
```

The "accum:init" indicates to DISCUS to initialize a list of faces/forms. The actual cut is put on hold via the "exec:hold" statement. the next two commands add a further face to the collection, while still keeping the execution on hold. The third statement finally adds the last face to make a completely closed shape in point group P2. As this is the final faces, the execution is performed via the "exec:run" parameter.

To shape an convex object, these two optional parameters are not needed, as each face may cut away the atoms independently. For that reason, the two parameters default to "accum:init" and "exec:run".

Three further optional parameters allow you to place the center of the shape at an arbitrary point. These parameters take the style:

```
boundary form, 1, 1,  1, 15.0, inside, centx:1.0, centy:2.0, centz:3.0
```

The colon ":" indicates that this is an optional parameter. The value after the colon defines the respective position of the center along the three crystallographic axes. All values default to zero.

A special setting within the `surface` menu allows to create core-shell structures.

```
set original:[Ni], replace:[Pt]
boundarx sphere, radius
```

This macro assumes that the original crystal consisted of Ni. The optional `replace` parameter in this macro instructs DISCUS to replace the Ni atoms by Pt instead of the VOIDs, which would be the default. The atom list to the `original` parameter can be the string "all" or a comma delimited list of current atom names. Likewse the atom list on the `replace` parameter can be a single atom type or a comma delimited list of atom types.

6.6 Decorating Surfaces

The `decoration` menu within DISCUS can be used to decorate a surface with a group of additional atoms. The steps involved in such a decoration process are two fold:

- Shape a crystal into the desired form via the `surface` menu

- Place one or several ligand molecules onto the surface

A common set of command to achieve this would be:

```
1  read
2     cell structure.cell, 20, 20, 20
3  surface
4     boundary form, 1,0,0, lat[1]*4.5, keep:inside
5  exit
6  #
7  decorate
8     reset
9     add example, normal
10    set example, ligand, ligand.stru, 0.015
11    set example, bond,    au, 1, 2.42
12    set example, axis, auto
13    set example, form,  1, 0, 0
14    show
15    run
16 exit
```

After the shaping of the crystal into a cube (here assuming a cubic structure in "structure.cell")
DISCUS enters the `decorate` menu in line 7. Once the mode is reset to ensure clean starting
conditions, a new decoration mode is `added` in line 9. The string "example" serves as a name
for further references. The bonding style for this decoration scheme is set to `normal`. This will
place the ligand molecule with a single bond onto the surface and align the molecule normal
to the local surface. DISCUS currently offers six bonding schemes that are detailed further
down. In line 10 we specify the source of the decorating molecule as the file `ligand.stru`.
The molecule shall be placed onto the surface at a (rough) density of 0.015 molecules per Å^2.
Line 11 specifies that a single bond between surface "Au" atoms and the first atom in ligand.stru
should be build at a bond distance of 2.42Å. DISCUS is instructed in line 12 to align the molecule
axis automatically. Finally in line 13 the decoration is limited to surfaces that belong to the form
{100} of the current crystal system. The standard `show` and `run` commands complete the action.
After this quick run through a more detailed explanation will follow. DISCUS will place the
molecule contained in the file specified on the `ligand` instruction onto the surface of the par-
ticles. Prior to this placement process, the particle must have been shaped via the `surface`
menu as DISCUS relies on the surface property that is generated in the `surface` menu. Once
you have chosen surface atoms onto which the guest molecules shall be placed, DISCUS will
internally select these surface atoms and arrange these as evenly as possible on the surface.
This distribution process attempts to avoid overlap between the individual guest molecules.
Admittedly this will not always be perfect, especially for a high surface density. If necessary
use a lower density or use the `mmc` menu to twist the guest molecules in order to avoid an
overlap.
Currently DISCUS offers six different bonding schemes for the nanoparticle decoration:

- `normal` The molecule is placed with a single bond onto the surface. This bond is oriented
  normally to the local surface on top of the surface atom. Either automatically or via the
  pair of atoms provided via the `axis` keyword, DISCUS aligns the molecule itself normal
  the the surface as well.

- `bridge` The molecule is placed with two bonds onto the surface, midway between the
  two surface atoms. Both bonds may have different lengths and a triangle results. The two

atoms within the surface and one atom within the ligand molecule form this triangle. The plane of this triangle is placed parallel to the local surface normal. The remainder of the molecule is aligned as in the `normal` mode.

- `chelate` This is essentially the `bridge` mode in reverse. The molecule is placed with two bonds onto the surface. The two bonds connet a single surface atom to two different ligand atoms. Both bonds may have different lengths and a triangle results. The two atoms within the ligand and the single surface atom form this triangle. The plane of this triangle is placed parallel to the local surface normal. The remainder of the molecule is aligned as in the `normal` mode.

- `double` The molecule is placed with two bonds onto the surface. In contrast to the `bridge` mode, two separate atoms from the ligand form respective bonds to two separate surface atoms. Since the two bond may have different lengths, the four atoms form a general quadrilateral. DISCUS attempts to place the vector between the two molecule atoms parallel The plane of this quadrilateral is placed parallel to the local surface normal. to the vector between the two surface atoms. The remainder of the molecule is aligned as in the `normal` mode.

- `multi` The molecule is placed with bonds between two molecule atoms to surface atoms. The first molecule atom builds two or more bonds of equal length to the corresponding number of surface atoms. The second molecule atoms forms a further single bond to a surface atom. The first atom is thus in a generalized bridge position atop of the surface atoms. If the surface group consists of four or more atoms, it will usually be impossible to obtain equal distances between the molecule atom and all surface atoms. DISCUS attempts to place the atoms with as similar bonds as possible. Once the first molecule atom is placed, DISCUS rotates the molecule to place the second molecule atoms into a position such that it forms a single bond of specified length. The remainder of the molecule is aligned as in the `normal` mode.

- `acceptor` This mode serves to build a hydrogen bond between a surface acceptor and a ligand molecule. The user can specify the distance between the surface acceptor atom and the "Hydrogen" atom in the molecule. This distance should of course be around 1.9Å. This Hydrogen bond is placed normally to the local surface. DISCUS does not check if the atom in the molecule is actually a Hydrogen atom, this responsibility is left to the user. As default DISCUS rotates the molecule to achieve a 170° bond angle in the Hydrogen atom. The remainder of the molecule is aligned as in the `normal` mode. In addition, the molecule is rotated by a random angle around the hydrogen bond.

- `donor` This counterpart to the `acceptor` model places the first molecule atom at the user specified hydrogen bond distance to the surface atom. Again, the distance should be in the range of 1.9Å and the surface atom should be a hydrogen bond. The molecule atom is placed such that the bond angle in the surface Hydrogen atom is by default 170°. The remainder of the molecule is aligned as in the `normal` mode. In addition, the molecule is rotated by a random angle around the covalent bond between the surface hydrogen atom and its closest surface neighbor.

For each of these bonding schemes the user specifies the source of the molecule. DISCUS expects a standard structure file format as source. If your molecule resides in a `CIF` file, please import this into the DISCUS format first. The unit cell parameters of the molecule do not have to be identical to those of the host structure. DISCUS will transform the coordinates internally.

For the bonding schemes `acceptor` and `donor` the hydrogen bond angle in the Hydrogen atoms defaults to 170°. To modify this angle DISCUS provides the optional parameter `angle:value` to the `set bond` command.

DISCUS will place the guest molecules at the user specified density (in molecules per square Ångstroem) onto the surface. This density is roughly estimates using the number of surface atoms and assigning a fixed area of 11Å^2 to each surface atom.

Depending on the bond scheme DISCUS is left with freedom to align the complete molecule. The schemes fix the position of the bonded molecule atoms only. The remainder of the molecule is aligned roughly normal to the surface. The user can specify a pair of atoms within the molecule or let DISCUS find the non-bonded molecule atom at furthest distance to the first bonded molecule atom. Within the degrees of freedom of the bond scheme the molecule is rotated to align the vector defined by these two atoms as close to the local surface normal as possible. For the `normal`, `bridge` and `donor` mode DISCUS can freely rotated the molecule to achieve this orientation. For the `double` and `multiple` mode the molecule will be rotated around the vector between the two bonded molecule atoms. As a result the molecule axis will be in the plane defined by the surface atoms and the two bonded molecule atoms. This plane itself is parallel to the surface normal. The molecule axis will be in this plane but may still form an angle to the surface normal. For the `acceptor` mode, DISCUS can rotate the molecule around the Hydrogen-donor bond to align the molecule axis as close as possible to the surface normal.

As default DISCUS will place the guest molecule onto any surface atom type that occurs in any bond assignment. The user has the option to restrict the placement onto an individual hkl plane or onto a form of symmetrically equivalent planes. Thus, if you shaped the crystal into a polyhedron of say the {100} and {111} faces, you have the option to restrict the placement onto either of the faces or forms.

When you use the `boundary` command within the `surface` menu to shape the original crystal, DISCUS will assign a surface property to those atoms that are close to the surface. Within the `decorate` menu DISCUS uses this surface property to select atoms onto which a guest molecule can be placed. Thus it is mandatory to shape the crystal prior to any decoration. You may desire to build an initial crystal and to assign an external surface without actually removing any atoms. This might be the case when you read a previously created structure that you want to decorate in a second step. In this case, you need to still "cut" the surface. In order not to remove any atoms, simply choose surfaces parallel to the present surfaces and place these new surfaces just a little bit outside the present surfaces. In combination with the `set distance` command in the `surface` menu atoms close to these new surfaces will be flagged as surface atoms and can be used in the decoration mode.

The atoms in the guest molecule receive a "Ligand" and a "Molecule" property, and these properties can be used for further manipulations. The atoms do not automatically receive a surface property. Currently DISCUS does not try to decide which of the guest atoms are at the "outside" in order to assign a correct surface property. It appears the the molecule shape is too impredictable to do so fully automatically. (DISCUS may try to learn this in the future...)

The user can, however, assist by specifying guest molecule atoms that shall inherit the surface properties of the surface atom. Use the command:

```
set example, surface, atom_no
```

With this command, the listed atoms of the ligand molecule will receive the same surface properties as the original surface atom. This will be useful, if you want to place a second layer of decorating molecules onto the surface.

Chapter 7

Analyzing defect structures

After describing how to create and change structures this chapter will give a summary of DIS-
CUS functions to analyze a given disordered structure. All those commands are accessed via
the `chem` sub level of the program. This segment is entered using the command `chem` and left
again by the command `exit`. The command `mode` determines whether the entered commands
operate on atoms (`mode atom`) or molecules (`mode mole`). If working with molecules the
molecule type number replaces the atom name or number as parameter for the different com-
mands. Note that not all commands are available in molecule mode. Also note, that the DISCUS
cookbook (Neder and Proffen, 2007) includes a complete chapter on the analysis of disordered
structures with many examples and exercises.

These commands within the `chem` menu allow a convenient way to analyze the whole struc-
ture. Individual atoms can be checked through the general command language, see section
7.6.

7.1   Occupancies

The command `elem` will display the chemistry of the current model crystal. Depending on the
selected mode the relative abundance of the present atom or molecule types is written on the
screen. The results are saved in the variable `res[i]` (see chapter FORTRAN style interpreter
in the package manual). Entry `res[1]` contains the relative amount of atom types `void` and
entries `res[1]` the relative amount of atom type 1.

The relative abundances given by the command `elem` are for the complete crystal and contain
no further information about its homogeneity. DISCUS allows the user to obtain concentration
as well as correlation (see section 7.3) distributions within the crystal using the command `homo`.
This is achieved by sampling the crystal using a predefined volume.

7.2   Distortions

After analyzing concentrations within a crystal, this section will focus on the analysis of dis-
tortions. The simplest way is to compute average distances between different atom types in a
given range. An example is given below. First the desired range of interatomic distances is lim-
ited to values between 4.5Å and 5.5Å (line 1). Next the bond length distribution is calculated

using the DISCUS command `blen`. Here distances between *all* atoms will be considered and the resulting distribution is written to the file *chem.2.blen* (line 2).

```
1   set blen,4.5,5.5
2   blen all,all,chem.2.blen
```

Rather than using `all` as parameters for the command `blen`, atom names or numbers could be used to calculate the average bond length only for specify atom types.

The command `blen` averages **all** distances within the given range specified with `set blen`. In cases where more specific information about distances in given directions is needed, the command `disp` must be used. Now it is necessary to enter neighbor definitions which can either be specified by a distance criteria or by individual vectors. In the example below the command `set vec` in line 1 defines vector 1 (first 1) to be from site 1 in one unit cell to site 1 (2nd and 3rd parameter) in the next unit cell in x-direction (1,0,0 - last 3 parameters). Next this vector is used as neighbor definition 1 (line 2). Finally the lattice averages are computed between all present atom types (line 3).

```
1   set vec,1,1,1,1,0,0
2   set neig,vec,1
3   disp all,all
```

7.3 Correlations

In this chapter the concept of correlations as a measure for those two-body interactions will be introduced. Although diffuse scattering contains only information about two-body interactions the concepts described here can easily be extended to include multi-site correlations. It should be noted that although these multi-site interactions do not show up in the diffraction pattern directly, they can have a constraining influence on two-body interactions and thus affect the diffraction pattern. However, DISCUS is currently limited to the calculation of two-body interaction averages.

We will talk about atom types in the following section, however, all correlation related commands are available for molecules as well. To work with molecules use the command `mode mole` and specify molecule types rather than atom types or names as parameters for the commands.

Occupational correlations

One definition of the correlation coefficient c_{ij} between a pair of sites i and j based on a statistical definition of correlation (Welberry, 1985) is given in equation 7.1.

$$c_{ij} = \frac{P_{ij} - \theta^2}{\theta(1 - \theta)} \tag{7.1}$$

P_{ij} is the joint probability that both sites i and j are occupied by the same atom type and θ is its overall occupancy. Negative values of c_{ij} correspond to situations where the two sites i and j tend to be occupied by *different* atom types while positive values indicate that sites i and j tend to be occupied by the *same* atom type. A correlation value of zero describes a random distribution. The maximum negative value of c_{ij} for a given concentration θ is $-\theta/(1 - \theta)$ ($P_{ij} = 0$), the maximum positive value is +1 ($P_{ij} = \theta$). Let us look at a simple example:

```
 1  read
 2  stru chem.1.stru
 3  #
 4  chem
 5  #
 6    set mode,quick,periodic
 7  #
 8    set vec,1,1,1, 1, 0, 0
 9    set vec,2,1,1,-1, 0, 0
10    set vec,3,1,1, 0, 1, 0
11    set vec,4,1,1, 0,-1, 0
12  #
13    set vec,5,1,1, 1, 1, 0
14    set vec,6,1,1,-1, 1, 0
15    set vec,7,1,1, 1,-1, 0
16    set vec,8,1,1,-1,-1, 0
17  #
18    set neig,vec,1,2,3,4
19    set neig,add
20    set neig,vec,5,6,7,8
21  #
22    corr occ,zr,void
23  #
24  exit
```

The macro starts with the reading of the disordered structure (lines 1-2). After the `chem` sub-level is entered (line 4) periodic crystal boundaries are selected (line 6). The parameter `quick` selects a faster neighboring finding algorithm which only works for crystals arranged in the DISCUS storage order (see section 2.3). Note that $c_{\langle 10 \rangle}$ stands for the nearest neighbor correlations in all four symmetrically equivalent <10> directions of the two dimensional cubic test crystal, i.e. c_{10}, $c_{\bar{1}0}$, c_{01} and $c_{0\bar{1}}$. All eight neighboring directions for $c_{\langle 10 \rangle}$ and $c_{\langle 11 \rangle}$ are defined as vectors 1 to 8 in lines 8 to 16 of the macro file. Next vectors 1 to 4 are grouped as neighboring definition for $c_{\langle 10 \rangle}$ (line 18) and vectors 5 to 8 for $c_{\langle 11 \rangle}$ (line 20). The command 'set neig,add' (line 19) stores the current neighboring definition and allows the definition of a new one. Finally the correlations for the defined neighboring directions are calculated (line 22). The screen output looks like this:

```
Calculating correlations
   Atom types : A = ZR   and B = VOID

   Neig.     AA          AB          BB       # pairs    correlation
   ----------------------------------------------------------------
      1     50.49 %     48.99 %      .51 %     40000       -.3061
      2     71.04 %      7.89 %    21.07 %     40000        .7897
```

The program lists the probabilities for AA, AB and BB pairs and the corresponding correlations c_{ij}. Here the value for $c_{\langle 10 \rangle}$ is negative, i.e. vacancy neighbors in <10> directions tend to be avoided. Neighboring vacancies in <11> direction on the other hand are much more likely compared to a random vacancy distribution indicated by the large positive value of $c_{\langle 11 \rangle}$.

Displacement correlations

The correlation coefficient c_{ij} for displacement correlations between two sites i and j is defined as:

$$c_{ij} = \frac{\langle x_i x_j \rangle}{\sqrt{\langle x_i^2 \rangle \langle x_j^2 \rangle}} \tag{7.2}$$

Here x_i is the displacement of the atom on site i from the average position in a given direction and $\langle \cdot \rangle$ stands for the average over the crystal. Again a negative value describes a situation where the pairing of corresponding displacements are less likely than in a crystal with random displacements whereas a positive value indicates a larger than random probability. The definition of neighbors is identical to the example in the previous section. Additionally the command `set neig,dir` is used to determine the displacement direction to be used. Note that the displacement direction for the two sites i and j is not necessarily the same, e.g. one could be interested in the correlation between the x-displacement on one site and the y-displacement on the neighboring site.

Correlation fields

In the previous sections a correlation c_{ij} for a given pair of neighboring atoms was computed. An interesting information, however, is how these correlations extend within the crystal. The program DISCUS allows the calculation of correlation fields for occupational and displacement correlations (command: `field`).

7.4   Bond valence sums

The concept of *bond valence* has found wide applicability in solid state chemistry (Brown and Altermatt, 1985; Brese and O'Keeffe, 1991). One application is the use of bond-valence sums at atoms as a check on the reliability of a determined local structure. The valence of an atom i is calculated by the following empirical expression:

$$V_i = \sum_{ij} \exp \left\{ \frac{r_{ij}^0 - d_{ij}}{b} \right\} \tag{7.3}$$

Here r_{ij}^0 and b are the so-called bond valence parameters, d_{ij} is the distance between the central atom i and the neighboring atom j. The sum goes over all nearest neighbors. The bond valence parameters used by DISCUS were taken from a list compiled by I.D. Brown, McMaster University, Hamilton, Canada from various references. Those parameters can be displayed using the command `show`. An example is given below:

```
discus > show bval,zr4+,o2-
Bond valence parameters ZR4+ - O2-  : r0 =  1.9280 b =  0.3700
```

The parameters are specific for a given atom pair, here Zr^{4+} and O^{2-}. Note that it is required to use the atom names indicating the oxidation state like in the example above. It should also be noted, that bond valence parameters are not available for all pairs of atoms.

7.5   Other tools

In this chapter a short summary of functions available in the 'chem' sublevel not discussed previously will be given.
The average structure of a crystal can be calculated using the command `aver`. Occupancies, average positions and standard deviations for these positions are calculated. This command is

not available when working with molecules. The neighborhood of a given atom or molecule can be examined using the commands `neig` and `env`. The command `neig` will use the currently stored neighbor definitions whereas `env` will display **all** atoms found in a given distance from the chosen origin. Finally the conversion between atom index and unit cell / atom site can be made using the equation given in section 2.3 or via the command `trans` in the `chem` level of DISCUS.

7.6 Checking individual atoms

Several commands allow to check the status of an individual atom. A simple command is 'show atom':

```
1 show atom, all
2 show atom, 4
3, show atom, 5, 9
```

The command in line one will show the name, type, positions, properties, etc. for all atoms. The next line does the same for a single atom here number 4, while the thirdd line would show all atoms from 5 to 9. The numbers can of course be given as expressions.
Individual coordinates are accesible through the variables `x[<iatom>]`, `y[<iatom>]`, `z[<iatom>]` that each take the number of the atom to be investigates as parameter between the square brackets.

```
1 evaluate x[4]
2 x[3] = x[3] + 0.03
```

The type of an atom is available though the variable `m[<iatom>]`, which will give you the type number of the atom <iatom>.
The atom name of an individual atom number <iatom> can be read out with the variable `at_name[<iatom>]`. Do not confuse this variable with the rather similar variable `at_type[<type>]`, which returns the name of atoms of type `<type>`.
The `show atom` command does list in which molecule an atom might be included. to work with this molecule number, use the variable `in_mole[<iatom>]`. This variable is the molecule number in which atom <iatom> in included. The value is zero, if the atom is not part of any molecule.
The `chem` menu allows to derive information on the average structure and on displacement fields. To quickly analyze the displacement of a single atom use the command `displacement <iatom>`.

```
1 displacement <iatom>
2 displacement <iatom>, out:yes
3 displacement <iatom>, aver:yes, indi:yes
```

In its simplest form the command will determine the displacement vector of atom <iatom> from its average position and store this vector in the result variable `res[1]` to `res[3]`. The optional parameter "out:yes" will additionally create a simple output on the screen.
It is strongly recommended to determine the average structure within `chem` prior to the use of the displacement command. Otherwise the displacement command may have to repeat this

task for each atom, which can be rather time consuming. If necessary the optional parameter "aver:yes" enforces the calculation of the average structure. If the optional parameter is ommitted or is "indi:no", all atoms of different type that share the same site in a unit cell will be averaged to a common position. If the optional parameter is "indi:yes" atoms of different types will each create their own average position, even if they share the same site in the unit cell. The displacement of atom <iatom> will then be calculated off from the average position of the proper atom type. See the `aver` command in the `chem` menu for more details on the average position.

Chapter 8

Monte Carlo simulation

In general Monte Carlo (MC) methods can be described as statistical simulation methods involving sequences of random numbers to perform the simulation. The MC algorithm goes back to Metropolis et al. (1953). It works as follows: The total energy of the crystal is expressed as a function of random variables such as as site occupancies or displacements from the average structure. The simulation process proceeds as follows: A site within the model crystal is chosen at random and the associated variables are altered by some random amount. The energy difference ΔE of the configuration before and after the change is computed. The new configuration is accepted if the transition probability P given by

$$P = \frac{\exp(-\frac{\Delta E}{kT})}{1 + \exp(-\frac{\Delta E}{kT})} \tag{8.1}$$

is less than a random number η, chosen uniformly in the range [0,1]. T is the temperature and k Boltzmann's constant. Thus the energy of the model crystal in minimized by the MC simulation. Note the a higher temperature T means that more moves will be accepted that lead to a higher total energy E. In order to analyze the defect structure of a particular system, the diffraction pattern of the generated defect structure can be calculated and compared to the experimental data. By adjusting near-neighbor interactions defining the energy E the model can be changed until a match with the experiment is obtained. On the other hand MC simulations can be used to explore the relationship between certain correlations and the corresponding diffraction pattern, e.g. for teaching purposes.

In order to be able to carry out MC simulations, the energy E of the crystal needs to be defined. The following sections give energies for occupational as well as displacement correlations based on the energy for an Ising model. Again readers are encouraged to have a look at the Discus cookbook (Neder and Proffen, 2007) which contains a detailed discussion, examples and exercises on MC simulations using Discus.

> *Note that the current version of* DISCUS *contains a new rewritten module* mmc *which allows to define more complex energies. The old* mc *module is no longer available.*

8.1 Occupational disorder

To represent the distribution of atom or molecule types within a crystal, it is convenient to use Ising spin variables $\sigma_i = \pm 1$. Here $\sigma_i = 1$ means site i is occupied with type A and $\sigma_i = -1$ stands for type B being present on site i. Using these variables, the Hamiltonian (or energy) takes the following form:

$$E_{occ} = \sum_i H\sigma_i + \sum_i \sum_n J_n \sigma_i \sigma_{i-n} + \ldots \tag{8.2}$$

The sums are over all sites i and neighbors n. The value σ_{i-n} refers to the occupancy (spin) of the neighboring site $i-n$ of site i. The quantities J_n are pair interaction energies corresponding to the neighboring vector defined by i and n. Although the Hamiltonian can easily be extended to include multi-site interactions, we will neglect those in this manual. The quantity H is a single site energy which has the effect of an external field in magnetic Ising models. Here it controls the overall concentration.

The interaction energies H and J_n are initially unknown and Discus employs a feedback mechanism to determine their values. This is done in the following way: After a MC cycle, defined as the number of MC steps needed to visit every crystal site once on average, has been carried out, the resulting correlations are computed and compared to the target values defined by the user. If the computed lattice averages are too low, then the corresponding H and J_n are decreased by an amount proportional to the difference between calculated and required value and *vice versa*. It should be noted that even the simplest 2D Ising model possesses a phase transition which is important to avoid when using the described feedback mechanism during a MC simulation. Starting with DISCUS version 5.1 a damping algorithm gradually decreases the changes of the interaction energies within the feed back loop. This reduces fluctuations and provides a better convergence towards the desired correlations.

8.2 Displacement disorder

The MC simulation technique described above to create structures with certain occupational correlations can quite simply been applied to the case of displacements. Displacement correlations were already discussed in section 7.3. The Ising spin variables σ_i are replaces by continuous variables x_i describing the displacement of the atom or molecule on site i. Furthermore we assume that the variable x_i is Gaussian distributed with mean zero ($\langle x_i \rangle = 0$). Thus the Hamiltonian becomes:

$$E_{dis} = \sum_i \sum_n J_n x_i x_{i-n} \tag{8.3}$$

Here, as before, the first sum is over all sites i and the second sum is over all neighbors n of the site i. In this equation there is no term that depends on x_i alone, since this would introduce a shift in the average value $\langle x_i \rangle$. The MC simulation operates as described before. Note that there are two different modes to model displacements `shift` and `swdisp`. Further details can be found in section 10.4.

Note that the displacements x_i are taken in the direction defined by `set neig,dir` and the energy defined in equation 8.3 is *blind* to displacement components in other directions. It is

---

recommended to use the 'swdisp' mode to maintain the overall displacements within the crystal. In cases, however, where the `shift` mode is used, the generated shifts should be restricted to directions corresponding to the correlations desired in the MC simulation. If e.g. the x-displacements of one atom are correlated with the y-direction of neighboring atoms, the shifts should be restricted via the command `set move` to the xy-plane.

8.3 Creating distortions

In the previous section displacement correlations were introduced, but the average displacements remained constant. The modeling of distortions *via* MC simulations works in a similar way. DISCUS offers two different potentials. The first one uses a Hamiltonian (equation 8.4, where the atoms or molecules move in harmonic potentials (Hooke's law).

$$E_h = \sum_i \sum_n k_n [d_{in} - \tau_{in} d_0]^2 \tag{8.4}$$

The sums are over all sites i within the crystal and all neighbors n around site i. The distance between neighboring atoms or molecules is given by d_{in} and the average distance is d_0. The desired distortions are defined by the factor τ_{in}. The value k_n is a force constant for each individual neighbor type n. The second potential is the Lennard-Jones potential :

$$E_{lj} = \sum_i \sum_{n \neq i} \left[\frac{A}{d_{in}^M} - \frac{B}{d_{in}^N} \right] \tag{8.5}$$

with

$$A = D \frac{N}{N - M} \tau_{in}^M \text{ and } B = D \frac{M}{N - M} \tau_{in}^N. \tag{8.6}$$

The sums are over all sites i within the crystal and all neighbors n around site i. The values for A and B are calculated from the target distance τ_{in}, *i.e.* the distance where Lennard-Jones has its potential minimum, and from the potential depth D which must be negative.
A standard Lennard-Jones potential is calculated with the repulsive term and M=12 and the attractive term with N=6. DISCUS allows you to use other exponents as well.
A purely repulsive potential is used to create an even distribution of defects within a crystal:

$$E = -E_\infty + \left(\frac{r - r_{min}}{scale} \right)^M \tag{8.7}$$

Here E_{infty} is the energy between two atoms at infinite distance. As the MC algorithm looks at changes, its value is not relevant. Atoms at distances shorter than r_{min} are placed at essentially infinite energy, for longer distances the energy changes according to the exponent M. A large value of M will cause a steep descent at short distances, but the descent at longer distances might will be less relevant. To extend the effect of the repulsive potential over long distances, choose a large value of `scale`.
Another potential function is the Buckingham potential:
Angular distortions are realized with a potential energy:

$$E = K \left(\Theta - \Theta_{ideal} \right)^2 \tag{8.8}$$

Here Θ_{ideal} is the indented ideal bond angle.

8.4 Working with molecules

Discus allows the use of molecules. The command 'set style, mole' selects the molecule types to be used for the MC simulation and automatically switches Discus to molecule mode. On the other hand 'set style, atoms' will return to atoms mode and select the atom types to be used for the MC simulation.

All neighboring definitions work exactly as for atoms. However, the site label used to define neighboring vectors (`set vec`) still refers to the atom site within the crystal regardless of the current working mode. Thus the user has to check which site of the unit cell is occupied by the origin of the selected molecules and define the vectors accordingly. The use of the 'distance' mode to define neighbors works straight forward, however, this mode is much slower compared to using the vector definitions. All operation modes work on rigid molecules. Note, there are currently no MC (or RMC) moves defining rotations of the molecules. Rotations and other symmetry operations can be realized by creating the wanted different orientations of the molecules as different types using the symmetry segment of Discus (see chapter 4.2) and subsequently using the *swap chemistry* mode.

8.5 Examples

Since the publication of the Discus book Neder and Proffen (2007) the Monte-Carlo part has undergone quite a few changes and updates. The examples in this section illustrate the current possibilities.

8.5.1 Simple Short range order

With this example a simple short range order example will be used to illustrate the main Monte-Carlo concepts as implemented into DISCUS.

The example uses a binary two dimensional crystal. This serves to speed up the simulations compared to a full 3D-model but does not impose a restriction with respect to the possible concepts.

A header of such a simulation might include the following lines:

```
 1  discus                  ! Switch to Discus section
 2  #                       ! Define useful variables
 3  variable integer,size_cr  ! Crystal size in unit cells
 4  variable real,comp      ! composition copper/gold
 5  variable real,c100      ! intended correlation along [100]
 6  variable real,c010      ! intended correlation along [010]
 7  variable real,a100      ! Achieved correlation along [100]
 8  variable real,a010      ! Achieved correlation along [010]
 9  variable real,heat      ! Pseudo temperature
10  #
11  size_cr = 100           ! Set crystal size
12  comp = 0.5              ! Set composition
13  #
14  c100 =  0.75            ! Set desired correlations
15  c010 = -0.75
16  heat =  1.5             ! set pseudo temperature
17  #
18  read                    ! Read asymmetric unit and build crystal
19    cell crystal.cell, size_cr, size_cr, 1
```

```
20  replace cu,Au,all,comp    ! replace cooper by gold on all sites in the
21                            !   unit cell
22  chem
23    set mode, quick, period, xy  ! Set proper periodic boundary conditions
24    element                      ! Check composition
25  exit
26  #
27  set parallel, useomp:parallel  ! Increase speed by parallel calculation
```

This header section does not contain any material that is specific to the Monte-Carlo section. The definitions in lines 11 through 16 are set within this macro. Alternatively you can make the macros more flexible by using parameters $1, $2 etc that are passed down to the macro.

The initial crystal is build in lines 18 and 19, here a simple primitive unit cell with a single Copper atom at 0,0,0 is used. A fraction of the Copper atoms are replaced by Gold atoms in line 20. The initial distribution of these Gold atoms throughout the crystal is random without any correlations between Copper and Gold.

Lines 22 to 25 set periodic boundary conditions along the x- and y-axes, and exclude these along the z-axis, as the crystal dimension along this axis is just a single unit cell. These settings are actually the default upon reading a unit cell that is expanded to a crystal of dimension 1 along any axis.

Line 27 uses an option introduced in Version 6 to use parallel processing in order to speed up the calculation. This is the default behaviour. At the moment several parallel threads will modify the crystal independently during the Monte-Carlo run.

```
 1  mmc                         ! Step into the Monte-Carlo menu
 2  rese                        ! Reset Monte-Carlo to initial state
 3  set neig,rese               ! Reset neighbor definitions
 4  set vec ,rese               ! Reset vectors between atom pairs
 5  #
 6  set vec,1, 1,1,  1, 0, 0    !  The first two vector pairs are for the
 7  set vec,2, 1,1, -1, 0, 0    !  targets
 8  set vec,3, 1,1,  0, 1, 0    !
 9  set vec,4, 1,1,  0,-1, 0    !
10  #
11  set neig,number:1,vec,1,2      ! Define a first neighborhood of vectors 1 and 2
12  #
13  set neig,number:next,vec,3,4   ! Define a next neighborhood
14  #
15  set mode, 1.0, swchem,all      ! Define the modification mode
16  #
17  set targ,1,corr,cu,Au,c100,0.0,CORR  !  target 1 is for [100]
18  set targ,2,corr,cu,Au,c010,0.0,CORR  !  target 2 is for [010]
19  #
20  set cyc, 500*n[1]              ! Define number of modification cycles
21  set feed, 50*n[1]             ! Define number of feedback intervals
22  set temp,heat                 ! Set pseudo temperature
23  set finish, stop:cycles       ! Define termination criteria
24  show                          ! Show current settings
25  run                           ! Start the actual Monte-Carlo run
26  set vec,5, 1,1,  1, 1, 0      ! These next vectors are to check achieved
27  set vec,6, 1,1, -1,-1, 0      ! correlations along additional directions
28  set vec,7, 1,1,  1,-1, 0      !
29  set vec,8, 1,1, -1, 1, 0      !
30  exit                          ! Go back to Discus menu
```

This is the actual Monte-Carlo macro to perform a short range order run. After entering the `mmc` Monte-Carlo menu, the menu is reset to the intial state that is encountered at program start.

Lines 6 to 9 define four vectors for the pair correlations. The first number on the `set vec` line is a numbering of all vector relationsships. The vectors will be referenced under these numbers in lines 11 and 13.

The next two parameters, here both set to "1" refer to the sites within the unit cells for which we want to establish a correlations. To obtain the list of sites within a unit cell, read an asymmetric unit and expand it to a single unit cell with commands like:

```
read
  cell crystal.cell
show atom, all
```

The list of all atoms will give you all sites within the structure.

The last three parameters describe whether the two sites are within the same unit cell or in neighboring unit cells. The command in line 6 establishes a relationship between an atom on site 1 in a chosen unit cell and an atom on site 1 in the neighboring unit cell separated by vector [1, 0, 0]. If there are more than one site per unit cell, you could also specify a command like `set vec, 5, 1, 3, 0, 0, 0` to establish as 5th vector a relationship between an atom on site 1 in a unit cell and an atom on site 3 within the same unit cell.

In this simple example, the four pairs will be immediate neighbors separated by $\pm\vec{a}$ and $\pm\vec{b}$. Lines 11 and 13 group these four vectors into two neighborhoods. The first neighborhood consists of vectors 1 and 2, the second neighborhood of vectors 3 and 4. For all pairs of atoms related by vectors within the identical neighborhood DISCUS will try to achieve identical pair correlations. In the current example, DISCUS will sort the crystal while trying to attempt that all atom pairs separated by either $\vec{a}$ or $-\vec{a}$ will be correlated to each other. The type of correlation will be defined by the targets in lines 17 and 18. Simultaneously with the first correlation DISCUS will sort the atoms to create the desired second correlation along $\vec{b}$ and $-\vec{b}$.

The optional parameter `number:1` or `number:next` defines the neighborhood number and was introduced in version 6.7. The `set target` command in lines 17 and 18 refers to this numbering scheme. In order to enhance flexibility, you can specify the number either as a value (or as an expression) or as the phrase `numer:next`, which will simply increment the number. This increment works for the first neighborhood as well.

Note that DISCUS versions prior to 6.7 required a `set neig, add` line between all neighborhood definitions, but **not** after the last neighborhood definitions. The corresponding macro section would have been:

```
10  #
11  set neig,vec,1,2              ! Define a neighborhood of vectors 1 and 2
12  set neig,add                  ! Add space for a next neighborhood
13  set neig,vec,3,4
14  #
```

Trivially, to sort the crystal, we must modify its structure. DISCUS allows several different modifications to be carried out while *optimizing* the structure. In this example, we only need a single type of modification, the exchange of two atom types. With the
`set mode, 1.0, swchem, all`
command DISCUS is instructed to choose two different atoms at random locations throughout the crystal. The initial energy state is computed and then the two atom types are exchanged. If the new energy state is lower than the inital, the new state is accepted. Otherwise the exchange is rejected with a probability defined by Eq. 8.1.

In this example only one type of modification is used, thus the probability for this move, the second parameter, is set to 1.0. The keyword `swchem` instructs DISCUS to switch the chemistry of the two selected atoms. The last parameter `all` defines that the two atoms my be at any location throughout the crystal. Other values of this last parameter allow to restrict the choice of the two atoms to be in close proximity or to be at the same site within their respective unit cells. See the on-line help for more details.

At the core of any Monte-Carlo simulation lies the choice of the energy that is to be minimized. DISCUS uses the

```
set target, 1, corr, Cu, Au, c100, 0.0, CORR
```

command to achieve this definition. In the macro listed above lines 17 and 18 perform the task. The first numerical parameter refers to the neighborhood defined in lines 11 and 13. The next parameter, in this example `corr`, defines the energy, here a chemical correlation or Occupational disorder as defined in section 8.1. With this correlation type the pair of atom types specified in parameters 3 and 4 `Cu, Au` will be sorted by atom type. If you give a positive value to the correlation parameter DISCUS will sort equal atom types as neighbors within the neighborhood that the current target points to. In this example, the last parameter, the string `CORR` specifies that DISCUS shall sort the structure in order to achieve a correlation value given as parameter 5. Here correlation is specified as variable `c100` to reflect the correlation along [1 0 0]. Any other variable name or an explicit numerical value would of course be valid as well. The 6th parameter `0.0` is the starting value for the Ising interaction energy of Eq. 8.2. Initially the value required to achieve the desired correlation of `c100` is unknown. DISCUS uses a feedback algorithm to determine this energy term. A positive correlation implies a negative value of J and a negative correlation a positive value of J.

As an alternative the command can be specified as

```
set target, 1, corr, Cu, Au, 0.0, e100, ENER
```

The last parameter as string `ENER` tells DISCUS to use a fixed value of `e100` for the Ising energy term J. If the value of this term is positive, a negative correlation will result. The exact value of the final correlation depends on the pseudo temperature used to accept *wrong* moves, see Eq. 8.1.

8.5.2 Sorting molecules

Usually it is faster to sort atoms instead of molecules, as the former involves less atom movements in the crystal than the latter. There are situation, when you should move molecules after all. If for example you want to keep intramolecular distances fixed the best choice is to move the entire rgid molecule.

8.5.3 Shifting molecules

This example will illustrate a rotation of a rigid molecule. The target idea is to adjust the distance bewteen enighboring molecules.

The example uses a crystal composed of linear O-C-O molecules with the linear orientation parallel to the a-axis. The lattice parameter is set a bit to short to accomodate a 3Å distance between two adjecent Oxygen atoms. To relax the structure the molecules are allowed to rotate around a random direction.

```
01 mmc
02   reset
03   set vec, reset
04   set nei, reset
05 #
06   set vect,   1,   2, 3,     1,   0,   0
07   set vect,   2,   3, 2,    -1,   0,   0
08   set vect,   3,   1, 1,     1,   0,   0
09   set vect,   4,   1, 1,    -1,   0,   0
10   set vect,   5,   1, 1,     2,   0,   0
11   set vect,   6,   1, 1,    -2,   0,   0
12   set vect,   7,   1, 1,     0,   1,   0
13   set vect,   8,   1, 1,     0,   1,   0
14 #
15   set neig, number:next, vec,  1, 2
16   set neig, number:next, vec,  3, 4
17 # set neig, number:next, vec,  5, 6
18   set neig, number:next, vec,  7, 8
19 #
20   set target, 1, lennard,  O,  O,    3.000, 10.000,  12, 6
21   set target, 2, lennard,  O,  O,   lat[1], 10.000,  12, 6
22 # set target, 3, lennard,  O,  O,2.*lat[1], 10.000,  12, 6
23   set target, 3, lennard,  O,  O,   lat[2], 10.000,  12, 6
24 #
25   set style, mole
26   set mode,1.0,rotate,all
27   set rota, 1, 5.000, 0.000,  0.000, type:mole
28 #
29   set cycle, 500*n[1]
30   set feed,  100*n[1]
31   set temp,   0.50
32 #
33   run
34 exit
```

Vectors 1 and to connect the Oxygen atoms from one molecule to the neighboring molecule. The remaining vector connect pairs of equivalent Oxygen atoms accross neighboring molecues. iThis ensures that neighboring molecules will end up to lie in parallel albeit tilted orientations and that the the overall periodicity is retained. Accordingly the distance for target 1 is set to 3Å while the other targets are set to the respective lattice parameters.

The main command is set style, mole in line 25, which instructs DISCUS to rotate entire molecules.

Chapter 9

Atomic pair distribution function

The atomic pair distribution function (PDF) can be obtained from powder diffraction data and is a valuable tools for the study of the *local* atomic arrangements in a material. This chapter describes how DISCUS can be used to calculate and refine a PDF. In real space the PDF of a given structure can be calculated using the relation:

$$G_c(r) = \frac{1}{r} \sum_i \sum_j \left[\frac{b_i b_j}{\langle b \rangle^2} \delta(r - r_{ij}) \right] - 4\pi r \rho_0, \tag{9.1}$$

where the sum runs over all pairs of atoms i and j within the model crystal separated by r_{ij}. The scattering power of atom i is b_i and $\langle b \rangle$ is the average scattering power of the sample. In case of neutron scattering b_i is simply the scattering length, in case of X-rays it is the atomic form factor evaluated at a user define value of Q. The default value is $Q = 0$ in which case b_i is simply the number of electrons of atom i.

As this calulation in direct space is an approximation that can be rather crude for X-rays and electrons, see Neder and Proffen (2021), DISCUS uses a different algorithm to calculate the PDF. Once you have build a model structure, use the powder menu to calculate a powder diffraction pattern. Within the output menu DISCUS offers the possibility to write the powder diffraction pattern in may different formats, as well as an option to write the PDF. Essentially DISCUS does the same as you will have done in an experiment. First the sample is subject to the Fourier transform to calculate the powder diffraction pattern and then this pattern is transformed back into direct space via a fast sine-Fourier transform.

9.1 Calculating the PDF

The PDF can be calculated for extended crystals as well as nanoparticles. For extended crystals it is usually faster to use the complete integration mode as described in section 3.2 and in the on-line help. For nanoparticles, the Debye-Scattering-Equation is well adapted.

The first example illustrates the effect of instrumental resolution on the PDF. The powder pattern of Nickel was calculated using the complete integration technique in both cases. Figure 9.1 shows the Nickel PDF and the diffraction pattern calculated with high angular resolution. As is expected, the peaks in the PDF extend with significant height over a large distance range, way beyond the 100Å shown in the Figure. Figure 2 9.2 shows the corresponding pattern for an
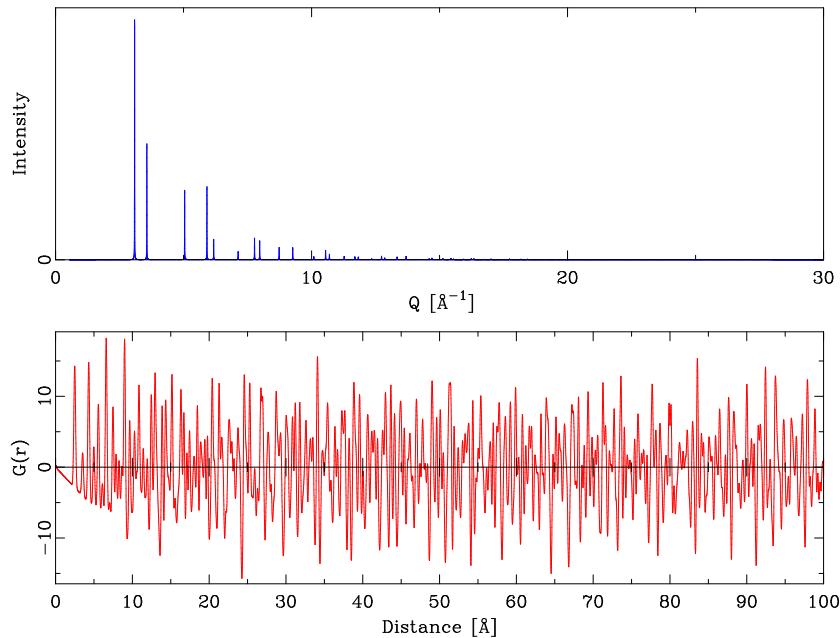
instrument with much broader instrumental resolution, compareable to many PDF beam lines at synchrotron sources. The observed Bragg reflections are much wider and the peaks in the PDF decrease rapidly with increasing distance r.

The effect of a finite Q-range is illustrated in Figure 9.3. Here the powder pattern were calculated to Q_{max} = 50, 20 and 15Å^{-1}, respectively. For the smaller Q_{max} values the width of the PDF peak increases and the finite Fourier ripples become more pronounced.
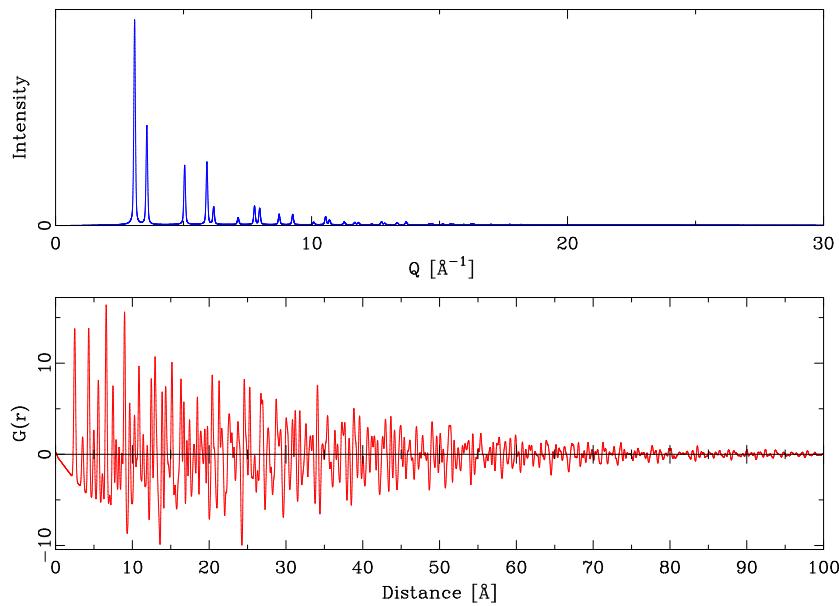
```
 1 variable real, P_u
 2 variable real, P_v
 3 variable real, P_w
 4 variable real, P_eta
 5 variable real, P_qmax
 6 P_u = 0.0000
 7 P_v = 0.0000
 8 P_w = 0.0050
 9 P_eta = 0.5000
10 P_qmax = 20.000
11 #
12 read
13 cell CELL/nickel.cell
14 #
15 powder                         # Switch to powder menu
16    reset
17    xray                        # Select X-rays
18    set axis,q                  # Perform calculation on equaly spaced Q grid
19    set calc,complete           # Use complete integration algorithm-algorithm
20    set disp,off                # Switch anomalous dispersion off
21    set delta,0.0               # Set simple convolution by Gaussian off
22    set qmin,0.5500             # Starting value for Q
23    set qmax,P_qmax             # Final value for Q
24    set dq,  0.0010             # Step size for Q
```
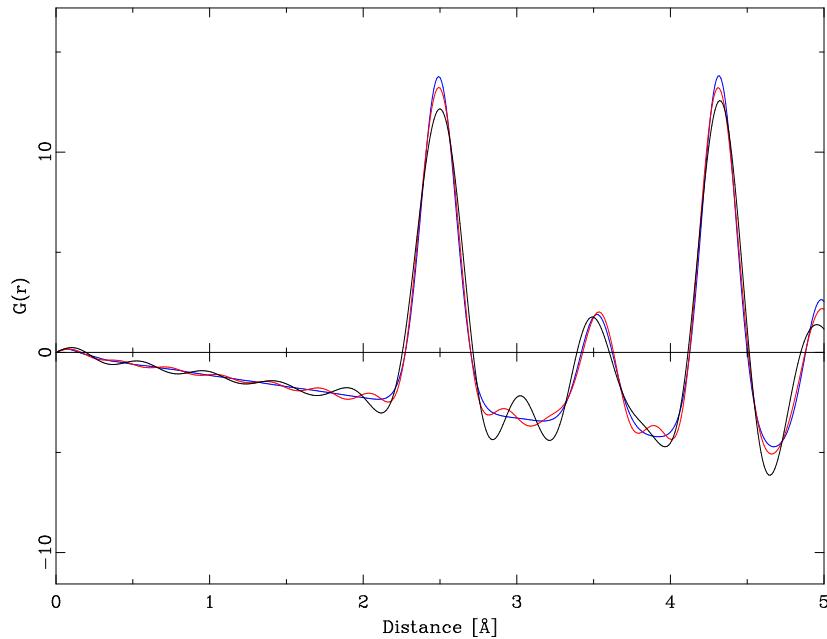

Figure 9.1: Calculated PDFs of Ni. High angular resolution was assumed for the powder pattern.

```
25   set dh, 1.0                  # Step sizes HKL
```


Figure 9.2: Calculated PDFs of Ni. Broad angular resolution was assumed for the powder pattern, as commonly used at PDF beam lines.


Figure 9.3: Calculated PDFs of Ni. Effect of different Q_{max} values. Blue: 50Å^{-1}; red: 20Å^{-1}; black: 15Å^{-1}.

```
26   set dk, 1.0
27   set dl, 1.0
28   set profile, off          # Switch convolution by Pseudovoigt function off
29   set profile, pseudo       # Use Pseudovoigt (instead of Gaussian)
30   set profile, uvw, P_u, P_v, P_w # Cagliotti u,v,w values
31   set profile, eta, P_eta   # Mixing parameter 1=Lorenzian 0=Gaussian
32   set profile, asym, 0.00, 0.00 # Asymmetry parameters
33   set temp,use              # Use the Atomic displacement parameters
34   set wvle,0.20000          # Set the wavelength
35   set four,four             # Just for stacking fault mode, set to normal
36   set lpcor,bragg, 2.00     # Define Lorentz-Pol to BraggBrentano Diffractometer
37   run                       # Do the actual calculation
38 exit                        # Go back to main DISCUS menu
39 #
40 output                      # Switch to output menu
41   value inte                # Select intensity as output value
42   outf  "POWDER/complete_%52F.inte", P_qmax   # Define output file name
43   form  powder,q            # Write output as powder data
44   run                       # Perform the actual output
45
46   value PDF                 # Select intensity as output value
47   outf  "PDF/complete_%5.2F.grcalc", P_qmax   # Define output file name
48   form  pdf, r,0.01, 100.0, 0.01  # Write output as powder data
49   run                       # Perform the actual output
50 exit                        # Go back to main DISCUS menu
```

The macro file used to calculate the Nickel PDFs is listed above. In the first 10 lines, the variables for the profile function are defined and given appropriate values. As the calculation used the complete integration, a single unit cell is sufficient (lines 12/13). Lines 15 to 38 define the powder diffraction pattern. The pattern is calculated for xrays (line 17), using the complete integration mode, line(19) Since we have a single unit cell steps in reciprocal space are limited to intger HKL, lines 25-27. The variable P_qmax is used in line 23 to set the upper Q limit. The powder pattern is convoluted with a Pseudo-Voigt function, lines 28 to 32. In this example a constant eta value is use, but note that DISCUS allows a linear and squre Q dependence of eta as well. Noasymmetry is used in this simple example.

Lines 40 to 50 write the output. Lines 41 to 44 write the intensity on a Q-scale. Here the Q-max parameter is used to create a flexible file name.

To write the PDF, DISCUS uses the output menu as well. In line 46 the output is defined to be a powder PDF `value PDF`. Capitalization of `PDF` is mandatory. Line 48 defines the output format to be a PDF on a distance scale in (optional) limits from 0.01Å to 100.0Å in steps of 0.01Å. At the run command in line 49, DISCUS will internally perform the transformation of the powder pattern into a powder PDF.

9.2   Refining a PDF

To refine an experimental model against a PDF, you can use the two refinement algorithms build into the DISCUS SUITE . Alternatively the Reverse Monte Carlo (RMC) algorithm can be used as well.

Essentially you would create the model structure based on a parameter set, calculate the powder diffraction pattern and write the PDF using the output menu as decribed in the previous section.

! to be revised !, example to come

Chapter 10

Reverse Monte Carlo

This chapter gives a short introduction into the RMC level of DISCUS. DISCUS can be used to either refine the scattering intensities directly of to refine the atomic pair distribution function (PDF) (see chapter 9). A more detailed description of the various commands can be found in the reference manual or the online help function.

10.1   Introduction

The **R**everse **M**onte **C**arlo (RMC) method (McGreevy and Pusztai, 1988) is another application of the Monte Carlo algorithm discussed in chapter 8. Here, rather than minimizing the total energy of the crystal, the difference between observed and calculated intensity is minimized. Although the method has been around for about 10 years, the method was first applied to *single crystal* diffuse scattering in a neutron diffraction study on ice *Ih* (Nield et al., 1995).
The RMC process starts also with the selection of a random site and changing its variables like occupancy or displacement by a random amount. The scattering intensity is recalculated for the generated move and the goodness-of-fit parameter χ^2 as given in equation 10.1 is computed.

$$\chi^2 = \sum_{i=1}^{N} \frac{[I_e(\mathbf{h}_i) - I_c(\mathbf{h}_i)]^2}{\sigma^2} \tag{10.1}$$

The sum is over all measured data points $\mathbf{h}_i$, I_e stands for the experimental and I_c for the calculated intensity. The change in the goodness-of-fit is given by $\Delta\chi^2 = \chi^2_{old} - \chi^2_{new}$. Every move which improves the fit to the data ($\Delta\chi^2 < 0$) is accepted. Those moves which worsen the fit ($\Delta\chi^2 > 0$) are accepted with a probability of $P = exp(-\Delta\chi^2/2)$. The parameter σ is assumed to be independent of $\mathbf{h}$ and is treated as a parameter of the modeling. The value σ can be identified with the temperature T in the (direct) Monte Carlo method described in chapter 8. How many *bad* moves are accepted depends on the value of σ or T.

10.2   RMC in more detail

For practical use it is necessary to include a scaling factor f and a background parameter b in the definition of the goodness-of-fit χ^2. A weight $w(\mathbf{h})$ is included as well. DISCUS allows the

user to choose a particular weighting scheme or to read weights from a separate input file. The definition of χ^2 used in the program is given in equation 10.2.

$$\chi^2 = \sum_{i=1}^{N} \frac{w(\mathbf{h}_i)[I_e(\mathbf{h}_i) - (f \cdot I_c(\mathbf{h}_i) + b)]^2}{\sigma^2} \tag{10.2}$$

As in the previous section $I_e(\mathbf{h}_i)$ stands for the measured intensity at the reciprocal point $\mathbf{h}_i$, and $I_c(\mathbf{h}_i)$ is the calculated intensity in that point. The summation is over all N experimental data points. The value σ is a parameter of the modeling and controls the fraction of *bad* moves which are accepted. The corresponding parameter in (direct) Monte Carlo simulations is the temperature T.

Three different ways to calculate the scale f and background b are implemented. First the user can define fixed values for both: $f = f_0, b = b_0$. Secondly, the background can be set to a fixed value $b = b_0$ and the scaling factor f is computed according to equation 10.3.

$$f = \frac{\sum_{i=1}^{N} w(\mathbf{h}_i)I_e(\mathbf{h}_i)I_c(\mathbf{h}_i) - b_0 \sum_{i=1}^{N} w(\mathbf{h}_i)I_c(\mathbf{h}_i)}{\sum_{i=1}^{N} w(\mathbf{h}_i)I_c^2(\mathbf{h}_i)} \tag{10.3}$$

Alternatively both values f and b can be refined during the RMC refinement. Equation 10.4 shows the corresponding definitions.

$$f = \frac{\sum_{i=1}^{N} w(\mathbf{h}_i) \sum_{i=1}^{N} w(\mathbf{h}_i)I_e(\mathbf{h}_i)I_c(\mathbf{h}_i) - \sum_{i=1}^{N} w(\mathbf{h}_i)I_e(\mathbf{h}_i) \sum_{i=1}^{N} w(\mathbf{h}_i)I_c(\mathbf{h}_i)}{\sum_{i=1}^{N} w(\mathbf{h}_i) \sum_{i=1}^{N} w(\mathbf{h}_i)I_c^2(\mathbf{h}_i) - \left(\sum_{i=1}^{N} w(\mathbf{h}_i)I_c(\mathbf{h}_i)\right)^2}$$

$$b = \frac{\sum_{i=1}^{N} w(\mathbf{h}_i)I_e(\mathbf{h}_i) - f \cdot \sum_{i=1}^{N} w(\mathbf{h}_i)I_c(\mathbf{h}_i)}{\sum_{i=1}^{N} w(\mathbf{h}_i)} \tag{10.4}$$

The scaling factor which the program prints on the screen during the RMC refinement is actually (for some yet unknown reason) $1/f$. The parameters f and b are computed during each RMC cycle and usually have large starting values as long as there are big differences between calculated and observed data. After every RMC move the resulting scattering intensity and the χ^2 value is calculated. In order to save computing time only the contribution of the modified atoms to the scattering is calculated. The difference $\Delta\chi^2 = \chi^2_{old} - \chi^2_{new}$ is taken to decide if the move will be accepted or not. If $\Delta\chi^2 < 0$ the agreement between calculated and measured data has improved and the move is accepted. Moves which result in a $\Delta\chi^2 > 0$ are only accepted with a probability of $P = exp(-\Delta\chi^2/2)$. As the value of $\Delta\chi^2$ is proportional to $1/\sigma^2$, the value of σ has an influence on the amount of *bad* moves which will be accepted. Obviously there are two extremes: For very large values of σ, the experimental data are ignored ($\chi^2 \approx 0$) and with very small values of σ the fit ends up in the local minimum closest to the starting point, because there is a negligible probability for *bad* moves. In order to be more independent of the actual

number of data points used, the goodness-of-fit parameter used in the program is given by $\chi^2/\sum w(\mathbf{h}_i)$. The program calculates separate scaling factors and background parameters for every used plane of experimental data. This allows to simultaneously use data measured with X-rays and neutrons, different wavelengths or from different instruments. The goodness-of-fit χ^2 is displayed as its total value and separate for each data plane.
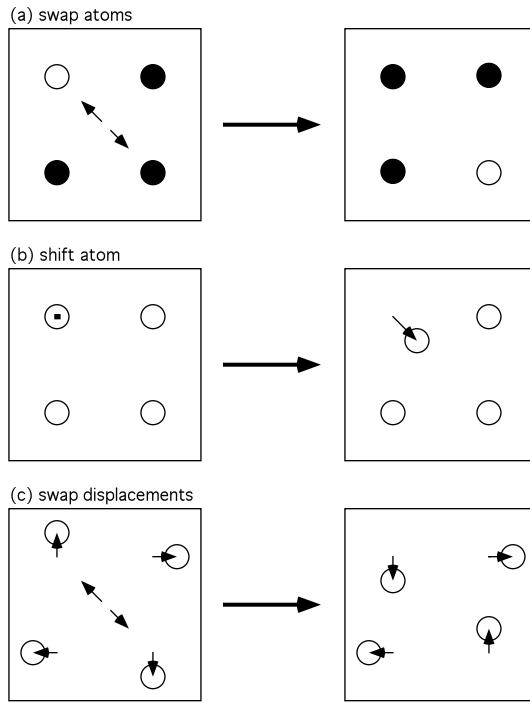
10.3 Setting up a model crystal

The first step in an RMC refinement is the creation of a model crystal of suitable size. In many cases the starting structure will be the (known) average structure for the compound under investigation. Since certain information of the crystal (e.g. symmetry) is used in the RMC segment, it is advisable to **to set the crystal before entering the RMC segment**. Depending on the kind of disorder to be modeled it might be necessary to introduce displacements according to the temperature factors (command 'therm', see section 6.1) or create the needed amount of vacancies before starting the RMC refinement. How to generate a crystal is described in chapter 2, tools to modify the crystal are discussed in chapters 5 and 6.

10.4 Operation modes

So far changes to the crystal made during the RMC refinement were simply called 'moves'. These moves can either be a displacement of an atom or the change of the occupancy of an atom site. Because the relative abundance of the elements is not allowed to change during the simulation, the later move is actually made by switching the atoms of two sites within the crystal. The program knows three different operation modes which involve three different kinds of moves shown in figure 10.1. Additionally user defined moves can be included in an external subroutine linked to the program DISCUS. A short description of the different RMC operation modes is given in the list below.

1. *switch chemistry*: This mode (Fig. 10.1a) allows to simulate occupational disorder by selecting two different atoms randomly and switch these two atoms. This operation mode can obviously work only, if at least two different types of atoms are present within the crystal. This mode is selected by the command `set mode, swchem`.

2. *shift atom*: If this mode (Fig. 10.1b) is set, a randomly selected atom is shifted by a random amount in a random direction. The size of the generated shift is chosen uniformly in the interval [-1,1] unit cell. The actual shift applied to the atom is the generated shift multiplied with a user defined factor ('set move,<x>,<y>,<z>'). These factors are also given in unit cell units. The shift mode is selected by with `set mode, shift`.

3. *switch displacements*: This mode (Fig. 10.1c) is selected by the command `set mode, swdisp` by swapping the displacement, i.e. the difference between the average and the actual position, of two randomly selected atoms of the same type and thus the overall average displacements remain constant in contrast the previous mode. The user has to make sure that initial displacements are present in the starting structure (e.g. thermal displacements e.g. by using the command `therm`.

(a) swap atoms

(b) shift atom

(c) swap displacements

Figure 10.1: RMC operation modes of DISCUS

4. *external*: The program DISCUS allows the user to define more complex RMC moves via an external subroutine. This subroutine is defined in the file *extrmc.f90*. For more details about the construction of such a subroutine, read the commented example in the file *extrmc.f90* which is part of the distribution.

The program allows to select (`sele <typ1>, <typ2>, ..`) and deselect (`dsel <typ1>, <typ2>, ..`) atom types which should be taken into account during the RMC simulation. Alternatively molecules (if present) can be selected using the commands `msel <typ1>, <typ2>,..` and `mdes <typ1>, <typ2>, ...` Here <typ> is the corresponding molecule type. All modes listed above can be used for these rigid molecules as well. Rotations and other symmetry operations can be realized by creating the wanted different orientations of the molecules as different types using the symmetry segment of DISCUS (see chapter 4.2) and subsequently using the 'swap chemistry' mode. After every generated move the minimal allowed distances (`set mdis, <atom1>, <atom1>, <dist>`) between all selected atoms are checked and if atoms are too close, the move is rejected.

10.5 Running RMC

Finally, the experimental data need to be read before the RMC refinement can start. The file format for the experimental input data is similar to the output formats KUPLOT and *PGM* for the Fourier transform. It might be necessary to remove Bragg peaks and other unwanted scattering (e.g. powder rings from a sample holder) from the input data set. With the `data`

command, the method (neutron or X-ray), weighting scheme and the corners of the input data
set in reciprocal space are entered. More than one plane of experimental data can be read by
repeating the 'data' command. After the data have been read, select the desired RMC mode,
select the appropriate atoms and start the refinement with the `run` command.

```
   Gen:   1200 try:    352 acc: (good/bad):    62 /     0 s2x2:   11829643.
    Plane  1: scal:  14.83    / back:  22.23    / s2x2:            11829643.
```

The screen output is updated in a user defined interval, an example output is shown above.
The fist line gives the number of generated moves ('Gen') and how many of those were ac-
tually tested ('try'). The difference is due to selecting atoms that should not take part in the
refinement and moves that violate minimal atom distances. The next two numbers give the
number of good ($\Delta\chi^2 < 0$) and bad ($\Delta\chi^2 > 0$) moves that have been accepted. The last num-
ber is the current value of the overall χ^2 for all data planes. Additionally the scaling factor f
and the background b are given for each data plane (one in this example) followed by χ^2 for
the corresponding data plane. The resulting structure as well as the calculated scattering can
be saved with the `save` command.

Chapter 11

Experimental data treatment

The information on the capabilities of DISCUS to treat experimental data for further analysis are found in the manual on the experimental data treatment EXPERI .

Appendix A

DISCUS commands

A.1   Summary

```
News       ! Summary of recent changes
append     ! Append atom to model crystal
asym       ! Show the contents of the asymmetric unit
boundary   ! Cuts off atoms along an hkl plane
branch     ! Switches to KUPLOT, if within the suite
change     ! Changes the character of an object
chem       ! Enter CHEM level (see chem level for details)
connect    ! Enter CONNECTIVITY level
copy       ! Copy atom or molecule within the crystal
export     ! Exports the structure in a different format
decorate   ! Enter  DECORATE level
demolec    ! Enter demolecularization level
d2r        ! Transform vector from real to reciprocal space
define     ! Define DISCUS specific settings
diff-four  ! Enter DIFFERENCE FOURIER transform level
domain     ! Enter the DOMAIN level
exp2pdf    ! Transform experimental powder pattern into PDF
find       ! Find atoms around a given point
four       ! Enter FOURIER transform level
import     ! Imports other formats into DISCUS cells
inser      ! Insert atoms or enter INSERT level
inverse    ! Enter inverse Fourier transform level
kick       ! Insert atom/molecule and possibly remove old atom/mole.
mmc        ! Enter MONTE-CARLO simulations level
output     ! Enter output level (save results)
patterson  ! Enter PATTERSON level
pdf        ! Enter PAIR DISTRIBUTION FUNCTION level
perioditize ! Transform a structure into a period ic array
plot       ! Enter PLOT level (export crystal for plotting)
powder     ! Enter POWDER diffraction level
project    ! Calculate projections in real and reciprocal space
property   ! Enter the PROPERTY level
purge      ! Remove voids from crystal (DO NOT USE !)
r2d        ! Transform vector from reciprocal to real space
read       ! Read structure or unit cell from file
remove     ! Remove atom or molecule from crystal
replace    ! Replace atom(s) or molecule(s) with other type
reset      ! Reset DISCUS to initial program start conditions
rmc        ! Enter REVERSE-MONTE-CARLO level
save       ! Enter SAVE level to save the structure
seed       ! Set seed for random number generator
shear      ! Enter the SHEAR level
```

```
show         ! Show various information
spacegroup   ! Sets the space group for the current structure
stack        ! Enter STACKING FAULT level
storage      ! Enter STORAGE level
super        ! Enter superspace level
switch       ! Swap two atoms or molecules within crystal
symm         ! Enter SYMMETRY transformation level
transform    ! Enter unit cell TRANSFORMATION  level
therm        ! Displace atoms or molecules according to B
vprod        ! Calculate vector product in real or reciprocal space
waves        ! Enter WAVE type modulations level
wyckoff      ! Shows symmetry operations of the space group
```

A.2 News

Here you find a list of recent changes, additions, bug corrections

2023_August

chem Added an optional parameter "site:" to homo powder Improved handling of corrlin and corrquad for a powder PDF plot BUG FIX, Avoiding crash if jmol had never been used before

2023_July

mmc Improved speed of the molecule version surface Changed the names of the optional ellipsoid parameters: long ==> third short ==> first Added the optional cylinder parameter axis Warnig this is not backward compatible.

2023_June

plot Added options to place Jmol window output Added better scaling options to "shelx", "hklf4", "list5" files export Improved storage of original space group into CIF file read/cell BUG FIX, multiple molecule were not properly recognized after a save show/symm Added a flag to indicated generator matrices

2023_May

plot Added a "posit:" optional parameter to the plot => run command

2023_April

export Added an optional style command to "scatty"

2023_March

fourier Started implementation of a Non-Uniform FFT output Started implementation of a transformation menu upon output

2023_February

save/read Added a format specifier. Files with this format can be read much faster fourier BUG FIX for grid that did not include 0,0,0 in reciprocal space, the phase of data points was off by half a grid point. fourier Preparation for major update mmc Added an option to display initial status output Added an option to transform data upon write read Added a new optional parameter to offer more detailed control over the assignemnt of atom types stack BUG FIX, Layers with many different atom types could cause a crash

2023_January

mmc Added an option to control amount of output during / after mmc loop powder Added a Pearson Type VII profile function stack BUG FIX: Improved atom modulo function. Corrected possible crash if 2nd layer had more atoms than previous

2022_November

GENERAL Corrected wave length CuKa1, 2 to 1.54092 and 1.544343 according to Mendenhal et al. J.Physics B (2017), 50, 115004 DOI 10.1088/1361-6455/aa6c4a powder BUG FIX Corrected calculation of S(Q), F(Q), PDF for complete mode powder BUG FIX Asymmetry correction was permanently off. powder BUG FIX Kalpha 1/2 correction was slightly off. export Enforced updatas of crystal dimensions

2022_October

stack BUG FIX Layers with differing chemical composition were not recognized

2022_September

stack BUG FIX Stacking faults with 'create' and no 'run' caused crash, FIXED stack BUG FIX if modulus was off, yet modulo vector were set, these were erroneously added to the layers. FIXED stack Added option to apply the modulo function to atoms as well, creating a smoother surface plot BUG FIX Modifies cif file output to adhere to CIF standard, unique atom labels This caused Jmol versions 14.32.59 and later to make faulty plot. export BUG FIX Corrected super cell dimensions Corrected file name for version 7 Corrected internal BUG that prevented multiple use. purge Improved estimation of crystal size perioditize BUG FIX Correctes estimate of unit cell numbers (y dimension was wrong)

2022_July

mmc BUG FIX, "set disallow, all" did not work Added a new modification mode "swneig" to exchange neighbors still under development Improved on "coordination number" energy, added this to help still under development

---

2022_June

bragg New command to convert between parameters in Braggs law powder Added new Time of flight neutron profile function Modified interperation of uvw parameters, added a new style "fwhm:poly".

2022_May

mmc BUG FIX, A 'reset' did not set all shifts to zero deco BUG FIX, added a 'mmc reset' to avoid unexpected siide effects to mmc powder Modified the profile parameters uvw to give the same FWHM irrespective of the axis used in the calculation. Values correspond to FHWM with respect to a 2Theta scale. Added FWHM function for powder diffraction with an area detectors as in Chernyshov et al. ActaCryst A77, 497-505, doi 10.1107/S2053273321007506 powder BUG FIX corrected Lorentz correction for complete mode

2022_April

perioditize BUG FIX, help did not work export BUG FIX, command could not be reached BUG FIX, Added error messege to 'shelxl' if crystal > 1 unit cell Modified unit cell size to avoid atoms at coordinates == 1.0 import BUG FIX, increased flexibility for RMC6f formats BUG FIX, shelxl import did not recognize SYMM transform BUG FIX, did not run if new lattice parameters were given

2022_March

EXP2PDF Continued development, improved limit selection, fine tuned graphics
EXP2PDF improved handling of data and background of various lengths Corrected error related to qscale PERIODITIZE New menu to transforma structure into a periodic array, possibly with inclusions of voids EXP2PDF New menu to transform an experimental powder diffraction pattern into a powder PDF.
DOMAIN enabled "orient' and 'shape' to handle format specifiers PLOT enabled a 'run kill:yes' line, just to kill all previous JMOLs POWDER FIXED BUG Corrected corrlin, corrquad action in powder menue FIND enabled format specifier for atom names FOURIER FIXED BUG, in some circumstances the average intensity output was missing the Friedel pairs. FOURIER Introduced an automatic scaling between Bragg and diffuse intensities that applies to a full 3D-PDF only. Not relevant for 3D-Delta-PDF or plain intensities. MMC FIXED BUG, Feedback algorithm was always on, even for target "corr" with last parameter set to "ENER". MOLEC-ULARIZE Improved the command by adding optional parameters. READ Improved input speed for larger structures SURRFACE FIXED BUG, orientation of ellipsoids did not follow instructions
Atomic Form factors Corrected values for the Elements:, International Tables Table 1.1.1.3 Gd xray a4 Sm xray b4 Cr electron a5 Mo electron a2 H1- xray b3 was wrong in ITC D1- xray b3 was wrong in ITC Ru4+ xray b3 was wrong in ITC Bi5+ xray b2 was wrong in ITC Implemented Waasmaier & Kirfel Parameters

2022_Jan

Added a new menu ==> 'perioditize'

2021_Dec

Atomic form factors Some values for the following elements were wrong: Gd a4 Sm b4 Cr a4 Mo a2 The parameterized values according to Waasmaier & Kirfel, ACA (1995) A51, 416-431 were added and are used as default values. Optional parameters table:waas or table:inter allow to choose the Waasmaier or the International Tables Volume C. DOMAIN enabled "orient' and 'shape' to handle format specifiers PLOT enabled a 'run kill:yes' line, just to kill all previous JMOLs POWDER FIXED BUG Corrected corrlin, corrquad action in powder menue FIND enabled format specifier for atom names FOURIER FIXED BUG, in some circumstances the average intensity output was missing the Friedel pairs. FOURIER Introduced an automatic scaling between Bragg and diffuse intensities that applies to a full 3D-PDF only. Not relevant for 3D-Delta-PDF or plain intensities. MMC FIXED BUG, Feedback algorithm was always on, even for target "corr" with last parameter set to "ENER". MOLECULARIZE Improved the command by adding optional parameters.

2021_Oct

Changed the reading of structures to enable files without BISO collumn.
SURFACE Added an optiona parameter center:<value> to the 'boundary' command. Added the option to set the value to "com" as in "center of mass" to all "cent*:com" parameters FIXED BUG Corrected the determination of a periodic boundary PDF calculated via Debye-Scattering-Equation

2021_Sept

MMC added an optional parameter "log:screen" to the 'set finish' command.

2021_June

Added a LAMMPS import Moderate improvements for plot to handle charged atoms Continued improvements within MMC Added improved output to 'show env' Added a file "private.f90" and a 'private' command to let the user develop own algorithms

2021_May

Depreciated the "set neig, add" command and added an optional parameter "number:value" to the individual "set neigh, ..." commands.

2021_April

Added an output format option 'vesta', that replaces 'xplor' This allows better control of the 3D visualization including an overlay by a calculated Patterson map.

2021_February

Corrected setting of anomalous dispersion coefficients. Added optional parameters to ==> 'index2cell' regarding output and periodic boundary conditions Added an option to accumulate several diffraction pattern prior to output ==> fourier/set Added an option to apply symmetry averaging to a diffraction pattern ==> fourier/set

2020_Dezember

Added "xplor" output file format for diffraction data. Added a 'range' command to output to go along with the "xplor" file format.
Corrected import of RMCprofile version 6 files
Added not on side effect to mmc ==> set ==> vec
Added an optional paarameter to the value command in 'output'. Allows to limit reciprocal-space prior to 3D-PDF calculations.

2020_November

Added a linear and square (TTH/Q)-dependence of asymmetry parameters to powder == 'set profile'
Corrected a bug that prevented the execution of macros at sub-levels

2020_October

Added support for file name constructors to ==> 'import' and ==> 'export' File names may now make use of constructors like: "NiceFile%4D.cif", 5 ==> "NiceFile0005.cif"
Corrected a bug that prevented a direct read of imported structures.

2020_September

Added finish criteria to mmc. These allow to terminate a MMC loop upon reaching several convergence/stagnation criteria.
Added 'show scat' command to Fourier and powder menu.
Added optional "geom:[<px>,<py>]" to ==> 'plot/run' command

2020_August

Added extensive parallel computing throughout the code

2020_July

Added a new energy "unid" to mmc. This is essentially the same as "corr", the SRO chemical short range order energy, except taken strictly unidirectional from the center to the neighbor. It works best with a "vector" environment in a locally noncentrosymmetric environment.
Improved speed of Fourier for the special case of a block of LxMxN pixels that are centrosymmetrically around the origin. All L,M,N must be odd and anomalous scattering be off.

2020_May

Added more parameters to the surface => boundary command to specify the orientation of a cylinder / ellipsoid.

2020_April

Many major updates:
Changed the PDF calculation to run via a "detour" through the powder diffraction. Makes the calculation exact for the X-ray case in in many circumstances much faster.
Added atomic masses
Added options to add multiple powder diffraction pattern.
Improved the internal precision of most variables.

2019_December

Added wave length symbols ′CU12′ etc that enable Kalpha1/Kalpha2 profile functions

2019_November

Added an optional parameter "metric:host" to the ==> import command.

2019 September

The powder calculation was thoroughly revised. Pattern calculated with the ′complete′ mode are put on a better scale, proportional to the number of atoms in the structure. A continuous background from uncorrelated thermal vibrations has been added. Added option parameter "distri" to ==> ′set bond′ command within the ==> ′deco′ menu.

2019_August

Added optional parameter "type" to purge command. Allows to clear list of atom types.
Added the option to add the surface vectors to the crystal structure file.

2019_May

Added the option to retain the original space group to the ==> ′export cif′ command Added further output options to menu ==> ′output′.

2019_Apr

Added an export option to write the structure in the format for Scatty.
powder Improved scaling of intensities to be independent of a profile function.

2019_Feb

Implemented all alternative settings for the orthorhombic space groups. See the entry ==> "Unit_cell" for details on how to define alternative settings and the entry ==> "space_groups" for a complete listing of all non-standard settings in the orthorhombic system.
Added a command 'displacement' that returns the displacement vector of an atom from its average position.
Added the surface vector to the 'show atom' command

2019_Jan

Added a demolcularize menu to delete a molecule entry while keeping the atoms
surface Added a thickness: optional parameter that allows to define individual thicknesses to hkl planes.

2018_Dec

mmc The displacement correlation did not really work, fixed.
CIF interpreter Further improvements to interpret CIF files. For unknown space groups DISCUS attempts to use the symmetry operations.

2018_Nov

m[<number>] If an atom is removed, its scattering curve is set to zero and the property 'normal' was cleared. If you manually change the scattering curve back to a non-zero value, the properties were not set back to 'normal' Fixed.
save enforced that "scat", "adp" and "occ" and either all written or all omitted.
powder For neutron diffraction, the output of the $<f>{**}2$ contained an error. The negative scattering length were added as |b|. Corrected.
mmc Added a 'set valid, <invalid_moves>' command, that gives you more control over a termination criterion.
CIF interpreter Added more flexibility to recognize non standard monoclinic space groups that are not listed as full Hermann-Mauguin symbol

2018_Oct

In ==> 2017_July the following "correction" was done, which proved to be false. The ==> 'therm' command erroneously used too large a displacement. The average $<u{**}2>$ were actually 3 times too large.
As of version 5.27.2 this has been corrected yet again to the original, correct state. A calculation of a powder /single crystal pattern either with "temp use" or a combination "therm and temp ignore" will now produce identical results again.
Added a 'show res' to the 'fourier' and 'connectivity' menu.
Improved the memory handling in pdf, makes the program use much less memory. Connected to this, all menus reduce memory usage through the local 'reset' command. The 'reset' command sets all default values within a menu back to program start and initializes all internal

arrays to the smallest default size. Most of the time this will not bee needed. The main effect will be within the Fourier menus (fourier, powder, output). The 'pdf' menu is reduced as much as possible automatically.

2018_Sep

Added new export options: to CIF export cif, <filename>
to RMCprofile version 6 and 7 export rmcprofile, <filename> [,version:7]
to VASP, POSCAR file export vasp, POSCAR export poscar, POSCAR
If atom types exist, for which there is no more actual atom in the crystal, the form factors are set to zero. This allowes you to define pseudo atoms for the domain without the need to use the 'fourier/scat' command to set dummy scattering functions.
purge, stack, domain, deco Turned on stricter behavior for the handling of periodic boundary conditions and the "chem" quick mode off. It can be turned back on with after explicit 'set crystal' command in the chemistry menu.
added a 'reset to the 'chem' menu added a 'reset to the 'domain' menu added a 'reset to the 'fourier', 'inverse', difffour', patterson' menu added a 'reset to the 'insert' menu added a 'reset to the 'pdf' menu added a 'reset to the 'shear' menu added a 'reset to the 'save' menu added a 'reset to the 'stack' menu added a 'reset to the 'symmetry' menu added a 'reset to the 'transform' menu added a 'reset to the 'waves' menu added to the global 'reset: most arrays, chem, deco, domain, fourier routines, insert, mmc, pdf, property, rmc, save, shear, symmetry, transform, waves

2018_Aug

Added a 'reset' command to DISCUS. (Work in progress) Added a ==> 'storage' menu. This menu allows to display and manage the internally stored crystal structure files.

2018_July

Added the possibility to define property like features that are based on the connectivities ==> property
Fixed a bug that did not allow a read free for origin choice 2 Added an 'exit' command to the read menu.
Added a 'reset' command to 'plot'
connectivity Modified the behavior for central atoms given by name instead of type number. Connectivity will perform the requested action for all atom types of this name. add Si, O, 1.2, 2.0, silicon Creates a connectivity for all Silicon atom types.

2018_June

Added the possibility for interactive plots. See ==> plot/back ==> plot/poly ==> plot/bond ==> plot/run
Revised the reaction to a CTRL-C
Added a ==> 'set error, ... , "save" option
functions

Added a new logical function that queries the properties of an atom.

decoration Added a "chelate" bonding scheme and corrected the descriptions for the "bridge" scheme. Added the missing descriptions to the "acceptor" and "donor" schemes.

2018_May

surface Corrected a BUG related to the location of cylindrical surfaces.

read / import Improved handling of CIF files with atom names like "Ti4+" Improved handling of CIF files with atom lines that contain just dots Ti Ti 0.00 0.00 0.00 . 1.0

2018_April

remove molecules Changed the behavior if whole molecules are removed. The molecule type is still set to zero and the atoms inside the molecule are switched to 'voids'. The molecule status of the individual atoms is, however, retained. This allows you to still work with these "removed" molecules, in strict analogy to removed atoms. The full molecule entry disappears with a 'purge'

occupancy An 'occupancy' was added as further atom property. The philosophy behind DIS-CUS is a simulation of an actual crystal structure, thus the occupancy is really just an emergency measure for the simulation where you want just very few or a single unit cell. As side effect the format of the 'cell' file has been augmented. For details see ==> 'read'

read / import Improved handling of CIF files that do not have an empty line following a 'loop'. Import the occupancy.

2018_Mar

save As of version 5.17.1 DISCUS saves the atoms in its internal sequence, even if molecules are involved. The molecule info is written into the file header and further columns have been added to the atom list that specify in which molecule an atom is located.

stack A simplified list of origin types has been added to the 'create' command. This list "internal.stacksimple.list" contains the atom types "L001", etc at reduced origins starting at 0,0,0 and incrementing with [0,0,1]. This list facilitates correlation analysis.

read A comment with the read menu caused an exit, changed to ignore the comment.

read / import Improved the reading of CIF files. Attempt to replace ion charges with the DISCUS standard as best as possible. If the space group is not given or is a '?', the symmetry operations are read instead.

2018_Jan

The logical comparisons may now take the operators: $<$, $<=$, $==$, $/=$, $>=$, $>/$ The classical fortran77 operators are still valid

New logical functions "isvar" and "isexp" can be used within an "if" construction. See help entry ==>'function' in the general "Command_lang" section.

Variable "mol_type[$<i>$]" has been made read/write

stack The ==> 'create' command copies the list of layer origins into an internal file with fixed name "internal.stacklist.stru"

2017_Nov

Added RMCprofile version 6 format to imports

Added an "export" command, currently just a "shelx" export format

Added a choice to the ==> 'symmetry/mode' command Added the possibility to use a space group symmetry matrix. Related is the new variable sym_n[1] that stands for the number of symmetry operations in the present space group.

Added optional parameter 'identical:' to read /cell command

Added a command 'spacegroup' that sets the space group.

The 'wyckoff' command returns the multiplicity of the site and the space group symmetry matrices that copy the site onto itself.

2017_Oct

Space Groups Added the space groups Aem2=Abm2 (39) Aea2=Aba2 (41) Cmca=Cmce (64) Cmma=Cmme (67) Ccca=Ccce (68)

Atom names Instead of atom names or numbers, the user can specify a user defined variable. variable character, string string = 'Si' ... sel Al, string, O ... replace Al,string,...

connectivity New optional parameters allows to specify if the connectivity is to be restricted to the same molecule as the central atom.

stack The number of layers for each type is recorded into the result variable.

surface The boundaries have improved and corrected behavior for internal hollow spaces created by the "outside" flag.

The boundary commands create a new variable that reflects the surface character and the direction of the normal to the local surface.

2017_Sep

Throughout the program the internal calculation of random numbers was changed to the FOR-TRAN 90 intrinsic function.

2017_July

The decoration of ligands onto surfaces can now be restricted to specific faces.

The ==> 'therm' command erroneously used too large a displacement. The average $<u**2>$ were actually 3 times too large.

2017_June

Atom "AL3+" had been misspelled internally.

The 'hkl' command in 'fourier' can now handle format strings for the file name.

2017_May

The boundary command at the top level menu and inside the surface menu can now create a polyhedron from a form of symmetrically equivalent hkl planes as well as a triaxial ellip-

soid. The command at the main discus level is depreciated and you are encouraged to use the (identical) command at the surface menu.

2017_April

The surface menu received a new command "char" that determines the surface character of an atom.

2017_March

The 'read' ==> 'cell' and 'stru' commands have been changed to accept CIF, CMAKER and RMCprofile formats as well.
The 'molecularize' command will add atoms to a previous molecule, if the first atom is inside a molecule.

2017_Feb

Within the powder menu, the energy of the radiation may also be specified via ==> 'set energy, <value>'

2017_Jan

A new command 'first' was added to 'stack' that allows to determine the first layer type
An unfortunate typing error in News/2016_Oct regarding the new refinement variable ref_para[1...] (was misspelled as ref_param[1...]) is corrected in the on-line help.

2016_Dec

At a few select points colors are introduced into the output. Currently these are just the error messages.
The new "molecularize" command allows to group atoms into molecules

2016_Oct

The output of the "blen" and "bang" commands within "chemistry" is now written to KUPLOT directly, if the filename starts with "kuplot"
The data stored by ==> 'chemistry' ==> 'aver' have been changed in a non-backwards compatible way. DISCUS now stores 9 values per atom: the site number, the atom type on this site, the position, the standard deviation of the average position and the occupancy.
If DISCUS detects more than one "data_" sections in a CIF file, the second etc sections are written to a separate file, augmented by a number
Global variables have been introduced that use the same syntax as user defined variables. This include just "pi" and variables related to the refinement. DIFFEV sets the value to these variables: REF_GENERATION Current generation REF_MEMBER Current population size REF_CHILDREN Current children size REF_DIMENSION Number of parameters REF_KID

Current child Updated for DISCUS and KUPLOT only REF_INDIV Current individuum Updated for DISCUS and KUPLOT only ref_para[1..] Current trial parameters for current child Insertion of a new atom will automatically turn off the periodic boundary conditions. Periodic boundary conditions in combination with the exact search mode have been enabled. A new menu is introduced that allows to group atoms into a molecule. The symmetry commands 'uvw' and 'orient' have been improved to allow to specify a pair, respectively one atom to define the axis/origin The symmetry command can now rotate partial molecule groups A new 'recreate' command allows to rebuild the connectivity list for a specific atom type while leaving the remainder intact.

2016_June

The mmc 'set move' command may now take a further parameter. This will actually restrict the movement of the specified atom along the direction of a vector. DISCUS may now be interrupted gracefully with a CTRL-c. This will cause DISCUS to write the current structure as a file called EMERGENCY.STRU.

2016_May

Added option to the save menu to utilize the atom properties.

2016_March

Within the ==> Fourier menu, the radiation may also be defined via its ==> 'energy' instead of its wavelength ==> 'wvle' Domains with an explicit shape (cube, sphere, cylinder) can now be generated with a size distribution. Fourier can calculate the intensities for a SHELXL file The 'find env' command will now return the neighbors sorted by distance. 2D and 2D images can be written in MRC file format The ==> 'add' command within the ==> 'connectivity' menu can now take a new optional parameter that restricts the connectivity to the closest N atoms.

2015_Dec

Added a command 'apply_symmetry' to the chemistry and mmc menus. These allow to generate symmetry equivalent correlation vectors. See mmc and chem ==> set vec; set neig; apply_symmetry for more help Added the option to calculate normalized intensities for single crystal pattern.

2015July_A

You can now write powder files as normalized scattering function $S(Q)$ or as reduced normalized scattering function $F(Q) = Q[S(Q)-1]$

2015July

With the release of the DISCUS_SUITE several new capabilities have been introduced.
Within the DISCUS_SUITE the output files can be written directly into KUPLOT. Simply start the file name with the fixed string 'kuplot'. The data set number in kuplot will be incremented automatically.
DISCUS contains the new ==> 'branch' command that allows you to change to a KUPLOT or DIFFEV section.

2015February

The speed of the PDF and powder(Debye) calculations has been improved considerably. No side effects on the user.
DISCUS does check if an atom in the unit cell file has a name "void" and it will ignore these atom types when calculating the PDF. This helps to get the proper PDF density and weights for molecules where the first atom is a void at the center of the molecule.

2014November

DISCUS will now import CIF files, see ==> import

2013September

The chemical short range order in mmc can now sort groups of atom types.

2013June

Finally this is it! DISCUS is now a fortran2008 program. Most of the substantial changes should not affect the user (or so we hope). Most of the large arrays are now allocated automatically, as needed, thus the need to compile DISCUS with different size versions should no longer exist.

Command_line_options

The command line may now take the additional option: program -macro <macro_name> [<par1> [<par2> ...]]

Electron

DISCUS now offers electron diffraction in the kinematic limit as well. Atom form factors are taken from Table 4.3.2.3 Int. table Vol C (2006) respectively. from Peng, Ren, Dudarev, Whelan Acta Cryst A52 (1996), 257
Electron diffraction works in all menus where you could switch between X-ray and neutrons i.e.: fourier, powder, rmc, pdf

internal_storage

DISCUS now offers an internal storage of crystal structures. Simply start the file name with the string "internal" and DISCUS will write the crystal structure into a dynamically allocated internal data structure, instead of onto the hard disk. If a structure was written into an "internal" file, it can be read as well with ==> 'read'.

mmc

The mmc menu can now offer a new "repulsive" energy, and may use the connectivity list to find interacting neighbors

powder

In previous versions, DISCUS would write an unevenly spaced powder pattern if you used a Q-axis and a 2Theta output or a 2Theta axis and a Q output. This has now been replaced by an evenly spaced output. If you use a Q-axis with proper Q-range, you also need to specify the 2Theta limits on which you want to write the data. Likewise for a 2Theta axis, you need to specify a Q-range for the output. This change mostly affects the DEBYE equation. Internally, DISCUS always uses an evenly space q-grid, as the calculation is faster. With this bug fix, the output will be evenly spaced, for all combinations of axis and output.

connectivity

This new menu allows you to define settings for the connectivity list. This connectivity list is used to quickly access neighboring atoms.

Centering

In previous versions, DISCUS used the centering generators before the symmetry generators. To be more in line with the International Tables, DISCUS uses now first all symmetry generators, followed by the centering generators. See also the ==> 'define' command to adjust the sequence to your personal taste.

waves

The 'mrepl' and 'repl' commands were augmented by a "vice versa" option that will cause atoms/molecules to be replaced by each other rather than only in one way.

2010Sep

property

A major upgrade of DISCUS adds properties to atoms. These can be N = normal, the atom is a normal atom (instead of a void) M = molecule, the atom is part of a molecule D = domain, the atom is part of a domain O = outside, the atom is outside of the crystal E = external, the atom is close to an external surface I = internal, the atom is close to an internal surface

---

The new ==> 'property' menu allows global settings, plot uses its local settings. The global settings affect 'replace', 'mmc', 'find env' Commands that add, replace, remove atoms all affect the properties.

surface

The new 'surface' menu covers the old boundary command and sets options to flag the atom property near an external or internal boundary.

domain

A new command 'distance' similar to the corresponding one in the 'surface' menu allows to set distances of atoms to internal boundaries. Atoms closer than these distances are flagged as "close".

plot

The default was changed to plot the entire crystal, even if the 'ext' or 'thick' commands had not been used.

2010_Aug

atom_coordinates

The user may now specify the fractional coordinates in a unit cell or a structure file as algebraic expression as in the example:
Si 1.-0.2, 1/3, zpos, bval
Remember, if you use a variable as "zpos" or "bval" in this example, the variable must have been defined up front and been given a sensible value.
To use this feature, it is mandatory, that the fractional coordinates are separated by a comma.
The old style is still possible.

unit_cell_dimensions

The user may now specify the unit cell dimensions in a unit cell or a structure file as algebraic expression as in the example:
cell alat, 2*alat, 1.0+4.0, 90.000, beta, gamma
Remember, if you use a variable as "alat" or "beta" in this example, the variable must have been defined up front and been given a sensible value.
To use this feature, it is mandatory, that the unit cell dimensions are separated by a comma.
The old style is still possible.

2008may

atom_sequence

In the original source code, the generators for each space group were applied strictly in the sequence as listed in the header of the International Tables.

Thus for centered space groups, each atom was followed immediately by those created by the centering translations. The one exception are those rhombohedral space groups described by hexagonal axes that have a horizontal 2 fold axis and/or a center of inversion. Here the sequence of rhombohedral centering vectors is mixed up.

To correct this, DISCUS now allows the user to choose whether centering generators are applied first or last. The second, new sequence, will correspond more clearly to the printing of symmetry operations for centered space groups as given in the International Tables. Currently the old sequence is still the default, this will be reversed in future releases. Presently the sequence can be defined by ==>

```
define generator, {"center" | "symmetry"}
```

which defines which group of generators comes first.

molecules

A major issues regarding the assembly of molecules was corrected. In several space groups, molecules were not assembled correctly. See ==> "data keywords molecule" for further details

symmetry

A new entry in the 'show' menu displays all symmetry operations of the current space group in a variety of formats

wyckoff

This is a new command at the main menu level that determines the local site symmetry.

define

This new command defines several settings, just like the ==>"set" command does. The "set" command at the main menu level is reserved for general settings, common to all programs in the DIFFUSE suite. "define" is specific to the individual program.

2008_Apr

formfactors

Within the "fourier" menu, you can define your own values for the atomic form factors as well as anomalous dispersion correction terms. A "reset" was added to switch back to the internal values.

stacking faults

The new parameter "list" added to the distribution mode, allows you to interpret the coordinates of atoms read from the input file as positions of the respective layers. Previously, the parameter "file" interpreted the atom type only

boundary

A cylindrical boundary has been added.

A.3 manual

```
manual ["section:"{"suite"  | "discus"  | "diffev" |
                   "kuplot" | "package" | "mixscat"}
       [,"viewer:"<name>]
```

Opens a PDF viewer for one of the Manuals
The section defaults to the current program section that you are working with. On Linux systems, the viewer defaults to "qpdfview", on Windows system it defaults to "firefox". If DISCUS does not find the default or the user provided viewer, DISCUS will search a list of common PDF viewers. If none is found an error message points to the folder that contains the manuals.

A.4 Menu

The DISCUS section is structured into several menus that are designed to modify the structure, to calculate the diffraction pattern etc. These menus are:
Read/save a structure read ! Read structure or unit cell from file or internal storage import ! Import a crystal structure from various formats save ! Save the structure to file or internal storage plot ! Write the structure in a format for plotting perioditize ! Rearrange the atom sequence in a periodic fashion
Modify a structure decorate ! Decorate the surface with ligands demolec ! De-molecularize a molecule domain ! Place domains into host structure insert ! Insert objects into the crystal mmc ! MONTE-CARLO simulations rmc ! REVERSE-MONTE-CARLO simulations stack ! Build a crystal with stacking faults symm ! Perform a generalized symmetry operation shear ! Perform a general shear/affine operation super ! Apply disordered superspace waves ! Apply modulation wave modifications transform ! Describe the structure with a different unit cell
Information chem ! Analyze a structure connect ! Build a connectivity table property ! Set / clear reaction to properties storage ! List / manipulate internally stored structures
Fourier calculations powder ! Perform powder diffraction calculations fourier ! Perform single crystal diffraction calculations rmc ! REVERSE-MONTE-CARLO simulations inverse ! Perform an inverse Fourier transformation diff-four ! Calculate a difference Fourier map patterson ! Calculate the Patterson function output ! Write Fourier results to disk
pdf ! Calculate a Pair distribution function

A.5 Unit_cell

The unit cell format adopts a key word controlled style of the data file. Except for the lines containing the atoms, each line contains a keyword in the first few characters, followed by the parameters needed for this keyword. The first line MUST contain the keyword "title", the sequence of the other keywords is not relevant. The last keyword MUST be "atoms" which starts the list of atoms. You may include comments or empty lines anywhere between the "title" and "atoms" keyword.
The keywords "title", "spcgr", "cell" and "atoms" are required, all other keywords are optional. The first character after the keyword must be a blank ot TAB. All following parameters must be separated by a comma. White spaces between the parameters may be either blanks or TABs. Example

```
title Single atom structure of NA
spcgr Pm-3m
symm  1.00,0.00,0.00,0.123, 0.00,1.0,0.00,0.00, 0.00,0.00,1.00,-0.567
cell  5.0,5.0,5.0,90.,90.,90.
atoms x,    y,    z,  Biso, Property, MoleNum, MoleAt, Occ
NA1+  0.0   0.0   0.0 0.1
NA1+  1/3, 1-0.5, 0.0, 0.1
NA1+  1/3, 1-0.5, 0.3, 0.1,  1
NA1+  1/3, 1-0.5, 0.3, 0.1,  1,  0,  0,   1.00
```

Further help topics are:

title

```
title <title string>
```

The title can be any character string up to 80 characters long

space

```
spcgr {<symbol> | no. } [, "2"]
```

[, setting:{'abc'| 'bac' | 'cab' | 'cba' | 'bca' | 'acb'}
The space group symbol is used to generate all atoms within the unit cell. The symbols used should be the Hermann-Mauguin symbols used in Int. Tables Vol.A. A center of inversion should be given as "-" sign immediately preceding the axis. Lattice types need to be given as capital characters, mirror planes as small characters.
Monoclinic cell choices 2,3 or unique c-axis will be assumed if the corresponding non standard full Hermann-Mauguin symbol is used.
Origin choice 2 is indicated by setting the optional second parameter to "2".
The lattice constants are used to distinguish between rhombohedral and hexagonal settings.
The space group symbol is checked for contradictions with the lattice constants. In case of error, the unit cell is not read.
The optional "setting:" parameter can be used in conjunction with an orthorhombic space group to describe a different setting. As an example you can use the space group # 28 Pma2 in the setting bac to get the non-standard space group Pbm2 with: spcgr Pma2, setting:bac Equivalently,

all non-standard orthorhombic settings can be used with the corresponding non-standard space group name. In this case omit the "setting:" parameter: spcgr Pbm2
In both cases, the atom coordinates x,y,z must be provided in the sequence for the non-standard setting.
See the DISCUS manual and the International Tables for Crystallography Volume A for a complete listing of all orthorhombic settings.

cell

```
cell  <a>,  <b>,  <c>, <alpha>,  <beta>,  <gamma>
cell  <a>   <b>   <c>  <alpha>   <beta>   <gamma>
```

The first form with comma separating the lattice constants is encouraged. Only with this format, can you use a variable instead of fixed unit cell values.
Alternatively, the lattice constants are read in free format <a b c alpha beta gamma> All six constants must be given.
The space group symbol is checked for contradictions with the lattice constants. In case of error, the unit cell is not read.

atoms

```
<name>  <x>, <y>, <z>, <B> , <propertyflag>, <MoleNo>, <MoleAt>, <occ>
<name>  <x>, <y>, <z>, <B> , <propertyflag>
<name>  <x>  <y>  <z>  <B>
<name>  <x>  <y>  <z>
```

If a unit cell is read, only those atoms in the asymmetric unit should be listed. For each of the atoms a line must be given with the name of the atom, the fractional coordinates and the isotropic thermal coefficient B. The property flag is optional. This is, however, automatically written to a saved structure file. The <MoleNo> and <MoleAt> specify in which molecule the atom is and at which entry within the atom is. The <occ> specifies an occupancy for the atom. The name (in capital characters) must be left bound in the first columns prior to a blank or TAB. The atom name and the coordinates x,y,z are mandatory. The optional parameters should be listed on the 'atom' keyword line.
The fractional coordinates and the B value can be in free format, if the lines contains no comma. In this case the property flag must be omitted. Otherwise, the fractional coordinates and the isotropic thermal coefficient B can also be specified as arithmetic expressions.
Optionally charged atoms can be symbolized by e.g. "NA1+". See the help entry 'atom names' for a complete list of atoms for which scattering curves are supplied.
Empty lines in the file are ignored.

keywords

The keywords "title", "spcgr", "cell", and "atoms" are required, see the help one level up. These further optional keyword allow you to fine tune the unit cell file.
You may include commentaries or empty lines anywhere between the "title" and "atoms" keyword.
Further help topics are:

# #

`#<comment>`

You can provide commands for the data file by including lines that start with a "#". The content of these lines is ignored.

title

`title <title string>`

The string used as title can be any character string up to 80 characters long

spcgrp

`spcgr {<symbol>, <no>} [,"2"]`

The "spcgr" keyword defines the space group of the crystal. The space group is used to generate all atoms within the unit cell. The symbols used should be the Hermann-Mauguin symbols used in Int. Tables Vol.A. A center of inversion should be given as "-" sign immediately preceding the axis. Lattice types need to be given as capital characters, mirror planes as small characters.
Monoclinic cell choices 2,3 or unique c-axis will be assumed if the corresponding non standard Hermann-Mauguin symbol is used.
Origin choice 2 is indicated by setting the optional second parameter to "2".
The lattice constants are used to distinguish between rhombohedral and hexagonal settings.
The space group symbol is checked for contradictions with the lattice constants. In case of error, the unit cell is not read.

scat

`scat <name> [,<name>...]`

By explicitly listing the atom names on a 'scat' command line, you can define a specific sequence of scattering curves that is independent of the sequence of atoms found in the structure file.
If the structure is read using the 'stru' or 'lcell' commands, DISCUS compares the names and atomic displacement parameters (B-values) to those found on the 'scat' and ==> 'adp' command lines. If an atom name and adp match, the atom is assigned to the scattering curve, otherwise an additional new scattering type is defined. If you read a unit cell with the 'cell' command, the scattering type read from any 'scat' and ==> 'adp' line will define the first scattering types. All atoms that follow will be assigned to new additional scattering types, regardless of their name and adp, even if they match the values from the 'scat' and ==> 'adp' command.
This command is needed if you want to create a list of layer type origins which are read by ==> 'stack'.
You can use more than one 'scat' and ==> 'adp' command in the structure file. The number of atoms on the individual 'scat' and ==> 'adp' lines does not matter, as long as the total number

of atom types and adp's is identical. Atom names and adps's are matched pairwise using the sequence with which they were read from the 'scat' and ==> 'adp' lines. Starting with DISCUS 4.0, the number of atom names on the 'scat' and 'adp' command lines must be identical. Thus, both commands must be present.

adp

```
adp <B-value> [,<B-value>...]
```

By explicitly listing the atomic displacement parameters (adp) on a 'adp' command line, you can define a specific sequence of scattering curves that is independent of the sequence of atoms found in the structure file.

If the structure is read using the 'stru' or 'lcell' commands, DISCUS compares the names and atomic displacement parameters (B-values) to those found on the ==> 'scat' and 'adp' command lines. If an atom name and adp match, the atom is assigned to the scattering curve, otherwise an additional new scattering type is defined. If you read a unit cell with the 'cell' command, the scattering type read from any ==> 'scat' and 'adp' line will define the first scattering types. All atoms that follow will be assigned to new additional scattering types, regardless of their name and adp, even if they match the values from the ==> 'scat' and 'adp' command.

This command is needed if you want to create a list of layer type origins which are read by ==> 'stack'.

You can use more than one ==> 'scat' and 'adp' command in the structure file. The number of atoms on the individual ==> 'scat' and 'adp' lines does not matter, as long as the total number of atom types and adp's is identical. Atom names and adps's are matched pairwise using the sequence with which they were read from the ==> 'scat' and 'adp' lines. Starting with DISCUS 4.0, the number of atom names on the 'scat' and 'adp' command lines must be identical. Thus, both commands must be present.

generator

```
generator g11,g12,g13,g14, g21,g22,g23,g24, g31,g32,g33,g34 [, power]
```

You can define additional generators through the optional "generator" keyword. These generators act identical to the generators defined through the space group symbol. All previously generated copies of the atom in the asymmetric unit are copied by this generator, and will in turn be copied by any generators following later.

The optional parameter power specifies whether this generator is to be applied only once (default) or whether the generator is to be applied again to the first image. A generator for a three fold axes must be applied once to copy atom x,y,z to -y,x-y,z and then again to copy atom -y,x-y,z to -x+y,-x,z.

Since these additional generators are applied after the space group generators, you can use these generators to create non-standard groups or to create a set of symmetries that does not from a group.

The generator forms the matrix algebra:

(g11 g12 g13) (x) (g14) (g21 g22 g23) * (y) + (g24) (g31 g32 g33) (z) (g34)

The generators:

```
gene 1,0,0,0.5, 0,1,0,0.5, 0,0,1,0.0, 1
gene 1,0,0,0.5, 0,1,0,0.0, 0,0,1,0.5, 1
```

for example would create the following copies of an atom at 0,0,0:

```
0  ,0  ,0
0.5,0.5,0.0
0.5,0.0,0.5
0.0,0.5,0.5
```

symmetry

`symmetry s11,s12,s13,s14, s21,s22,s23,s24, s31,s32,s33,s34 [, power]`

You can define additional symmetry operations through the optional "symmetry" keyword. These symmetry operations act different than the generators defined through the space group symbol or listed as additional generators. The symmetry operations copy only those atoms created by the generators. The symmetry operations do not act on copies of the atoms created by previous symmetry operations.

The optional parameter power specifies whether this symmetry operation is to be applied only once (default) or whether the symmetry operation is to be applied again to the first image. A symmetry operation for a three fold axes must be applied once to copy atom x,y,z to -y,x-y,z and then again to copy atom -y,x-y,z to -x+y,-x,z.

Since these additional symmetry operations are applied after the space group generators, you can use these symmetry operations to create non-standard groups or to create a set of symmetries that does not from a group.

The symmetry operation forms the matrix algebra:

(s11 s12 s13) (x) (s14) (s21 s22 s23) * (y) + (s24) (s31 s32 s33) (z) (s34)

The symmetry operations:

```
symm 1,0,0,0.5, 0,1,0,0.5, 0,0,1,0.0, 1
symm 1,0,0,0.5, 0,1,0,0.0, 0,0,1,0.5, 1
```

for example would create the following copies of an atom at 0,0,0:

```
0  ,0  ,0
0.5,0.5,0.0
0.5,0.0,0.5
```

cell

```
cell  <a>,  <b>,  <c>, <alpha>,  <beta>,  <gamma>
cell  <a>   <b>   <c>  <alpha>   <beta>   <gamma>
```

The first form with comma separating the lattice constants is encouraged. Only with this format, can you use a variable or an arithmetic expression instead of fixed unit cell values.

Alternatively, the lattice constants are read in free format <a b c alpha beta gamma> All six constants must be given.

The space group symbol is checked for contradictions with the lattice constants. In case of error, the unit cell is not read.

All six constants must be given.

The space group symbol is checked for contradictions with the lattice constants. In case of error, the unit cell is not read.

ncells

```
ncells  nx, ny, nz, ncatoms
```

This command tells DISCUS how large the crystal is in terms of unit cells along the x,y and z axis. This is useful, if you have stored the crystal created by a previous simulation and want to continue work on this crystal. In order for fast references between atom number and the location of its corresponding unit cell, DISCUS must now how many unit cells were created by the original ==> 'read/cell' command. <ncatoms> is the number of atoms in each unit cell. You can determine this number by typing the command "eval n[3]".

If you saved the structure, using the keyword controlled format, see help on the 'save' menu, you can tell DISCUS to save the number of unit cells and number of atoms per unit cell values for you.

atoms

```
<name>  <x>, <y>, <z>, <B> [, <propertyflag>]
<name>  <x>   <y>   <z>   <B>
```

If a unit cell is read, only those atoms in the asymmetric unit should be listed. For each of the atoms a line must be given with the name of the atom, the fractional coordinates and the isotropic thermal coefficient B. The property flag is optional. This is, however, automatically written to a saves structure file.

The name (in capital characters) must be left bound in the first four columns. The fractional coordinates and the B value can be in free format, if the lines contains no comma. In this case the property flag must be omitted. Otherwise, the fractional coordinates and the isotropic thermal coefficient B can also be specified as arithmetic expressions.

Optionally charged atoms can be symbolized by e.g. "NA1+". See the help entry 'atom names' for a complete list of atoms for which scattering curves are supplied.

Empty lines in the file are ignored.

molecule

```
molecule [parameter]
```

This keyword is allowed anywhere between the atoms of the unit cell file. It marks the beginning of a group of atoms that are grouped to form a molecule. Optional parameters are used to specify further details. The first 'molecule' keyword must have no parameters.

Molecules are used in DISCUS for three purposes. Regular molecules define a group of atoms that may be moved around as a group. The do not have to form a molecule in strict chemical sense but are just a group of atoms linked for convenience. Objects define a three dimensional object used to simulate small angle scattering. The pseudo atoms within the "object" describe the position, orientation, and distortion of the "object". Domains are representations of domain origins ==> 'domain'. They allow you to manipulate just the positions, shape, and size of the domains. Once the positions are fixed, you can insert the structure represented by the domain into the host crystal. Valid parameters are:

end

```
molecule end
```

The parameter "end" signals the end of a molecule. All atoms still listed in the unit cell file are treated as individual atoms.

type

```
molecule type,<type>
```

The molecule is of type no. <type>. All molecules of the same type can be handled together by several commands like the symmetry menu.

generator

```
molecule generator
```

This parameter has become obsolete!
DISCUS determines the internal symmetry of the molecule by determining the Wyckoff site symmetry of the first atom within the molecule.
The first atom of any molecule that lies on a symmetry element of the space group must be located at the point of highest symmetry of the molecule. If the structure does not have an atom at this site you must include a "void" on this site. This could be the case .e.g. if you have an empty triangle on a threefold axis.
This site symmetry is taken as the internal symmetry of the molecule.
DISCUS compares the lists of atoms created by the space group and by the internal molecule symmetry. Identical sections are linked to one molecule.
Atoms created by other symmetry operations (like lattice centering...) will form a new molecule of the same type.
See the section on site symmetry in the International Tables for further details.

symmetry

```
molecule symmetry
```

This parameter has become obsolete!

character

```
molecule character,{"atoms"|"cube"|"cylinder"|"sphere"}
molecule character,{"domain_cube"|"domain_cylinder"|"domain_sphere"}
```

A molecule may either represent a list of real atoms, or represent an extended object. The distinction is given through the molecule character. If the character is set to "atoms", the molecules consists of true atoms, and may contain many different atoms. You should calculate the Fourier transformation using the ==> 'fourier' command ==> 'set internal'.
Alternatively, the molecule may represent extended objects. They are considered to consist of continuous matter, without any further internal structure. The boundary is a step function, i.e. outside there is nothing, inside is continuous matter of constant density. Their Fourier

transform is calculated using analytical expressions. For each of the available characters, a specific Fourier transformation is calculated. Each object must consist of four pseudo atoms. The first atom defines the atoms, which represent the matter and defines the location of the object within space. The next three atoms represent the orientation and lengths of the axes of the object. If these are not Cartesian, the object will not be the ideal object reflected by its name but a distorted object. Each of the axes is the 'half' axis of the object, i.e. half the edge length of a cube, radius and half the length of a cylinder, and the radius of a sphere.

Each axis is represented by the difference vector between the position of the corresponding pseudo atom ((Numbers two to four) and the first atom. This allows you to apply all DISCUS commands to the individual pseudo atoms, thereby changing the size of the object.

The axes refer to the base system you used in the ==> 'cell' keyword of the unit cell file, or set using the ==> 'free' command at the ==> 'read' menu prompt. If you want to create a sphere of 10 Angstrom radius, centered at 0.5,0.5,0.5 in a base system of orthogonal 1 Angstrom long base vectors, the entry should be:

cell 1.00, 1.00, 1.00, 90.00, 90.00, 90.00 molecule character,sphere molecule density, 5.0 c 0.500, 0.500, 0.500, 1.45 xaxi 10.500, 0.500, 0.500, 0.0 yaxi 0.500, 10.500, 0.500, 0.0 zaxi 0.500, 0.500, 10.500, 0.0 molecule end

Useful commands to shift objects are the ==> 'symm' menu, to deform an object, use the ==>'shear' menu. New objects can be added to the structure with the ==> 'insert' menu.

content

```
molecule content, <type> [,<number>]
```

This keyword signals the beginning of a list of content for a molecule. The parameter <type> gives the type number of the molecule. All molecules of identical type can be referred to by a number of DISCUS commands. The optional <number> gives the number of the molecule in the list of molecules. This number is written by the ==> 'save' command. On input the number is ignored! All molecules read via 'molecule content' commands are appended to the list of existing molecule.

The molecule content must be followed by ==> 'molecule atom' commands and terminated by a 'molecule end' command.

biso

```
molecule biso, <value>
```

This keyword sets an isotropic displacement parameter for the molecule. All intra-molecular distances in a PDF and powder pattern only use the atomic displacement parameters, while intermolecular distances use this molecular isotropic displacement factor as well.

density

```
molecule density, <value>
```

This keyword helps you to create objects of identical volume yet different scattering power. The density is expressed in number of atoms per unit volume within the object.

---

atoms

```
molecule atoms [,<i> ...]
```

The atom number <i> is part of the molecule. The number <i> refers to the number of the atom in the current crystal. Up to 20 atoms may be specified on the 'molecule atom' command line.
The molecule content must be terminated by a 'molecule end' command.

old_format

```
Format of unit cell data file
```

The old unit cell data file contains 3 initial lines and one line per atom in the unit cell. All lines are in free format, beginning in the left most column.

```
Line 1:    Title up to 80 characters
line 2:    Space group symbol [,origin choice number]
line 3:    Lattice constants a,b,c,alpha,beta,gamma
```

and for each atom a line with:

```
       Name(4 characters) x  y  z  isotropic B
```

Empty lines be in the cell data file will be ignored.
Example

```
Single atom structure of NA
Pm-3m
  5.0 5.0 5.0 90. 90. 90.
NA1+  0.0 0.0 0.0 0.1
```

A.6   append

```
appe <name>, <x>,<y>,<z> ,<t>, <na>,<ne>, <delx> [,<dely> [,<delz>]]
appe <name>, <x>,<y>,<z> ,<t>, <na>,<ne>, -<bondlength>
```

Conditionally inserts an atom of type <name> at the position <x,y,z> in crystal space. The temperature coefficient must be given. 'appe' checks the position <x>,<y>,<z> with respect to all atoms numbered <na> to <ne>. The new atom is NOT inserted, if any of these atoms is within the block <x> +- <delx>; <y> +- <dely>; <z> +- <delz>. If <dely> and/or <delz> are not given, then they default to the value of the last <del.> given on this 'appe' command. Note, that the <del.> are in fractional coordinates, NOT in Angstrom.
In the alternative form, the eight's parameter is interpreted as a bond distance. To distinguish the two forms, the <bondlength> must be less than zero. The new atom is inserted, if no atom is less than <bondlength> Angstrom from the position <x>,<y>,<z>.
If the crystal is empty, i.e. n[1]=0, the 'append' command ignores the parameters <na>,<ne>, <delx> [,<dely> [,<delz>]]. Since there is no atom within the crystal, there is no need for a comparison. Effectively, the 'append' command acts just as an equivalent 'insert' command.

The parameters <na> and <ne> must be in the range 1 to current number of atoms in the crystal i.e. the value of variable "n[1]", and <na> must be less or equal to <ne>. The one exception is <na> = n[1]+1 and <ne> = n[1]. This allows you to start building e.g. a new layer within which you would like to optionally insert atoms without affecting the previous layer. Example i[0] = n[1]+1 do i[1]=1,10 insert al,ran(0),ran(0),0.5, 1.0, i[0],n[1], -2.4 enddo This loop inserts up to 10 Al atoms in the x,y range 0 to 1 at z=0.5 with a minimum distance of 2.4 Angstrom. Since i[0], the value for <na> is initially at n[1]+1, all previously inserted atoms are not affected. Since i[0] remains fixed within the loop, each new Al atom affects all Al atoms previously inserted within this loop.

A.7 anis

```
anis type:<number>, values:[<U11>,<U22>,<U33>,<U32>,<U31>,<U21>]
anis type:<number>, values:[<Uiso]
anis type:<number>, prin:[<u>, <v>, <w>, <value>]
anis wyckoff:[<x>, <y>, <z>]
```

This command set anisotropic displacement parameters of the atom type defined by the number on the "type:" parameter. The command takes either all six U_ij or a single isotropic displacement parameter.
The "wyckoff:" version checks the site symmetry for x,y,z and prints possible U_ij values and the constraints that apply to this Wyckoff site. Result values 1 to 6 are possible U_ij, result values 7 to 16 encode the 10 possible constraints that apply: Rule number Constraint Values 1 U11 = U22 1; -1 Equal; opposite sign 2 U11 = U33 1; -1 Equal; opposite sign 3 U22 = U33 1; -1 Equal; opposite sign 4 U23 0 Must be zero 5 U13 0 Must be zero 6 U12 0 Must be zero 7 U23 = U13 1; -1 Equal; opposite sign 8 U23 = U12 1; -1 Equal; opposite sign 9 U13 = U12 1; -1 Equal; opposite sign 10 U11 = 2*U12 2 U11 is twice U12 If a rule does not apply, the result value for this rule is set to 10.

A.8 asym

```
asym
```

Shows the content of the asymmetric unit. The names of those atoms, a number that is used as index for its scattering type, their position and temperature coefficient are listed. The number that is listed, is the number that refers to the scattering curve of that atom. It is contained in the variable m[<index>]. If a cell was read, all atoms are considered to be different, even if they are chemically identical and have the same temperature coefficient. If a whole structure was read, all atoms that are in the unit cell 0 <= xyz < 1, are chemically unique and have a different temperature coefficient are included in the asymmetric unit.

A.9 bragg

```
bragg  dval:<value_in_A>, theta:<theta_in_degrees]
bragg  lambda:<value_in_A>, theta:<theta_in_degrees]
bragg  lambda:<value_in_A>, dval:<value_in_A>
```

Optional parameters are :

```
radiation:xray        ! These choices affect the energy only
radiation:neutron
radiation:electron
output:none           !
output:screen         ! Short summary on screen, default
```

Performs calculations of Braggs law: lambda = 2 d sin(theta) Two of the three parameters, "lambda, dvalue, theta" must be given, the third is evaluated.

The wave length can be specified with either: lambda:<numerical_value> lambda:0.70926 ! Angstrom lambda:<Element_name> lambda:MoA1 or energy:<value> energy:17.48078 ! in [keV] for xray ! in [keV] for electrons ! in [meV] for neutrons The d-spacing can be specified with either: dval:<numerical_value> dval:3.800 ! Angstrom dstar:<numerical_value> dstar:0.2631 ! Angstrom == 1/d q:<numerical_value> q:1.6535 ! Angstrom == 2pi/d

The angle can be specified with either: theta:<numerical_value> theta:10.000 ! degrees twotheta:<numerical_va twotheta:20.000 ! degrees == 2*Theta

The results are stored into the res[] variable res[1] : lambda res[2] : d-value res[3] : Theta res[4] : Energy res[5] : d*star value = 1/d-value res[6] : Q = 2PI / d-valuea res[7] : 2*Theta

A.10   boundary

The boundary command at the main discus level is depreciated. Please use the ==> 'boundary' command at the ==>'surface' menu, further options beyond a simple removal of outside atoms are available through the 'surface' menu.

A.11   branch

```
branch kuplot [, "-macro" <macro_name> [ <par1> [ , <par2> ...]]]
branch diffev [, "-macro" <macro_name> [ <par1> [ , <par2> ...]]]
```

Active within the discus suite only!

Branches to the "kuplot" or "diffev" section.

Within this section any standard KUPLOT command can be given. The behavior of "kuplot" is essentially the same as in the stand alone version. Likewise for DIFFEV.

The main use will branch to KUPLOT while the discus section is run via run_mpi from a DIFFEV slave.

Optionally the "-macro" qualifier instructs the suite to run the macro <macro_name> (with its optional parameters) before the interactive session is started.

A.12   change

```
change "object","character",<number>,{"atom"|"cube"|"cylinder"|"sphere"}
```

The 'change' command allows you to change settings and characteristics. So far, it has only been applied to objects.

```
change "object","character",<number>,{"atom"|"cube"|"cylinder"|"sphere"}
```

This command allows you to change the character of an extended object. The character of the object Number <number> will be changed to one of {"atom"|"cube"|"cylinder"|"sphere"}.

A.13   chemistry

```
chemistry
```

Enters the 'chem' sub level. Here all kind of statistics about the model crystal can be obtained like relative amount of elements, distribution of bond-lengths, ...
NOTE: The MMC and CHEM sub level share some variables which define neighbors etc. and settings made in this sub level might be altered when using the other sub level. It is always save to repeat settings when entering this sub level if both levels are used.
Further information is available for commands

commands

Valid commands at this sub level are:

```
@        ! Executes a macro (see main help level)
=        ! Algebra (see main help level)
angl     ! Calculate average bond angles for given neighbors
apply_symmetry ! Use the space group symmetry to generate vectors
aver     ! Calculates average structure and occupancies
bang     ! Calculates distribution of bond angles
blen     ! Calculates distribution of bond lengths
bval     ! Calculates the bond valence sum for given atom
continue! Continue a stopped macro (see main help level)
corr     ! Calculates neighbor frequencies/correlations
disp     ! Calculated average distances for given neighbors
echo     ! Echos a string (see main help level)
elem     ! Show relative abundances of elements in crystal
env      ! Find neighbors around atom/site or position
eval     ! Evaluate an expression (see main help level)
exit     ! Terminates the chem sub level.
field    ! Calculates a correlation field
help     ! Gives on-line help for 'chem' (see main help level)
homo     ! Calculate concentration/correlation distributions
mode     ! Toggles between atoms and molecules mode
neig     ! Show neighbors for given neighbor definition
set      ! Change settings of 'chem' parameters
show     ! Show current parameters
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
trans    ! Transforms between atom-index and unit-cell/site
wait     ! Waits for user input (see main help level)
```

angle

```
angle {<name>|<number>|"all"},{<name>|<number>|"all" },
      {<name>|<number>|"all" } [,<file>]
```

This command calculates the average bond angles and sigmas for the specified triplets of atom types for the given neighbors. The neighbor determination parameters are set via the commands -> 'set neig' and -> 'set ang'. The first atom on the 'angl' command forms the center

of the bond angle. If an additional parameter <file> is given, all position parameters of the central atom (in lattice units) are written to the corresponding file for possible further analysis or plotting.

apply_symmetry

```
apply_symmetry [<output_file>]
```

This command applies the space group symmetry to generate the symmetrically equivalent vectors. You must have defined one or several vectors with the ==> 'set vec' command and grouped these into neighborhoods via ==> 'set neigh'.

The general concept assumes that you have defined the vector correlations for the position x,y,z within the asymmetric unit and would like DISCUS to generate the corresponding vector correlations for the other atom positions as well.

Example if an atom is on a general position you could specify: set vec, 1, 1, 2, 0,0,0 set neig, vec, 1 apply_symmetry This would generate the corresponding vector correlations for the other atoms in the unit cell that are symmetrically equivalent to xyz.

If an atom is on a special position, like a mirror plane, the apply_symmetry command will generate the mirror image of the vector.

If the optional <output_file> is given, DISCUS will write the original "set vec, ..." and the generated ones into this file.

aver

```
aver [ "one" | "ind"]
```

This command allows to calculate the average structure of the current model crystal. The average position, the standard deviation and the occupancy of every site within the unit cell is given.

CHANGE as of Version 5.6.5 !!! Non-backwards compatible !!! The site number, the atom type that occupies this site, the average positions and sigmas and the occupancies are stored in the variable array 'res'. Per site and multiple atom type on a site nine numbers are written to the result variable.

Without parameters or with parameter "one", all atoms on a site are treated as one type and a common average position is calculated. With parameter "ind", the average position is calculated individually for different atom types that may be present on a given site.

```
Warning: This command works only correct, if the structure is given
         in the order created by the 'read cell' command. Appended
         atoms are simply ignored ! However, using the PURGE command
         is not recommended when using this commands !!!!
```

bang

```
bang { "all" | <name> | <number> },
     { "all" | <name> | <number> },
     { "all" | <name> | <number> }, [<filename>]
```

This command calculates the bond angle distribution between the selected atoms (by <name>, <number> or "all"). The histogram is saved to the file <filename>. The number of points of the histogram is given by 'set bin,<ip>'. The calculation mode can be switched between "quick" and "exact". The mode "exact" calculates the bond angle for EVERY atom pair within the crystal whereas the mode "quick" only looks at neighboring unit cells. BUT in the "quick" mode errors can occur due to using the "purge" command or appending atoms to the structure. Allowed bond angle range, bond length range, calculation mode and sigmas for a neighbor matching criteria are given with the 'set' command.

If the first letters of the filename are "kuplot", the output is transferred directly to the KUPLOT section. This is valid within the suite only, a stand-alone DISCUS program will write a file that starts with the letters "kuplot".

blen

```
blen { "all" | <name> | <number> },
     { "all" | <name> | <number> }, [<filename>]
```

This command calculates the bond length distribution between the selected atoms (by <name>, <number> or "all"). The histogram is saved to the file <filename>. The number of points of the histogram is given by 'set bin,<ip>'. The calculation mode can be switched between "quick" and "exact". The mode "exact" calculates the bond length for EVERY atom pair within the crystal whereas the mode "quick" only looks at neighboring unit cells. BUT in the "quick" mode errors can occur due to using the "purge" command or appending atoms to the structure. Allowed bond length range, calculation mode and sigmas for a neighbor matching criteria are given with the 'set' command. The resulting distances, standard deviations and number of pairs of atoms are stored in the res[i] variables.

If the first letters of the filename are "kuplot", the output is transferred directly to the KUPLOT section. This is valid within the suite only, a stand-alone DISCUS program will write a file that starts with the letters "kuplot".

bval

```
bval "atom",<ia>,<radius>
bval "site",<nx>,<ny>,<nz>,<site>,<radius>
```

This command calculates the bond valence sum for a given atom. The parameters are identical to those of the command -> env. The atom can either be specified by its index ("atom") or unit cell and site number ("site"). The radius specifies the maximum distance for the function to search for neighbors. The radius is given in Angstroms. Note that bond valence parameters are only available for certain pairs of atoms in a specific oxidation state. It is also required to use the corresponding atom names, e.g. Zr4+ rather than Zr in the structure file. The bond valence parameters are taken from a list compiled by I.D. Brown, McMaster University, Canada. The resulting bond valence sum is stored in variable res[1].

corr

```
corr "occ", { <name>|<number>}, {<name>|<number>}
corr "disp",{ <name>|<number>|"all"},{<name>|<number>|"all" }
```

This command starts the calculation of occupational ("occ") or displacement ("dis") correlations within the crystal. The occ. correlations require the name or number of two DIFFERENT atom types to be given. The displacement correlation calculations requires atom names or "all" to use all atom types. The neighbor determination parameters are set via the commands -> 'set neig' and -> 'set vec'. The displacement directions used to calculate disp. correlations are set using the -> 'set dir' command. Correlation values can be :

```
correlation > 0 : preferred same neighbors / displacement
correlation = 0 : random distribution
correlation < 0 : preferred different neighbors / displacements
```

The correlations are calculated according to the following equations:

```
          occupational                        displacement
-----------------------------------------------------------------
        P(i,j) - theta**2                    <x(i)x(j)>
 c(i,j) = ----------------     c(i,j) = -----------------------
        theta * (1-theta)                sqrt(<x(i)**2><x(j)**2>)

 P(i,j): pair prob. sites i,j   x(i): displacement from average pos.
 theta : concentration          <>  : average
```

The resulting correlations are stored in the variables res[<i>] with <i> being the number of the corresponding definition for neighboring distances or vectors. The variable res[0] contains as usual the number of variables returned by the command.

disp

```
disp { <name>|<number>|"all"},{<name>|<number>|"all" } [,<file>]
```

This command calculates the average distances and sigmas for the specified pairs of atom types for the given neighbors. The neighbor determination parameters are set via the commands -> 'set neig' and -> 'set vec'. If an additional parameter <file> is given, all difference vectors (in lattice units) are written to the corresponding file for possible further analysis or plotting.

element

```
element [ { "on" | "off" } ]
```

Displays information about elements in the model crystal like relative abundances of the elements. 'void' stands for a vacancy. The results (range 0..1 rather than %) are stored in the variable res[<i>+1], with <i> the number of the atom type. Entry res[0] give the number of atom types. Entry res[1] gives the relative number of VOIDS. The optional parameter "on" or "off" controls if screen output is produced.

env

```
env "atom",<ia>,<radius>
env "site",<nx>,<ny>,<nz>,<site>,<radius>
env "pos",  <x>,<y>,<z>,<radius>
```

The command 'env' allows the user to print out all atoms within a distance of <radius> A around the given positions. This position can be given by an atom index <ia> ("atom"), a unit cell <nx>,<ny>, <nz> and site <site> ("site") or as absolute position <x>,<y>,<z> in lattice units ("pos"). The neighbor determination mode and possible periodic boundaries can be defined using the command -> 'set mode'. The atom indices of the neighboring atoms are stored in the variable res[i]. The variable res[0] contains the number of neighbors that were found.

field

```
field {"occ" | "disp"},<a1>,<a2>,<file>,<xmin>,<xmax> [,<ymin>,<ymax>]
```

This command allows the calculation of a correlation field. The first parameter determines the use of 'occupational' or 'displacement' correlations. The next two parameters specify the atoms to be used. They can be specified as number or name (-> corr). The next parameter is the name <file> for the output graphics file that can be read by the plot program KUPLOT. The last parameters define the range in x- and y-direction. If no y-values are given, only a 1-dimensional field is calculated. The values of <xmin>, <xmax>, <ymin> and <ymax> are multiples of the neighbor vectors defined.
The following example will compute the correlation field along <100> up to the 20th neighbor:

```
discus/chem> set vec,1,1, 1,0,0
discus/chem> set neig,vec,1
discus/chem> field zr,void,corr.xy,0,20
```

homo

```
homo "occ",<atom>,<file> [, site:site_selection]
homo "cor",{"occ" | "dis"},<at1>,<at2>,<file>
```

This command computes the concentration or correlation distribution within the crystal. The sampling volumes are set using the -> 'set lots' command. The size of the resulting histogram can be altered using the command -> 'set bin'. The following two options are available:
"occ": Here the concentration distribution of the atom <atom> is computed. The resulting histogram is written to the file named <file>. The optional parameter site:selection can take on the forms: site:all ! all sites will be searched, Default site:<number> ! Only atoms on the site <number> are counted site:[<number>] ! Only atoms on the site <number> are counted ! More than one site may be specified as in ! site:[1, 3, 4]
"cor": Here the correlation distribution for all defined neighbors is computed. The third parameter determines whether occupational or displacement correlations are computed. The next two parameters specify the atom types and the last parameter the filename for the histogram output. Note that *all* defined neighbors will be written to the file. To get a specific correlation distribution use a single neighbor definition.

mode

```
mode [ { "atom" | "mole" } ]
```

This command determines whether CHEM commands operate on atoms or on molecules. Note that when working with molecules the atom names or numbers in the command descriptions have to be replaced with molecule types. Note that not all commands are available in molecule mode (e.g. 'aver', 'blen').

neig

```
neig <index>,<ic>
```

This command determines and displays the neighboring atoms or molecules around a given atom or molecule given by <index> for the neighbor definition <ic> (-> 'set neig').
The resulting atom or molecule indices of the determined neighbors are stored in the variable res[i] where i is the number of the neighbor. The variable res[0] contains the number of found neighbors.

set

```
set <subcommand>,...
```

This command sets various parameters within the 'chem' section of the DISCUS program. Valid <subcommands> are:

commands

```
"ang"     : defines the neighbors for bond angle calculations
"bang"    : sets range for bond angle calculations
"blen"    : sets range for bond length calculations
"bin"     : sets number of points for bond length histogram
"cryst"   : sets crystal dimensions (if determined wrong by read stru)
"lots"    : sets crystal sampling volumes for 'homo'
"mode"    : sets calculation mode ("quick"/"exact")
"neig"    : defines how correlations are calculated
"vec"     : defines correlation vectors
```

ang

```
set "ang",<iv>,<is1>,<is2>,<dx2>,<dy2>,<dz2>,<is3>,<dx3>,<dy3>,<dz3>
```

This command is used to define the neighbors for bond angle calculations. To group these neighbor definitions use the ==> 'set neig' command. The first parameter <iv> is the number of the angle to be defined. The variable <is1> gives the number of the crystal site at the center of the angle. Variables <is2> and <is3> give the number of the crystal site at each end of the angle. The values of <dx*>,<dy*> and <dz*> define relative position of the neighbors in the same or adjacent unit cells. Neighbors within one unit cell have <dx*>,<dy*> and <dz*> set to zero. If you want to define angles crossing the unit cell boundaries set <dx*>,<dy*> and <dy*> accordingly.

bang

`set "bang",<min>,<max>`

This command sets the range <min> to <max> of the bond-angle to be calculated and binned to the histogram.

blen

`set "blen",<min>,<max>`

This command sets the range <min> to <max> of the bond-length to be calculated and binned to the histogram.

bin

`set "bin",<number>`

This command sets the number of points for the histograms created by 'chem'.

cryst

`set "cryst",<nx>,<ny>,<nz>,<natoms>`

A bug in the current version of DISCUS sometimes causes wrong crystal- dimensions. This command allows to rewrite the values for the crystal- dimension <nx>,<ny>,<nz> and number of atoms in a unit cell <natoms>.

lots

`lots "off"`
`lots {"box" | "ellipsoid"},<lx>,<ly>,<lz>,<n>,{"yes" | "no"},[<fname>]`

This command allows the definition of 'small crystal volumes' to be used to check the homogeneity of the crystal (-> 'homo'). For details about the command and its parameters see the help file entry for -> 'four lots'.

mode

`set "mode",{"quick"|"exact"} [, {"period" | "normal"}] [,<axes>]`

This command switches the calculation mode between "quick" and "exact". In the "quick" mode only the neighboring unit cells around a certain atom are taken into account. If the command 'purge' is performed or a structure is read where the atoms are not listed in the DISCUS order this mode might fail, because the calculation of the index of an atom in a neighboring unit cell can be wrong. In this case the "exact" mode should be used which taken ALL other atoms into account.
The second optional parameter determines if periodic boundaries are used. Note that this only works if the "quick" mode is selected and NO extra atoms were added to the crystal.
Note that this setting does not affect the calculation of correlations defined by vectors using the 'set vec' command.

neig

```
set "neig",<subcommand>
```

This command determines how DISCUS calculates the correlation value using the command 'corr'. These settings might be overwritten when using the MMC sub level. The following subcommands are valid:

commands

```
"ang"  : Defines neighbors via given angles
"add"  : Adds current definition to list of correlations
"dis"  : Defines neighbors via distance criteria
"dir"  : Defines directions for displacement correlation calculations
"rese" : Resets list of defined neighbors
"vec"  : Defines neighbors via given vectors
    [,"number:"<value>]
    [,"number:next"]
```

The optional parameter "number:" replaces the "set neig, add" command that was required in between two "set neig,..." commands. You can either specify a specific numerical value: set neig, number:1, vec, 1, 2 or increment the neighborhood number with set neig, number:next, vec, 1, 2

ang

```
set "neig","ang",<iv1> [,<iv2>,..]
set "neig","ang",<iv1> ,<iv2>, "range"
    [,"number:"<value>]
    [,"number:next"]
```

"Ang" results in DISCUS using correlation angles $<iv>$ to be used for the calculation. The parameters are the numbers $<iv>$ of the correlation angles defined with 'set ang'. If the second command form is given, the correlation angles in the range $<iv1>$ to $<iv2>$ will be used for the calculation.

The optional parameter "number:" replaces the "set neig, add" command that was required in between two "set neig,..." commands. You can either specify a specific numerical value: set neig, number:1, vec, 1, 2 or increment the neighborhood number with set neig, number:next, vec, 1, 2

add

```
set "neig","add"
```

This command is depreciated. Best use the optional parameter number:value on the regular "set neig, ..." commands instead.

"Add" adds the current neighbor definition to the list of definitions executed later by the -> 'corr' command. Use 'show' to get a list of defined correlation vectors/distances. DISCUS saves actually a copy so that no "add" is needed after the last definition is entered.

dis

```
set "neig","dis",<u>,<v>,<w>,<fsig> [, <wsig>] [, {"sym" | "nosym"}]
    [,"number:"<value>]
    [,"number:next"]
```

"Dis" causes DISCUS to determine the neighbors used for the calculation of the correlations using the given distance and direction (optional). Those neighbors having a distance corresponding to the vector <u>,<v>,<w> given in unit cell units are taken into account for the calculation. The parameter <fsig> sets the allowed difference of the observed distance to the theoretical value (+- <fsig>/2). The next optional parameter gives the allowed deviation from the neighboring direction given by <u>,<v>,<w>. The last parameter determines if symmetrically equivalent directions to <u>,<v>,<w> should be used as well. If this parameter is omitted ALL neighbors with the correct distance are valid.
The optional parameter "number:" replaces the "set neig, add" command that was required in between two "set neig,..." commands. You can either specify a specific numerical value: set neig, number:1, vec, 1, 2 or increment the neighborhood number with set neig, number:next, vec, 1, 2
The optional parameter "number:" replaces the "set neig, add" command that was required in between two "set neig,..." commands. You can either specify a specific numerical value: set neig, number:1, vec, 1, 2 or increment the neighborhood number with set neig, number:next, vec, 1, 2

dir

```
set "neig","dir",<x1>,<y1>,<z1> [,<x2>,<y2>,<z2>]
set "neig","dir","all"
    [,"number:"<value>]
    [,"number:next"]
```

This command sets the displacement directions to be used for the calculation of displacement correlations. The parameter "all" will calculate the displacement correlations for displacements in all directions, i.e. the resulting correlation gives the correlation of displacements of two neighboring atoms regardless of the direction of the displacement itself. One can also give specific directions, e.g. if one wants to determine the correlations of displacements in z-direction of site i with the ones in x direction of site j one would enter the command 'set dir,0,0,1,1,0,0'. Note that the neighboring vector between sites i and j is set by the command 'set neig'. If <x2>,<y2>,<z2> are not given their values are set to those of <x1>,<y1>,<z1>.
The optional parameter "number:" replaces the "set neig, add" command that was required in between two "set neig,..." commands. You can either specify a specific numerical value: set neig, number:1, vec, 1, 2 or increment the neighborhood number with set neig, number:next, vec, 1, 2

reset

```
set "neig","reset"
```

"reset" resets the list of correlation vector/distances definitions.

---

vec

```
set "neig","vec",<iv1> [,<iv2>,..]
set "neig","ang",<iv1> ,<iv2>, "range"
    [,"number:"<value>]
    [,"number:next"]
```

"Vec" results in DISCUS using correlation vectors <iv> to be used for the calculation. The parameters are the numbers <iv> of the correlation vectors defined with 'set vec'.
If the second command form is given, the correlation vectors in the range <iv1> to <iv2> will be used for the calculation.
The optional parameter "number:" replaces the "set neig, add" command that was required in between two "set neig,..." commands. You can either specify a specific numerical value: set neig, number:1, vec, 1, 2 or increment the neighborhood number with set neig, number:next, vec, 1, 2

vec

```
set "vec",<iv>,<is1>,<is2>,<dx>,<dy>,<dz>
```

This command is used to define correlation vectors that might be used to calculate correlations within the crystal (-> 'set neig'). The first parameter <iv> is the number of the vector to be defined. The variables <is1> and <is2> give the number of the crystal site at each end of the vector. The values of <dx>,<dy> and <dz> define the vector in unit cells. Correlation vectors within one unit cell have <dx>,<dy> and <dz> set to zero. If you want to define correlation vectors crossing the unit cell boundaries set <dx>,<dy> and <dy> accordingly.

show

```
show
```

This command lists the current settings and correlation vector definitions on the screen.

trans

```
trans <atomindex> or
trans <x>,<y>,<z>,<site>
        ",output:screen" | ",output:none"
        ",boundary:periodic" | ",boundary:crystal"
```

This command allows to transform between a given index of the storage array <atomindex> and the corresponding unit cell <x>,<y>,<z> and site <site>. The parameter not given is calculated ...
The optional parameter "boundary:periodic" will apply periodic boundary conditions to the unit cell indices <x>, <y>, <z>
The optional parameter "output:none" will suppress the screen output.
Trans stores the results in the "res[]" variable: res[1] : <atomindex> res[2] : <x> res[3] : <y> res[4] : <z> res[5] : <site>
The command is identical to the ==> 'index2cell' command in the main menu.

A.14   connectivity

Switches to the connectivity menu. At this menu all parameters related to interatomic distances and the connectivity list are defined.

Modification Note: As of version 5.24 DISCUS will perform the requested action (add, set, remove) for all central atom types with this name if the central atom is specified by name instead of type number. add Si, O, 1.2, 2.0, silicon Creates a connectivity for all Silicon atom types. add 3, O, 1.2, 2.0, silicon Creates a connectivity for atom type 3 only.

Commands

```
add      ! Adds a new definition
create   ! Creates the actual connectivity list for each atom
delete   ! Deletes the connectivity list around atoms
recreate ! Recreates the actual connectivity list for each atom
remove   ! Removes an individual connectivity definition
reset    ! Resets to program start, removes all def.s and the
         !        connectivity list
set      ! Sets an individual definition
show     ! Shows the current definitions
```

add

```
add <central_atom_type>, <neigh types>, <rmin>, <rmax>
```

[,<max_neig], <def_name>, ["molescope:"{"ignore"|"within"|"outside"}]

Add a new connectivity definition. For the atom type <central_atom_type> a new definition is added. Atoms of types <neigh types> that are in a distance shell from <rmin> to <rmax> are entered into this definition.

If <rmin> is set to zero, all neighbors, even at infinitely small distances are included in the connectivity. The central atom itself is excluded from the list. Keep in mind that the central atom number may appear in the connectivity list, if periodic boundary conditions are included. This will be the case if <rmax> is large enough to wrap around the complete crystal.

The last but one parameter is optional and can be used to limit the number of neighbors in the connectivity. If present, it takes the form: "first:1" or "first:2" etc. The string "first:" is mandatory and is followed by the actual number of neighbors. If DISCUS finds less than these neighbors, the connectivity is of course less than the intended number of neighbors.

The last parameter is a name for this definition.

As of version 5.14 a new optional parameter allows you to restrict the scope of the connectivity to atoms that are within the same molecule as the central atom. This parameter takes the form: "molescope:"{"ignore"|"within"|"outside"} Choice "ignore" signals that connectivity will ignore if the central atoms and any of its neighbors are within a molecule or not. Choice "within" creates a connectivity between a central atom that must be within a molecule and other atoms within the very same molecule. Choice "outside" creates a connectivity for a central atom that must be within a molecule and other atoms that are not part of this molecule.

This command just defines the settings of this connectivity. The actual list is created via ==> 'create'.

Keep in mind that many different atom types may exist, even if the names and ADP of two or more atoms in the cell file were identical. If you read the structure with the "cell" command all atoms within this asymmetric unit are treated as individual atom types! Use the atom type number instead of the atom name for a clear distinction. Alternatively read the structure with the "lcell" command

create

```
create {<central_atom_type>, {<def.no.>| <def_name>}}
```

Creates the actual connectivity list. The program performs a loop over all atoms and searches for neighbors that are valid under any of the connectivity definitions. The valid neighbors are recorded into the connectivity list
Warning, keep in mind that ==> 'property' settings will affect the search for neighbors as well. Any old connectivity list is replaced by a new 'create' command.
If the second command form with parameters is chosen, only the connectivity for the specified atom will be created. This is helpful if: you create (several) connectivities first, then perform a modification of the crystal that may affect the original connectivities Finally need to create a new connectivity but do not want to change the original connectivities. This command for is identical to a recreate <central_atom_type>, {<def.no.>| <def_name>}

delete

```
delete
delete <central_atom_type>, <def_name>
```

Deletes the current connectivity list for all atom types.
The second form deletes the named connectivity list for the specified atom types. The remaining connectivities persist for the atom type. All connectivities for all other atom types remain as well.

recreate

```
recreate <central_atom_type>, {<def.no.>| <def_name>}
```

Recreates the connectivity list for the specified atom type and connectivity definition. All neighbors to a central atom of this type are redetermined. The previous entry in the connectivity list is completely lost.
If the connectivity does not create, it is created without any error message.
All other connectivities are not affected by this command.

remove

```
remove <central_atom_type>, {<def.no.>| <def_name>}
```

Removes the definition number <def.no.> or name <def_name> for central atom type <central_atom_type>. If further definitions exist for this atom type, they are shifted forward, and their definition no. is decremented.

---

reset

Resets the connectivity to program start conditions. All definitions are removed and the connectivity list is deleted.

set

`set <central_atom_type>, {<def.no.>| <def_name>}, <neigh types>, <rmin>,`

<rmax>
This command allows you to replace the settings for the definition number <def.no.> or name <def_name> for the central atom type <central_atom_type>. The definition must have been created by a previous 'add' command.

show

`show`
`show "connectivity", <atom_no.>, <def.no.>`

Without arguments, the 'show' command shows a list of all connectivity definitions that have been added.
With the fixed first parameter "connectivity", the show command will show the list of neighbors around atom no. <atom_no.>, which have been found using definition number <def.no.> for the scattering type of atom no. <atom_no.> The list of atom numbers is copied into the result variable "res".

switch

`switch`
`switch <is>, {<conn_number>|<conn_name>}`
`switch <iatom>,<is>, {<conn_number>|<conn_name>}`

Switch the connectivity between a pair of atoms. DISCUS will find a random atom A and randomly choose a neighbor atom B among the selected connectivity. For atom A a further neighbor C and for atom B a further neighbor D is chosen among the respective connectivities. Atom C is then connected to atom B and atom D to atom A:
prior afterwards A ==> B A ==> B A ==> C A ==> D B ==> A B ==> A B ==> D B ==> C
With no parameters any atom with any atom type is selected and the connectivity is the connectivity number one.
With two parameters, an atom of type <is> is chosen and the connectivity must be specified.
With three parameters, atom number <iatom> is chosen and the scattering type <is> must match this atom, and the connectivity must be specified.

A.15   copy

`copy <s> , < n> , <x> , <y> , <z>`

Copies an atom. If <s> is "a" the atom number <n> is copied to absolute positions <x>,<y>,<z> in crystal space, if <s> is "r", a relative motion copies the atom number <n> to the sum of present position and vector <x>,<y>,<z>.

A.16   exp2pdf

Menu to transform an experimental powder pattern into a powder PDF
DISCUS performs an empirical transformation similar to Billinge & Farrow, J.Phys:Condens.
Matter (2013), 25, 454202.
The (optional, scaled) background data are subtracted from the intensity. $I\_pure(Q) = I(Q) -$ Back(Q) The remaining intensity is divided by the square of the average form factors $<f>**2$, and point wise multiplied by Q. $F\_init(Q) = I\_pure(Q) / <f>**2 * Q$ A polynomial is fitted through the resulting intensity, and this polynomial is subtracted from the resulting intensity. $F(Q) = F\_init(Q) - SUM\_1**N p\_i**N * Q$ Finally G(r) is calculated as sine Fourier transform of F(Q). $G(r) = Int F(q) sin(Qr) dQ$ Integration limits are zero to a user defined Qmax.
As this empirical algorithm lacks a proper scaling of S(Q) DISCUS performs an empirical scaling to get S(Q), F(Q) and G(r) onto an approximate absolute scale.

back

```
back  xy,  <background_file>  [, scale:<background_scale>]
back  csv, <background_file>  [, scale:<background_scale>]
back "kuplot", <data_set_number>
```

Load a file with the background to the ==> 'data' file. The file need not to be on the same Q-scale and step width as the 'data' file, yet its lower and upper limit must include the Q-range of the 'data' file.
The intensity on the data file will be multiplied by the optional 'scale' parameter. Defaults to 1.0

composition

```
composition composition:<string>
composition             <string>
```

Defines the samples composition. Specify the composition string as Atom names, followed by the relative abundance: examples: Pure gold : Au Ceria : CeO2 Ceriai : Ce O2.0 Nonstoiciometric : Fe1.0 O0.978
Blanks are not relevant. Atom names must have a first capital letter and a lower case second letter.

data

```
data  xy,  <data_file>
data  csv, <data_file>
data "kuplot", <data_set_number>
```

Load a file with the powder diffraction pattern. At the moment the file has to be on a Q-scale.
The Q-scale does not have to be equidistant.

exit

```
exit
```

Return to the main DISCUS menu

faver

```
faver  xy,  <form_factor_file>
faver  csv, <form_factor_file>
faver "kuplot", <data_set_number>
```

Load a file with the average atomic form factors that correspond to the ==> 'data' file. The file need not to be on the same Q-scale and step width as the 'data' file, yet its lower and upper limit must include the Q-range of the 'data' file.

limits

```
limits inst:[<qmin>,<qmax>], [fourier:<qmin>,<qmax>]
```

Defines optional limits in which to handle the observed intensity.
The intensity data might become very noisy at high Q, or might be subject to detector problems. Especially for electron diffraction data the low Q-section might be overshadowed by the primary beam or contain spurious signal. To avoid these sctions, you can limit the lower and upper Q-range that is used for the empitical polynomial fit with [inst:<qlow>, <qmax>] Defaults: <qlow> lower Q-value in the data <qmax> upper Q-value in the data
For similar reasons you might want to limit the lower/upper Q limit that is used in the sine-Fourier transformation of F(Q) into the PDF. At low Q DISCUS will take the user supplied value or if this is omitted the position where F(Q) is minimum. Ar the high Q limit, DISCUS will take the user supplied value and find a point nearby at which F(Q) is of value zero. This reduces Fourier ripples in G(r). fourier:[<qmin>, <qmax>] Defaults to the maximum instrumental Q-value of inst:<qmax>.
For each optional parameter you can omit either of the two values, illustrated for inst:

```
inst:[,<qmax>]    == inst:[0.0, <qmax>]
inst:[<qmax>]     == inst:[0.0, <qmax>]
inst:<qmax>       == inst:[0.0, <qmax>]
inst:[<qmin>,]    == inst:[<qmin>, <upper Q limit of experimental data >]
```

outfile

```
outf  [gr:<GR_obs_file>] [,rmin:<value>] [,rmax:<value>] [,rstep:<value>]
outputfile  [gr:<GR_obs_file>] [,rmin:<value>] [,rmax:<value>] [,rstep:<value>]
outputfile  [iq:<Intensity_file>]
outputfile  [sq:<Normalized_Intensity_file>]
outputfile  [fq:<Reduced_normalized_Intensity_file>]
```

Defines the output to be written. For all values, the commands 'outf' and 'outputfile' are equivalent.
gr:<GR_obs_file> Defines the observed PDF file. The file is Written as simple 2-column file: r G(r) Always written, defaults to "discus.grobs"

For the PDF, optionally limits can be specified with rmin:<value> Lowest R-value in [Angstroem] rmax:<value> Largest R-value in [Angstroem] rstep:<value> Step interval in [Angstroem] Default to 0.01, 100.0, 0.01 respectively

iq:<Intensity_file>] Defines the observed intensity file to be written. This is I - Back, on an equdistant Q-scale Written is present, defaults to "discus.iqobs" [sq:<Normalized_Intensity_file>] Defines the observed normalized intensity S(Q). Written is present, defaults to "discus.sqobs" [fq:<Reduced_normalized_Intensity_file>] Defines the reduced observed normalized intensity F(Q). F(Q) = Q*[S(Q) - 1]. Written is present, defaults to "discus.fqobs"

poly

```
poly order:<value>
```

Defines the order of the polynomial fitted through F(Q) to ensure that F(Q) oscillates around F(Q)= zero. Usually a vale around 7 to 9 will be fine. Defaults to 7.

radiation

```
radiation "xray"
radiation "neutron"
radiation "electron"
```

Defines the radiation that was used to measure the observed intensity.

run

```
run
run   [mode:silent]
run   [mode:inter]
```

Perform the actual calculation.
The optional parameter mode enables to run the calculation silently without intermediate screen output, or with graphical and screen output. Defaults to mode:silent.

qscale

```
qscale qcrystal:<value> [, qobs:<value>]
qscale dcrystal:<value> [, qobs:<value>]
```

For electron diffraction data you might need to correct the effective camera length. Provide either the expected Q-value "qcrystal" (in [A∗∗-1]) or the expected d-value "dcrystal" (in [A]). Optionally you can provide the (approximate) experimental Q value "qobs".
DISCUS will search for a maximum in F(Q) near "qcrystal" or "qobs" and scale the Q-axis by a factor qcrystal / qobs , where qobs will be the actual peak position found near qobs.

A.17 export

```
export "cif",<file{".cif"}> [, {["spcgr:original"] | ["spcgr:"<spacegroup>]}
export "shelx",<file{".ins"}> [, "cycle:"<number>] [, "force:"{"yes"|"no"}]
              [, "origin:"{"no"|"yes"|"shift"}] [, "lambda:"{<symbols>|<value>}]
export "rmcprofile",<file{".rmc6f"]>  ["version:"{"6"|"7"}]
export "vasp",<file]>
export "poscar",<file]>
export "scatty",<file]> [,site:{"average","indiv"}]
```

cif Exports the atoms of the current structure as a minimal CIF file. With the optional parameter "spcgr" set to original you can retain this original space group instead of the default space group "P1". You can even specify another space group instead of the string "original" if you want to change the space group. Be aware the 'export' does not check the compatibility of the space group and lattice parameters.

shelx Exports the atoms of the current structure as a SHELXL instruction file The optional parameter "cycle:" allows to set the number of least-squares cycles to be written as "L.S." instruction. If the structure contains molecules, the atom coordinates might extend over more than one unit cell. Usually this triggers an error message. To avoid the error message, use the optional parameter "force:yes" For centrosymmetric structures SHELX expects origin choice 2 with the center of symmetry at 0,0,0. If the current structure uses orign choice 1 the default is to triger an error message. With the optional parameter "origin:yes" or "origin:shift" DISCUS will shift the center of symmetry to 0,0,0. This will change the current structure and a warning is displayed. Optionally you can specify the wave length with either lambda:MOA1 Or any other valid symbol lambda:0.70926 Your numerical choice The wave length defaults to Molybdenum Kalpha

rmcprofile Exports the atoms of the current structure as a RMCprofile version 6 or 7 input file

vasp poscar Exports the atoms of the current structure as a VASP "POSCAR" file.

scatty [,site:{"average","indiv"}] Exports the atoms of the current structure as a SCATTY file Only the "*_atoms_01.txt" file is written. The user has to edit his "scatty_config.txt" file individually. The optional parameter lets you write a file, in which either: the "site" lines in the scatty file correspond to the positions in the average unit cell, the default, or the "site" lines correspond to each individual atom "site:indiv". In this version, the displacements on the atom lines are all set to zero.

A.18 d2r

```
d2r <u>,<v>,<w>
```

This command transforms direct space vector <uvw> into its corresponding reciprocal space coordinates. See "r2d" for the reverse transformation. The resulting vector is stored in "res[1]", "res[2]", "res[3]".
The parallel, equivalent vector, normalized to 1/(length of [u,v,w]) is stored in "res[4]", "res[5]", "res[6]"

A.19 decorate

The main menu for surface decoration. The commands allow you to define a variety of decoration modes for guest structures (molecules=ligands) to be placed on top of an external surface. The operation tries to do this quietly in the background but does have a few side effects. Once started the program will: Distribute the surface atoms (=anchors) that are bonded to the ligands as evenly as possible on the surface. To achieve this, it builds resets the ==> 'connectivity', builds its own and then uses ==> 'mmc' with a repulsive potential to distribute the anchors evenly and to avoid close anchor-anchor distances. Eventually the internally saved core is placed back into the surface shell via the 'domain' concept. These steps overwrite previous user settings within the connectivity, mmc, and domain menu's. The ligands are read from their structure files and automatically transformed to the metric of the host structure. There is no need to transform the ligand metric to that of the host structure prior to being used. As the ligand molecules need to be rotated anyway to fit the local surface, the orientation relationship between the ligand metric and the host metric is not relevant.

If deco successfully placed molecules onto the surface, the crystal is no longer periodic. Thus periodic boundary conditions and the "chem" quick mode are off. They can be turned back on after explicit 'set crystal' command in the chemistry menu.

Commands

```
add      ! Add a new definition
reset    ! Reset and clear all definitions
run      ! starts the calculation
set      ! Define parameters for a definition
show     ! Show all current definitions
```

add

```
add <name>, <type>
```

Adds a new definition, called <name> of type <type>. The definition is referenced by its name in the details on the 'set' command. Valid types are: "normal" The ligand molecule is place on top of a single surface atom. A single ligand atom is bonded to the surface atoms. The orientation of the molecule is defined by its axis which is placed normal to the local surface.
Exactly one 'set bond...' command is required. "chelate" The ligand molecule is placed on top of a single surface atom. A pair of ligand atoms is bonded to the identical surface atom. The orientation of the molecule is defined by its axis which is placed normal to the local surface.
Exactly two 'set bond...' commands are required. "bridge" The ligand molecule is placed bridging between two different surface atoms. A single ligand atom is bonded to two (neighboring) surface atoms. The orientation of the molecule is defined by its axis which is placed normal to the local surface.
Exactly two 'set bond...' commands are required. "double" The ligand molecule is bonded with individual bonds to two surface atoms. Each bond connects a single surface atom to a single ligand atom. The orientation of the molecule is defined by its axis which is placed normal to the local surface.

Exactly two 'set bond...' commands are required. "multi" The ligand molecule is bonded with several bonds to a group of surface atoms. A first bond connects (another) ligand atom to a (group of) surface atom(s). A second bond connects a ligand atom to a single surface atom. The orientation of the molecule is defined by its axis which is placed normal to the local surface. Exactly two 'set bond...' commands are required. "acceptor" Place a ligand molecule with a hydrogen bond type. The surface atom acts as acceptor atom. The surface atom should be a reasonable acceptor atom like O, N... and the ligand atom should be a Hydrogen. The ligand atom is placed at a user specified distance to the surface atom. This distance should be around 1.9 Angstrom. Within the H atom DISCUS will create a roughly 170 degree bond angle. The orientation of the molecule is defined by its axis which is placed normal to the local surface. Exactly one 'set bond...' command is required. "donor" Place a ligand molecule with a hydrogen bond type. The surface atom acts as donor atom. The surface atom should be a reasonable donor atom like H, and the ligand atom should be a reasonable acceptor atom like O, N, ... The ligand atom is placed at a user specified distance to the surface atom. This distance should be around 1.9 Angstrom. Within the H atom DISCUS will create a roughly 170 degree bond angle. The orientation of the molecule is defined by its axis which is placed normal to the local surface.
Exactly one 'set bond...' command is required.

reset

```
reset
```

Reset all decoration definitions back to the default values at system start.

run

```
run
```

Starts the decoration.

set

```
set
set <name>, "ligand", <file>, <density>
set <name>, "bond", <atom_type> [, <atom_type>,...], <ligand atom no>,
                                        <distance>
                       [, "angle:"<b_angle>]
                       [, "anchor:"<n_anchors>]
                       [,{"distri:even" ! "distri:fixed"} ]
set <name>, "axis", <ligand atom no 1 >, <ligand atom no 2>

set <name>, "tilt", "angle", <value>
set <name>, "tilt", "atoms", "auto"
set <name>, "tilt", "atoms", <n1>, <n2>, <n3>, <n4>

set <name>, {"hkl"| "form"}, {<h>, <k>, <l>| "none"}
set <name>, "surface", <ligand atom no> [, <ligand atom no>...]
```

This command allows to set the detailed definitions needed to decorate the surface.
set <name>, "ligand", <file>, <density> Defines the input file from which the ligand is read. The last parameter <density> defines the ligand density on the surface in units of [ligands per

A∗∗2]. DISCUS estimates the surface area by counting the number of atoms with the "external" property flag. An average area of 11.0 A∗∗2 is assigned to each surface atom.

set <name>, "bond", <atom_type> [, <atom_type>,...], <ligand atom no>, <distance>, [, "angle:"<b_angle>] [, "anchor:"<n_anchors>] Defines the bonds of length <distance> between surface atoms of type <atom_type> and atom number <ligand atom no> of the ligand.

The number of surface atom types can be flexible, the number of 'set bond ' commands depend on the decoration that is defined by the "add" command. For the "normal", "acceptor" and "donor" type exactly one 'set bond' command is required. For the "bridge", "chelate", "double", and "multi" type exactly two 'set bond' commands are required. For the "multi" type exactly three surface type parameters must be given. Two 'set bond' commands are required. The first 'set bond' command requires three surface types.

The surface types can be given as type number or as type name. If the type number is given, this atom type will act as anchor. If the type name is given, all atom types with this atom name will act as anchors.

The optional parameter "angle:" is relevant to "acceptor" and "donor" bond schemes only. For these schemes the angle is interpreted as the hydrogen bond angle in the Hydrogen atom and should be close to 170°.

The optional parameter "distri" with default value "even" allows you to influence the distribution of surface sites onto which the ligand molecules at bonded. In the default case of "even", the surface site will be -reasonably- evenly spread across the surface. With "fixed", the surface sites will be randomly chosen among the surface atoms defined by the <atom_type> parameter, but their locations will remain fixed. As optional parameters default if absent, make sure that you specify this optional parameter on the last 'set bond' command.

set <name>, "tilt", "angle", <value> set <name>, "tilt", "atoms", "auto" set <name>, "tilt", "atoms", <n1>, <n2>, <n3>, <n4> Defines a tilt of the molecules relative to the local surface normal. If the molecule is reasonably flat, one can expect that the molecules will be tilted around an axis that is parallel to both, the local surface and the molecule plane. With "tilt, atoms, auto" DISCUS will calculate a least squares plane through the ligand molecule and tilt the molecule around the axis described above. Alternatively, DISCUS can construct the normal to the molecule if you provide the four atom numbers <n1>, <n2>, <n3>, <n4>. The normal will be calculated via the vector product of the two difference vectors between the pairs: n2-n1 and n4-n3. Make sure that these two vectors are not parallel. One of the atoms may be present in both vectors. The value of <n1> could for example be identical to either <n3> or <n4>.

set <name>, "axis", "auto" set <name>, "axis", <ligand atom no 1 >, <ligand atom no 2> Defines the ligand molecule axis. This axis is placed normal to the local surface, in as much as the decoration type allows. The two atom numbers define the ligand axis.

For the default mode "auto", DISCUS searches for the furthest molecule atom that is not involved in a bond. The axis is then defined by the first bonding atom and this furthest atom. If a molecule consists of just one atom or just of the two atoms involved in the "double" or "multiple" bonds, the "auto" mode will flag an internal message not to rotate the molecule at all.

For the types "normal" and "bridge" the axis is placed parallel to the normal by an appropriate rotation of the ligand. For the types "double" and "multi" the orientation of the molecule is partially defined by the bonds. The remaining freedom is to rotate the molecule around the axis defined by the two atoms involved in the two bonds. The molecule is rotated to place the

axis into a plane that is parallel to the local surface normal.

set <name>, {"hkl"|"form"}, {<h>, <k>, <l>| "none"} Allows to define a restricted placement of the ligands onto one or several surface faces or complete forms. With "hkl" the restriction applies to a single face <h>, <k>, <l>, with "form" the Miller indices <h>, <k>, <l> are expanded to the complete form for the current crystal system. You can use several commands to combine more than one restriction. DISCUS calculates the local surface normal and applies a strict adherence to the specified Miller indices. If the last parameter is "none", all restrictions are removed.

set <name>, "surface", <ligand atom no> [, <ligand atom no>...] Allows to define, which ligand atom numbers will be flagged as new surface atom types. All these molecule atoms will inherit the surface properties from the site to which this molecule as bonded to.

show

```
show
```

Show all current definitions

A.20   define

```
define generator, {"center" | "symmetry"}
```

The define commands sets a variety of parameters that are specific to the special program, here DISCUS. See the ==> "set" command at the "Command Language" help entry for information of general settings common to all programs within the DIFFUSE suite.

```
define generator, {"center" | "symmetry"}
```

In the original program development, all generators of a given space group were applied strictly in the sequence as printed in the header of the International Tables. Specifically, the centering generators were applied prior to all other generators. Each atom is immediately followed by the corresponding atom to which the centering vector has been added.

In R-centered space groups, a horizontal 2-fold axis, or a center of inversion turns the centering vector upside down, effectively converting it into the other centering vector. Thus in these space groups, the sequence of some atoms is not exactly that as printed in the Tables.

To adjust this, DISCUS now offers a new mode, in which the centering generators are applied after all other generators. Under this new mode, all atoms are in the sequence as printed in the International Tables. For centered space groups, this block of atom is followed by a corresponding block of atoms that is simply translated by the centering vector.

define generator,symmetry turns on the sequence in which symmetry generators are applied first define generator,center turns on the sequence in which centering generators are applied first

The "symmetry" mode now the default value.

A.21 demolecularize

```
demolecularize
```

Enters the menu to de-molecularize a molecule. The molecule entry is removed from internal list, and the molecule properties of the atoms that were in the selected molecul(s) are cleared.

deselect

```
deselect moletype:{"all"|"none"| <number> | (<n1>, <n2>...}
deselect atomtype:{"all"|"none"| <number> | (<n1>, <n2>...}
```

The optional parameter "moletype" deselects which molecule types are to be demolecularized. The parameter defaults to "all", which means that no molecule type will be demolecularized. The optional parameter "atomtype:" defines which atom types must be present in the selected molecule types. The parameter defaults to "all", which means that no atom type needs to be present. A molecule may contain any atom type to be demolecularized.

exit

```
exit
```

Leave the demolecularize menu, go back to the main DISCUS menu.

include

```
include molerange:{"none"|"all"| <number> | (<n1>, <n2>...)}
include atomrange:{"none"|"all"| <number> | (<n1>, <n2>...)}
```

The optional parameter "molerange" selects which molecule range are to be demolecularized. The parameter defaults to "none", which means that no molecule range will be demolecularized.
The optional parameter "atomrange:" defines whether atoms must be within the specified range if the selected molecules are to be demolecularized. The parameter defaults to "none", which means that no specific atom range needs to be present. A molecule may contain any atom range to be demolecularized.

prop

```
prop {"ignore"|"present"|"absent"} , <property> [, <property> ...]
```

Defines which properties all atoms in the molecule must have for the molecule to be demolecularized.
"ignore" The properties listed will be ignored. The listed properties do not have an affect on the choice to demolecularize the molecule or not. "present" All atoms in the molecule must have the properties listed, for the molecule to be demolecularized. "absent" All atoms in the molecule are not allowed to have the properties listed.
The properties can be one of: "all" Apply the rule to all properties. "normal" The rules apply if an atom is a normal atom (not a void) "molecule" The rules apply if an atom is part of a molecule

---

"domain" The rules apply if an atom is part of a domain "outside" The rules apply if an atom is outside of the crystal. This status is set if the atom has been cut off by a ==> 'boundary' command. "external" The rules apply if an atom is close to an external surface, either on the inside or on the outside. The flag is set by the ==> 'boundary' command. Individual distances to the boundary can be set within the ==> 'surface' menu through the ==> 'surface/fuzzy' command. "internal" The rules apply if an atom is close to an internal surface, either within the host or within the guest structure. The flag is set by placing a guest structure into the crystal through the ==> 'domain' menu. Individual distances to the boundary can be set within the ==> 'domain' menu through the ==> 'domain/fuzzy' command.

select

```
select moletype:{"none"|"all"| <number> | (<n1>, <n2>...)}
select atomtype:{"none"|"all"| <number> | (<n1>, <n2>...)}
```

The optional parameter "moletype" selects which molecule types are to be demolecularized. The parameter defaults to "none", which means that no molecule type will be demolecularized. The optional parameter "atomtype:" defines which atom types must be present in the selected molecule types. The parameter defaults to "none", which means that no atom type needs to be present. A molecule may contain any atom type to be demolecularized.

show

Show the current settings

reset

Reset to system start, which means that: no molecule types are selected the molecule range is set to zero any atom may be present no specific atom range needs to be present

run

Perform the actual calculation

A.22   diff-four

```
diff
```

Switches to the difference Fourier level of DISCUS. All input parameters are identical to "inverse" see the help to "inverse" for further details.
The difference Fourier can only be calculated from a "shelxl" type file. The program calculates (Fobs-Fcalc) and then calculates the corresponding inverse Fourier.

A.23   displacement

displacement <iatom> [,out:{"no"|"yes"}] [,aver:{"no"|"yes"|"off"}]

[,indi:{"no"|"yes"}]

This command calculates the displacement vector of atom number <iatom> from its average position. The result is stored in the result variables res[1] to res[3].

The optional parameters do: out:{"no"|"yes"} A short output will be written to screen Default is "no" aver:{"no"|"yes"|"off"} The average positions are calculated. Default is "no", it is much faster to do this once in the chem menu prior to an analysis of the displacement vector for several atoms. The first time that the 'displacement' command is used, DISCUS will automatically calculate the average structure, unless you explicitly turn this off with "aver:off".

The automatic calculation of the average structure is turned back on only with a DISCUS ==> 'reset' or the 'chem' 'reset' commands. If you modify the structure or read a new cell/structure the automatic calculation will NOT be turned on again. As this might give very wrong displacement vectors, it is highly recommended to calculate the average position with the "chem" command "aver" prior to the displacement commands. indi:{"no"|"yes"} The displacement vector is by default taken relative to the average of all atom species that may be on a given site. In analogy to the ==> 'aver indi' command in the ==> 'chem' menu the displacement vector of an atom may also be calculated relative to the average position of all atoms of the same chemical species.

A.24   domain

This command branches to the domain menu level.

Domains are understood as any group of atoms which is to replace part of the original structure. A domain may be encoded within the file in two different ways: A) A list of pseudo-atoms B) A list of explicit domains

The pseudo-atom file structure is identical to any DISCUS structure file. The atoms are interpreted as positions of the domains within the host structure. Size, character, and orientation of the domain can be set by the user via the 'assign' commands at the domain level.

The domain lists encode the size, character, and orientation within the file itself.

If domain successfully placed guests into the crystal, the crystal is no longer periodic. Thus periodic boundary conditions and the "chem" quick mode are off. They can be turned back on after explicit 'set crystal' command in the chemistry menu.

Valid commands at this sub level are:

commands

```
assign     ! Assigns size, character etc to the pseudo-atoms
inputfile  ! Defines the input file name
mode       ! Defines the interpretation of the input file
reset      ! Set definitions back to system start
run        ! Perform the actual replacement
set        ! Sets various definitions
show       ! Show current settings
```

Instructions on usage

These are some quick hints on how to use the domain level.

An explicit domain is encoded similar to a molecule:

domain domain character,domain_fuzzy domain file,domain_a.stru domain fuzzy,0.8 posi 0.0000 0.0000 0.0000 0.000 xaxi 1.0000 0.0000 0.0000 1.000 yaxi 0.0000 1.0000 0.0000 1.000 zaxi 0.0000 0.0000 1.0000 1.000 cent 0.0000 0.0000 0.0000 0.000 xdim 1.0000 0.0000 0.0000 1.000 ydim 0.0000 1.0000 0.0000 1.000 zdim 0.0000 0.0000 1.0000 1.000 domain end

This encoding allows you to use all of the DISCUS tools to create a distribution of domains within the host crystal. An alternative encoding is the use of "pseudo-atoms". These are regular atoms, which are interpreted by the domain level as positions of a domain.

See the ==> 'insert' command for instructions to insert a domain into a dummy structure.

The domain character may be any of "domain_cube", "domain_cylinder", "domain_fuzzy", or "domain_sphere". A "domain_cube" is a cube with dimensions given by the xdim, ydim, and zdim dimensions. These values are elements of a distortion matrix which allows you to rotate and distort the cube. A "domain_cylinder" is a cylindrical domain. The initial rotation axis is parallel to the z-axis. A "domain_fuzzy" domain has no predefined boundary. DISCUS reads all corresponding atoms from the file and deletes all host atoms, that are less than the fuzzy separation away from any of the domain atoms. A "domain_sphere" is an initially spherical domain.

A domain replaces all atoms that exist prior within the crystal. The exception are those atoms with special names that define a domain or an object. These atoms are not replaced by a new domain. This allows you to replace a structure in part with other domains, where these domains remain intact, even if they overlap.

See the manual for full details.

The file contains the actual atoms that will be inserted into the host structure. You must make sure that the atoms from this file to fill out the space freed by the domain.

The fuzzy value gives the minimum distance between domain atoms and host atoms. Any atom closer than this separation is deleted for the host structure. If the fuzzy distance is set to zero or a negative value, the check is not performed and all old host atoms remain in the structure.

The "posi" line is understood as the center of the domain atoms, the following three lines as the transformation matrix. The actual position onto which the domain atom is placed is calculated as: (t11 t12 t13) (xcoordinate) (posi(x)) (t21 t22 t23) (ycoordinate) + (posi(y)) (t31 t32 t33) (zcoordinate) (posi(z))

Each of the matrix elements is calculated as follows: t11 = xaxi(x) - posi(x) t12 = xaxi(y) - posi(y) .. t21 = yaxi(x) - posi(x) .. Thus the transformation matrix is actually encoded by adding the position to all matrix elements. This seemingly complication actually allows you to modify the domain by all DISCUS elements like the generalized symmetry operation ==> 'symm'. A rotation for example will thus actually leave the matrix nicely intact.

Likewise the next four lines are used to encode the transformation matrix used to define the domain shape. Again, the actual transformation matrix is the difference between the lines "*dim" and the "cent" line. By increasing the dimension, you increase the domain size. You can similarly distort the domain shape.

The generalized symmetry level ==> "symm" allows you to change the position, orientation

and shape of the domain and/or the atom transformation matrix.

assign

```
assign "character",<pseudoatom name>,{"cube"|"cylinder"|"fuzzy"|"sphere"}
assign "content",<pseudoatom name>,<contentfile>
assign "fuzzy",<pseudoatom name>,<radius>
assign "orient",<pseudoatom name>,<row>,<ti1>,<ti2>,<ti3>,<ti4
assign "shape",<pseudoatom name>,<row>,<ti1>,<ti2>,<ti3>,<ti4 [, <sigma>]
```

These commands allow you to assign extended properties to the pseudo atoms. Since the pseudo atoms are identical to regular atoms, they can only encode the position and type of a domain. The character, size, and orientation of the atoms with respect to the host structure may be assigned via these commands.

```
assign "character",<pseudoatom name>,{"cube"|"cylinder"|"fuzzy"|"sphere"}
```

A domain may be any of four different principal shapes. The "cube" domain is a domain consisting of the six {100} faces. The default distance of each face from the center is one respective unit cell. A "cylinder" domain has cylindrical geometry with the default cylinder axis parallel to the c-axis. The radius is one b-axis length. Top and bottom are terminated by the two {001} faces. A "sphere" domain initially is spherical with radius one b-axis.

All host atoms that are within the domain are removed, the void is filled by the atoms from the corresponding <contentfile>. The size and shape of these domains may be changed by adjusting the ==> ' assign shape' of the domain.

The "fuzzy" domain follows a very different concept. It does not have a predefined size or shape. Instead all atoms that are read from the <contentfile> replace host atoms that are close to their position.

```
assign "content",<pseudoatom name>,<contentfile>
```

The <contentfile> contains a DISCUS type structure file with all the atoms that are to be placed inside the host crystal. You need to make sure that the structure listed in the <contentfile> is large enough to fill the empty void created by the respective domain. The space group and unit cell dimensions of the domain structure and the host crystal must be identical.

```
assign "fuzzy",<pseudoatom name>,<radius>
```

All host atoms that are closer than this <radius> to any of the domain atoms are removed from the original host structure. This holds for all domain character types. If the <radius> is set to zero or any negative number, the check is not performed for this domain type. Atoms of this domain type could thus be right on top of an atom of the host structure!

```
assign "orient",<pseudoatom name>,<row_i>,<ti1>,<ti2>,<ti3>,<ti4>
```

The orientation provides you a transformation matrix that is applied to each domain atom prior to insertion into the host. To keep the input lines short, each of the three rows <row_i> must be given separately. Each atom is transformed by: (t11 t12 t13) (xcoordinate) (t14) (posi(x)) (t21 t22 t23) (ycoordinate) + (t24) + (posi(y)) (t31 t32 t33) (zcoordinate) (t34) (posi(z)) The translational part <ti4> is added to the position of the pseudo atom.

```
assign "shape",<pseudoatom name>,<row>,<ti1>,<ti2>,<ti3>,<ti4> [, <sigma>]
```

The shape transformation matrix is used to determine the size and shape of the domain. Default size at program start are +- one unit cell along each axis. Enlarge the domain by increasing the elements $<tii>$ of the shape matrix, distort by changing the off-axis elements. The shape is distorted by: (t11 t12 t13) (xaxis) (t14) (posi(x)) (t21 t22 t23) (yaxis) + (t24) + (posi(y)) (t31 t32 t33) (zaxis) (t34) (posi(z)) The translational part $<ti4>$ is added to the position of the pseudo atom. The shape transformation works independent from the atom transformation matrix. If the character of the domain is "cube", "cylinder", or "sphere", DISCUS checks whether the position of the domain atom after its transformation fits into the domain shape. If true, it is inserted into the guest structure.

The optional $<sigma>$ allows for a size distribution of the domains. The matrix part is multiplied by a factor (1 + gran(sigma)). Discus cuts the factor at 0.01, all large values are allowed.

inputfile

```
inputfile <inputfile>
```

DISCUS read the domain positions from the file $<inputfile>$. See the ==> 'mode' command on how DISCUS interprets this file.

mode

```
mode {"domain" | "pseudoatoms"}
```

This command tells DISCUS how to interpret the input file. With the "pseudoatoms" option, DISCUS expects to find a standard DISCUS structure file. Each "atom" in the file is interpreted as a domain position. See the ==> 'assign' commands for further details.

With the "domain" option DISCUS expects to read a file structured like: A general header like all DISCUS structure files followed by as many domain descriptors as needed. Each domain descriptor is designed similarly to the molecule and object descriptors:

reset

```
reset
```

Reset all Fourier definitions back to the default values at system start.

run

```
run
```

This starts the actual incorporation of the domains into the host crystal. The positions, characters etc. for each domain are read from file $<inputfile>$. The corresponding domain structure is used to replace part of the host structure.

---

set

```
set
```

Set various definitions

distance

```
set distance, {"external" | "internal" |"all" }, <at_list>, <distance>
```

Set the distances at which an atom from the atom list shall be considered to be close to a surface.

remove

```
set remove {"strict" | "initial" | "trust" | "none"}
```

When DISCUS replaces (parts of) an old structure with the content of domains. it needs to remove the old atoms. You have three choices how this shall be done:
strict Old atoms are removed EVERYTIME an individual domain is inserted into the crystal. Old atoms are considered to be all atoms that were in the crystal just before this domain was inserted. This includes domains that were inserted by the very same 'run' command. You need to do this, if you suspect or know that domains do overlap. You must use this for example to simulate overgrowth of domains on top of each other.
initial This time only those atoms are removed that were in the crystal prior to the current 'run' command. The atoms that are inserted during the current 'run' command are not tested. They may overlap if you are not careful.
trust OK, you trust that no domains overlap with each other and that you know exactly which host atoms have to be removed. The removal is performed after all domains have been inserted. DISCUS collects all fuzzy distances from the ==> assign commands and uses the maximum value of any assign command. This is necessary, since at this point the information from which domain type a new atom was taken has been lost. This removal is MUCH faster than the previous two.
none The ultimate trust! No host atoms are removed at all. You can do this, if you read a crystal that consisted only of dummy atoms and each of these represents a domain. You should remove these dummy atoms after you leave the 'domain' menu via a ==> replace dummy, void, all,1.0 command. This combination is the fastest but be advised that you really need to know where the dummy atoms and the inserted atoms are. Otherwise you might end up with lots of strange overlapping atoms.

show

```
show
```

Shows the current settings.

A.25   find

```
find "env",{"all"|{<name>|<number>} [, ...]},x,y,z,min_radius,max_radius
find "menv",{"all"|{<name>|<number>} [, ...]},x,y,z,min_radius,max_radius
```

This command serves to find something. The first parameter defines what DISCUS shall find.

```
"env"
```

DISCUS finds the environment around a given position. This environment consists of all atoms that are closer than a given radius. Only those atoms are included that fulfill the global property selection rules ==> 'property'.

```
"menv"
```

DISCUS finds the environment around a given position. It consists of all molecules that are closer than the given distance.

The last two parameters define the minimum and maximum radii of the shell within DISCUS will search for neighbors. The three parameters immediately prior to the radii define the central position whose environment is to be found. The second to last but five parameter(s) define which of the surrounding atoms are to be included in the environment. If the second parameter is "all" all atoms are included in the environment, else only those atoms specified by <name> or <number> of the scattering curve.

The number of neighboring atoms is stored in "env[0]", the indices are stored in the next <env[0]> elements of "env". The resulting bond lengths are stored in the first <env[0]> elements of "res". "res[0]" is set to the same value as "env[0]". Thus, for example, it is possible to search selectively for the next oxygen atoms, while ignoring all other atoms. The command returns the list of neighbors sorted by distance, shortest first.

Examples

```
find all,0,0,0, 0.1,3.05
```

Finds the environment consisting of all atoms at distances larger or equal to 0.1 Angstrom up to 3.05 Angstrom from the origin.

```
find env,o,n, 1.25,-5.25,3.34, 2.0,2.9
```

Finds all oxygen and nitrogen atoms that are between 2.0 and 2.9 Angstrom from position 1.25,-5.25,3.34

A.26   fourier

```
fourier
```

This sub level of DISCUS calculates the Fourier transform of the given crystal structure. The algorithm to speed up the explicit Fourier is based on the program 'diffuse' by B.D. Butler. See also: B.D. Butler & T.R. Welberry, (1992). J. Appl. Cryst. 25, 391-399. An even faster algorithm was introduced in Versions 6.15. This uses a non-uniform FFT.

The program allows to calculate X-ray, electron and neutron scattering including isotropic temperature factors and anomalous scattering. The program also allows to subtract the structure factor of the average structure from the calculated diffraction pattern.

You may calculate a single line through reciprocal space, a layer or a three dimensional volume. Further help topics are:

commands

Valid commands at this level are:

```
@         ! execute a macro file (see main help level)
=         ! assigns the value to a variable (see main help level)
abs       ! defines the coordinate of the abscissa
delf      ! defines the anomalous scattering corrections for an element
disp      ! switches anomalous dispersion on/off
calc      ! calculates at a single point in reciprocal space
continue! continue a stopped macro (see main help level)
echo      ! echo a string (see main help level)
electron! switches to electron calculation
energy    ! defines the energy of the radiation
eval      ! evaluates an expression for interactive check
exit      ! terminates Fourier level
help      ! help to Fourier commands (see main help level)
hkl       ! calculate a list of Bragg reflection intensities
layer     ! defines the layer in reciprocal space to be calculated
ll        ! defines the lower left corner in reciprocal space
lots      ! defines the crystal volume to be used for the Fourier transform
lr        ! defines the lower right corner in reciprocal space
nabs      ! defines the number of points along the abscissa
neut      ! switches to neutron diffraction
nord      ! defines the number of points along the ordinate
ntop      ! defines the number of points along the top axis
ord       ! defines the coordinate of the ordinate
reset     ! Reset and clear all definitions
run       ! starts the calculation
scat      ! defines the scattering factor for an element
set       ! sets Fourier mode, calculation technique
show      ! shows the current settings for the Fourier transform
stop      ! stops execution of a macro (see main help level)
system    ! executes operating system command (see main help)
temp      ! defines whether temperature coefficients are to be used
tl        ! defines the top   left corner in reciprocal space
top       ! defines the coordinate of the top axis
ul        ! defines the upper left corner in reciprocal space
wait      ! waits for user input (see main help level)
wvle      ! defines the wave length to be used
xray      ! switches to X-ray calculation
```

abs

```
abs <switch>
```

Defines which the component of the vector parallel to the abscissa is written to the output file.

calc

```
calc <h>,<k>,<l>
```

Calculates the structure factor at a single point h,k,l. The real value is stored in res[1], the imaginary value in res[2]. The corresponding values normalized to one unit cell are stored in res[3] and res[4], respectively. res[0] is set to 4 to indicate that 4 values may be found in res.

delf

```
delf { <name> | <number> } , <f'> , <f''>
delf { <name> | <number> | "all" } , "internal"
```

Defines for the element <name> or the scattering curve number <number> the real <f'> and the imaginary <f''> correction terms for the scattering factors. DISCUS calculates the effective atomic form factor as: f + f' + i*f'' This means that you should enter the value of f' as a negative number and f'' as a positive number.
The correction terms are used only, if the dispersion is set to anomalous ==> 'disp'.

```
delf { <name> | <number> | "all" } , "internal"
```

The second command form allows you to set the values back to the internally tabulated values. Keep in mind that these only work with wavelengths specified by their name as in ==> wvle moa1'. This command just sets the flag for the element. The values are actually not changed, until the anomalous dispersion is set on ==> 'disp'.

disp

```
disp [<switch>]
```

Switches anomalous dispersion on /off. If <switch> is "anom" anomalous dispersion will be calculated, if the switch is omitted, no anomalous dispersion is calculated.

electron

```
electron
```

switches to electron calculation

energy

```
energy <value>
```

Defines the energy of the radiation. For X-rays and electrons the value is to be given in units of keV, for neutrons in units of meV.

hkl

```
hkl "in:"<infile>, "out:<outputfile>, "scale:"<scale>, style:"hklf4"
hkl "in:"<infile>, "out:<outputfile>, "scale:"<scale>, style:"cif"
```

This command reads the SHELX HKLF4 file <infile> and calculates the intensity using the current structure. The data are written in a SHELX hklf4 type format (3I4,2x, F8.2,F8.2). The calculated intensities are multiplied by <scale>

---

layer

```
layer <e11,e12,e13, e21,e22,e23, e31,e32,e33, inc1,inc2>
```

Sets the lower left, lower right and upper left corner of the Fourier plane to be calculated and sets the number of data points along the horizontal and the vertical.

ll

```
ll <e11> , <e12> , <e13>
```

Sets the lower left corner of the Fourier plane to be calculated.

lots

```
lots "off"
lots {"box" | "ellipsoid"},<lx>,<ly>,<lz>,<n>,{"yes" | "no"}
lots {"box" | "ellipsoid"},<lx>,<ly>,<lz>,"all",{"yes" | "no"}
```

The program allows to Fourier transform large crystals by averaging the scattering from smaller volumes selected at random. This allows to avoid very long correlations which might occur if the complete crystal scatters in phase.

If this option is not wanted and the complete crystal should be transformed at once, this option can be turned off by the command 'lots off'. Otherwise the following parameters must be specified: First the lot shape. The choices are "box" or "ellipsoid". Usually the first letter of the parameter is sufficient. The following parameters $<lx>,<ly>$ and $<lz>$ describe the size of the sub-volumes in unit cells. The parameter $<n>$ specifies the number of such volumes to be averaged. If the number is chosen with the keyword "all", DISCUS will place the origin of the lots onto each unit cell and the number of the lots corresponds to the total number of unit cells in the crystal. The last parameter can be "yes" or "no" and selects if periodic boundary conditions should be applied.

The lot size should be slightly larger than the longest correlations present in the model crystal studied. The number of lots to be averaged should be set to a value that complete crystal will be covered once on average.

WARNING:

If the command 'purge' is performed or a structure is read where the atoms are not listed in the DISCUS order this command should be set to 'lots off'.

lr

```
lr <e21> , <e22> , <e23>
```

Sets the lower right corner of the Fourier plane to be calculated.

nabs

```
nabs <inc1>
```

Sets the number of data points that will be calculated along the abscissa.

---

neut

`neut`

Switches to neutron diffraction

nord

`nord <inc2>`

Sets the number of data points that will be calculated along the ordinate.

ntop

`ntop <inc2>`

Sets the number of data points that will be calculated along the top axis.

ord

`ord <switch>`

Defines which the component of the vector parallel to the ordinate is written to the output file.

reset

`reset`

Reset all Fourier definitions back to the default values at system start.

run

`run [ sigabs:[<sigma>, <h>, <k>, <l>]] [, sigord:[<sigma>, <h>, <k>, <l>]]`
`    [,sigtop:[<sigma>, <h>, <k>, <l>]]`

Starts the Fourier calculation.
The optional parameters enable you to apply a convolution with a triaxial Gaussian resolution function. The sigma's are in units of reciprocal Angstrom, keep in mind that DISCUS uses the definition a* = 1/a instead of a* = 2pi/a.
The sigma and the directions for the three half axes of the ellipsoid are independent. The directions <h>, <k>, <l> can be omitted. In this case the directions default to the abscissa, ordinate and top axis of the Fourier calculation.
The folowing information is stored in the variable res[] in order: Diffuse intensity minimun, maximum, average, standard deviation and Bragg intensity minimum and maximum.

scat

```
scat { <name> | <number> },<a1>,<b1>,<a2>,<b2>,<a3>,<b3>,<a4>,<b4>,<c>
scat { <name> | <number> | "all" }, <name>
scat { <name> | <number> | "all" }, "internal"
```

The first command form defines for the element <name> or the scattering curve number <number> a new scattering factor in the exponential form. The scattering factor is calculated as: f(h) = a(i) * e**(b(i)/4*h**2) + c(i) For neutron scattering lengths, set a(i) and b(i) to zero.

```
scat { <name> | <number> | "all" }, <name>
```

The second form of the command allows you to set the scattering factor of two elements to identical values. This comes in handy when you would like to use elements like O1, O2 etc for different oxygen species, or for dummy atom names like xxxx. The command scat O1,O would set the scattering factor of O1 to those values of oxygen. The second parameter must be a valid atom name. See the entry "atom names" for a list of valid names.

```
scat { <name> | <number> | "all" }, "internal"
```

The third form finally allows you to set the scattering factor back to the internally tabulated values.

set

```
set "aver" [, <per>]
set {"external"|"internal"}
set {"exti:"|<par>}
set {"mode:single" | "mode:init" | "mode:accumulate" | "mode:finish"}
set {"symm:apply" | "symm:friedel" | "symm:ignore" }
set {"technique:turbo" | "technique:nufft"}
set {"filter:lanczos", "damp:"<r>, "width:"<npoints>, "scale:"<factoor>yy}
set "aver" [, <per>]
```

The Fourier level of DISCUS allows to subtract the average structure factor <F> by setting the parameter <per> to a value not equals zero. The value of <per> sets the percentage of the crystal to be used to calculate <F>.
The average is calculated by putting <per> % of the crystals atoms into a single unit cell which is Fourier transformed and multiplied with the interference function of the lot shape.

WARNING:

If the command 'purge' is performed or a structure is read where the atoms are not listed in the DISCUS order the average structure will not be calculated correctly !

```
set {"mode:single" | "mode:init" | "mode:accumulate" | "mode:finish"}
```

As default mode, DISCUS will calculate a single Fourier transform that can be written to an output file. As an alternative to the ==> 'lots' command, you can also perform a simulation and calculate a diffraction pattern and then accumulate several such calculations into the final pattern. Prior to such a loop, use within the fourier menu the command set mode:init During the loop use 'set mode:accumulate' After the loop go back to fourier and use: set mode:finish before you use the ==> 'output' menu to write the data.

`For a small to moderate crystal, the actual realization of the disorder`

might not match the crystals symmetry very well. you can improve the situation by applying the point group symmetry to the diffraction pattern with: set symm:apply This optional parameter can be combined with the 'mode:finish' parameter and will apply both to a single as well as to an accumulated pattern.

```
set "symm:apply"
set "symm:friedel"
set "symm:ignore"
```

If set to "apply" Discus will apply the point group symmetry to the data calculated in reciprocal space. This will ensure that the calculated diffraction pattern reflects the crystals point group symmetry. If the crystal is small, statistical fluctuation introduced for example in the Monte-Carlo part may otherwise create a diffraction pattern that will seem to reflect lower symmetry. If set to "friedel", DISCUS will reduce the calculation time for the ==> "technique:turbo" mode. Only half of reciprocal space is calculated, the remainder is generated by the "-1" symmetry operation. For the "friedel" technique to work, anonalous scattering must be off and the Fourier volume must consist of an odd number of pixels symmetrically around reciprocal origin.

```
set {"external"|"internal"}
```

Usually DISCUS will assume that all atoms in the crystal represent real atoms. In this case the internally tabulated atomic form factors are used to calculate the Fourier transformation. You can explicitly set this mode by typing "set internal". As of version 3.4.2 DISCUS can handle extended objects such as cubes, cylinders, and spheres. Their Fourier transform is calculated using the appropriate analytical expression. In order to activate this state, you must set the Fourier to "external".

```
set "exti:"<par>
```

Set an extinction correction parameter. The amplitudes are multiplied by [1.0 + 0.001 * <par> * ampl**2 * lambda**3 / sin(2Theta)] ** (-0.25) This equation is taken from SHELXL. Development note, currently the extinction correction is applied only to output format "hklf4" and "list5"

```
set {"technique:turbo"}
set {"technique:nufft"}
```

DISCUS offers two techniques for the Fourier calculation. The "turbo" technique is based on the algorithm described in Butler and Welberry. Its acceleration is based on a tabulated cos/sine function and quick addressing of the corresponding Fourier parameters. The "nufft" technique uses the Non-Uniform FFT. It is about two orders of magnitude faster than the "turbo" technique. Currently the technique is restricted to a Fourier volume/plane/line that is parallel to the three reciprocal base vectors: abscicca parallel a*; ordinate parallel b*, top axis parallel c*. Correspondingly the original crystal must consist of a block terminated by the {100}; {010} and {001} facets. If the diffraction pattern should be calculated in a different orientation, use the transformation option at the ==> 'output' menu or while reading the data in KUPLOT. The "turbo" mode in combination with the ==> 'lots' option relies on the crystal to consist of a block shaped crystal with identical atom numbers per unit cell. The "nufft" mode does not require all unit cells to have the same amount of atoms per unit cell, this holds for the subtraction of the Bragg scattering as well.

```
{"filter:lanczos", "damp:"<r>, "width:"<npoints>, "scale:"<factor>}
```

The nufft techniques calculates the FFT based on all atoms of the crystal. This will often give rise to high frequency oscillations that arise from correlations between atoms at large distances. These distances are in most cases beyond the intended correlation length achieved for example in the ==> 'mmc' menu. To avoid these oscillations a Lanczos filter can be used. The parameter "damp:" defines the dampening of points at larger distance. A smaller value will reduce the influence of points further away. The parameter "width:" determines how many neighboring points are included in the filter operation. The "scale:" parameter allows to create a new grid that is finer by the <factor>, i.e. has <factor> times more data points. "scale" must be an integer number >=1

show

```
show
show "scat", {<name> | <number> } [, {<name> | <number> }...]
```

Shows the current settings for the Fourier transform file
The second form shows the scattering curve of the selected atom types, which may be given as the scattering number of as the element name.
The values for the scattering curve are stored into the res[] variable in the order: a1, a2, a3, a4, b1, b2, b3, b4, c

temp

```
temp <switch>
```

Sets whether temperature coefficients are to be used or not. Valid values for <switch> are "use" and "ignore". The first two letters of the switch are sufficient.

tl

```
tl <e41> , <e42> , <e43>
```

Sets the top left corner of the Fourier volume to be calculated.

top

```
top <switch>
```

Defines which the component of the vector parallel to the top axis is written to the output file.

ul

```
ul <e31> , <e32> , <e33>
```

Sets the upper left corner of the Fourier plane to be calculated.

wvle

```
wvle { <name> | <value> }
```

Defines the wave length to be used. Anomalous dispersion will only be calculated if the wave length is given as symbol.

xray

```
xray  [table:{"waas" | "inter"}
```

Switches to X-ray calculation
The parameterized values according to Waasmaier & Kirfel, ACA (1995) A51, 416-431 were added and are used as default values. Optional parameters table:waas or table:inter allow to choose the Waasmaier or the International Tables Volume C.

A.27   functions

The following DISCUS specific functions exist. For a listing of general intrinsic functions see help entry 'functions' in the 'Command language' section of the online help.

```
bang(u1,u2,u3,v1,v2,v3[,w1,w2,w3])
                             ! Returns the bond angle at the site v
                               If only vectors u and v are given, the
                               angle between u and v is returned.
bang(atom,i1,i2[,i3])        ! Returns the bond angle at atom i2.
                               If only atoms i1 and i2 are given, the
                               angle at the origin between the vectors
                               to atoms i1 and i2 is calculated.
bang(envi,i1,i2)             ! Returns the bond angle in the center of
                               the last find env command. The angle
                               between the vectors from the center to
                               entries i1 and i2 in the environment is
                               calculated.
blen(u1,u2,u3[,v1,v2,v3])    ! Returns the length of vector v-u.
                               Vector v defaults to zero.
blen(atom,i1,i2)             ! Returns the length of the vector
                               between atoms i1 and i2
blen(envi,i1)                ! Returns the distance from the center of
                               the last environment to atom i1 in the
                               environment
dstar(h1,h2,h2[,k1,k2,k3])   ! Returns the length of reciprocal
                               vector k-h. Vector k defaults to zero.
mol_test(i1)                 ! Returns the molecule number that contains
                                atom number <i1>. If the atom is not part
                               of any molecule zero is returned. The
                               result is stored as res[1] as well.
                               res[2] receives the position of the atom
                               within the molecule.
rang(h1,h2,h3,k1,k2,k3[,l1,l2,l3])
                             ! Returns the angle between vectors
                               k-h and k-l at reciprocal site k.
                               If l is omitted, the angle between the
                               reciprocal vectors h and k is returned.
scalpro(u1,u2,u3,v1,v2,v3 [,{"dd"|"rr"|"dr"|"rd"}])
                             ! Returns the scalar product between
                               the two vectors u and v.
```

```
                                  Both vectors may be given in direct
                                  or real space coordinates, flagged
                                  in parameter no 7. "d" means real space,
                                  "r" reciprocal space.
gran(val,typ)                   ! Returns Gaussian distributed pseudo
                                  random number with mean zero and a width
                                  given by parameter <val>. If <typ> is
                                  "s" <val> is taken as sigma, if <typ>
                                  is "f", <val> is taken as FWHM.
gbox(r1,r2,r3)                  ! Returns pseudo random number with
                                  distribution given by a box centered
                                  at 0 with a width of <r2> and two half
                                  Gaussian distributions with individual
                                  sigmas of <r1> and <r3> to the left and
                                  right, respectively.
```

The arguments to any of these functions are any arithmetic expression.

```
isprop(i1, propstring)
isprop(i1, and:propstring [,or:propstring])
                                ! Returns a logical true if the properties
                                  encoded in <propstring> are true for
                                  atom number <i1>.
                                  Valid forms for the property string are
                                  individual names of the properties as:
                                  'normal', 'molecule', 'domain',
                                  'outside', 'external', 'inside',
                                  'ligand'.
                                  The function will thus be used to query
                                  exactly one property.
                                  Alternatively the propstring can take
                                  the form 'NMDOEIL' or 'nmdoeil'
                                  The number of letters and their sequence
                                  is up to the user.
                                  With a property string in capital
                                  letters as "and:'NMDOEIL'" the function
                                  is true if all properties that are
                                  listed are true for atom <i1>.
                                  With a property string in small
                                  letters as "and:'nmdoeil'" the function
                                  is true if all properties that are
                                  listed are false for atom <i1>.
                                  Capital and small letters can be
                                  mixed at will.
                                  With a leading "or:" the function is
                                  true if any of the properties in
                                  capital letters is present and any
                                  of the properties in small letters is
                                  absent.
                                  Both, the "and:" and "or:" arguments
                                  may be used simultaneously to form
                                  complex queries.
```

A.28   import

```
import "cif",<file[".cif"]> [, "names:cif", | "names:chem"]
import "cmaker",<file[".xyz"]>
import "lammps",<file>
import "shelx",<file[".ins"]> [, "names:shelx" | "names:chem"]
```

```
                    [, "refine:[parameters]" | "diffev:[parameters]"]
import "rmcprofile",<file[".cssr"]>
          [, "sort:discus" ]
import "cell",<file>
       [, "metric:host" [,"space:host"] ]

import "stru",<file> , "metric:host" [,"space:host"]
```

This command converts other crystal structure formats into a DISCUS unit cell.

The file name may be presented with format specifications as in "gold%3D.cif", 3 for a file name 'gold003.cif'

The optional parameter "metric:host" instructs DISCUS to perform a unit cell transformation of the input structure into the metric of the crystal structure that is currently present within DISCUS. As the two coordinates system will usually differ considerably, the space group of the imported structure is set to P1 unless the optional parameter space:host is used which changes the space group to that of the present structure, the host.

In order to transform a DISCUS file format the "metric:host" parameter becomes mandatory.

cif

```
"cif", <file{".cif"]>[, "names:cif", | "names:chem"]
```

Imports a CIF file into the corresponding discus "stru" file.

As default, the atom names are taken from the "symbol" entry in the atom list and used for the individual atoms. With the optional parameter "names:cif" the atom labels are used instead.

Multiple "data_" entries are written to separate files. Entries 2 etc have the corresponding number appended to the file name.

cmaker

```
"cmaker", <file{".xyz"]>
```

Imports a CrystalMaker file into the corresponding discus "stru" file.

lammps

```
import "lammps",<file>
```

Imports a LAMMPS output file. At the present, a simple dump output file is imported. DISCUS will read the input file and create a file <file>".stru" in the DISCUS cell format.

shelx

```
"shelx",<file{".ins"]>
                       [, "names:shelx" | "names:chem"]
     [, "refine:[parameters]" | "diffev:[parameters]"]
```

If the first parameter is "shelx" the input file is assumed to be a SHELX "*.ins" or "*.res" file. The extension ".ins" may be omitted. DISCUS will read the input file and create a file <file>".cell" in the DISCUS cell format. DISCUS will interpret the Lattice type and the symmetry instructions

to determine the space group. In the unlikely event of this failing, the space group is set to P1, and the SHELX symmetry instructions are written as generators. As default, the atom names are taken from the SHELX "SFAC" instruction and used for the individual atoms. with the optional parameter "names:shelx" the actual atom names that follow the "FVAR" instruction are used instead. Atom coordinates are properly interpreted including the use of SHELX free variables.

If the optional parameter "refine:[parameters]" is present, DISCUS will write the cell file, a RE-FINE main macro, a main DISCUS macro and a KUPLOT macro ready to perform the equivalent refinement throught the REFINE section. the "parameters" are a list of optional parameters to specify details: style:single ! The macros are set up for a single crystal refinement style:powder ! Setup for a powder diffraction refinement style:pdf ! Setup for a PDF refinement lambda:value ! Defines a numerical value for the wave length lambda:name ! Defines a wave length symbol like Cu, CUA1, CU12 etc form:waas ! Use the Waasmeier-Kirfel atomic form factors form:discamb ! Use the DISCAMB generated atomic form factors

If the optional parameter "DIFFEV:[parameters]" is present, DISCUS will write the cell file, a DIFFEV main macro, a main DISCUS macro and a KUPLOT macro ready to perform the equivalent refinement throught the DIFFEV section. the "parameters" are a list of optional parameters to specify details: style:single ! The macros are set up for a single crystal refinement style:powder ! Setup for a powder diffraction refinement style:pdf ! Setup for a PDF refinement lambda:value ! Defines a numerical value for the wave length lambda:name ! Defines a wave length symbol like Cu, CUA1, CU12 etc form:waas ! Use the Waasmeier-Kirfel atomic form factors form:discamb ! Use the DISCAMB generated atomic form factors compute:serial ! If multiple individual calculations a needed, they are set up to be done serially compute:parallel ! If multiple individual calculations a needed, they are set up to be done in parallel

LIMITATIONS The "DISP" instruction is ignored. Explicit form factors on the "SCAT" instruction are ignored ALL instructions following the "FVAR" instruction up to the "HKLF" instructions are interpreted as atoms, unless they are valid SHELX commands. The site occupation factors are ignored. Currently DISCUS does not recognize the internal symmetry of a molecule.

rmcprofile

DISCUS will read a RMCProfile *.cssr or "rmcf6" file and write a corresponding structure file. If the optional parameter "sort:discus" is present, DISCUS will attempt to sort the structure in the DISCUS style, breaking it down into the individual unit cells.

A.29   index2cell

```
index2cell <atomindex> or
index2cell <x>,<y>,<z>,<site>
        ",output:screen" | ",output:none"
        ",boundary:periodic" | ",boundary:crystal"
```

This command allows to switch between a given index of the storage array <atomindex> and the corresponding unit cell <x>,<y>,<z> and site <site>. The parameter not given is calculated ...

The optional parameter "boundary:periodic" will apply periodic boundary conditions to the unit cell indices <x>, <y>, <z>

The optional parameter "output:none" will suppress the screen output.

index2cell stores the results in the "res[]" variable: res[1] : <atomindex> res[2] : <x> res[3] : <y> res[4] : <z> res[5] : <site>

The command is identical to the ==> 'trans' command in the chemistry menu.

A.30   insert

```
insert <name>, <x>,<y>,<z> [,<t>]
insert {"domain"|"object"|"molecule"}
```

If an atom name or atome type number is given, it will insert an atom of type <name> at the position <x,y,z> in crystal space. Optionally the temperature coefficient may be given. This will automatically turn off the periodic boundary conditions.

If one of the keywords "domain", "object" or "molecule" is given as second parameter, the menu to insert domains, objects or molecules opens.

Objects are special molecules, that represent extended objects. DISCUS treats these objects as uniformly and continuously filled matter. Depending on the shape of the object, specialized Fourier transformations are calculated. Objects consist of four pseudo atoms with special names, that characterize the location, shape and orientation of the object.

Domain are special molecules as well, which are used to represent a guest structure within a host.

Further help topics are:

commands

```
adp        ! Defines the atomic displacement parameter of an object
character  ! Defines the object/domain character = cube, cylinder, sphere
cent       ! Defines the center of atoms within a domain
density    ! Defines the scattering density of the object
file       ! Defines the file name with the actual domain content
fuzzy      ! Defines the distance between domain atoms and host atoms
origin     ! Places the origin of the object/domain in real space coordinates
reset      ! Reset and clear all definitions
run        ! Does the actual insertion
show       ! Show the current parameters
type       ! Defines the object type number
xaxis      ! Gives a deformation/scaling of the X-axis
yaxis      ! Gives a deformation/scaling of the Y-axis
zaxis      ! Gives a deformation/scaling of the Z-axis
xdim       ! Gives a deformation/scaling of the domain shape
ydim       ! Gives a deformation/scaling of the domain shape
zdim       ! Gives a deformation/scaling of the domain shape
```

adp

```
adp <value>
```

character

`character {"cube"|"cylinder"|"sphere"}[,<atom name>]`

Defines the character of the object, which may be one of {"cube"|"cylinder"|"sphere"}. Depending on the character, different formulae are used to calculate the Fourier transformation.

A "cube" object is defined by the six {100} faces and thus extends along each axis from -1. to +1. Its volume is thus eight times the volume of the unit cell. For a non-cubic unit cell, the faces are inclined accordingly.

A "cylinder" object has it's cylinder axis along the [001] axis and extends from -1 to +1. The cross section in the x,y plane has a radius of 1. Thus it's volume is 2*pi*2 times the unit cell volume. For non cubic unit cells, the cylinder will be ellipsoidal and possibly oblique. The top and bottom limiting ellipsoids are parallel to the two {001} faces.

A "sphere" object has a radius of 1. and a volume of 4pi/3 times the unit cell. For non cubic unit cells, a general triaxial ellipsoid results.

The second, optional, parameter sets the atom type for the object. DISCUS will now assume that the object consists of atoms of this type, or at least that most of the atoms correspond to this type.

cent

density

`density <value>`

Defines the scattering density of the object type. The scattering factor calculated for objects of this type is multiplied by this density. This allows to simulated objects of identical volume yet different scattering density like bubbles of a denser material versus bubbles of empty space.

file

fuzzy

origin

`origin <x>,<y>,<z>`

The center of the object is placed at the vector <x>,<y>,<z>. DISCUS describes this position by the coordinates of the first of the atoms. It's name represents the 'character' of the object

reset

`reset`

Reset all insert definitions back to the default values at system start.

run

Performs the actual insertion.

show

Shows the current settings

type

`type {<number>|"new"}`

Each object/molecule is assigned a type number and you may use this to perform the same operation on objects/molecules of identical type (like ==> symmetry, shear, plot).
If <number> is given as a positive integer, the object/molecule will receive this number, which must correspond to an existing type.
If "new" is given, the object/molecule will receive the next higher unused type number.

xaxis

`xaxis  <x>,<y>,<z>`

Defines in crystal space the scaling /and/or deformation of the x-axis. A set of values like 2,0,0 would mean a doubling of the objects x-axis.
DISCUS does not perform a check, whether the values that you entered correspond to any special deformation tensor. The ==> 'shear' menu provides elegant ways to deform an object.

yaxis

`yaxis  <x>,<y>,<z>`

Defines in crystal space the scaling /and/or deformation of the y-axis. A set of values like 2,0,0 would mean a doubling of the objects x-axis.
DISCUS does not perform a check, whether the values that you entered correspond to any special deformation tensor. The ==> 'shear' menu provides elegant ways to deform an object.

zaxis

`zaxis  <x>,<y>,<z>`

Defines in crystal space the scaling /and/or deformation of the z-axis. A set of values like 2,0,0 would mean a doubling of the objects x-axis.
DISCUS does not perform a check, whether the values that you entered correspond to any special deformation tensor. The ==> 'shear' menu provides elegant ways to deform an object.

A.31   internal

`Internal storage of the structure`

When reading a new cell, the program stores the content of each unit cell consecutively. The program always generates a block of nx * ny * nz unit cells. The innermost -fastest- loop is nx, the slowest nz.

A.32   inverse

```
inverse
```

Branches to the inverse Fourier transform level of DISCUS The inverse Fourier transform is calculated from two input files. The two files must contain the intensity, real or imaginary part amplitude or phase in one of the following combinations:

```
file a          file b
------------------------
intensity       phase
amplitude       phase
real            imaginary
```

The user must specify the Fourier plane that had been calculated and the real space plane to be calculated.
Further help topics are:

commands

Valid commands at this level are:

```
@        ! execute a macro file (see main help level)
=        ! assigns the value to a variable (see main help level)
abs      ! defines the coordinate of the abscissa
continue! Continue a stopped macro (see main help level)
echo     ! echo a string (see main help level)
eval     ! Evaluates an expression (see main help level)
exit     ! terminates Fourier level
file     ! defines the input file name(s)
form     ! defines the input file format
help     ! help to Fourier commands (see main help level)
layer    ! defines the layer in reciprocal space to be calculated
ll       ! defines the lower left corner in reciprocal space
lr       ! defines the lower right corner in reciprocal space
na       ! defines the number of points along the abscissa
no       ! defines the number of points along the ordinate
ord      ! defines the coordinate of the ordinate
reset    ! Reset all Fourier definitions
reset    ! Reset all Fourier definitions
rhoabs   ! defines the coordinate of the real space abscissa
rholayer! defines the layer in real space to be calculated
rholl    ! defines the lower left corner in real space
rholr    ! defines the lower right corner in real space
rhona    ! defines the number of points along the real space abscissa
rhono    ! defines the number of points along the real space ordinate
rhoord   ! defines the coordinate of the real space ordinate
rhoul    ! defines the upper left corner in real space
reset    ! Reset and clear all definitions
run      ! starts the calculation
scale    ! defines an overall scale factor
set      ! sets various parameters
show     ! shows the current settings for the inverse Fourier transform
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
ul       ! defines the upper left corner in reciprocal space
wait     ! Waits for user input (see main help level)
```

abs

```
abs <switch>
```

Defines which the component of the vector parallel to the abscissa is written to the output file.

file

```
file {"a"|"b"},<filename>
```

Defines the filenames needed for the inverse Fourier.
The inverse Fourier transform is calculated from two input files. The two files must contain the intensity, real or imaginary part amplitude or phase in one of the following combinations:

```
file a          file b
-----------------------
intensity
amplitude
real            imaginary
```

If the input format is 'shelxl' then one input file is sufficient. DISCUS assumes the input file to be a SHELXL List type 5 containing <h> <k> <l> <Fobs> <Fcalc> <phase angle in degree> The file is read as free format.

format

```
form {"gnu" | "stan" | "shelxl"}
```

Defines the input format that will be used. Valid parameters are:

```
"gnu"
```

Sets the type of the output file to gnuplot. The intensities are written in blocks of intensities along the abscissa, with an empty line separating each block. Each line of each block contains: index along abscissa, index along ordinate, intensity, third index.

```
"stan"
```

Sets the output to standard file format, which is used by KUPL. The output file contains two initial lines and then a block for each data line separated by an empty line. The first line contains the number of data points along the abscissa and the ordinate, the second line the minimum and maximum value of the indices in reciprocal space along the abscissa and ordinate respectively.

```
"shelxl" [, {"fobs" | "fcalc"}]
```

Sets the input file format to SHELXL. DISCUS assumes the input file to be a SHLEXL List type 5 containing <h> <k> l> <Fobs> <Fcalc> <phase angle in degree> The file is read as free format. All reflections are expanded by the appropriate symmetry operations, including proper phase transformation.
If the optional parameter is "fobs", the observed structure factor is used for the calculation, if the parameter is "fcalc", the calculated structure factor is used.

---

layer

`layer <e11,e12,e13, e21,e22,e23, e31,e32,e33, inc1,inc2>`

Sets the lower left, lower right and upper left corner of the Fourier plane to be calculated and sets the number of data points along the horizontal and the vertical

ll

`ll <e11> , <e12> , <e13>`

Sets the lower left corner of the Fourier plane to be calculated.

lr

`lr <e21> , <e22> , <e23>`

Sets the lower right corner of the Fourier plane to be calculated.

na

`na <inc1>`

Sets the number of data points that will be calculated along the abscissa.

no

`no <inc2>`

Sets the number of data points that will be calculated along the ordinate.

ord

`ord <switch>`

Defines which the component of the vector parallel to the ordinate is written to the output file.

rhoabs

`rhoabs <switch>`

Defines which the component of the vector parallel to the real space abscissa is written to the output file.

rholayer

`rholayer <e11,e12,e13, e21,e22,e23, e31,e32,e33, inc1,inc2>`

Sets the lower left, lower right and upper left corner of the real space plane to be calculated and sets the number of data points along the horizontal and the vertical

---

rholl

`rholl <e11> , <e12> , <e13>`

Sets the lower left corner of the real space plane to be calculated.

rholr

`rholr <e21> , <e22> , <e23>`

Sets the lower right corner of the real space plane to be calculated.

rhona

`rhona <inc1>`

Sets the number of data points that will be calculated along the real space abscissa.

rhono

`rhono <inc2>`

Sets the number of data points that will be calculated along the real space ordinate.

rhoord

`rhoord <switch>`

Defines which the component of the vector parallel to the real space ordinate is written to the output file.

rhoul

`rhoul <e31> , <e32> , <e33>`

Sets the upper left corner of the real space plane to be calculated.

reset

`reset`

Reset all Fourier definitions back to the default values at system start.

run

`run`

Starts the inverse Fourier calculation.

scale

```
scale <scale-factor>
```

Defines an overall scale factor. Together with the automatic scaling by 1/vol(unit cell), this scale factor serves to put the electron density on an absolute scale of electrons per cubic Angstrom. The scale factor should effectively multiply the observed F(000) such that it is equal to F(000) calculated for one unit cell of the average structure.

Every time you enter "inverse" the scale factor is calculated to be number of atoms in crystal / number of atoms in one unit cell. If you use experimental data, you MUST specify the scale factor every time you enter the "inverse" sub menu.

set

```
set "accu",{"init"|"add"}
set "excl",{none"|<value>}
set "rsym",{appl"|"igno"}
```

Set defines various parameters.

set "accu",{"init"|"add"}

Sets the mode for the accumulation of inverse Fourier transforms.

```
"init"
```

The electron density array is initialized, i.e. set to zero.

```
"add"
```

The next inverse Fourier transform is added to the previous value. This way several inverse Fourier transforms will contribute to a single layer in direct space, which is necessary to create a inverse Fourier section rather than a projection onto the layer. See the manual for further details.

If you use DISCUS to calculate the input data for the inverse Fourier, you MUST separate the loop that calculates the Fourier from the loop that calculates the inverse Fourier!!! Both parts of the program use the same variable to store the results and by mixing the loops, the result will be unpredictable nonsense !

"excl",{"none"|<value>}

Signals whether certain data points should be ignored.

```
"none"
```

All data points are treated as normal data points, regardless of their value. This is the default at program start.

```
<value>
```

Data points that are equal to <value> are excluded from the inverse Fourier transform. You can use this option to mask out regions of the input data by setting their values to <value>. In the companion program KUPLOT, data points of value equal to -9999. are treated as excluded regions.

set "rsym",{"appl"|"igno"}

Determines whether the space group symmetry is applied to all reflections. If the second parameter is "appl", all symmetry elements are applied to all reflections and the phases are transformed accordingly. This allows you to read a set of unique reflections and calculate the inverse Fourier.

show

```
show
```

Shows the current settings for the Fourier transform file

type

```
type {"a"|"b"},
     {"intensity"|"amplitude"|"real"|"imaginary"|"phase"|"fobs"|"fcalc"}
```

Defines the content of file "a" or "b".
The inverse Fourier transform is calculated from two input files. The two files must contain the intensity, real or imaginary part amplitude or phase in one of the following combinations:

```
file a          file b
-----------------------
intensity       phase
amplitude       phase
real            imaginary
```

If the input format is "shelxl", DISCUS reads just one file. You can calculate the inverse Fourier from Fobs or Fcalc data by choosing the appropriate file type from one of the following choices:

```
file a          file b
-----------------------
amplitude
fobs
fcalc
```

ul

```
ul <e31> , <e32> , <e33>
```

Sets the upper left corner of the Fourier plane to be calculated.

A.33   kick

```
kick <name>, <x>,<y>,<z> ,<t>, <na>,<ne>, <delx> [,<dely> [,<delz>]]
kick <name>, <x>,<y>,<z> ,<t>, <na>,<ne>, -<bondlength>
```

Inserts an atom of type <name> at the position <x,y,z> in crystal space. The temperature coefficient must be given. 'kick' checks the position <x>,<y>,<z> with respect to all atoms numbered <na> to <ne>. All atoms that are within the block <x> +- <delx>; <y> +- <dely>; <z> +- <delz> are removed from the structure.

If <dely> and/or <delz> are not given, then they default to the value of the last <del.> given on this 'kick' command.

Note that the <del.> are in fractional coordinates, NOT in Angstrom.

In the alternative form, the eight's parameter is interpreted as a bond distance. To distinguish the two forms, the <bondlength> must be less than zero. The new atom is inserted and all atoms within <bondlength> Angstrom from the position <x>,<y>,<z> are removed.

The parameters <na> and <ne> must be in the range 1 to current number of atoms in the crystal i.e. the value of variable "n[1]", and <na> must be less or equal to <ne>. The one exception is <na> = n[1]+1 and <ne> = n[1]. This allows you to start building e.g. a new layer within which you would like to optionally insert atoms without affecting the previous layer. Example i[0] = n[1]+1 do i[1]=1,10 insert Al,ran(0),ran(0),0.5, 1.0, i[0],n[1], -2.4 enddo This loop inserts up to 10 Al atoms in the x,y range 0 to 1 at z=0.5 with a minimum distance of 2.4 Angstrom. Since i[0], the value for <na> is initially at n[1]+1, all previously inserted atoms are not affected. Since i[0] remains fixed within the loop, each new "Al" atom affects all "Al" atoms previously inserted within this loop.

A.34   mc

```
mc
```

NOTICE: The "mc" level has been depreciated and is no longer supported. Please move to the "mmc" level, which has the same functionality plus lots of extra possibilities.

The "mc" level has been removed in this version of the program.

Enter "Monte Carlo" section of DISCUS. This sub level allows to create disordered structures with given correlations and/or displacements using MC simulations. See help entry 'commands' for a complete list of commands available in this sub level.

NOTE: The MC and CHEM sub level share some variables which define neighbors etc. and settings made in this sub level might be altered when using the other sub level. It is always save to repeat settings when entering this sub level if both levels are used.

commands

Valid commands at this sub level are:

```
@       ! Executes a macro (see main help)
=       ! Algebra (see main help)
continue! Continue a stopped macro (see main help level)
echo    ! Echos a string, just for interactive check
eval    ! Evaluates an expression (see main help)
exit    ! Terminates the MC sub level, returns to the main DISCUS level.
help    ! Gives on-line help for 'mc' (see main help)
reset   ! Reset and clear all definitions
run     ! Starts MC simulation
save    ! Saves structure (see main help -> save)
set     ! Sets most MC parameters
```

```
show    ! Shows current settings
stop    ! Stops execution of a macro (see main help level)
system  ! Executes operating system command (see main help)
wait    ! Waits for user input (see main help)
```

run

```
run
```

This command starts the MC simulation for the given parameters and number of MC moves.

set

```
set <subcommand>
```

This command allows to define the wanted correlations and most other parameters for the MC simulation. The help entry 'commands' contains a list of valid subcommands for the 'set' command. Further help can be found for the following subcommands:

commands

```
angle   : Definition of interaction angles
atom    : Setting of atom types for MC simulation
const   : Setting of interaction constants
cyc     : Setting of maximum number of MC moves
energy  : Setting of MC mode (energy calculation)
feed    : Setting of display/feedback interval
mode    : Setting of MC mode (crystal modification)
mole    : Setting molecule types for MC simulation
move    : Setting size of generated SHIFT moves
neig    : Setting of the neighbor definitions
target  : Setting of correlations values to be achieved
temp    : Setting of MC simulation temperature
vec     : Definition of interaction vectors
```

angle

```
set "ang",<iv>,<is1>,<is2>,<dx1>,<dy1>,<dz1>,<is3>,<dx2>,<dy2>,<dz2>
```

This command is used to define correlation angles that might be used to calculate correlations within the crystal (-> 'set corr'). The complete description is given in the CHEM section of the online help (-> set vec).

atom

```
set "atom",{ <name> | <number> }, { <name> | <number> }
        [,{ <name> | <number> }]
```

This command allows the user to select the atoms types used for the MC simulation. They can either be specified by <name> (e.g. Zr) or <number>. Not that different MC energy definitions have different restrictions on the selection of the atoms. The angular distortions need three angles, while the other ones need just two.

const

set "const",<ic>,<const> [,<fac>]

The introduction of certain correlations depend on a given set of near neighbor interactions. These values are initially unknown and are adjusted using a feedback method (-> set feed). This command allows the user to set initial interaction parameters <const> for the corresponding interaction/correlation <ic>. The optional last parameter <fac> is a factor used in the feedback process and controls how 'fast' the interactions are changed in each feedback cycle.

cyc

set "cyc",<number>

This command sets the number of MC moves (i.e. visited sites) to be executed by the 'run' command. The number <number> actually defines the number of 'tried' MC moves. Note that we refer to a MC cycle as the number of moves needed to visit every crystal site once on average.

energy

set "energy",{"angle" | "cocc" | "cdis" | "disp"}

This command selects the type of MC energy to be used for the simulation. The MC sub level of DISCUS currently allows to introduce occupational correlations ("cocc"), displacement correlations ("cdis"), distortions ("disp") and angles ("angle"). The energy for the correlation modes is based on an Ising model whereas the distortions are expressed using Hookes law. The angular distortions are expressed as the square of the deviation from the target value.

feed

set "feed",<number>

The interaction values for the given correlation structure are determined by a feedback process. This command sets the number of moves between each feedback run. This value is also used for the interval of screen outputs. Note that the energy mode "disp" does not use the feedback mechanism and the value <number> given here only determines the output interval.

mode

set "mode",{ "shi[ft]" | "swc[hem]" | "swd[isp]" | "ext[ernal]" },
**   [{ "a[ll]" | "l[ocal]" | "sl[ocal]" | "si[te]"   }]**

This command determines the type of MC move used for modifying the crystal. The settings are explained in the RMC section under -> set mode.

neig

```
set "neig","add"
set "neig","ang",<iv1> [,<iv2>,..]
set "neig","dis",<u>,<v>,<w>,<fsig> [, <wsig>] [, {"sym" | "nosym"}]
set "neig","dir",<x1>,<y1>,<z1> [,<x2>,<y2>,<z2>]
set "neig","rese"
set "neig","vec",<iv1> [,<iv2>,..]
```

This command is used to define the neighbors used to determine the correlations. The sub-command 'dis' uses the distance as a criterion whereas 'vec' uses defined interaction vectors (-> set vec). The subcommand 'ang' uses an angle at a site, defined by (-> set ang). The subcommands 'add' and 'rese' allow the storage and reset of the list of definitions. The details of these commands are given in the online help for the same command 'set neig' in the CHEM sub level.

mole

```
set "mole",<typ1>,<typ2>
```

This command allows the user to select the molecule types used for the MC simulation. Not that different MC energy definitions have different restrictions on the selection of the molecule types.

move

```
set "move",{ "all" | <name> | <number> },<sx>,<sy>,<sz>
```

Sets an user defined sigma for the generates moves. The created shifts are Gaussian distributed. The default is a sigma of 0.2 unit cell. Note that the values of $<sx>$, $<sy>$ and $<sz>$ are given in unit cell units to speed up the MC runs. In case molecules are selected $<number>$ specifies the molecule type, otherwise atoms are addressed by their $<number>$ or $<name>$.

range

target

```
set "target",<ic>,<value>
set "target",<ic>,{<number> | <name> | "all"},
                  {<number> | <name> | "all"}, <dist>
```

The first command sets the desired correlation value $<value>$ for modes "cocc" and "cdis" for the neighbor definition $<ic>$.
The second version of the command sets the desired distance between the given atom types to $<dist>$ A. This setting is used for mode "disp". If $<disp>$ is set to zero the corresponding atom pair is not used in the MC simulation.

temp

`set "temp",<kt>`

This command sets the temperature <kt> used for the MC simulation. In order to be able to use the feedback procedure to determine the interaction energies the value of <kt> should be set to 1.0 (default). As a result the interaction energies will be in units of kT. If no feedback is used, <kT> can be set to the desired simulation temperature.

vec

`set "vec",<iv>,<is1>,<is2>,<dx>,<dy>,<dz>`

This command is used to define correlation vectors that might be used to calculate correlations within the crystal (-> 'set corr'). The complete description is given in the CHEM section of the online help (-> set vec).

show

`show`

This command lists all current settings for the MC simulation part of DISCUS on the screen.

A.35   mmc

`mmc`

Enter "Multi Energy Monte Carlo" section of DISCUS. This sub level allows to create disordered structures with given correlations and/or displacements using Monte-Carlo simulations. Several different "energies" can be minimized simultaneously.
See help entry 'commands' for a complete list of commands available in this sub level.
NOTE: The MMC and CHEM sub level share some variables which define neighbors etc. and settings made in this sub level might be altered when using the other sub level. It is always save to repeat settings when entering this sub level if both levels are used.
The mmc menu modifies the crystal according to the moves, ==> 'set move' that you define. Only those atoms will take part of the 'moves' and be included in the energy calculations, whose properties fulfill the global property selection rules ==> 'property'.

commands

Valid commands at this sub level are:

```
@        ! Executes a macro (see main help)
=        ! Algebra (see main help)
apply_symmetry ! Use the space group symmetry to generate vectors
continue! Continue a stopped macro (see main help level)
echo     ! Echos a string, just for interactive check
eval     ! Evaluates an expression (see main help)
exit     ! Terminates the MMC sub level, returns to the main DISCUS level.
help     ! Gives on-line help for 'mmc' (see main help)
```

```
rese     ! Reset the MMC to startup values
run      ! Starts MMC simulation
save     ! Saves structure (see main help -> save)
set      ! Sets most MMC parameters
show     ! Shows current settings
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
wait     ! Waits for user input (see main help)
```

apply_symmetry

```
apply_symmetry [<output_file>]
```

This command applies the space group symmetry to generate the symmetrically equivalent vectors. You must have defined one or several vectors with the ==> 'set vec' command and grouped these into neighborhoods via ==> 'set neigh'.

The general concept assumes that you have defined the vector correlations for the position x,y,z within the asymmetric unit and would like DISCUS to generate the corresponding vector correlations for the other atom positions as well.

Example is an atom is on a general position you could specify: set vec, 1, 1, 2, 0,0,0 set neig, vec, 1 apply_symmetry This would generate the corresponding vector correlations for the other atoms in the unit cell that are symmetrically equivalent to xyz.

If an atom is on a special position, like a mirror plane, the apply_symmetry command will generate the mirror image of the vector.

If the optional <output_file> is given, DISCUS will write the original "set vec, ..." and the generated ones into this file.

reset

```
reset
```

Reset all mmc definitions back to the default values at system start.

run

```
run
```

This command starts the MMC simulation for the given parameters and number of MMC moves.

set

```
set <subcommand>
```

This command allows to define the wanted correlations and most other parameters for the MMC simulation. The help entry 'commands' contains a list of valid subcommands for the 'set' command. Further help can be found for the following subcommands:

commands

```
allowed : Define which atom types are part of a neighborhood
angle   : Definition of interaction angles
con     : Definitions of interaction connectivities
cyc     : Setting of maximum number of MMC moves
envir   : Definitions of interaction environments
feed    : Setting of display/feedback interval
finish  : Convergence criteria to finish MMC
fixed   : Setting of atom ranges that are fixed
limit   : Setting of a limited atom range
mode    : Setting of MMC mode (crystal modification)
move    : Setting size of generated SHIFT moves
neig    : Setting of the neighbor definitions
output  : Setting of output during mmc run
range   : Setting of the neighboring ranges
style   : Set the style to atoms or molecules
target  : Setting of correlations values to be achieved
temp    : Setting of MMC simulation temperature
valid   ! Sets maximum number of non_valid cycles
vec     : Definition of interaction vectors
```

allowed

```
set "allowed", <is1> [,<is2]...]
```

Usually mmc will automatically determine, which atom types play a role in the Monte-Carlo process. To do so mmc analyzes the ==> 'set target' command to find out which atoms are involved.

In some cases, you might have defined a target simply as "set target, 1,repulsive,void,void, ..." Here the purpose is to define a repulsive interaction between voids, while all other atoms do not play a role. MMC needs to know, however, whether it may shift/switch chemistry of voids with other atoms through the "set mode" definitions.

To allow atoms to take part in the "set mode" modifications of the crystal, although they do not influence the energy of the crystal, you need to list these atoms on this "set allowed" command.

angle

```
set "ang",<iv>,<is1>,<is2>,<dx1>,<dy1>,<dz1>,<is3>,<dx2>,<dy2>,<dz2>
```

This command is used to define correlation angles that might be used to calculate correlations within the crystal (-> 'set target'). The complete description is given in the CHEM section of the online help (-> set ang).

con

```
set "con",<iv>, {<is1>| <at_name>}, {<def.no>| <def.name>}
```

This commands is used to define correlations between atoms through their connectivity list. See main menu 'connectivity' for further details. The second and third parameter allow alternative input. <iv> is simply the number of the connectivity definition within mmc. One or several connectivity definitions can be grouped to a neighborhood.

<is1> is the atom type of the central atom around which neighbors shall be searched via the connectivity list. <at_name> is the name of the central atom type around which neighbors hall be searched via the connectivity list.

<def.no.> is the number of the definition that has been added/set within the connectivity menu. <def.name> is the name of the definition that has been added/set within the connectivity menu.

cyc

```
set "cyc",<number>
```

This command sets the number of MMC moves (i.e. visited sites) to be executed by the 'run' command. The number <number> actually defines the number of 'tried' MMC moves. Note that we refer to a MMC cycle as the number of moves needed to visit every crystal site once on average.

disallowed

```
set "disallowed", <is1> [,<is2]...]
```

Usually mmc will automatically determine, which atom types play a role in the Monte-Carlo process. To do so mmc analyzes the ==> 'set target' command to find out which atoms are involved.

Sometimes, you might want to fix an atom to a place within the crystal structure. At the same time, though you might want the correlations between this atom and other atoms to be taken into account. This is needed, for example if you want to sort atoms of type A into the center of a sphere, while keeping atoms of type B on the outside. In this case a dummy atom of type C should be fixed at the center, and negative chemical short range order be realized between this C atom and atoms of type A.

To allow an atom type to participate in the moves defined by ==> set move, use the set allowed command.

As the ==> 'set target' command allows all atoms that it finds to participate in a move, the 'set disallowed' command must be place after the last 'set target' instruction.

environment

```
set "env",
```

feed

```
set "feed",<number>
```

The interaction values for the given correlation structure are determined by a feedback process. This command sets the number of moves between each feedback run. This value is also used for the interval of screen outputs. Note that the energy mode "disp" does not use the feedback mechanism and the value <number> given here only determines the output interval.

finish

```
set "finish" [  {stop:cycles | stop:converge}]
              [, feed:<nfeed>] [, diff:<maxdiff>] [, rdiff:<maxrdiff]
              [, change:<maxchange>] [, aver:<maxaver>]
              [, {log:none | log:screen}]
```

Set criteria that MMC uses to finish the cycles.

Default or "stop:cycles" disables the convergence criteria. MMC will run the full amount of
==> set cycles.

With "stop:converge", MMC will apply the two convergence criteria and stop running once both are met.

diff:<maxdiff> If the largest deviation: |(target-achieved)| is less than <maxdiff> MMC has converged

rdiff:<maxrdiff> If the largest deviation: |(target-achieved)/target| is less than <maxrdiff> MMC has converged

change:<maxchange> If the largest change of |(target-achieved)/target| observed over the last <nfeed> feedbacks is less than <maxchange>, MMC is considered to be stagnant and will be stopped.

aver:<maxaver> If the average changes over the last nfeed cycles are less than <maxaver>, MMC is considered to be stagnant.

feed:<nfeed> Number of feedback that are considered for the "change" criterion.

log:screen or log:none Turns a display of the convergence status on/off. At "screen", a short report on the current values of the convergence criteria is displayed on the screen. This helps to find good valus for the parameters.

limited

```
set "limited", "OFF"
set "limited", "atom", <nmin>, <nmax>
```

Allows to limit the atom selection the interval [<nmin>:<nmax>]. Atoms outside this interval are ignored and thus do not take part in the Monte Carlo process.

The "set limit, OFF" command switched the limitation off.

mode

```
set "mode", <probability>, <modification_type>,
            [ <site_selection> [ , <atom_list>] ]
```

This command determines the type of MMC move used for modifying the crystal. You can simultaneously use any of the moves. The user defined values of the probabilities <prob> are internally scaled to add up to one and give the probability for each type of move. The settings are explained in the RMC section under -> set mode.

The possible modes are:

```
"shift"    : The selected atoms are displaced by a random move. The
             maximum size of a move can be specified via "set move .."
             command.
             Example:
```

```
set mode, 1.0, shift, Cu, Au
set mode, 1.0, shift, all     ! shift all atom types

"swchem"   : Here two atoms are selected at random and their chemistry
             is switched (if the atoms are selected via "sel" command).
             If your structure consists of only ONE atom type this mode
             will just waste computer time :-)
             Example:
set mode, 1.0, swchem, Cu, Au

"swneig"   : Two central atoms are selected. During the modification
             two of their neighbors are exchanged.
             Example:
set mode, 1.0, swneig, Cu, Au
set mode, 1.0, swneig, Cu, Au, site:[5, 7]
             The optional "site:" parameter allows to select further
             sites in the unit cell that might not be included in the
             list of neighbors around the central atoms.
"swdisp"   : Here the displacement from the average structure of two
             selected atoms of the SAME type are switched. The mode
             can only properly work, if there are initial displacements
             generated e.g. by the "therm" command.
             Example:
set mode, 1.0, swdisp, Cu, Au

"external" : The modification of the structure for each RMC move is done
             using an user supplied external subroutine. Check the
             source file 'extrmc.f90' for details how to supply such a
             subroutine.
```

The third optional parameter specifies if the pair of the atoms to be selected for the SWCHEM and SWDISP mode should be restricted in any way. The following settings are valid, the setting "all" is the default, which will be taken if no second parameter is entered.

```
"all"      : Any atom within the crystal can be chosen as second atom.
"local"    : Only atoms within +-1 unit cell from the first selected
             atom will be chosen.
"slocal"   : Same as "local" only the second atom must be on the same
             site within the unit cell as the first one.
"site"     : Atom on the same site in any unit cell are allowed as
             second atom.
```

If the atom list is given, the fourth parameter that specifies the site must also be present. The atom list allows you to "move" or "exchange" atoms, even if they are not involved in any of the target energies.

move

```
set "move",{ "all" | <name> | <number> },<sx>,<sy>,<sz>
set "move",{ "all" | <name> | <number> },<u>,<v>,<w>,  <sig>
```

Sets an user defined sigma for the generates moves. The created shifts are Gaussian distributed. The default is a sigma of 0.2 unit cell. Note that the values of $<sx>$, $<sy>$ and $<sz>$ are given in unit cell units to speed up the RMC runs. In case molecules are selected $<number>$ specifies the molecule type, otherwise atoms are addressed by their $<number>$ or $<name>$.
If the second form is used, the parameters $<u>$,$<v>$,$<w>$ define a vector. The movement of the atom is restricted along this vector. The last parameter is a multiplicative sigma along this direction.

---

neig

```
set "neig","rese"
set "neig","add"
set "neig","ang",<iv1>,{ [<iv2>,..] | <iv2>, "range" }
set "neig","env",<iv1>,{ [<iv2>,..] | <iv2>, "range" }
set "neig","vec",<iv1>,{ [<iv2>,..] | <iv2>, "range" }
    [,"number:"<value>]
    [,"number:next"]
```

This command is used to define the neighbors used to determine the correlations. The subcommand 'env' uses defined interaction environments (-> set env). The subcommand 'vec' uses defined interaction vectors (-> set vec). The subcommand 'ang' uses an angle at a site, defined by (-> set ang). Subcommand "reset" is used to set the number of neighborhoods back to zero. The optional parameter "number:" replaces the "set neig, add" command that was required in between two "set neig,..." commands. You can either specify a specific numerical value: set neig, number:1, vec, 1, 2 or increment the neighborhood number with set neig, number:next, vec, 1, 2

The subcommand 'add' is depreciated and it is reccommended to use the optional parameter "number:" instead. The details of these commands are also given in the online help for the same command 'set neig' in the CHEM sub level.

If the last parameter is "range", and exactly two arguments <iv1>,<iv2> are given, then the contiguous range of correlations from <iv1> to <iv2> is used.

Examples set neig, reset ! Back to initial state set neig, number:1, vec 1, 2 ! define neighborhood 1 with vectors. set neig, number:next, vec 3, 4 ! define the next neighborhood set neig, number:3, vec, 1,2,3,4,5,6 set neig, number:4, vec, 1,6,range

Old examples: set neigh, reset ! Back to initial state set neig, vec, 1,2 ! use the listed vectors set neig, add ! Increment number of neighborhoods set neig, vec, 1,2,8,7 set neig, add ! Increment number of neighborhoods set neig, vec, 1,2,3,4,5,6 set neig, add ! Increment number of neighborhoods set neig, vec, 1,6,range ! Use vectors 1 to 6 ! ! Do not use "set neig, add" after the ! ! last neighborhood

move

```
set "move",{ "all" | <name> | <number> },<sx>,<sy>,<sz>
```

Sets an user defined sigma for the generated moves. The created shifts are Gaussian distributed. The default is a sigma of 0.2 unit cell. Note that the values of <sx>, <sy> and <sz> are given in unit cell units to speed up the MMC runs. In case molecules are selected <number> specifies the molecule type, otherwise atoms are addressed by their <number> or <name>.

output

```
output "feed:on"  , "final:on"
output "feed:off" , "final:off"
```

Sets the amount of output written to screen during an MMC run. Default is to write output during the run and a summary after the run is finished.
set output, feed:off Turn output off during the run
set output, final:off Turn output off at the end of the run

style

```
set style, atom
set style, molecule
```

Sets the type of objects that are moved/modified in the MMC cycles. With "atom" individual atoms are modified, With "molecule" entire molecules are modified.

range

```
set "range"
```

target

```
set "target",<ic>,"corr",     {<number> | <name> | "all"},
                              {<number> | <name> | "all"},
                              <correlation>, <energy>, {"CORR" | "ENER"}
set "target",<ic>,"corr",     ({<number> | <name> | "all"}[, {<number> | <name> | "all"}...]),
                              ({<number> | <name> | "all"}[, {<number> | <name> | "all"}...]),
                              <correlation>, <energy>, {"CORR" | "ENER"}
set "target",<ic>,"unid",     {<number> | <name> | "all"},
                              {<number> | <name> | "all"},
                              <correlation>, <energy>, {"CORR" | "ENER"}
set "target",<ic>,"unid",     ({<number> | <name> | "all"}[, {<number> | <name> | "all"}...]),
                              ({<number> | <name> | "all"}[, {<number> | <name> | "all"}...]),
                              <correlation>, <energy>, {"CORR" | "ENER"}
set "target", <ic>,"cn",      ({<number> | <name> | "all"}[, {<number> | <name> | "all"}...]),
                              ({<number> | <name> | "all"}[, {<number> | <name> | "all"}...]),
                              <coordination_number>, <energy>
set "target",<ic>,"cd",       {<number> | <name> | "all"},
                              {<number> | <name> | "all"},
                              {"dir:["<u>,<v>,<w>"]" | "dir:["<u>,<v>,<w>,<u2>,<v2>,<w2>"]"},
                              <correlation>, <energy>, {"CORR" | "ENER"}

set "target",<ic>,"angle",    {<number> | <name> | "all"},    ! Central atom
                              {<number> | <name> | "all"},    ! neighbors
                              {<number> | <name> | "all"},    ! neighbors
                              <angle>,        <energy>

set "target",<ic>,"spring",   {<number> | <name> | "all"},
                              {<number> | <name> | "all"},
                              <distance>,     <energy>
set "target",<ic>,"lennard",  {<number> | <name> | "all"},
                              {<number> | <name> | "all"},
                              <distance>,     <energy> [, <pr>, <pa>]
set "target",<ic>,"bucking",  {<number> | <name> | "all"},
                              {<number> | <name> | "all"},
                              <a>, <rho>, <b>
set "target",<ic>,"repulsive",{<number> | <name> | "all"},
                              {<number> | <name> | "all"},
                              <energy>,     <scale> [, <rmin> [, <pa>] ]
```

The target command sets the desired correlation values for each of the defined interaction neighborhoods <ic> (-> set neigh).
Depending on the neighborhood type you must specify exactly one or several target energy types for each <ic>. The neighborhoods "vector", "cor", and "angle" require one energy type,

while the neighborhood "environment" may take several different energies. For each interaction <ic> you may specify several targets for different atom combinations.

set "target",<ic>,"corr", {<number> | <name> | "all"}, {<number> | <name> | "all"}, <correlation>, <energy>, {"CORR" | "ENER"} set "target",<ic>,"corr", ({<number> | <name> | "all"} [, {<number> | <name> | "all"}...]), ({<number> | <name> | "all"} [, {<number> | <name> | "all"}...]),

A chemical short range order correlation is defined between the two atom types. The last keyword defines how the target shall be achieved. With "CORR", you want to achieve the target <correlation>. DISCUS uses the feedback loop to adjust the value of the Ising Energy in order to achieve the desired correlation. With "ENER", you fix the value of the Ising energy to the value of <energy>. For negative <energy> values like atoms will tend to be neighbors, while a positive <energy> favors opposite atoms next to each other. If you use several targets at the same time, it is necessary to use the "ENER" keyword. Otherwise, the feedback mechanism will increase the absolute value of the Ising energy at each feedback until the target <correlation> is achieved. This target will therefore overrule the other targets that all have fixed energy values. If you specify a positive correlation, like tom types will tend to cluster together, while for a negative correlation you will get mostly opposite atom types as first neighbors.

The second command form allows you to define correlations between groups of atom types. Place each of the two sets of atom types in round brackets. In this case a positive correlation will lead to clusters that consist of any atom of the first group to be neighbor to another atom of the first group and likewise at a different place in the crystal for the second group. A negative correlation will give you a crystal in which an atom of the first group has an atom of the second group as neighbor.

set "target",<ic>,"unid", {<number> | <name> | "all"}, {<number> | <name> | "all"}, <correlation>, <energy>, {"CORR" | "ENER"} set "target",<ic>,"unid", ({<number> | <name> | "all"} [, {<number> | <name> | "all"}...]), ({<number> | <name> | "all"} [, {<number> | <name> | "all"}...]),

This energy form is essentially identical to the chemical correlation energy. The difference is that this energy form serves to distinguish directional pairs. For the target: set target, 1, unid, A, B, -1.0, 0.00, CORR the program will sort the structure favoring A => B pairs over all other pairs (A => A; B =>B; B =>A). This energy works best for "vector" neighborhoods, and you should only use a given vector and not the opposite vector.

```
set "target", <ic>,"cn",     ({<number> | <name> | "all"}[, {<number> | <name> | "all"}...]),
                             ({<number> | <name> | "all"}[, {<number> | <name> | "all"}...]),
                              <coordination_number>, <energy>
```

This energy form drives the structure into neighborhoods with user defined coordination numbers. The first atom is considered the central atom, the second atom is considered the preferred atom type within the neighborhood. The program sorts by switching atoms between the neighborhoods of two central atoms.

set "target",<ic>,"cd", {<number> | <name> | "all"}, {<number> | <name> | "all"}, {"dir:["<u>,<v>,<w>"]" | "dir:["<u>,<v>,<w>,<u2>,<v2>,<w2>"]"}, <correlation>, <energy>, {"CORR" | "ENER"}

The intended correlation is a correlation between the displacements of the atoms in the neighborhood. A positive correlation means that the displacements from the average position shall be in the same direction, while a negative correlation means a preference for opposing displacements. Equal displacements are characterized by a positive value of the scalar product of the

two displacement vectors that are given by "dir:...". Naturally this energy works best in a local environment that is not centrosymmetric.

set "target",<ic>,"lennard", {<number> | <name> | "all"}, {<number> | <name> | "all"}, <distance>, <energy> [, <pr>, <pa>]

A Lennard Jones type potential energy is calculated for the two atom types. The energy is calculated as: $E = A/<r>**<pa> - <B>/<r>**<pa>$ Here <r> is the interatomic distance and A and B are calculated to give the minimum of the Lennard-Jones potential at distance <distance> with a potential depth of -1.*|<energy>|

set "target",<ic>,"repulsive",{<number> | <name> | "all"}, {<number> | <name> | "all"}, [<energy>[,<scale> [, <rmin> [, <pr>]]]]

A purely repulsive energy is calculated between the two atom types. The energy is calculated as: $-1.*|<energy>| + ((<distance> - <rmin>)/<scale>)**<pr>$ Thus <energy> is the energy between the two atoms at infinite distance. As mmc compares the energy differences between two atom configurations, the value of this energy is not really relevant.

For interatomic distances shorter than <rmin> the program calculates: $-1.*|<energy>| + ((0.000001)**<pr>$

For a steep descent, choose a high value of the power <pr>. With a high value of <pr>, the potential levels off at shorter distances and energy differences become less relevant. Thus the atoms will be pushed apart more strongly at shorter distances, yet at longer distances the influence of this potential is less critical.

To push atoms far apart over the whole range of distances, choose a large value of <scale>. <energy> defaults to zero <scale> defaults to 1.0 <rmin> defaults to zero <pr> defaults to 1.0

set "target",<ic>,"angle", {<number> | <name> | "all"}, ! Central atom {<number> | <name> | "all"}, ! neighbors {<number> | <name> | "all"}, ! neighbors <angle>, <energy>

Define an angular potential. The first atom type is the central atom, at whose position the bond angle is located. The potential is <energ>*(alpha-<angle>)**2. Compared to the Lennard-Jones potential, the energy values should usually be smaller by a factor 10 or so.

temp

```
set "temp",<kt>
```

This command sets the temperature <kt> used for the MMC simulation. In order to be able to use the feedback procedure to determine the interaction energies the value of <kt> should be set to 1.0 (default). As a result the interaction energies will be in units of kT. If no feedback is used, <kT> can be set to the desired simulation temperature.

valid

```
valid <invalid_moves>
```

MMC will pick an atom or an atom pair at random. If a pair of atoms has the same chemistry, a "switch chemistry" move will not change anything. Such a pair is considered an invalid trial. As the structure converges into a highly ordered state, the probability of picking an invalid pair increases. In order to prevent an infinite loop, DISCUS terminates mmc once the

number of invalid tried configurations exceeds the number of <invalid_moves>. Default for <invalid_moves> is 1000. For large structures it might be helpful to increase this number, for example to a value around the number of atoms.

vec

```
set "vec",<iv>,<is1>,<is2>,<dx>,<dy>,<dz>
```

This command is used to define correlation vectors that might be used to calculate correlations within the crystal (-> 'set neig'). The first parameter <iv> is the number of the vector to be defined. The variables <is1> and <is2> give the number of the crystal site at each end of the vector. The values of <dx>,<dy> and <dz> define the vector in unit cells. Correlation vectors within one unit cell have <dx>,<dy> and <dz> set to zero. If you want to define correlation vectors crossing the unit cell boundaries set <dx>,<dy> and <dy> accordingly.
As a side effect of the UNIDIRECTIONAL correlations, you must make sure that vectors that start on different sites are in separate targets, even though they might be symmetrically related. (To be fixed).

show

```
show
```

This command lists all current settings for the MMC simulation part of DISCUS on the screen.

A.36 molecularize

```
molecularize "conn" [, start:<from>] [, finish:<to>]
                    [, conn:<list>] [, central:<list>]
                    [, exclude:<list>]
                    [, moletype:<type>] [, biso:<value>]
                    [, corrlin:<value>] [, corrquad:<value>]
molecularize "range" [, start:<from>] [, finish:<to>]
                    [, moletype:<type>] [, biso:<value>]
                    [, corrlin:<value>] [, corrquad:<value>]
molecularize "conn", <central> [, <exclude> ...]
molecularize "range", <from>, <to>, <molecule_type> , <molecule_Biso>
```

This command allows to group a set of atoms into a molecule.
molecularize conn

'molecularize conn' and its optional parameters will place atoms and their connectivities into a molecule. At least one suitable connectivity must have been created within the ==> 'conn' menu.
molecularize "range"

'molecularize range" and its optional parameters will place the consecutive range of atoms from start to finish into a molecule. DISCUS will not check how close these atoms are.
optional parameters
start:<from>

start:all Defines the first atom number <from> to be at the center of a molecule or sets that "all" atoms are checked. Defaults to "all"

finish:<**to**>

finish:all Defines the last atom number <to> to be at the center of a molecule. If <to> is set to "last", the search runs to the last atom in the crystal, i.e. up to n[1]. If <to> is set to "all", all atoms from 1 to n[1] are examined. Defaults to "last"

conn:<**list**>

Defines which of the connectivities of the current central atom are used to build the molecule. The list can take the forms: "-1" to include all connectivities a single numerical value such as 1 a list of comma separated numerical values enclosed in [...] Defaults to "-1", which includes all connectivities.

central:<**list**>

Defines which central atom types are used. The list can take the forms: "-1" or "all" to include all atom types a single numerical value such as 1 or 4 a list of comma separated numerical values and/or atom type names enclosed in [...] Defaults to "-1", which includes all atom types.

moletype:<**value**>

Defines the molecule type that will be used for the new molecules. Please make sure that this will be a unique number

biso:<**value**>

Defines a molecular isotropic displacement parameter, used in the Fourier and powder menues

corrlin:<**value**>

Defines a correlation term that is used to make PDF maxima at short distances narrower than those at larger distance. corrlin defines a relationship proportional to $1/r$.

corrquad:<**value**>

Defines a correlation term that is used to make PDF maxima at short distances narrower than those at larger distance. corrquad defines a relationship proportional to $1/r**2$.

older versions, discouraged

The third version requires that for atom <central> and all other atoms in the molecule a suitable connectivity list has been prepared, at the connectivity menu, see ==> 'conn' for further details. You can list one or more excluded atoms. They and the connectivity behind these atoms are excluded from the molecule. This in turn allows you to group a sub section of a full molecule. Internally this algorithm is used within the symmetry menu to rotate a partial molecule.

The last version places the consecutive range of atoms into a molecule of type <molecule_type>. DISCUS will not check how close these atoms are.

A.37   output

`output`

Switches to the output level of discus. All Fourier output is written at this level. Several graphic formats are available.
Further help topics are:

commands

```
@       ! Execute a macro file (see main help)
=       ! Assigns the value to a variable (see main help)
continue! Continue a stopped macro (see main help level)
echo    ! Echo a string (see main help)
eval    ! Evaluates an expression for interactive check (see main help)
exit    ! Returns to the main discus level
form    ! Defines the output format to be used
help    ! Gives help for the output level (see main help)
input   ! Reads an old data file
outfile ! Sets name of output file
range   ! Limit the range of a 3DPDF, most useful in combination with
        ! format "xplor", and "vesta"
reset   ! Reset and clear all definitions
run     ! Starts writing of output
show    ! Shows the current settings
thre    ! Sets the thresholds used for bitmap output
valu    ! Sets the value that is written to file
stop    ! Stops execution of a macro (see main help level)
system  ! Executes operating system command (see main help)
wait    ! Waits for user input (see main help)
```

format

```
form {"gnu" | "pgm" | "ppm" |
     "powder" [,{"dst"|"tth"|"stl"|"q"} [,<min>,<max>,<step>] ]
     "pdf" [,"r" [,<min>,<max>,<step>] ]
   | "post" | "stan" | "nexus" |
     "vtk" | "mrc" | "vesta" | "xplor"
   | {"shel"|"hklf4"@"list5"} [,"maxval:"{"auto"|"bragg"|<value>"]" }
   | "hdf5" [,"maxval:"{"auto"|"data"|<value>"]" }
```

Defines the output format that will be used. Valid parameters are:

`"gnu"`

Sets the type of the output file to gnuplot. The intensities are written in blocks of intensities along the abscissa, with an empty line separating each block. Each line of each block contains: index along abscissa, index along ordinate, intensity, third index.

`"hdf5" [,maxval:"auto"]`
`"hdf5" [,maxval:"data"]`
`"hdf5" [,maxval:<value>]`

Sets the output to HDF5 file format. This applies to single crystal data only, not to powder diffraction pattern.
The file format is compatible to the Yell program by Arkadiy Simonov. See the manual for details on the file format.
By default the data are scaled by a multiplier 1.0 to give the maximum value in the output file as in the data. The same is achieved by maxval:"data". Alternatively a scale factor can be applied such that the maximum value in the output file is equal to <value>.

`"list5"`

Sets the output type to a SHELXL list 5. Each line of the output contains " h k l F F phase" in the format (3I4, 2F10.2,F2.2). The hkl are converted by an "int" command to integers. The user has to make sure that the increments used in the Fourier resulted in integer spacing. Independent of the output value set by the 'value' command, DISCUS always writes the amplitude and the phase angle in degrees. DISCUS calculates the value of F(000) and scales the data such that F(000) is less than 10**8 to fit into the format.

"mrc"

Set the output file format to the MRC file format. Applies to 2D and 3D data only. Images can be visualized for example with Chimera.

"nexus"

Sets the output to NeXuS file format.

"pgm"

Sets the type of the output to portable gray map. A bitmap of the intensity is written in portable gray map style. All values below the minimum threshold are set to this value, all values above the maximum threshold are set to this maximum value.

"ppm"

Sets the type of the output to portable any-map. A bitmap of the intensity is written in portable any-map style. The color scheme is read from file "color.map". All values below the minimum threshold are set to this value, all values above the maximum threshold are set to this maximum value.

"powder" [,{"dst"|"tth"|"stl"|"q"} [,<min>,<max>,<step>]]

Sets the output format to powder. You must have calculated a powder diffraction pattern using the ==> 'powder' menu of DISCUS. DISCUS writes a two column output, with the intensity in the second column. The value in the first column is set by the optional second parameter and may be: "dst" : dstar = 1./d(hkl) = 2* sin(theta)/lambda "tth" : 2-Theta "stl" : sin(theta)/lambda "q" : Q = 2*pi* dstar = 4*pi*sin(theta)/lambda
For the two types of axes in the ==> 'powder' menu, "Q" and "TTH", you need to remember to specify limits for the axis. If you intend to write the output in the other format, the default is to convert these limits to the intended output axis.
If the values [,<min>,<max>,<step>] are specified, they will overwrite the limits set in the powder menu. This especially helpful if you want to calculate on a Q-axis but write a file on a 2Theta axis or vice versa. Likewise you might want to limit the powder output compared to the calculation or write the output with a different step size.
If DISCUS is operated within the DISCUS_SUITE, a file name that starts with "kuplot" is copied directly into the KUPLOT data array, and NOT written to disk.

"pdf" [,"r" [,<min>,<max>,<step>]]

Sets the output format to a Pair distribution function PDF.
If the values [,<min>,<max>,<step>] are specified, they will define the PDF range to be written.

`"post"`

Sets the type of the output to color postscript. A bitmap of the intensity is written in color postscript. The color scheme is read from file "color.map". All values below the minimum threshold are set to this value, all values above the maximum threshold are set to this maximum value.

`"shel"`
`"hklf4"`

Both commands sets the output format to SHELX type hklf4 data. The data are written in format (3I4,2F8.2). The hkl are converted by an "int" command to integers. The user has to make sure that the increments used in the Fourier resulted in integer spacing. Use the ==> 'value' "intensity" or "amplitude" for the output. DISCUS calculates the value of I(000) or F(000) respectively and scales the data such that I(000) is less than 10**6 to fit into the format. The sigma is just the square root of the final intensity after scaling.
By default or "maxval:data" the intensities are scaled such that the intensity of Bragg reflection 000 fits into the output format. With "maxval:bragg" the data are scaled such that the most intense Bragg intensity fits into the format. Be aware that this might cause a different scale factor if the range of hkl is changed. Otherwise maxval:<value> will scale the Bragg intensities by user suplied factor, even if this might cause an overflow.

`"stan"`

Sets the output to standard file format, which is used by KUPLOT. The output file contains two initial lines and then a block for each data line separated by an empty line. The first line contains the number of data points along the abscissa and the ordinate, the second line the minimum and maximum value of the indices in reciprocal space along the abscissa and ordinate respectively.

`"vesta"`
`[, {"patt:none" | "patt:unit" | "patt:full" ]`
`[, {"sel:all" | "sel:none" | "sel:name" | "sel:[name1,name2 {, ...}]"]`
`[, {"des:all" | "des:none" | "des:name" | "des:[name1,name2 {, ...}]"]`

Writes an ASCII formatted 3D-map of the intensity or 3DPDF in the GSAS grd file format. This map can be viewed with VESTA.
Optionally a 3D-PDF map can be overlayed by a sketch of the Patterson structure, which is calculated from the average structure. Choices are a "full" Patterson overlay throughout the entire 3D-PDF, or a restriction to a single "unit" cell in the range [-0.5, 0.5].
The "sel", "des" options allow to select or deselect atom types from which the Patterson overlay is constructed. "VOID" atoms are deselected by default. As a Patterson overlay will easily become very crowded a restriction to few atom types might help for clarity. For an X-ray 3D-PDF for example you might consider to deselect Hydrogen.
Default values are: "patt:none"; "sel:all"; "des:none"

`"xplor"`

Writes an ASCII formatted 3D-map of the intensity or 3DPDF in the X-plor file format. This map can be viewed with several crystallographic viewers, for example VESTA.

`"vtk"`

Sets the output to vtk.

---

input

```
input
```

Reads an old data file

outfile

```
outfile <filename>
```

Sets the name of the output file, to which the intensities are written.
Within the DISCUS_SUITE the output files can be written directly into KUPLOT. Simply start
the file name with the fixed string 'kuplot'. The data set number in kuplot will be incremented
automatically.

range

```
range "full"
range "center:["<cx>,<cy>,<cz> "]", "pixel:["<px>,<py>,<pz>"]"
range "center:middle",            "pixel:["<px>,<py>,<pz>"]"
range "quad:lll"
```

This command allows you to limit the output of 3D pixel maps.
With "full" as first and single parameter, the full map is written.
With "center:..." the central position is defined, which can be either the "middle" of the complete
calculated volume or a suser defined triplet of pixel values.
With "pixel:.." you set the width of the map to be written. Specify as "pixel:[51, 31, 61]" with the
numbers replaced by values of your choice.
With "quad:...." as single parameter, you choose a quadrant. The three letters must be any com-
bination of "r" and "l" for "right" and "left" quadrant along each axis.

reset

```
reset
```

Reset all output definitions back to the default values at system start.

run

```
run  {"trans:yes" | "trans:no"}
```

Starts writing the output file. Without this command the output file is not written!
The optional parameter "trans:" applies to a 3D Fourier volume or a 3D PDF only. If the optional
paramter is "trans:yes", a transformation menu is enterd. Within this menu the transformation
to a different coordinate system is specified. The commands at this menu level are:

Examples

All examples assume a 3D volume parallel a*; b*; c*. In all examples the new data set is transformed to an orientation with the z-axis parallel to [1,1,1] and the abscissa parallel to [1, -1, 0].

Sensible combinations of the optional parameters are: 1) Define an abscissa and a zone axis 2) Define the abscissa, ordinate and top axis 3) define the corners in space as: ll = lower_left = left_lower_bottom lr = lower_right = right_lower_bottom ul = upper_right = left_upper_bottom tl = top_left = left_lower_top

1) Define an abscissa and a zone axis set abs:[0.04, -0.04, 0.00], zone:[1,1,1]

The new zone axis is set to [111], the abscissa is in steps of [0.04, -0.04, 0.00]. The ordinate will be calculated as vector product zone * abs, steps along the abscissa are of equal length as along the abscissa. The number of data points is determined by transformation of the hkl limits in the original data set.

set abs:[0.04, -0.04, 0.00], zone:[1,1,1] set npoints:[251,251, 125]

Same as above. The number of data points is given by the user. Along the abscissa the maximum value will be [0.04, -0.04, 0.00]*(125), where 125 = (251-1)/2. Same for the other three axes. Datza points in the new data set outside the original data receive values of zero.

2) Define the abscissa, ordinate and top axis set abs:[0.04, -0.04, 0.00] set ord:[0.04, 0.04,-0.08] set top:[0.04, 0.04, 0.04]

The new axes are all explicitely given as increment vectors. hkl limits and number of data points are derived from the orginal data set.

set abs:[0.04, -0.04, 0.00] set ord:[0.04, 0.04,-0.08] set top:[0.04, 0.04, 0.04] set npoints:[251,251, 125]

Same as above. The number of data points is given by the user.

3) define the corners in space as: set ll:[-15.0, -3.0, 9.0] set lr:[-3.0, -15.0, 9.0] set ul:[-3.0, 9.0, -15.0] set tl:[-9.0, 3.0, 15.0] set npoints:[251,251, 25

Four new corners are defined and the number of data points along the abscissa, ordinate and top axis are provided.

set

```
set "abs:["<h>, <k>, <l>"]"
set "ord:["<h>, <k>, <l>"]"
set "zone:["<h>, <k>, <l>"]"
set "ll:["<h>, <k>, <l>"]"
set "lr:["<h>, <k>, <l>"]"
set "ul:["<h>, <k>, <l>"]"
set "tl:["<h>, <k>, <l>"]"
set "npoints:["<h>, <k>, <l>"]"
```

Specified details of the intended transformation

```
set "abs:["<h>, <k>, <l>"]"
```

Sets the new abscissa

```
set "ord:["<h>, <k>, <l>"]"
```

Sets the new ordinate

```
set "zone:["<h>, <k>, <l>"]"
```

Sets the new zone axis. The zone axis will be the "top" coordinate of the transformed data.

```
set "ll:["<h>, <k>, <l>"]"
set "lr:["<h>, <k>, <l>"]"
set "ul:["<h>, <k>, <l>"]"
set "tl:["<h>, <k>, <l>"]"
```

The combination of these four settings define the new 3D volume. The new axes are the vectors from the lower left "ll" corner to: abscissa :: lower right "lr" ordinate :: upper left "ul" top axis :: top left "tl"

```
set "npoints:["<nx>, <ny>, <nz>"]"
```

Set the number of data points in the new transformed grid.

reset

```
reset
```

Set all values back to program start.

run

Perform the actual transformation.

exit

```
exit
```

Go back to the output menu.

show

```
show
```

Shows the current settings

threshold

```
thresh { "high"|"low"|"sigma"|"zmin"|"zmax"} <value>
```

This command sets the threshold that is used when writing BITMAP output. All values less than the minimum threshold are set to zero, all values higher than the maximum threshold are set to the maximum threshold. The values in between are linearly scaled from zero to 255. Depending on the first parameter, the second parameter is interpreted in five different ways:

```
"high"  : Sets maximum threshold for BITMAP in percent of the maximum
          diffuse intensity.
"low"   : Sets minimum threshold for BITMAP in percent of the maximum
          diffuse intensity.
"sigma" : Sets threshold for BITMAP to average diffuse intensity
          +- <value> times standard deviation of diffuse intensity.
"zmax"  : Sets maximum threshold for BITMAP
"zmin"  : Sets minimum threshold for BITMAP
```

value

```
valu {  "int"  |  "amp"  |  "pha" [,"random"]  |  "real"  |  "imag" }

valu { "S(Q)"  | "F(Q)" | "I(Q)"}

valu { "f2aver"  | "faver2" }

valu { "f2averb"  | "faver2b" }

valu { "pdf" }

valu { "3DPDF" } {, "dsmax:"<value> | "qmax:"<value>
                    "hklmax:["<h>,<k>,<l>"]"
```

Sets what value is written to the output file. The keyword given in $<\ >$ corresponds to the value of the average structure factor $<F>$ whereas the plain word stands for the resulting value of the Fourier transform which can be F or F-$<F>$ depending on the selected mode. The command 'value int' will save the resulting intensity whereas 'value $<int>$' will give the intensity of the average structure. The allowed values are summed up in the following list:

```
"int" :    intensity I(hkl)
"amp" :    modulus of structure factor |F(hkl)|
"pha" :    phase angle in degrees
           If the optional second parameter is given as "random",
           the phases for integer HKL are written as calculated.
           All other phases are assigned a random number between
           -180 and +180 degrees.
"real":    real part of structure factor
"imag":    imaginary part of structure factor
"pdf"      Powder PDF
"3DPDF"    3DPDF == generalized Patterson function
           Exactly one optional parameter can be specified that
           sets a limit in reciprocal space. All intensities further
           outside are multiplied with zero.
           "dsmax"   : value is in 2*sin(Theta)/lambda
           "qmax"    : value is in 2PI*2*sin(Theta)/lambda
           "hklmax"  : value is length of reciprocal vector
           WARNING, as the intensity is multiplied with zero,
              all further output is truncated as well!
              To avoid, write 3D-PDF output last.
           Defaults to no limit to be applied.
"S(Q)":    Normalized total scattering function
"F(Q)":    Reduced normalized total scattering function Q[S(Q)-1]
"I(Q)":    Intensity divided by the number of atoms.
"f2aver": Squared atomic form factors , averaged over all atoms
"faver2": Average atomic form factor squared aver average
"f2averb": Squared atomic form factors , averaged over all atoms
           Multiplied by averaged DebyeWaller term
"faver2b": Average atomic form factor squared aver average
           Multiplied by averaged DebyeWaller term
```

The last seven values are available for powder output only and require a Q axis.
The file name is not adapted automatically!

A.38   patterson

patterson

Branches to the Patterson transform level of DISCUS The Patterson transform is calculated from two input files. The two files must contain the intensity, real or imaginary part, or amplitude in one of the following combinations:

```
file a          file b
------------------------
intensity
amplitude
real            imaginary
```

The user must specify the Fourier plane that had been calculated and the real space plane to be calculated.
Further help topics are:

commands

Valid commands at this level are:

```
@        ! Execute a macro file (see main help)
=        ! Assigns the value to a variable (see main help)
abs      ! Defines the coordinate of the abscissa
continue! Continue a stopped macro (see main help level)
echo     ! Echo a string (see main help)
eval     ! Evaluates an expression for interactive check (see main help)
exit     ! Terminates Fourier level
file     ! Defines the input file name(s)
form     ! Defines the input file format
help     ! Help to Fourier commands (see main help)
layer    ! Defines the layer in reciprocal space to be calculated
ll       ! Defines the lower left corner in reciprocal space
lr       ! Defines the lower right corner in reciprocal space
na       ! Defines the number of points along the abscissa
no       ! Defines the number of points along the ordinate
ord      ! Defines the coordinate of the ordinate
reset    ! Reset all Fourier definitions
rhoabs   ! Defines the coordinate of the real space abscissa
rholayer! Defines the layer in real space to be calculated
rholl    ! Defines the lower left corner in real space
rholr    ! Defines the lower right corner in real space
rhona    ! Defines the number of points along the real space abscissa
rhono    ! Defines the number of points along the real space ordinate
rhoord   ! Defines the coordinate of the real space ordinate
rhoul    ! Defines the upper left corner in real space
run      ! Starts the calculation
scale    ! Defines an overall scale factor
set      ! Sets various parameters
show     ! Shows the current settings for the inverse Fourier transform
stat     ! Shows statistics on the normalized structure factor
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
```

```
type    ! Defines the file type (intensity, amplitude ...)
ul      ! Defines the upper left corner in reciprocal space
wilson  ! Calculates Wilson statistics
wait    ! Waits for user input (see main help)
```

abs

```
abs <switch>
```

Defines which the component of the vector parallel to the abscissa is written to the output file.

file

```
file {"a"|"b"},<filename>
```

Defines the filenames needed for the patterson.
The Patterson transform is calculated from one or two input files. The two files must contain the intensity, real or imaginary part, or amplitude in one of the following combinations:

```
file a          file b
------------------------
intensity
amplitude
real            imaginary
```

If the input format is 'shelxl' then one input file is sufficient. DISCUS assumes the input file to be a SHEXL List type 5 containing <h> <k> l> <Fobs> <Fcalc> <phase angle in degree> The file is read as free format.

format

```
form {"gnu" | "stan" | "shelxl" | "hklf4" }
```

Defines the input format that will be used. Valid parameters are:

```
"gnu"
```

Sets the type of the output file to gnuplot. The intensities are written in blocks of intensities along the abscissa, with an empty line separating each block. Each line of each block contains: index along abscissa, index along ordinate, intensity, third index.

```
"stan"
```

Sets the output to standard file format, which is used by KUPL. The output file contains two initial lines and then a block for each data line separated by an empty line. The first line contains the number of data points along the abscissa and the ordinate, the second line the minimum and maximum value of the indices in reciprocal space along the abscissa and ordinate respectively.

```
"shelxl" [, {"fobs" | "fcalc"}]
```

Sets the input file format to SHELXL. DISCUS assumes the input file to be a SHLEXL List type 5 containing:

```
<h> <k> l> <Fobs> <Fcalc> <phase angle in degree>
```

The file is read as free format. All reflections are expanded by the appropriate symmetry operations, including proper phase transformation.
If the optional parameter is "fobs", the observed structure factor is used for the calculation, if the parameter is "fcalc", the calculated structure factor is used.

```
"hklf4" [,"all"]
```

Sets the input file format to the SHELX HKLF4 format. Without the optional parameter, DISCUS assumes the input file to be a SHLEXL HKLF4 file AFTER merging symmetrically equivalent reflections containing:

```
<h> <k> l> <Intensity> <sigma>
```

The file is read as free format. All reflections are expanded by the appropriate symmetry operations, including proper phase transformation.
If the optional parameter "all" is given, DISCUS will not expand the reflections by the reciprocal space symmetry operations. This is useful, if you have a data set that represents all of reciprocal space and has not been merged to the asymmetric unit in reciprocal space.
If you use this option, DISCUS automatically assumes that a PATTERSON will be calculated.

layer

```
layer <e11,e12,e13, e21,e22,e23, e31,e32,e33, inc1,inc2>
```

Sets the lower left, lower right and upper left corner of the Fourier plane to be calculated and sets the number of data points along the horizontal and the vertical

ll

```
ll <e11> , <e12> , <e13>
```

Sets the lower left corner of the Fourier plane to be calculated.

lr

```
lr <e21> , <e22> , <e23>
```

Sets the lower right corner of the Fourier plane to be calculated.

na

```
na <inc1>
```

Sets the number of data points that will be calculated along the abscissa.

no

`no <inc2>`

Sets the number of data points that will be calculated along the ordinate.

ord

`ord <switch>`

Defines which the component of the vector parallel to the ordinate is written to the output file.

rhoabs

`rhoabs <switch>`

Defines which the component of the vector parallel to the real space abscissa is written to the output file.

rholayer

`rholayer <e11,e12,e13, e21,e22,e23, e31,e32,e33, inc1,inc2>`

Sets the lower left, lower right and upper left corner of the real space plane to be calculated and sets the number of data points along the horizontal and the vertical

rholl

`rholl <e11> , <e12> , <e13>`

Sets the lower left corner of the real space plane to be calculated.

rholr

`rholr <e21> , <e22> , <e23>`

Sets the lower right corner of the real space plane to be calculated.

rhona

`rhona <inc1>`

Sets the number of data points that will be calculated along the real space abscissa.

rhono

`rhono <inc2>`

Sets the number of data points that will be calculated along the real space ordinate.

rhoord

```
rhoord <switch>
```

Defines which the component of the vector parallel to the real space ordinate is written to the output file.

rhoul

```
rhoul <e31> , <e32> , <e33>
```

Sets the upper left corner of the real space plane to be calculated.

reset

```
reset
```

Reset all Fourier definitions back to the default values at system start.

run

```
run
```

Starts the Patterson calculation.

scale

```
scale <scale-factor>
```

Defines an overall scale factor. Together with the automatic scaling by 1/vol(unit cell), this scale factor serves to put the electron density on an absolute scale of electrons per cubic Angstrom. The scale factor should effectively multiply the observed F(000) such that it is equal to F(000) calculated for one unit cell of the average structure.
Every time you enter "patterson" the scale factor is calculated to be number of atoms in crystal / number of atoms in one unit cell. If you use experimental data, you MUST specify the scale factor every time you enter the "patterson" sub menu.

set

```
set "accu",{"init"|"add"}
set "excl",{"none"|<value>}
set "mode",{"normal"|"sharp"|"super"}
set "origin",{"normal"|"subtract"}
set "rsym",{"apply"|"ignore"}
```

Set defines various parameters.

accu

`"accu",{"init"|"add"}`

Sets the mode for the accumulation of Patterson transforms.

`"init"`

The electron density array is initialized, i.e. set to zero.

`"add"`

The next Patterson transform is added to the previous value. This way several Patterson transforms will contribute to a single layer in direct space, which is necessary to create a Patterson section rather than a projection onto the layer. See the manual for further details.
If you use DISCUS to calculate the input data for the Patterson, you MUST separate the loop that calculates the Fourier from the loop that calculates the Patterson!!! Both parts of the program use the same variable to store the results and by mixing the loops, the result will be unpredictable nonsense !

excl

`set "excl",{"none"|<value>}`

Signals whether certain data points should be ignored.

`"none"`

All data points are treated as normal data points, regardless of their value. This is the default at program start.

`<value>`

Data points that are equal to <value> are excluded from the Patterson transform. You can use this option to mask out regions of the input data by setting their values to <value>. In the companion program KUPLOT, data points of value equal to -9999. are treated as excluded regions.

mode

`set "mode",{"normal"|"sharp"|"super"}`

Determines the calculation mode of the Patterson function. This mode may be a: "normal" Patterson using |F|*|F| "sharp" Patterson using |F|*|E| "super"-sharpened Patterson using |E|*|E| Currently the mode can only be used in combination with SHELX HKLF4 files.

origin

`set "origin",{"normal"|"subtract"}`

Determines whether the origin peak should be subtracted from the Patterson function. Since this depends on the data to be on an absolute scale, a Wilson plot is calculated and DISCUS puts the data on an approximately absolute scale.
Currently the origin peak can only be subtracted in combination with SHELX HKLF4 files.

rsym

```
set "rsym",{"appl"|"igno"}
```

Determines whether the space group symmetry is applied to all reflections. If the second parameter is "appl", all symmetry elements are applied to all reflections and the phases are transformed accordingly. This allows you to read a set of unique reflections and calculate the Patterson.

show

```
show
```

Shows the current settings for the Fourier transform file

type

```
type {"a"|"b"},
     {"intensity"|"amplitude"|"real"|"imaginary"|"phase"|"fobs"|"fcalc"}
```

Defines the content of file "a" or "b".
The Patterson transform is calculated from one or two input files. The two files must contain the intensity, real or imaginary part, or amplitude in one of the following combinations:

```
file a          file b
-----------------------
intensity
amplitude
real            imaginary
```

If the input format is "shelxl", DISCUS reads just one file. You can calculate the inverse Fourier from Fobs or Fcalc data by choosing the appropriate file type from one of the following choices:

```
file a          file b
-----------------------
amplitude
fobs
fcalc
```

stat

```
stat {"screen" | <outputfile> }
```

This command calculates the normalized structure factor, and displays relevant statistical values like $<|E^{**}2 - 1|>$ for all reflections and several sections in reciprocal space.
It requires the input file to be a SHELX HKLF4 file. The results are written to screen only or optionally into the output files: <outputfile>.ehkl Normalized structure factors for all input reflections <outputfile>.statistics Statistical values for the E <outputfile>.short Short summary of statistical values <outputfile>.histogram Number of E versus |E| for plotting
The input file must have been defined by ==> "file a,<infile>", For the statistics to be correct, the correct unit cell must have been defined.

ul

```
ul <e31> , <e32> , <e33>
```

Sets the upper left corner of the Fourier plane to be calculated.

wilson

```
wilson
```

This command reads a HKLF4 file and calculates Wilson statistics from a plot of ln(average(formfactor**2)/avera
versus sin(theta)/lambda. The scale factors: Fcalc = k *Fobs Icalc = k**2*Fobs and an overall B,
respectively U are calculated and displayed. A histogram of the logarithms versus sin(Theta)/lambda
is written to file <file>.wilson, where <file> is the base of the HKLF4 filename.
The input file must have been defined by ==> "file a,<infile>", The correct radiation and wave-
length must have been defined within the ==> 'fourier' menu, and the structure must contain
elements. For the Wilson statistics to be correct, the relative amount of atoms should be equal
to or at least close to the composition of the actual substance to which the data set belongs.

A.39   pdf

```
pdf
```

This command switches to the PDF mode of DISCUS. PDF stands for Pair-Distribution-Function.
The function actually used in DISCUS is the so-called reduced radial distribution function G(r)
which is defined as

```
  G(r) = 4 pi r ( rho(r) - rho(0) )
```

The value of "rho(r)" is the probability density of finding an atom at the distance "r" from a given
atom. Besides calculating G(r) from a given structure, the structure can be modified to match
a observed G(r) via the RMC method. Note, that this sub level uses many RMC commands
and will overwrite possible settings made in the RMC segment. All commands are listed in the
help entry 'commands'. Only commands unique to this level are described below.

commands

Valid commands at this sub level are :

```
@       ! Executes a macro (see main help)
=       ! Algebra (see main help)
```

calc ! Calculates PDF for current structure

```
data    ! Reads observed PDF for RMC refinement
dese    ! Deselects used atom types (see RMC level)
continue! Continue a stopped macro (see main help level)
echo    ! Echos a string, just for interactive check
eval    ! Evaluates an expression (see main help)
exit    ! Terminates the PDF sub level
help    ! Gives on-line help for 'pdf' (see main help)
```

```
i/jdese ! Deselect atoms (i/j) for PDF calculation
i/jsele ! Select atoms (i/j) for PDF calculation
mdes    ! Deselects molecules (see RMC level)
msel    ! Selects molecules for refinement (see RMC level)
reset   ! Reset PDF module
run     ! Start PDF refinement (see RMC level)
save    ! Saves structure or PDF to given file
sele    ! Selects used atom types (see RMC level)
set     ! Sets most PDF parameters
show    ! Show current PDF settings
stop    ! Stops execution of a macro (see main help level)
system  ! Executes operating system command (see main help)
wait    ! Waits for user input (see main help)
```

calc

```
calc
```

This command will calculate the PDF of the current structure. Note that the result is NOT automatically saved. This must be done using the command 'save' (-> save).

data

```
data <file>
```

This command will read an observed PDF from the file <file>. The file needs to be ASCII format with the following information on each line: r G dr w(G). Here 'r' is the distance and G the observed value of the reduced radial distribution function G(r). The value 'dr' is ignored and the last value gives the weight w(G) of G(r) to be used in the refinement (e.g. 1/(error of G)). NOTE: The data file must be in constant r steps and the first points MUST be equal to r=stepsize.

i/jdese

```
idese { "all" | <name> | <number> }, [ ... ]
jdese { "all" | <name> | <number> }, [ ... ]
```

This command deselects atom types given either by <name> or <number> for the PDF calculation. The two commands allow one to deselect atom types for each atom in a pair 'ij' contributing to the PDF calculation.

i/jsele

```
isele { "all" | <name> | <number> }, [ ... ]
jsele { "all" | <name> | <number> }, [ ... ]
```

This command selects atom types given either by <name> or <number> for the PDF calculation. All other atoms are ignored. This allows the calculation of differential or partial PDFs. The two commands allow one to select atom types for each atom in a pair 'ij' contributing to the PDF calculation.

reset

```
reset
```

This command resets most PDF module settings.

run

```
run
```

This command starts the RMC refinement to match the calculated PDF with the observed one.

save

```
save {"pdf" | "stru" | "mark"}, <file>
```

This command allows to save either the current PDF ("pdf") or the current structure ("stru") to the file named <file>. The parameter "mark" saves the distances of all pairs in <file> for plotting markers below a PDF. Markers can only be saved if the Gaussian mode is disabled. If DISCUS is operated within the DISCUS_SUITE, a file name that starts with "kuplot" is copied directly into the KUPLOT data array, and NOT written to disk.

set

```
set "subcommand"
```

This command allows to set most of the PDF parameters. Some of the commands are explained in the RMC section (-> rmc). The following "subcommands" are valid:

commands

Valid subcommands are:

```
"bound"    : toggles the use of periodic boundaries
"cycl"     : sets number of cycles to be calculated (see RMC level)
"delta"    : sets quadratic correlation factor (Gaussian mode only)
"corrquad" : sets quadratic correlation factor (Gaussian mode only)
"density"  : sets rho0 value manually
"diameter" : sets particle diameter for finite objects
"disp"     : sets output interval (see RMC level)
"finite"   : sets the correction type for 4Pi Rho for finite objects
"frange"   : sets the range in r used for the refinement
"gamma"    : sets linear correlation factor (Gaussian mode only)
"corrquad" : sets linear correlation factor (Gaussian mode only)
"mdis"     : sets minimal allowed distances between atoms (see RMC level)
"mode"     : sets RMC mode (relaxation/switch atoms) (see RMC level)
"move"     : sets sigma for generated RMC shifts (see RMC level)
"partial"  : sets partial weights for PDF (use with MIXSCAT program)
"poly"     : sets the background polynomial for finite objects
"qalp"     : sets resolution broadening (Gaussian mode only)
"qbroad"   : sets resolution broadening (Gaussian mode only)
"qmax"     : sets maximal Q value used for termination correction
"qsig"     : sets SIGMA Q for resolution correction
"qdamp"    : sets SIGMA Q for resolution correction
"rad"      : sets radiation used to calculate PDF
```

```
"range"    : sets r range for PDF calculation
"refine"   : sets which parameters are fined in an RMC cycle
"rdensity" : sets rho0 correction factor manually (see "density")
"scal"     : sets scale factor and refinement flag (see RMC level)
"shape"    : sets particle shape parameter for finite objects
"sigm"     : sets SIGMA for CHI2 calculation (see RMC level)
"srat"     : sets sharpening of low r peaks (Gaussian mode only)
"therm"    : toggles the convolution with thermal Gaussian
"weight"   : sets weight correction factor for finite objects
```

bound

```
set "bound","periodic" [,{"2D" | "3D"}]
set "bound","crystal" [,{"quick" | "exact"}]
```

This command allows the user to select whether periodic boundaries should be applied or not. In case of "periodic", periodic boundaries will be applied and the optional third parameter will determine whether periodic boundaries are applied in all directions ("3D") or only in those direction of the crystal that extends over more than a single unit cell ("2D"). The default is "3D". Note that in case of periodic boundaries, the calculation mode is always unit cell requiring the proper order of atoms in memory (i.e. no purge). If periodic boundary conditions are used, finite spherical objects can still be simulated by the use of ==> "set finite,sphere,<diameter>. The parameter "crystal" disables periodic boundaries. In this case the last optional parameters selects the calculation mode. The parameter "quick" will result in only neighboring unit cells being included again relying on the correct order of atoms in memory. The value "exact" will include all atoms making the calculation much slower, but it will work independent of the atom order, e.g. after a 'purge' command or with an arbitrary structure not generated by DISCUS. If periodic boundary conditions are not used, the background line -4 pi rho r should be corrected by parameters defined by ==> "set finite".

corrlinear

```
set "corrlinear",<g>
```

If the Gaussian convolution mode (-> set therm) is used, the PDF peak width will be modified by the term - g/r. This linear dynamic correlation factor <g> is specified with this command. A value of 0.0 will turn off this correction.
Identical to "gamma"

corrquadratic

```
set "corrquadratic",<d>
```

If the Gaussian convolution mode (-> set therm) is used, the PDF peak width will be modified by the term - d/r**2. The quadratic dynamic correlation factor <d> is specified with this command. A value of 0.0 will turn off this correction.
Identical to "delta"

delta

```
set "delta",<d>
```

If the Gaussian convolution mode (-> set therm) is used, the PDF peak width will be modified by the term - d/r**2. The quadratic dynamic correlation factor <d> is specified with this command. A value of 0.0 will turn off this correction.
Identical to "corrquadratic"

density

```
set "density", [<rho0> | "auto"]
set "rdensity", [<rho0> | "auto"]
```

This commands allows to specify a user defined number density <rho0>, e.g. in cases where no unit cell is defined. To use the number density determined by the unit cell volume, simply use the parameter "auto" (default). The command "rdensity" works in the same way, except <rho0> is treated as a factor for the calculated rho0 value.

diameter

```
set "diameter",<diameter>
```

This commands allows to specify a user defined particle diameter that is used in combination with the finite size treatment ==>"set finite,tanh"

finite

```
set "finite", {"periodic"| "poly",<diameter> | "sphere",<diameter> | "tanh"}
```

This commands defines how to deal with finite particles. Since their PDF does not contain distances beyond the diameter, the background line -4Pi Rho0 r must be corrected.
"periodic" The program is instructed not to apply a correction. The crystal is periodic. Also, use ==> "set bound,periodic".
"poly",<diameter> A polynomial function of order N is used to correct the -4Pi Rho0 r line. See ==> "set poly" for the corresponding parameters. The -4Pi Rho0 r line is set to zero for distances longer than the diameter.
"sphere",<diameter>
The effect of this setting depends on the periodic boundary flag. If periodic boundaries are chosen, the 'periodic' g(r) is multiplied by the shape function of a sphere of diameter <diameter>. f = 1 - 3/2 r/diameter + 1/2 (r/diameter)**3 For r > diameter g(r) is set to zero.
If periodic boundaries are disabled, only the -4Pi Rho0 r line is multiplied by the shape function of a sphere of diameter <diameter>. f = 1 - 3/2 r/diameter + 1/2 (r/diameter)**3 For r > diameter the background line is set to zero.
"tanh" A tanh function is used to correct the -4Pi Rho0 r line. See ==> "set diameter" and "set shape" for the corresponding parameters.
The only settings that may be used if the boundaries are set to periodic, are "periodic" and "sphere".

frange

```
set "frange",<rmin>,<rmax>
```

This command sets the range in "r" that will be used for the refinement. The range of the PDF that will be saved using the command 'save' is also determined by these settings.

gamma

```
set "gamma",<g>
```

If the Gaussian convolution mode (-> set therm) is used, the PDF peak width will be modified by the term - g/r. This linear dynamic correlation factor $<g>$ is specified with this command. A value of 0.0 will turn off this correction.
Identical to "corrlinear"

partial

```
set "partial","internal"
set "partial",<a1>,<a2>,<w>
```

This command allows one to specify the weight $<w>$ for each partial from atoms $<a1>$ and $<a2>$. This can be used in conjunction with the program MIXSCAT to calculate the correctly weighted differential PDFs from a model. If the second parameter is "internal", the weights are calculated from the scattering lengths of the elements (default).

poly

```
set "poly",<p1> [,<p2>...]
```

This command allows to set the parameters of a polynomial function that is used in combination with the finite size treatment ==>"set finite,poly". The background line -4 pi rho r is modified by subtracting this polynomial: -4 pi rho r - p1*r**1 - p2*r**2 - p3*r**3 - p4*r**4 - p5*r**5 The order of the polynomial is defined by the number of parameters given, which can be at most equal to five.
Notice that the polynomial starts with the linear term and does not contain a term p0*r**0 !

qalp

```
set "qalpha",<alpha>
```

PDF peaks are broadened at large values of r due to the instrument resolution of the measurement. This command sets the parameter $<alp>$ controlling the broadening which has the functional form $<alp>**2 * r**2$. Identical to "qbroad"

qbroad

```
set "qbroad",<qbroad>
```

PDF peaks are broadened at large values of r due to the instrument resolution of the measurement. This command sets the parameter $<qbroad>$ controlling the broadening which has the functional form $<qbroad>**2 * r**2$. Identical to "qalpha"

qmax

```
set "qmax",<qmax>
```

Since we can only measure the scattering up to a maximum value in Q, the resulting PDF will show termination ripples. In order to account for these, termination ripples can be calculated for the computed PDF. In order to do this the actual maximum Q value <qmax> in A**-1 must be entered. If the value is zero NO termination correction will be applied.

qdamp

```
set "qdamp",<qdamp>
```

This allows to enter a value <qdamp> for the correction of the limited resolution of the experiment. This results in a dampening of the G(r) function with increasing values of r. Again if the entered value is zero, no correction will be applied. Identical to "qsigma"

qsigma

```
set "qsigma",<qsig>
```

This allows to enter a value <qsig> for the correction of the limited resolution of the experiment. This results in a dampening of the G(r) function with increasing values of r. Again if the entered value is zero, no correction will be applied. Identical to "qdamp"

rad

```
set "rad","neutron"
set "rad","xray" [,<qx>]
set "rad","electron" [,<qx>]
```

This command selects the radiation used to calculate the PDF. The weighting factor for each atom pair is B(i)*B(j)/<B**2>. In case of neutrons ("neutron"), the values B(i) are simply the scattering length of the individual elements. For Xrays ("xray"), or electrons ("electron") the situation is more complicated since the scattering power varies as a function of Q. As an approximation the scattering power at a give Q value <qx> in A**-1 is used. The value <qx> can be taken as zero (default) in which case the number of electrons determines the weight. One other common way is to use the Q of the first peak in the diffraction pattern.

range

```
set "range",<rmax>,<dr>
```

This command allows to set the maximum r and the step width used to calculate the PDF. Note that if observed data are used, these values are set to the corresponding values of the actual data file. All values are entered in units of A.

shape

```
set "shape",<shape>
```

This commands allows to specify a user defined particle shape parameter that is used in combination with the finite size treatment ==>"set finite,tanh"

srat

```
set "srat",<rat>,<rcut>
```

This command allows the user to sharpen PDF peaks below a value of r=<rcut> by a factor of <srat> (usually < 1.0). Note that this will effect only calculations using the Gaussian mode (-> set therm).

therm

```
set "therm", {"off" | "gauss"}
```

This command allows the user to select between two calculation modes. If the parameter "gauss" is given, the contribution of each atom pair to the PDF is convoluted with a Gaussian. The width of this Gaussian is given by the individual thermal parameters B. If "off" is used, the contribution is a single "delta" type function.

weight

```
set weight,<weight>
```

Allows you to correct the weight factor used to multiply g(r) before adding it to -4pi rho0 r. For infinite objects this should be left at the default value of one. For finite objects or in case of uncertainties in the chemical composition it may differ.

show

```
show [{"all" | "pdf" | "mode" | "atom"}]
```

This command shows the various settings. If the command is entered without further parameters, the PDF settings are shown similar to the situation when the parameter is "pdf". The parameter "mode" results in the output of the current RMC mode. Note that some settings are listed that are not used by the PDF RMC refinement. The parameter "atom" corresponds to the listing of parameters like selected atoms, sizes of moves and minimal allowed distances between atom types. Finally the parameter "all" will cause all settings to be printed on the screen.

A.40 perioditize

perioditize

Switches to the periodititzer menu. The task in this menu attempts to rearrange a structure to be periodic and to be in the standard DISCUS sequence. Additional "VOIDs" might be introduced to fill the empty sites in a unit cell
Further help topics are:

reset

reset

Reset the menu to start-up conditions.

run

run

Start the task at hand.

set

set

Guide the perioditizer by setting: The expected number of unit cells The expected number of atoms per unit cell The expected positions within the unit cell

ncell

ncell {"auto" | <nx>, <ny>, nz>}

Set the number of unit cells, either explicitly with $<nx>$, $<ny>$, nz> or to be determined automatically with "auto"

natoms

natoms {"auto" | <natom>}

Set the number of atoms per unit cells, either explicitly with $<natoms>$ or to be determined automatically with "auto"

site

site atoms:[<names>], pos:[<x>,<y>,<z>]
     {, number:<value>}
     {, number:"next"}

Set which atoms are allowed at a site with coordinates pos. Both parameters are required. atoms:[$<names>$] Requires a comma delimited list of atom names or atom type numbers pos:[$<x>,<y>,<z>$] Requires a comma delimited position within the unit cell.
number:$<value>$ Apply the settings to this site number number:next Apply the settings to the next free site

wyck

```
wyck atoms:[<names>], pos:[<x>,<y>,<z>]
```

Set which atoms are allowed at a Wyckoff site with coordinates pos. Both parameters are required. atoms:[<names>] Requires a comma delimited list of atom names or atom type numbers pos:[<x>,<y>,<z>] Requires a comma delimited position within the unit cell.
All symmetrically equivalent sites are generated for the current space group!

show

```
show
```

Show the current settings.

A.41   plot

```
plot
```

Switches to the structure plotting sub menu. The structure can be written in a format suitable for plotting by several structure plotting programs such as KUPL, GNUPLOT, DIAMOND; ATOMS.
In particular a simplified CIF file can be written, which can be displayed by many different programs..
Further help topics are:

commands

Valid commands at this sub level are:

```
@        ! Executes a macro (see main help)
=        ! Assigns the value to a variable (see main help)
absc     ! Defines the abscissa of the plot
asym     ! Lists the content of the asymmetric unit
back     ! Selects the plot background for JMOL
bond     ! Selects the bond plotting style
chem     ! Lists all type of atoms present in the crystal
col      ! Defines the sequence in which the xy and z coordinates are written.
continue ! Continue a stopped macro (see main help level)
des      ! Deselects atoms
echo     ! Echos a string, just for interactive check (see main help)
exit     ! Terminates the plot sub level, returns to the main DISCUS level.
ext      ! Sets the extend of crystal space to be plotted
help     ! Gives on-line help for 'plot' (see main help)
hkl      ! Sets reciprocal space direction normal to plot slice
line     ! Draw a line into a "Jmol" plot
ordi     ! Defines the ordinate of the plot
outfile  ! Name of output file to which the structure is written
poly     ! Selects the polyhedron plotting style
prog     ! Selects the plotting program
prop     ! Selects the atom properties, which will be checked
reset    ! Reset and clear all definitions
run      ! Starts writing of the structure
sel      ! Selects which atoms are to be included in the plot
```

```
set     ! Sets type, color and size of atom in plot file
show    ! Shows settings for the plotting parameters
thick   ! Sets the half thickness of the plot slice
uvw     ! Sets direct space direction normal to plot slice
stop    ! Stops execution of a macro (see main help level)
system  ! Executes operating system command (see main help)
type    ! Sets type of output - crystal or unit cell projection
wait    ! Waits for user input (see main help)
vec     ! Sets a point inside plot slice
```

absc

```
abscissa <x>,<y>,<z>
```

Defines the abscissa of the plot. Instead of using the atom coordinates with respect to the crystallographic axes, the atoms are transformed into the system based on the abscissa, the ==> 'ordinate' and the normal to the plot slice ==> 'uvw', 'hkl'. Use these commands to plot the projection onto any plain other than the xy-, xz- or yz-plane.

asym

```
asym
```

Shows the content of the asymmetric unit. The names of those atoms, a number that is used as index for its scattering type, their position and temperature coefficient are listed. The number that is listed, is the number that refers to the scattering curve of that atom. It is contained in the variable m[<index>]. If a cell was read, all atoms are considered to be different, even if they are chemically identical and have the same temperature coefficient. If a whole structure was read, all atoms that are in the unit cell $0 <= xyz < 1$, are chemically unique and have a different temperature coefficient are included in the asymmetric unit.

back

```
back <r>, <g>, <b>
```

Sets the background color for a JMOL plot. Valid RGB values are in the range [0:255]

bonds

```
bonds { "all" | <name> | <number> } ,{ "all" | <name> | <number> },
      <rmin>, <rmax>, <diameter>, { <gray> | <r>,<g>,<b>}
```

Defines which bonds the plotting program JMOL / XBS will draw. This command does not affect the other plotting programs. Choose the pairs of atoms to be connected by bonds. Bonds that fall into the interval <rmin> to <rmax> are plotted. The thickness of the bond is set to <diameter>. You can specify the bond color as either a gray value in the range [0 - 1] or an RGB triplet.

---

chem

```
chem
```

Shows the type of all atoms present in the crystal. For each different atom present, its identifying number, its name and its temperature coefficient are listed. The list contains all the atoms in the asymmetric unit plus any atoms added to the structure. If atoms are removed from the structure, the program does not check whether there are any atoms of this type left. If a particular type of atom is completely removed from the structure, it will remain in the list of different atoms, and will be displayed by 'chem'.

col

```
col { "xyz" | "yzx" | "zxy" | "zyx" | "yxz" | "xzy" }
```

Defines the sequence in which the xy and z coordinates of the atoms are written. If the 'prog' "gnuplot" is used, this will allow the user to view the structure from different directions. For "kupl" this will give projections of the structure along the last direction.
If you plot the projection along a vector selected by ==> 'uvw', 'hkl' the interpretation of the coordinate triplets takes a different meaning: x represents the coordinate along the ==> abscissa y represents the coordinate along the ==> ordinate z represents the coordinate along the ==> 'uvw' direction The coordinates are, in this case, written as: 1.coordinate 2.coordinate 3.coordinate xyz abscissa ordinate uvw zxy uvw abscissa ordinate yzx ordinate uvw abscissa yxz ordinate abscissa uvw xzy abscissa uvw ordinate zyx uvw ordinate abscissa
For "xyz" the plot will show the projection along uvw onto the plane of the abscissa and ordinate. For "zxy" the plot will show the projection along the ordinate onto the plane of the uvw-direction and the abscissa. etc.
Since the thickness of the slice will often be limited by the ==>'thick' command, the sequence 'xyz' would be the most useful one.

des

```
dese { "all" | <name> | <number> } [ , <name> | <number> } ... ]
```

Deselects the atoms.

```
"all"      All atoms are deselected, no atom will be plotted.
<name>     Only the atom specified by <name> is deselected.
<number>   Only the atom specified by the scattering type <number> is
           deselected.
```

More than one atom can be deselected by the command.

ext

```
ext { "all" | "x",<xmin>,<xmax> | "y",<ymin>,<ymax> | "z",<zmin>,<zmax> }
```

Sets the extend of crystal space to be plotted.

```
"all"   The whole crystal is written to the plot file.
"x"     The extend along the x direction is limited to (including)
        <xmin> and <xmax>.
"y"     The extend along the y direction is limited to (including)
        <ymin> and <ymax>.
"z"     The extend along the z direction is limited to (including)
        <zmin> and <zmax>.
```

If any of the parameters "x","y","z" is used, the extend along the other directions is not changed, even if previously set with the "all" command.

hkl

```
hkl <h>,<k>,<l>
```

Sets the reciprocal space direction normal to the plot slice. The corresponding direct space direction ==> 'uvw' is calculated automatically. If <hkl> is set to "000", all atoms are plotted, otherwise all atoms are written to the output file, if they are within a slice normal to the <hkl> (and <uvw>) direction. The thickness of the plot slice is defined by the ==> 'thick' command, while the ==> 'vec' command defines a point that represents the center of the slice. Further restrictions on the atoms to be included can be applied by the ==> 'sel' and the ==> 'exte' commands.

mdes

```
mdes { "all" | <number> } [ , <number> ...]
```

This command deselects all or individual molecule types used for the plotting output.

mole

```
mole { "all" | "origin" }
```

This command allows the user to specify whether all ("all") atoms of a molecule should be used in the plot or only the origin ("origin") of the molecule should be used.

msel

```
msel { "all" | <number> } [ , <number> ...]
```

This command allows to select the molecule types to be used for the plotting output. By using 'msel' the atom selection becomes invalid. The parameter "all" will select ALL molecule types. Alternatively individual molecule types can be selected using the corresponding <number>. In order to use atoms again, use the 'sele' command.

line

```
line number:{<value>}, start:[<u>,<v>,<w>], end:[<u>,<v>,<w>]
line number:{"next" }, start:[atom,<index>], end:[atom,<index>]
```

This command will draw a line into a "JMOL" plot. Since several lines can be drawn, you can refer to a fixed line number <value> especially if you want to modify an existing line. Usually the easiest will be to specify "number:next" to draw another line.
The line start and end point are at 0,0,0 by default. If the optional start / end parameters are given, the line will use these points <u>,<v>,<w>, or the coordinates of the atoms specified by its index.

ordi

```
ordinate <x>,<y>,<z>
```

Defines the ordinate of the plot. Instead of using the atom coordinates with respect to the crystallographic axes, the atoms are transformed into the system based on the ==> 'abscissa', 'ordinate' and the normal to the plot slice ==> 'uvw', 'hkl'. Use these commands to plot the projection onto any plain other than the xy-, xz- or yz-plane.

outfile

```
outfile <filename>
```

Sets the name of the output file to which the structure is written.

poly

```
poly "off"
poly <center>, <neighbor>
     [, "nmin:"<nmin>], [, "nmax:"<nmax>]
     [, "dmin:"<dmin>], [, "dmax:"<dmax>]
     [, "face:"<style>], [, "hue:"<style>], [, "color:"<color>]
```

ets the definitions for plotting polyhedra in an interactive plot. urrently only an interactive plot with JMOL is supported. he command ==> 'run plot:inter' gives the details how to start an nteractive plot session.

```
"off"      Turn off any polyhedra.
<center>   Specifies the central atom type(s).
<neighbor> Specifies the neighboring atom type(s).
        A single central / neighbor atom type may be specified
        as its atom name, for example Si.
        To specify several choices as central atom type, place
        the list in single quotation marks: 'Si, Al'
```

The optional parameters are:

```
"nmin:"  Minimum number of neighboring atoms for the polyhedron
```

Defaults to zero or value of "nmax:" if "nmax:" is present

```
"nmax:"  Maximum number of neighboring atoms for the polyhedron
```

Polyhedra will be plotted if the central atom has a number of neighbors in the range nmin to nmax. Defaults to "nmax:0" If neither is specified, JMOL uses the bonds from the 'bond' command or the range from "dmin:" to "dmax:".

```
"dmin:"  Minimum distance of neighboring atoms
```

Defaults to "dmin:0.0"

```
"dmax:"  Maximum distance of neighboring atoms
```

Defaults to "dmax:0.0" If "dmin", "dmax:" are given, they effectively override the connectivity specified by a 'bond' command.

```
"face:"  {"flat" | "collapsed"}
         Plot the polyhedra with flat or collapsed faces.
"hue:"   {"solid" | "trans"}
         Plot the polyhedra with solid or transparent faces.
"col:"   {"auto" | <color_name>}
         Plot the polyhedra with the color of the central
         atom =="auto" or a user defined JMOL color.
```

prog

```
prog { "gnuplot" | "kuplot" | "atoms" |
       "xbs" | "frames" [,"append" | "init"] |
       "diamond" | "jmol" | "drawxtl" |
       "cif" }
```

Selects the plotting program for which the output is to be formatted.

```
"gnuplot"  The xyz coordinates of all selected atoms are written in the
           sequence defined by the ==> 'col' command.
"kuplot"   The xyz coordinates of all selected atoms are written in the
           sequence defined by the ==> 'col' command. A projection of the
           structure along the third direction of the 'col' parameter is
           plotted. The file contains the type, color and size code set
           with the ==> 'set' command. Use 'load cr,<file>' to read the
           file with KUPLOT.
"atoms"    The coordinates are written in a format that can be imported
           in ATOMS using the 'import - free format' option. In order
           to trick ATOMS into displaying the complete crystal, a
           new unit cell with the size of the crystal is used and the
           space group is set to P1.
"xbs"      The coordinates are written in the input format for XBS.
           All atoms are automatically transformed into Cartesian
           space. Cartesian b axis is parallel to the crystal b axis,
           the Cartesian c axis is parallel to the crystal c* axis,
           and the Cartesian a axis is normal to b and c*, right-handed.
           The atom color are set with ==> 'set', the bonds that
           will be plotted are set with ==> 'bonds'.
           The scale is set automatically, the initial viewing
           direction is along the ==> 'uvw' or 'hkl', the
           initial abscissa is along ==> 'abs'. The abscissa you
           specify is projected into the plane normal to the 'uvw'
           direction. You do not have to worry about the angle,
           as long as it is not equal to zero degrees.
"frames"   A movie file in the format of XBS is written to the
           output file.
```

```
              Remember to change the output file name to consist of an
              identical base as had been used for the main XBS output.
              If the optional second parameter is "init", the frame file
              is initialized.
              If the second parameter is "append", or if no
              second parameter is given, the current atom positions
              are added to the existing file. If the file does not
              exist, it is created.
"diamond"     The coordinates are written as the DIAMOND XYZ file type.
              space. Cartesian b axis is parallel to the crystal b axis,
              the Cartesian c axis is parallel to the crystal c* axis,
              and the Cartesian a axis is normal to b and c*, right-handed.
              All atoms are automatically transformed into Cartesian
              space. Import the file into DIAMOND by opening the file
              under the 'file' menu. Use file type 'XYZ'. Display all
              atoms using the "Add all Atoms" switch in the "Structure"
              menu.
              Caution, if only a partial crystal is written, the number
              of atoms is currently wrong. Use a "cif" format instead.
"jmol"        The coordinates are written as the JMOL XYZ file type.
              space. Cartesian b axis is parallel to the crystal b axis,
              the Cartesian c axis is parallel to the crystal c* axis,
              and the Cartesian a axis is normal to b and c*, right-handed.
              All atoms are automatically transformed into Cartesian
              space. Import the file into JMOL by opening the file
              under the 'file' menu. Use file type 'XYZ'. Display all
              atoms using the "Add all Atoms" switch in the "Structure"
              menu.
              Caution, if only a partial crystal is written, the number
              of atoms is currently wrong. Use a "cif" format instead.
"drawxtl"     The atoms are written in a format to be read by DRAWXTL.
"cif"         This creates a CIF file from the selected atoms. All
              coordinates are converted to a single unit cell.

              Use this output format for Atoms, Diamond, or JMOL as well.
```

prop

```
prop {"ignore"|"present"|"absent"} , <property> [, <property> ...]
```

Defines which properties an atom must have to be included in the plot.

"ignore" The properties listed will be ignored. An atom will be included in the plot regardless, whether it has the properties or not, "present" An atom must have the properties listed, to be included in the plot "absent" An atom is not allowed to have the properties listed.

The properties can be one of: "all" Apply the rule to all properties. "normal" The rules apply if an atom is a normal atom (not a void) "molecule" The rules apply if an atom is part of a molecule "domain" The rules apply if an atom is part of a domain "outside" The rules apply if an atom is outside of the crystal. This status is set if the atom has been cut off by a ==> 'boundary' command. "external" The rules apply if an atom is close to an external surface, either on the inside or on the outside. The flag is set by the ==> 'boundary' command. Individual distances to the boundary can be set within the ==> 'surface' menu through the ==> 'surface/fuzzy' command. "internal" The rules apply if an atom is close to an internal surface, either within the host or within the guest structure. The flag is set by placing a guest structure into the crystal through the ==> 'domain' menu. Individual distances to the boundary can be set within the ==> 'domain' menu through the ==> 'domain/fuzzy' command.

reset

```
reset
```

Reset all plot definitions back to the default values at system start.

run

```
run  ["plot:inter"] [, "geom:"[<px>,<py>]]
```

[, "kill:"["yes"|"no"]] [,"keep:"["yes"|"no"]] [, {"posit:default" | "posit:old" | "posit:step" | posit:[<px>,<py>] }] [, {"export:file.png" | "export:file.pdf" }]
Starts writing of the structure. If no output file name has been provided or no atoms been selected, an error message is given.
If the plot progamm was selected as "jmol", you can start an interactive plot session with the optional parameter "plot:inter". It may take a while for JMOL to open, you might want to add a ==> 'wait return' command after the 'run' command.
The default jmol window size is 500 by 500 pixels. You can change this via the optional "geom" parameter.
The default jmol window position is at pixels:[70,70]. You can change the position with the optional "posit:" parameter: "posit:old" Use the previous window position "posit:step" Increment the window by [+30,+30], Jumps to top/left at too large values "posit:["<px>,<py>"]" Place at user provided pixel positions Example posit:[200,300]
The optional parameter "export:" lets you export the initial jmol image into a file of type "png" or "pdf". Substitute "file" with a file name of your choice.
If previous instances of "JMOL" are to be terminated prior to the new plat provide the optional parameter "kill:yes". On a Linux type operating system processes "jmol" are killed, on Windows processes "java" will be killed. Be careful if other java programs are active! The intention is to kill only processes started by DISCUS.
At program end, DISCUS will kill all Jmol windows that were opened by this DISCUS. If you need to preserve these, use the optional "keep:yes" parameter.

sel

```
sel { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

Defines which atoms are included in the output. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"      all atoms of the crystal are included.
```

This includes the "voids" in the structure, which are stored as scattering curve number zero.

```
<name>    all the atoms called <name> of the crystal are included.
          This includes symmetrically not equivalent atoms.
<number>  all atoms of the crystal that are of scattering type <number>
          are included.
```

The selection made stay valid until explicitly deselected!

---

set

```
set { <name> | <number> },  <type>,<col>,<size>
set { <name> | <number> },  <type>,<r>,<g>,<b>,<size>
```

This command allows to set the marker type <type>, its color <col> and size <size> for the given atom or molecule. In case of atoms, the input can be given by its name <name> or scattering type number <number>. For molecules only <number> as the molecule type is a valid input. The resulting file will contains the coordinates of the atom and type, color and size in a row. This file can be read by KUPLOT using the command

```
  load cr,<file>
```

The possible values for <type> are:

```
  0 : no marker       5 : filled square    10 : line (/)
  1 : dot             6 : empty triangle   11 : line (\)
  2 : empty circle    7 : cross (x)        12 : line (−)
  3 : filled circle   8 : cross (+)        13 : vertical line
  4 : empty square    9 : line (|)              from y−axis
```

The possible values for the colors <col> are:
The allowed color values <icol> are:

```
  1 : red        5 : yellow       9 : dark blue      13 : cyan
  2 : green      6 : black       10 : dark magenta   14 : dark cyan
  3 : blue       7 : dark red    11 : dark yellow    15 : white (!)
  4 : magenta    8 : dark green  12 : gray
```

The size of the markers <size> is given relative to the marker size set in the plot program KUPLOT, i.e. a value of 1.0 will result in the full marker size whereas 0.5 will reduce the size by 50%.
If you are using 'gnuplot' as program type (-> 'prog') the settings will be ignored.
If you are using the XBS program, the color can be represented as an RGB triplet. If you give one value only, it is interpreted as gray shade in the range [0-1].

show

```
show
```

Shows the settings for the plotting parameters.

thick

```
thic <value>
```

Sets the half thickness of the plot slice in Angstrom. All atoms are written to the output file, if they are within a slice of thickness 2*<value> that contains the direct space point defined by ==> 'vec' and is normal to the direction defined by ==> 'hkl' or 'uvw'. Further restrictions on the atoms to be included can be applied by the ==> 'sel' and the ==> 'exte' commands.

---

title

`title <title>`

A string is read, which is used as title for the current frame of an XBS ==> 'prog frame' frame series.

uvw

`uvw <u>,<v>,<w>`

Sets the direct space direction normal to the plot slice. The corresponding reciprocal space direction <hkl> is calculated automatically. If <hkl> is set to "000", all atoms are plotted, otherwise all atoms are written to the output file, if they are within a slice normal to the <uvw> (and <hkl) direction. The thickness of the plot slice is defined by the ==> 'thick' command, while the ==> 'vec' command defines a point that represents the center of the slice. Further restrictions on the atoms to be included can be applied by the ==> 'sel' and the ==> 'exte' commands.

type

`typ { "crystal" | "projection" }`

This command switches between to plot modes. The type "crystal" will write the atom coordinates of the selected (-> 'sele') atoms on a file. The type "projection" will remove the translation and all atoms will be projected in a single unit cell. The second type can be read with KUPLOT using 'load cr,<file>' or 'load de,<file>,<dx>,<dy>'. See the KUPLOT online help for more details on the different file formats.

vector

`vec <v1>,<v1>,<v3>`

Sets the vector that represents the center of the plot slice. All atoms are written to the output file, if they are within a slice that is normal to the <hkl> (and <uvw>) direction and contains the direct space vector <v1 ,v2, v3>. The thickness of the plot slice is defined by the ==> 'thick' command, while the ==> 'vec' command defines a point that represents the center of the slice. Further restrictions on the atoms to be included can be applied by the ==> 'sel' and the ==> 'exte' commands.

A.42   powder

Switches to the powder diffraction sub level of DISCUS.
Two modes are used, the Debye scattering equation algorithm and a complete integration of reciprocal space.
The Debye mode is best suited for nanoparticles, as size effects are automatically taken into account. It becomes quite slow for large crystals.
The complete integration is best suited for large crystals, especially if large sections of reciprocal space are free from diffuse scattering.

---

commands

Valid commands at this sub level are:

```
@       ! Executes a macro (see main help)
=       ! Assigns the value to a variable (see main help)
asym    ! Lists the content of the asymmetric unit
chem    ! Lists all type of atoms present in the crystal
continue! Continue a stopped macro (see main help level)
echo    ! Echos a string, just for interactive check (see main help)
exit    ! Terminates the plot sub level, returns to the main DISCUS level.
help    ! Gives on-line help for 'plot' (see main help)
electron! Switches to electron calculation
neut    ! Switches to neutron calculation
reset   ! Reset and clear all definitions
run     ! Starts writing of the structure
set     ! Sets Theta limits, grid in reciprocal space, ...
show    ! Shows settings for the plotting parameters
stop    ! Stops execution of a macro (see main help level)
system  ! Executes operating system command (see main help)
wait    ! Waits for user input (see main help)
xray    ! Switches to electron calculation
```

electron

```
electron
```

Switches to electron calculation

neut

```
neut
```

Switches to neutron diffraction.

reset

```
reset
```

Reset all powder definitions back to the default values at system start.

run

```
run
run        ["mode:{"single"|"multiple"}]
           ["current:"<number>, "fraction:"<weight_fraction>]
```

Starts the calculation of the powder diffraction pattern.
The optional arguments define the mode for multiple phases versus a single phase calculation.
Default behavior is that of a single phase calculation.
The powder pattern of multiple phases are scaled to correspond to the weight fractions of all phases that you define. Currently it is the users responsibility to make sure all the <weight_fractions> add up to 1.0

---

show

```
show
show "scat", {<name> | <number> } [, {<name> | <number> }...]
```

Shows the current settings relevant to the calculation of a powder pattern.
The second form shows the scattering curve of the selected atom types, which may be given as the scattering number of as the element name.
The values for the scattering curve are stored into the res[] variable in the order: a1, a2, a3, a4, b1, b2, b3, b4, c

set

Sets various parameters for powder diffraction.

axis

```
set axis,{"q"|"tth"}
```

Defines the units of the x-axis you intend to use. "q" The units are in q=2PI h = 2pi dstar = 2pi 2 sin(theta)/lambda "tth" The units are in 2Theta
The internal calculations will use the corresponding limits and increments. Thus you need to match: <qmin>, <qmax>, <deltaq> to a "q" axis <tthmin>, <tthmax>, <delatth> to a "tth" axis.

back

```
set back, <p0> [, <p1> [,...]]
```

Set parameters for a background polynomial.

bragg

```
set bragg, {"incl"|"excl"}
```

This command applies to the full integration mode only. It allows you to include (=default) the intensities of the Bragg reflections or to exclude these. Actually rarely used these days...

calc

```
set calc,{"compl"|"debye"}
```

Defines the calculation mode. By selecting "compl" DISCUS will calculate the Fourier transform on a complete reciprocal lattice defined by ==> 'dh', 'dk', 'dl'. Alternatively, the switch "debye" will tell DISCUS to calculate the powder pattern by using the Debye formula. For details see the user manual.

corrlin

```
set corrlin, <value>
```

Active for PDF calculations only. The parameter corrlin allows to sharpen the width of inter-atomic distances at short values with an $1/r$ dependence. $sigma**2(thermal) - <value>/r$

corrquad

```
set corrquad, <value>
```

Active for PDF calculations only. The parameter corrquad allows to sharpen the width of inter-atomic distances at short values with an $1/r**2$ dependence. $sigma**2(thermal) - <value>/r**2$

delta

```
set delta,<res>
```

This command allows to specify the instrument resolution <res>. The calculated powder pattern is convoluted with a Gaussian of the width of <res>. If <res> is set to zero, no convolution is carried out.
See ==>'set profile' for a convolution by full profile function.

disp

```
set disp,[<switch>]
```

Switches anomalous dispersion on /off. If <switch> is "anom" anomalous dispersion will be calculated, if the switch is omitted, no anomalous dispersion is calculated.

dtth

```
set dtth,<value>
```

The powder pattern is printed at 2-Theta steps of <value>. All reciprocal points are binned to a grid along 2-Theta with intervals of <value>

dq

```
set dq,<value>
```

The powder pattern is printed at q steps of <value>. All reciprocal points are binned to a grid along q with intervals of <value>

dh

```
set dh,<value>
```

The calculation is carried out in reciprocal space intervals of <value> long the h00 direction.

dk

`set dk,<value>`

The calculation is carried out in reciprocal space intervals of $<value>$ long the 0k0 direction.

dl

`set dl,<value>`

The calculation is carried out in reciprocal space intervals of $<value>$ long the 00l direction.

energy

`energy <value>`

Defines the energy of the radiation. For X-rays and electrons the value is to be given in units of keV, for neutrons in units of meV.

four

`set four,{"four"|"stack"}`

Sets the switch for the Fourier calculation mode. For all normal crystals this should be set to "four". If you created a crystal using the ==>'stack" menu, you can speed up The Fourier calculation by setting the switch to "stack". See the stacking fault information regarding the details of this Fourier mode.

ka21

`ka21 <ratio>`

Sets the ratio of the intensities Kalpha2/Kalpha1, which is set by default to 0.500. This has an effect only if the corresponding wave length is defined with names like "CU12", "MO12", i.e. with the last two characters equal to "12".

lpcor

```
set lpcor, "none"
set lpcor, "bragg" [,<theta_mono>]
set lpcor, "neutron"
set lpcor, "synchrotron" ,<polarisation_fraction> [,<theta_mono>]
```

This command defines how the powder menu of DISCUS calculates the Lorentz-Polarisation correction.
"none" No correction is calculated
"bragg" [,<theta_mono>] Bragg-Brentano powder diffraction geometry is set. Optionally you can specify the monochromator angle <theta_mono>. If the diffractometer does not have a monochromator, set the value of <theta_mono> to zero. This corresponds to unpolarized radiation.

"neutron" A neutron powder diffractometer in Debye-Scherrer geometry is used.
"synchrotron" ,<polarisation_fraction> [,<theta_mono>] A Debye-Scherrer geometry at a synchrotron source is defined. The polarisation fraction is required. Optionally you can specify the monochromator angle.

partial

```
set partial, <i>, <j>
```

Instruct powder to calculate a partial powder pattern for atom type <i> and <j> in preparation for a partial PDF. Atom tyes <i> and <j> ay be the same atom type. This option is available for the Debye-Scattering-Equation only.
These "partial" PDFs are going to be of approximate scale. While the sum of all "partial" PDFs will add up to roughly the full PDF, contributions by atoms with small scattering power are likely overestimated. To obtain a quantitative coordination number for any given pair we strongly suggest to use the ==> 'chem/blen' command to obtain precise numbers.
Within the ==> 'output' menu save the PDF with identical setting as for a full PDF.

period

```
set period, "off"
set period, <diameter>
```

To calculate th PDF of an extended highly disorderd structure requires to calculate a powder pattern either via complete integration with a very fine grid or from a finite object that is very large. A good compromise is to calculate the PDF for a moderately sized spherical object via the Debye-Scattering-Equation. Discus will devide the resulting PDF by the shape function of a sphere with diameter <diameter>.

preferred

```
set preferred, "off"
set preferred, {"rietveld"|"march"}
set preferred, {"g1"|"damping"}, <damping>
set preferred, {"g2"|"portion"}, <portion>
set preferred, {"hkl"}, <h>,<k>,<l>
```

This command enables to calculate preferred orientation effects. Currently they can only be used if the full integration of reciprocal space is used, see ==> 'set calc'
"rietveld" Preferred orientation effects are calculated according to the equation: P(h) = portion + (1-portion)*exp(damping*alpha**2)
"march" Preferred orientation effects are calculated according to the modified March equation: P(h) = portion + (1-portion)*((damping*cos(alpha))**2+sin**2(alpha)/damping)**(3/2)
"portion" or "g1" is the fraction of the powder that is not affected by preferred orientation
"damping" or "g2" is a factor that determines how close a section of reciprocal space has to be oriented to the preferred axis.
"hkl" Defines the preferred orientation axis in reciprocal space
alpha**2 is the square of the acute angle between the vector h and the preferred orientation axis hkl. alpha is measured in degrees

---

profile

```
set profile,{"off" | "gauss" | "pseudo" | "tof"}
set profile,"eta",<eta>
set profile,"uvw",<u>,<v>,<w> ["fwhm:cagliotti" | "fwhm:area" | "fwhm:poly"]
set profile,"asym",<a1>,<b1>,<a2>,<b2>
set profile,"asym_i1",<a1_l>,<b1_l>,<a2_l>,<b2_l>
set profile,"asym_l",<a1_l>,<b1_l>,<a2_l>,<b2_l>
set profile,"asym_q",<a1_q>,<b1_q>,<a2_q>,<b2_q>
set profile,"width",<width>
set profile,"bank", difc:<P_difc>, difa:<P_difa>, difb:<P_difb>,
                    zero:<P_zero>, angle:<P_angle>
```

Defines the instrumental profile function.

The powder pattern is convoluted by this profile function.

"off" No convolution by an instrumental profile function is performed. A side effect, the parameter of ==> 'set delta' is set to zero.

"gauss" A simple Gaussian function is used as profile function with sigma as set by the ==> 'set delta' command.

"pseudo" A Pseudo-Voigt function defined as eta*Lorentzian + (1-eta)*Gaussian is used.

"tof" The neutron time-of-flight profile function as defined in GSAS-II is used-

"eta" Defines the mixing parameter ETA for the Pseudo-Voigt function

"uvw" Defines the FWHM for the Pseudo-Voigt function. As default or with the optional parameter "fwhm:cagliotti", the FWHM is calculated as: FWHM = sqrt(u*tan(2theta)**2 + v*tan(2theta) + w).

With "fwhm:area", the FWHM is calculated as: FWHM = sqrt(u*cos(2theta)**4 + v*cos(2theta)**2 + w), as defined in Chernyshov et al, Acta Cryst.A77, 497-505; doi 10.1107/S2053273321007506 .

Internally the parameters are transformed to give the identical FWHM for a Q-scale and a 2Theta-scale. The input values u,v,w must be given based on a 2Theta scale.

With "fwhm:poly", a purely empirical FWHM is calculated as FWHM = sqrt(u*Q**2 + v*Q + w)

"asym" The Asymmetry of the Pseudo-Voigt/Pearson VII function is calculated according to the definition as in Fullprof.

"asym_i1" Inverse TTH/Q dependent varition of the asymmetry parameters

"asym_l" Linear TTH/Q dependent varition of the asymmetry parameters

"asym_q" Square TTH/Q dependent varition of the asymmetry parameters

"width" Defines the width in multiples of the FWHM, over which the profile function is applied. The <width> defaults to 20*FWHM. This is a reasonable compromise between cut of errors at the tail of a peak and computational time.

bank" Defines the location and distances of the detector bank. The time-of-flight is calculated as: TOF = difc*d**1 + difa*d**2 + zero + difb/d where d is the d-spacing of a reflection: d = 2pi/Q = 1/d* = lambda/(2sin(Theta)) The angle is the banks central 2Theta and is used for the Lorentz correction L = d**4 * sin(Theta)

"tab" Defines the alpha and beta parameter of the TOF profile function.

"tsig" Defines the sigma parameter of the TOF profile function.

"tgam" Defines the gamma parameter of the TOF profile function. Note in GSAS-II the parameters gamma0, gamma1 and gamma2 are called Z,Y, X The "size" parameter is in microns and

defaults to 100my_m which results in no appreciable line broadening. The strain parameter
defaults to zero.

refine

```
set refine, scale
set refine, density
set refine, lattice [, "a"|"b"|"c"|"alpha"|"beta"|"gamma"|
```

Defines which parameters are refined during a RMC cycle. With the "set move" command at
the RMC menu, atom positions may be moved. This command allowes to additionally refine:
"scale" The scale parameter from "set weight" "density" the number density "lattice" The lattice
parameters, while maintaining restrictions imposed by the point group symmetry, or with the
optional parameters explicitely for a given parameter

scale

```
set scale,<value>
```

Sets a scale factor. The calculated powder pattern will be multiplied by this scale factor. Most
useful in combination with the background mode ==>'set back'

sh

```
set sh,<shift_h>
```

This command applies to the full integration mode only. It sets a shift of the grid on which
intensities are calculated by $<shift\_h>$. This will allow a substantial gain in computational
speed for situations in which only a part of reciprocal space has to be integrated. If, for example,
rods of diffuse intensity have to be calculated that have a finite width in reciprocal space in the
HK0 plane, then repeated calculations of the powder pattern with shift_h = plus/minus small
values can be used to cover this finite width, while dh can be set to 1.0

sk

```
set sk,<shift_k>
```

This command applies to the full integration mode only. It sets a shift of the grid on which
intensities are calculated by $<shift\_k>$.
See ==> 'set sh' for further info.

sl

```
set sl,<shift_l>
```

This command applies to the full integration mode only. It sets a shift of the grid on which
intensities are calculated by $<shift\_l>$.
See ==> 'set sh' for further info.

stepr

`set stepr,<value>`

This command applies to the Debye mode only. The interatomic distances are binned to a histogram with step width <value>. This defaults to 0.01 Angstrom at program start. As the histogram array is of fixed size, this limits the maximum distance that debye can handle. See ==> 'show config' at the main DISCUS level for the Debye histogram size. The maximum interatomic distance is then Histogram size * stepr

qmin

`set qmin,<value>`

Sets the lower limit for q. This must be set, if you intend to use the "Q" axis. You also need to set a 2Theta-range if you want to write the output on a 2Theta scale.

qmax

`set qmax,<value>`

Sets the upper limit for q. The powder pattern is calculated up to this q value. This must be set, if you intend to use the "Q" axis. You also need to set a 2Theta-range if you want to write the output on a 2Theta scale.

qzero

`set qzero, <value>`

Sets the zero point offset of experimental data compared to the true Q scale. The value will be subtracted from the calculated Q values. If the zero point of the experimental data is at say +0.1 A∗∗-1 on the absolute, true Q scale, we need to subtract this shift of +0.1 A∗∗-1 from the calculated data in order to match the experimental Q scale. All peaks in the output will be at a Q value of 0.1 A∗∗-1 less than the true value.

tthmin

`set tthmin,<value>`

Sets the lower limit for 2-Theta. This must be set, if you intend to use the "2Theta" axis. You also need to set a Q-range if you want to write the output on a 2Theta scale.

tthmax

`set tthmax,<value>`

Sets the upper limit for 2-Theta. The powder pattern is calculated up to this 2-Theta value. This must be set, if you intend to use the "2Theta" axis. You also need to set a Q-range if you want to write the output on a 2Theta scale.

---

tthzero

```
set tthzero, <value>
```

Sets the zero point offset of experimental data compared to the true 2THETA scale. The value will be subtracted from the calculated 2Theta values. If the zero point of the experimental data is at say +0.1R° on the absolute, true 2THETA scale, we need to subtract this shift of +0.1° from the calculated data in order to match the experimental 2THETA scale. All peaks in the output will be at a 2Theta value of 0.1° less than the true value.

temp

```
set temp,<switch>
```

Sets whether temperature coefficients are to be used or not. Valid values for <switch> are "use" and "ignore". The first two letters of the switch are sufficient.

wvle

```
set wvle,{ <name> | <value> }
```

Defines the wave length to be used. Anomalous dispersion will only be calculated if the wave length is given as symbol.

xray

```
xray  [table:{"waass" | "inter"}
```

Switches to X-ray calculation
The parameterized values according to Waasmaier & Kirfel, ACA (1995) A51, 416-431 were added and are used as default values. Optional parameters table:waas or table:inter allow to choose the Waasmaier or the International Tables Volume C.

A.43   private

```
private List_of_parameters
```

In the source code you will find a template for a "private" subroutine. Feel free to modify this to perform your tasks at hand.
Keep in mind that an update will overwrite any local private.f90.

A.44   project

```
project <u1>,<v1>,<w1>,<u2>,<v2>,<w2> [ ,<flag> ]
```

This command calculates the projection of vector <uvw 1> onto the second vector <uvw 2> and the projection of the first vector onto the plane normal to the second vector. The <flag> is a string of four characters that indicates whether the input vectors or the output vectors are given in direct or reciprocal space coordinates.

```
The four characters correspond to:
    first input vector,
    second input vector,
    projection of vector 1 onto vector 2,
    projection of vector 1 onto the plane normal to vector 2.
"drdd" means: vector one is given in direct space coordinates, vector 2
              is in reciprocal space coordinates (hkl) and the two
              projections are to be given in direct space coordinates.
Any combination of "d" and "r" is allowed.
The two output vectors are stored in the result variable:
"res[1]","res[2]","res[3]" projection of vector 1 onto vector 2
"res[4]","res[5]","res[6]" projection of vector 1 onto plane normal to
                           vector 2
"res[7]"                   angle between vector 1 and vector 2
```

A.45   property

```
property
```

Switches to the property menu of DISCUS. Here you can perform operations related to the properties of atoms. Such properties can be: N = normal, the atom is a normal atom (instead of a void) M = molecule, the atom is part of a molecule D = domain, the atom is part of a domain O = outside, the atom is outside of the crystal E = external, the atom is close to an external surface I = internal, the atom is close to an internal surface L = ligand, the atom is in a ligand molecule place by deco

The atom properties are set at: reading a unit cell or structure file inserting a domain into a crystal creating an external surface

The atom properties are used when atoms are selected. The global property selection rules apply to commands: 'replace', 'mmc', 'find env' The respective operations for these commands are carried out only if the selected atom fulfills the property selection criteria.

Local property selection rules are used in the menus: 'plot', To apply these, use the local 'property' command within the menu. The local property selection criteria are valid within the menu only.

Further help topics are:

commands

```
property  ! Define how command like replace, remove use the property
set/clear ! Set or clear flag for atoms
show      ! Show current property related settings
```

prop

```
prop {"ignore"|"present"|"absent"} , <property> [, <property> ...]
prop {"ignore"|"present"|"absent"} , atom:<type>,
     conn:{<number>|<name>}, nmin:<number>, nmax:<number>,
     emin:<number>, emax:<number>, no:<entry>
```

Defines which properties an atom must have to be included in commands ==> 'remove' ; 'replace'

"ignore" The properties listed will be ignored. An atom will be included in the plot regardless, whether it has the properties or not, "present" An atom must have the properties listed, to be included in the plot "absent" An atom is not allowed to have the properties listed.

The properties can be one of: "all" Apply the rule to all properties. "normal" The rules apply if an atom is a normal atom (not a void) "molecule" The rules apply if an atom is part of a molecule "domain" The rules apply if an atom is part of a domain "outside" The rules apply if an atom is outside of the crystal. This status is set if the atom has been cut off by a ==> 'boundary' command. "external" The rules apply if an atom is close to an external surface, either on the inside or on the outside. The flag is set by the ==> 'boundary' command. Individual distances to the boundary can be set within the ==> 'surface' menu through the ==> 'surface/fuzzy' command. "internal" The rules apply if an atom is close to an internal surface, either within the host or within the guest structure. The flag is set by placing a guest structure into the crystal through the ==> 'domain' menu. Individual distances to the boundary can be set within the ==> 'domain' menu through the ==> 'domain/fuzzy' command. "ligand" The rules apply if an atom is inside a ligand molecule that was placed onto the surface within the ==> 'deco' menu.

The second form with the optional arguments allows to define property like behavior based on the connectivities that must have been defined for the atom(s) of type <type>. nmin and nmax are minimum and maximum coordination numbers that the atom must have for the property to be true. emin and emax define an interval of excluded coordination numbers.

If you specify several properties of the second kind, the following rules apply: Any property applies only to atom types specified by "atom:" If an atom is not of any of the kinds specified by the "atom:" parameters the atom is never selected. If several properties are specified for a given atom type, they work with a logical AND. All conditions must be met for an atom to be selected. Keep in mind that the parameter "atom:O" will apply to all Oxygen atom types.

Example: prop present, atom:si,conn:1, nmin:0, nmax:3 This is true only if the selected atom is of type silicon with zero to 3 neighbors in its current connectivity number 1.

Keep in mind that you may have to update the connectivity list for this atom type / these atom types.

If you specify the atom by its name, the property will apply to all atom types with this atom name. Alternatively specify the atom type number instead of the atom name as in: prop present, atom:7,conn:1, nmin:0, nmax:3 !Apply to atom type 7

set/clear

```
set   <flag>, "types",  {"all" | <iscat> [, <iscat...]}
set   <flag>, "atoms",  {"all" | <start>,<end>}
clear <flag>, "types",  {"all" | <iscat> [, <iscat...]}
clear <flag>, "atoms",  {"all" | <start>,<end>}
```

Set or clear the property bit for atoms of selected type or for a range of atom numbers.
The flag can be any of "domain", "outside", "external", "internal"

A.46 purge

```
purge ["type:"{"no"|"yes"}]
```

Deletes all vacancies in the structure array, thus speeds up Fourier calculation. Due to internal calls, the result variables are changed!

If the 'purge' command does actually remove atoms or molecules from the structure array, the crystal will usually no longer be periodic or sorted in a fashion of contiguous unit cells that all have the same amount of atoms. DISCUS therefore needs to turn off the periodic boundary conditions and the "chem" mode "quick". These can be turned back on with an explicit ==> 'set crystal' command in the ==> 'chemistry' menu.

The optional parameter allows control over the handling of atom types. If "type:yes" is used the list of atom type will be purged as well. Any atom type for which no atom is present in the crystal will be deleted. Atom type numbers for the actual atoms will change! The affects all atoms whose type number is larger than the deleted atom type!

A.47   r2d

```
r2d <h>,<k>,<l>
```

This command transforms reciprocal space vector <hkl> into its corresponding coordinates in direct space. See "r2d" for the reverse transformation.

The resulting vector is stored in "res[1]", "res[2]", "res[3]".

The parallel, equivalent vector, normalized to 1/(length of [h,k,l]) is stored in "res[4]", "res[5]", "res[6]"

A.48   read

```
read
```

Branches to reading level of DISCUS. At this level a new unit cell or structure may be read. The old structure is lost ! If necessary save it with the –> 'save' command.

The 'read' menu is an old one. After a 'cell', 'stru', 'free' command DISCUS will leave the menu even without an 'exit'. As of version 5.17.1 comments no longer cause an 'exit'.

Further help topics are:

commands

Valid commands at this level:

```
@       ! Execute a macro file (see main help)
cell    ! Reads a unit cell from file
lcell   ! Reads a unit cell from file (large cells)
echo    ! Echo a string (see main help)
exit    ! Exits read sub level
free    ! Allow free modification of a structure.
help    ! Help to read (see main help)
stru    ! Reads a file containing a whole crystal
wait    ! Waits for user input (see main help)
```

cell

```
cell <name> [,<nx> [,<ny> [,<nz>]]]
      [,"identical:"{"tolerate"|"none"}]
      [,"radius:"<value>]
      [,"occupancy:"{"clear"|"apply"|"keep"}]
      [,"unique:"{"site"|"name"|"charge"|"biso"|"occ"}]
```

Reads the contents of a single asymmetric unit from file <name> and then generates the unit cell. Optionally the number of unit cells along x,y,z can be given to generated the whole crystal. The number of unit cells default to 1. The unit cells are arranged symmetrically around the origin, .i.e. from -nx/2 to +nx/2 etc. After the unit cell has been read the program returns to the main DISCUS level.

If the file ends in 'cell', 'stru', 'CELL', 'STRU' the standard DISCUS file format is assumed. If the file ends in 'cif', 'CIF', a CIF-file format is assumed. If the file ends in 'txt', 'TXT', a CMAKER-file format is assumed.

The optional parameters define how discus will handle atoms that are on identical positions. This might arise for example if you try to describe a split position with the partially occupied positions close to each other. If the optional parameter "identical" is set to "tolerate" atoms will be tolerated that are no closer than the radius specified by the optional radius parameter. To tolerate atoms on identical positions you must set the radius to zero. Defaults are "none" which means DISCUS tolerates atoms that are at least 1.0*10**-5Angstrom apart.

As of version 5.18.0 occupancies were introduced to DISCUS. The optional parameter "occupancy" tells DISCUS how to interpret the values found in the input file. The default is "clear" All occupancies are reset to 1.0 and DISCUS behaves as prior to 5.16.0 A warning message is issued, but a macro will continue. "apply" During the expansion of the asymmetric unit to the full crystal the actual atoms are in part replaced by "voids" to achieve (at least approximately) the occupancy "keep" The occupancies are kept at the values specified in the input file. The actual atom list will be that of a fully occupied crystal.

```
      [,"unique:"{"site"|"name"|"charge"|"biso"|"occ"}]
```

This optional parameter defines if atoms are assigned to a new atom type or are considered uniqe new atom types "site" With this parameter or without the optional parameter each atom in the input file is treated as a new atom type. "name" All atoms, whose elemental symbol are identical are treated as one atom type "charge" All atoms, whose elemental symbol and charge (if present) are identical are treated as one atom type "biso" In addition to name, charge the B-value must be identical "occ" In addition to all other the occupancies must be identical

lcell

```
lcell <name> [,<nx> [,<ny> [,<nz>]]]
      [,"unique:"{"site"|"name"|"charge"|"biso"|"occ"}]
```

This command is similar to -> cell, however in contrast to 'cell' only those atoms are treated as new atom type that have a different name or B value. This allows one to read large asymmetric units (e.g. from a protein) without creating that many different atom types.

```
      [,"unique:"{"site"|"name"|"charge"|"biso"|"occ"}]
```

This optional parameter defines more in detail if atoms are assigned to a new atom type or are considered uniqe new atom types "site" With this parameter or without the optional parameter each atom in the input file is treated as a new atom type. Essentially the command acts as a "cell" with default conditions "name" All atoms, whose elemental symbol are identical are treated as one atom type "charge" All atoms, whose elemental symbol and charge (if present) are identical are treated as one atom type "biso" In addition to name, charge the B-value must be identical "occ" In addition to all other the occupancies must be identical

free

```
free [<a>,<b>,<c>,<alpha>,<beta>,<gamma> [,<name> | <number> [, "2"]]
     [, setting:abc]
```

No structure is read. The space group is set to "P1". The lattice constants are set to the optional values on the command line. Default values are 1,1,1 Angstrom and 90,90,90 degrees. The structure does not contain any atoms.

The optional 7th parameter allows you to set a space group for this empty structure. The parameter may either be the space group name or the space group number. This has no effect on the current structure or on atoms you may insert later. It does allow you to save this structure and then to read it as a unit cell file. It will then be expanded to a full unit cell content according to the space group For those space groups where a second origin choice is allowed, the optional eighth parameter may be set to "2" to indicate the alternative origin choice. The last optional parameter "setting:" allows to use a non-standard setting within the orthorhombic space groups. specify the desired setting as either of: abc, bca, cab, acb, cba, bac

stru

```
stru <name> [, site:{"differ"|"equal"}]
     [,"unique:"{"site"|"name"|"charge"|"biso"|"occ"}]
```

Reads a whole structure from file <name>. The space group information is ignored, nor is a check for consistency of space group and lattice constants performed.

If the file ends in 'cell', 'stru', 'CELL', 'STRU' the standard DISCUS file format is assumed. If the file ends in 'cif', 'CIF', a CIF-file format is assumed. If the file ends in 'txt', 'TXT', a CMAKER-file format is assumed. If the file ends in 'cssr', 'CSSR', a RMCprofile-file format is assumed.

The optional parameter "site:" determines how atoms are treated that are on different sites in the unit cell, but are otherwise identical. The default "site:equal" is to treat these as the same atom type. With the option "site:differ" these atoms are treated as separate different atom types. The option is only applied, if the crystal consists of a total number of atoms that is an integer multiple of the number of atoms per unit cell.

```
[,"unique:"{"site"|"name"|"charge"|"biso"|"occ"}]
```

This optional parameter defines if atoms are assigned to a new atom type or are considered uniqe new atom types "site" With this parameter each atom in the input file is treated as a new atom type. Be aware that this might result in a huge number of atom types beyond the memory capabilities of you computer "name" All atoms, whose elemental symbol are identical are treated as one atom type "charge" All atoms, whose elemental symbol and charge (if present)

are identical are treated as one atom type "biso" In addition to name, charge the B-value must be identical "occ" In addition to all other the occupancies must be identical

exit

`exit`

Exits the read menu, in case no 'cell', 'free', or 'stru' command is used.

A.49   remove

```
remove <start> [, <end> ]
remove <start> [, <end> ] , "mole" [ ,"type", <t_start> [, <t_end>]]
```

The parameters <start> and <t_start> can take one of the following values: { <start> | "all" | "last" }

Removes atom(s) number <start> to <end> from the structure.

If you use the keyword "all", all atoms are removed. If you use the keyword "last", only the last atom is removed.

Internally, this command only sets the scattering type of these atom(s) to zero. Otherwise the atom(s) is treated as if present. Specifically, in the array containing the structure the sequence of atoms is not changed. For this command variation, the atoms are removed, no matter what their individual properties are and whether they fulfill the global property selection rules ==> 'property'. For all atoms that are removed, the property NORMAL is turned off.

The second form of the command removes atoms that are part of the molecule(s) <start> through <end>. Optionally you can specify which molecule type will be removed. Only those molecules of a type in the range <t_start> through <t_end> will be removed. The default for this option is all type of molecules.

If you use the keyword "all", all molecule types are removed. If you use the keyword "last", only the last molecule type is removed.

For all atoms that are removed, the property NORMAL is turned off.

A.50   replace

```
replace <index>,<at> [,<dbw>]
replace <at1>,<at2>,{ "all" | <site> },<prob>
replace <mo1>,<mo2>,"mol"
replace <mtyp>,<mol>,"mol",<prob>
```

This command allows to replace atoms and molecules. The first two variations operate on atoms. The first variation replaces the atom at index <index> with atom name (or number) <at>. It the atom type does not already exist in the crystal, the temperature factor <dbw> has to be given as additional parameter. For this command variation, the atom no. <index> is replaced, no matter what its individual properties are and whether it fulfills the global property selection rules ==> 'property'

Alternatively the atom type <at1> can be replaced by <at2> within the complete crystal with a probability of <prob>. This works much (!!) faster than replacing using DISCUS loops. The

third parameter specifies if all "all" occurring atoms of type <at1> might be replaced or just those on the given site <site>. For this replacement by probabilities, only those atoms are included that fulfill the global property selection rules ==> 'property'.

For both variations, if the atom is replaced by a void, the property NORMAL is turned off, if a void is replaced by another atom type the property NORMAL is turned on.

The last two variations of the 'replace' command operate on molecules. The third variant replaces molecule <mo1> by <mo2>. The positions are exchanged relative to the origin of the molecule (atom 1) and both molecules MUST have the same length. The last variation works the same way, just all molecules of type <mtyp> will be replaced with molecule <mol> with the given probability of <prob>.

A.51   rmc

```
rmc
```

Enter "Reverse Monte Carlo" section of discus. This part of the program allows to "fit" a model atom configuration to measured diffuse scattering data. The difference between calculated scattering and observed scattering is minimized. Moves minimizing this difference are always accepted, other moved only with a probability of exp (-diff**2).
(Nield et al., Acta Cryst A51 (1995), 763-771 for more information)

commands

Valid commands at this sub level are:

```
@        ! Executes a macro (see main help)
=        ! Algebra (see main help)
data     ! Read experimental data for RMC fit
dese     ! Deselects the atom types allowed for RMC moves/switches (-> sele)
continue ! Continue a stopped macro (see main help level)
echo     ! Echos a string, just for interactive check
eval     ! Evaluates an expression (see main help)
exit     ! Terminates the rmc sub level, returns to the main DISCUS level.
help     ! Gives on-line help for 'rmc' (see main help)
mdes     ! Deselects molecules
msel     ! Selects molecules for RMC refinement
reset    ! Reset RMC settings (i.e. reset plane counter)
run      ! Start RMC fit
save     ! Saves structure or intensity to given file
sele     ! Selects the atom types allowed for RMC moves/switches (-> dese)
set      ! Sets most RMC parameters
show     ! Show current RMC settings
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
wait     ! Waits for user input (see main help)
```

data

```
data {"x-ray"|"neutron"},{<name>|<value>},<file>,<wic>,
     <e11>,<e12>,<e13>,<e21>,<e22>,<e23>,<e31>,<e32>,<e33>
```

This command reads the experimental data stored in <file>. The parameter "x-ray" / "neutron" selects what kind of scattering data are stored in the file. The second parameter is the wavelength which can be given as <name> (like cua1 ..) or as value (like 1.23). After the file name <file> the weighting scheme to be used can be specified. The parameter <wic> can take the following values:

```
"one"  : w(Int) = 1.0              "squa" : w(Int) = Int**2
"sqrt" : w(Int) = SQRT(Int)        "inv"  : w(Int) = 1/Int
"log"  : w(Int) = LOG(Int)         "isq"  : w(Int) = 1/SQRT(Int)
"lin"  : w(Int) = Int
```

All other values for <wic> are taken as the name of a file containing the weights. This file and the data file have to be the same type ! Finally the three corners of the input data plane are given in the following order:

```
<e11>,<e12>,<e13> : hkl's of the lower left corner
<e21>,<e22>,<e23> : hkl's of the lower right corner
<e31>,<e32>,<e33> : hkl's of the upper left corner
```

The format of the input file is set by the 'set data' command. For more information about the supported data file formats see help for the 'set data' command.
By repeating the data command you can read several planes with experimental data for the RMC fit. To start again use the 'rese' command.

dese

```
dese { "all" | <name> | <number> }, [ ... ]
```

This command deselects atom-types given either by <name> or <number> for the RMC moves, i.e. 'dese' those atom-types which should not be moved or switched.

msel

```
msel { "all" | <number> } [ , <number> ...]
```

This command allows to select the molecule types to be used for the RMC refinement. By using 'msel' the atom selection becomes invalid. The parameter "all" will select ALL molecule types. Alternatively individual molecule types can be selected using the corresponding <number>. In order to use atoms again, use the 'sele' command.

mdes

```
mdes { "all" | <number> } [ , <number> ...]
```

This command deselects all or individual molecule types used in the RMC refinement.

reset

```
reset { "all" }
```

This command resets the RMC parameters. In this version only the 'plane counter' is reset and the 'data' command will start reading plane 1 again.

run

```
run
```

Start the RMC fit. After the run is finished, the following values are stored in the res[i] variables:

```
res[1] : last value of CHI2
res[2] : number of tried moves
res[3] : number of accepted 'good' moves
res[4] : number of accepted 'bad' moves
res[5] : DELTA CHI2 average during the last run
res[6] : DELTA CHI2 standard deviation during the last run
res[7] : DELTA CHI2 maximum value during the last run
res[8] : elapsed time / cycle in seconds
```

save

```
save <subcommand>
```

This command is for saving the calculated intensities or the resulting structure. Valid subcommands are:

lots

```
save "lots",<name>
```

This command saves the current set of lot origins to the file named <name>. These lot origins can be used as default in subsequent refinement rather than creating a new set of random lots (-> 'set lots,..').

stru

```
save "stru",<name>
```

This command saves the current structure to the file named <name>. The format is identical to the unit cell or structure file. Each atom is written in format A4,3(2X,F11.6),5X,F6.4. The file name can be created by an format string and one or more numbers -> filenames.

scat

```
save "scat",<plane>,<name>
```

Save "scat" saves the current computed intensities of plane <plane> in the following format: h,k,l,skal*(a**2+b**2)+back (= intensity).

sym

```
save "sym",<plane>,<sym>,<name>
```

Does the same as 'save scat,..', but the user can save also the resulting intensities of symmetric equivalent planes by giving the number of the sym.-transformation (1=data corresponding to input data).

---

sele

`sele { "all" | <name> | <number> }, [ ... ]`

This command selects atom-types given either by <name> or <number> for the RMC moves, i.e. 'sele' those atom-types which should be moved or switched during the RMC cycles.

set

`set "subcommand"`

This command allows to set most of the RMC parameters. The following "subcommands" are valid:

commands

Valid subcommands are:

```
"aver"     ! Sets the portion of the crystal to be used for <F> calc.
"back"     ! Controls if background is calculated during RMC cycles
"const"    ! Sets possible constrains for scaling factor/background
"cycl"     ! Sets number of RMC cycles to be calculated
"data"     ! Sets data type of input/output diffuse scattering files
"dbw"      ! Ignore/use temperature factor
"disp"     ! Sets output interval
"log"      ! Controls if "rmc.log" will be created during calculation
"mdis"     ! Sets minimal allowed distances between atoms
"mode"     ! Sets RMC mode (relaxation/switch atoms)
"move"     ! Sets sigma for generated RMC shifts
"range"    ! Sets Q-range of data used for RMC refinement
"scal"     ! Controls if scaling factor is calculated during RMC cycles
"sigm"     ! Sets SIGMA for CHI2 calculation
"sym"      ! Controls if symmetric equivalent planes of input data should
             be used for RMC calculations.
```

aver

`set "aver",<per>`

This command sets the portion of the crystal, that should be used to calculate <F>. The variable <per> is given in %. The average structure factor <F> is calculated and subtracted from the overall structure factor calculated at the beginning of the RMC refinement. Starting from these values the contributions of the changes atoms during the RMC cycles are calculated and the structure factor adjusted. There is no check weather <F> changes significantly during the refinement.
To switch the use of <F> off, just set the value of <per> to zero. After setting the value of <per> the structure factor will be recalculated. This can also be used to recalculate the Fourier transform and the average from time to time during the RMC refinement if the average structure is changing.

back

```
set "back",{"on"|"off"},[<value>], [<plane>]
```

This command controls if a background parameter should be calculated and used for the RMC fits. If the calculation of the background parameter is switched off its value can be set to <value> otherwise the default is 0. The second optional parameter allows to specify the plane the current value of <value> is set for. If the parameter <plane> is omitted, the value of <value> is set for ALL experimental data planes.

const

```
set "const",<p1>,<p2>
```

This command allows to constrain the scaling factor and background parameter of the two input data planes number <p1> and <p2>. If <p1> and <p2> are equal, no constrain is set, and the planes have their individual scaling factor/background. If two planes should have the same values for scaling factor/background, the constrain is set by <p1> not equal <p2>. The value of <p1> is the plane which should take the values for scaling factor/background of plane <p2>. Not, that the value of <p1> has to be smaller than <p2>.
If for example scaling factor/background for plane 2 should be the same as for plane 1, use the command 'set const,2,1'. If there are more than two planes to constrain use the command several times. Plane 1 to 3 can be constrained by 'set const,2,1' and 'set const,3,2'. To unconstrain the planes again type 'set const 1,1', 'set const,2,2' and 'set const,3,3'.

cycl

```
set "cycl",<icyc>
```

This command sets the number of RMC moves to be done after the 'run' command. Note that the number given is the actual number of RMC moves to be tried and *NOT* the number of RMC cycles, which is normally defined as the number of moves necessary to visit every crystal site once on average.

data

```
set "data",{ "nipl" | "pgm" }
```

This commands sets the data type of the experimental data files used as input for RMC and the corresponding output files. There are currently two file formats supported:

```
"nipl" : An ASCII file containing the Z-values in the following format:

        nx ny                  ! # pixels in x- and y direction
        xmin xmax ymin ymax    ! Dimensions in x- and y-direction (ignored)
        data                   ! ny rows of nx intensity values
"pgm"  : PGM file (pbmplus package). In the current version, only
        ASCII PGM files are supported. To convert the binary files
        into the ASCII PGM version, use the 'pnmnoraw' program,
        which is part of the 'pbmplus' package.
```

disp

```
set "disp",<id>
```

This command sets the update interval of the output.

dbw

```
set "dbw",{"on"|"off"} [,<plane>,...]
```

This command controls whether the Debye-Waller factor is ignored or used when the scattering amplitude is calculated. If no parameter <plane> is given, the setting affects ALL read data planes.

log

```
set "log",{"on"|"off"}
```

This command controls if in the RMC main loop the screen output should also be logged to the file "rmc.log". This file contains only the last output and should give the possibility to watch the programs progress because cached output redirection might not update often enough.

lots

```
lots "off"
lots {"box" | "ellipsoid"},<lx>,<ly>,<lz>,<n>,{"yes" | "no"},[<fname>]
```

This command allows to use the 'lot' option during the RMC refinement. For details about the command and its parameters see the help file entry for 'four lots'.
The optional last parameter is a file name for a file containing a predefined set of lot origins. This file can be created using 'save lots,..' after a RMC refinement was carried out. If no file name <fname> is given, DISCUS creates a new random set of lots for the RMC fit.
The advantage of using lots is the combination of a smooth calculated diffraction pattern as average of the Fourier transform of small crystal volumes (lots). This forces RMC on a 'local' scale in real space without the problem of large errors and noisy patterns as for small crystal sizes.
It is important to subtract the average structure factor (-> set aver) when using this mode even if "periodic boundary conditions" are fulfilled, because the Fourier transform of the lots shape needs to be subtracted as well as <F>. This can be done using the command 'set aver ..'.

mdis

```
set "mdis",{ "all" | <name> | <number> },
          { "all" | <name> | <number> },<dist>
```

This command sets the minimal allowed distance between two atom types to the value of <dist> A. A generated move will only be accepted if the moved atom is not closer than <dist> to the others. The atom-types to be selected can be specified by either name or number or all atom-types can be selected by typing "all".

mode

```
set "mode",<prob>,
         { "shi[ft]" | "swc[hem]" | "swd[isp]" | "ext[ernal]" },
         [{ "a[ll]"   | "l[ocal]"  | "sl[ocal]" | "si[te]"     }]
```

This command allows to select one of the three RMC operation modes.
The first parameter specifies the probability for the current mode. The probabilities do not have
to add up to one, they will be scaled at run time. The possible modes are:

```
"shift"    : The selected atoms are displaced by a random move. The
             maximum size of a move can be specified via "set move .."
             command.
"swchem"   : Here two atoms are selected at random and their chemistry
             is switched (if the atoms are selected via "sel" command).
             If your structure consists of only ONE atom type this mode
             will just waste computer time :-)
"swdisp"   : Here the displacement from the average structure of two
             selected atoms of the SAME type are switched. The mode
             can only properly work, if there are initial displacements
             generated e.g. by the "therm" command.
"external" : The modification of the structure for each RMC move is done
             using an user supplied external subroutine. Check the
             source file 'extrmc.f90' for details how to supply such a
             subroutine.
```

The third optional parameter specifies if the pair of the atoms to be selected for the SWCHEM
and SWDISP mode should be restricted in any way. The following settings are valid, the setting
"all" is the default, which will be taken if no second parameter is entered.

```
"all"      : Any atom within the crystal can be chosen as second atom.
"local"    : Only atoms within +-1 unit cell from the first selected
             atom will be chosen.
"slocal"   : Same as "local" only the second atom must be on the same
             site within the unit cell as the first one.
"site"     : Atom on the same site in any unit cell are allowed as
             second atom.
```

move

```
set "move",{ "all" | <name> | <number> },<sx>,<sy>,<sz>
```

Sets an user defined sigma for the generates moves. The created shifts are Gaussian distributed.
The default is a sigma of 0.2 unit cell. Note that the values of $<sx>$, $<sy>$ and $<sz>$ are given
in unit cell units to speed up the RMC runs. In case molecules are selected $<number>$ specifies
the molecule type, otherwise atoms are addressed by their $<number>$ or $<name>$.

range

```
set "range","all"
set "range",<qmin>,<qmax>
set "range",<hmin>,<kmin>,<lmin>,<hmax>,<kmax>,<lmax>
```

This command allows to set a range in Q which will be used for the RMC refinements. This can be useful if e.g. occupational disorder should only be refined based on the low Q part of the input data whereas the displacements should be refined using the complete diffuse scattering data set.

There are 3 ways to specify the Q range: "all" will use the complete input data set for the fit. The Q-range can be given as <qmin>, <qmax> (in reciprocal A) or as hkl's (<hmin>,...,<lmax>).

scal

```
set "scal",{"on"|"off"},[<value>], [<plane>]
```

This command controls if a scaling factor should be calculated and used for the RMC fits. If the calculation of the scaling factor is switched off it can be set to the value given by <value>. If no value is given, the default is 1. The second optional parameter allows to specify the plane the current value of <value> is set for. If the parameter <plane> is omitted, the value of <value> is set for ALL experimental data planes.

sigm

```
set "sigm",<sigma>
```

This command sets the value for SIGMA. CHI2 is calculated by

```
CHI2 = SUM (Fobs-Fcalc)**2/SIGMA**2
```

The value of SIGMA is important for the calculation. A too large value will allow nearly every move and a too small value will drive the fit in the nearest local minimum. The default value is 0.01 If the value of SIGMA is set to zero, only moved improving the fit will be accepted. For test and debug runs a setting of SIGMA to -9999. will result in the acceptance of ALL moves.

sym

```
set "sym",{"on"|"off"}
```

This commands controls the use of symmetric equivalent planes to the read data-planes for the RMC calculations. If this flag is set to "off", only the experimental data are used for CHI2 calculation, if the flag is "on" the symmetry of the measured data is taken into account.

show

```
show [<subcommand>]
```

Shows the current parameter settings of the RMC subsection. The following subcommands allow to select certain parameters of interest. The allowed subcommands are:

```
"all"  : show all settings (same as calling 'show' without parameter)
"atom" : show selected atoms, minimal distances, ...
"data" : show information about experimental data
"mode" : show general RMC calculation settings
"rval" : show R-values for current refinement
"sym"  : show information about using symmetry information of read planes
```

A.52   reset

reset

Places DISCUS into the state at program start.
Version info:v.5.24.0 resets: crystal, connectivity, plot and internal storage

A.53   save

save
save <name>

Saves the current structure to file <name>
The format is identical to the unit cell or structure file. Each atom is written in format A4,3(2X,F11.6),5X,F6.4
If the 'save' command is used with one parameter, the parameter is interpreted as output file name. See the entry ==> 'filenames' under the command language entry in the main help section on information about valid file names and file name substitution. The save menu is executed using the current default values and the standard DISCUS prompt returns.
If the 'save' command is used without parameter, the save menu is called. At this menu you can define the output format of the structure file, its name, which information and which atoms to include.

commands

Valid commands at this level are

```
@        ! Execute a macro file (see main help)
=        ! Assigns the value to a variable (see main help)
asym     ! Shows asymmetric unit
chem     ! Shows the atoms present in the crystal
continue! Continue a stopped macro (see main help level)
des      ! Deselects atoms
echo     ! Echo a string (see main help)
eval     ! Evaluates an expression for interactive check (see main help)
exit     ! Terminates 'save'
file     ! Defines the file name to which the structure is written
format   ! Defines the format of the structure file
help     ! Gives on line help for saving the structure (see main help)
omit     ! Defines which information to omit from the file
outf     ! Defines the name of the output file
prop     ! Selects the atom properties, which will be checked
reset    ! Reset and clear all definitions
run      ! Starts the saving of the structure
sel      ! Selects atoms to be included in the structure file
show     ! Shows the current parameters
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
wait     ! Waits for user input (see main help)
write    ! Defines which information to write to the file
```

asym

```
asym
```

Shows the content of the asymmetric unit. The names of those atoms, a number that is used as index for its scattering type, their position and temperature coefficient are listed. The number that is listed, is the number that refers to the scattering curve of that atom. It is contained in the variable m[<index>]. If a cell was read, all atoms are considered to be different, even if they are chemically identical and have the same temperature coefficient. If a whole structure was read, all atoms that are in the unit cell $0 <= xyz < 1$, are chemically unique and have a different temperature coefficient are included in the asymmetric unit.

chem

```
chem
```

Displays the type of atoms present in the crystal. For each type of atom, its scattering curve number, its name and its temperature factor are listed. Warning, even, if all atoms of a particular type have been deleted, its scattering type will remain in the list. This list could therefore include more types of atoms than are actually present in the crystal.

des

```
des { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

Deselects choices made by ==> 'sel' . Possible values for the parameter are mutually exclusively:

```
"all"      all atoms of the crystal are deselected.
<name>     all the atoms called <name> of the crystal are deselected.
           This includes symmetrically not equivalent atoms.
<number>   all atoms of the crystal that are of scattering type <number>
           are deselected.
```

More than one atom may be deselected at once.

outfile

```
outfile <filename>
```

Sets the name of the output file to which the structure is written. If the filename starts with the string 'internal', the file is not written to disk but stored internally.

prop

```
prop {"ignore"|"present"|"absent"} , <property> [, <property> ...]
```

Defines which properties an atom must have to be included in the plot.
"ignore" The properties listed will be ignored. An atom will be included in the saved file regardless, whether it has the properties or not, "present" An atom must have the properties listed, to be included in the file "absent" An atom is not allowed to have the properties listed.

The properties can be one of: "all" Apply the rule to all properties. "normal" The rules apply if an atom is a normal atom (not a void) "molecule" The rules apply if an atom is part of a molecule "domain" The rules apply if an atom is part of a domain "outside" The rules apply if an atom is outside of the crystal. This status is set if the atom has been cut off by a ==> 'boundary' command. "external" The rules apply if an atom is close to an external surface, either on the inside or on the outside. The flag is set by the ==> 'boundary' command. Individual distances to the boundary can be set within the ==> 'surface' menu through the ==> 'surface/fuzzy' command. "internal" The rules apply if an atom is close to an internal surface, either within the host or within the guest structure. The flag is set by placing a guest structure into the crystal through the ==> 'domain' menu. Individual distances to the boundary can be set within the ==> 'domain' menu through the ==> 'domain/fuzzy' command.

format

```
format {"keyword" | "nokeyword" | "nexus" }
format {"format:keyword" | "format:nokeyword" | "format:nexus" }
```

The "keyword" controlled format uses keywords analogous to the commands of DISCUS to describe the information in the structure file. See the help on "data" at the main help level for a complete listing of all keywords. A flexible number of keywords can be included at the header of the structure file ==> 'write' and 'omit'. The file is a simple ASCII text file
The "nexus" file format is a binary HDF5 file. It follows the NeXus style.
The non-keyword controlled format writes a simplified file that contains only the following information:

```
Line 1:    Title up to 80 characters
line 2:    Space group symbol [,origin choice number]
line 3:    Lattice constants a,b,c,alpha,beta,gamma
```

and for each atom a line with:

```
        Name(4 characters) x  y  z  isotropic B
```

incl

```
incl { <start>,<end> | "all" }
```

The unit cell transformation includes all atoms numbered <start> to <end> inclusively. All other atoms are ignored. If, instead of explicit numbers, the parameter "all" is given, the save process will include all atoms of the crystal. This holds even, if at a later time you include further atoms in the crystal. Thus, you can define a setup for the save process early in a lengthy macro, then modify the crystal and just run the save process later on. In addition you can define the atoms that are affected by the save process with the ==>'sele' and 'dese' commands.

omit

```
omit { "all" | "generator" | "ncell" | "symmetry" | "molecule" |
        "object" | "scat" | "adp" | "occ" }
```

The keyword controlled format includes the mandatory lines : title spcgrp cell atoms
All other keywords are optional.
"all" All optional keywords are omitted from the structure file.
"scat" The list of all atom names is omitted from the structure file, while the status of the other keyword is not changed.
"adp" The list of all Atomic displacement parameters is omitted from the structure file, while the status of the other keyword is not changed.
"occ" The list of all occupancy parameters is omitted from the structure file, while the status of the other keyword is not changed.
"generator" The additional generators are omitted from the structure file, while the status of the other keyword is not changed.
"ncell" The number of unit cells and the number of atoms in each unit cell are omitted from the structure file, while the status of the other keyword is not changed.
"symmetry" The additional symmetry operators are omitted from the structure file, while the status of the other keyword is not changed.
"molecule" Although the crystal contains molecules their type and content is not written to the end of the structure file. WARNING: All information about molecules is lost!
"object" Although the crystal contains objects their type and content is not written to the end of the structure file. WARNING: All information about objects is lost!

reset

Resets the save menu to program start conditions.

run

```
run
```

Starts the writing to the output file.

sel

```
sel { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

This command serves to select those atoms that will be included in the saving to the output file.
Possible values for the first mandatory parameter are mutually exclusively:

```
"all"     all atoms of the crystal are included.
```

This includes the "voids" in the structure, which are stored as scattering curve number zero.

```
<name>    all the atoms called <name> of the crystal are included.
          This includes symmetrically not equivalent atoms.
<number>  all atoms of the crystal that are of scattering type <number>
          are included.
```

More than one atom may be selected at once.
The selection made stay valid until explicitly deselected!

show

```
show
```

Shows the current parameters of the save menu.

write

```
write { "all" | "generator" | "ncell" | "symmetry" | "molecule" |
        "object" | "domain" | "scat" | "adp" | "occ" }
```

The keyword controlled format includes the mandatory and automatically included lines : title
spcgrp cell atoms
All other keywords are optional.
"all" All optional keywords are written to the structure file.
"scat" The list of all atom names is written to the structure file.
"adp" The list of all atomic displacement parameters is written to the structure file.
"occ" The list of all occupancy parameters is written to the structure file.
"generator" If additional generators were read from the input file they will be written to the
structure file as well.
"ncell" The number of unit cells and the number of atoms in each unit cell are written to the
structure file. The number of unit cells corresponds to the parameter used for the 'read/cell'
command. These numbers are needed for a fast association between atom number and its
unit cell inside a large crystal. Warning!!!! If you purged "voids" from the crystal, the number
of atoms per unit cell is no longer constant throughout the crystal and no fixed association
exists between atom number and the corresponding unit cell. Use of the purge command is
discouraged !!!!!!
"symmetry" If additional symmetry operators were read from the input file they will be written
to the structure file as well.
"molecule" If the crystal contains molecules their type and content is written to the end of the
structure file.
"object" If the crystal contains objects their type, content, and density is written to the end of
the structure file.
"domain" If the crystal contains domains their type, character, and content is written to the end
of the structure file.

A.54   seed

```
seed [ <value> ]
```

Reinitializes the pseudo random generator. The seed passed to the random generator is -
abs(nint(value)). If the <value> is omitted, the random generator will be passed the num-
ber of hundredth of seconds passed since midnight, essentially initializing the sequence at a
unknown fairly random point.

A.55 shear

shear

This command enters the deformation menu. The commands in this subsection allow you to apply a deformation tensor to an object or to a molecule.

You can define the deformation tensor by providing: - the matrix - a deformation plane and the deformation vector parallel to this plane. - the eigenvectors and eigenvalues of the deformation tensor

Further help is available on:

commands

```
calc    ! Calculate the deformation for a single vector
des     ! Deselect the atom types
eigen   ! Define the deformation through it's eigenvectors
hkl     ! Define the deformation plane
incl    ! Include a range of atoms
matrix  ! Define the deformation tensor directly
mincl   ! Include a range of molecules
mdes    ! Deselect the molecule types
msel    ! Select molecule types
mode    ! Set the input mode (matrix, plane, eigenvectors)
oincl   ! Include a range of objects
origin  ! Define the origin for the deformation
osel    ! Select the object types
odes    ! Deselect the object types
reset   ! Reset and clear all definitions
run     ! Perform the actual deformation
sel     ! Select the atom types
show    ! Show the current settings
uvw     ! Define the deformation plane
vector  ! Define the deformation vector
```

calc

calc <x>,<y>,<z> {,<flag>}

Calculates the effect of the current shear on vector $<x>,<y>,<z>$. The flag is allowed to be either "d" or "r", specifying a direct or reciprocal vector. The result is displayed on screen and stored in the result variable "res[]". 3 values are stored, $<x'>,<y'>,$and $<z'>$.

des

des { "all" | <name> | <number> } [, { <name> | <number> } ...]

Deselects choices made by ==> 'sel' . Possible values for the parameter are mutually exclusively:

```
"all"    all atoms of the crystal are deselected.
<name>   all the atoms called <name> of the crystal are deselected.
         This includes symmetrically not equivalent atoms.
<number> all atoms of the crystal that are of scattering type <number>
         are deselected.
```

More than one atom may be deselected at once.

eigen

`eigen <nr>, <x>,<y>,<z> {,<value>}`

Define the direction of the eigenvector number <nr>. <x>,<y>,<z> are the coordinates of the eigenvector in direct space.

If you give the last parameter, this is interpreted as the corresponding eigenvalue. If the last parameter is omitted, DISCUS will calculated the length of the eigenvector and take this as the eigenvalue.

In both cases, the eigenvector is normalized to unit length.

The ==> 'show' command displays the eigenvectors, the determinant of the eigenvector matrix, and the angles between the three eigenvectors. This allows you to verify that all three vectors are normal to each other.

hkl

`hkl <h>,<k>,<l> [,<length>]`

Defines the deformation by providing a plane ==> 'hkl', 'uvw' and a deformation vector ==> 'vector'.

'hkl' is the normal to the plane in reciprocal space coordinates.

The object experiences a shear parallel to the plane. The magnitude of the shift is defined by the deformation vector, that shifts the point at the end of 'uvw' from the ==> 'origin' by its value =='vector'. If the optional last parameter is given, the length of the normal ==> 'uvw' to the plane is normalized to this length, otherwise the normal is taken as is. This parameter helps you e.q. in a monoclinic space, to apply a shift parallel to the (001) plane by specifying the ==> 'hkl' 0,0,1 and then giving a distance from this plane at which the shear vector ==> 'vector' is to act, e.g. the d-spacing of the (001) planes. All points are shifted by an amount parallel to the deformation vector that is proportional to their distance from the plane. Points at a distance equal to the length of 'uvw' are shifted by the vector, those twice as far away are shifted twice as much. Points on opposite sides of the plane are shifted in opposite directions.

incl

`incl {<start>,<end> | "all" | "envi" }`

The shear operation includes all atoms numbered <start> to <end> inclusively. All other atoms are ignored. If, instead of explicit numbers, the parameter "all" is given, the shear operation will include all atoms of the crystal. This holds even, if at a later time you include further atoms in the crystal. Thus, you can define a setup for the shear operations early in a lengthy macro, then modify the crystal and just run the shear operation later on. If the parameter "envi" is specified, the shear operation will include all atoms in the current environment that was located by the last ==> 'find env' command. In addition you can define the atoms that are affected by the shear operation with the ==>'sele' and 'dese' commands.

matrix

`matrix <11>,<12>,<13>,<21>,<22>,<23>,<31>,<32>,<33>`

This command defines explicitly the deformation tensor. All nine elements must be given.

mincl

`mincl {<start>,<end> | "all" }`

The shear operation includes all molecules numbered <start> to <end> inclusively. All other molecules are ignored. If, instead of explicit numbers, the parameter "all" is given, the shear operation will include all molecules of the crystal. This holds even, if at a later time you include further molecules in the crystal. Thus, you can define a setup for the shear operations early in a lengthy macro, then modify the crystal and just run the shear operation later on. In addition you can define the molecules that are affected by the shear operation with the ==>'msele' and 'mdese' commands.

mdes

`mdes { "all" | <name> | <number> } [ , { <name> | <number> } ...]`

Deselects choices made by ==> 'msel' . Possible values for the parameter are mutually exclusively:

```
"all"    all molecules of the crystal are deselected.
<number> all molecules of the crystal that are of type <number>
         are deselected.
```

More than one molecule type may be deselected at once.

msel

`msel { "all" | <number> } [ , <number> ...]`

This command serves to select those molecules that will be modified by the symmetry operation.
Defines which molecules are included in symmetry operation. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"    all molecules of the crystal are included.
         This includes the empty molecules in the structure, which are
         stored as scattering curve number zero.
<number> all molecules of the crystal that are of type <number>
         are included.
```

More than one atom may be selected at once.
The selection of molecules by 'msel', objects by 'osel', and atoms by 'sel' are mutually exclusive.
The symmetry operation uses either atoms,objects or molecules.

mode

```
mode {"eigen"|"matrix"|"plane"|"rmatrix"}
```

You can tell DISCUS which input you intend to use for the definition of the deformation tensor. The commands ==> 'uvw', 'hkl' 'vector', 'matrix', rmatrix', and 'eigen' will automatically set this flag as well. The last of these commands type before the final ==> 'run' determines the actual input mode that is assumed.

odes

```
odes { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

Deselects choices made by ==> 'osel' . Possible values for the parameter are mutually exclusively:

```
"all"     all objects of the crystal are deselected.
<number>  all objects of the crystal that are of type <number>
          are deselected.
```

More than one objects type may be deselected at once.

oincl

```
oincl {<start>,<end> | "all" }
```

The shear operation includes all objects numbered <start> to <end> inclusively. All other objects are ignored. If, instead of explicit numbers, the parameter "all" is given, the shear operation will include all objects of the crystal. This holds even, if at a later time you include further objects in the crystal. Thus, you can define a setup for the shear operations early in a lengthy macro, then modify the crystal and just run the shear operation later on. In addition you can define the objects that are affected by the shear operation with the ==>'osele' and 'odese' commands.

origin

```
origin <x>,<y>,<z> [,{"crystal"|"molecule"]
```

This commands sets the origin to which the shear operation refers to.
If you selected atoms by ==> 'sel', the last optional parameter will always default to an origin with respect to the crystal base system. The specified origin is subtracted from each atom position prior to the shear operation.
If you selected molecules ==> 'msel' or objects ==> 'osel', the last optional parameter will be interpreted. If it is set to "molecule", the origin is considered to be at vector <x>,<y>,<z> relative to the origin of each individual molecule or object. All distances within the molecule or object are sheared accordingly. If it is set to "crystal", the origin is considered to be at vector <x>,<y>,<z> relative to the crystal origin. The shear operation is applied to the origin of each molecule or object. The molecules or objects are now considered to be rigid units. Thus the shear operation will shift all atoms or objects with the shear that is applied at their respective origin but will not apply a shear to the internal distances.

osel

```
osel { "all" | <number> } [ , <number> ...]
```

This command serves to select those objects that will be modified by the symmetry operation. Defines which objects are included in symmetry operation. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"     all objects of the crystal are included.
          This includes the empty objects in the structure, which are
          stored as scattering curve number zero.
<number>  all objects of the crystal that are of type <number>
          are included.
```

More than one object may be selected at once.
The selection of objects by 'osel','msel' and atoms by 'sel' are mutually exclusive. The symmetry operation uses either atoms or molecules, or objects.

reset

```
reset
```

Reset all shear definitions back to the default values at system start.

run

```
run
```

Performs the actual deformation operation on the selected atom, molecule or object types.

sel

```
sel { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

This command executes serves to select those atoms that will be modified by the symmetry operation.
Defines which atoms are included in symmetry operation. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"     all atoms of the crystal are included.
```

This includes the "voids" in the structure, which are stored as scattering curve number zero.

```
<name>    all the atoms called <name> of the crystal are included.
          This includes symmetrically not equivalent atoms.
<number>  all atoms of the crystal that are of scattering type <number>
          are included.
```

More than one atom may be selected at once.
The selection made stay valid until explicitly deselected!

show

`show`

Shows the current settings.

uvw

`uvw <u>,<v>,<w> [,<length>]`

Defines the deformation by providing a plane ==> 'hkl', 'uvw' and a deformation vector ==> 'vector'.
'uvw' is the normal to the plane in direct space coordinates.
The object experiences a shear parallel to the plane. The magnitude of the shift is defined by the deformation vector, that shifts the point at the end of 'uvw' from the ==> 'origin' by its value =='vector'. If the optional last parameter is given, the length of the normal ==> 'uvw' to the plane is normalized to this length, otherwise the normal is taken as is. This parameter helps you e.q. in a monoclinic space, to apply a shift parallel to the (001) plane by specifying the ==> 'hkl' 0,0,1 and then giving a distance from this plane at which the shear vector ==> 'vector' is to act, e.g. the d-spacing of the (001) planes. All points are shifted by an amount parallel to the deformation vector that is proportional to their distance from the plane. Points at a distance equal to the length of 'uvw' are shifted by the vector, # those twice as far away are shifted twice as much. Points on opposite sides of the plane are shifted in opposite directions.

vector

`vector <x>,<y>,<z>`

Defines the deformation by providing a plane ==> 'hkl', 'uvw' and a deformation vector ==> 'vector'.
The object experiences a shear parallel to the plane. The magnitude of the shift is defined by the deformation vector, that shifts the point at the end of 'uvw' from the ==> 'origin' by its value =='vector'. All points are shifted by an amount parallel to the deformation vector that is proportional to their distance from the plane. Points at a distance equal to the length of 'uvw' are shifted by the vector, # those twice as far away are shifted twice as much. Points on opposite sides of the plane are shifted in opposite directions.

A.56   show

`show <subcommand>,..`

This command allows to obtain various information about the model crystal, DISCUS settings etc.
Valid subcommands are :

adp

```
show adp:all
show adp:uij
show adp:prin
```

Show the atomic displacment parameters. With option "all" both the Uij and the principal displacemnt vectors are shown, the other two options reduce output to the specified value. For Uij that correspond to an isotropic displacemnt, the output is reduced, full output is available with the optional parameter:

```
show adp:all, "style:long"
```

The optional parameter can be used for any of the three "adp:" versions Default is "style:short".

asym

```
show "asym"
```

Shows the content of the asymmetric unit. The names of those atoms, a number that is used as index for its scattering type, their position and temperature coefficient are listed. The number that is listed, is the number that refers to the scattering curve of that atom. It is contained in the variable m[<index>]. If a cell was read, all atoms are considered to be different, even if they are chemically identical and have the same temperature coefficient. If a whole structure was read, all atoms that are in the unit cell $0 <= xyz < 1$, are chemically unique and have a different temperature coefficient are included in the asymmetric unit.

atom

```
show "atom",<number> [,<end>]
show "atom","all"
show "atom","last"
```

The type, position and isotropic B of atom number <number> are shown. If the optional second parameter <end> is given, all atoms in the range <number> to <end> are shown. If the second parameter is "all", all atoms in the crystal are shown. WARNING, this could last a while :-) If the second parameter is "last", only the last atom of the crystal is shown. This is identical to setting the second parameter to "n[1]", which contains the number of atoms in the crystal.
If <number> is higher than the number of atoms in the crystal, nothing is shown and no error message is displayed. If <end> is higher than the number of atoms in the crystal, <end> is reset to the last atom in the crystal.

bval

```
show "bval",<a1>,<a2>
```

This command shows the bond valence parameters r0 and b for the specified atom pair <a1> and <a2>. The parameters are taken from a table prepared by I.D. Brown, McMaster University, ON, Canada.

chem

```
show "chem"
```

Displays the type of atoms present in the crystal. For each type of atom, its scattering curve number, its name and its temperature factor are listed. Warning, even, if all atoms of a particular type have been deleted, its scattering type will remain in the list. This list could therefore include more types of atoms than are actually present in the crystal. A short information about the content of the asymmetric unit is printed.

cdim

```
show "cdim"
```

Prints the current crystal dimensions. The smallest and largest coordinates of any atom along x,y and z are printed.

config

```
show "config"
```

This command lists the current configuration of DISCUS such as the maximum number of atoms.

connectivity

```
show "connectivity", <atom_no.>, <def.no.>
```

Show the list of neighbors around atom no. <atom_no.>, which have been found using definition number <def.no.> for the scattering type of atom no. <atom_no.> The list of atom numbers is copied into the result variable "res".

envi

```
show "environment"
```

All atoms in the current environment ==> 'find env' are listed.

error

```
show "error"
```

The error status of DISCUS is displayed.

menv

```
show "menvironment" [,"full"]
```

All molecules in the current environment ==> 'find env' are listed. If the optional "full" parameter is given, all atoms within each molecule are printed.

---

mass

```
show mass
```

Calculates and displays the current crystal mass in multiples of atomic units u.
The results are stored into the result variable as: res[0] = 3 res[1] = mass res[2] = mass / number_of_atoms res[3] = number_of_atoms

metric

```
show metric
```

Displays the current unit cell dimensions, the unit cell volume, the metric tensor and the corresponding reciprocal space values. This command is for display only. You may change the direct space unit cell dimensions with the assignment lat[<i>]=<value>. <i>=1 through 3 are a,b,c and <i>=4 through 6 are alpha, beta, gamma.

molecule

```
show "molecule",{"all" | "last"}
show "molecule",<number> [,<end>]
```

For each molecule in the range <number> to <end> the type, number of the atom in the crystal, position and isotropic B of all atoms in the molecule are shown.
If the second parameter is "all", all molecules in the crystal are shown. WARNING, this could last a while :-) If the second parameter is "last", only the last molecules of the crystal is shown.
If <number> is higher than the number of molecules in the crystal, nothing is shown and no error message is displayed. If <end> is higher than the number of molecules in the crystal, <end> is reset to the last atom in the crystal.

object

```
show "object",{"all" | "last"}
show "object",<number> [,<end>]
```

For each object in the range <number> to <end> the type, number of the atom in the crystal, position and isotropic B of all atoms in the object are shown.
If the second parameter is "all", all objects in the crystal are shown. WARNING, this could last a while :-) If the second parameter is "last", only the last objects of the crystal is shown.
If <number> is higher than the number of objects in the crystal, nothing is shown and no error message is displayed. If <end> is higher than the number of objects in the crystal, <end> is reset to the last atom in the crystal.

result

```
show "result"
```

The result array is shown on screen. The number of values shown depends on the last calculation performed.
Further information about the crystal, its composition etc can be obtained by evaluating the corresponding ==> variable.

scat

```
show "scat", {<name> | <number> } [, {<name> | <number> }...]
```

Shows the scattering curve of the selected atom types, which may be given as the scattering number of as the element name.

The values for the scattering curve are stored into the res[] variable in the order: a1, a2, a3, a4, b1, b2, b3, b4, c

symmetry

```
show "symmetry" [,{"full" | "symbol" | "xzy" | "matrix" }
```

Shows all symmetry operations for the current space group.

```
"full"
```

The default setting, produces a full listing. This includes: the symmetry matrix as (3x4) matrix the symbol as printed in the International Tables the resulting atom positions xyz, as printed in the International Tables

For each symmetry operation DISCUS lists the number in square brackets, and for centered space groups the number within each block that corresponds to one centering vector in round brackets. With the setting ==> set generator,symmetry the sequence corresponds to that listed in the Tables.

```
"symbol"
```

This produces a short list that consists of the symbol as used in the International Tables.

```
"xyz"
```

This produces a short list that consists of the symbolic atom positions xyz, as used in the International Tables.

```
"matrix"
```

This produces a list that consists of the (3x4) symmetry matrices.

A.57   spacegroup

```
spacegroup {<spacegroup_name>|<spacegroup_nummer>} [, <origin_choice>}
origin:<origin_choice>
setting:<setting>
```

Sets the space group for the current structure. Be careful with this command, as DISCUS will blindly set the space group! DISCUS does not check if the symmetry does match! A consistency check with the lattice parameter is the only check that is performed. The command is mostly useful when you want to lower the symmetry to a subgroup.

A.58   stack

stack

Switches to the stacking fault sub menu of DISCUS. At this sub menu you create a crystal containing stacking faults. Several types of layers can be stacked on top of each other. At present these layers are read from file. In future releases you will be able to call a macro that creates the layer. A layer could be a 2-dimensional layer of atoms or unit cells of any thickness or a 1-dimensional row of atoms or unit cells. The translation vectors from one layer type to the next can be defined and need not be exact values but can be Gaussian distributed. A correlation matrix defines the probabilities of layer sequences. These probabilities only include first neighbor interactions.

Alternatively to the stacking faults ruled by the correlations, you can include random stacking faults. These stacking faults choose at random the next layer type and the translation vector.

You can create the crystal following two alternate choices:

1) The origin of the current layer is the sum of the previous origin and the translation vector

2) The above origin is taken modulo the modulus vectors.

By using the first option, the crystal continuously grows in one direction as given by the translation vector(s). By using the second option the origins are restricted to a finite range around the average translation vector, which results in a zig-zag shaped crystal.

The menu includes a new approach to calculate the Fourier transform that saves considerable computational time.

If stack successfully placed atoms into the crystal with the 'run' command, the crystal is no longer periodic. Thus periodic boundary conditions and the "chem" quick mode are off. They can be turned back on after explicit 'set crystal' command in the chemistry menu.

Further help topics are

commands

Valid commands at this level are

```
@        ! Execute a macro file (see main help)
=        ! Assigns the value to a variable (see main help)
aver     ! Sets the intended average translation
ccol     ! Reads a single column of the correlation matrix
cele     ! Reads a single element of the correlation matrix
continue! Continue a stopped macro (see main help level)
create   ! Creates the list of layer origins
crow     ! Reads a single row of the correlation matrix
dist     ! Defines the distribution of layer types
echo     ! Echo a string (see main help)
eval     ! Evaluates an expression for interactive check (see main help)
exit     ! Terminates 'stack' (see main help)
```

rst ! Deter mines the first layer type

```
four     ! Calculates the Fourier transform in quick mode
help     ! Gives on line help for symmetry operations (see main help)
layer    ! Reads the name of a file that contains a layer
modulus  ! Reads the modulus vector of the translation
number   ! Reads the number of layers to be created
random   ! Sets values for random stacking faults
```

```
reset   ! Resets the number of layers to zero
rotate  ! Set values for rotational disorder of the layers
reset   ! Reset and clear all definitions
run     ! Decorates the list of origins with atoms to create the crystal
set     ! Sets parameters
show    ! Shows the current settings
sigma   ! Reads the sigmas for the translation vectors
stop    ! Stops execution of a macro (see main help level)
system  ! Executes operating system command (see main help)
trans   ! Reads the translation vectors
wait    ! Waits for user input (see main help)
```

aver

`aver <u>,<v>,<w>`

Sets the intended average translation. If the modulus is switched on ==> 'set modulus', and the mode of the translation ==> 'set trans' is set to "fixed", this vector is used.

ccol

`ccol <nr>,<c1nr>,<c2nr>,...,<cmdtypenr>`

Read a single column <nr> of the correlation matrix. Following the index <nr>, as many parameters as there are layer types must follow. The sum of any column need not be equal to one.

cele

`cele <i>,<j>,<value>`

Reads the value <value> that is stored in element <i>,<j> of the correlation matrix.

create

`create`

Creates the list of layer origins. No changes are done to the actual structure, this is performed by the command ==> 'run'.
The command copies the list of layer origins into an internal file with fixed name "internal.stacklist.stru". The layer types are saved as atoms with names "L001", "L002" etc. The coordinates correspond to the layer positions.
A simplified list of origin types called "internal.stacksimple.list" contains the atom types "L001", etc at reduced origins starting at 0,0,0 and incrementing with [0,0,1]. This list facilitates correlation analysis.

crow

`crow <nr>,<cnr>1,<cnr>2,...,<cnrmdtype>`

Read a single row <nr> of the correlation matrix. Following the index <nr>, as many parameters as there are layer types must follow. The sum of any row need not be equal to one, as long as the sum of all rows is identical.

---

dist

```
dist { "matrix" | "file", <inputfile> | "list", <inputfile> }
```

This commands allows you to define how the sequence of layer types is determined. The parameter "matrix" forces DISCUS to determine the type of each layer from the correlation matrix ==> 'crow', 'ccol'. Only nearest neighbor correlations are realized.

The parameter "file" forces DISCUS to read a standard structure file from <inputfile>. The atom types found in this file are interpreted as to correspond to the respective layer type. Make sure this is a one-dimensional structure and that you have written this file using the ==> 'save' options to write the explicit list of scattering types and atomic displacement parameters. You can use the MonteCarlo section of DISCUS ==> 'mmc' to create the list of layer types, which allows you to include nearest as well as next nearest etc neighbor correlations. Presently 'stack' ignores the atom positions found in the input file. The layer origins are calculated using the translations ==> 'tran'.

The parameter "list" acts very similarly to "file". The only difference is that the atom positions are interpreted as layer positions as well.

DISCUS does not check whether the atoms in <inputfile> form a 1-dimensional list of atoms, or whether this 1-dimensional line is not parallel to the layers! This can actually be used to distribute equally sized and oriented domains throughout the crystal. In combination with the stacking fault fourier option, very fast Fourier transformations of molecular crystals can be calculated.

first

```
first {"random" | <type_number>}
```

Determines the type of the first layer type. with the parameter "random", the first layer type is determined by a weighted random choice calculated from the correlation matrix. This is the default behavior. Alternatively a <type number> fixes the first layer type to this particular type.

four

```
four
```

A crystal with stacking faults can be created in the following step wise manner:

```
1. Create the distribution of layer origins.
2. Convolute each origin with the corresponding layer.
```

At this point, the convolution theorem saves a lot of computation time since the Fourier transform of a convolution is equal to the product of the individual Fourier transforms. The Fourier transform approach realized in the stacking fault menu operates by the following steps:

```
0. The current parameters of the ==> 'fourier' menu are used.
1. For each layer type calculate the Fourier transform of the
   list of origins.
   Calculate the Fourier transform of the actual atoms in the layer.
   Multiply the resulting complex amplitudes and store the results
   in temporary files.
2. Add the complex amplitudes from all layer types and store the
   result in internal memory.
```

The user can then write any desired output value using the ==>'output' sub menu.

Note:

In order to calculate the Fourier transform of each layer type the atoms in the corresponding file are read into the internal structure array. A previous structure is lost! If necessary save this structure to file by the ==> 'save' command.

Note:

DISCUS does not compare the space group or lattice constants in the different layer files. You must make sure that these are identical.

layer

```
layer <filename>
```

Defines a new layer type. The file <filename> must be a standard DISCUS input file that contains a list of atoms. No restrictions apply to the position and types of atoms in the layer. You have to create the layer beforehand using the tools of DISCUS.

modulus

```
modulus <x1>,<y1>,<z1>,  <x2>,<y2>,<z2>
```

Sets the two modulo vectors that are used to restrict the origins to a finite volume around the average translation vector. The plane defined by these two vectors need not to be normal to the average translation.

```
==> 'set mod'
==> 'set tran'
==> 'aver'
```

You can create the crystal following two alternate choices:

```
1) The origin of the current layer is the sum of the previous origin
   and the translation vector
2) The above origin is taken modulo the modulus vector.
```

By using the first option, the crystal continuously grows in one direction as given by the translation vector(s). By using the second option one or two coordinates can be constrained to a finite range, which results in a zig-zag shaped crystal. If any of the parameters is not equal to zero, the corresponding coordinate of the origin is taken modulo this parameter.
DISCUS does NOT check whether the modulo vectors are translation vectors of the current space group!

number

```
number <n>
```

Reads the number of layers to be created.

---

random

```
random prob,<value>
random offset,< u>,< v>,< w>
random sigma ,<su>,<sv>,<sw>
```

This command sets values for the random stacking faults. The program selects the next layer type by weighted random choice. The weights are chosen according to the overall probabilities for each layer type. The next layer is then shifted by the corresponding translation vector between the resulting neighbors. In addition a random fraction of the two modulo vectors is added, which results in a random distribution of the next layer origin.

```
random prob,<value>
```

Sets the probability for random stacking faults. <value> must be between zero and one. Set the probability to zero to exclude random stacking faults.

```
random offset,< u>,< v>,< w>
```

If specified this vector is added to the translation vector. You can use this offset to force an increased layer separation for random faults compared to regular stacking faults.

```
random sigma ,<su>,<sv>,<sw>
```

Specifies the sigma's for a Gaussian distribution applied to the components of the offset.

reset

```
reset
```

Sets the number of layers and layer types back to zero.

rotate

```
rotate axis,  "normal",<u>,<v>,<w> [ "d" | "r" ]
rotate axis,  "mod1",  <u>,<v>,<w> [ "d" | "r" ]
rotate axis,  "mod2",  <u>,<v>,<w> [ "d" | "r" ]
rotate mode, [ "all" | "fault" ]
rotate sigma, "normal",<snormal>
rotate sigma, "mod1",  <smod1>
rotate sigma, "mod2",  <smod2>
rotate status,  [ "on" | "off" ]
```

This command sets values for rotational disorder of the stacking faults. The layers are rotated around three axes, which are defined by the 'axis' command. The angle of rotation is Gaussian distributed with mean value zero and sigma defined by the 'sigma' command.

```
rotate axis,  "normal",<u>,<v>,<w> [ "d" | "r" ]
rotate axis,  "mod1",  <u>,<v>,<w> [ "d" | "r" ]
rotate axis,  "mod2",  <u>,<v>,<w> [ "d" | "r" ]
```

The parameter "axis" defines the three axes around which you can rotate the layers. The axis can be given in real space coordinates "d" or reciprocal space coordinates "r". Default is real space. DISCUS does not make any assumptions about the three axes. The layer is rotated first around the "normal" axis, then around "mod1" and finally around "mod2". It is suggested to define "normal" as the axis perpendicular to the layers and "mod1" and "mod2" within the plane of the layers.

Attention!! If you tilt the layer around "mod1" or "mod2" the outer sections of the layer might overlap with the previous layer!

```
rotate mode, [ "all" | "fault" ]
```

The rotational disorder can affect all layers individually, regardless whether the previous layer was of the same type as the current layer or not. Alternatively, a new rotation angle is computed only at a stacking fault, i.e. if the current layer type is different than the previous layer type. All following layers of identical type up to the next stacking fault are rotated by the same angle.

```
rotate sigma,  "normal",<snormal>
rotate sigma,  "mod1",  <smod1>
rotate sigma,  "mod2",  <smod2>
```

The rotation is Gaussian distributed with individual sigma for each axis. The mean value of the distribution is zero. Each new angle is calculated individually from a Gaussian distribution irrespective of the previous value. This forces the average layer orientations to remain constant throughout the crystal. A sigma of zero fixes the rotations around the respective axis to zero.

```
rotate status,  [ "on" | "off" ]
```

The rotational disorder is only effective if its status is switched "on".

reset

```
reset
```

Reset all stack definitions back to the default values at system start.

run

```
run
```

Decorates the origins with the corresponding atom distribution. Any previously existing structure is lost.

set

```
set "aver",<value>
set "modulus" , { "off" | "on" }
set "trans" , { "calculate" | "fixed" }
set "aver:"<value>
set {"modulus:off" | "modulus:on" }
set {"trans:"calculate" | "trans:fixed" }
set {atoms:off} | "atoms:on" | "atoms:strict" }
set "aver",<value>
set "aver:"<value>
```

If the value of <value> is not equal to zero, the average <F> is subtracted from the actual structure factor F. Otherwise the average <F> is calculated using the whole crystal and subtracted from the actual F.

```
set "modulus" , { "off" | "on" }
set {"modulus:off" | "modulus:on" }
```

If the modulus is set "off", each origin is calculated by adding the corresponding translation vector to the previous origin. This way the crystal grows along the average translation vector. The resulting average translation vector is the sum of all translation vectors weighted by the corresponding probability as stored in the correlation matrix. If the modulus is set "on", each origin is calculated as above. The origin is then expressed as multiples of the two modulo vectors and the average translation. If any of the two multipliers corresponding to the two modulo vectors is larger than |1|, the modulo vector is subtracted from the origin. This has the effect that all origins are no further than the modulo vectors from the average translation. Even though all translation vectors might point in one direction, you can restrict the crystal to any arbitrary direction by defining the average vector ==> 'set trans', ==> 'aver'.

```
set "trans" , { "calculate" | "fixed" }
set {"trans:"calculate" | "trans:fixed" }
```

You have two choices to restrict the crystal growth once the modulus is set on. The average translation can be fixed to the value given by ==> 'aver' by setting the parameter of the 'set trans' command to "fixed". Alternatively the average translation is computed using the array of the translation vectors and the correlation matrix.

```
set {atoms:off} | "atoms:on" | "atoms:strict" }
```

In adition to shifting the whole layer according to the modulus vectors, DISCUS allows to shift individual atoms at the edge of the crystal. DISCUS calculates the extend of the crystal. If individual atoms are shifted outside the original dimensions, periodic boundary conditions are applied for "set atoms:on". This will give a smoother edge of the crystal. Due to rounding issues, a more stringent reshape operation is applied with "set atoms:strict". In this case DISCUS will reduce the overall dimensions of all layers in the final crystal by 2 unit cells, one at the lower, one at the upper edge. Atoms in these outer shell are shifted using periodic boundary conditions or removed to restrict the crystal edges to the smaller number of unit cells. This ensures a smooth edge.

show

```
show
```

Shows the current settings.

sigma

```
sigma <i>,<j>,<sx>,<sy>,<sz>
```

Read the sigmas of the translation vector between layer type <i> and <j>. The current origin of any layer is calculated by adding the translation vector ==> 'trans' between layer types <i> and <j> to the previous origin. If any sigma is non-zero, a Gaussian distributed value with mean value of zero is added to the corresponding vector component.

---

trans

```
trans <i>,<j>,<tx>,<ty>,<tz>
```

Reads the components of the translation vector between layers <i> and <j>. The current origin of any layer is calculated by adding the translation vector between layer types <i> and <j> to the previous origin.

A.59 storage

Menu to display and manage internally stored crystal structures
Further help is available on:

commands

```
show        ! Show list of stored structures
reset       ! Clear list of stored structures
remove      ! Clear list of stored structures
exit        ! Leaves the storage menu
```

show

```
show
show "all"
show <name>
   display:{"short" | "crystal" | "connect" | "full"}
```

Displays the complete list (no parameters or "all") or an individual structure.
The optional parameter "display:" determines the amount of information that is displayed:
"short" (default) Show the file names only "crystal" Additionally show number of atoms, atom types and molecule info This is stored into the result variable at entries 1 to 4. "connect" Additionally show the internal connection in the binary storage tree. This is really just for debug purposes "full" Combines "crystal" and "connect"

reset

```
reset {<name>}
```

Clears the complete internal list or the single structure named <name> Commands 'reset' and 'remove' are synonymous

remove

```
remove {<name>}
```

Clears the complete internal list or the single structure named <name> Commands 'reset' and 'remove' are synonymous

A.60   superspace

super

Menu to apply (disordered) superspace concepts. The physical crystal is modified accoring to one or several waves that are defined in superspace as u = ampl * function(q*r + phase) where u is the resulting displacement vector ampl the ampltude vector q the satellite vector r the current atom position phase an additialn constant phase shift

assign

assign site:<number>,   additional parameters

Assigns the details of the wave function, such as the distinction of a displacement versus substitution ==> "character:", wave type ==> "function:" displacement vector ==> "displacement" For displacment waves replace characters ==> "replace" For Rsubstitution waaves

character

character:density
character:displacement

displacement

displacement:[ampl:<amplitude>, vec:[<vx>, <vy>, <vz>],

phase:<value>, number:<n>]

function

function:sine
function:crenel

pseudo

pseudo:[<pseudo atom types>]

replace

replace:[old:[<old atom types],new:[<new atom types>],
        prob[<low>, <high>], phase:<value>]

run

run

Start the actual calculation

set

```
set file:internal.filenname
set qvec:[<qx>, <qy>, <qz>]
set current:<number>
```

show

```
show
```

Display current settings on screen.

reset

```
reset
```

Set all settings back to system start.

A.61 surface

Menu to define crystal surfaces and related settings Further help is available on:

commands

```
boundary    ! Cut atoms that are outside a boundary
set         ! Make settings related to surfaces
show        ! Show the current surface related settings
```

boundary

```
boundary "hkl",h,k,l [,<distance>] [,{"inside"}|{"outside"}]
boundary "form",h,k,l [,<distance>] [,{"inside"}|{"outside"}]
boundary "form","cubeoct",<distance> [,{"inside"}|{"outside"}]
boundary "sphere",<radius> [,{"inside"}|{"outside"}]
boundary "ellipsoid",<diam_a>, <diam_b>, <diam_c> [,{"inside"}|{"outside"}]
boundary "cylinder",<radius>,<half-height> [,{"inside"}|{"outside"}]
boundary "local"
```

Optional parameters for all forms are: "centx:"<center_x> The central location for the "centy:"<center_y> boundary is at this point "centz:"<center_z> Defaults to (0, 0, 0) "center:"<value> Identical centers for x, y, z. The units are fractional coordinates The value to all four parameter may be specified as "com", the center of mass. This is the average postion of all atoms. "thick:"<thickness> Defaults to value set by set distance, external for atom type 1 Atoms within a distance of <thickness> to the surface are flagged as surface atoms. Units are Angstrom "keep:"{"inside"}|{"outside"} Keep atoms on the inside or outside of boundary Defaults to "inside"
Optional parameters for the "hkl" and "form" style are: "accum:"{"init"|"add"} Initialize the accumulation of surfaces or add another surface Defaults to "init" See description below "exec:"{"run"|"hold"}

Execute the actual cut == "run" or hold the cut for another surface Defaults to "run" See description below

Optional parameters for the "cylinder" style are: "axis:[",<u>,<v>,<w>,{"d"|"r"} "]" Defines the direction of the cylinder axis, <u>,<v>,<w> are fractional components of the direction vector. The fourth parameter is optional and defaults to "d". It specifies if the components of the vector are in direct or reciprocal space.

Optional parameters for the "ellipsoid" style are: "third:[",<u>,<v>,<w>,{"d"|"r"} "]" Defines the direction of the third principal "ellipsoid" axis. <u>,<v>,<w> are fractional components of the direction vector. The fourth parameter is optional and defaults to "d". It specifies if the components of the vector are in direct or reciprocal space. "first:[",<u>,<v>,<w>,{"d"|"r"} "]" Same for the first principal axis of the "ellipsoid". For the "cylinder" its meaningless, as it results in a rotation around the cylinder axis.

This commands removes all atoms that are outside a boundary surrounding the crystal. The distance of the surface from the point 0,0,0 is measured in Angstrom. "hkl" The surface is defined as a plane through the Miller indices <h>,<k>,<l>. The optional <distance> determines the distance between 0,0,0 and the boundary surface. Without this parameter, the distance is automatically calculated from the user supplied <h>,<k>,<l> as 1./dstar(h,k,l). The optional last parameter determines which atoms remain. If the parameter is "inside" or omitted, all atoms remain that are on the same side of the plane as is the origin. If the parameter is "outside", those atoms remain that are on the other side of the plane as the origin. "form" Same as "hkl", except that DISCUS will create all symmetrically equivalent forms of h,k,l and apply them to shape the crystal

With the word "cubeoct" as second parameter, a combination of {100} and {111} faces is created that will build a cubeoctahedron in a cubic system. The <distance> is the distance of the {100} faces from the origin. If the crystal system is not cubic, an error message is displayed and the boundary is not applied. "sphere" The boundary is defined as a sphere at the origin of direct space. The size of the sphere is defined by its radius in Angstrom. The optional last parameter determines which atoms remain. If the parameter is "inside" or omitted, all atoms remain that are inside the sphere. If the parameter is "outside", those atoms remain that are on the outside of the sphere. "ellipsoid" The boundary is defined as a triaxial ellipsoid with user defined diameters along the three orthogonal main axes. The ellipsoid is in fixed orientation with respect to the lattice. Generally the axes are parallel to the base vectors except for the non-orthogonal crystal systems. The axis are: cubic a b c tetragonal a b c orthorhombic a b c hexagonal a* b c trigonal a* b c rhombohedral (bxc*) b c* monoclinic (bxc*) b c* triclinic (bxc*) b c* If a differently oriented shape is required, you must use the ==> 'symmetry' menu to rotate the crystal as a whole prior to the removal of atoms outside the boundary. "cylinder" The boundary is defined as a cylinder of radius <radius> and half-height <height>, both measures are in Angstrom. The cylinder axis is always along the c-axis. The top and bottom planes are parallel to the (001) plane. If the a- and b-axes are not at 90 degrees to the c-axis, the top will be tilted accordingly. If necessary, transform into an orthogonal cell prior to the application of the boundary command. The optional last parameter determines which atoms remain. If the parameter is "inside" or omitted, all atoms remain that are inside the sphere. If the parameter is "outside", those atoms remain that are on the outside of the sphere. "local" With this command version, no atoms are removed. The structure is analyzed if any atom has a very one sided environment, meaning that all neighboring atoms are displaced in the same

direction compared to the central atom. If this is the case the central atom is flagged as a surface atom.

The optional parameters "accum" and "exec" apply to planar surfaces only. Their main purpose is to create interior closed surfaces in combination with the "keep:outside" parameter. In low symmetry crystal systems, a closed hollow space has to be created by a combination of several symmetrically non-equivalent faces or forms. Use the combination: "accum:init, exec:hold" For the first face/form "accum:add, exec:hold" For further faces and forms "accum:add, exec:run" For the last face. As long as the "exec:hold" parameter is given, the faces are accumulated. With the "exec:run" the actual removal of atoms is started. After a "exec:run" parameter, the list of faces is automatically reset. An accumulated set of planes will refer to a common origin, either the default 0,0,0 or the value set by "centx:", "centy:", "centz". Also the keep status is the same for all hkls in the accumulated set. Thus only the last occurrence of these optional parameters will actually be effective. It is therefore best to use the optional parameters for accumulated surfaces only once at the last hkl for which the optional parameter "exec:run" is used. A warning message will be flagged otherwise.

character

```
char <atom_no> [,{"equal"}|{any}] [, "show"]
```

Determines the surface character of atom number <atom_no>. The result can be: 0: Ill defined surface character, atom is probably slightly below the surrounding atoms 1: planar surface 2: edge 3: corner

The optional "show" tells DISCUS to write the character and the resulting normal to screen.

The optional parameter {"equal"|"any"} tells DISCUS to look either for equal atom types only or to include all atom types surrounding the central atom. If the crystal structure consist of many different sites a more precise result will be obtained by restricting the surface to atoms equal in type to the central atom. If, however, there is a mixed occupancy on a given site in the unit cell, all atom types need to be included in order to find the local surface.

Values are always saved into "res[]": Unknown: res[1] = 0 Plane : res[1] = 1, res[2:4] Surface normal Edge : res[1] = 2, res[2:4] Edge vector, res[5:7] 1st. Surface normal res[8:10] 2nd. Surface normal Corner : res[1] = 3, res[2:4] 1st. Surface normal res[8:...] 2nd. and further surface normals

set

Make various setting related to surfaces

```
set distance, {"external" | "internal" | "all"} , <atom_list>, <distance>
set distance, {"external" | "internal" | "all"} , {"default" | "off"}
set original:[<atom_list>]
set  replace:[<atom_list>]
```

The first two commands define the distances between those atoms in <atom_list> and the specified surface. All atoms of the selected type that are closer than <distance> to the specified surface are marked as "close". The respective property can be used everywhere an atom type is selected. ==> 'property' menu and the 'sel' / 'des' commands in various menus.

"external" The surface created by the ==> 'boundary' command, see the ==> 'surface' menu and the main menu 'boundary' command.

"internal" The surface is an internal surface between atoms that are inserted through a ==> 'domain' and the host structure.

"all" The distances apply to both surface type

<atom_list> List of atoms for which the <distance> is to apply. The list can be the "all" keyword or a list of atom names.

<distance> Distance in Angstrom between an atom and the surface. If set to zero, the flag is not set.

Instead of the atom list, the values can be set for all atoms to:

"default" A default value of 2.55 Angstrom for all atoms.

"off" A value of zero Angstrom. No surface property flag will be set.

set original:[<atom_list>] set replace:[<atom_list>]

The combination of these two optional parameters defines what DISCUS does at the surface. As default, we have original:[all], replace:[void] This implies the normal behaviour that atoms outside the boundary are replaced by VOID atoms, i.e. effectively removed. If the original<atom_list> is restricted and the replace:[<atom_list>] contains atoms other than "VOID" DISCUS will act very differently. In this case atoms outside the boundary and in the original:[] list will be replaced by atoms from the replace:[<atom_list>]. This can be used to create an epitaxial core-shell structure.

show

```
show
```

Show the current surface related settings.

A.62 switch

```
switch <atom1>,<atom2>
switch <mole1>,<mole2>,"mol"
```

The atoms number <atom1> and <atom2> switch position. Internally, the chemistry of the two atoms is switched, while the positions remain as before. The corresponding properties are switched as well.

If 'mol' is given as a third parameter, molecules <mole1> and <mole2> are interchanged. The origin of the two molecules (position of atom 1) are unchanged.

A.63 symmetry

```
symmetry
```

Switches to the symmetry sub menu of DISCUS. You can define a general symmetry operation in triclinic space such as a rotation around a general axis by a general angle. DISCUS performs a symmetry operation in general triclinic space according to the matrix notation:

```
r' = S*r + t
```

Here, r is the original position, r′ its image, S the symmetry matrix and t the translation vector. There are no restrictions on the direction of the symmetry axis, the rotation angle or the translational part. You can use this feature to create a twinned crystal, to rotate a group of atoms around a general axis etc.

Further help topics are:

commands

Valid commands at this level are

```
@        ! Execute a macro file (see main help)
=        ! Assigns the value to a variable (see main help)
angle    ! Sets the angle of a rotation axis
asym     ! Shows asymmetric unit
calc     ! Calculates the symmetry transformation for a single point
chem     ! Shows the atoms present in the crystal
continue! Continue a stopped macro (see main help level)
des      ! Deselects atoms
echo     ! Echo a string (see main help)
eval     ! Evaluates an expression for interactive check (see main help)
exit     ! Terminates 'symm'
help     ! Gives on line help for symmetry operations (see main help)
hkl      ! Sets the symmetry axis in reciprocal space coordinates
incl     ! Sets the range of atoms to be included in the symmetry operation
mdes     ! Deselects molecules
mincl    ! Sets the range of molecules to be included in the
         ! Symmetry operation
mode     ! Allows choice of creating a copy or replacing the atom
msel     ! Selects molecules to be included in the symmetry operations
odes     ! Deselects objects
oincl    ! Sets the range of objects to be included in the symmetry operation
origin   ! Sets the origin for the symmetry operation
osel     ! Selects objects to be included in the symmetry operations
power    ! Sets the power of the symmetry operation
reset    ! Sets the status back to program start conditions
run      ! Starts the calculation of the symmetry operations
sel      ! Selects atoms to be included in the symmetry operations
show     ! Shows the current parameters
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
trans    ! Sets the translational part of the symmetry operation
type     ! Allows choice between proper and improper rotations
uvw      ! Sets the symmetry axis in direct space coordinates
use      ! Defines the space group symmetry matrix to be used
wait     ! Waits for user input (see main help)
```

angle

```
angle <value>
```

Sets the angle of the symmetry operation. The angle must be given in degrees. No limit applies to the numerical value.

asym

```
asym
```

Shows the content of the asymmetric unit. The names of those atoms, a number that is used as index for its scattering type, their position and temperature coefficient are listed. The number that is listed, is the number that refers to the scattering curve of that atom. It is contained in the variable m[<index>]. If a cell was read, all atoms are considered to be different, even if they are chemically identical and have the same temperature coefficient. If a whole structure was read, all atoms that are in the unit cell $0 <= xyz < 1$, are chemically unique and have a different temperature coefficient are included in the asymmetric unit.

calc

```
calc x,y,z [,"d" | "r" ]
```

Calculates the symmetry transformation for a single point <x,y,z>. The current settings for the symmetry operation are used to transform the vector <x,y,z>. The result is displayed and stored in the result array "res". By default, or if the optional fourth parameter is set to "d", DISCUS assumes that the vector is a real space vector.i If the optional fourth parameter is set to "r" the vector is interpreted as a reciprocal space vector and the corresponding symmetry matrix in reciprocal space is used.

chem

```
chem
```

Displays the type of atoms present in the crystal. For each type of atom, its scattering curve number, its name and its temperature factor are listed. Warning, even, if all atoms of a particular type have been deleted, its scattering type will remain in the list. This list could therefore include more types of atoms than are actually present in the crystal.

des

```
des { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

Deselects choices made by ==> 'sel' . Be sure to combine the 'des', 'sel' and 'incl' commands and make sure that no 'msel', mdes' or 'mincl' command follows prior to the 'run' command. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"     all atoms of the crystal are deselected.
<name>    all the atoms called <name> of the crystal are deselected.
          This includes symmetrically not equivalent atoms.
<number>  all atoms of the crystal that are of scattering type <number>
          are deselected.
```

More than one atom may be deselected at once.

hkl

```
hkl <h>,<k>,<l>
```

Specifies the direction of the symmetry axis in reciprocal space coordinates. The corresponding direct space coordinates are calculated by the program. Alternatively you may specify the direction in direct space by the ==> 'uvw" command.

incl

```
incl {<start>,<end> | "all" | "envi" }
```

This command serves to choose wich actual atoms will be modified by the symmetry operation.
Chose which atom types will be included with the ==> 'incl' command.
Be sure to combine the 'sel' 'des' and 'incl' commands and make sure that no 'msel', mdes' or 'mincl' command follows prior to the 'run' command.
The symmetry operation includes all atoms numbered <start> to <end> inclusively. All other atoms are ignored. If, instead of explicit numbers, the parameter "all" is given, the symmetry operation will include all atoms of the crystal. This holds even, if at a later time you include further atoms in the crystal. Thus, you can define a setup for the symmetry operations early in a lengthy macro, then modify the crystal and just run the symmetry operation later on. If the parameter "envi" is specified, the symmetry operation will include all atoms in the current environment that was located by the last ==> 'find env' command. In addition you can define the atoms that are affected by the symmetry operation with the ==>'sele' and 'dese' commands.

mdes

```
mdes { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

Deselects choices made by ==> 'msel'.
Be sure to combine the 'mdes', 'msel' and 'mincl' commands and make sure that no 'sel', des' or 'incl' command follows prior to the 'run' command.
Possible values for the parameter are mutually exclusively:

```
"all"     all molecules of the crystal are deselected.
<number>  all molecules of the crystal that are of type <number>
          are deselected.
```

More than one molecule type may be deselected at once.

mincl

```
mincl {<start>,<end> | "all" } [ , "group", <first> [, <excl1>...]]
```

This command serves to select those actual molecule numbers that will be modified by the symmetry operation.
Chose wich molecule types will be included with the ==> 'mincl' command.
Be sure to combine the 'msel', 'mdes' and 'mincl' commands and make sure that no 'sel', des' or 'incl' command follows prior to the 'run' command.
The symmetry operation includes all molecules numbered <start> to <end> inclusively. All other molecules are ignored. If, instead of explicit numbers, the parameter "all" is given, the symmetry operation will include all molecules of the crystal. This holds even, if at a later time you include further molecules in the crystal. Thus, you can define a setup for the symmetry operations early in a lengthy macro, then modify the crystal and just run the symmetry operation later on. In addition you can define the molecules that are affected by the symmetry operation with the ==>'msele' and 'mdese' commands.

The optional parameters specify that a partial molecule group shall be rotated. The group starts at the atom number <first> within the current molecule. DISCUS uses the connectivity to build this group. Thus you must have specified a connectivity for all atoms within the molecule. Atoms <excl1> etc and their respective connectivity are excluded from the group. This allows you to rotate part of a molecule with respect to the rest of the molecule. Beware that you must have specified a proper connectivity list! This list must contain a connectivity definition for all atom types within the molecule. If you read the structure with the "cell" command, different atom types will exist even is the name and the ADP of two atoms in the cell file were identical.

mode

```
mode { "copy" | "repl" } [ , { "new" | "old" } ]
```

[, "occupied:"{"any"|"empty"}] [, "radius:"<value>]
The symmetry operation can have two different effects on the current atom. In the "copy" mode, the symmetry operation creates a copy of the current atom. In the "repl" mode, the symmetry operation moves the current atom from its old position to the new position.
The optional second parameter has only an effect if molecules are used. It determines if the new molecule created by the symmetry operation has is of the same type as the original ("old") or is treated as a new molecule type ("new").
The optional parameters "occupied" and "radius" are effective only if atoms are selected. These parameters allow you to choose if the position onto which the atom is copied or moved can be occupied by any other atom or must be empty within a radius of <value> Angstroms. If the power of the symmetry operation is greater than one and multiple copies are chosen, this option applies to each individual copy separately.
Try to avoid this option with "empty", as the required inner loop over all atoms in the crystal is bound to take some time.
Default values are "any" and a radius of 1.0*10∗∗-8 Angstrom. With these defaults the new position may be occupied by previous atoms.

msel

```
msel { "all" | <number> } [ , <number> ...]
```

This command serves to select those molecule types that will be modified by the symmetry operation.
Chose wich actual molecules will be included with the ==> 'mincl' command.
Be sure to combine the 'msel', 'mdes' and 'mincl' commands and make sure that no 'sel', des' or 'incl' command follows prior to the 'run' command.
Possible values for the first mandatory parameter are mutually exclusively:

```
"all"    all molecules of the crystal are included.
         This includes the empty molecules in the structure, which are
         stored as scattering curve number zero.
<number> all molecules of the crystal that are of type <number>
         are included.
```

More than one molecule type may be selected at once.

The selection of molecules by 'msel' and atoms by 'sel' are mutually exclusive. The symmetry operation uses either atoms or molecules. Be careful with unintended side effects. Currently DISCUS uses the same internal variable to store which atoms/molecules are selected. Be careful when switching between msel/sel mdes/des.

odes

odes { "all" | <name> | <number> } [, { <name> | <number> } ...]

Deselects choices made by ==> 'osel' . Possible values for the parameter are mutually exclusively:

```
"all"     all objects of the crystal are deselected.
<number>  all objects of the crystal that are of type <number>
          are deselected.
```

More than one object type may be deselected at once.

oincl

oincl {<start>,<end> | "all" }

The symmetry operation includes all objects numbered <start> to <end> inclusively. All other objects are ignored. If, instead of explicit numbers, the parameter "all" is given, the symmetry operation will include all objects of the crystal. This holds even, if at a later time you include further objects in the crystal. Thus, you can define a setup for the symmetry operations early in a lengthy macro, then modify the crystal and just run the symmetry operation later on. In addition you can define the objects that are affected by the symmetry operation with the ==>'osele' and 'odese' commands.

origin

origin <x>,<y>,<z> [, { "mol" | "cryst" }]
origin "atom", <number> [, { "molecule" | "crystal" }]

Specifies the origin of the symmetry operation in direct space coordinates. The symmetry axis intersects this point and has the direction specified by ==> 'hkl' or 'uvw'. If the symmetry operation is an improper rotation, the center of symmetry is at <x>,<y>,<z>.

In case of molecules to be transformed, the optional last parameter specifies if the given origin is relative to the origin of the molecule ("mol") or the crystal ("cryst"). Note that the first atom within a molecule has to be on the molecules origin (see manual).

With the second form, the origin is specified as the coordinates of the atom number <number>. The last optional parameter tells DISCUS if the atom number is absolute within the crystal, or a relative atom number within a molecule content. In the second case, an atom number "2" for example would be the second atom within a chosen molecule. If several molecules are included in the symmetry operation, the origin is adapted for each individual molecule.

osel

`osel { "all" | <number> } [ , <number> ...]`

This command serves to select those objects that will be modified by the symmetry operation. Defines which objects are included in symmetry operation. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"    all objects of the crystal are included.
         This includes the empty objects in the structure, which are
         stored as scattering curve number zero.
<number> all molecules of the crystal that are of type <number>
         are included.
```

More than one atom may be selected at once.
The selection of objects by 'osel' and atoms by 'sel' are mutually exclusive. The symmetry operation uses either atoms or objects.

power

`power <n> {, "multiple" | "single" }`

Sets the power of the symmetry operation. The symmetry operation is applied $<n>$ times.
If the ==> 'mode' of the symmetry operation is "repl", the second parameter is meaningless. The symmetry operation that applies in this case is :

$$r' = S^n * r + t^n$$

The atom is moved to the new position that results from an $<n>$ fold application of the symmetry operation.
If the ==> 'mode' of the symmetry operation is "copy", $<n>$ new atoms are created. The position of the first atom results from a single application of the symmetry operation, the position of the second from a twofold application etc.

reset

`reset`

Reset all symmetry definitions back to the default values at system start.

run

`run`

Starts the symmetry operation.

sel

`sel { "all" | <name> | <number> } [ , { <name> | <number> } ...]`

This command serves to select those atom types that will be modified by the symmetry operation. Chose which actual atoms will be included with the ==> 'incl' command. Be sure to combine the 'sel' and 'incl' commands and make sure that no 'msel', mdes' or 'mincl' command follows prior to the 'run' command.
Possible values for the first mandatory parameter are mutually exclusively:

`"all"     all atoms of the crystal are included.`

This includes the "voids" in the structure, which are stored as scattering curve number zero.

```
<name>    all the atoms called <name> of the crystal are included.
          This includes symmetrically not equivalent atoms.
<number>  all atoms of the crystal that are of scattering type <number>
          are included.
```

More than one atom may be selected at once.
The selection made stay valid until explicitly deselected!

show

`show`

Shows the current parameters of the symmetry operation.

trans

`trans <t1>,<t2>,<t3>`

Sets the translational part of the symmetry operation. The translation is applied after the rotation according to the matrix notation:

`r' = S*r + t`

Here, r is the original position, r' its image, S the symmetry matrix and t the translation vector.

type

`type {"proper" | "improper" }`

A general rotation may be either a "proper" or an "improper" rotation. An improper rotation is a normal rotation followed by an inversion. To create, for example, a mirror operation specify a 180 degree angle and an improper rotation.

---

uvw

```
uvw <h>,<k>,<l>
uvw "atoms", <from>, <to> [ ,{"crystal"|"molecule"}]
```

Specifies the direction of the symmetry axis in direct space coordinates. The corresponding reciprocal space coordinates are calculated by the program.

With the second form, the axis is specified as the direction from the atom number <from> to atom number <to>. The last optional parameter tells DISCUS if the atom numbers are absolute within the crystal, or relative atom numbers within a molecule content. In the second case, an atom number "2" for example would be the second atom within a chosen molecule. If several molecules are included in the symmetry operation, the axis is adapted for each individual molecule.

Alternatively you may specify the direction in reciprocal space by the ==> 'hkl" command.

use

```
use <number>
```

Define the space group symmetry matrix to be used. The symmetry matrix is used as is, therefore this command is incompatible with the commands: uvw, hkl, origin, trans, angle, power, type The use of any of these commands will automatically switch off the space group symmetry matrix.

A.64   transform

```
tran
```

Switches to the unit cell transformation sub menu of DISCUS. At this sub level you can define the relationship between an old and a new unit cell and perform the transformation of the atoms in the crystal. An interactive transformation allows to calculate the result for any single real and reciprocal space vector in both directions: old ==> new and new ==> old.

You can specify the relation ship between the two unit cells in any of six possible options: define the new base vectors a,b,c in terms of the old base define the old base vectors a,b,c in terms of the new base define the new coordinates x,y,z in terms of the old coordinates define the old coordinates x,y,z in terms of the new coordinates define the new base vectors a*,b*,c* in terms of the old reciprocal base define the old base vectors a*,b*,c* in terms of the new reciprocal base Independent of the choice above, you can define an optional shift of the origin by: defining the coordinates of the new origin in terms of the old base defining the coordinates of the old origin in terms of the new base

If all atoms in the crystal are transformed to the new base vectors, then the unit cell dimensions and the metric tensors are transformed as well. The space group is set to "P1" to prevent erroneous symmetry operations once the present crystal is saved to file and read again. All original symmetry operations are transformed and stored as additional generators.

commands

Valid commands at this level are

```
@        ! Execute a macro file (see main help)
=        ! Assigns the value to a variable (see main help)
anew     ! Sets the new base vector "a" in terms of the old base
aold     ! Sets the old base vector "a" in terms of the new base
asnew    ! Sets the new reciprocal base vector "a" in terms of the old base
asold    ! Sets the old reciprocal base vector "a" in terms of the new base
asym     ! Shows asymmetric unit
bnew     ! Sets the new base vector "b" in terms of the old base
bold     ! Sets the old base vector "b" in terms of the new base
bsnew    ! Sets the new reciprocal base vector "b" in terms of the old base
bsold    ! Sets the old reciprocal base vector "b" in terms of the new base
c2new    ! Calculates the transformation to "new" for a single vector
c2old    ! Calculates the transformation to "old" for a single vector
chem     ! Shows the atoms present in the crystal
cnew     ! Sets the new base vector "c" in terms of the old base
cold     ! Sets the old base vector "c" in terms of the new base
csnew    ! Sets the new reciprocal base vector "c" in terms of the old base
csold    ! Sets the old reciprocal base vector "c" in terms of the new base
continue! Continue a stopped macro (see main help level)
des      ! Deselects atoms
echo     ! Echo a string (see main help)
eval     ! Evaluates an expression for interactive check (see main help)
exit     ! Terminates 'tran'
help     ! Gives on line help for unit cell transformations (see main help)
h2new    ! Transforms a list of reflections to the new base system
h2old    ! Transforms a list of reflections to the old base system
incl     ! Sets the range of atoms to be included in the transformation
onew     ! Sets the new origin in terms of the old base
oold     ! Sets the old origin in terms of the new base
reset    ! Reset and clear all definitions
run      ! Starts the transformation for the selected atoms
sel      ! Selects atoms to be included in the transformation
set      ! Sets parameters
show     ! Shows the current parameters
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
wait     ! Waits for user input (see main help)
xnew     ! Sets the new coordinate "x" in terms of the old x,y,z
xold     ! Sets the old coordinate "x" in terms of the new x,y,z
ynew     ! Sets the new coordinate "y" in terms of the old x,y,z
yold     ! Sets the old coordinate "y" in terms of the new x,y,z
znew     ! Sets the new coordinate "z" in terms of the old x,y,z
zold     ! Sets the old coordinate "z" in terms of the new x,y,z
```

anew

`anew a,b,c`

Defines the new base vector "a" in terms of multiples <a>, <b>, <c> of the old base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'anew','bnew' and 'cnew'.

aold

`aold a,b,c`

Defines the old base vector "a" in terms of multiples <a>, <b>, <c> of the new base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'aold','bold' and 'cold'.

asnew

```
asnew a,b,c
```

Defines the new reciprocal base vector "a" in terms of multiples <a>, <b>, <c> of the old reciprocal base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'asnew','bsnew' and 'csnew'.

asold

```
asold a,b,c
```

Defines the old reciprocal base vector "a" in terms of multiples <a>, <b>, <c> of the new reciprocal base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'asold','bsold' and 'csold'.

asym

```
asym
```

Shows the content of the asymmetric unit. The names of those atoms, a number that is used as index for its scattering type, their position and temperature coefficient are listed. The number that is listed, is the number that refers to the scattering curve of that atom. It is contained in the variable m[<index>]. If a cell was read, all atoms are considered to be different, even if they are chemically identical and have the same temperature coefficient. If a whole structure was read, all atoms that are in the unit cell $0 <= xyz < 1$, are chemically unique and have a different temperature coefficient are included in the asymmetric unit.

bnew

```
bnew a,b,c
```

Defines the new base vector "b" in terms of multiples <a>, <b>, <c> of the old base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'anew','bnew' and 'cnew'.

bold

```
bold a,b,c
```

Defines the old base vector "b" in terms of multiples <a>, <b>, <c> of the new base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'aold','bold' and 'cold'.

bsnew

`bsnew a,b,c`

Defines the new reciprocal base vector "b" in terms of multiples <a>, <b>, <c> of the old reciprocal base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'asnew','bsnew' and 'csnew'.

bsold

`bsold a,b,c`

Defines the old reciprocal base vector "b" in terms of multiples <a>, <b>, <c> of the new reciprocal base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'asold','bsold' and 'csold'.

c2new

`c2new x,y,z [, ["d" | "r" } ]`

Calculates the transformation from the old coordinate system to the new system for a single vector <x>,<y>,<z>. Default is a direct space vector, the fourth optional parameter allows you to define <x>,<y>,<z> as a reciprocal space vector. The result of the transformation is displayed on the screen and stored in the first three elements of the result array "res[i]".

c2old

`c2old x,y,z [, ["d" | "r" } ]`

Calculates the transformation from the new coordinate system to the old system for a single vector <x>,<y>,<z>. Default is a direct space vector, the fourth optional parameter allows you to define <x>,<y>,<z> as a reciprocal space vector. The result of the transformation is displayed on the screen and stored in the first three elements of the result array "res[i]".

csnew

`csnew a,b,c`

Defines the new reciprocal base vector "c" in terms of multiples <a>, <b>, <c> of the old reciprocal base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'asnew','bsnew' and 'csnew'.

csold

`csold a,b,c`

Defines the old reciprocal base vector "c" in terms of multiples <a>, <b>, <c> of the new reciprocal base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'asold','bsold' and 'csold'.

---

chem

`chem`

Displays the type of atoms present in the crystal. For each type of atom, its scattering curve number, its name and its temperature factor are listed. Warning, even, if all atoms of a particular type have been deleted, its scattering type will remain in the list. This list could therefore include more types of atoms than are actually present in the crystal.

cnew

`cnew a,b,c`

Defines the new base vector "c" in terms of multiples <a>, <b>, <c> of the old base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'anew','bnew' and 'cnew'.

cold

`cold a,b,c`

Defines the old base vector "c" in terms of multiples <a>, <b>, <c> of the new base vectors. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'aold','bold' and 'cold'.

des

`des { "all" | <name> | <number> } [ , { <name> | <number> } ...]`

Deselects choices made by ==> 'sel' . Possible values for the parameter are mutually exclusively:

```
"all"      all atoms of the crystal are deselected.
<name>     all the atoms called <name> of the crystal are deselected.
           This includes symmetrically not equivalent atoms.
<number>   all atoms of the crystal that are of scattering type <number>
           are deselected.
```

More than one atom may be deselected at once.

h2new

`h2new <infile>`

Transforms the reflections read from <infile> into reflections according to the new base system. The input format is the SHEXL format(3i4,2f8.2,i4). All new reflections that are integer are written to the output file named <infile>".trans" in SHELX format. You can set an accuracy for being integer by the ==> 'set deltahkl' command. All other reflections, that are more than +- <deltahkl> away from integer h,k,l are written to the file <infile>".rest" as hkl(new), intensity, sigma, batch, hkl(old). Here hkl(new) are real values.

h2old

```
h2old <infile>
```

Transforms the reflections read from <infile> into reflections according to the old base system. The input format is the SHEXL format(3i4,2f8.2,i4). All old reflections that are integer are written to the output file named <infile>".trans" in SHELX format. You can set an accuracy for being integer by the ==> 'set deltahkl' command. All other reflections, that are more than +-<deltahkl> away from integer h,k,l are written to the file <infile>".rest" as hkl(old), intensity, sigma, batch, hkl(new) Here hkl(old) are real values.

incl

```
incl { <start>,<end> | "all" }
```

The unit cell transformation includes all atoms numbered <start> to <end> inclusively. All other atoms are ignored. If, instead of explicit numbers, the parameter "all" is given, the unit cell transformation will include all atoms of the crystal. This holds even, if at a later time you include further atoms in the crystal. Thus, you can define a setup for the unit cell transformations early in a lengthy macro, then modify the crystal and just run the unit cell transformation later on. In addition you can define the atoms that are affected by the unit cell transformation operation with the ==>'sele' and 'dese' commands.

onew

```
onew a,b,c
```

Defines the position of the new origin in terms of multiples <a>,<b>,<c> of the old base vectors. The default at program startup is 0.0, 0.0, 0.0

oold

```
oold a,b,c
```

Defines the position of the old origin in terms of multiples <a>,<b>,<c> of the new base vectors. The default at program startup is 0.0, 0.0, 0.0

reset

```
reset
```

Reset all transformation definitions back to the default values at system start.

run

```
run
```

Starts the transformation operation.

---

sel

```
sel { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

This command serves to select those atoms that will be modified by the unit cell transformation. Defines which atoms are included in unit cell transformations. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"     all atoms of the crystal are included.
```

This includes the "voids" in the structure, which are stored as scattering curve number zero.

```
<name>    all the atoms called <name> of the crystal are included.
          This includes symmetrically not equivalent atoms.
<number>  all atoms of the crystal that are of scattering type <number>
          are included.
```

More than one atom may be selected at once.
The selection made stay valid until explicitly deselected!

set

```
set "deltahkl",<value>
```

Sets the value of deltahkl to <value>, as needed for ==> 'h2new' and 'h2old'. When DISCUS transforms a list of reflections, it checks, how close the transformed values are to integer values h,k,l. The allowed deviation from integer numbers is set by this command.

show

```
show
```

Shows the current parameters of the transformation operation.

xnew

```
xnew a,b,c
```

Defines the transformation through the relationship between the new "x" coordinate of an atom in terms of multiples $<x>$, $<y>$, $<z>$ of the old coordinates of the atom. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'xnew','ynew' and 'znew'.

xold

```
xold a,b,c
```

Defines the transformation through the relationship between the old "x" coordinate of an atom in terms of multiples $<x>$, $<y>$, $<z>$ of the new coordinates of the atom. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'xold','yold' and 'zold'.

ynew

```
ynew a,b,c
```

Defines the transformation through the relationship between the new "y" coordinate of an atom in terms of multiples <x>, <y>, <z> of the old coordinates of the atom. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'xnew','ynew' and 'znew'.

yold

```
yold a,b,c
```

Defines the transformation through the relationship between the old "y" coordinate of an atom in terms of multiples <x>, <y>, <z> of the new coordinates of the atom. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'xold','yold' and 'zold'.

znew

```
znew a,b,c
```

Defines the transformation through the relationship between the new "z" coordinate of an atom in terms of multiples <x>, <y>, <z> of the old coordinates of the atom. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'xnew','ynew' and 'znew'.

zold

```
zold a,b,c
```

Defines the transformation through the relationship between the old "z" coordinate of an atom in terms of multiples <x>, <y>, <z> of the new coordinates of the atom. If you choose this definition of the unit cell transformation, you MUST define all three ==> 'xold','yold' and 'zold'.

A.65   ther

```
ther [{"all" | "2d"}]
ther "mol",[{"all" | "2d"}]
```

Error notice: Previous to Version 5.13.0 the ==> 'therm' command erroneously used too large a displacement. The average $<u**2>$ were actually 3 times too large.

Randomly displaces all atoms according to their respective isotropic thermal coefficient. The directions of the displacements are distributed in uniform random distribution, the amplitude of the displacements are Gaussian distributed with FWHM calculated from the isotropic B. As default the displacements are performed in all 3 dimensions. For 2d simulations however, the

displacement in all directions corresponding to a size of one unit cell can be switched off by the parameter "2d". The value "all" is equal to the default.

If the second parameter is set to "mol", the complete rigid molecule is displaced according to the isotropic temperature factor defines via a "mole_biso[i]=" statement or specified on a "molecule biso,<value>" statement in the ==> Unit cell file.

The temperature factor of the atom at the origin of the molecule is NO longer used!

A.66   variables

The program recognizes free variables:

```
"i[ ]"  integer variables
"r[ ]"  real variables
```

The current implementation allows 501 free variables [0] through [501] each.

```
"res[ ]"  Result of several calculations that return more than one
          value.
          res[0] contains the number of values returned, res[1] to
          res[res[0]] contain the actual values.
          The values of "res" change every time, a command is applied
          that returns more than one value. The old values are lost !
```

Several other variables are linked to structural values:

```
"x[<i>]",
"y[<i>]" and
"z[<i>]" are the fractional coordinates of the atom number <i>.
"at_name[<i>]" is a character variable that holds the name of
        of atom number <i>
"at_type[<i>]" is a character variable that holds the name of
        of atom type <i>
"m[<i>]" is the number of the scattering curve for atom <i>.
"b[<t>]" is the isotropic B value for atom type <t>.
"occ[<t>]" is the occupancy for atom type <t>.
"surf[<i>, <j>] are the j components of the surface vector for
```

atom number <i>. These will all be zero unless the atom is close to an external surface.

```
"n[1]"   is the total number of atoms in the crystal, this variable
         is read only.
"n[2]"   is the total number of different scattering curves in the
         crystal, this variable is read only.
"n[3]"   is the total number of atoms in the original unit cell,
         this variable is read only.
"n[4]"   is the total number of molecules in the crystal, this variable
         is read only.
"n[5]"   is the total number of different molecule types in the
         crystal, this variable is read only.
"n[6]"   is the total number of molecules in the original unit cell,
         this variable is read only.
"n[7]"   is the total number of non-void atoms in the original unit cell,
         this variable is read only.
"cdim[1,1]" lowest  x coordinate of any atom in the crystal.
"cdim[1,2]" highest x coordinate of any atom in the crystal.
"cdim[2,1]" lowest  y coordinate of any atom in the crystal.
"cdim[2,2]" highest y coordinate of any atom in the crystal.
"cdim[3,1]" lowest  z coordinate of any atom in the crystal.
"cdim[3,2]" highest z coordinate of any atom in the crystal.
```

The variable "cdim" is read only.

```
"lat[<i>]"   Lattice constants a,b,c,alpha,beta,gamma
"vol[1]"     Unit cell volume in A**3
"rvol[1]"    Reciprocal unit cell volume in A**(-3)
"sym_n[1]"   Number of symmetry operations for the space group
"env[<i>]"       Index of neighboring atoms as found by
                 ==> "find"
                 "env[0]" is the number of neighbors.
"menv[<i>]"       Index of neighboring molecules as found by
                 ==> "find"
                 "menv[0]" is the number of neighbors.
"mol_cont[i1,i2]" Content of molecule number i1.
                 mol_cont[i1,0] gives the number of
                 atoms in the molecule. For <i2> > 0, the
                 information about the <i2>'nd atom in the
                 molecule is given. The number of the
                 atom in the crystal is returned.
                 read only
"mol_len[i1]"    Number of atoms in molecule i1.
                 read only
"mol_type[i1]"   Type of molecule no. i1
                 read/write
"mol_biso[i1]"   Isotropic B-value for Molecule type i1
                 read/write
"in_mole[i1]"    Molecule number in which atom i1 is located
                 read only
"pdf_dens[0]"    Number density for PDF calculations
"pdf_scal[0]"    Scale factor  for PDF calculations
```

The variables can be assigned constant numbers:

```
i[0] =  1                ! Integer variable no 0 is set to "1"
r[4] =  1.5              ! Real variable no 4 is set to "1.5"
x[5] = -2.7888           ! X-coordinate of atom 5 is set to "-2.7888"
```

Arithmetic expressions may be used on the right hand side:

```
i[i[4]+2] = 2.0*(i[2]+i[0])/(i[2]**2)
m[i[0]+50] = m[i[0]] ! The atom 50 atoms down the line from
                     atom no <i[0]> is assigned the same
                     scattering curve, thus essentially switching
                     its chemistry around !
```

As shown in the example above, arithmetic operations can be nested and grouped by brackets "(" and ")".

A.67   vprod

```
vprod <u1>,<v1>,<w1>,<u2>,<v2>,<w2> [ ,<flag> ]
```

This command calculates the vector product <uvw 1> X <uvw 2>. The <flag> is a string of three characters that indicates whether the input vectors or the output vector are given in direct or reciprocal space coordinates. The three characters correspond to:

```
first input vector,
second input vector
resulting vector
```

```
"drd" means: vector one is given in direct space coordinates, vector 2
             is in reciprocal space coordinates (hkl) and the
             resulting vector product is to be given in direct space
             coordinates
```

Any combination of "d" and "r" is allowed.

The output vector is stored in the result variable: "res[1]","res[2]","res[3]" vector product of vector 1 and vector 2

A.68   waves

```
wave
```

Displaces the atoms along a wave. It generates sub level with prompt "discus/wave >". Further help topics are:

commands

Valid commands at this level are

```
@        ! Execute a macro file (see main help)
=        ! Assigns the value to a variable (see main help)
acco     ! Sets the wave type to acoustic
amp      ! Sets the amplitude in Angstrom
asym     ! Shows asymmetric unit
chem     ! Shows the atoms present in the crystal
continue! Continue a stopped macro (see main help level)
dens     ! Sets the wave type to density wave
des      ! Deselects atoms
echo     ! Echo a string (see main help)
eval     ! Evaluates an expression for interactive check (see main help)
exit     ! Terminates 'wave'
func     ! Sets the wave function
help     ! Gives on line help for waves (see main help)
len      ! Sets the wave length in Angstrom
long     ! Sets the wave type to longitudinal
mdese    ! Deselects molecule types
mrepl    ! Selects molecules to be replaced by density wave
msel     ! Selects molecule types
opti     ! Sets the wave type to optical
osci     ! Oscillation vector for transverse waves
phase    ! Sets the phase of the wave at 0,0,0 in direct space
phigh    ! Sets the high probability for density waves
plow     ! Sets the low probability for density waves
repl     ! Selects atoms to be replaced by a density wave
rot      ! Sets type to rotational wave (molecules only)
reset    ! Reset and clear all definitions
run      ! Starts the calculation of the wave
sel      ! Selects atoms to be included in the wave
shift    ! Allows a constant shift to be added to the displacement
show     ! Show current settings
stop     ! Stops execution of a macro (see main help level)
system   ! Executes operating system command (see main help)
tran     ! Sets the wave type to transverse
vect     ! Sets wave vector
wait     ! Waits for user input (see main help)
```

acco

`acco`

Sets the wave type to acoustic, atoms of opposite charge are displaced in the same direction.

amp

`amp <value>`

Sets the amplitude in Angstrom

asym

`asym`

Shows the content of the asymmetric unit. The names of those atoms, a number that is used as index for its scattering type, their position and temperature coefficient are listed. The number that is listed, is the number that refers to the scattering curve of that atom. It is contained in the variable m[<index>]. If a cell was read, all atoms are considered to be different, even if they are chemically identical and have the same temperature coefficient. If a whole structure was read, all atoms that are in the unit cell $0 <= xyz < 1$, are chemically unique and have a different temperature coefficient are included in the asymmetric unit.

chem

`chem`

Displays the type of atoms present in the crystal. For each type of atom, its scattering curve number, its name and its temperature factor are listed. Warning, even, if all atoms of a particular type have been deleted, its scattering type will remain in the list. This list could therefore include more types of atoms than are actually present in the crystal.

dens

`dens`

Sets the type of the wave to be a density wave. All atoms selected by 'sel' or 'repl' will be modulated by the wave. The probability with which an atom is present oscillates between 'plow' and 'phigh'.

des

`des { "all" | <name> | <number> } [ , { <name> | <number> } ...]`

Deselects choices made by ==> 'sel' or ==> 'repl'. Possible values for the parameter are mutually exclusively:

```
"all"     all atoms of the crystal are deselected.
<name>    all the atoms called <name> of the crystal are deselected.
          This includes symmetrically not equivalent atoms.
<number>  all atoms of the crystal that are of scattering type <number>
          are deselected.
```

More than one atom may be deselected at once.

func

```
func "box" [,<delta>]
func "sinus"
func "triang"
```

Sets the type of wave function used. The parameter can be any of the strings listed.
A) displacement waves:

```
"box"   : A box shaped wave function is used. The function is
          symmetrical around the origin. This, however, can be
          changed by changing the phase at the origin, see command
          ==> 'phase'. The default is a symmetrical box function,
          i.e. a value of <delta> of 0.5. The value of <delta>
          can be given as additional parameter to us an asymmetric
          box function, i.e. maxima and minima have different widths.
"sinus" : A sinusoidal wave function is used. The displacement is given
          by amp*SIN(arg).
"triang": A triangular displacement is applied. The displacement linearly
          increases from zero to amp and drops back to zero at the end
          of the wave. The point at 0/4 = 4/4 wave length has displacement
          zero.
```

The constant shift is added to these displacements.
B) replacement waves:

```
"box"   : A box shaped wave function is used. The occupation probability
          of the original atom, see ==> 'repl', is equal to <plow> along
          half the wave length and <phigh> else. Other details were
          given above.
"sinus" : A sinusoidal wave function is used. The occupation probability
          is given by
              amp * SIN(arg) + amp0, where
              amp = 0.5*( phigh-plow)
              amp0= 0.5*( phigh+plow)
          This will give a sinusoidal wave with minimum at <plow> and
          maximum value at <phigh>.
"triang": A triangular occupation probability is applied. The occupation
          probability linearly increases from <plow> to <phigh> and drops
          back to zero at the end of the wave.
          The point at 0/4 = 4/4 wave length has the occupation
          probability of <plow>.
```

len

```
len <value>
```

Sets the wave length in Angstrom

---

long

```
long
```

Sets the wave type to longitudinal

mdes

```
mdes { "all" | <number> } [ , <number> ...]
```

This command deselects all or individual molecule types used for the plotting output.

msel

```
msel { "all" | <number> } [ , <number> ...]
```

This command allows to select the molecule types to be used for the plotting output. By using 'msel' the atom selection becomes invalid. The parameter "all" will select ALL molecule types. Alternatively individual molecule types can be selected using the corresponding <number>. In order to use atoms again, use the 'sele' command.

mrep

```
mrep <typ1>,<typ2> [,"viceversa"]
```

This command selects the molecules that will be replaced by a density function. The parameter <typ1> specifies the molecule type to eventually be replaced by type <typ2>. Repeat the command to select more than one molecule type. To deselect use 'mdes'. To remove the molecules rather than replacing them, set <typ2> to zero.
If the third optional parameter "viceversa" is given, the molecules of type <typ2> are replaced by molecules of type <typ1> with the opposite probability. This is useful, if you have a random arrangement of molecule types <typ1> and <typ2> and would like to obtain a density wave that fluctuates between these two types. If the optional parameter is omitted, only molecules of type <typ1> are replaced and all molecules of type <typ2> are not changed. Keep in mind that you should select only those molecules that you want to exchange, and do not use the ==> msel all construction.

rot

```
rot <u>,<v>,<w> [,<x>,<y>,<z>]
```

This command selects the rotational wave mode which is only available when using molecules. The wave amplitude is here used as rotation angle rather than as displacement or probability as for the other modes. The parameters <u>,<v>,<w> define the rotation axis in real space. The origin of the rotation axis is the origin of the molecule. However, an offset <x>,<y>,<z> for this origin can be specified as optional parameters.

opti

```
opti
```

Sets the wave type to optical, atoms of opposite charge are displaced in opposite directions.

osci

```
osci <x,y,z>
```

Sets the direction of the oscillation vector for transverse waves.

phase

```
phase { <value> | "random" }
```

Sets the phase of the wave at 0,0,0 in direct space in degrees. The initial phase is zero. With phase = 0, a sine wave results, with phase = 90 a cosine wave.

phigh

```
phigh <value>
```

Sets the upper probability limit, with which an atom is retained by a density wave. The maximum occupancy of the atom reaches <value>.

plow

```
plow <value>
```

Sets the lower probability limit, with which an atom is retained by a density wave. The minimum occupancy of the atom reaches <value>.

repl

```
repl { "all" | <name> | <number> } , { <name> | <number> } [,"viceversa"]
```

This command selects those atoms that will be replaced by a density function. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"      all atoms of the crystal are replaced by the atom defined
           by the second parameter.
<name>     all the atoms called <name> of the crystal are replaced by
           the atom defined by the second parameter. This includes
           symmetrically not equivalent atoms.
<number>   all atoms of the crystal that are of scattering type <name>
           are replaced by the atom defined by the second parameter.
```

The second parameter defines the atom, the original is replaced by. If a name is given, it need not to be present in the original crystal. A new scattering type is automatically added to the table. If a number is given, to which no corresponding atom exists, an error message is displayed.

If the third optional parameter "viceversa" is given, the atoms of the first type are replaced by atoms of the second type with the opposite probability. This is useful, if you have a random arrangement of atoms types and would like to obtain a density wave that fluctuates between these two types. If the optional parameter is omitted, only atoms of the first type are replaced and all atoms of the second type are not changed. Keep in mind that you should select only those atoms that you want to exchange, and do not use the ==> sel all construction.

reset

```
reset
```

Reset all wave definitions back to the default values at system start.

run

```
run
```

Starts the calculation of the wave

sel

```
sel { "all" | <name> | <number> } [ , { <name> | <number> } ...]
```

This command serves to select those atoms that will be modified by a wave function.
Defines which atoms are included in a wave. Possible values for the first mandatory parameter are mutually exclusively:

```
"all"    all atoms of the crystal are included.
```

This includes the "voids" in the structure, which are stored as scattering curve number zero.

```
<name>    all the atoms called <name> of the crystal are included.
          This includes symmetrically not equivalent atoms.
<number>  all atoms of the crystal that are of scattering type <number>
          are included.
```

More than one atom may be selected at once.
The selection made stay valid until explicitly deselected!

shift

```
shift <amount>
```

A constant shift <amount> is added to the displacement of the atoms

---

show

`show`

Shows the current settings for the wave.

tran

`tran`

Sets the wave type to transverse

vect

`vect <x,y,z>`

Sets the direction of the wave vector in units of the lattice constants. The wave length has to be set with ==> "len".

A.69   wyckoff

`wyckoff <x>,<y>,<z> [,{"full" | "symbol" | "xzy" | "matrix" }`

Shows all symmetry operations that copy an atom at position <x>,<y>,<z> onto itself. These are the symmetry operations of the current space group, not the simple point group symmetry. The following results are written into res[*]: res[1] Multiplicity of this site res[2] Highest multiplicity divided by multiplicity of this site i.e. number of symmetry operations that copy the site onto itself. res[3] Multiplicity of a general position xyz in this space group == highest multiplicity res[4...] Number of the space group symmetry elements that copy the position onto itself.

`"full"`

The default setting, produces a full listing. This includes: the symmetry matrix as (3x4) matrix the symbol as printed in the International Tables the resulting atom positions xyz, as printed in the International Tables
For each symmetry operation DISCUS lists the number in square brackets, and for centered space groups the number within each block that corresponds to one centering vector in round brackets. With the setting ==> set generator,symmetry the sequence corresponds to that listed in the Tables.

`"symbol"`

This produces a short list that consists of the symbol as used in the International Tables.

`"xyz"`

This produces a short list that consists of the symbolic atom positions xyz, as used in the International Tables.

`"matrix"`

This produces a list that consists of the (3x4) symmetry matrices.

A.70   bugs

Known bugs

Here is a list of known bugs of DISCUS which are on our 'repair list':

```
- DISCUS hangs if the input is redirected from a file ('discus < file')
  and the prompt status is ON if command line editing is used.
  Use 'set prompt,off' or 'set prompt,redirect' in the first
  line of the file to be redirected to avoid the problem.
```

A.71   errors

The program has been written such that it should handle almost any typing error when giving commands and hopefully all errors that result from calculation with erroneous data. When an error is found an error message is displayed that should get you back on track. See the manual for a complete list of error messages.

Error messages concerning the command language are explained in the -> command language section of the online help. Application specific commands are described here and are grouped as follows:

```
APPL    Errors specific to DISCUS usage
CHEM    Errors at the chemistry sub level
FOUR    Errors related to Fourier Transform
MMC     Errors at the Monte Carlo level
PDF     Errors at the PDF level
RMC     Errors at the Reverse Monte Carlo level
```

Each error message is displayed together with the corresponding category <cccc> and the error number <numb> in the form:

```
****CCCC****message                **** numb ****
```

Type help error <cccc> <numb> to get an explanation for the error message and hints for possible steps to correct the situation.

In the default mode DISCUS returns the standard prompt and you can continue the execution from this point. You can set the error status to "exit" by the ==>'set' command. In this case DISCUS terminates if an error is detected. This option is useful to terminate a faulty sequence of commands when running DISCUS in the batch mode of your operating system. If you are a bit more daring, you can use the "live" error setting. With this setting, DISCUS will continue to run a macro after an error has occurred. This is helpful if you attempt to read a file of unknown length with ==> 'fget' and need to catch the EOF signal. Other than that be careful!

appl

This category lists error messages that are specific to DISCUS usage. You will find error messages regarding improper usage of atoms, space group violations, etc.

Error -1: Maximum number of Orient. matrices read

The maximum number of orientation matrices that can be used for microdomains is limited by the parameter "MAX_MD_ORI" in the file "microdis.inc". If necessary adjust this parameter and recompile the program.

Error -2: Improper limits for atom number

Either of the upper or lower limits used on the 'append' command is outside the range of atoms present in the crystal. Check whether the limits are both positive, the upper limit is higher or equal to the lower limit and whether both limits are less or equal to the number of atoms present in the crystal. The number of atoms in the crystal can be checked with the command: "eval n[1]".

Error -3: No atoms selected yet

The plot and waves can only be run for selected atoms. Use the 'select' command to select individual atom types or to select all atoms present in the crystal.

Error -4: Extend of plot space is zero

The dimensions of the space from which atoms are to be plotted is zero. Check the values displayed by the plot 'show' command and use the 'exte' command to give appropriate values.

Error -5: Too many microdomains within crystal

DISCUS stores all microdomain positions in an array. The size of this array is defined by parameter MC_MAXMICRO in file "microdis.inc". If necessary readjust the value of this parameter and recompile the program.

Error -6: Unknown microdomain type

The microdomain type number used with the 'place' command does not correspond to an existing microdomain type in the list. Check with 'show types' for a list of microdomain types that have been inserted into the list of available microdomains.

Error -7: Unknown space group symbol

The crystal file contains an unknown space group. Check the spelling of the space group symbol. The allowed space groups are all 230 space groups in the Int. Tables for Crystallography plus the space groups given for alternative settings and cell choices of the monoclinic space groups. Refer to appendix A of the manual for a complete listing of space group symbols.

Error -8: Too many points in reciprocal layer

The number of points in reciprocal space that can be calculated by the Fourier transform is limited by the parameter "MAXQXY" in the file "qspace.inc". Check that the product of <na> and <no> is less or equal to "MAXQXY". If necessary adjust the parameter "MAXQXY" and recompile the program.

Error -9: Unknown Output Format

An unknown output format was defined for the output level. Check that either of: "stan", "gnu", "pgm" or "post" was used as output format with the 'format' command.

Error -10: Too many Atoms in crystal

The maximum number of atoms that can be stored in the structure is determined by the parameter MAXATOMS in the file "crystal.inc". By inserting new atoms or by defining too large a crystal on the 'read' command, this number was exceeded. If necessary, change the value of the parameter MAXATOMS and recompile the program.

Error -11: Unknown threshold type

The parameter given on the 'thresh' command does not correspond to a valid entry. Check the spelling of the parameter.

Error -12: Number of points must be > zero

The value of the parameter given on the 'abs' or 'ord' command is less than one. This value represents the number of data points calculated along the respective direction. The value must be at least one or higher.

Error -13: Correlation matrix index outside limits

The index of the correlation matrix element(s) is either less than one or higher than the number of defined microdomain types. Check the 'ccol', 'crow' and 'cele' commands for proper number of parameters. Use the 'show type' command at 'microdomain' sub level to see the number of microdomains currently defined.

Error -14: Invalid space group & lattice constants

While reading a new cell the program checks the space group and the lattice constants for consistency. Either a space group was given that is not included in the program, or the lattice constants do not fulfill the constraints imposed by the space group. Check the space group symbol and the lattice constants given in the input file.

Error -15: No microdomain input file name defined

An attempt was made to insert a microdomain type without specifying the input file name. With the exception of content type 'free' all microdomains require an input file from which the new atom positions are read. Check the current defaults by 'show def' and the parameters given on then 'cont' command.

Error -16: Status for log must be : "on" or "off"

The second parameter for the 'log' command must be either "on" or "off".

Error -17: Index of matrix outside limits

The first parameter on the 'mrow' command is the index of the row that is to be inserted. Allowed values are one, two or three.

Error -18: No orientation with this number exists

The orientation matrix requested as parameter on the 'orie' command does not exist. Use the 'show orie' command to view the list of defined orientation matrices. Check whether three 'mrow' commands were used for each orientation matrix that you intended to set up.

Error -19: Atom number outside limits

The number of the atom is either less than one, larger than the current number of atoms in the crystal or even larger than the maximum number of atoms allowed in your implementation. Check the value of the parameter(s) on the 'remove' and 'switch' commands or check the index of the variables "m", "x", "y" or "z". Check the number of atoms present in the crystal by the command: 'eval n[1]'.

Error -20: Unknown element, no Fourier calculated

An element was detected in the list of atoms for which there is no scattering curve available. The Fourier transform is not calculated at all. Check the name of all atoms present in the crystal using the 'asym' and 'chem' commands. If a charged ion was given, this valence might not be present in the list of scattering curves. Refer to Appendix b of the manual for a list of internally stored names. If the 'scat' and/or 'delf' commands were used, any name may be used. Check whether the commands were used properly.

Error -21: No element present, no Fourier calculated

There are no elements present at all in the crystal. The Fourier transform is not calculated at all. Most likely, the Fourier was called before a structure or unit cell was read, or an error occurred during the reading of the structure or unit cell.

Error -22: Incompatible standard file sizes

While attempting to add two Standard files, DISCUS encountered different length of rows or columns in the two input files. The adding is interrupted at this stage. The indices along the abscissa and ordinate at which the error occurred, is displayed as well, check the input file for erroneous content. Also check, whether the parameters nx,ny in the two input files are identical as well as the xmin,xmax,ymin and ymax parameters in the second line of both input files.

Error -23: Incompatible GNUPLOT file sizes

While attempting to add two GNUPLOT files, DISCUS encountered different length of rows or columns in the two input files. The adding is interrupted at this stage. The line number in which the error occurred, is displayed as well, check the input file for erroneous content.

Error -24: Different coordinates in GNUPLOT files

While attempting to add two GNUPLOT files, DISCUS encountered different coordinates in the two input files. The adding is interrupted at this stage. The line number in which the error occurred, is displayed as well, check the input file for erroneous content.

Error -25: Different coordinates in 1-dim files

While attempting to add two 1-dim files, DISCUS encountered different coordinates in the two input files. The adding is interrupted at this stage.The line number in which the error occurred, is displayed as well, check the input file for erroneous content.

Error -26: Too many different atoms in crystal

The maximum number of different atoms allowed in your implementation was exceeded. No more new types of atoms can be inserted into the structure. Check the chemistry of your crystal by the 'asym' and 'chem' commands. All atoms are considered different types that are chemically different, have different charge or a different temperature coefficient. If all types are needed, modify the parameter "maxscat" in the file "param.inc" and recompile the program. See chapter 9.1 of the manual for further information.

Error -27: No atom of this type present in crystal

An atom was selected for displacement by a wave or for plotting that does not exist within the crystal. Check the spelling of the atom name, and the chemistry of the crystal by the 'asym' and 'chem' commands.

Error -28: Input parameters must be $>$ zero

This function/command requires non-negative parameters. Check the values of the parameters and the explanation for the function or command used for valid ranges of numerical input.

Error -29: Too many different microdomain types

The maximum number of microdomain types is defined by the parameter MAX_MD_TY in file "microdis.inc". If necessary adjust this parameter and recompile the program.

Error -30: Unknown boundary type

The boundary type of microdomains is not set to either of "block", "face", "fuzzy" or "sphere". Check the 'boundary' command at sublevel 'microdomains' for proper spelling.

Error -31: Unknown distribution mode

The distribution type for microdomains is not set to either of "lattice", "paracrystal" or "random". Check the 'distribution' command at sublevel 'microdomains' for proper spelling.

Error -32: Length of vector is zero

An attempt was made to calculate the angle between two vectors while one of them is of length zero. Check the parameters given on the 'bang' or 'rang' command for proper numbers.

Error -33: No microdomain types defined yet

An attempt was made to use the 'create' or 'run' command before any microdomains were defined by the 'insert' command. Check for proper sequence of commands and missing or misspelled 'insert' commands.

Error -34: Form does not appear to be closed

A microdomain type of boundary type 'face' was defined for which the form supplied by the 'bound' command does appear not to be closed. DISCUS checks whether the origin of the microdomain is inside the microdomain and the positions at $1.0*10**9$ along each of the base vectors is outside the microdomain. If one of these conditions is not met, the error message is given. Check the 'boundary' commands used to define the microdomain type.

Error -35: Volume of unit cell <= zero

The volume of the unit cell was calculated as zero or a negative value. Check the lattice parameters given in the input file. Are there any accidental "-" signs ? Do the angles form an impossible shape ?

Error -36: Unsuitable file types for inverse Fourier

Not every combination of file types is allowed for an inverse Fourier synthesis. Allowed file combinations are:

```
file a      file b
intensity   phase
amplitude   phase
real part   imaginary part
```

Error -37: No filename defined yet

An attempt was made to write output to or read from a file without defining a file name. DISCUS does not provide default names for the output of the 'plot' command or the input file names. Check the 'outfile' command at sublevel 'output' and 'plot' or the 'content' command at sublevel 'microdomains'.

Error -38: Unsuitable file types for Patterson

Not every combination of file types is allowed for a Patterson synthesis. Allowed file combinations are:

```
file  a      file b
intensity    any file type
amplitude    any file type
real part    imaginary part
```

Error -39: All elements of correlation matrix zero

Since the correlation matrix gives the probabilities for existence of a microdomain type, not all elements may be equal to zero. The default values at startup are 1 for the diagonal elements and zero for all other elements. Display the correlation matrix with the command 'show corr' and check the 'ccol', 'cele' and 'crow' commands for proper values.

Error -40: Microdomain type cannot be removed

DISCUS can only remove those microdomain types from the list of types that have not yet been used by the 'create' command, i.e. only those microdomain types that have been inserted into the list by the 'insert' command after the last 'create' command.

Error -41: File specifier must be "a" or "b"

The two files for inverse Fourier and Patterson synthesis are referred to by "a" and "b". The first parameter on the 'file' and the 'type' commands must be either "a" or "b".

Error -42: Type must be: inten,ampl,phase,real,imag

The file type for inverse Fourier and Patterson synthesis must be specified as one of the following strings: "intensity","amplitude","phase","real","imaginary" They may be abbreviated to a unique string.
Check the spelling of the parameter.

Error -43: Not enough parameter for filename format

An attempt was made to generate a file name from a string like "text%dtext" without supplying enough numerical parameters. Check that the spelling of the sting within " " is correct. Are there any unwanted %d combinations?. Check the number and type of parameters following the file name.

Error -44: Right quotation mark missing in format

An attempt was made to generate a file name from a string like "text%dtext" without supplying the right quotation mark. Check the line and provide the missing ".

Error -45: Too many atoms in environment

The indices of all atoms found are stored in the internal variable "res". More atoms were found that fit into the dimensions of "res". Restrict the search for the environment to a smaller shell or change the dimension of "res_para" in file "param.inc".

Error -46: Error reading title of structure

An error occurred while reading the title line of a structure or unit cell file. Check the file for any garbage.

Error -47: Error reading space group symbol

An error occurred while reading the space group line of a structure or unit cell file. Check the file for any garbage.

Error -48: Error reading lattice constants

An error occurred while reading the lattice constants of a structure or unit cell file. Check the file for any garbage or accidental letters.

Error -49: Error reading atom coordinates

An error occurred while reading the atom coordinates of the atom listed. Check the file for any garbage. Is the line of the type Name x y z B Are there letters among the numerical values?

Error -50: Wrong format for 1-dimensional data

You tried to write a one-dimensional section of space as either Postscript, PGM or PPM Bitmap. One-dimensional data can only be written as "gnuplot" or "standard" format.

Error -51: Unsuitable file type for SHELXL format

If the file format is set to "shelxl" to calculate the inverse Fourier of Patterson form SHELXL data, the file type must be set to either of "fobs" or "fcalc".
Check the 'type' command for erroneous input.

Error -52: Unsuitable input value for SHELXL format

If the file format is set to "shelxl" to calculate the inverse Fourier of Patterson form SHELXL data, the optional second parameter must be either of "fobs" or "fcalc". Check the input line for the 'form' command.

Error -53: Too many different layer types

You tried to define too many different layer types for a crystal with stacking faults.
If necessary modify the value of "ST_MAXTYPE" in file "stack.inc" and compile the program.

Error -54: Index outside limits

The value given is outside the proper limits allowed by this command. This usually means that an array element is outside the current dimension of an array, for example a correlation matrix or you are trying to include too many atoms in the crystal. Check the section on dimensions in the manual.

Error -55: No layers created at all

During the creation of the stacking faults, none of the layer types met the conditions imposed by the correlation matrix. Check the correlation matrix for proper values.

Error -56: Av. Transl. in plane of modulo vectors

The average translation vector, - calculated from the matrix of translation vectors and the correlation matrix or read via the 'aver' command is in the plane defined by the two modulo vector that were read using the 'modu' command. Use the 'show' command to check the values of the vectors. Correct the vectors by the 'aver', 'modu' commands.

Error -57: Av. Translation is zero

The average translation vector is a vector of length zero. Use the 'show' command to check the values of the vector. If the average vector is set explicitly by ==> 'set trans,fixed', then correct the vector by the 'aver' command. If the average vector is calculated from the translation and correlation matrix, the weighted sum of translation vectors is equal to zero. Use the 'show' command to check the values of the two matrices. Correct the values by the ==> 'trans', 'crow', 'ccol', 'cele' commands.

Error -58: No atoms written to file

No atoms were written to the plot file after the run command. Check the settings using the command 'show' and check the online help for further details. The most common problem is a too small value for 'thick'.

Error -59: Invalid color or type selected for atom

The color or type selected for plotting atoms is invalid. Check the online help for a list of valid numbers.

Error -60: Output value NOT allowed using lots

You calculated the Fourier using ==> 'lots'. In this case you can write intensities only as output.

Error -61: Too many additional generators

You tried to specify too many additional generators through the "generator" key word in the structure file. The maximum number of generators is specified in the file "gen_add.inc". If necessary, increase the value and compile DISCUS.

Error -62: Too many additional symmetry operators

You tried to specify too many additional symmetry operators through the "symmetry" key word in the structure file. The maximum number of symmetry operators is specified in the file "gen_add.inc". If necessary, increase the value and compile DISCUS.

Error -63: Molecule number outside limits

You tried to select a molecule number that is less than zero or higher than the number of molecules in the crystal. Or the upper limit of a range of numbers could be less than the lower limit.

Error -64: Molecule type outside limits

You tried to select a molecule type that is less than zero or higher than the number of molecule types in the crystal. Or the upper limit of a range of numbers could be less than the lower limit.

Error -65: Too many molecules created

You created more molecules than fit into the corresponding arrays. Check the size of the crystal or the limits of loops used to create the molecules. If necessary adjust the maximum number of molecules in the file "molecule.inc" and compile DISCUS.

Error -66: Too many molecules types created

You created more molecule types than fit into the corresponding arrays. Check the number of molecules in the crystal file or the limits of loops used to create the molecules. If necessary adjust the maximum number of molecule types in the file "molecule.inc" and compile DISCUS.

Error -67: Molecules have different number of atoms

You tried to swap two molecules of different length. This mode is possible for molecules of equal length only.

Error -68: Mode only available for molecules

Rotational waves are can act only onto molecules, not onto individual atoms

Error -69: Microdomains overlap

An atom was found that is inside at least two microdomains. The atom number and the microdomain numbers are written onto the screen. Check the distribution of the microdomains and the corresponding microdomain radii. Either the two microdomains are too close to each other or the radii of the microdomains are too large.

Error -70: Delta value must be in interval 0 -> 1

A wave function of type "box" was chosen. The optional second parameter gives the width of the lower half of the box in multiples of the wave length. This value must be in the range 0 -> 1. The values 0 and 1 are not allowed either.

Error -71: Too many points in direct space layer

The number of data points you selected with the 'rhona' and 'rhono' commands exceeds the maximum number of data points DISCUS can handle. The maximum value is defined in the variable MAXQXY in the file "config.inc". If necessary adjust this parameter and compile DISCUS again.

Error -72: Too many atom types in file

The number of different atom types in the structure file is too large. Note that DISCUS treats atoms of the same name but with different B values as different atom types. The maximum value is defined in the variable MAXSCAT in 'config.inc'. If necessary adjust this parameter and compile DISCUS again.

Error -73: Transform. requires too many generators

The old symmetry operations of the space group are always changed into additional generators and the new space group is P1. This allows you to generate the structure from the list of atoms in the old asymmetric unit without running into a conflict between the symmetry and the unit cell shape. If you have transformed the old unit cell into an oddly shaped or into a very large new cell, the old translations will be transformed into a very large number of generators.

Error -74: Molecule atom-number outside limits

The argument to the variable "mol_cont" is outside the number of atoms in this molecule, i.e. negative or too large. Check the number of atoms in this molecule through "mol_cont[i1,0], where <i1> is the number of the molecule. If you are reading a crystal structure, then the number of atoms is too large to fit into the space allocated for molecules. Adjust the parameter MOLE_MAX_ATOM in the file "molecule.inc"

Error -75:

No bond valence parameters for atom pair

Error -76: Function not available for molecules

The selected items for the program ATOMS must be individual atoms, not molecules.

Error -77: Unknown wave length symbol used

You used a wave length symbol unknown to DISCUS. Check for a typing error and consult the list of allowed wave length symbols in the help file through "help wavelength". The wave length of non listed radiations must be explicitly given as real value in Angstrom.

Error -78: Wrong optional parameter for HKLF4 format

Error -79: Too many atoms in result array

Through finding environments or similar operations too many atoms were found. Reduce the radius of the environment that you are searching. Alternatively increase the size of the variable "MAXPAR_RES" in file "param.inc"

Error -80: Angle between Normal and Abscissa is 0

The angle between the normal and the abscissa of the plot is zero degrees. Check the values you selected with the ==> 'uvw', 'hkl' and ==> 'abs' commands, and verify the angle in the ==> 'show' output.

Error -81: Invalid flag for space type

Commands like ==> 'projection' and the ==> 'function' "scalpro" take an optional flag that indicates whether the vectors are to be taken from direct or reciprocal space. Accordingly the flags are 2 to 4 characters long and may contain only the letters "d" and "r".

Error -82: Invalid molecule character

The character keyword for a molecule may only be "atom". If the molecule represents an extended object, used in small angle scattering calculations, it may be "cube", "cylinder", or "sphere". If the molecule represents a domain, it may be "cube", "cylinder", "sphere", or "fuzzy".

Error -83: Molecule buildup failed

DISCUS could not create the molecule. Check the molecule parameter lines in the structure file.

Error -84: Error reading molecule parameters

There seems to be a wrong keyword in the molecule entries.

Error -85: Sharpened patterson requires HKLF4 file

In order to calculate a sharpened Patterson function, the input file must be a SHELX HKLF4 file, i.e. contain h,k,l, I, sigma(i).

Error -86: Unknown import format

The import command was called with an unknown input format. Check the documentation on the ==> 'import" command for allowed values.

Error -87: Different lattice constants

You tried to include a domain into a host structure, whose lattice constants are different from those of the host crystal. Check the header of the domain file and the host structure for different metric. If necessary, ==> 'transform' the domain structure into the unit cell metric of the host crystal prior to the insertion.

Error -88: Bravais types differ

You tried to include a domain into a host structure, whose Bravais lattice type is different from that of the host crystal. Check the header of the domain file and the host structure for different space group symbols If necessary, ==> 'transform' the domain structure into the unit cell metric of the host crystal prior to the insertion.

Error -89: Unknown keyword in unit cell file

There is an unknown keyword in the unit cell file. See help entry data for a complete listing of allowed keywords.
Check the spelling of all keywords in the file. Is the first keyword identical to "title" ? Did you include the keyword "atoms" as last keyword of the header? Do all molecules end with "molecule end" ?

Error -90: Unknown diffractometer geometry

The powder diffraction module can handle Lorentz/Polarisation corrections for the following geometries: bragg Bragg-Brentano Geometry neutron A neutron Diffractometer in Debye-Scherrer geometry synchrotron Synchrotron

Error -91: Unexpected pseudoatom name read

DISCUS was reading a structure file in which the atom names stand for a cluster. the cluster read, does not occur in the list of defines cluster names.

Error -92: Error reading generators from structure

DISCUS was reading a structure file which contained a "generator" key word when the error occurred.
Check that there are all 13 required parameters, that there are no alphabetic characters etc.

Error -93: Unit cell constants <= zero

At least one of the unit cell constants given in the current structure or unit cell file is zero or less than zero.
Check the 'cell' line in the input file for these errors.

Error -94: Invalid domain descriptor in input file

This error occurred while reading the file that contains descriptions of domains. The file name was set by ==> 'input' within the ==> 'domain' menu. The first parameter of the "domain" keywords in this file must be "character" or "content" or "file" or "fuzzy" or "end" or blank for the first "domain" keyword.

Error -95: First domain keyword has parameters

This error occurred while reading the file that contains descriptions of domains. The file name was set by ==> 'input' within the ==> 'domain' menu. The first "domain" keyword must not have any parameters, and must be terminated by a "domain end" keyword, once all setting have been defined.

Error -96: Illegal keyword in domain input file

This error occurred while reading the file that contains descriptions of domains. The file name was set by ==> 'input' within the ==> 'domain' menu.
After the standard header, a domain input file may contain only domain descriptors. These start with a "domain" keyword and end with a "domain end" keyword". Valid specifiers within a domain are "POSI", "XAXI", "YAXI, "ZAXI", "CENT", "XDIM", "YDIM", "ZDIM"

Error -97: Atom type outside proper limits

The atom type number is less than zero, or larger than the current number of different atom types present in the crystal. Check the value, compare its value to n[2].

Error -98: No atom types exist at present

Wilson statistics requires that DISCUS knows which atom types were present in the sample, as a plot of ln(average(formfactor**2)/average(intensity)) versus sin(theta)/lambda is evaluated. Insert the atom types into a (dummy) structure and make sure that their relative amount is close to the actual sample composition.

Error -99: No wavelength has been set

Wilson statistics requires that DISCUS knows which wave length was used in the experiment, as a plot of ln(average(formfactor**2)/average(intensity)) versus sin(theta)/lambda is evaluated. Set the wave length through the ==> 'wvle' command in the ==> 'fourier' menu.

Error -100: Space group symbol missing in cell file

The required "spcgr" <symbol> line in the input file does not contain a space group symbol. Check the input file and provide a proper space group symbol.

Error -101: Dimension of lots < 0 or > than crystal

You are trying to use lots in the single crystal Fourier mode. At least one of the lots dimension is less than zero or larger than the crystal along that dimension. Check the line and inquire the crystal size with the ==> 'show cdim' command from the main menu.

Error -102: Property value outside defined range

While reading a cell file or structure, DISCUS encountered a property flag that is either zero or larger than currently defined. May be the input file was edited manually? Set the property flag to 1 to define the questionable atom as a standard atom type.

Error -103: Too many atoms per unit cell

The ==> 'aver' command in the ==> 'chem' menu experiences a situation in which there seem too many atom per unit cell. This error should never occur. Please document and report to the author.

Error -104: Powder output not defined as TTH or Q

You did a ==> 'powder' calculation without defining the axis as either ==> 'set axis,tth' or ==> 'set axis,q'. The output menu does not know which axis you intended. Please rerun the powder calculation with the correct axis set.

Error -105: Atom number outside limits

While reading an internally stored structure, the actual atom number is found to be larger that the maximum atom number stored in the header.
This error should never occur. Please document and report to the author.

Error -106: HKL steps for complete powder must be >0

All three step sizes for the HKL increment ==> 'set dh', 'set dk', 'set dl' are equal to zero! Make sure that at least one of these is set to a value larger than zero.

Error -107: 2Theta limits or step width are illegal

You are using the ==> 'set axis,tth' option and either: The upper limit of 2Theta is lesser than the lower limit, ==> 'set tthmin', ==> 'set tthmax' The 2Theta step size is zero or negative ==> 'set dtth'
Check the values and provide correct values.

Error -108: Q limits or step width are illegal

You are using the ==> 'set axis,q' option and either: The upper limit of Q is lesser than the lower limit, ==> 'set qmin', ==> 'set qmax' The Q step size is zero or negative ==> 'set dq' Check the values and provide correct values.

Error -109: Connectivity definition does not exist

You are referencing a connectivity definition by name or number, that DISCUS cannot find. Was this connectivity never defined, or are you using a definition with a different atom type?

Error -110: No connectivity definitions exist at all

You are referencing a connectivity definition by name or number, but no definitions exist. Did you forget to define connectivity definitions or did you read a new unit cell/structure after the connectivity was defined?

Error -111: Error reading SCAT instruction

Error -112:

DISCUS encountered an error while reading the "scat" line in the current input file. Check if there are non printable characters in this line or if the line is empty. Error reading ADP instruction
DISCUS encountered an error while reading the "apd" line in the current input file. Check if there are non printable characters in this line or if the line is empty.

Error -113: Could not find internal storage

DISCUS cannot find the internally saved structure. Check for typing differences between the save and read lines.

Error -114: Error allocating

DISCUS could not allocate the memory space for internal storage.
Does your computer offer enough memory space, have you written huge files into the internal storage?

Error -115: Different atom no on SCAT and ADP

DISCUS uses the names and values on the "scat" and "adp" lines in the cell/structure input file to assign atom types. As there is a one to one correspondence between the parameters on these two lines, the number of parameters on both lines must be identical.

Error -116: Could not find definition

You referenced a connectivity definition which does not exist. Check that the connectivity definition was added and that the connectivities were run. Check for spelling errors on all respective commands

Error -117: This DISCUS has NeXus support disabled

In order for DISCUS to be able to write NeXus files, it must be compiled and linked with the NeXus library. Get the NeXus development library, with all dependencies and build a new DISCUS with NeXus.

Error -118: No Fourier calculated yet, no output

Before DISCUS can write an output file you need to calculated a diffraction pattern. This could be a: single crystal pattern via ==> 'fourier' powder pattern via ==> 'powder' single crystal pattern vis stacking faults ==> 'stack' Patternson ==> 'patterson' inverse Fourier vie ==> 'inverse'

Error -119: Error reading atom number from RMCPROFILE

DISCUS encountered an error while reading the number of atoms from an RMCPROFILE *.cssr file. Check line 3 of the input file. DISCUS expects an integer number.

Error -120: Conn. Name is equal to variable name

You used a connectivity name that is already used as a variable name. This may lead to odd errors, as DISCUS would in most cases use the variable value instead of the connectivity name. You must choose a connectivity name that is not used as variable name.

Error -121: Error calculating x-position for powder

Error -122: Atom type number outside limits

Error -123: Atoms are at identical positions

While calculation a powder pattern DISCUS found a pair of atoms on identical positions.

Error -124: Powder output type wrong /= I, S(Q), F(Q)

Error -125: S(Q), F(Q) require Q-axis

Error -126: H-M symbol in CIF file is a question mark

The Herman Mauguin Symbol in the CIF file is a question mark. Please correct the space group in the structure file. Currently DISCUS does not interpret the symmetry operations listed in the CIF file

Error -127: Empty content file

One of the domain content files does not contain any atoms. Check the files to ensure that they all contain at least one atom. If an empty file is intended, set the error handling to 'live'. See 'set error, live'

Error -128: Could not add the new decoration

Error -129: No decoration definition exists yet

DISCUS was instructed to start decorating the crystal, yet no definitions for the decorations had been set. Use ==> 'add' to start a decoration type

Error -130: No surface atoms found

DISCUS could not find any of the surface atoms specified for the decoration model. Plot the crystal and inspect its surface, use ==> 'chem' to check the composition.

Error -131: No surface sites for ligands found

DISCUS could not find any of the surface atoms specified for the decoration model. Plot the crystal and inspect its surface, use ==> 'chem' to check the composition.

Error -132: Mismatch between corners and increments

You used 'na', 'no', 'nt' to specify the number of data points along an axis in reciprocal space. The corresponding corners are, however, identical. Adjust the number of data points along this axis to "1" or adjust the corners vie 'll', 'lr', 'ul', 'tl'.

Error -133: Refinement param index outside limits

You used a refinement parameter "ref_para[<index>]" with an index outside the dimension defined in the diffev section. Has DIFFEV been initialized properly? Did you use the DIFFEV command "run_mpi"? DID you use the DIFFEV "write kid" command ?

Error -134: Connectivity list has not been created

A connectivity list was referenced, but has never been created. Use the ==> 'connectivity' command "create" first.

Error -135: Did not find a connectivity for this atom

A connectivity list was referenced for an atom, but has never been created. Use the ==> 'connectivity' command "create" first.

Error -136: Connectivity name does not match

A connectivity list was referenced by its name but does not seem to exist. Use the ==> 'connectivity' command "create" first. Check for typing mismatched between the creation and usage.

Error -137: Atom number is outside crystal

An atom number was referenced that is either zero or negative or higher that the current number of atoms in the structure. Check for typing errors in the expression that is used to calculate the atom.

Error -138: Atom is already inside a molecule

The molecularizer detected that an atom is already part of a valid molecule. Currently any given Atom may be part of a single molecule only.

Error -139: Layer type outside limits

The ==> 'first' command in the stack menu allows to specify the type of the first layer. By default it is randomly chosen from all layer types. If you specify a number on the 'first' command, this number must be in the range [1: number_of_layers_specified] Check the number of ==> 'layer' commands and the value of the number on the 'first' command line.

Error -140: CSSR file not allowed for read cell

The RMCprofile cssr file will in all likelihood contain atoms from several unit cells. You have to read this as a "structure". If the file does indeed contain atoms just from a asymmetric unit, read the file as "structure" then save it as a "cell" file and expand this file to a full unit cell.

Error -141: Atoms are too close to each other

During a read/cell, DISCUS discovered that atoms are at identical places in a unit cell. Check the values for the atom coordinates in the cell file. There might be two atoms in the file that are separated by integer unit cell vectors.

Error -142: Cubeoctahedron only allowed in cubic systems.

A cobeoctahedron was specified as the form on the boundary command for a crystal that is not cubic. The cubeoctahedron is a special form for cubic crystals only. Unpredictable forms would result for other crystal systems.

Error -143: Wrong parameters for this bond type

The various bond models require a specific number of anchor atom types for the surface. normal 1 surface atom bridge 2 surface atom double 1 surface atom for each bond acceptor 1 surface atom donor 1 surface atom

Check if the current environment is properly named on the ==> "add" command within the decoration menu. Check the parameters on the 'bond' command.

Error -144: Too few atoms in ligand molecule

The bond or axis instruction within the decoration menu contained an atom number for the ligand molecule that is outside the range of atoms in the molecule. Check if an atom is missing in the ligand structure file or if the atom number on the bond/axis command is erroneous.

Error -145: Decoration name not recognized

The first parameter on the ==> 'decoration' 'set' command must be the name of one of the decoration schemes that were started with an 'add' command. add <name>, <deco_type> set <name>, ... For the current command the value of <name> differs from all names given on any 'add' command.

Error -146: No. of atoms not an integer multiple of sites

An attempt was made to create structure with the atoms in a unit cell sequence from an RMCprofile input file. The total number of atoms in the input file should have been an integer multiple of the number of sites per unit cell.
Check the RMCprofile setting that were used to create the structure. Another reason might be that the atoms are very disordered and DISCUS cannot find the correct number of sites per unit cell.

Error -147: Unknown export format

The export format that was given is not known. Check the command line for erroneous spelling.

Error -148: 1bar not at origin

A centrosymmetric structure that you want to export to SHELX must have its center of symmetry at the origin. Check the setting. If necessary shift the origin with the ==> 'transform' menu.

Error -149: Error reading OCC instruction

An error occurred while reading the OCC instruction in a crystal file. Check the input file for erroneous numbers or values.

Error -150: Occupancy outside [0:1]

An atom occupancy must be in the interval [0:1] Check all parameters and expressions from which the occupancy was derived.

Error -151: AND and DEFAULT simultaneously in isprop

The function 'isprop' takes arguments like: isprop('NMDOEIL') ! By default == AND isprop(AND:'NMDOEIL') isprop(OR:'NMDOEIL') isprop(AND:'NMDOEIL', OR:'NMDOEIL') The combination isprop('NMDOEIL', AND:'NMDOEIL') is illeagal, as the first argument without keyword corresponds to the AND condition as well.

Error -152: Interactive plot for JMOL only

Currently the only interactive plot program associated by the Discus-suite is JMOL. Linux users: Make sure you have JMOL and JAVA installed. Windows users, make sure you have the Java Runtime Environment JRE installed.

Error -153: Angle between Normal and Ordinate is 0

A plot was attempted with this angle at zero degrees. Check the setting for the normal, given by the ==> 'plot' 'uvw' or ==> 'plot' 'hkl' command and the ordinate ==> 'plot' 'ord'.

Error -154: Angle Ordinate to (Normal/Abscissa) is 0

A plot was attempted with this angle at zero degrees. Check the setting for the normal, given by the ==> 'plot' 'uvw' or ==> 'plot' 'hkl' command and the abscissa ==> 'plot' 'abs'.

Error -155: Coordination range is erroneous

For the user defined property like behavior, the coordination number limits must be given as nmin:<lower>, nmax:<upper> emin:<lower>, emax:<upper> For each of the pairs nmin/nmax and emin/emax the value of <lower> must be less or equal to the value of <upper>.

chem

Errors at the chemistry level of DISCUS

Error -1: Too many points for histogram

The number of points you specified for the histogram exceeds the dimensions of the array. Check whether you need this many points. If so enlarge the value of "chem_max_bin" in the file "chem.inc"

Error -2: Can't store all information in variable RES

The number of items produced by the command exceeds the dimension of the result array.

Error -3: Invalid range for bond-length histogram

The minimum value of the bond lengths is +0.01, and the maximum value (parameter #2) must be greater than the minimum value. Check the parameters on the ==>'set bl' command.

Error -4: Invalid SIGMA entered

The sigmas must all be larger than zero. Check the parameters on the ==>'set fs', 'set wsig' commands.

Error -5: Too many different atoms on ONE site

The number of different atoms found on one site in the unit cell exceeds the dimensions of the array.
You have to set the value of "chem_max_ave" to a larger value and compile DISCUS again.

Error -6: Atom name ALL not allowed for this command

The ==>'corr' command requires explicit atom names. You cannot use the atom name "ALL".

Error -7: Atoms need to be different

The ==>'corr' command requires the two atom types to be different. The self correlation of atom1 with itself is stored by the 'freq' command as well. If you know that there are no other atom types in the given direction, set the second atom type to "void".

Error -8: No neighboring atoms found

The 'freq' command found no neighboring atoms for the given distance and direction. Check the online help for more information on 'neig'.

Error -9: Invalid correlation vector index given

The given correlation vector index was invalid. Check your input.

Error -10: Invalid crystal site given

A invalid site number for the current crystal was entered. Check that the structure is loaded and your site number is valid.

Error -11: No correlations selected

Before you can calculate correlations you need to define the calculation more using the 'set neig' command in the 'chem' sublevel.

Error -12: Too many correlation definitions

The command 'set corr,add' was executed but there is no more space for the additional correlation definition. Use 'set corr,rese' to discard previous definitions or adjust the value of CHEM_MAX_COR in 'chem.inc' and recompile DISCUS.

Error -13: Correlation direction invalid

The entered direction for the calculation of the displacement correlations is invalid (e.g. length zero), check your input.

Error -14: Too many displacement directions

You have entered too many displacement directions for the calculation of displacement correlations. Use 'set dir,rese' to discard the previous input our adjust the parameter CHEM_MAX_DIR in 'chem.inc' and recompile DISCUS.

Error -15: No displacement directions selected

You have tried to calculate displacement correlations but there are no directions defined that are used to obtain the displacement. Use the command 'set dir,..' to enter the desired values.

Error -16: Failed to apply periodic boundaries

DISCUS is not able to apply periodic crystal boundaries (should not happen).

Error -17: Correlation fields require same mode

To calculate correlation fields, all neighbor definitions must be made using the same mode, i.e. 'dist' or 'vec'. Redefine the neighbors to be used via the command 'set neig'. This condition is only valid for 2-dimensional correlation fields.

Error -18: Correlation fields require same # vectors

If working with neighboring vector definitions to determine a 2D correlation field, the number of vectors must be the same for both directions.

Error -19: No neighboring molecules found

For the current neighbor definitions no neighboring molecules were found. Check your input.

Error -20: No molecules present in crystal

You cannot use the 'molecule' mode without having at least one molecule within the model crystal. Check the manual how to define molecules.

Error -21: Molecule types need to be different

In order to execute the selected operation (e.g. determine correlations), the selected molecule types need to be different. Check your input.

Error -22: Command not available in molecule mode

This particular command (e.g. aver) is not available in 'molecule' mode. Switch to 'atoms' mode using the command 'mode atom' and reenter the desired command.

Error -23: Too many neighboring atoms/molecules

The current operation has found too many neighboring atoms or molecules. Either restrict the neighbor definitions of adjust the variable CHEM_MAX_NEIG in the file 'config.inc' and recompile DISCUS.

Error -24: Invalid correlation angle index given

You tried to reference an environment of "angle" type that has not been defined. Check the list of defined environments and the values of parameters in the current command line.

Error -25: Invalid range for bond-angle histogram

The upper and lower limits of the bond angle histogram are invalid. Check that none is lower than 0 degrees or larger than 180 degrees, and that the upper limit is larger than the lower.

Error -26: Invalid correlation environment index given

You tried to reference an environment of "correlation" type that has not been defined. Check the list of defined environments and the values of parameters in the current command line.

Error -27: No atoms present in crystal

A number of commands can only be used if the crystal contains any atoms. Check the number of atoms "n[1] present in your crystal.

Error -28: Invalid correlation conn index given

The given correlation conn index was invalid. Check your input. The index must be larger than 0.

Error -29: Atom type outside valid range

Error -30: Multiple identical sites in unit cell

Several sites exist within the unit cell with (almost) identical coordinates. Thus DISCUS has a tough time to decide which of these sites you have in mind. The symmetry expansion is carried out, but the results are written to file only. This gives you a better chance to sort them out manually.

four

These are the possible error messages related to the calculation of the Fourier Transform

Error -2: Invalid lot shape selected

The only two supported lot shapes are 'eli' and 'box' ! Select a proper shape.

Error -3: SIN(THETA)/LAMBDA $>$ lookup table limits

To speed up the calculation of the Fourier Transform, DISCUS uses a lookup table for SIN(THETA)/LAMBDA. You need to adjust the maximum h,k,l or the wave length. Alternatively, the values CFPKT and CFINC in the file 'config.inc' might be adjusted to the specific needs and DISCUS recompiled.

Error -4: Component of increment vector is zero

The corners of the plane in reciprocal space to be calculated are invalid since two or more corners are the same. Adjust your input.

mmc

These are the error messages of the MC section of DISCUS

Error -1: Invalid or no energy type selected

Before running a MC simulation, a valid energy definition (e.g. occupational correlations or distortions) must be selected using the 'set energy' command.

Error -2: No valid move after 1000 cycles

After 1000 cycles NO valid MC move was generated and DISCUS has terminated the simulation. Check your input.

Error -3: Invalid mode selected for COCC MC run

Occupational ordering requires the SWCHEM mode to be selected. Use the command 'set mode' to select the correct mode and restart the simulation.

Error -4: Number of MC cycles is zero

A Monte Carlo run was tried with zero number of refinement cycles. The number of cycles must be at least equal to one. Check the parameters on the last ==> 'set cycle' command, check that a ==> 'set cycle' command was used at all.

Error -5: Number of feedback intervals is zero

The feedback interval must be at least equal to one. Check the parameters on the last ==> 'set feed' command. check that a ==> 'set feed' command was used at all.

pdf

Errors at the PDF Level

Error -1: Too many points in PDF

The specified range and grid size for the PDF results in too many points. Either adjust your input or change the parameter MAXDAT in 'config.inc' and recompile DISCUS.

Error -2: Cannot extend r-range for corr. convolution

In order to carry out the convolution with the SINC function defined by the value of Q(max), the calculation range for the PDF needs to be extended. This error message occurs when there is not enough space for the additional data points. Either adjust your input or change the parameter MAXDAT in 'config.inc' and recompile DISCUS.

Error -3: Crystal too large to create periodic bound.

The size of the crystal is too large for the periodic boundary array DISCUS uses for rapid calculation of the PDF. Either use a smaller model crystal or change the parameter MAXBND in 'config.inc' and recompile DISCUS.

Error -4: No structure defined yet (>= 1 atoms)

You do not have a crystal structure with at least one atom. Create a structure first.

Error -5: PDF data must start with r=Dr

The data in the input file must be evenly spaced with step size ==> 'set range'. The first data point must be at a distance value r equal to this set size.
Check the values on the ==> 'set range' command and check the data file. If necessary repeat the determination of the experimental PDF or extrapolate the data from the current minimum r-value down to r(min) = Dr.

Error -6: PDF range fixed with data loaded

Error -7: Disable Gaussian mode and recalculate

Error -8: Atom type ALL not allowed

The ==> 'set partial' command must receive individual atom names, not the atom name "ALL".

Error -9: No atoms in asymmetric unit

The crystal does not contain any atoms in the asymmetric unit. It may have been read with the 'structure' command or generated via stacking faults. Under these circumstances you must use the exact mode in connection with the ==> 'set bound'

Error -10: User Fit minimum outside data range

You specified a fit range via ==> 'frange' that is outside the distance range of the data set. Check your data set and the 'frange' command and ensure consistency.

Error -11: User Fit maximum outside data range

You specified a fit range via ==> 'frange' that is outside the distance range of the data set. Check your data set and the 'frange' command and ensure consistency.

rmc

Errors at the Reverse Monte Carlo Level

Error -1: Too many experimental data planes

This message appears if the user tries to read an additional plane of experimental data and the maximal number of planes is exceeded. If that many data planes are really needed, the variable RMC_MAX_PLANES has to be adjusted in the file 'config.inc' and DISCUS must be recompiled.

Error -2: Too many experimental data points

The experimental data file contains too many data points. In order to be able to use the particular data file the variable RMC_MAX_Q must be altered in 'config.inc' and DISCUS must be recompiled.

Error -3: No atoms present in model crystal

The command 'run' was entered but there is no model crystal present. The starting structure has to be set up before entering the RMC sublevel.

Error -4: No experimental data present

The command 'run' was entered but no experimental data are present. Use the command 'data' to read the experimental data.

Error -5: Invalid method (x,n) selected

When using the 'data' command an invalid scattering method was used. The present version of the program recognizes "neutron" and "x-ray" data. The method parameter can be abbreviated to a single letter. The input is not case sensitive.

Error -7: Invalid plane selected

An invalid data plane was selected. Use the command 'show' for a list of loaded data planes.

Error -8: Invalid symmetry number selected

The selected symmetry number is invalid. Use the command 'show' for a list of the current settings.

Error -9: Invalid RMC mode selected

An invalid RMC mode was selected. Check the online help entry for 'set mode' in the RMC sublevel for a list of allowed RMC moves.

Error -10: No atom types selected for RMC run

No atoms were selected for the RMC refinement. Use the command 'sele' to select the atom types to be used for the RMC run.

Error -11: Only ONE atom type present in SWCHEM mode

The RMC mode SWCHEM requires two different atom types to be selected. Vacancies are to be selected as VOID.

Error -12: Displacements too small for SWDISP mode

The displacements present in the starting structure are too small to use the SWDISP mode. The limit is set in 'rmc.f' if there is any need to change that.

Error -13: Too many symmetrically equivalent planes

The number of symmetrically equivalent planes exceeds the limit set by RMC_MAX_SYM in 'config.inc'. You can either adjust that value and recompile DISCUS or set the space group symbol in the structure file to P1 if you don't want to use the symmetry information.

Error -14: No experimental data within given q limits

There are no experimental data in the gives q limits. Check the current settings with 'show' and adjust the limits using the command 'set range' in the RMC sublevel.

Error -15: Invalid data type selected

DISCUS can read PGM and NIPL files as input data. You selected a non supported data type. Check the online help for 'data' for a description of the valid data formats.

Error -16: Invalid weighting scheme or weighting file

The entered weighting scheme is invalid. Check the online help for 'data' for a description of the valid weighting schemes.

Error -17: Data and weight file have different sizes

If you read weights from a file, data file and weight file MUST have the same size.

Error -18: Data file is not an ASCII PGM file

In the present version, DISCUS can only read ASCII PGM files with the code P2 in the first line. Use the tool 'pnmnoraw' to transform from a binary to an ASCII version of your input file. The tool 'pnmnoraw' is part of the 'netpbm' package.

Error -19: Invalid constrain entered

The entered constrain is invalid. Check the online help for more information.

Error -20: No valid move after 1000 display intervals

After 1000 display intervals, DISCUS has not generated a single valid move. Check the settings for minimal allowed atom distances and all other RMC settings.

Error -21: Number of LOTS exceeds maximum

The number of 'lots' (crystal volumes to be averaged) exceeds the maximum number RMC_MAX_LOTS defined in 'config.inc'.

A.72   space_groups

List of allowed Space groups

a) Standard space groups settings

```
#   1 : P1          #   2 : P-1
#   3 : P2          #   4 : P21         #   5 : C2          #  6 : Pm
#   7 : Pc          #   8 : Cm          #   9 : Cc          #  10 : P2/m
#  11 : P21/m       #  12 : C2/m        #  13 : P2/c        #  14 : P21/c
#  15 : C2/c
#  16 : P222        #  17 : P2221       #  18 : P21212      #  19 : P212121
#  20 : C2221       #  21 : C222        #  22 : F222        #  23 : I222
#  24 : I212121     #  25 : Pmm2        #  26 : Pmc21       #  27 : Pcc2
#  28 : Pma2        #  29 : Pca21       #  30 : Pnc2        #  31 : Pmn21
#  32 : Pba2        #  33 : Pna21       #  34 : Pnn2        #  35 : Cmm2
#  36 : Cmc21       #  37 : Ccc2        #  38 : Amm2        #  39 : Abm2=Aem2
#  40 : Ama2        #  41 : Aba2=Aea2   #  42 : Fmm2        #  43 : Fdd2
#  44 : Imm2        #  45 : Iba2        #  46 : Ima2        #  47 : Pmmm
#  48 : Pnnn        #  49 : Pccm        #  50 : Pban        #  51 : Pmma
#  52 : Pnna        #  53 : Pmna        #  54 : Pcca        #  55 : Pbam
#  56 : Pccn        #  57 : Pbcm        #  58 : Pnnm        #  59 : Pmmn
#  60 : Pbcn        #  61 : Pbca        #  62 : Pnma        #  63 : Cmcm
#  64 : Cmca=Cmce   #  65 : Cmmm        #  66 : Cccm        #  67 : Cmma=Cmme
#  68 : Ccca=Ccce   #  69 : Fmmm        #  70 : Fddd        #  71 : Immm
#  72 : Ibam        #  73 : Ibca        #  74 : Imma
#  75 : P4          #  76 : P41         #  77 : P42         #  78 : P43
#  79 : I4          #  80 : I41         #  81 : P-4         #  82 : I-4
#  83 : P4/m        #  84 : P42/m       #  85 : P4/n        #  86 : P42/n
#  87 : I4/m        #  88 : I41/a       #  89 : P422        #  90 : P4212
#  91 : P4122       #  92 : P41212      #  93 : P4222       #  94 : P42212
#  95 : P4322       #  96 : P43212      #  97 : I422        #  98 : I4122
#  99 : P4mm        # 100 : P4bm        # 101 : P42cm       # 102 : P42nm
# 103 : P4cc        # 104 : P4nc        # 105 : P42mc       # 106 : P42bc
# 107 : I4mm        # 108 : I4cm        # 109 : I41md       # 110 : I41cd
```

```
# 111 : P-42m        # 112 : P-42c        # 113 : P-421m      # 114 : P-421c
# 115 : P-4m2        # 116 : P-4c2        # 117 : P-4b2       # 118 : P-4n2
# 119 : I-4m2        # 120 : I-4c2        # 121 : I-42m       # 122 : I-42d
# 123 : P4/mmm       # 124 : P4/mcc       # 125 : P4/nbm      # 126 : P4/nnc
# 127 : P4/mbm       # 128 : P4/mnc       # 129 : P4/nmm      # 130 : P4/ncc
# 131 : P42/mmc      # 132 : P42/mcm      # 133 : P42/nbc     # 134 : P42/nnn
# 135 : P42/mbc      # 136 : P42/mnm      # 137 : P42/nmc     # 138 : P42/ncm
# 139 : I4/mmm       # 140 : I4/mcm       # 141 : I41/amd     # 142 : I41/acd
# 143 : P3
# 144 : P31                               # 145 : P32
# 146 : R3: hexagonal axes                # 147 : P-3
# 148 : R-3: hexagonal axes               # 149 : P312
# 150 : P321                              # 151 : P3112
# 152 : P3121                             # 153 : P3212
# 154 : P3221                             # 155 : R32: hexagonal axes
# 156 : P3m1                              # 157 : P31m
# 158 : P3c1                              # 159 : P31c
# 160 : R3m: hexagonal axes               # 161 : R3c: hexagonal axes
# 162 : P-31m                             # 163 : P-31c
# 164 : P-3m1                             # 165 : P-3c1
# 166 : R-3m: hexagonal axes              # 167 : R-3c: hexagonal axes
# 168 : P6           # 169 : P61          # 170 : P65         # 171 : P62
# 172 : P64          # 173 : P63          # 174 : P-6         # 175 : P6/m
# 176 : P63/m        # 177 : P622         # 178 : P6122       # 179 : P6522
# 180 : P6222        # 181 : P6422        # 182 : P6322       # 183 : P6mm
# 184 : P6cc         # 185 : P63cm        # 186 : P63mc       # 187 : P-6m2
# 188 : P-6c2        # 189 : P-62m        # 190 : P-62c       # 191 : P6/mmm
# 192 : P6/mcc       # 193 : P63/mcm      # 194 : P63/mmc
# 195 : P23          # 196 : F23          # 197 : I23         # 198 : P213
# 199 : I213         # 200 : Pm-3         # 201 : Pn-3        # 202 : Fm-3
# 203 : Fd-3         # 204 : Im-3         # 205 : Pa-3        # 206 : Ia-3
# 207 : P432         # 208 : P4232        # 209 : F432        # 210 : F4132
# 211 : I432         # 212 : P4332        # 213 : P4132       # 214 : I4132
# 215 : P-43m        # 216 : F-43m        # 217 : I-43m       # 218 : P-43n
# 219 : F-43c        # 220 : I-43d        # 221 : Pm-3m       # 222 : Pn-3n
# 223 : Pm-3n        # 224 : Pn-3m        # 225 : Fm-3m       # 226 : Fm-3c
# 227 : Fd-3m        # 228 : Fd-3c        # 229 : Im-3m       # 230 : Ia-3d
```

b) non standard settings of space groups

```
# 231 : P112:        #   3: P2 : unique c axis
# 232 : P1121:       #   4: P21 : unique c axis
# 233 : A121:        #   5: C2 : unique b axis: cell choice 2
# 234 : I121:        #   5: C2 : unique b axis: cell choice 3
# 235 : A112:        #   5: C2 : unique c axis: cell choice 1
# 236 : B112:        #   5: C2 : unique c axis: cell choice 2
# 237 : I112:        #   5: C2 : unique c axis: cell choice 3
# 238 : P11m:        #   6: Pm : unique c axis
# 239 : P1n1:        #   7: Pc : unique b axis: cell choice 2
# 240 : P1a1:        #   7: Pc : unique b axis: cell choice 3
# 241 : P11a:        #   7: Pc : unique c axis: cell choice 1
# 242 : P11n:        #   7: Pc : unique c axis: cell choice 2
# 243 : P11b:        #   7: Pc : unique c axis: cell choice 3
# 244 : A1m1:        #   8: Cm : unique b axis: cell choice 2
# 245 : I1m1:        #   8: Cm : unique b axis: cell choice 3
# 246 : A11m:        #   8: Cm : unique c axis: cell choice 1
# 247 : B11m:        #   8: Cm : unique c axis: cell choice 2
# 248 : I11m:        #   8: Cm : unique c axis: cell choice 3
# 249 : A1n1:        #   9: Cc : unique b axis: cell choice 2
# 250 : I1a1:        #   9: Cc : unique b axis: cell choice 3
# 251 : A11a:        #   9: Cc : unique c axis: cell choice 1
# 252 : B11n:        #   9: Cc : unique c axis: cell choice 2
# 253 : I11b:        #   9: Cc : unique c axis: cell choice 3
# 254 : P112/m:      #  10: P2/m : unique c axis
```

```
# 255 : P1121/m:    #  11: P21/m : unique c axis
# 256 : A12/m1:     #  12: C2/m : unique b axis: cell choice 2
# 257 : I12/m1:     #  12: C2/m : unique b axis: cell choice 3
# 258 : A112/m:     #  12: C2/m : unique c axis: cell choice 1
# 259 : B112/m:     #  12: C2/m : unique c axis: cell choice 2
# 260 : I112/m:     #  12: C2/m : unique c axis: cell choice 3
# 261 : P12/n1:     #  13: P2/c : unique b axis: cell choice 2
# 262 : P12/a1:     #  13: P2/c : unique b axis: cell choice 3
# 263 : P112/a:     #  13: P2/c : unique c axis: cell choice 1
# 264 : P112/n:     #  13: P2/c : unique c axis: cell choice 2
# 265 : P112/b:     #  13: P2/c : unique c axis: cell choice 3
# 266 : P121/n1:    #  14: P21/c : unique b axis: cell choice 2
# 267 : P121/a1:    #  14: P21/c : unique b axis: cell choice 3
# 268 : P1121/a:    #  14: P21/c : unique c axis: cell choice 1
# 269 : P1121/n:    #  14: P21/c : unique c axis: cell choice 2
# 270 : P1121/b:    #  14: P21/c : unique c axis: cell choice 3
# 271 : A12/n1:     #  15: C2/c : unique b axis: cell choice 2
# 272 : I12/a1:     #  15: C2/c : unique b axis: cell choice 3
# 273 : A112/a:     #  15: C2/c : unique c axis: cell choice 1
# 274 : B112/n:     #  15: C2/c : unique c axis: cell choice 2
# 275 : I112/b:     #  15: C2/c : unique c axis: cell choice 3
# 276 : Pnnn:       #  48: origin choice 2
# 277 : Pban:       #  50: origin choice 2
# 278 : Pmmn:       #  59: origin choice 2
# 279 : Ccca:       #  68: origin choice 2
# 280 : Fddd:       #  70: origin choice 2
# 281 : P4/n:       #  85: origin choice 2
# 282 : P42/n:      #  86: origin choice 2
# 283 : I41/a:      #  88: origin choice 2
# 284 : P4/nbm:     # 125: origin choice 2
# 285 : P4/nnc:     # 126: origin choice 2
# 286 : P4/nmm:     # 129: origin choice 2
# 287 : P4/ncc:     # 130: origin choice 2
# 288 : P42/nbc:    # 133: origin choice 2
# 289 : P42/nnn:    # 134: origin choice 2
# 290 : P42/nmc:    # 137: origin choice 2
# 291 : P42/ncm:    # 138: origin choice 2
# 292 : I41/amd:    # 141: origin choice 2
# 293 : I41/acd:    # 142: origin choice 2
# 294 : R3:         # 146 rhombohedral axes
# 295 : R-3:        # 148 rhombohedral axes
# 296 : R32:        # 155 rhombohedral axes
# 297 : R3m:        # 160 rhombohedral axes
# 298 : R3c:        # 161 rhombohedral axes
# 299 : R-3m:       # 166 rhombohedral axes
# 300 : R-3c:       # 167 rhombohedral axes
# 301 : Pn-3:       # 201: origin choice 2
# 302 : Fd-3:       # 203: origin choice 2
# 303 : Pn-3n:      # 222: origin choice 2
# 304 : Pn-3m:      # 224: origin choice 2
# 305 : Fd-3m:      # 227: origin choice 2
# 306 : Fd-3c:      # 228: origin choice 2
# 307 : Pbnm:       #  62: Pnma non-standard orientation
# 308 : Pmnn:       #  58: Pnnm non-standard orientation
```

Non-standard settings of all orthorhombic space groups are recognized either via the spcgr line in the form: spcgr Pnma, setting:cab or by using the explicit non-standard space group name:

```
        abc       bac       cab       cba       bca       acb
#  16   P222      P222      P222      P222      P222      P222
#  17   P2221     P2221     P2122     P2122     P2212     P2212
#  18   P21212    P21212    P22121    P22121    P21221    P21221
#  19   P212121   P212121   P212121   P212121   P212121   P212121
```

```
#  20   C2221    C2221    A2122    A2122    B2212    B2212
#  21   C222     C222     A222     A222     B222     B222
#  22   F222     F222     F222     F222     F222     F222
#  23   I222     I222     I222     I222     I222     I222
#  24   I212121  I212121  I212121  I212121  I212121  I212121
```

abc bac cab cba bca acb

```
#  25   Pmm2     Pmm2     P2mm     P2mm     Pm2m     Pm2m
#  26   Pmc21    Pcm21    P21ma    P21am    Pb21m    Pm21b
#  27   Pcc2     Pcc2     P2aa     P2aa     Pb2b     Pb2b
#  28   Pma2     Pbm2     P2mb     P2cm     Pc2m     Pm2a
#  29   Pca21    Pbc21    P21ab    P21ca    Pc21b    Pb21a
#  30   Pnc2     Pcn2     P2na     P2an     Pb2n     Pn2b
#  31   Pmn21    Pnm21    P21mn    P21nm    Pn21m    Pm21n
#  32   Pba2     Pba2     P2cb     P2cb     Pc2a     Pc2a
#  33   Pna21    Pbn21    P21nb    P21cn    Pc21n    Pn21a
#  34   Pnn2     Pnn2     P2nn     P2nn     Pn2n     Pn2n
#  35   Cmm2     Cmm2     A2mm     A2mm     Bm2m     Bm2m
#  36   Cmc21    Ccm21    A21ma    A21am    Bb21m    Bm21b
#  37   Ccc2     Ccc2     A2aa     A2aa     Bb2b     Bb2b
#  38   Amm2     Bmm2     B2mm     C2mm     Cm2m     Am2m
#  39   Aem2     Bme2     B2em     C2me     Cm2e     Ae2m
#  40   Ama2     Bbm2     B2mb     C2cm     Cc2m     Am2a
#  41   Aea2     Bbe2     B2eb     C2ce     Cc2e     Ae2a
#  42   Fmm2     Fmm2     F2mm     F2mm     Fm2m     Fm2m
#  43   Fdd2     Fdd2     F2dd     F2dd     Fd2d     Fd2d
#  44   Imm2     Imm2     I2mm     I2mm     Im2m     Im2m
#  45   Iba2     Iba2     I2cb     I2cb     Ic2a     Ic2a
#  46   Ima2     Ibm2     I2mb     I2cm     Ic2m     Im2a
        abc      bac      cab      cba      bca      acb
#  47   Pmmm     Pmmm     Pmmm     Pmmm     Pmmm     Pmmm
#  48   Pnnn     Pnnn     Pnnn     Pnnn     Pnnn     Pnnn
#  49   Pccm     Pccm     Pmaa     Pmaa     Pbmb     Pbmb
#  50   Pban     Pban     Pncb     Pncb     Pcna     Pcna
#  51   Pmma     Pmmb     Pbmm     Pcmm     Pmcm     Pmam
#  52   Pnna     Pnnb     Pbnn     Pcnn     Pncn     Pnan
#  53   Pmna     Pnmb     Pbmn     Pcnm     Pncm     Pman
#  54   Pcca     Pccb     Pbaa     Pcaa     Pbcb     Pbab
#  55   Pbam     Pbam     Pmcb     Pmcb     Pcma     Pcma
#  56   Pccn     Pccn     Pnaa     Pnaa     Pbnb     Pbnb
#  57   Pbcm     Pcam     Pmca     Pmab     Pbma     Pcmb
#  58   Pnnm     Pnnm     Pmnn     Pmnn     Pnmn     Pnmn
#  59   Pmmn     Pmmn     Pnmm     Pnmm     Pmnm     Pmnm
#  60   Pbcn     Pcan     Pnca     Pnab     Pbna     Pcnb
#  61   Pbca     Pcab     Pbca     Pcab     Pbca     Pcab
#  62   Pnma     Pmnb     Pbnm     Pcmn     Pmcn     Pnam
#  63   Cmcm     Ccmm     Amma     Amam     Bbmm     Bmmb
#  64   Cmce     Ccme     Aema     Aeam     Bbem     Bmeb
#  65   Cmmm     Cmmm     Ammm     Ammm     Bmmm     Bmmm
#  66   Cccm     Cccm     Amaa     Amaa     Bbmb     Bbmb
#  67   Cmme     *Cmmb    Aemm     *Acmm    Bmem     *Bmam
#  68   Ccce     *Cccb    Aeaa     *Acaa    Bbeb     *Bbab
#  69   Fmmm     Fmmm     Fmmm     Fmmm     Fmmm     Fmmm
#  70   Fddd     Fddd     Fddd     Fddd     Fddd     Fddd
#  71   Immm     Immm     Immm     Immm     Immm     Immm
#  72   Ibam     Ibam     Imcb     Imcb     Icma     Icma
#  73   Ibca     Icab     Ibca     Icab     Ibca     Icab
#  74   Imma     Immb     Ibmm     Icmm     Imcm     Imam
```

Note: for space groups 67 Cmme and 68 Ccce, the settings bac, cba and acb are given in the traditional name instead of using the 'e' glide plane name. This choice allows you to distinguish these settings.

A.73   wavelength

```
List of allowed wavelength symbols
TIA1  2.748410   TIA2  2.752070   TIB   2.513810   TI    2.740000
CRA1  2.28970    CRA2  2.29361    CRB   2.0848      CR    2.2909
FEA1  1.193604   FEA2  1.93998    FEB   1.75653    FE    1.9373
COA1  1.788920   COA1  1.792780   COB   1.620750   CO    1.780000
CUA1  1.54056    CUA2  1.54439    CUB   1.39217    CU    1.5418
MOA1  0.709309   MOA2  0.71335    MOB   0.63225    MO    0.7107
AGA1  0.55941    AGA2  0.56380    AGB   0.49701    AG    0.5608
TAA1  0.215484   TAA2  0.220290   TAB   0.190076   TA    0.560800
WA1   0.208992   WA2   0.213813   WB    0.184363,  W     0.560800
AUA1  0.180185   AUA2  0.185064   AUB   0.158971,  AU    0.560800
```

If the last two characters of the wave length symbol are "12", the powder pattern are calculated as the sum of Kalpha1 + Kalpha2

A.74   atom_names

```
List of allowed atom names
H    H.   H1-  D.   HE   LI   LI1  BE   BE2+ B    C    N    O    O1-
O2-  F    F1-  NE   NA   NA1+ MG   MG2+ AL   AL3+ SI   SI.  SI4+ S
P    CL   CL1- AR   K    K1+  CA   CA2+ SC   SC3+ TI   TI2+ TI3+ TI4+
V    V2+  V3+  V5+  CR   CR2+ CR3+ MN   MN2+ MN3+ MN4+ FE   FE+  FE3+
CO   CO2+ CO3+ NI   NI2+ NI3+ CU   CU1+ CU2+ ZN   ZN2+ GA   GA3+ GE
GE4+ AS   SE   BR   BR1- KR   RB   RB1+ SR   SR2+ Y    Y3+  ZR   ZR4+
NB   NB3+ NB5+ MO   MO3+ MO5+ MO6+ TC   RU   RU3+ RU+  RH   RH3+ RH4+
PD   PD2+ PD4+ AG   AG1+ AG2+ CD   CD2+ IN   IN3+ SN   SN2+ SN4+ SB
SB3+ SB5+ TE   I    I1-  XE   CS   CS1+ BA   BA2+ LA   LA3+ CE   CE3+
CE4+ PR   PR3+ PR4+ ND   ND3+ PM   PM3+ SM   SM3+ EU   EU2+ EU3+ GD
GD3+ TB   TB3+ DY   DY3+ HO   HO3+ ER   ER3+ TM   TM3+ YB   YB2+ YB3+
LU   LU3+ HF   HF4+ TA   TA5+ W    W6+  RE   OS   OS4+ IR   IR3+ IR4+
PT   PT2+ PT4+ AU   AU1+ AU3+ HG   HG1+ HG2+ TL   TL1+ TL3+ PB   PB2+
PB4+ BI   BI3+ BI5+ PO   AT   RN   FR   RA   RA2+ AC   AC3+ TH   TH4+
PA   U    U3+  U4+  U6+  NP   NP3+ NP4+ NP6+ PU   PU3+ PU4+ PU6+ AM
CM   BK   CF
```

The symbols ending with a period have special meanings as listed:

```
H.   Hydrogen HF scattering factor
C.   Carbon HF scattering factor
SI.  Silicon HF scattering factor
D.   Deuterium (for neutron diffraction only)
O2-  Oxygen, taken from Acta Cryst. 19, p 486 (1965).
```

Other special names:

```
E1-  Single electron with form factor =1 independent of sin( )/
```

Other names may be used as well, if scattering curves are supplied for these names. See the 'delf, and 'scat' commands at 'fourier' sublevel for details.

Bibliography

M.I. Aroyo; A.J.C. Wilson; U. Shmuelli; T. Hahn. *International Tables for Crystallography* . Dordrecht, Holland 6 edition 2016.

A.H. Barnett; J.F. Magland; L. af Klinteberg. A parallel non-uniform fast fourier transform library based on an "exponential of semicircle" kernel. SIAM J.Sci:Comput. **41** (2019) C479–C504.

N.E. Brese; M. O'Keeffe. Bond-valence Parameters for Solids. Acta Cryst. **B47** (1991) 192–197.

I.D. Brown; D. Altermatt. Bond-valence Parameters Obtained from a Systematic Analysis of the Inorganic Crystal Structure Database. . Acta Cryst. **B41** (1985) 244–247.

B.D. Butler; T.R. Welberry. Calculation of Diffuse Scattering from Simulated Disordered Crystals: a Comparison with Optical Transforms . J. Appl. Cryst. **25** (1992) 391–399.

D. Chernyshov; V. Dyadin; H. Emerich; G.Valkovskiy; Ch.J. McMonagie; W. van Beek. On the resolution function for powder diffraction with area detectors. Acta Cryst. **A77** (2021) 497–505.

K. Fourmont. PhD thesis Univ. Muenster, Germany 1999.

R.L. McGreevy; L. Pusztai. Reverse Monte Carlo Simulation: a New Technique for the Determination of Disordered Structures . Mol. Simul. **1** (1988) 359–367.

N. Metropolis; A.W. Rosenbluth; M.N Rosenbluth; A.H Teller; E.J. Teller. Equation of State Calculations by Fast Computing Machines . J. Chem. Phys. **21** (1953) 1087–1094.

R.B. Neder; Th. Proffen. *Diffuse scattering and defect structure simulations - A cook book using the program DISCUS* . Oxford University Press, Oxford, UK 2007.

R.B. Neder; Th. Proffen. Exact and fast calculation of the X-ray pair distribution function. J. Appl. Cryst. **53** (2021) 710–721.

V.M. Nield; D.A. Keen; R.L. McGreevy. The Interpretation of Single Crystal Diffuse Scattering using Reverse Monte Carlo Modelling . Acta Cryst. A **51** (1995) 763–771.

Th. Proffen; R.B. Neder. DISCUS, a Program for Diffuse Scattering and Defect Structure Simulations . J. Appl. Cryst. **30** (1997) 171–175.

D.E. Sands. *Vectors and Tensors in Crystallography* . Dover Publications Inc. 1 edition 1995.

B. H. Toby; R.B. von Dreele. GSAS-II: the genesis of a modern open-source all purpose crystal-lography software package. J. Appl. Cryst. **46** (2013) 544–549.

T.R. Welberry.   Diffuse X-ray Scattering and Models of Disorder . Rep. Prog. Phys. **48** (1985) 1543–1593.