Users guide

D I S C U S

Version 3.2

Reinhard Neder

Institut für Mineralogie
Universität Würzburg
Am Hubland
D-97074 Würzburg
Germany
Email: *reinhard.neder@mail.uni-wuerzburg.de*

Thomas Proffen

Department of Physics and Astronomy
Michigan State University
East Lansing, MI 48824-1116
USA
Email: *proffen@pa.msu.edu*

Document created March 4, 1999

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.

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

Acknowledgments

The list of structure factors was adapted from the program Lazy-Pulverix written by K. Yvon, W. Jeitschko & E. Parthe. The random number generators and interpolations routines were taken from the Numerical Recipes by Press, Flannery, Teukolsky & Vetterling, Cambridge University Press, 1989. The routines for the line editing was taken from the program GNUPLOT.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is DISCUS ?

DISCUS [18] 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 which includes a specially suited plotting program *KUPLOT* and a new program *PDFFIT* [17] which allows the full profile structural refinement of the atomic pair distribution function (PDF). 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, microdomains, 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 or on any plane 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. Neutron- as well as X-ray 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 4. 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 1.3.

1.2 What is new ?

If you have used *DISCUS* before, you might ask what is new in version 3.2 of the program. The major difference is the addition of two new modules: The first is the possibility to calculate powder diffraction patterns (see section 4.2). Secondly *DISCUS* can now calculate and refine the atomic pair distribution function (PDF). The refinement is done using the Reverse Monte Carlo algorithm already incorporated in *DISCUS*. For details refer to chapter 10 of this users guide. As usual some bugs were fixed and some functions were added, e.g. one can now save intensities in *SHELXL* format. A detailed list of all changes can be found in the file '*CHANGES.LOG*' in the source directory of the *DISCUS* distribution.

1.3 Getting started

After the program *DISCUS* is installed properly and the environment variables are set, the program can be started by typing 'discus' at the operating systems prompt. Information about the installation of *DISCUS* is given in appendix B.

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

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. If *DISCUS* has been compiled using the "`-DREADLINE`" option (see installation files) 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 shell. 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. A line can be marked as comment by inserting a '#' as first character in the line.

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.

1.4 More help

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>. The online help is also available as printed version in the file 'dis_cmd.ps' in the directory 'doc' of the distribution. The manual you are reading describes scientific background and principle functions of *DISCUS* and should give some insight in the ways to use this program. Additionally there is an **interactive tutorial** guiding through the different functions of the program. To start the tutorial, go to the directory 'tutorial' of the distribution, start *DISCUS* and enter '@tutorial' to begin the tour.

To find out about recent updates of *DISCUS* or to get further information visit the *DISCUS* WWW homepage at the following sites:

http://www.pa.msu.edu/∼proffen/discus/discus.html
http://www.uni-wuerzburg.de/mineralogie/crystal/discus/discus.html

1.5 Further reading

Unfortunately there are currently (at least to the authors knowledge) no books available giving a modern overview about disordered crystals and the analysis of diffuse scattering aided by computer simulation methods. However, there is a number of recent review articles about these topic, e.g. [7, 8, 9, 10, 30, 31]. A very good book on the backgrounds of diffraction physics was written by Cowley [5]. More specific literature about the scientific background is given in the individual chapters of this manual.

The program *DISCUS* can be used as an aid to teaching diffraction physics and diffuse scattering [15]. The corresponding WWW tutorial including *DISCUS* macro files can be found on the sites below and on further mirror sites around the world.

http://www.pa.msu.edu/∼proffen/teaching/teaching.html
http://www.uni-wuerzburg.de/mineralogie/crystal/teaching/teaching.html

Chapter 2

Creating structures

The first step towards any simulation is creating the desired structure and possibly modifying 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) and an isotropic thermal coefficient (B). 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 6. The second group includes at present the following modifications of the crystal at large, for more details see chapter 7: thermal displacement, waves, stacking faults and microdomains.

2.1 Reading structure files

A 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 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 2.3).

2.1.1 Structure file format

The new *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 by its name (e.g. ZR), the fractional coordinates (x,y,z) and an isotropic temperature factor B. A list of valid keywords within a structure file is given in table 2.1.

Keyword	**Description**
#	Allows comments in the keyword section of the structure file..
atoms	Marks the start of the atom list and must be the **last** keyword except for 'molecule'.
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	Parameters nx, ny, nz and nc define the size of the crystal in unit cells and the number of atoms within a unit cell. This information is needed if a complete structure is read in order to allow *DISCUS* to determine the crystals size.
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

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    0.00 0.00 0.00  0.5
O     0.25 0.25 0.25  1.0
```

The first line is the required 'title' line describing the structure. The next line specifies the space group $Fm\bar{3}m$. The symbols used should be the Hermann-Mauguin symbols used in International Tables for Crystallography Vol. A [33]. 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 unique c-axis will be assumed if the corresponding non standard Hermann-Mauguin symbol is used. *DISCUS* checks the given space group symbol for contradictions with 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' 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 example file which must

be the last keyword 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.

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 new 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'

Keyword	Description
molecule	Defines the start of a molecule.
molecule atoms	Lists atom numbers belonging to current molecule.
molecule content	Defines the start of particular molecule type.
molecule generator	Defines generators for the internal symmetry of the molecule.
molecule symmetry	Defines internal symmetry operations for the molecule.
molecule end	Defines end of a molecule atom list.

Table 2.2: List of keywords molecules

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.

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 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 [33] 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. The following example structure file contains a water (H_2O) molecule in a structure with the space group *Cmm*2.

```
title Water in Cmm2
spcgr Cmm2
cell  10.0,10.0,10.0, 90.0,90.0,90.0
atoms
molecule
molecule gene, -1,0,0,0, 0,1,0,0, 0,0,1,0
O   0.00  0.20   0.00   0.1
H   0.13  0.17   0.00   0.2
molecule end
```

The first four lines of this example file are similar to the previous example and define title, space group and lattice constants. In the 'atoms' section, however, one oxygen and one hydrogen atom define the water molecule between the 'molecule' and 'molecule end' keywords. The second hydrogen atom for the H_2O molecule is generated by a yz-mirror plane defined by the 'gene' sub-keyword. The mirror plane goes through the origin of the molecule which is defined as the first atom in the molecule list, here oxygen. The coordinates of the four created H_2O molecules per unit cell in the given space group *Cmm*2 are shown below.

```
Molecule Number:          1 Type:          1
 Name       Number      x            y            z             B
 O(1)            1       .000000      .200000      .000000      .100000
 H(2)            5       .130000      .170000      .000000      .200000
 H(2)           11      -.130000      .170000      .000000      .200000

 Molecule Number:          2 Type:          1
```

```
Name       Number      x            y            z             B
O(1)          2         .500000      .700000      .000000      .100000
H(2)          6         .630000      .670000      .000000      .200000
H(2)         12         .370000      .670000      .000000      .200000

Molecule Number:          3 Type:          1
Name       Number      x            y            z             B
O(1)          3         .000000      .800000      .000000      .100000
H(2)          7        -.130000      .830000      .000000      .200000
H(2)          9         .130000      .830000      .000000      .200000

Molecule Number:          4 Type:          1
Name       Number      x            y            z             B
O(1)          4         .500000      .300000      .000000      .100000
H(2)          8         .370000      .330000      .000000      .200000
H(2)         10         .630000      .330000      .000000      .200000
```

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 in the example 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 2.3 for more details about saving structures.

2.2 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' sublevel, 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 [33], Vol. A. The sequence of symmetrically equivalent atoms is therefore identical to that of the International Tables. In accordance with the International Tables, the generators required for centered space groups are applied first. The atoms generated by these generators immediately follow each position generated by the ordinary symmetry operators as 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
$x+\frac{1}{2},y+\frac{1}{2},z$
$\overline{x},y,\overline{z}$
$\overline{x}+\frac{1}{2},y+\frac{1}{2},\overline{z}$

The 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' sublevel of *DISCUS* allows to transform between the index of an atom and its unit cell and site number. 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 section 3.1) contains the number of atoms per unit cell. Some calculations of *DISCUS* (e.g. usage of lots in Fourier sublevel, 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.

2.3 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' sublevel 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.

N
tion 7

2.4

The s
spon
packa
migh

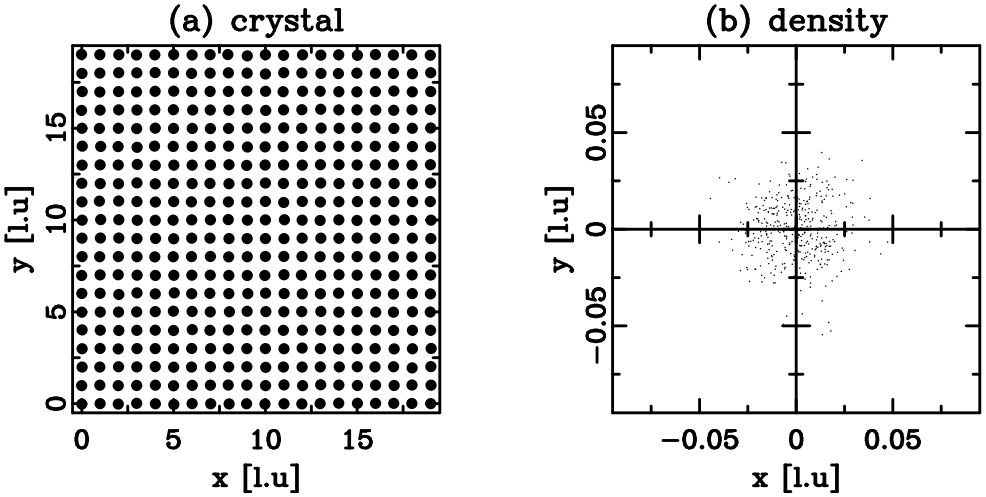

Figure 2.1: Example of structure export options

Different export option are available in the 'plot' sublevel of *DISCUS*. Individual atom or molecule types to be included in the output might be selected. Atoms might also be selected according to their microdomain status (see section 7.3 for details). 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 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. Figure 2.1 shows an example structure (20x20x1 unit cells) showing thermal displacements and demonstrates another export option of *DISCUS*, either the crystal might be saved (Fig. 2.1a) or the density within a unit cell (Fig 2.1b), i.e. all atoms are projected into a single unit cell.

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, colour and size for the individual atom or molecule representation. These properties can be altered using the 'set'

command within the 'plot' sublevel. 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. The *GNUPLOT* set of commands to plot the file '*atoms.xy*' would look like this:

```
set parametric      ! specifies data are x(t),y(t)
set view 60,45      ! sets nice view angles
splot 'atoms.xy'    ! plot data file
```

Use any desired values for the 'set view' command to view the structure along different directions.

Since the exported file is a simple text file containing the atoms coordinated in the defined sequence in each line, any plotting software capable of importing this type of file can be used to view the exported structure. However, only the use of the program *KUPLOT* allows the user to have individual colours, markers and sizes for the different atoms or molecule types within the crystal.

Chapter 3

FORTRAN style interpreter

The program includes a FORTRAN style interpreter that allows the user to program complex modifications. The interpreter provides variables, linked to the structure and free variables, loops, logical construction, basic arithmetic and built in functions. Commands related to the FORTRAN interpreter are '=', 'break', 'do', 'else', 'elseif', 'enddo', 'endif', 'eval', 'if'.

3.1 Variables

All variables are denoted by a name, followed by a left square bracket [, one or more indices and a right square bracket].

Example : i[1], r[0], i[i[1]], cdim[1,2]

This version of *DISCUS* deletes all blanks from user input lines. To keep a nicely looking macro file you can insert blanks between the variable name and the left square bracket. Blanks within the square brackets are not significant. The index can be any integer expression, especially any of the integer variables again. Two free variables are provided, an integer, i[n], and a real, r[n] variable. Each of these variables is actually an array of dimension n given in squared brackets. The allowed range for n is 0 to MAXPARAM, which is defined at compilation time by the parameter MAXPARAM in the file '*param.inc*', see the file INSTALL for further help.

Beside these variables i[n] and r[n] for general use a large variety of variables is linked to structural information of the current structure stored in *DISCUS*. Some of these variables cannot be modified, others like the position of individual atoms can be altered, thus allowing to modify the model crystal. The following lists give a summary of the different variables available.

The table 3.1 shows a summary of crystal related variables. The values of these variables can not be altered. Some molecule related variables have been added in this version of *DISCUS*. Table ⋆ 3.2 lists variables related to each atom. These variables can be changed (e.g. x[1] = x[1]+0.06) allowing the user to alter each atom within the crystal. Another variable is res[i] which contains the results of a particular *DISCUS* command. The variable res[0] contains the number of elements returned in res[i]. Check the online help for the different commands to see what results might be returned to this variable. Another large set of variables is related to the use of microdomains. For a list of these variables refer to chapter 7.3.

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
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)

Table 3.1: Crystal related variables

Variable	Description
m[i]	Number of scattering type (i.e. atom type) for atom 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

Table 3.2: Variables related to individual atoms

3.2 Arithmetic expressions

DISCUS allows the use of arithmetic expressions using the same notation as in FORTRAN. Valid operators are '+', '-', '*', '/' and '**'. Expressions can be grouped by round brackets (and). The usual hierarchy for the operators holds. Values of expressions can be assigned to any modifiable variable. If you know FORTRAN (or another programming language) you will have no problems with these examples.

```
i[0] = 1
r[3] = 3.1415
r[i[1]] = 2.0*(i[5]-5.0/6.5)
```

3.3 Logical expressions

Logical expressions are formed similar to FORTRAN. They may contain numerical comparisons using the syntax: *<arithmetic expression><operator><arithmetic expression>*. The allowed operators within *DISCUS* are *.lt., .le., .gt., .ge., .eq.* and *.ne.* for operations less than, less equal, greater than, greater equal, equal and not equal, respectively.

Logical expressions can be combined by the logical operators *.not., .and.* and *or*. The following example shows an expression that is true for values of i[1] within the interval of 3 and 11, false otherwise.

```
i[1].ge.3 .and. i[1].le.11
```

Logical operations may be nested and grouped using round brackets (and). For more examples see section 3.6.

3.4 Intrinsic functions

Several intrinsic functions are defined. Each function is referenced, as in FORTRAN, by its name followed by a pair of parentheses (and) that include the list of arguments. The (does not have to immediately follow the function name. Trigonometric and arithmetic functions are listed in table 3.3. Table 3.4 contains various random number generating functions and table 3.5 lists crystallographic functions.

Type	Name	Description
real	sin(r) cos(r) tan(r)	Sine, cosine and tangent of <r> in radian
real	sind(r) cosd(r) tand(r)	Sine, cosine and tangent of <r> in degrees
real	asin(r) acos(r) atan(r)	Arc sin, cosine, tangent of <r>, result in radian
real	asind(r) acosd(r) atand(r)	Arc sin, cosine, tangent of <r>, result in degrees
real	sqrt(r)	Square root of <r>
real	exp(r)	Exponential of <r>, base e
real	ln(r)	Logarithm of <r>
real	sinh(r) cosh(r) tanh(r)	Hyperbolic sine, cosine and tangent of <r>
real	abs(r)	Absolute value of <r>
integer	mod(r1, r2)	Modulo <r1> of <r2>
integer	int(r)	Convert <r> to integer
integer	nint(r)	Convert <r> to nearest integer
real	frac(r)	Returns fractional part of <r>

Table 3.3: Trigonometric and arithmetic functions

Type	Name	Description
real	ran(r)	Uniformly distributed pseudo random number between 0.0 inclusively and 1.0 exclusively. Argument <r> is a dummy
real	gran(r1, typ)	Gaussian distributed random number with mean 0 and a width given by <r1>. If <typ> is "s" <r1> is taken as sigma, if <typ> is "f" <r1> is taken as FWHM.
real	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.

Table 3.4: Random number functions

Additionally the function *md_test* allows the user to test weather a given real space position is inside any defined microdomain. More details about using microdomains and related variables and functions are given in chapter 7.3.

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	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.

Table 3.5: Crystallographic functions

3.5 Loops

Loops can be programmed in *DISCUS* using the 'do' command. Three different types of loops are implemented. The first type executes a predefined number of times. The syntax for this type of loop is

do <variable> = <start>,<end> [,<increment>]
. . . commands to be executed . . .
enddo

Loops may contain constants or arithmetic expressions for <start>, <end>, and <increment>. <increment> defaults to 1. The internal type of the variables is real. The loop counter is evaluated from *(<end> - <start>)/<increment> + 1*. If this is negative, the loop is not executed at all. The parameters for the counter variable, start end and increment variables are evaluated only at the beginning of the do - loop and stored in internal variables. It is possible to change the values of <variable>, <start> and/or <end> within the loop without any effect on the performance of the loop. This practice is not encouraged, could, however, be an unexpected source of errors.

The second type of loop is executed while <logical expression> is true. Thus it might not be executed at all. The syntax for this type of loop is

do while <logical expression>
. . . commands to be executed . . .
enddo

The last type of loop is executed until <logical expression> is true. This loop, however, is always executed once and has the following syntax

do
. . . commands to be executed . . .
enddo until <logical expression>

In the body of commands any valid *DISCUS* command can be used. This includes calls to the sublevels, further do loops or macros, even if these macros contain do loops themselves. The maximum level of nesting is limited by the parameter *MAXLEV* in the file *'doloop.inc*'. If necessary adjust this parameter to allow for deeper nesting. All commands from the first 'do' command to the corresponding 'enddo' are read and stored in an internal array. This array can take at most *MAXCOM* (defined in file '*doloop.inc*' as well) commands at every level of nesting. If lengthy macro files are included in the do loop, this parameter might have to be adjusted.

If a do loop (or an if block) needs to be terminated, the 'break' command will perform this function. The parameter on the 'break' command line gives the number of nested levels of 'do' and 'if' blocks to be terminated. The interpreter will continue execution with the first command following the corresponding 'enddo' or 'endif' command. An example is given below, note, that the line numbers are only given for better orientation and are no actual part of the listed commands.

```
1  do i[2]=1,5
2    do i[1]=1,5
3      if ((i[1]+i[2]) .eq 6) then
4        break 2
5      endif
6    enddo
7  enddo
```

In this example, the execution of the inner do-loop will stop as soon as the sum of the two increment variables i[1] and i[2] is equal to 6. The program continues with the 'enddo' line of the outer do - loop. Notice that two levels need to be interrupted, the if block and the innermost do loop. If the parameter had bin equal to one, only the if block would have been interrupted, while the innermost do loop would have continued without break.

3.6 Conditional statements

Commands can be executed conditionally by using the 'if' command. Analogous to FORTRAN, the if-control structure takes the following form:

if (<logical expression>) then
... commands to be executed ...
elseif (<logical expression>) then
... commands to be executed ...
else
... commands to be executed ...
endif

The logical expressions are explained in section 3.3. Enclosed within an if block any valid *DISCUS* command can be used. This includes calls to the sublevels further if blocks, do loops or macros, even if these macros contain if blocks or do loops themselves. The 'elseif' and 'else' section is optional. The maximum level of nesting is limited by the parameter *MAXLEV* in the file '*doloop.inc*'. If necessary adjust this parameter to allow for deeper nesting. All commands from the first 'if' command to the corresponding 'endif' are read and stored in an internal array. This array can take at most *'MAXCOM'* (defined in file '*doloop.inc*' as well) commands at every

level of nesting. If lengthy macro files are included in the do loop, this parameter might have to be adjusted.

If an if block (or a do loop) needs to be terminated, the 'break' command will perform this function. The parameter on the 'break' command line gives the number of nested levels of 'do' and 'if' blocks to be terminated. The interpreter will continue execution with the first command following the corresponding 'enddo' or 'endif' command. See the example in section 3.5 for further explanations.

```
 1  #
 2  # Read crystal file
 3  #
 4  read
 5  cell cell.cll,10,10,10
 6  #
 7  # Remove atoms with probability 0.3
 8  #
 9  do i[0]=1,n[1]
10    if(ran(0).lt.0.3) then
11      remove i[0]
12    endif
13  enddo
```

The example listed above illustrates the use of loops and conditional statements within *DISCUS*. Again, the line number are given for easy reference and not part of the actual *DISCUS* input. The first three lines are just comments. In lines 4 and 5 an asymmetric unit is read from the file '*cell.cll*' and expanded to a crystal size of 10x10x10 unit cells. In line 9 starts a do-loop over all atoms within the crystal. The variable n[1] contains this information (see table 3.5 in section 3.1). Since the function 'ran' produces a uniformly distributed pseudo random number in the range 0.0 to 1.0, the if statement in line 10 is true in about 30% of its calls, at least for large enough crystal sizes. Thus approximately 30% of the atoms are removed (line 11), and the corresponding amount of vacancies (VOID) created within the crystal.

3.7 Filenames

Usually, file names are understood as typed, including capital letters. Unix operating systems distinguish between upper and lower case typing ! However, sometimes it is required to be able to alter a file name e.g. within a loop. Thus, *DISCUS* allows the user to construct file names by writing additional (integer) numerical input into the filename. The syntax for this is:

"string%dstring",<integer expression>

The file format MUST be enclosed in quotation marks. The position of each integer must be characterized by a '%d'. The sequence of strings and '%d's can be mixed at will. The corresponding integer expressions must follow after the closing quotation mark. If the command line requires further parameters (like 'addfile' for example) they must be given after the format-parameters. The interpretation of the '%d's follows the C syntax. Up to 10 numbers can be written into a filename. All of the following examples will result in the file name '*a1.1*':

```
i[5]=1
outfile a1.1
outfile "a%d.%d",1,1
outfile "a%d.%d",4-3,i[5]
```

The second example shows how filenames are changed within a loop. Here the output (e.g. Fourier transform) will be written to the files '*data1.calc*' to '*data11.calc*'.

```
do i[1]=1,11
  ..
  outfile "data%d.calc",i[1]
  ..
enddo
```

3.8 Macros

Any list of valid *DISCUS* commands can be written to an ASCII file and executed indirectly by the command '@<filename>'. The commands must start in the left most column of the file and are otherwise executed as typed. Macro files can be written by any editor on your system or be generated by the 'learn' command. 'learn' starts to remember all the commands that follow and saves them into the file given on the 'learn' command. The learn sequence is terminated by the 'lend' command. The default extension of the macro file is '*.mac*'. Macro files can be nested and even reference themselves directly or indirectly. This referencing of macro files is, however, just a nesting of the corresponding text of each macro, not a call to a function. All variable retain their values. If an error occurs while executing a macro, *DISCUS* immediately stops execution of all macros and returns to the interactive prompt. If the macro switched to a sublevel, and the error occurred inside of this sublevel, *DISCUS* will remain within this sublevel the interactive prompt corresponding to this sublevel is returned. The command 'stop' allows the user to interrupt the execution of a macro, enter commands and continue the macro using the command 'cont'. Note, that the macro needs to continued in the same sublevel it was interrupted.

On the command line of the macro command '@', optional parameters can be supplied. Within the macro these have to be referenced as '$1', '$2' etc. Upon execution of the macro the formal parameters '$n' are replaced by the character string of the actual values from the command line. As any other command parameters, these parameters must be separated by comma. If a formal parameter is referenced inside a macro without a corresponding parameter on the command line, an error message is given. An example is given below:

```
# Adds two numbers supplied as command line parameters.
# The value is stored in variable defined by parameter three
#
$3 = $1 + $2
```

If this macro is called with the following line, @add 1,2,i[4], the result is stored in variable i[4] which now has the integer value 3.

Chapter 4

Fourier transform

4.1 Calculating scattering intensities

Basically *DISCUS* calculates the Fourier transform (neutron or X-ray) according to the standard formula for kinematic scattering given in equation 4.1.

$$F(\mathbf{h}) = \sum_{i=1}^{N} f_i(\mathbf{h}) e^{2\pi i \mathbf{h}\mathbf{r}_i} \cdot e^{-\frac{B|\mathbf{h}|^2}{4}} \tag{4.1}$$

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 **h** in reciprocal space. The form factors are tabulated and calculated once at each **h** for all species present in the crystal. Optionally the Debye-Waller factor is calculated. Isotropic B are used for the representation of thermal disorder. It should be kept in mind, that this is strictly true only for elastic neutron scattering. Alternatively the atoms can be displaced according to their B values using the command 'therm' and B is ignored for the subsequent calculation of the Fourier transform. *DISCUS* can calculate neutron- and X-ray intensities including anomalous X-ray scattering. The calculation of the explicit Fourier transform has several advantages over the use of a Fast-Fourier-Transform (FFT) in terms of required memory for a suitable resolution in atom positions and representation of the scattering density for X-rays. A more detailed discussion can be found in [18]. The realization of the actual code to calculate the Fourier transform is based on the program *DIFFUSE* [3]. 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 4.1 in a straight forward way. More details about the algorithm used can be found [3].

4.1.1 Finite size effects

The simulated crystals will generally be small compared to the crystal size used in a scattering experiment (although computers get more powerful every day). Finite size effects result from the convolution of the scattering density distribution within one unit cell with a finite train of delta distributions. Since the scope of the program is to calculate the scattering by real crystals, it is helpful to write the scattering density of a real crystal as:

$$\begin{aligned}\rho(\mathbf{r}) &= \rho_\infty(\mathbf{r}) \cdot box(\mathbf{r}) \\ &= \left[\Delta\rho(\mathbf{r}) + \rho_0(\mathbf{r}) \star \sum_{i=-\infty}^{\infty} \delta(\mathbf{r}-\mathbf{R}_i)\right] \cdot box(\mathbf{r}) \end{aligned} \tag{4.2}$$

Here $\rho(\mathbf{r})$ is the scattering density of the real crystal, $\rho_\infty(\mathbf{r})$ is the scattering density of the infinite crystal and $box(\mathbf{r})$ is the function limiting the crystal. If an average structure of the crystal can sensibly be defined, $\rho_\infty(\mathbf{r})$ can be separated into the average scattering density $\rho_0(\mathbf{r})$ convoluted with the real space lattice and the deviation from that average scattering density $\Delta\rho(\mathbf{r})$.

The Fourier transform of the scattering density given in equation 4.2 is then:

$$F(\mathbf{h}) = \Delta F(\mathbf{h}) \star BOX(\mathbf{h}) + (F_0(\mathbf{h}) \cdot G^*) \star BOX(\mathbf{h}) \tag{4.3}$$

Here F_0 is the Fourier transform of the average unit cell and G^* is the reciprocal lattice. $BOX(\mathbf{h})$ is the Fourier transform of the crystal limiting function $box(\mathbf{r})$. Even for a size of a few unit cells, the convolution of the defect scattering $\Delta F(\mathbf{h})$ with the $BOX(\mathbf{h})$ can be neglected in most cases. Since the main scope of *DISCUS* is the calculation of diffuse scattering, a reasonable way to separate the subsidiary maxima of the Bragg reflections from the defect scattering is thus to subtract the scattering of the average structure from the scattering of the whole crystal. The average structure factor $\langle F \rangle = (F_0(\mathbf{h}) \cdot G^*) \star BOX(\mathbf{h})$ can be calculated from a user defined part of the crystal and subtracted from the calculated Fourier transform. For further information refer to the command reference and the example given in section 4.1.3. However, this procedure will fail for defects that are well ordered and thus produce sharp satellites and the convolution of the defect scattering with $BOX(\mathbf{h})$ can no longer be neglected. The resulting satellites will show subsidiary maxima. The Fourier transform for a perfect crystal is zero at the positions of the satellites, and therefore no intensity will be subtracted at the position of the subsidiary maxima near satellites.

An alternative possibility to avoid finite size effect contributions are so called periodic boundaries. Let us assume the crystal is limited by a box of the length a in one dimension. Thus the Fourier transform $BOX(\mathbf{h})$ of the box function is given by:

$$BOX(h) = \frac{\sin(\pi a h)}{\pi h} \tag{4.4}$$

Here the function $BOX(h)$ is zero for all points where $a \cdot h$ is integer. This condition is fulfilled for all points which are on a grid given by

$$h = \frac{h'}{a} \quad with\, h' \in integer\ numbers \tag{4.5}$$

If the scattering is calculated only at points in reciprocal space, for which $\Delta h = 1/a$ with a the dimension of the model crystal, the contribution of the average structure vanishes and the calculated intensity is free of finite size effect contributions. Using this condition a suitable grid size in reciprocal space showing no finite size effect contributions would be $\Delta h = 0.05$ reciprocal lattice units for a crystal size of 20 unit cells. This concept is sometimes called *super cell* thinking of the complete crystal as a super unit cell and calculating the Fourier transform only for Bragg positions which are free of finite size effect contributions. This straightforward calculation does

apply only to crystals limited by a block of unit cells. If the crystal is limited by, for example, a sphere, periodic boundaries cannot be defined. Since the property of interest is the diffuse scattering, rather than the shape function of a model, the crystal can always be arranged to consist of a block of unit cells.

4.1.2 Coherence

In a diffraction experiment only atoms within the lateral and transverse coherence of the incident radiation will scatter coherently. Averaging over space and time gives incoherent scattering by adding the intensities. However, when calculating the Fourier transform given by the equation 4.1 **all** atoms are treated as if they scatter coherently. Even though, there might be no structural coherence between different parts of the crystal, the calculation nevertheless adds the amplitudes of all waves. For large crystals this may lead to unexpected high frequency oscillations severely modulating the diffuse scattering. Small crystal sizes, however, might give only a statistically poor description of the particular disorder model.

A tentative explanation for these oscillations will now be given. The contribution to the scattering amplitude $F_{ij}(\mathbf{h})$ for two atoms at $\mathbf{r}_i$ and $\mathbf{r}_i + \mathbf{R}$ can be written as:

$$\begin{aligned} F_{ij}(\mathbf{h}) &= f_i(\mathbf{h}) e^{2\pi i \mathbf{h} \mathbf{r}_i} + f_j(\mathbf{h}) e^{2\pi i \mathbf{h} \mathbf{r}_i + \mathbf{R}} \\ &= f_i(\mathbf{h}) e^{2\pi i \mathbf{h} \mathbf{r}_i} \left(1 + \frac{f_i(\mathbf{h})}{f_j(\mathbf{h})} \cdot e^{2\pi i \mathbf{h} \mathbf{R}} \right) \end{aligned} \tag{4.6}$$

The term $e^{2\pi i \mathbf{h} \mathbf{R}} = \cos(2\pi \mathbf{h} \mathbf{R}) + i \sin(2\pi \mathbf{h} \mathbf{R})$ represents the discussed oscillation as a function of the vector $\mathbf{R}$. For $\mathbf{R} \ll$ crystal dimension, a large number of different atoms will be separated by $\mathbf{R}$ or vectors of very similar length. Their contribution will give a good approximation of the expected value of an infinite crystal. If the length of $\mathbf{R}$ becomes comparable to the dimension of the crystal dimension, only few atoms will be separated by $\mathbf{R}$ and their contribution to the scattering might be far from the expected average from a infinite crystal. Obviously these arguments are independent from the actual size of the model crystal, but for large crystals the above contribution represents a high frequency wave more severely modulating the observed scattering.

In order to be able to calculate smooth diffraction patterns, *DISCUS* offers an option ('lots') to calculate the scattering intensities as average of intensities calculated from small randomly chosen volumes (lots) within the crystal. It should be noted, that using lots will give only **intensities** and it is not possible to save structure amplitudes or phases as would otherwise be possible if the 'lot' option is switched off. It is important to make sure that the selected number of lots is sufficiently large to cover the complete model crystal. Furthermore the size of the lots must be large enough to include all significant neighbour interactions for the given defect structure.

4.1.3 Example

In this section the different options available for calculating diffraction patterns will be demonstrated using a disordered 2D structure of 50x50 unit cells in size. Each unit cell contains a Zr on (0,0,0) with an occupancy of 0.83. The vacancies show short range order and size effect distortions. The following macro file was used to generate the different diffraction patterns shown in figure 4.1.

```
 1  #
 2  # Read structure first
 3  #
 4  read
 5  stru demo.stru
```

At the beginning of this macro in lines 4 and 5 of this macro the disordered structure stored in file '*demo.stru*' is read.

```
 6  #
 7  # Calculate 'plain' Fourier
 8  #
 9  four
10  xray
11  wvle moa1
12  ll 0.0,0.0,0.0
13  lr 2.0,0.0,0.0
14  ul 0.0,2.0,0.0
15  na 81
16  no 81
17  set aver,0.0
18  lots off
19  run
20  exit
21  #
22  @output plain.scat
```

This second part of the macro file calculates the Fourier transform straight forward according to equation 4.1. After entering the Fourier segment of *DISCUS* (line 9), $MoK\alpha$ radiation is selected in lines 10 and 11. Next the lower left, lower right and upper left corner of the desired plane in reciprocal space are specified (lines 12-14). The number of grid points in both directions are set in lines 15 and 16. Note that the selected grid size $\Delta h = 0.025$ *r.l.u.* does not match the periodic boundary condition described in section 4.1.1. Finally the subtraction of $\langle F \rangle$ is disabled (line 17) and the lot option is switched off (line 18). After the Fourier transform is calculated (line 19), the scattering intensities are written to the file '*plain.scat*' using the macro file '*output.mac*'.The resulting scattering pattern is shown in figure 4.1 in the lower left corner. The finite size effect contributions can clearly be seen as 'dotted lines' intersecting the Bragg positions.

```
23  #
24  # Now we calculate on a grid
25  # corresponding to 1/50
26  #
27  four
28  na 101
29  no 101
30  run
31  exit
32  #
33  @output period.scat
```

The second diffraction pattern (Figure 4.1, lower right corner) is calculated on a grid corresponding to $\Delta h = 1$/(dimension of the model crystal)$= 0.02$ *r.l.u.*. Thus the number of grid points

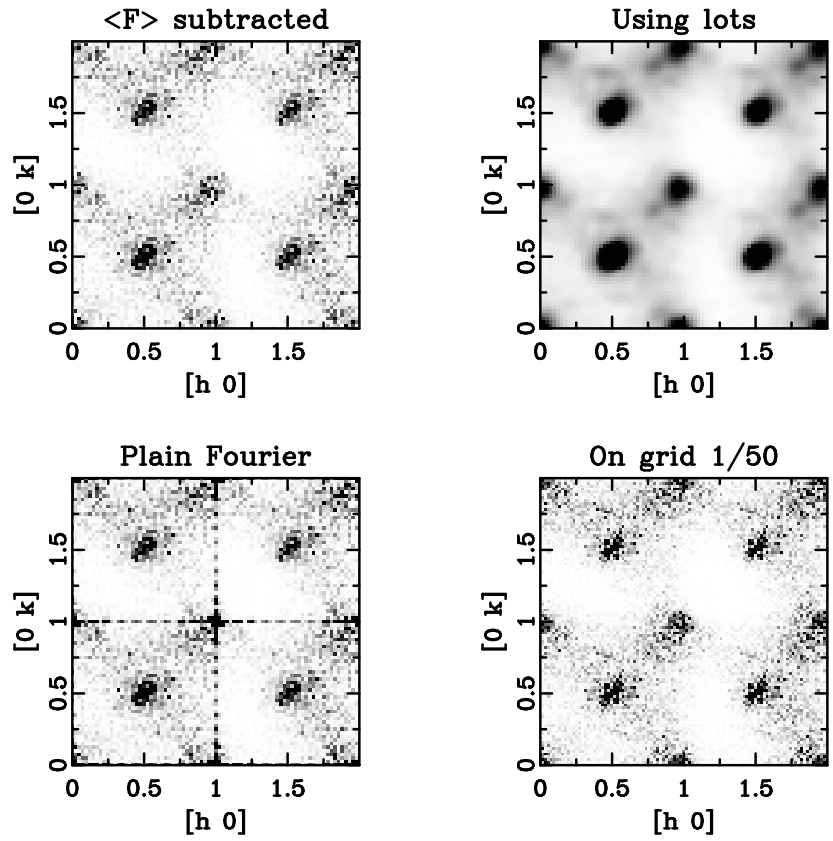

Figure 4.1: Diffuse scattering calculated using the different *DISCUS* modes

is increased accordingly (lines 28-29). Note that all previous settings are still valid. The Fourier transform is recalculated (line 30) and the output written to the file '*period.scat*'. As expected the finite size effect contributions have disappeared (Figure 4.1).

```
34  #
35  # Now recalculate with subtracted
36  # <F> calculated from 50% of the
37  # model crystal
38  #
39  four
40  na 81
41  no 81
42  set aver,50.
43  run
44  exit
45  #
46  @output aver.scat
```

As an alternative to the use of periodic boundaries as before, the average structure factor $\langle F \rangle$ can be subtracted. First the number of grid points are returned to their previous values (lines 40-41) and $\langle F \rangle$ will be calculated from 50 % of the crystal volume (line 42). Especially for large crystals

this value might be reduced in order to save computing time. The Fourier transform is recalculated (line 43) and the output written to the file '*aver.scat*'. The corresponding diffraction pattern is shown in figure 4.1 top left corner. Again the finite size effect contributions have disappeared although the original grid size was used and the pattern looks remarkably similar to the previous one calculated on a grid matching the periodic boundary conditions. Actually the Bragg peaks have disappeared as well which is hard to see in this example, because diffuse scattering contributions exist at the positions of the Bragg peaks.

```
47  #
48  # Now use 100 lots, 10x10x1 unit cells
49  # in size
50  #
51  four
52  lots eli,10,10,1,100,y
53  run
54  exit
55  #
56  @output lots.scat
```

Finally the diffuse intensity is calculated using the lot option, i.e. the intensity from small crystal volumes chosen at random is averaged. Here those volumes are set to be ellipsoids with a size of 10x10x1 unit cells. A total of 100 lots is averaged (line 52). The Fourier transform is recalculated (line 53) and the output written to the file '*lots.scat*'. Inspection of the resulting diffraction pattern (Figure 4.1 top right corner) reveals a much smoother picture of the significant diffuse scattering features.

More details about the commands in the Fourier transform segment of *DISCUS* can be found in the reference manual and the online help. Additionally there is a Fourier transform demonstration macro as part of the interactive tutorial (see 1.4).

4.2 Powder diffraction

In addition to the calculation of single crystal scattering intensities, *DISCUS* can also computer a powder diffraction pattern of a given structure. The current algorithm is not optimized for speed and works as follows: The intensity along a line parallel to h in reciprocal space is calculated using the 'plain' Fourier transform. The range in h that is actually computed depends on the user defined range of the scattering angle $2\Theta_{min} < 2\Theta < 2\Theta_{max}$. The calculated intensity is than mapped onto the 2Θ array. Next k is incremented and once the complete plane in reciprocal space is covered, the coordinate l is incremented until all reciprocal space in between the two spheres defined by $2\Theta_{min}$ and $2\Theta_{max}$ is covered.

Let us consider a simple example, which is also part of the interactive *DISCUS* tutorial. The macro file is listed below, the line numbers are only for easy reference as before.

```
1 read
2 cell lmo.cll,5,5,5
3 #
4 powder
5 set dh,0.2
6 set dk,0.2
```

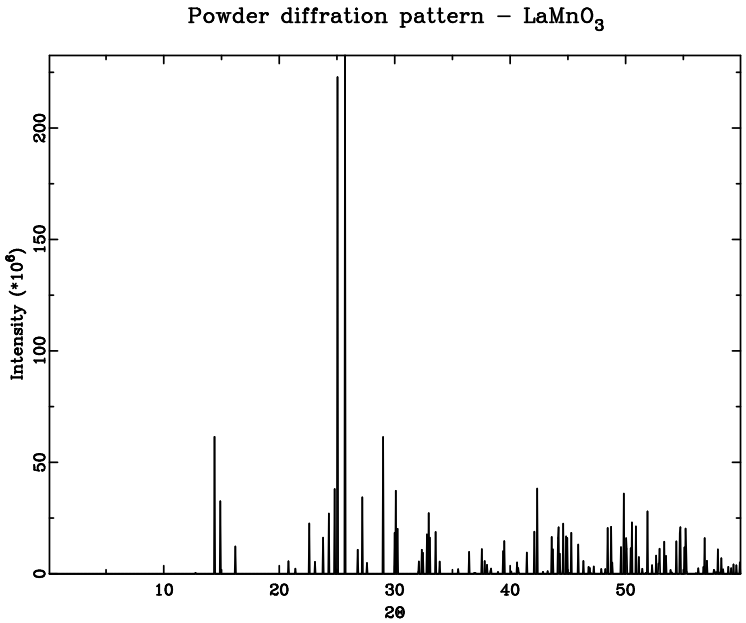

Figure 4.2: Simulated powder diffraction pattern of $LaMnO_3$.

```
 7 set dl,0.2
 8 set dtth,0.05
 9 set tthmin, 0.1
10 set tthmax,60.0
11 #
12 neut
13 set wvle,1.0
14 set temp,ignore
15 #
16 run
17 exit
18 #
19 output
20 format powder
21 outf powder.dat
22 run
23 exit
```

Lines 1–2 read the unit cell file and expand it to a crystal of 5x5x5 unit cells in size. Next we enter the powder diffraction sublevel (line 4). The grid size in h,k and l is set to 0.2 r.l.u in lines 5–7. As we have already discussed in section 4.1.1 we need to choose a grid in reciprocal space of 1/N to avoid finite size effect contributions to our scattering. Next we specify the grid in 2Θ (line 8) and the minimum (line 9) and maximum (line 10) value of 2Θ. Those values are given in degrees. Lines 12–13 select neutron scattering at a wavelength of λ=1Å and disable thermal factors, respectively. Now we are ready to start the calculation (line 16). Finally we need to save the result, which is done in lines 19–23. The resulting diffraction pattern is shown in figure 4.2.

4.3 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:

$$\rho(\mathbf{r}) = \frac{1}{V}\sum_{i=1}^{N} F_i(\mathbf{h}) e^{-2\pi i \mathbf{h}\mathbf{r}_i} \tag{4.7}$$

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 4.1.

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

Table 4.1: List of available Fourier methods

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.

4.4 Output file formats

All results of the Fourier transforms are written at the 'output' segment of *DISCUS*. The following values can be saved :

intensity, amplitude, real part, imaginary part and phase angle (in degrees)

The 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 4.2.

The first two 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

Output format	Description
'stan'	Output is saved as real numbers in a format suitable for *KUPLOT* which is part of the *DISCUS* program package.
'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 4.2: Output formats for scattering intensities

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 sublevel. If for example the $(hkl) * [1\overline{1}0] = 0.0$ 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 *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.

Chapter 5

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 [25].

5.1 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 3.5 in section 3.4 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 5.1: Commands for crystallographic calculations

The crystallographic calculation commands provided by *DISCUS* are shown in table 5.1. All commands display the results on the screen and store them in the variable res[i] (see section 3.1) 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**)).

5.2 Generalized symmetry operations

Lets assume that you want to rotate the oxygen atoms of a SiO_4 tetraeder 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. In general a symmetry operation will be characterized by:

- Orientation of the symmetry axis
- Rotation angle
- Proper or improper rotation (i.e. pure rotation or rotation plus inversion)
- Translational components

A general symmetry operation can be described by a 4x4 matrix of the following form:

$$S = \begin{pmatrix} w_{11} & w_{12} & w_{13} & t_1 \\ w_{21} & w_{22} & w_{23} & t_2 \\ w_{31} & w_{32} & w_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.1}$$

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 [25].

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°.

The use of generalized symmetry operations will be illustrated by the following example. Figure 5.1a shows a unit cell containing four H_2O molecules. The molecule in the lower left corner (molecule 1) shall be rotated by 30° around a symmetry axis parallel to the z-direction (normal to drawing plane) through the origin of the molecule, here the oxygen atom. Furthermore the molecule in its new orientation shall be a new molecule type and replace the original molecule. The result is seen in figure 5.1b. Note that the rotated molecule is plotted in a different colour from the other molecules indicating the new type. The *DISCUS* macro file used to perform this symmetry operation is listed below. Again the line numbers are only included for better reference and not part of the actual macro file. Furthermore the parts reading and generating the starting crystal are omitted.

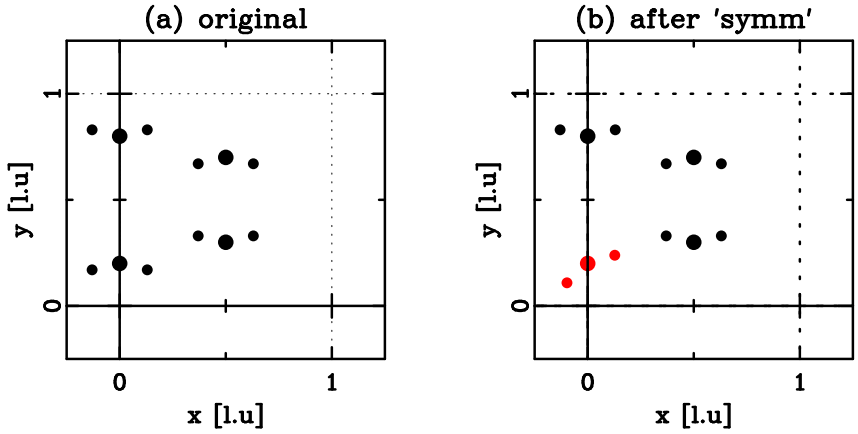

Figure 5.1: Example of a symmetry operation of H_2O molecule

```
 1  symm
 2  #
 3  uvw 0,0,1
 4  angl 30.0
 5  orig 0,0,0,mol
 6  trans 0,0,0
 7  type proper
 8  power 1,single
 9  mode repl,new
10  #
11  msel 1
12  minc 1,1
13  #
14  run
15  exit
```

After entering the 'symm' level of *DISCUS* (line 1), the direction of the symmetry axis is set parallel to the z-axis (line 3). Next the rotation angle (line 4), the origin for the symmetry operation (lines 5) and the translation part (line 6) are set. The additional parameter 'mol' in line 5 specifies the origin relative to the origin of the molecule rather than that of the crystal. Since we only want to operate the 30° rotation once, the power is set to one (line 8). Replacing the original molecule and creating a new type is selected in line 9. Finally we select molecule type 1 (line 11) and include only molecule number 1 (line 12) before the symmetry operation is executed (line 14).

The current settings in the 'symm' level as in any other segment of *DISCUS* are displayed via the 'show' command. For this example, the screen output would look like this:

```
Generalised Symmetry Operation
   Axis in direct space      :       .0000      .0000     1.0000
   Axis in reciprocal space  :       .0000      .0000   100.0000
   Origin of symmetry element:       .0000      .0000      .0000 rel.to molecule
   Rotation angle            :     30.0000
   Translation part          :       .0000      .0000      .0000

   Real space matrix         :       .8660     -.5000      .0000      .0000
```

```
                            :       .5000      .8660      .0000      .0000
                            :       .0000      .0000     1.0000      .0000

Reciprocal space matrix     :       .8660     -.5000      .0000
                            :       .5000      .8660      .0000
                            :       .0000      .0000     1.0000

Power of symmetry element :      1
Mode of power level       :   Single copy of original
Type of symmetry element  :   Proper rotation
Mode of symmetry operation:   Move atom/molecule to new position
Molecule status           :   Create new molecule type

Select status             :   Microdomains and host structure
Given origin relative to  :   Molecule
Selected molecule types   :      1
Range of sel. molecules   :         1 to         1
```

Besides the input such as symmetry axis or rotation angle, the transformation matrix as defined in equation 5.1. The newly created molecule type could now be distributed within the crystal with a given probability using the 'replace' command (see section 6.2).

5.3 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 6

Simple crystal modifications

How to create a model crystal from the information of its asymmetric unit was discussed in chapter 2. This section will give an overview of simple modifications of single atoms within a model crystal. Section 7 will describe *DISCUS* tools to modify a complete crystal.

6.1 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>] describing its position and the variable m[<i>] containing the atom type (see also section 3.1). In principle every desired defect structure might be realized using these variables and the FORTRAN style loops and conditional statements (see section 3). 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 2.2 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 neighbours surrounding the vacancy should be relaxed towards the vacant site. The perfect starting structure and the resulting disordered structure are shown in figure 6.1.

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 archive 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 neighbours 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
```

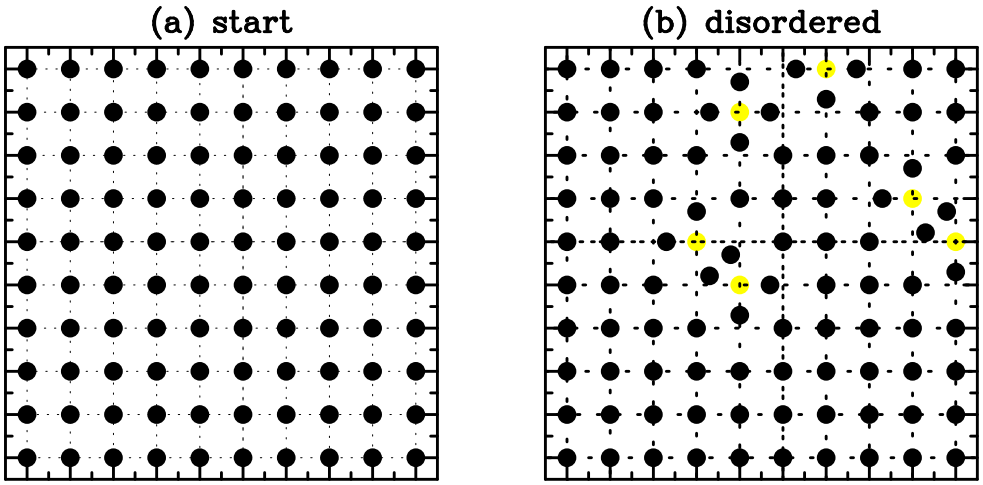

Figure 6.1: Structures created by crystal modification example

```
 7  #
```

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 6.1a).

```
 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 form the command 'find' (line 24), otherwise our simple way of relaxing the neighbours 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 section 3.1). 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 neighbours are stored in the variables env[<i>] and env[0] contains the number of found neighbours. Finally the positions of all neighbouring 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 6.1b. The introduced vacancies are shown as yellow circles (or light grey). Since this simple macro is not checking whether neighbouring atoms are already displaced, the clustered defects in the upper right quadrant of figure 6.1b have a different local environment compared to the isolated defects.

6.2 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 6.1 summarizes the available commands.

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 6.1: *DISCUS* commands for single atoms or molecules

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 5.2.

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** 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.

Chapter 7

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.

7.1 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:

$$\langle u^2 \rangle = \frac{B}{8\pi^2} \tag{7.1}$$

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.

7.2 Waves

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 as a new ⋆ feature 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 7.2, 7.3 and 7.4, 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.

$$w(\mathbf{r}) = A\cos(2\pi\left[\frac{\mathbf{kr}}{\lambda}+\psi\right])+A_0 \tag{7.2}$$

$$w(\mathbf{r}) = \begin{cases} A+A_0 & \text{for } \frac{\delta}{2} \le \left|\frac{\mathbf{kr}}{\lambda}+\frac{\psi}{360^\circ}\right| < 1-\frac{\delta}{2} \\ A_0 & \text{otherwise} \end{cases} \tag{7.3}$$

$$w(\mathbf{r}) = A\left[\frac{\mathbf{kr}}{\lambda}+\frac{\psi}{360^\circ}\right]+A_0 \tag{7.4}$$

The following list describes the different properties common to all wave types that can be defined by the user. Example macro files to further illustrate the 'wave' segment of *DISCUS* are given in the different sections below.

- *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 7.2.1). 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'. Alternatively a random phase ψ can be used which in case of using microdomains (see section 7.3) leads to a different random phase for the modulation within each selected microdomain.

The argument $[\mathbf{kr}/\lambda + \psi/360^\circ]$ 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 **r** is either the crystals origin or the origin of the current microdomain. *DISCUS* allows the user to select which of the atoms in the crystal are to be modified by the wave function. Besides choosing different types of atoms, one can choose how to treat atoms that are inside any microdomain (see section 7.3). The wave will displace **all** selected atoms of the whole crystal or the selected microdomains coherently. At present, no phase shift which in itself is a function of space is included. In order to simulate the effect of waves of finite coherence length, small crystals or microdomains should be simulated. By default, the initial phase shift is equal to zero and identical for all microdomains. The phase shift can also be defined to assume a random value within each of the selected microdomains. This can be used to create finite wave trains of incoherent phase.

7.2.1 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. An example for such a wave is displayed in figure 7.1. The direction of the sinusoidal wave is the x-direction. The wavelength λ is 40Å and the lattice constant of this cubic sample structure is 5Å.

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:

$$\begin{aligned} A &= \frac{1}{2}(P_{high} - P_{low}) \\ A_0 &= \frac{1}{2}(P_{high} + P_{low}) \end{aligned} \tag{7.5}$$

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

$$\begin{aligned} A &= P_{high} - P_{low} \\ A_0 &= P_{low} \end{aligned} \tag{7.6}$$

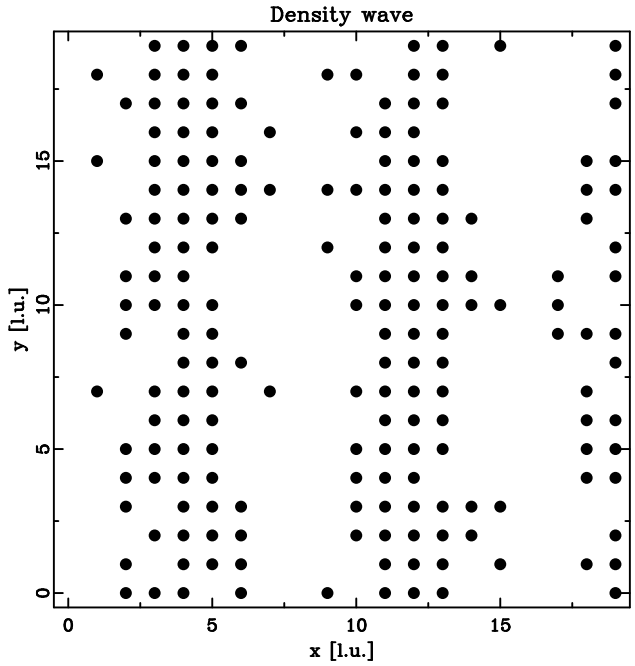

Figure 7.1: Example of a density wave

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.

7.2.2 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. An example can be seen in figure 7.2a.

Both directions, the wave vector and the oscillation vector, are parallel to x in our example. The wavelength λ is 35Å corresponding to the length of 7 unit cells for our cubic example structure. Although setting the oscillation vector to the same direction as the waves propagation vector will create longitudinal waves, *DISCUS* additionally provides the command 'long' which does the same, but ignores the 'osci' command. In cases where propagation and oscillation are normal to each other we speak of a transversal wave. Figure 7.2 shows such a transversal wave for the different wave functions, i.e. sinusoidal (b), square (c) and saw tooth (d). Note that although we have selected a symmetrical box shaped wave function (figure 7.2c) the result shows three displaced atoms and four atoms at their average position. This is due to the fact that the selected wavelength of 7 unit cells would require 3.5 atoms to be displaced which is obviously not possible.

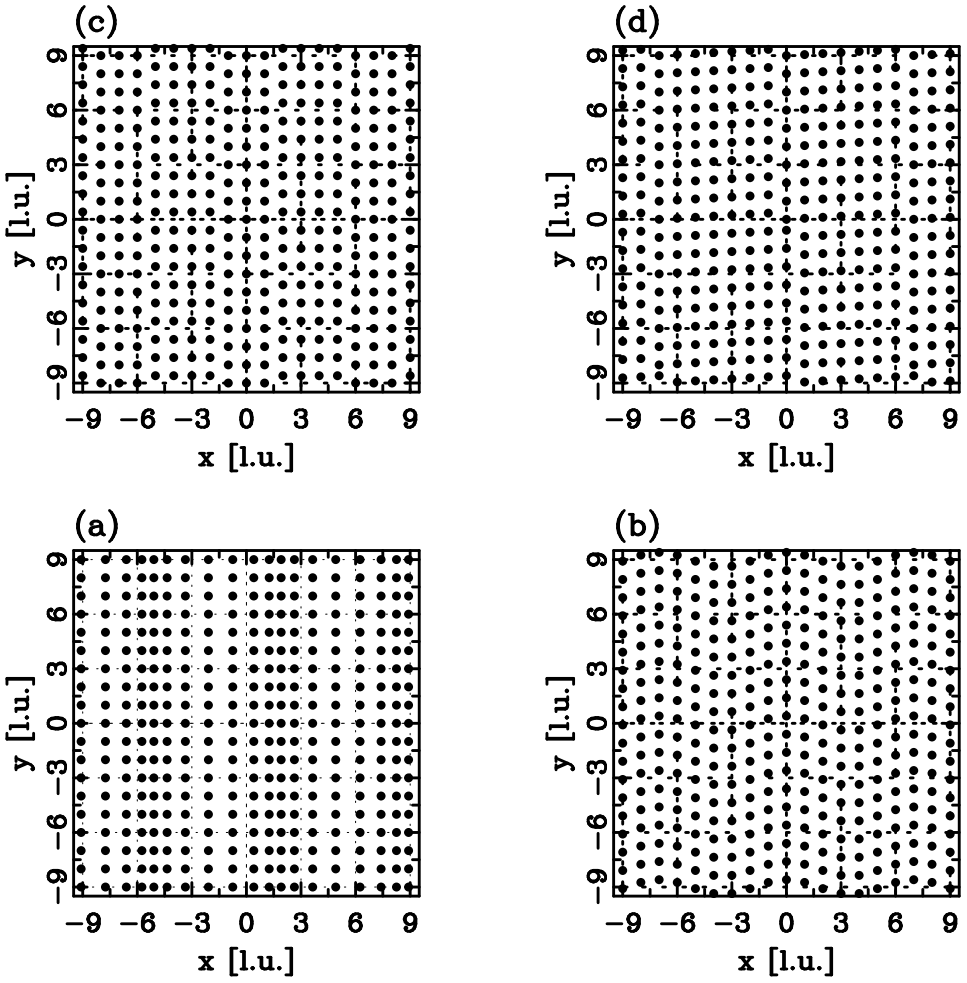

Figure 7.2: Examples of displacement waves

The following macro was used to generate a sinusoidal modulation of a 20x20x1 unit cell cubic test structure with a lattice constant of 5Å and a Zr on (0,0,0).

```
 1  read
 2  cell cell.cll,20,20,1
 3  #
 4  wave
 5  trans
 6  acco
 7  func sinus
 8  len 35.0
 9  amp 2.0
10  phase 0.0
11  shift 0.0
12  vect 1,0,0
13  osci 0,1,0
14  sel zr
```

```
15  run
16  exit
```

After the unit cell is read and expanded to a crystal of 20x20x1 unit cells in size (lines 1-2), the wave segment of *DISCUS* is entered (line 4). Transversal acoustic waves are selected (line 5 and 6) and the wave function is set to sinusoidal (line 7) corresponding to the expression in equation 7.2. Next the wave defining parameters are set: wave length λ (line 8), amplitude A (line 9), phase ψ (line 10), constant A_0 (line 11), propagation vector (line 12) and oscillation vector (line 13). The Zr atom shall be displaced and is selected in line 14. Finally the modulation is computed via the 'run' command (line 15). The other examples shown in figure 7.2 simply use a different wave function $w(\mathbf{r})$.

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. If atoms inside microdomains (see section 7.3) are selected and the phase mode is set to random, the identical wave can only be recreated, if the random number generator is reset by 'seed' each time before the wave function is calculated.

7.2.3 Rotational waves

A new 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 7.3. 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.

7.2.4 Side effects for the Fourier transform

Even for a fairly small crystal, the satellites will be fairly sharp diffraction peaks. Therefore, the statement in section 4.1.1 that the convolution of the diffuse scattering with the Fourier transform of the box function can be neglected no longer holds. If you subtract the average structure to avoid finite size effects, this will not remove the finite size effects from the satellites. Also, be careful when using *lots*. If you use lots that are just a few times larger than the wave length, the satellites will be much broader than they really are.

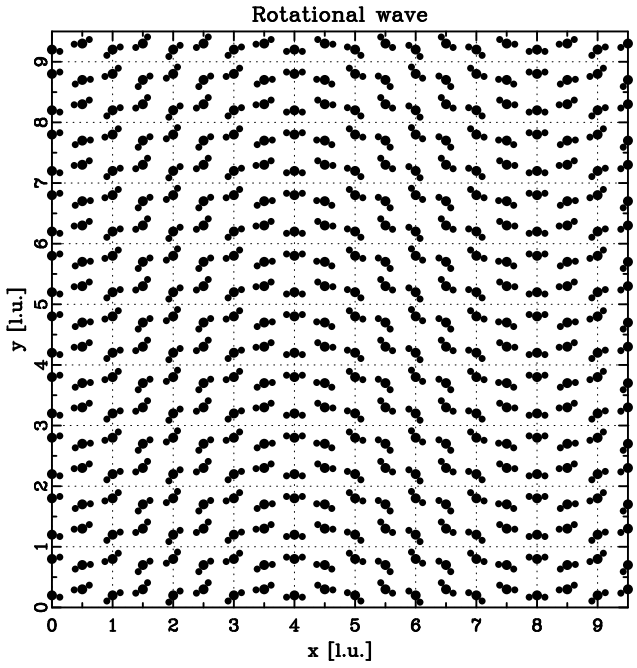

Figure 7.3: Example of a rotational wave

7.3 Microdomains

In the context of this manual the term microdomain refers to any small region intergrown in the lattice of the host crystal. The authors are well aware of the fact that this does not match the exact crystallographic definition of a domain. A microdomain may consist of just a single vacancy with relaxed next neighbours or a completely different structure extending over several unit cells of the host lattice. A theory for diffuse scattering by correlated microdomains was introduced by [14]. Subsequently this theory was applied to the diffuse scattering of cubic stabilized zirconia [13, 20, 19]. *DISCUS* follows another path and offers the possibility to define different microdomain types and distribute these types throughout the crystal. The diffuse scattering is then calculated from the simulated crystal containing the microdomain distribution. Depending on the type of microdomain, the host atoms are removed and replaced by the microdomain atoms. The options are laid out in a very general fashion in order to facilitate generation of many different types of defects. With this technique, small defects, extended defects, anti phase domains, finite wave trains or an incoherent intergrowth of two phases can be generated.

7.3.1 Properties of microdomains

DISCUS uses several properties to characterize the individual microdomains and the distribution of microdomains throughout the crystal. The properties characterizing an individual microdomain are briefly listed below and discussed in somewhat more detail at the end of this section.

- *Shape :*
  The shape of the interface between the microdomain and the surrounding host structure can

take four different forms: a sphere, a set of faces hkl, a rectangular block of $\{100\}$ faces or an irregular group of atoms without a clearly defined boundary.

- *Size :*
  The dimension of the microdomain can be set by the user. These dimensions can either be identical for all microdomains of a given type or be Gaussian distributed.

- *Content :*
  The type of structure that fills the space inside the microdomain. It is usually a previously generated structure.

- *Orientation :*
  Microdomains can have different orientations generated by user defined symmetry operations.

- *Distribution type :*
  DISCUS allows the user to select a random distribution for the microdomains as well as distributing them on a lattice or using a paracrystalline distribution. Furthermore microdomain origins can be read from a file (see below).

- *Correlations :*
  Probabilities for the different microdomain types to be nearest neighbours can be defined. Alternatively, a desired distribution could be generated using Monte Carlo simulations (see section 9).

- *Force to host lattice :*
  The origins of the microdomains can either be at any real space position within the crystal or can be forced to coincide with a host structure lattice point.

Four steps are necessary to modify a crystal by microdomains. First, a list of microdomain types must be set up. Secondly, the distribution of these microdomain types must be characterized. Once the microdomains and their distribution have been characterized, a list of microdomain origins covering the host crystal is set up in the third step. This list can still be edited by the user before, in the fourth step, replacing the structure inside the microdomains by the new structure. These steps can be repeated and after each step the previous steps could be repeated before continuing with the next step. For example a list of microdomain types might be created and the corresponding list of microdomain origins be created. A new list of microdomain types can now be defined as well as a new distribution type. After creating a second list of origins, the crystal structure is finally modified, using both lists of microdomain origins. The program does not delete the previous microdomain types. Instead the new types are added to the list. The first set, however, is not used again. This ensures that a unique microdomain type number exists.

Microdomain distributions : The microdomains can be distributed throughout the crystal by three different distribution types. They may be distributed at random, on a perfect lattice or on a paracrystalline lattice. For the random distribution, an average density of microdomains per host unit cell is to be supplied. Microdomains are created within the crystal at random position until this density is reached. *DISCUS* checks for a possible overlap of microdomains. The procedure

stops once the estimated number of microdomains has been created, or if the introduction of a new microdomain fails for ten consecutive trials. Note, that if the crystal is not limited by a block of unit cells the algorithm will give erroneous densities.

For the lattice types, the user has to supply the base vectors in units of the direct or reciprocal sublattice. A microdomain is introduced at every grid point of the super lattice. No need exists for any commensurate relationship between the two lattices. This type of distribution will create a perfectly periodic distribution of microdomains and subsequently sharp superlattice reflexions will appear.

In case of the paracrystalline distribution, the base vectors represent the average separation between neighboring microdomains. The actual separation is modified by a three dimensional Gaussian distribution with definable sigmas. At any grid point of the paracrystalline superlattice a microdomain is introduced. The position of a new microdomain is calculated from the positions of three previous microdomains. The first microdomain used is the microdomain separated by one time the superlattice base vector **a** in negative direction. The other two are separated by $-\mathbf{b}$ and $-\mathbf{c}$ respectively. The base vector **a** is added to the position of the first microdomain. A Gaussian distributed value is added to each of the three components of the vector. The same procedure is used for the second microdomain with respect to superlattice vector **b** and the third microdomain respectively. The average of the three positions is used as the position for the new microdomain. The distribution of microdomains grows from the negative most corner of the crystal to the positive most corner of the crystal. A paracrystalline distribution results that is characterized by an average separation between microdomains but no long range order. The method described, does not introduce defects into the paracrystalline distribution. Each unit cell of the distribution, though of oblique shape, consists of eight microdomains at the corners. No irregularly shaped unit cells are created, nor do dislocations appear.

DISCUS also allows a microdomain distribution to be read from a file. The file type is identical ⋆ to the normal structure files and the atom type number is taken as type of the microdomain. This enables the user to e.g. generate a microdomain distribution using MC simulations (see section 9).

By default, the origin of the microdomain distribution coincides with the position 0,0,0 within the crystal. The user can define a vector that is added to each microdomain origin before it is inserted into the list. No constrains apply to this vector. Two or more microdomain distributions can be created with different shift and otherwise identical distribution.

The fit of the microdomain origins to the host lattice defines whether the microdomain origin can be at any position within the sublattice or whether it is constrained to integer (and centering) vectors of the sublattice. If constrained, the origin of each microdomain is shifted to the next nearest sublattice point. You can set the type of fit to both values for each distribution type, but the effects on the microdomain distributions will be different for each distribution type. In case of a random distribution, the microdomain origin shifts to the next nearest sublattice vector. Tests for overlap of microdomains are performed after the shift. In case of a lattice distribution, the initial position of each microdomain origin is calculated as an integer multiple of the superlattice base vectors. Each individual microdomain is then shifted to the next nearest sublattice vector without effecting the other microdomains. In case of the paracrystalline distribution the position of each microdomain is calculated from the actual position of three previous microdomains. If the fit type is set to coincide with the sublattice grid, the necessary shift of any one microdomain will effect the next microdomain as well.

A correlation matrix can be set up defining probabilities for two microdomain types to be next

neighbors in case of the lattice or paracrystalline distribution. The correlation matrix is currently used as an isotropic correlation. For both distribution types, the type of a current microdomain is determined from the types of the three microdomains separated by $-\mathbf{a}$, $-\mathbf{b}$ and $-\mathbf{c}$ respectively. The correlations 'grow' from the negative most corner of the crystal to the positive most corner. The correlation matrix C is internally normalized such that the sum of all elements is one. The element c_{ij} gives the probability of microdomain type j to be next neighbor to microdomain type i. The sum of a column c_{i*} gives the a priory probability for the first microdomain to be of type i. The three rows of the correlation matrix representing the three neighbors are averaged and the type of microdomain determined by random number generation weighted with the respective probabilities.

Microdomain properties : For each microdomain type an origin is assumed. The positions of atoms are referenced with respect to this origin, and the origins of the actual microdomains are distributed throughout the crystal. The shape property, defined by the 'boundary' command defines the boundary surface between the microdomain and the surrounding host structure. Currently this boundary surface can be a sphere, a set of planes, a rectangular block of atoms or an irregular conglomerate of atoms limited by a fuzzy boundary. The boundary type 'sphere' creates a sphere of definable radius centered at the microdomain origin. Alternatively a microdomain type can be limited by a set of planes, type 'face'. The user must supply all faces, even symmetrically equivalent faces. The distance of each face from the origin can be set individually, which allows for a very flexible adjustment to different shapes. Since the distance is set with a separate command, the two set of faces (1 0 0) and (2 0 0) represent the identical surface, unless the distance is modified. *DISCUS* tries to check whether the form created by the faces is closed. The origin must be inside the microdomain, while the six points at 10^9 along each of the three sublattice base vectors must be outside. This test is certainly not a foolproof algorithm, yet a reasonable compromise. An open face will include all atoms up to the limits of the crystal. The third boundary type, 'block', is a special case of the second, and identical to the only type offered in *DISCUS* version 1.0. The microdomain is limited by the six $\{100\}$ faces. The distances to the origin are determined from the extend of the structure read. The block defined by these six faces is cut from the host crystal and replaced by the atoms read from the structure file. It is necessary that the content type of these microdomains is a structure. *DISCUS* has no way of finding out, whether the new structure that was read, completely fills up the block, or whether any voids are left. For this reason it is better to use the boundary type 'fuzzy'. The fourth type of microdomain boundaries, 'fuzzy', does not set up a definable surface. Instead the extend of the microdomain is solely defined through the atoms that are read from the structure file. As for shape type 'block' the content must be set to 'structure'. Contrary to type 'block', no voids are created. For all atoms of the host crystal the distance to all microdomain atoms is calculated. If any of these distances is less than a user definable value, the host atom is removed. In effect the boundary between host and microdomain becomes fuzzy and a microdomain of boundary type 'fuzzy' does not even have to be contiguous.

Closely associated with the boundary property is the radius of the boundary surface. This property applies only to boundary types 'sphere' and 'face'. For boundary type 'block' it is determined automatically and for boundary type 'fuzzy' it does not apply. For boundary type 'sphere', the radius gives the radius of the microdomain surface. For boundary type 'face', the radius gives the distance between the microdomain origin and the face. Internally, the hkl of each face are converted to new hkl that represent the distance from the microdomain origin. The average radius

of the boundary surface is the same for all microdomains throughout the crystal. The individual radius can be subject to a Gaussian distribution with user definable sigma. All atoms within this boundary surface are considered to be part of the microdomain structure and are replaced accordingly.

The content of the microdomain can be a completely new structure or a modification of the present structure. If the content is defined as structure, a new structure is read from a file identical in format to the crystal structure file. The space group and lattice constants are currently ignored. All fractional coordinates are interpreted in multiples of the host lattice. This limitation will hopefully be corrected soon. The positions of the atoms are the sum of the microdomain origin and the fractional coordinates read from the file. If applicable, a symmetry operation is applied to the fractional coordinates before adding them to the microdomain origin. The program tests whether the positions are inside the host crystal and inserted if inside. Since *DISCUS* cannot test for left over VOIDs, the user has to make sure that the structure is large enough to cover the whole microdomain.

Each type of microdomains can further be characterized by the orientation of the microdomain structure. Symmetry matrices can be read that will transform the structure and the corresponding boundary surface. The symmetry operation includes both a rotational and a translational part, as described in section 5.2. This symmetry operation is applied to each microdomain atom prior to inserting it into the crystal. The rotation is equally well applied to the boundary of the microdomain, while the translation is not. After the symmetry operation has been applied, *DISCUS* tests whether the atom is inside the microdomain or not. A large translation can therefore move atoms out of the microdomain, and these atoms will not be inserted into the microdomain. If a shift of the microdomain is needed, move the origin of the microdomain with the shift property of the distribution.

7.3.2 Related variables

The crystallographic function 'md_test' and a number of variables listed in table 7.1 are associated with microdomains. Variables marked with '(ro)' are read only variables and can not be altered by the user.

The function 'md_test' returns an integer value which corresponds to the index of the microdomain nearest to the point in real space given by the three arguments x, y, z of the function. As for the variables 'md_next' and 'md_dist', a negative resulting number indicates that the given position is inside the microdomain. In contrast a positive number shows that the given point is outside the returned microdomain.

The use of these variables allows a flexible use of microdomains within the program *DISCUS*. However, the interpreter might be quite slow when working with large model crystals. As an alternative, a microdomain distribution can be generated externally and read from a file in the 'micro' sublevel of *DISCUS*.

7.3.3 Working with microdomains

Most sublevels of *DISCUS* allow not only the selection of atoms to be used for the particular function, the user can also specify if only atoms within or outside of microdomains shall be used by

Variable	Description
md_num[1]	Number of different microdomains (ro).
md_cre[1]	Number of different microdomains already created with the 'create' command but not yet used to modify the crystal (ro).
mc_num[1]	Number of microdomains whose origins have been distributed throughout the crystal with the 'create' command (ro).
md_rad[<i>,<j>]	Radius of the microdomain type <i>. For boundary types 'block' and 'fuzzy' <j> must be 0, for type 'sphere' <j> must be 1 and for 'face' <j> is the number of the face the radius is returned for (ro).
mc_run[1]	Number of microdomains used to modify the crystal with the 'run' command (ro).
mc_type[<i>]	Returns the microdomain type for microdomain <i> (ro).
mc_orig[i,<j>]	Coordinates for microdomain <j> (i=1,2,3 for x,y,z). This variable can only be modified for microdomains <j> which have been created but not used to modify the crystal via the command 'run'.
mc_rad[<i>]	Individual radius for microdomain <i>. This variable can only be altered for microdomains <i> not used with the 'run' command before.
md_next[<j>]	Number of the microdomain nearest to atom <i>. Negative numbers indicate that the atom is within the microdomain, positive numbers are for atoms outside.
md_dist[<i>]	Distance of the atom <i> to the surface of the nearest microdomain. If the number is negative, the atom is inside the microdomain, else outside.

Table 7.1: Variables related to microdomains

selecting the corresponding microdomain status with the command 'sele'. The following choices can be made:

- **Everything :** The microdomain status is ignored and all atoms within the crystal are treated identically.
- **All :** Now all atoms that are within any microdomain are included but all atoms of the host structure outside any microdomain are excluded.
- **None :** Just the opposite from above, only atoms that are outside any microdomain are included.
- **Individual :** This mode allows one to only select atoms that are within a specific microdomain type and to exclude all other atoms.

Once a microdomain distribution is established within the crystal, this selection mechanism allows the user to modify only atoms outside or inside those microdomains, e.g. to generate a modulation wave. A short example shall illustrate the use of the microdomain section of *DISCUS*. The resulting structure containing microdomains in form of a triangle are shown in figure 7.4.

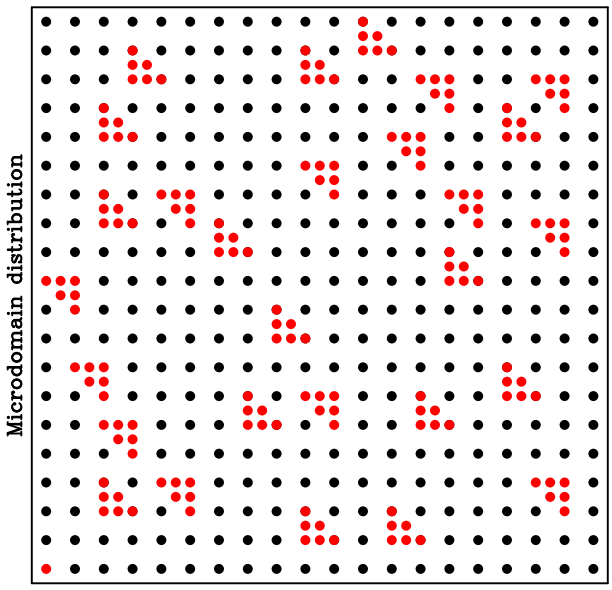

Figure 7.4: Example of microdomain distribution within a crystal

The corresponding macro file is listed below. As in previous examples, the line numbers shown allow easy reference to the different parts of the macro file but are not part of the actual file.

```
 1  micro
 2  init
 3  dist random
 4  dens 0.1
 5  grid host
 6  radi 5.10
 7  bound fuzzy
 8  sep fuzzy,1.0
 9  #
10  content struc,md.inp
11  orient 1
12  insert
13  #
14  mrow 3, 0.0, 0.0,-1.0, 0.0
15  mrow 2, 0.0,-1.0, 0.0, 0.0
16  mrow 1,-1.0, 0.0, 0.0, 0.0
17  orient 2
18  insert
19  #
20  create
21  run
22  exit
```

After the microdomain level is entered (line 1), the sublevel is initialized because we start a new microdomain distribution. A random distribution with a density of 10% is selected (lines 3-4). The generated origins are forced to coincide with the host lattice (line 5) and the size of the microdomains is set to 5.1 Å. This is to prevent microdomains to overlap. The shape of the mi-

crodomain is set to 'fuzzy' (line 7) which means that the actual shape of the microdomain defined by its contents is used. All atoms within the host structure closer than 1.0 Å to the microdomains will be removed (line 8). Next the structure file '*md.inp*' containing the microdomain structure itself is read (line 10). The original structure is inserted as orientation 1 (lines 11-12). A second orientation is obtained by simple inversion given by the matrix defined in lines 14 to 16. This 'upside down' microdomain is inserted as second orientation (lines 17-18). Now the origin distribution is generated (line 20). So far the crystal itself has not been modified. This is done in line 21 when the 'run' command generates the crystal with the given microdomain distribution as seen in figure 7.4.

7.4 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 neighbouring 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' sublevel 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 neighbouring layers of each type must be provided. Thus for 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 7.2.

- *Uncertainties for translation vectors:*
  In some materials small deviations in the translation vectors might occur. This behaviour 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 neighbours. No further correlations are taken into account.

- *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 moduli 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	$\frac{1}{2}(\overline{1},\overline{1},0)$
B	B	$(1,1,1)$
B	C	$\frac{1}{2}(0,1,1)$
C	A	$\frac{1}{2}(\overline{1},0,\overline{1})$
C	B	$\frac{1}{2}(0,\overline{1},\overline{1})$
C	C	$(1,1,1)$

Table 7.2: 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.

$$\rho(\mathbf{r}) = \sum_{i=1}^{nl} \left\{ \sum_{j=1}^{no} o_{ij}(\mathbf{r}) \right\} \star l_i(\mathbf{r}) \tag{7.7}$$

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

$$\mathcal{F}\{\rho(\mathbf{r})\} = \sum_{i=1}^{nl} \mathcal{F}\left\{ \sum_{j=1}^{no} o_{ij}(\mathbf{r}) \right\} \cdot \mathcal{F}\{l_i(\mathbf{r})\} \tag{7.8}$$

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.

A simple example of the stacking fault segment of *DISCUS* will be given here. The resulting layered structure is shown in figure 7.5. The filled circles represent atoms of layer type A whereas the empty circles stand for atoms of layer type B. The layers are stacked in y-direction and are shifted in x-direction by ± 0.5 lattice units between the layers. The structure consists of preferred AB and BA sequences, however AA and BB pairs are present as well.

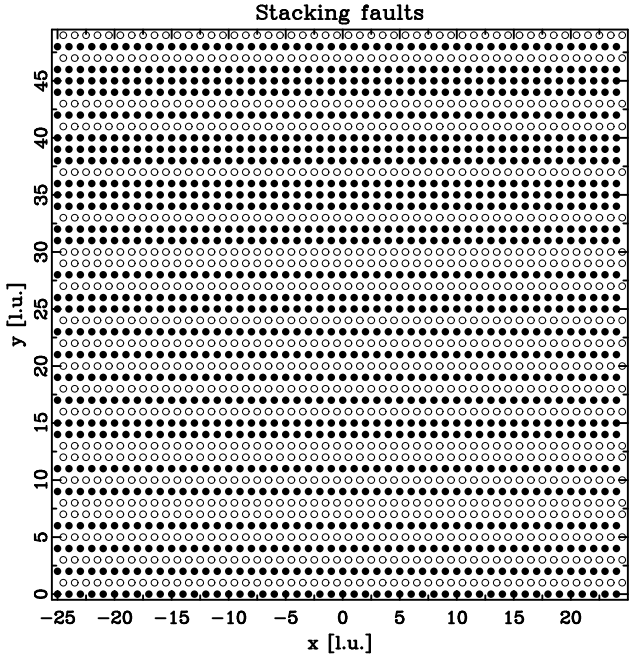

Figure 7.5: Example of stacking faults

The following *DISCUS* macro file was used to generate the layers structure shown in figure 7.5. Note that the line numbers are only given for easy reference and are not part of the actual macro file.

```
 1  stack
 2  layer l1.stru
 3  layer l2.stru
 4  number 50
 5  #
 6  trans 1,1, 0.0,1.0,0.0
 7  trans 1,2, 0.5,1.0,0.0
 8  trans 2,1,-0.5,1.0,0.0
 9  trans 2,2, 0.0,1.0,0.0
10  #
11  crow 1,0.3,0.7
12  crow 2,0.7,0.3
13  #
14  mod 1,0,0,0,0,1
15  set mod,on
16  set trans,aver
17  show
18  #
19  create
20  run
21  exit
```

After entering the 'stack' level of *DISCUS*, the files containing the structures of layer type A (line 2) and B (line 3) are specified. In this case those structures are rows of 50 atoms in

x-direction. The next command (line 4) sets the number of layers to be stacked to 50. The translation vectors between layer types AA, AB, BA and BB are set to $(0,1,0), (\frac{1}{2},1,0), (-\frac{1}{2},1,0)$ and $(0,1,0)$ respectively (lines 6-9). The correlation matrix is given in rows and gives a probability for AA and BB neighbours of 30% and AB or BA neighbours of 70% (lines 11-12). Note that the sum of all elements in the correlation matrix is actually not unity but two. *DISCUS* internally normalizes those values to give correct probabilities. In lines 14 to 16 the shape of the resulting crystal is determined. The resulting crystal is limited in x- and z-direction and the crystal growth is determined by the average of the given translation vectors weighted by the appropriate probabilities. Finally the origin distribution is created (line 19) and the resulting structure is generated (line 20).

For more information refer to the command reference, the online help or study the stacking fault example of the interactive *DISCUS* tutorial (see section 1.4).

Chapter 8

Analyzing defect structures

After describing how to create and change structures this chapter will give a summary of *DISCUS* functions to analyze a given disordered structure. All those commands are accessed via the 'chem' sublevel 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 commands. Note that not all commands are available in molecule mode.

8.1 Occupancies

The command 'elem' will display the chemistry of the current model crystal. Depending on the selected mode ('atom' or 'mole'), the relative abundance of the present atom or molecule types is written on the screen. A sample output can be seen below:

```
Size of the crystal (unit cells) :    5 x    5 x    5
 Total number of atoms            :   1500
 Number of atoms per unit cell    :     12
 Number of different atoms        :      2

    Element : VOID(0)   rel. abundance :  .065  (    98 atoms)
    Element : ZR(1)     rel. abundance :  .333  (   500 atoms)
    Element : O(2)      rel. abundance :  .601  (   902 atoms)
```

Some general information about the crystal is given as well as the number and relative abundance for the atom types present in the crystal. The results are saved in the variable res[i] (see chapter 3.1). However, the relative abundance 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 8.3) distributions within the crystal using the command 'homo'. This is achieved by sampling the crystal using 'lots' analogous to those used to calculate smooth diffraction patterns (see section 4.1.2). The following short macro illustrates the use of the 'homo' command. The 'chem' sublevel is entered in line 1. The size of the sample volume is set to an ellipsoid of 5x5x1 unit cells. A total of 2000 lots will be used (line 2). Next the number of bins is set to 20 (line 3). Obviously the largest number of bins corresponds to one

CHA

over
conc
write

F

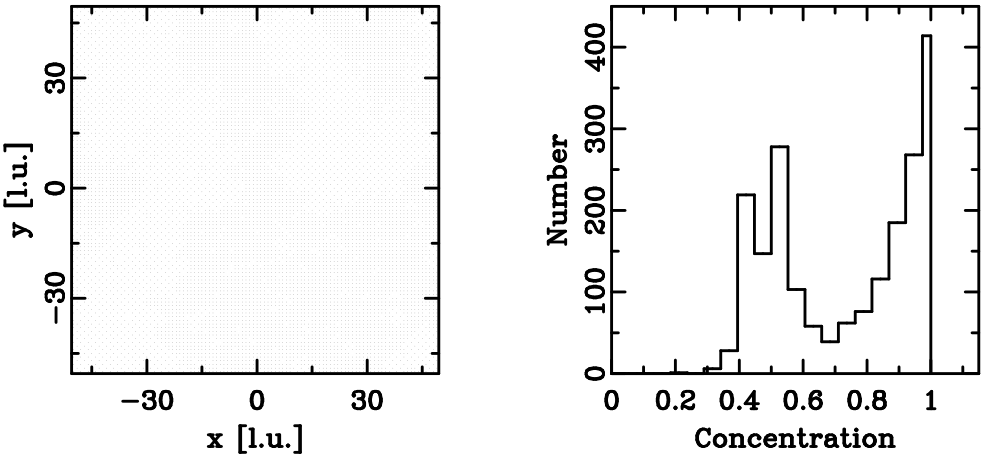

Figure 8.1: Analysis of crystal homogeneity

In this example the relative abundance of Zr given by 'elem' is 0.75. The command 'homo' determines an average of 0.76 ± 0.05. Viewing figure 8.1 reveals that the crystal actually consists of two phases, one with a concentration of 0.5 and one with a concentration of 1.0. This can also be seen in the concentration distribution on the right hand side of the same figure. The two maxima correspond to a concentration of 0.5 and 1.0. By adjusting the size of the sample volume ('lots') additional information about the size of the phase regions can be obtained.

8.2 Distortions

After analyzing concentrations within a crystal, this section will focus on the analysis of distortions. 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 limited 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 screen output for the above example looks like this:

```
Calculating bond-length distribution
   Allowed range :  4.50 A to  5.50  A / File : chem.2.blen  ( 250 pts)
```

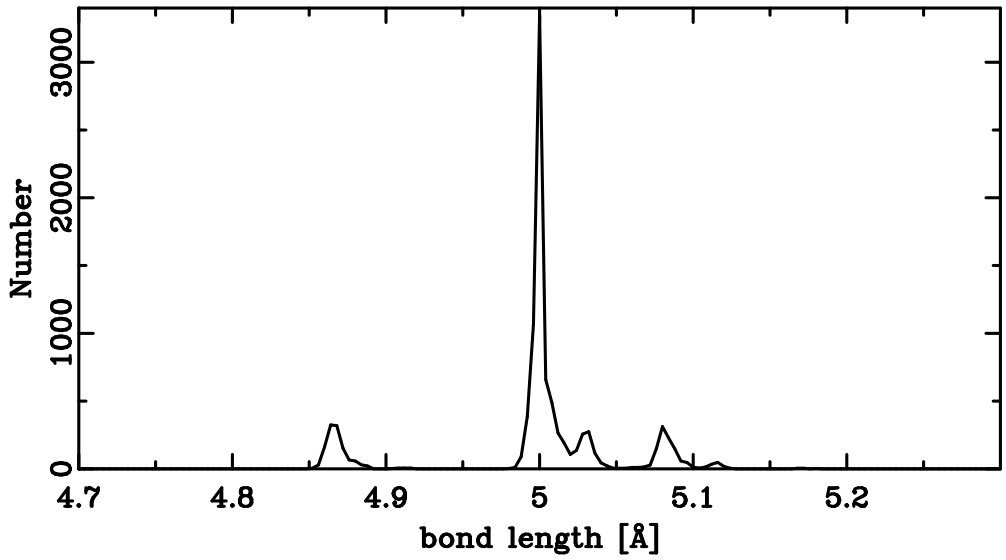

Figure 8.2: Example for bond length distribution

In this example the Zr-vacancy distances are shorter than the average distance of 5.0Å whereas the Zr-Zr distances are slightly longer. This describes *size effect* like distortions where the atoms surrounding a vacancy are distorted towards the vacancy. All vacancies are at their average position indicated by the average distance of 5.0Å with a standard deviation of 0.0. An example how to create these kinds of disordered structures using Monte Carlo simulations is given in chapter 9. The resulting bond length distribution from the previous example is shown in figure 8.2. The individual peaks correspond to the various Zr-vacancy and Zr-Zr distances. Note that the output of *DISCUS* given above only lists the average distances for the specified bond length interval, e.g. there are some Zr-vacancy distances longer than 5Å and some Zr-Zr distances shorter than the average distance (see minimal and maximal values above). Thus it can be hard to identify the individual peaks present in figure 8.2. One way to obtain further information would be to calculate the bond length distributions for Zr-Zr ('blen zr,zr,zz.blen') and Zr-vacancy ('blen zr,void,zv.blen') individually.

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 neighbour 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 neighbour 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
```

More examples of neighbour definitions can be found in section 8.3. The output of the 'disp' command is shown below:

```
Calculating distortions
   Atom types : A =  all and B =  all

   Neig.  Atom A      Atom B       distance   sigma     # pairs
   ------------------------------------------------------------
     1   VOID(0)     VOID(0)         5.000      .000        332
     1   VOID(0)     ZR(1)           4.880      .037        372
     1   ZR(1)       ZR(1)           5.022      .032       3824
```

Again average distances and standard deviations are listed giving a Zr-vacancy separation shorter than 5 Å and a Zr-Zr distance slightly longer. However, the actual numbers are different from the results of the 'blen' command. The reason for this is that 'blen' averaged **all** distances in the given range whereas the second example is limited to neighbours in x-direction. By adding the other directions to the neighbour definitions the same result as before would be achieved.

8.3 Correlations

Diffuse scattering contains two-body information of the system under investigation and is thus a valuable source how atoms or molecules interact. In contrast Bragg scattering can only reveal *average* one-body information such as atomic positions, site occupancies and thermal ellipsoids. 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 effecting 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. The command 'homo' (see section 8.1) can also be used to determine the homgeniety of correlations within the crystal.

8.3.1 Occupational correlations

First occupational correlations will be discussed. One definition of the correlation coefficient c_{ij} between a pair of sites i and j based on a statistical definition of correlation [28] is given in equation 8.1.

$$c_{ij} = \frac{P_{ij} - \theta^2}{\theta(1-\theta)} \tag{8.1}$$

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 ($P_{ij} = \theta$). This definition of correlations can easily be converted to the Warren-Cowley short-range order parameter $\alpha^{ij}_{lmn} = 1 - P^{ij}_{lmn}/\theta$ [5]. As an example the correlations $c_{\langle 10\rangle}$ and $c_{\langle 11\rangle}$ of the disordered structure shown in figure 8.1 are calculated using the *DISCUS* macro file listed below:

```
 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' sublevel is entered (line 4) periodic crystal boundaries are selected (line 6). The parameter 'quick' selects a faster neigbouring finding algorithm which only works for crystals arranged in the *DISCUS* storage order (see section 2.2). Note that $c_{\langle 10\rangle}$ stands for the nearest neighbour 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 neigbouring 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 neighbouring 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 neighbouring definition and allows the definition of a new one. Finally the correlations for the defined neighbouring directions are calculated (line 22). The screen output for the disordered structure shown in figure 8.1 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 neighbours in <10> directions tend to be avoided. Neighbouring 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 $c_{\langle 11\rangle}$. This leads to the large areas with a doubled unit cell and a Zr concentration of 0.5 as seen in section 8.1.

8.3.2 Displacement correlations

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

$$c_{ij} = \frac{\langle x_i x_j \rangle}{\sqrt{\langle x_i^2 \rangle \langle x_j^2 \rangle}} \tag{8.2}$$

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 neighbours 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 neighbouring site.

8.3.3 Correlation fields

In the previous sections a correlation c_{ij} for a given pair of neighbouring 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').

Again taking the disordered structure displayed in figure 8.1 as an example, the correlation fields in <10> and <11> direction are shown in figure 8.3. This figure shows the correlation between a site i and sites separated by integer multiples of the vectors used in the neighbor definitions. The neighbouring definitions are the same as in the example for a single correlation coefficient in section 8.3.1. The first neighbour correlation is negative for $c_{\langle 10\rangle}$ and positive for $c_{\langle 11\rangle}$ as in section 8.3.1. The correlation $c_{\langle 11\rangle}$ is decaying as a function of the distance within the crystal, giving a measure of the extension of the area with preferred <11> vacancy neighbours. Eventually the value gets negative for distances above about 26 times the $c_{\langle 11\rangle}$ vector which can be

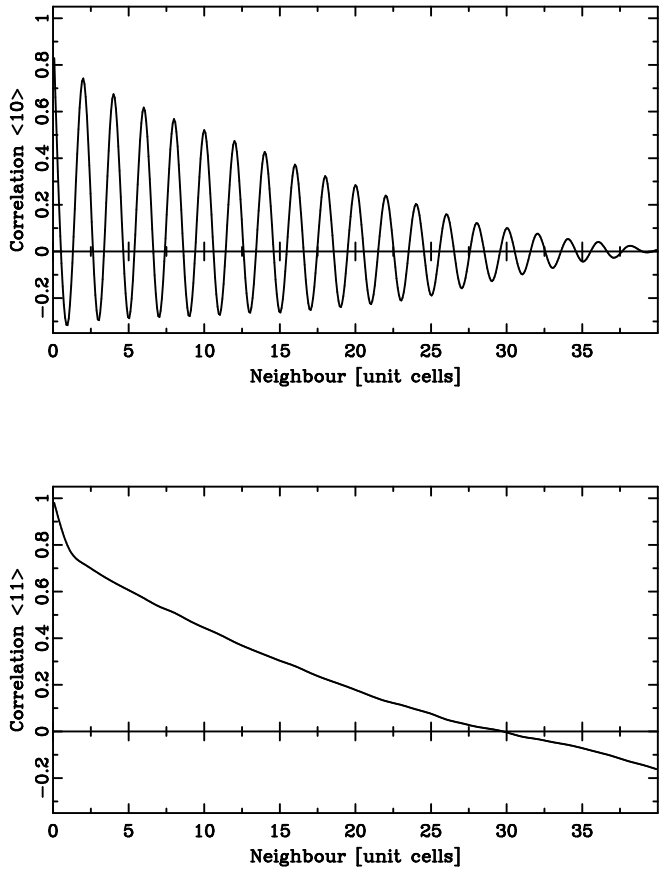

Figure 8.3: Example for a correlation field

understood taking the size of the two phase regions (figure 8.1) into account. The absolute value of correlation $c_{\langle 10\rangle}$ decays as well but oscillates between negative an positive values. This can be understood by thinking of a perfect ABAB sequence. All odd neighbours (i.e. 1,3,5, ...) are AB or BA resulting in a negative correlation whereas even neighbours (i.e. 2,4,6, ...) are AA or BB giving a positive correlation.

8.4 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 neighbourhood of a given atom or molecule can be examined using the commands 'neig' and 'env'. The command 'neig' will use the currently stored neighbour 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 2.2 or via the command 'trans' in the 'chem' level of *DISCUS*.

Chapter 9

Monte Carlo simulation

In general Monte Carlo methods can be described as statistical simulation methods involving sequences of random numbers to perform the simulation. In the past several decades this simulation technique has been used to solve complex problems in nuclear physics, quantum physics, chemistry as well as for simulations of e.g. traffic flow or econometrics. The name 'Monte Carlo' was coined during the Manhattan Project of World War II, because of the similarity of statistical simulation to games of chance, and because the capital of Monaco was a center of gambling. In this analogy the 'game' is a physical system and the scientist might 'win' a solution for his particular problem. An excellent application for this kind of statistical methods is the study of diffuse scattering and subsequently the solution of the underlying defect structure. One possible approach is the (direct) Monte Carlo (MC) modeling of a defect structure described in this section. Further information about the MC method in solid state physics can be found in [2]. A review article covering MC simulations to analyze diffuse scattering was written by [30]. An alternative method using the MC algorithm is the so-called Reverse Monte Carlo simulation technique described in chapter 11.

9.1 Introduction

A brief summary of the MC algorithm originally invented by [12] should be given here. 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

$$P = \frac{\exp(-\frac{\Delta E}{kT})}{1+\exp(-\frac{\Delta E}{kT})} \tag{9.1}$$

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-neighbour 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.

9.2 Introducing correlations

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.

9.2.1 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:

$$E_{occ} = \sum_i H\sigma_i + \sum_i \sum_n J_n \sigma_i \sigma_{i-n} + \ldots \tag{9.2}$$

The sums are over all sites i and neighbours n. The value σ_{i-n} refers to the occupancy (spin) of the neighbouring site $i-n$ of site i. The quantities J_n are pair interaction energies corresponding to the neighbouring 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 [1] 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.

The following example macro was used to generate the disordered structure shown in figure 8.1.

```
1  mc
2  set neig,rese
3  #
4  # <10> neighbours
5  #
6  set vec,1,1,1, 1, 0, 0
7  set vec,2,1,1,-1, 0, 0
8  set vec,3,1,1, 0, 1, 0
```

[1] A MC cycle is defined as the number of MC steps needed to visit every crystal site once on average

```
 9  set vec,4,1,1, 0,-1, 0
10  set neig,vec,1,2,3,4
11  set neig,add
12  #
13  # <11> neighbours
14  #
15  set vec,5,1,1, 1, 1, 0
16  set vec,6,1,1,-1,-1, 0
17  set vec,7,1,1,-1, 1, 0
18  set vec,8,1,1, 1,-1, 0
19  set neig,vec,5,6,7,8
20  #
21  set atom,zr,void
22  set mode,swchem
23  set energy,cocc,all
24  #
25  set target,1,-0.3
26  set target,2, 0.8
27  #
28  set temp,1.0
29  set feed,100*100
30  set cyc,200*100*100
31  run
32  #
33  exit
```

After entering the 'mc' sublevel of *DISCUS* (line 1) the <10> and <11> neighbours are defined the same way as for the calculation of correlations (see section 8.3.1). Note that the command 'set neig,add' (line 11) is not given after the last neighbour definition (lines 15–19), because the actual neighbour definition is used anyway and the command 'set neig,add' puts the current definition on a stack of definitions to be used. If you would give the command 'set neig,add' after line 19 again, the second neighbour definition would be used twice.

The atoms to be used for the MC simulation are Zr and vacancies (line 21) and the operating mode is 'swchem'. More information about the different operation modes are given in section 11.4 in the RMC chapter of this manual. The energy to be used is for occupational disorder as defined in equation 9.2 (line 23). Next the desired target correlations of $c_{\langle 10\rangle} = -0.3$ and $c_{\langle 11\rangle} = 0.8$ are set (lines 25–26). The value of kT is set to 1.0[2] A total number of 200*100*100 MC moves will be computed (line 30). We will refer to a MC cycle as the number of MC moves necessary to visit every crystal site once on average. In our case 100x100 steps correspond to one MC cycle, thus we run the simulation for 200 cycles. Note that the arguments for the commands 'set cyc' and 'set feed' are given as number of moves rather than cycles. Each move consists of a single swap of two randomly chosen atoms. The feedback should be executed every cycle, i.e. every 100x100 moves for the crystal size of 100x100x1 unit cells (line 29). The MC simulation is started with the 'run' command (line 31).

Every 10000 moves or every cycle (interval set by 'set feed') the following information is written to the screen:

```
Gen: 10720699 try:  2000000 acc: (good/bad):   48371 /   59000  MC moves
    Neig.   1  const:      0.065   achieved: -0.306  target: -0.300
```

[2]This is recommended when using the feedback mechanism since the J_n values are then determined in units of kT.

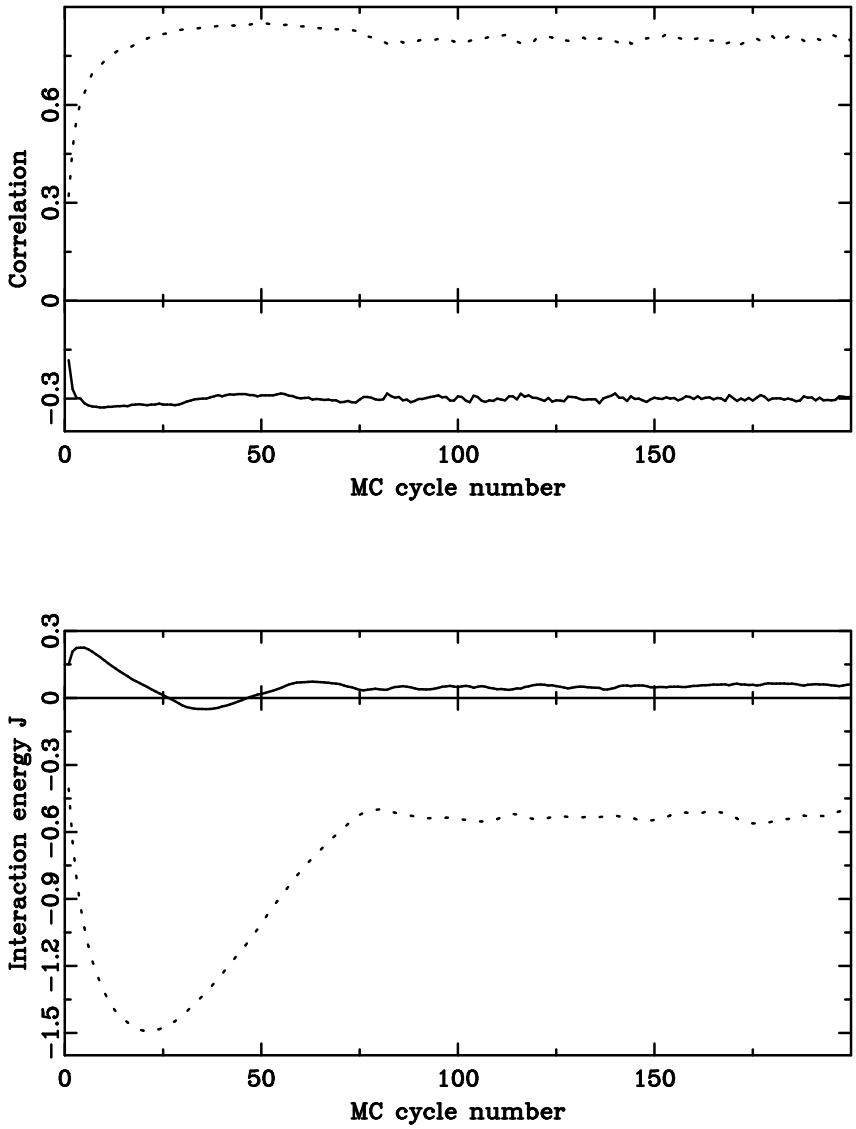

Figure 9.1: Interaction energies and correlations during a MC simulation

```
Neig.   2  const:     -0.485   achieved:  0.788  target:  0.800
```

The first line contains the number of generated and tried MC steps followed by the number of accepted moves with $\Delta E < 0$ (good) and $\Delta E > 0$ (bad). Next the interaction energy J_n and the achieved and desired correlation value is given for each neighbour definition. Note that the mode 'swchem' actually swaps the atom types of two sites and therefore the overall concentration of the atom types remains constant and no term H is needed in the Hamiltonian. Figure 9.1 shows the interaction energies J_n and achieved correlations as a function of the MC feedback number (line 29 of the example). The solid line represents the <10> neighbour energies and correlations whereas the dashed line marks <11> neighbours. It can be seen that after about 75 cycles correlations

and interaction energies J_n have stabilized. Rather than using the feedback mechanism to achieve a certain set of correlation values, a disordered structure can directly be created by entering the interaction energy values J_n using the command 'set const' in the 'mc' sublevel. In this case the temperature T becomes a parameter of the modeling. The command 'set const' also allows to influence the feedback process by altering the size of the change of J_n for a given difference between achieved and desired correlation value.

9.2.2 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 8.3.2. 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:

$$E_{dis} = \sum_i \sum_n J_n x_i x_{i-n} \tag{9.3}$$

Here, as before, the first sum is over all sites i and the second sum is over all neighbours n of the site i. In this equation is no term depending solely on x_i 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 11.4.

Note that the displacements x_i are taken in the direction defined by 'set neig,dir' and the energy defined in equation 9.3 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 neighbouring atoms, the shifts should be restricted via the command 'set move' to the xy-plane.

9.3 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, using a Hamiltonian (equation 9.4, where the atoms or molecules move in harmonic potentials (Hooke's law).

$$E_h = \sum_i \sum_n k_n [d_{in} - \tau_{in} d_0]^2 \tag{9.4}$$

The sums are over all sites i within the crystal and all neighbours n around site i. The distance between neighbouring 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 neighbour type n.

The following example will illustrate the usage of the MC distortion mode. A *size effect* [4, 29, 5] like distortion shall be introduced in the disordered structure created in section 9.2.1. Neighbouring Zr atoms will be shifted towards the vacancies present in the crystal. The Zr-Zr distance shall be slightly larger than average to keep the average structure constant. Here is the *DISCUS* macro file to create the distortions.

```
 1  mc
 2  set neig,rese
 3  #
 4  # <10> neighbours
 5  #
 6  set vec,1,1,1, 1, 0, 0
 7  set vec,2,1,1,-1, 0, 0
 8  set vec,3,1,1, 0, 1, 0
 9  set vec,4,1,1, 0,-1, 0
10  set neig,vec,1,2,3,4
11  #
12  set atom,zr,void
13  set mode,shift
14  set move,zr,0.05,0.05,0.00
15  set energy,disp
16  #
17  set target,1,Zr,Zr,  5.05
18  set target,1,Zr,Void,4.80
19  set const, 0,25.0
20  set const, 1,10.0
21  #
22  set feed,100*100
23  set cyc, 200*100*100
24  #
25  run
```

Lines 1 to 12 are similar to the example given in section 9.2.1, i.e. defining the neighbours (here only <10>) and selecting the atoms. Now 'shift' mode is selected (line 13) which displaces the selected atoms by a Gaussian distributed random amount in a random direction. The width of the distribution is set to 0.05 lattice units in x and y-direction (line 14). Since the model is only two dimensional, no shifts in z-direction should be created. Next the Hamiltonian according to equation 9.4 is selected (line 15). The desired distances are set to $d_{zz} = 5.05$Å for Zr-Zr <10> neighbours and to $d_{zv} = 4.80$Å for Zr-vacancy pairs. Note that the ratio of the distortions corresponds to the average ratio of Zr-Zr and Zr-vacancy pairs for this disordered crystal. The force constant k in equation 9.4 is set in lines 19 and 20. The constant k_0 (line 19) describes a restoring force to the lattice, i.e. a situation where $\tau_{in} = 0$. The force constant for k_0 in this example is set higher than the value k for the nearest neighbour interaction to avoid atoms moving to far away from their average position. Every neighbour definition can have an individual value of k. The settings in lines 22 and 23 correspond to those of the example in section 9.2.1. Finally the MC simulation is started using the 'run' command (line 25).

9.4 Working with molecules

DISCUS allows the use of molecules. The command 'set mole' selects the molecule types to

be used for the MC simulation and automatically switches *DISCUS* to molecule mode. On the other hand 'set atoms' will return to atoms mode and select the atom types to be used for the MC simulation.

All neighbouring definitions work exactly as for atoms. However, the site label used to define neighbouring 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 neighbours 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 5.2) and subsequently using the 'swap chemistry' mode.

Chapter 10

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. It might be interesting to know about the existence of the program *PDFFIT* [17] which allows the full profile structural least square refinement of a PDF. *PDFFIT* uses a command language and structure file format similar to *DISCUS* and can be obtained from the same source.

10.1 Introduction

The determination of crystal structures is an important part of chemistry, physics and of course crystallography. Conventional structure determination is based on the analysis of the intensities and positions of Bragg reflections which only allows one to determine the long range *average* structure of the crystal. Determination of the *average* structure based on powder diffraction data is now routinely done using the Rietveld [24] method. As we have discussed before, any deviations from the *average* structure result in the occurance of diffuse scattering which contains information about two-body interactions [31, 7]. One method to reveal the *local* structure of a crystals is the analysis of the PDF. This method is long known in the field of studying short range order in liquids and glasses but has recently been applied to crystalline materials [6, 1]. The PDF is obtained from the powder diffraction data via a simple Fourier transform of the normalized scattering intensity $S(Q)$:

$$G(r) = 4\pi r[\rho(r) - \rho_0] = \frac{2}{\pi}\int_0^\infty Q[S(Q) - 1]\sin(Qr)dQ, \tag{10.1}$$

where $\rho(r)$ is the microscopic pair density, ρ_0 is the average number density and Q is the magnitude of the scattering vector, for elastic scattering $Q = 4\pi\sin(\theta)/\lambda$ with 2θ being the scattering angle and λ the wavelength of the radiation used. Details about the determination of an experimental PDF can be found e.g. in [6, 27] and are not discussed here.

Since the PDF contains Bragg and diffuse scattering, the information about *local* arrangements is preserved. The PDF can be understood as a bond-length distribution between all pairs of atoms i and j within the crystal (up to a maximum distance), however each contribution has a weight

corresponding to the scattering power of the two atoms involved. The PDF of a given structure can be calculated using the relation:

$$G_c(r) = \frac{1}{r}\sum_i\sum_j\left[\frac{b_i b_j}{\langle b\rangle^2}\delta(r-r_{ij})\right] - 4\pi r\rho_0, \tag{10.2}$$

where the sum goes 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. Generally there are two different ways to account for displacements (either thermal or static) from the average position. First one can use a large enough model containing the desired displacements and perform an ensemble average. This is the method used by *DISCUS* where thermal displacements can be introduced according to a given (isotropic) Debye-Waller factor. Secondly one can convolute each contribution given by $\delta(r-r_{ij})$ in (10.2) with a Gaussian accounting for the displacements which is done e.g. in *PDFFIT*.

In order to carry out the Fourier transform in (10.1) we would need to measure data up to $Q = \infty$, which of course is not possible. Thus the termination at a value of $Q = Q_{max}$ will cause so-called *termination ripples* in the PDF which can be simulated by convoluting the calculated PDF with the Fourier transform of a box function. With the availability of modern synchrotron and neutron sources it is possible to collect powder diffraction data up to high values in Q, however in many cases a sufficient PDF can be obtained using a conventional X-ray tube. One last correction applied to the calculated PDF, $G_c(r)$, accounts for the limited resolution of the experiment in Q-space. This leads to a decrease of the PDF peak as a function of r according to the relation $\exp(-\sigma_Q^2 r^2/2)$. A detailed discussion of the accuracy of PDF analysis is given in [26].

10.2 Calculating the PDF

The calculated PDF for Nickel is shown in figure 10.1. The PDF was calculated for two different situations: The PDF shown as dotted line was calculated without applying convolution given by Q_{max}. This is achieved in *DISCUS* by setting the value of Q_{max} to zero. The second PDF shown as solid line in figure 10.1 was calculated for $Q_{max} = 20\text{Å}^{-1}$. The resulting termination ripples are clearly visible.

```
 1 read
 2 cell ni.cll,5,5,5
 3 #
 4 therm
 5 #
 6 pdf
 7 set rang,10.0,0.02
 8 set qmax,20.0
 9 set qsig,0.0
10 set rad,xray
11 #
12 calc
13 #
14 save pdf,ni.pdf
```

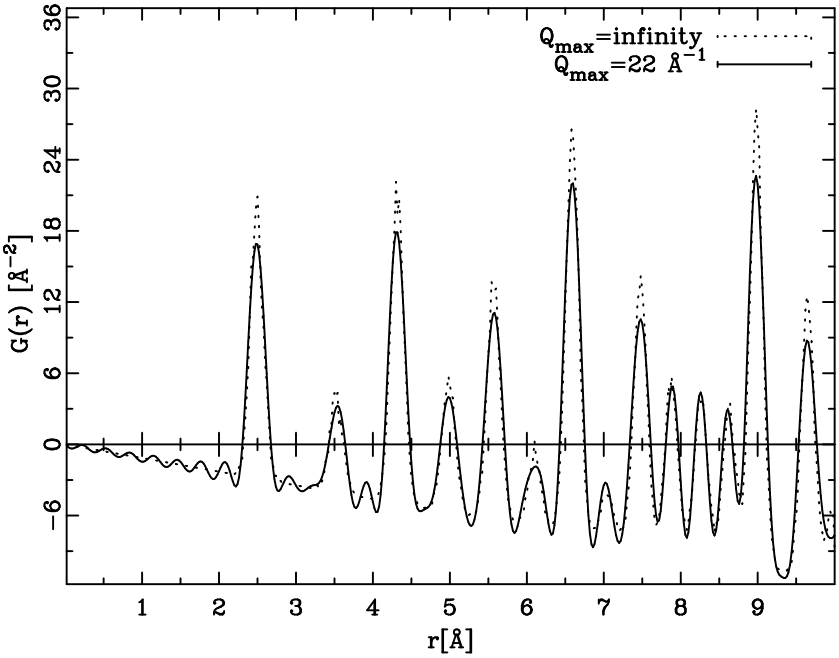

Figure 10.1: Calculated PDFs of *Ni*

```
15 exit
```

The macro file used to calculate the Nickel PDFs is listed above. First we read the unit cell for Nickel and expand it to a size of 5x5x5 unit cells (lines 1–2). Next we introduce thermal vibrations according to the given isotropic Debye-Waller factor (line 4). After entering the PDF sublevel (line 6) we specify the maximum value of r and grid size Δr (line 7). In our case we calculate up to a value of $r = 10$Å using a grid of $\Delta r = 0.02$Å. The next command (line 8) specifies the value of Q_{max} to be set to 20Å^{-1}. Finally we set σ_Q to zero (line 9) meaning no Q-resolution correction and select X-ray radiation (line 10). Now we are ready to calculate the PDF, done in line 12. All we have to do now is to save the result to a file (line 14). The result is shown in figure 10.1.

10.3 Refining a PDF

In principle an experimental PDF can be refined based on a structural model in two different ways. A relatively small model can be refined using *PDFFIT*. On the other hand a larger model can be refined using the Reverse Monte Carlo (RMC) algorithm in *DISCUS*. Details about the principle of RMC are discussed in chapter 11 of this manual. The only difference is that rather than refining the scattering intensity directly, the PDF is refined. An example refinement of a Nickel PDF is part of the online tutorial of *DISCUS*.

Chapter 11

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 10). A more detailed description of the various commands can be found in the reference manual or the online help function. Additionally there are RMC demonstration macros included in the interactive tutorial.

11.1 Introduction

The **R**everse **M**onte **C**arlo (RMC) method [11] is another application of the Monte Carlo algorithm discussed in chapter 9. 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* [16].

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 11.1 is computed.

$$\chi^2 = \sum_{i=1}^{N} \frac{[I_e(\mathbf{h}_i) - I_c(\mathbf{h}_i)]^2}{\sigma^2} \tag{11.1}$$

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 **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 9. How many 'bad' moves are accepted depends on the value of σ or T.

Although this method generally is able to generate **one** disordered structure consistent with the observed data, the question remains if the resulting structure in plausible from a chemical or physical point of view. Certain constrains integrated in *DISCUS*, e.g. user defined minimal allowed distances between different atom types, allow to avoid certain unlikely structures. A series of test RMC refinements [22], using scattering data from simulated disordered structure

with know properties as input, investigated the viability of the RMC method. It could be shown that the RMC method is capable to determine the correct type of defect structure even for systems showing occupational and displacement disorder in combination. Absolute values, however, must be interpreted with care. RMC studies of the diffuse scattering of stabilized zirconia [23] and $TlSbOGeO_4$ [32] revealed a problem regarding the size of the model crystal chosen for the RMC refinement. Large crystal sizes allow the RMC 'fit' to be obtained by adjusting many longer range interactions rather than the few short range interactions of interest (in most cases). A small model crystal results in a poor statistical description of the defect structure and noisy diffraction patterns. A new extension of the RMC segment allows to use 'lots' as described in section 4.1.2 during the ⋆ RMC refinement. Thus a large crystal is used, but the RMC process acts only in the limited area of one lot. First RMC simulations using this new method are described in [21].

11.2 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 11.2.

$$\chi^2 = \sum_{i=1}^{N} \frac{w(\mathbf{h}_i)[I_e(\mathbf{h}_i) - (f \cdot I_c(\mathbf{h}_i) + b)]^2}{\sigma^2} \tag{11.2}$$

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 11.3.

$$f = \frac{\sum\limits_{i=1}^{N} w(\mathbf{h}_i) I_e(\mathbf{h}_i) I_c(\mathbf{h}_i) - b_0 \sum\limits_{i=1}^{N} w(\mathbf{h}_i) I_c(\mathbf{h}_i)}{\sum\limits_{i=1}^{N} w(\mathbf{h}_i) I_c^2(\mathbf{h}_i)} \tag{11.3}$$

Alternatively both values f and b can be refined during the RMC refinement. Equation 11.4 shows the corresponding definitions.

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

$$b = \frac{\sum_{i=1}^{N} w(\mathbf{h}_i) I_e(\mathbf{h}_i) - f \cdot \sum_{i=1}^{N} w(\mathbf{h}_i) I_c(\mathbf{h}_i)}{\sum_{i=1}^{N} w(\mathbf{h}_i)} \tag{11.4}$$

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.

11.3 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 7.1) 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 6 and 7.

It is obviously important to avoid finite size effect contributions to the calculated diffuse scattering. In cases where the average structure will remain constant during the RMC refinement, the average structure factor <F> might simply be subtracted. The corresponding command 'set aver' is the same as used in the Fourier transform part (see section 4.1.1) of *DISCUS*. However, in cases where <F> might change significantly periodic boundary conditions must be applied (see chapter 4.1.1). Now the required size of the crystal is determined by the resolution of the experimental data to be used. A grid size of $\Delta h = 0.05 r.l.u.$ for example would require a model crystal of 20x20x20 unit cells. Limitations of computer memory may require a rebinning of the experimental data to a larger grid in order to reduce the size of the model crystal.

11.4 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 11.1. Additionally user defined moves can be included in an external subroutine linked to the program *DISCUS.*

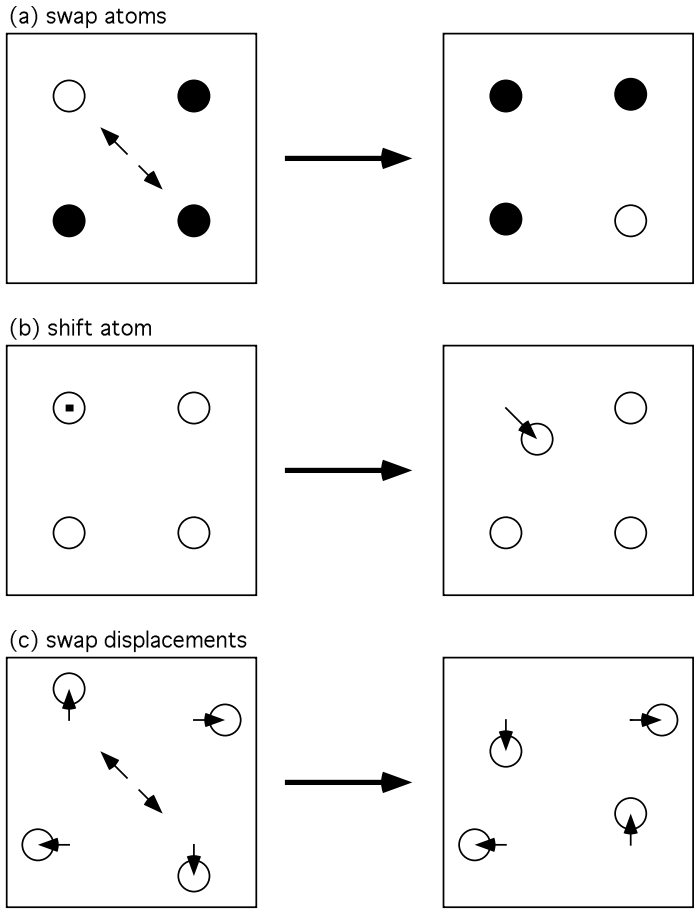

Figure 11.1: RMC operation modes of *DISCUS*

A short description of the different RMC operation modes is given in the list below. For details check the online help, command reference or refer to the RMC tutorial example.

1. *switch chemistry*
   This mode (Fig. 11.1a) 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. 11.1b) 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. 11.1c) 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'.

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.f'*. For more details about the construction of such a subroutine, read the commented example in the file *'extrmc.f'* 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 realised by creating the wanted different orientations of the molecules as different types using the symmetry segment of *DISCUS* (see chapter 5.2) 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.

11.5 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 (see section 4.4). 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 actually 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 number 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.

It should be noted, that *DISCUS* may require a large amount of memory to run an RMC simulation especially using the 'lot' option, since the structure factor for each data point for each lot needs to be stored in memory. Typically several hundred lots are used. The array sizes are set in the file *config.inc*. Detailed information about all commands in the RMC section can be found in the online help or the command reference. The RMC section of the interactive tutorial might also give a further insight how to use *DISCUS* for RMC simulations.

Appendix A

List of commands

In this appendix, a list of all current *DISCUS* commands is given. Note that commands marked with an asterix '*' branch to a sublevel of *DISCUS* where further commands not included in these lists are used. Detailed descriptions of **all** commands can be found in the command reference manual and the online help of the program.

A.1 Alphabetical list of commands

Command		**Description**
#		Comment, the rest of the line will be ignored
@		Execution of a macrofile
=		Assigns the value of an expression to a variable
addfile		Adds the contents of two files
append		Appends an atom or molecule, if location is not occupied
asym		Shows the content of the asymmetric unit
break		Interrupts a loop or conditional statement
chem	*	Switches to the chemistry level of *DISCUS*
continue		Continues *DISCUS* after 'stop' command
copy		Copies an atom
d2r		Converts a vector from real to reciprocal space
do		Start of a do loop
diff	*	Switches to the Difference Fourier level of *DISCUS*
echo		Echoes a string
else		Default block in an if construction
elseif		Alternative block in an if construction
enddo		End of a do loop
endif		End of an if construction
eval		Evaluates an expression for interactive display
exit		Ends the *DISCUS* program
find		Finds the environment around an atom
fourier	*	Switches to the Fourier level of *DISCUS*
help		Gives on line help

if		Begin of an if construction
ins		Inserts an atom
inverse	*	Switches to the Inverse Fourier Level of *DISCUS*
kick		Deletes atom/molecule and inserts a new one
learn		Starts a learn sequence
lend		Ends a learn sequence
m[j]=<exp>		Change scattering type of atom *j* to value of expression
mc	*	Switches to the Monte Carlo simulation level
micro	*	Switches to the microdomain level of *DISCUS*
output	*	Switches to the output level of *DISCUS*
patterson	*	Switches to the Patterson level of *DISCUS*
pdf	*	Switches to the PDF level of *DISCUS*
plot	*	Writes the structure in a format ready for display
powder	*	Switches to the powder diffraction level of *DISCUS*
proj		Project a vector onto another or onto a plane
purge		Deletes empty lines from the crystal
r2d		Converts a vector from reciprocal to real space
read	*	Switches to the reading level of *DISCUS*
remove		Deletes an atom from the crystal
replace		Replaces atom/molecule with given probability with a new type
rmc	*	Switches to the Reverse Monte Carlo level of *DISCUS*
save		Saves the current structure to file
seed		Initialize the random number generator
set		Set various parameters
show		Displays various *DISCUS* settings and results
stack	*	Switches to Stacking fault sublevel of *DISCUS*
stop		Stops *DISCUS* macro, resume with 'cont'
switch		Switches the position of two atoms or molecules
symm	*	Switches to symmetry calculation sublevel of *DISCUS*
system		Executes a shell command
thermal		Introducing displacements according to the temperature factor
trans	*	Switches to the unit cell transformation segment
vprod		Calculates the vector product
wait		Wait for user input
wave	*	Switches to the wave generating level of *DISCUS*
x[j]=<exp>		Change x position of atom *j* to value of expression
y[j]=<exp>		Change y position of atom *j* to value of expression
z[j]=<exp>		Change z position of atom *j* to value of expression

A.2 Functional list of commands

Program control

Command **Description**

@	Execution of a macrofile
=	Assigns the value of an expression to a variable
break	Interrupts a loop or conditional statement
continue	Continues *DISCUS* after 'stop' command
do	Start of a do loop
else	Default block in an if construction
elseif	Alternative block in an if construction
enddo	End of a do loop
endif	End of an if construction
exit	Ends the *DISCUS* program
if	Begin of an if construction
learn	Starts a learn sequence
lend	Ends a learn sequence
seed	Initialize the random number generator
set	Set various parameters
stop	Stops *DISCUS* macro, resume with 'cont'
system	Executes a shell command
wait	Wait for user input

Crystallographic calculations

Command	**Description**
d2r	Converts a vector from real to reciprocal space
proj	Project a vector onto another or onto a plane
r2d	Converts a vector from reciprocal to real space
vprod	Calculates the vector product

Modification of individual atoms

Command	**Description**
append	Appends an atom or molecule, if location is not occupied
copy	Copies an atom
ins	Inserts an atom
kick	Deletes atom/molecule and inserts a new one
remove	Deletes an atom from the crystal
switch	Switches the position of two atoms or molecules
m[j]=<exp>	Change scattering type of atom *j* to value of expression
x[j]=<exp>	Change x position of atom *j* to value of expression
y[j]=<exp>	Change y position of atom *j* to value of expression
z[j]=<exp>	Change z position of atom *j* to value of expression

Structure modifications at large

Command		Description
mc	*	Switches to the Monte Carlo simulation level
micro	*	Switches to the microdomain level of *DISCUS*
purge		Deletes empty lines from the crystal
replace		Replaces atom/molecule with given probability with a new type
stack	*	Switches to Stacking fault sublevel of *DISCUS*
symm	*	Switches to symmetry calculation sublevel of *DISCUS*
thermal		Introducing displacements according to the temperature factor
trans	*	Switches to the unit cell transformation segment
wave	*	Switches to the wave generating level of *DISCUS*

Fourier levels

Command		Description
diff	*	Switches to the Difference Fourier level of *DISCUS*
fourier	*	Switches to the Fourier level of *DISCUS*
inverse	*	Switches to the Inverse Fourier Level of *DISCUS*
patterson	*	Switches to the Patterson level of *DISCUS*
powder	*	Switches to the powder diffraction level of *DISCUS*

Input and Output

Command		Description
addfile		Adds the contents of two files
output	*	Switches to the output level of *DISCUS*
plot	*	Writes the structure in a format ready for display
read	*	Switches to the reading level of *DISCUS*
save		Saves the current structure to file

Information and crystal analysis

Command		Description
#		Comment, the rest of the line will be ignored
asym		Shows the content of the asymmetric unit
chem	*	Switches to the chemistry level of *DISCUS*
echo		Echoes a string
eval		Evaluates an expression for interactive display
eval n[1]		Displays the number of atoms within the crystal
eval n[2]		Displays the number of different atoms within the crystal
eval n[3]		Displays the number of atoms in (original) unit cell
eval n[4]		Displays the number of molecules within the crystal

eval n[5]		Displays the number of different molecule types within the the crystal
eval n[6]		Displays the number of molecules in (original) unit cell
eval m[<i>]		Displays atom type of atom <i>
eval x[<i>]		Displays x-coordinate of atom <i>, similar for y and z
find		Finds the environment around an atom
help		Gives on line help
rmc	*	Switches to the Reverse Monte Carlo level of *DISCUS*
pdf	*	Switches to the PDF level of *DISCUS*
set		Set various parameters
show		Displays various *DISCUS* settings and results

Appendix B

Installation

This appendix contains a short introduction how to obtain, compile and customize the program *DISCUS*. *DISCUS* is part of the diffuse scattering software package which consists of the three programs *DISCUS* itself, the plot program *KUPLOT* and the new program *PDFFIT*. However, *DISCUS* is also available in a stand-alone distribution. We assume that you install the software on a UNIX type platform. In order to install the programs a FORTRAN and C compiler are required. The GNU compilers `gcc` and `g77` used in the development of the programs are available for various platforms and can be downloaded from the internet free of charge. However, the source code might be successfully compiled on other platforms. Thanks to a 'gcc cross-compiler', MSDOS binaries of *DISCUS* and *DISCUS* can be obtained from the authors. Contact Thomas Proffen (Email: *proffen@pa.msu.edu*) for details.

Obtaining and unpacking DISCUS

First one needs to obtain the complete 'diffuse' archive named `diffuse-xx.xx.xx.tar.gz` or just the *DISCUS* archive called `discus-3.2.tar.gz` from the internet. Note that `xx.xx.xx` in the file name stands for the creation date of the complete archive. For the remainder of this section we assume you obtained the complete archive and you need to unpack it using the commands

```
gzip -d diffuse-xx.xx.xx.tar.gz
tar -xvof diffuse-xx.xx.xx.tar
```

This will create a directory 'diffuse' containing the distribution. Within this directory there are separate directories `discus`, `kuplot` and `discus` containing the three different programs as well as a directory `lib_f77` which contains command language related routines common to all three programs. In each program directory, you will find the following directory structure:

```
./prog     : Contains the source code
./doc      : Contains POSTSCRIPT version of the documentation
./tutorial : Contains the tutorial for the program
```

Customizing and installing DISCUS

After proceeding to the `discus/prog` directory you need to choose a `Makefile` that suits your needs and customize the `Makefile` as well as one configuration file '`config.inc`'. There are four different Makefiles for UNIX platforms depending on the FORTRAN compiler you are using.

```
cp Makefile.f77 Makefile    (f77 compiler version)
cp makefile.f2c Makefile    (f2c compiler version)
cp makefile.g77 Makefile    (g77 compiler version)
cp makefile.f90 Makefile    (f90 compiler version)
```

Next edit the Makefile and alter the location where the program(s) shall be installed defined by the variable `BINDIR`. Read the comments in the Makefile to select the appropriate compiler switches for your platform. Finally edit '`config.inc`' and adjust array sizes to suit your needs. An explanation of the variables in are found in the header of the file '`config.inc`'. Keep the memory size of your computer in mind when adjusting those array sizes ! Next the program is compiled and linked by executing the command `make` followed by `make install` if all went well. Make sure an appropriate path for the binaries to be installed is set in the Makefile. After that you can perform a `make clean` to remove the binary and the object files from the source directory. If you want to install the program manually you have to put the files '`discus`' and '`discus.hlp`' in the same directory.

Before you actually can use the online help of the program, an environment variable `DISCUS` has to be set to the path where the program is installed in. This can be done e.g. in the `.login` or `.cshrc` file using the command `setenv DISCUS /path/to/discus` for the `csh` of `set DISCUS=/path/to/discus; export DISCUS` is you are using the bourne shell. If this path is also included in your search path you can start the program simply by entering `discus`.

The progam *DISCUS* can also be installed on a DEC VMS machine. Tools like `gzip` and `tar` are freely available from the internet. The file `Makefile.vax` is a compilation script that is part of the distribution.

Please register

Please register yourself as a user of one or more programs of the *DISCUS* program package. In the top directory of the distribution you will find the file `REGISTER`. Please fill in the corresponding information and send the file via email to *proffen@pa.msu.edu.* Thank you for registering, a feedback is always a strong motivation to proceed with the program development and making it available to the public. By registering you also enable us to inform you of program updates, workshops and other related topics.

Bibliography

[1] S.J.L. Billinge and T. Egami. Short-Range Atomic Structure of $Nd_{2-x}Ce_xCuO_{4-y}$ Determined by Real-Space Refinement of Neutron-Powder-Diffraction Data. *Phys. Rev. B*, 47:14386–14406, 1993.

[2] K. Binder, editor. *The Monte Carlo Method in Condensed Matter Physics* , volume 71 of *Topics in Applied Physics* . Springer, 1 edition, 1995.

[3] B.D. Butler and T.R. Welberry. Calculation of Diffuse Scattering from Simulated Disordered Crystals: a Comparison with Optical Transforms . *J. Appl. Cryst.*, 25:391–399, 1992.

[4] B.D. Butler, R.L. Withers, and T.R. Welberry. Diffuse Absences due to the Atomic Size Effect . *Acta Cryst. A*, 48:737–746, 1992.

[5] J.M. Cowley. *Diffraction Physics* . Elsevier Science B.V., 3 edition, 1995.

[6] T. Egami. PDF Analysis applied to Crystalline Materials. In S. J. L. Billinge and M. F. Thorpe, editors, *Local Structure from Diffraction*, page 1, New York, 1998. Plenum.

[7] F. Frey. Diffuse Scattering from Disordered Crystals . *Acta Cryst. B*, 51:592–602, 1995.

[8] F. Frey. Diffuse Scattering from Periodic and Aperiodic Crystals . *Z. Kristallogr.*, 212:257–282, 1997.

[9] H. Jagodzinski. Diffuse X-ray Scattering from Crystals . *Prog. Cryst. Growth*, 14:47–102, 1987.

[10] H. Jagodzinski and F. Frey. *Disorder Diffuse Scattering of X-rays and Neutrons* , international tables of crystallography b 4.2, pages 392–432. IUCr, 1993.

[11] R.L. McGreevy and L. Pusztai. Reverse Monte Carlo Simulation: a New Technique for the Determination of Disordered Structures . *Mol. Simul.*, 1:359–367, 1988.

[12] N. Metropolis, A.W. Rosenbluth, M.N Rosenbluth, A.H Teller, and E.J. Teller. Equation of State Calculations by Fast Computing Machines . *J. Chem. Phys.*, 21:1087–1094, 1953.

[13] R.B. Neder, F. Frey, and H. Schulz. Defect Structure of Zirconia ($Zr_{0.85}Ca_{0.15}O_{1.85}$) at 290 and 1550K . *Acta Cryst. A*, 46:799–809, 1990.

[14] R.B. Neder, F. Frey, and H. Schulz. Diffraction Theory for Diffuse Scattering by Correlated Microdomains in Materials with Several Atoms per Unit Cell . *Acta Cryst. A*, 46:792–798, 1990.

[15] R.B. Neder and Th. Proffen. Teaching diffraction with the aid of computer simulations. *J. Appl. Cryst.*, 29:727–735, 1996.

[16] V.M. Nield, D.A. Keen, and R.L. McGreevy. The Interpretation of Single Crystal Diffuse Scattering using Reverse Monte Carlo Modelling . *Acta Cryst. A*, 51:763–771, 1995.

[17] Th. Proffen and S.J.L. Billinge. *PDFFIT*, a Program for Full Profile Structural Refinement of the Atomic Pair Distribution Function. *J. Appl. Cryst.*, 1999. in press.

[18] Th. Proffen and R.B. Neder. DISCUS, a Program for Diffuse Scattering and Defect Structure Simulations . *J. Appl. Cryst.*, 30:171–175, 1997.

[19] Th. Proffen, R.B. Neder, and F. Frey. Neutron and X-ray Diffuse Scattering of Calcium Stabilized Zirconia. *Acta Cryst. B*, 52:59–65, 1996.

[20] Th. Proffen, R.B. Neder, F. Frey, and W. Assmus. Defect Structure and Diffuse Scattering of Zirconia Single Crystals doped with 7 mol% CaO. *Acta Cryst. B*, 49:599–604, 1993.

[21] Th. Proffen and T.R. Welberry. An Improved Method for Analysing Single Crystal Diffuse Scattering using the Reverse Monte Carlo Technique . *Z. Kristallogr.*, 212:764–768, 1997.

[22] Th. Proffen and T.R. Welberry. Analysis of Diffuse Scattering *via* Reverse Monte Carlo Technique: a Systematic Investigation . *Acta Cryst. A*, 53:202–216, 1997.

[23] Th. Proffen and T.R. Welberry. FOURDEM: A Demonstration Program for Fourier Synthesis and Other Crystallographic Concepts . *J. Appl. Cryst.*, 30:567, 1997.

[24] H.M. Rietveld. A profile refinement method for nuclear and magnetic structures. *J. Appl. Cryst.*, 2:65–71, 1969.

[25] D.E. Sands. *Vectors and Tensors in Crystallography* . Dover Publications Inc., 1 edition, 1995.

[26] B.H. Toby and T. Egami. Accuracy of Pair Distribution Function Analysis Applied to Crystalline and Non-Crystalline Materials. *Acta Cryst. A*, 48:336–346, 1992.

[27] B.E. Warren. *X-ray Diffraction* . Dover Publications Inc., New York, 1990.

[28] T.R. Welberry. Diffuse X-ray Scattering and Models of Disorder . *Rep. Prog. Phys.*, 48:1543–1593, 1985.

[29] T.R. Welberry. Multi-Site Correlations and the Atomic Size Effect . *J. Appl. Cryst.*, 19:382–389, 1986.

[30] T.R. Welberry and B.D. Butler. Interpretation of Diffuse X-ray Scattering *via* Models of Disorder . *J. Appl. Cryst.*, 27:205–231, 1994.

[31] T.R. Welberry and B.D. Butler. Diffuse X-ray Scattering from Disordered Crystals . *Chem. Rev.*, 95:2369–2403, 1995.

[32] T.R. Welberry and Th. Proffen. Analysis of Diffuse Scattering from Single Crystals *via* Reverse Monte Carlo: I. Comparison with Direct Monte Carlo . *J. Appl. Cryst.*, 1997. in press.

[33] A.J.C. Wilson, U. Shmuelli, and T. Hahn. *International Tables for Crystallography* . Dordrecht, Holland, 1 edition, 1983.