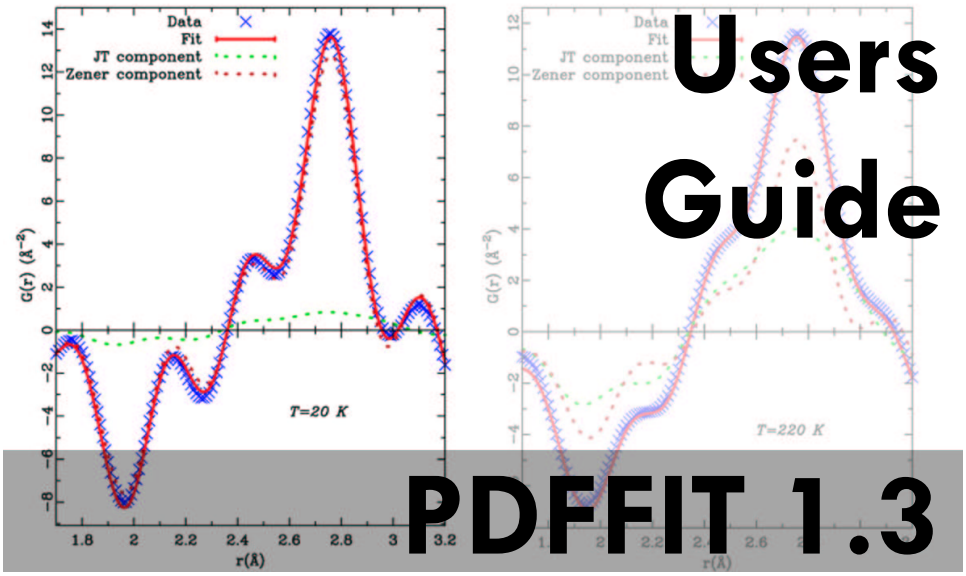

Users Guide

PDFFIT 1.3

written by

Thomas Proffen
Email: tproffen@lanl.gov

Simon Billinge
Email: billinge@pa.msu.edu

Document created: July 15, 2003

Preface

Disclaimer

The *PDFFIT* 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 *PDFFIT*. 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 *PDFFIT* program may be changed without notice.

PDFFIT is intended as a public domain program. It may be used free of charge. Any commercial use is, however, not allowed without permission of the authors.

Acknowledgments

The basic concept of the full profile refinement of the pair distribution function (PDF) was adapted from the program *RESPAR* written by Simon Billinge (Billinge, 1998). The routines for the processing of the command language and the FORTRAN style interpreter are taken from the program package *DISCUS* by Reinhard Neder and Thomas Proffen (Proffen and Neder, 1997). *PDFFIT* is distributed as part of that package as well as a stand-alone program. The routines for the line editing were taken from the program GNUPLOT.

Using PDFFIT

Publications of results totally or partially obtained using the program *PDFFIT* should state that *PDFFIT* was used and contain the following reference:

> PROFFEN, TH. & BILLINGE, S.J.L. (1999) "PDFFIT, a Program for Full Profile Structural Refinement of the Atomic Pair Distribution Function" *J. Appl. Cryst.*, **32**, 572-575

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is PDFFIT ?

Now you might have installed *PDFFIT* and can start it simply by typing `pdffit` but in case you are not a PDF wizard here is a little introduction into the PDF method first. For those who hate to read long manuals, there is section 2 which shows how to run a refinement in no time. The remaining users guide explains the various features of *PDFFIT* more detail.

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. Only one-body information such as atomic positions, site occupancies and temperature factors can be extracted. Determination of the *average* structure based on powder diffraction data is now routinely done using the Rietveld (Rietveld, 1969) method which is very similar to the full profile refinement of the atomic pair distribution function (PDF) as we will see later. It should be kept in mind that the analysis of Bragg scattering assumes a prefect long range periodicity of the crystal. However, many materials are quite disordered and even more important the key to the deeper understanding of their properties is the study of deviations from the *average* structure or the study of the *local* atomic arrangements. Deviations from the *average* structure result in the occurrence of diffuse scattering which contains information about two-body interactions (Welberry and Butler, 1995; Frey, 1995). In recent years the analysis of diffuse scattering from single crystals as well as powders using computer simulations have made great advances, in particular using the Monte Carlo (MC) and the Reverse Monte Carlo (RMC) technique (Welberry and Butler, 1995; Nield et al., 1995; Welberry and Proffen, 1998; Proffen and Welberry, 1998, 1997).

Another 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 (Egami, 1998; Billinge and Egami, 1993). The PDF is obtained from the powder diffraction data via a simple Fourier transform of the normalized scattering intensity $S(Q)$:

$$G(r) = 4\pi r[\rho(r) - \rho_0] = \frac{2}{\pi}\int_0^\infty Q[S(Q) - 1]\sin(Qr)dQ, \tag{1.1}$$

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 Egami (1998); Warren (1990) 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 as we will see a little later. In order to carry out the Fourier transform in (1.1) 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. A short discussion of these *termination ripples* is given in appendix A or for a more detailed discussion of the accuracy of PDF analysis see Toby and Egami (1992). With the availability of modern synchrotron and neutron sources it is possible to collect powder diffraction data up to high values in Q.

The study of a measured PDF ranges from a simple peak width analysis revealing information about correlated motion (Jeong et al., 1998) to the full profile refinement of the PDF using e.g. *PDFFIT* the manual of which you are currently reading. In order to refine an experimental PDF one needs to calculate a PDF from a structural model. This can be done using the relation

$$G_{calc}(r) = \frac{1}{r}\sum_i\sum_j\left[\frac{b_i b_j}{\langle b\rangle^2}\delta(r - r_{ij})\right] - 4\pi r\rho_0, \tag{1.2}$$

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 (set by the command `xray`). 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 the program *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 (1.2) with a Gaussian accounting for the displacements (for details see appendix A).

Now we are ready for the first *PDFFIT* refinement example presented in the next section. A detailed discussion about the structural parameters that can be obtained and how they relate to their 'long range' counterparts is given in section 6.

1.2 What is new ?

If you have used *PDFFIT* before, you might ask what is new in this version of the program. Below you find sections describing the major changes in the different releases of *PDFFIT*. A detailed list of all changes can be found in the file '*CHANGES.LOG*' in the source directory of the *PDFFIT* distribution.

Version 1.3

The main addition in this version are additional parameters to control the PDF peak width. In addition to the quadratic term variable, δ, a linear correction has been added (γ). In addition a PDF broadening correction determined by the instrument resolution has been added (α). For details on the new r dependence of the PDF peak width refer to section 6.1.2.

Following the work by Thorpe et al. (2002), the true PDF peak profile is not purely a Gaussian. The corrected profile has now been implemented in PDFFIT. Details can be found in Appendix A.

Version 1.2

Besides various bug fixes, this new version of *PDFFIT* is now distributed as self-extracting installer for the Windows operating system. The program now supports system macros that can be globally installed as well as reading macro files names from the command line.

Version 1.1

The following features were added to the program: *PDFFIT* is now capable to export a structure in a format that can be read by the program *ATOMS* (Dowty, 1997) for 3D structural plots (see section 3.3). The program now allows to refine the difference between a PDF and a reference PDF. The command language now includes simple commands for file input and output (see section 5.9).

1.3 Getting started

After the program *PDFFIT* 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 *PDFFIT* is given in appendix C.

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 *PDFFIT*. It could, for example be the actual name of a file, or a numerical value.
'text'	Text in single quotes exclusively refers to *PDFFIT* 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 *PDFFIT* 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. *PDFFIT* is case sensitive, all commands and alphabetic parameters MUST be typed in lower case letters. If *PDFFIT* 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 *PDFFIT* are listed in Table 1.1.

1.4 More help

There are several sources of information, first *PDFFIT* 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 'pdf_cmd.ps' in the directory 'doc' of the distribution. The manual you are reading describes scientific background and principle functions of *PDFFIT* 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 *PDFFIT* and enter '@tutorial' to begin the tour.

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

http://www.totalscattering.org/discus/pdffit.html
http://www.uni-wuerzburg.de/mineralogie/crystal/discus/pdffit.html

Chapter 2

Quick start

In order to quickly illustrated the usage of *PDFFIT* for those impatient readers, a PDF refinement of Ni will be illustrated. The data used here were collected at room temperature using the diffractometer at beamline X7A at NSLS, Brookhaven National Laboratory. The data are terminated at $Q_{max} = 22\text{Å}^{-1}$. The data as well as the resulting PDF are shown as filled circles in Figure 2.1.

The sequence of commands used to carry out the refinement is listed below. Note that the line numbers are only added for easy reference and are not actually part of the input. The commands can either be written

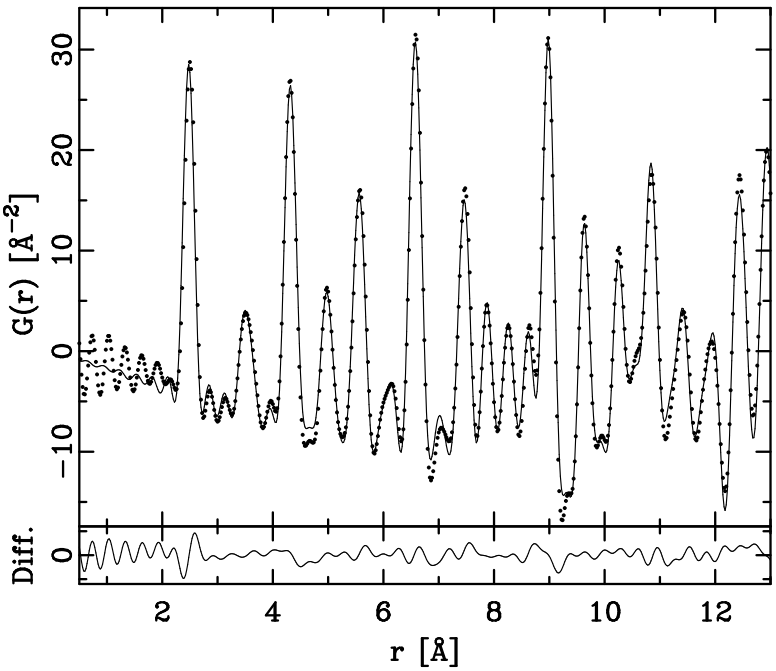

Figure 2.1: Result of PDF refinement of Ni. The solid line is the calculated PDF, the filled circles are the data. A difference curve is plotted below the data.

using a text editor and executed as macro file of typed in at the *PDFFIT* command prompt.

```
 1 reset
 2 read stru,ni.stru
 3 read data,x,22.0,0.0,ni.data
 4 #
 5 par lat[1]=p[1],1.0
 6 par lat[2]=p[1],1.0
 7 par lat[3]=p[1],1.0
 8 #
 9 do i[1]=1,n[1]
10   par u[1,i[1]]=p[2],1.0
11   par u[2,i[1]]=p[2],1.0
12   par u[3,i[1]]=p[2],1.0
13 enddo
14 #
15 par csca[1]=p[10],1.0
16 par qsig[1]=p[11],1.0
17 par delt[1]=p[12],1.0
18 #
19 p[1]=lat[1]
20 p[2]=u[1,1]
21 #
22 p[10]=1.50
23 p[11]=0.03
24 p[12]=0.10
25 #
26 range 1,0.5,13.0
27 run
28 #
29 save pdf,1,ni_ref.calc
30 save dif,1,ni_ref.dif
31 save stru,1,ni_ref.stru
32 save res,ni_ref.out
```

The command `reset` in line 1 resets *PDFFIT* to have a defined initial state. Next a structural model (line 2) and the experimental PDF (line 3) are read. The format of these text files are discussed in section 3 of this manual. The parameters of the command `read data` are the radiation type (here X-ray), Q_{max} and the resolution factor σ_Q associated with the data file '*ni.data*'. *PDFFIT* allows the user to freely define relations between structural and experimental parameters and the refinement parameters `p[i]`. This is done using the command `par`. The lattice parameters a, b, c are stored in *PDFFIT* variables `lat[1], lat[2]` and `lat[3]`. In lines 5–7 of the example above refinement parameters `p[1]` is used to refine all three lattice parameters since Ni is cubic and $a = b = c$. The additional parameter `1.0` is the derivative of the parameter on the left with respect to the refinement parameters. These derivatives are needed for the least-squares refinement (see Appendix A for details). In our case we have a simple one to one correspondence and the derivative is trivial. In lines 9–13 the temperature factors U_{lm} of the atoms are associated with a

single refinement parameter `p[2]`. Note that we have four Ni sites in our model and rather than typing the definitions for each atom separately, we are using a loop. Details about loops and variables are given in section 5, all we need to know for now is that the variable `n[1]` contains the number of sites. In our example we only refine an isotropic temperature factor, so we use the same parameter for U_{11}, U_{22} and U_{33} stored in `u[1,n], u[2,n]` and `u[3,n]`. Next we associate individual refinement parameters with the scale factor f_p (line 15), the resolution factor σ_Q (line 16) and the dynamic correlation factor δ (line 17). Note that there is no need for a consecutive numbering of the parameters. Finally we need to initialize the refinement parameters `p[i]` with suitable starting values which is done in lines 19–24. Last we define the r-range we want to use for the refinement (line 26), here from 0.5Å to 13.0Å. The argument of the `range` command is the number of the dataset (see section 6.1.3). There is generally no predefined order for the commands, however, data need to be read first and obviously all desired settings must be entered before the refinement is started using the command `run` (line 27). The screen output during the refinement is shown below:

```
Starting refinement ...
------------------------------------------------------------------------
Iteration  :   0  Sum:  48995.5     Urf:  1.00798
                  R  : 0.155190     Rw : 0.154659
Parameters :
  1:  3.52032       2: 0.515471E-02  10:  1.50000      11: 0.300000E-01
 12: 0.100000
------------------------------------------------------------------------
Iteration  :   1  Sum:  44646.7     Urf:  1.00165
                  R  : 0.148137     Rw : 0.147636     DRw: -.702314E-02
Parameters :
  1:  3.51974       2: 0.505564E-02  10:  1.48549      11: 0.313635E-01
 12: 0.127834
------------------------------------------------------------------------
Iteration  :   2  Sum:  44352.0     Urf:  1.00089
                  R  : 0.147658     Rw : 0.147148     DRw: -.488088E-03
Parameters :
  1:  3.52014       2: 0.501998E-02  10:  1.48752      11: 0.319241E-01
 12: 0.131231
------------------------------------------------------------------------
Iteration  :   3  Sum:  49266.2     Urf:  1.00054
                  R  : 0.155582     Rw : 0.155086     DRw: 0.793780E-02
Parameters :
  1:  3.52056       2: 0.501416E-02  10:  1.48125      11: 0.286770E-01
 12: 0.135971
------------------------------------------------------------------------
```

For each iteration the program prints $\sum(G_{obs} - G_{calc})^2$ (`sum`), the R-value (`R`), the weighted R-value (`Rw`) and the difference of the R-value compared to the previous cycle (`DRw`). Furthermore the actual values of all refinement parameters `p[i]` are listed. Note that iteration 3 has a slightly higher R-value compared to iteration 2, so the refinement converged or exploded depending on the size of the (now positive) difference.

The resulting parameters correspond actually to the output of the $(n-1)$th cycle. However, the display of the iteration that lead to the termination of the refinement might help to judge how well the fit converged.

In lines 29–32 we have to save the resulting PDF, the difference between observed and calculated PDF, the resulting structure and fit results, respectively. The calculated PDF can be plotted using e.g. *KUPLOT* which is part of the *DISCUS* software package. Figure 2.1 was generated using *KUPLOT*. The final structure can also be exported in *DISCUS* format for further analysis.

After this quick run through a quite simple example refinement we will discuss all features of the program *PDFFIT* in more detail. Additionally the command reference and the online help of the program might be a valuable source of information. Use the command `help` when in *PDFFIT* to activate the online help facility of the program.

Chapter 3

File formats

3.1 PDF data files

The file format for observed as well as calculated PDFs used by *PDFFIT* are simple text files of the following general format per data point:

$r \cdot G(r) \cdot \sigma(r) \cdot \sigma(G(r))$

Each data point is on a separate line in the file. The first column contains the value of r given in units of Å. The second column is $G(r) = 4\pi r[\rho(r) - \rho_0]$ in units of Å^{-2}. The third column is ignored by *PDFFIT* but is required to be compatible with the plot program *KUPLOT* where it takes the value of the standard deviation of r. The last column contains the standard deviation of $G(r)$ which is used to calculate the weight $w(r)$ according to $w(r) = 1/\sigma^2(G(r))$. A short listing of a PDF file is shown below:

```
 0.020000    11.687810      0.0       0.081738
 0.040000    22.659353      0.0       0.155914
 0.060000    31.915903      0.0       0.212270
    :
19.960000   -14.941482      0.0       0.162837
19.980000   -14.858703      0.0       0.163509
20.000000   -14.135623      0.0       0.164080
```

As one can see, column 3 is just occupied by some dummy number. However, the current version of *PDFFIT* will also read two-column files ($rG(r)$) and three-column files ($rG(r)w(r)$). In order to save disk space, archived files might be compressed using e.g. `gzip`. The results can be plotted using *KUPLOT* or any other visualization software that can import ASCII files.

3.2 Structure files

The program *PDFFIT* uses a keyword driven structure file format similar to the one used by *DISCUS*. However, the philosophy of *DISCUS* is based on simulating large disordered structures. Thus *DISCUS*

does not support site occupancies and includes only an isotropic temperature factor. Furthermore, *PDFFIT* calculates standard deviations for its structural parameters which need to be saved as well. We will have a look at the Ni structure files used in the last section. The *DISCUS* version of this file looks like this:

```
title Ni
spcgr P1
cell  3.520323,    3.520323,    3.520323,   90.0,   90.0,   90.0
ncell 1,1,1, 4
atoms
NI      .000000   .000000   .000000     0.4070
NI      .000000   .500000   .500000     0.4070
NI      .500000   .000000   .500000     0.4070
NI      .500000   .500000   .000000     0.4070
```

The file contains a title, the space group and lattice parameters in the first three lines. The lattice parameters are again given in units of Å. Some explanation needs to be given concerning the space group. The program *PDFFIT* ignores the space group and just stores it to write it back in the output file. Thus no symmetrically equivalent atom positions are created using the space group symbol and there are no symmetry restrictions during the refinement unless coded in the refinement parameter definitions. A convenient way to handle systems with many symmetrically equivalent positions is to create a similar file with the correct space group symbol and read it with *DISCUS* as a `cell` file. This will generate all required positions and when saving it as a `stru` file, all positions are saved and can be read by *PDFFIT*. The `ncell` keyword in the input file above specifies the number of unit cells in x, y and z-direction and the number of atoms within one unit cell. This way it is possible to construct a model using a supercell based on several basic unit cells without having to change the metric. The last keyword needs to be `atoms` followed by a list of atoms. Each line contains the atom name, the position in fractional coordinates and the isotropic temperature factor B (see below). When using more than one unit cell, the fractional coordinates range from $-N/2$ to $N/2$ for N unit cells. The atom list must follow a specific order outlined in the *DISCUS* manual.

```
title  Ni
format pdffit
scale  1.4875
spcgr  P1
cell   3.520144,  3.520144,  3.520144, 90.000000, 90.000000, 90.000000
dcell  0.000031,  0.000031,  0.000031,  0.000000,  0.000000,  0.000000
ncell  1,1,1, 4
atoms
NI      0.00000000        0.00000000        0.00000000       1.0000
        0.00000000        0.00000000        0.00000000       0.0000
        0.00501998        0.00501998        0.00501998
        0.00000515        0.00000515        0.00000515
        0.00000000        0.00000000        0.00000000
        0.00000000        0.00000000        0.00000000
```

This is the corresponding file in *PDFFIT* format. Only one Ni atom is shown for brevity. We note three new keywords, `format` specifies that this is a *PDFFIT* file which can not be read by *DISCUS*. The key-

word `scale` gives the scale factor used for the structural phase defined in this file and `dcell` defines the standard deviations of the lattice parameters which might be needed to calculate bond-length and angles with propagated errors. For each atom, the following information is given: atom name, position x, y, z, occupancy o, standard deviations $\sigma(x), \sigma(y), \sigma(z)$ and $\sigma(o)$ followed by the three diagonal elements of the anisotropic temperature factor U_{11}, U_{22}, U_{33} with errors and the three elements U_{12}, U_{13}, U_{23} with errors. *PDFFIT* recognizes both types of structure file automatically and converts the B value to U_{ii} according to $U_{11} = U_{22} = U_{33} = /8\pi^2$ and $U_{12} = U_{13} = U_{23} = 0$ (Sands, 1995).

When saving structure files one can choose between *PDFFIT* format (`save pdf,..`) and *DISCUS* format (`save disc,..`). When using the *DISCUS* format, information about site occupancies and errors are lost and the B value is calculated as $B = 8\pi^2[U_{11} + U_{22} + U_{33}]/3$. Obviously any anisotropy is lost. However, *DISCUS* can be used to export the structure for plotting, doing further analysis or calculating single crystal diffuse scattering from the refined model. We will give a short example of how to use *DISCUS* to expand the content of unit cell using the space group symbol. The input file for *DISCUS* for $GaAs$ would look like this:

```
title GaAs
spcgr F-43m
cell 5.6537  5.6537  5.6537  90.0  90.0  90.0
atoms
GA    0.0000  0.0000  0.0000  1.0000
AS    0.2500  0.2500  0.2500  1.0000
```

The following sequence of *DISCUS* commands can be used to read the unit cell file, expand it to a structure of 1x1x1 unit cells and save the result to the file *'gaas.stru'*:

```
read
cell gaas.cll,1,1,1
save gaas.stru
```

The structure file saved by *DISCUS* contains all atom positions generated by the space group $Fm\bar{3}m$. The output file containing the expected eight atoms is listed below:

```
title GaAs
spcgr F-43m
cell    5.6537    5.6537    5.6537   90.0   90.0   90.0
ncell   1,1,1, 8
atoms
GA     0.0000    0.0000    0.0000      1.000
GA     0.0000    0.5000    0.5000      1.000
GA     0.5000    0.0000    0.5000      1.000
GA     0.5000    0.5000    0.0000      1.000
AS     0.2500    0.2500    0.2500      1.000
AS     0.2500    0.7500    0.7500      1.000
AS     0.7500    0.2500    0.7500      1.000
AS     0.7500    0.7500    0.2500      1.000
```

Note that this version of the file contains the `ncell` keyword. This example was quite simple since all the atoms are on special positions and only the F-centering is applied. However, when dealing with atoms on general positions in a high symmetry space group using *DISCUS* to generate all symmetrically equivalent positions is a real time saver.

3.3 Exporting structures

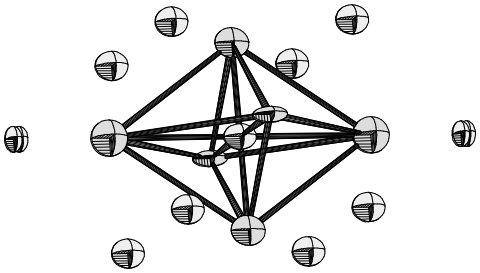

Figure 3.1: ATOMS structure plot displaying thermal ellipsoids.

PDFFIT can also export the structure in a format using the command `save atoms, 1, filename` (assuming phase number 1 to be saved). This file can be imported by the structure plotting program *ATOMS* (Dowty, 1997) using the menu *File* → *Import File* → *Free-Form (.INP)*. An example displaying thermal ellipsoids can be seen in Figure 3.1. The bonds were added in *ATOMS*. In order for *ATOMS* to display the complete exported structure, the lattice parameters and fractional coordinates are adjusted so that the complete crystal is a single unit cell. Additionally the space group is set to be $P1$.

Chapter 4

PDF calculation

4.1 Simple PDF calculation

Sometimes rather than refining a PDF one simply wants to calculate one from a structural model e.g. to explore the effect of a certain parameter on the calculated PDF. Rather than starting a refinement with no refinement parameters *PDFFIT* offers the simple command `calc` to compute the desired PDFs. As an example we want to calculate the PDF of Ni with two different termination values Q_{max}. The corresponding sequence of *PDFFIT* commands is shown below.

```
 1 reset
 2 read stru,ni.stru
 3 alloc n,25.0,0.0,1.5,6.5,251
 4 delt[1]=0.3
 5 #
 6 calc
 7 save pdf,1,cal_25.calc
 8 #
 9 qmax[1]=35.0
10 calc
11 save pdf,1,cal_35.calc
```

The first two command should be familiar already from the example in section 2. In line 3 we allocate the required space for the PDF we want to calculate without actually reading any experimental data. The first three parameters of the `alloc` command are similar to *read data* in the earlier example, they specify the radiation type and values for Q_{max} and the resolution factor σ_Q. The next two values specify the desired r-range which in our example is 1.5 to 6.5Å. The last parameter is the number of data points we want which in our example leads to a grid size of $\Delta r = (6.5 - 1.5)/(251 - 1) = 0.02$Å. In order to see a larger effect, we set the dynamic correlation factor δ to a rather large value in line 4 which gives us quite a sharp first PDF peak. Now the PDF can be calculated using the command `calc` (line 6) and the result is saved to the file '*cal_25.calc*' (line 7). In order to recalculate the same PDF with a different value of Q_{max}, we simply alter

the value using the corresponding variable `qmax[1]` (line 9). Note that the '1' specifies the number of the data set. Finally we calculate the PDF again (line 10) and save the result to a different file (line 11).

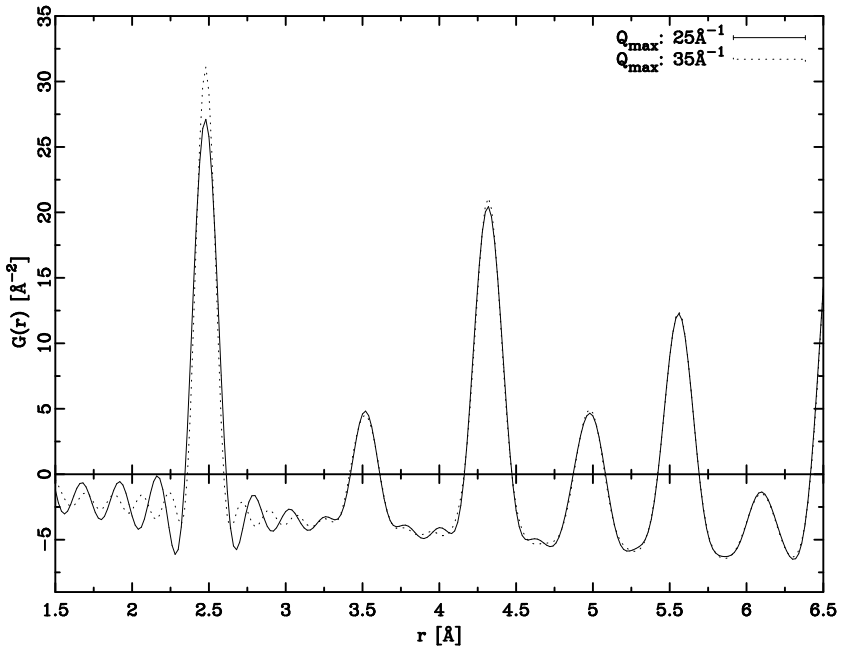

Figure 4.1: Calculated PDFs for Ni at different values Q_{max}. The solid line is the PDF with $Q_{max} = 25\text{Å}^{-1}$ and the dotted line is for $Q_{max} = 35\text{Å}^{-1}$.

The result of the example given above is shown in Figure 4.1. As one might expect, the termination ripples for the calculation with the higher Q_{max} show a higher frequency. In addition we can observe in our particular example that the intrinsic width of the first peak is smaller than the width of the convolution at $Q_{max} = 25\text{Å}^{-1}$, in other words the peak is broadened due to the Q termination.

4.2 Calculating partial or differential PDFs

When discussing equation (1.2) used to calculate the PDF from a structural model, we just stated that the sum over ij goes over *all* pairs of atoms i and j within the model crystal. This is perfectly correct to calculate a total PDF. However sometimes it might be desired to calculate just a partial or differential PDF. Let us consider $GaAs$ as a simple example. Figure 4.2 shows the total PDF as well as the differential and partial Ga PDF of $GaAs$. The total PDF is shown in the top view graph of Figure 4.2. Numbering the PDF peaks from the left, we find that the first peak corresponds to the shortest $Ga - As$ distance. The second peak is from $Ga - Ga$ and $As - As$ separated by the F-centering, the next peak is again from $Ga - As$, the fourth peak corresponds to the lattice repeat and so on. Let us consider just the first two peaks. A differential PDF contains all pairs of one specific atom, here Ga, and all other atoms. So the first $Ga - As$ peak is the same in the differential PDF (Fig. 4.2 middle) as in the total one. The second peak, however, contains contributions

of $Ga-Ga$ and $As-As$ in the total PDF, whereas the differential PDF shows no $As-As$ pairs. Since there is the same number of both types, the second peak in the differential PDF has only half the height compared to the total PDF. The partial Ga PDF (Fig. 4.2 bottom) contains only contributions of $Ga-Ga$ pairs. Thus the first peak is missing whereas the second peak is equivalent to the one observed in the differential PDF.

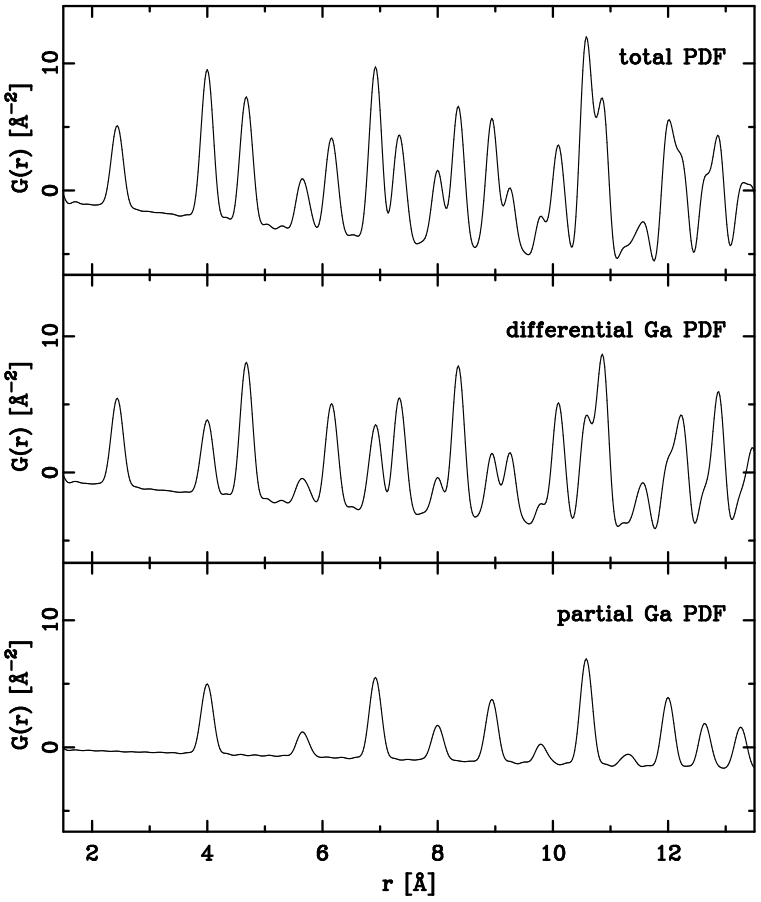

Figure 4.2: Top: Total $GaAs$ PDF. Middle: Differential Ga PDF of $GaAs$. Bottom: Partial Ga PDF of $GaAs$.

The determination which atoms contribute to which peaks in the PDF is not always as simple as in our $GaAs$ example and the calculation of various partial PDFs might help to understand a more complex PDF.

The macro that was used to calculate the PDFs shown in Figure 4.2 is listed below. Again the line number were added for convenience. The first three lines are similar to the example in the last section. The commands to select particular atom types are `isel` and `jsel`.

```
1 reset
2 read stru,gaas.stru
3 alloc x,25.0,0.0,1.5,13.5,601
4 #
5 isel 1,all
```

```
 6 jsel 1,all
 7 calc
 8 save pdf,1,tot.calc
 9 #
10 ides 1,all
11 jdes 1,all
12 isel 1,ga
13 jsel 1,all
14 calc
15 save pdf,1,dif_ga.calc
16 #
17 ides 1,all
18 jdes 1,all
19 isel 1,ga
20 jsel 1,ga
21 calc
22 save pdf,1,par_ga.calc
```

To calculate the total PDF, we simply select all atom types present in the structure (lines 5–6) and calculate and save the PDF (lines 7–8). Note that the selection of all atoms is the default when starting *PDFFIT* or using the command `reset`. The first parameter of the `i/jsel` and `i/jdes` commands is the corresponding data set number, here simply one. To calculate the differential PDF we select Ga for atom i (line 12) and all atoms for atom j (line 13). Since the command `isel` or `jsel` adds the specified atom to the list of allowed atoms, we need to deselect all previously selected atoms first as done in lines 10–11 of our example. Again we calculate and save the new PDF (lines 14–15). Finally we select just Ga for atoms i and j (lines 17–20) to obtain the partial Ga PDF and recalculate and save the result (lines 21-22).

Note, that *PDFFIT* can also refine a experimental differential PDF which for example can be obtained from anomalous X-ray scattering. The required commands to select the specific atom types are exactly the same as in this example. Note, that *PDFFIT* normalizes the number density ρ_0 by a weighting factor given below:

$$w(kl) = \frac{\sum_{kl} c_k c_l b_k b_l}{\sum_{ij} c_i c_j b_i b_j} \tag{4.1}$$

Here the sum over kl is only over those atom that are selected and contribute to the PDF whereas ij sums over all atoms within the crystal. It is immediately obvious that $w(kl) = 1.0$ for the total PDF. This definition is consistent with the use of Faber-Ziman partial structure factors. For details refer to Waseda (1986) and appendix A.

Chapter 5

FORTRAN style interpreter

The program includes a FORTRAN style interpreter that allows the user to program complex modifications. The interpreter provides variables, linked to data sets 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` and `if`. The command `eval` allows to examine the contents of a variable of evaluate an expression, e.g. `eval r[1]*0.5`.

5.1 Variables

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

Example : `i[1], r[0], i[i[1]], x[2]`

The left square brackets **must** immediately follow the name of the variable. 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*'. More details are given in appendix C.

Beside these variables i[n] and r[n] for general use a large variety of variables is linked to specific information of *PDFFIT*, e.g. of the currently loaded structural data or PDF data. Some of these variables c an not be modified, others like atomic positions or thermal factors can be altered, thus allowing to modify the loaded structure. Table 5.1 shows a summary of *PDFFIT* specific variables related to a loaded structure. Note that all structure related variables are linked to the currently active phase. To alter structural information of another phase use the command `phase` first. Variables related to PDF data are listed in Table 5.2 and refinement variables can be found in Table 5.3.

Variable	Description
`n[1]`*	Total number of atoms of current phase
`n[2]`*	Number of different atoms of current phase
`n[3]`*	Number of atoms per unit cell (current phase)
`n[4]`*	Number of phases
`n[5]`*	Number of currently active phase
`x[n]`	x-position of atom n of current phase
`y[n]`	y-position of atom n of current phase
`z[n]`	z-position of atom n of current phase
`m[n]`	Number of atom type on site n (current phase)
`u[i,n]`	Anisotropic thermal factor U_{lm} for site n with ($i = 1..6$ for $U_{11}, U_{22}, U_{33}, U_{12}, U_{13}, U_{23}$)
`o[n]`	Occupancy of site n (current phase)
`lat[i]`	Lattice parameters ($i = 1..6$ for $a, b, c, \alpha, \beta, \gamma$)
`csca[p]`	Scaling factor f_p for phase p
`rhoz[p]`	Number density ρ_0 for phase p

Table 5.1: *PDFFIT* structural variables. Variables marked with * are read-only and can not be altered.

Another variable is `res[i]` which contains the results of a particular *PDFFIT* 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 in this variable.

5.2 Arithmetic expressions

PDFFIT 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)
```

5.3 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 *PDFFIT* are *.lt., .le., .gt., .ge., .eq.* and *.ne.* for operations less than, less equal, greater than, greater equal, equal and not equal, respectively.

Variable	Description
`n[6]`*	Number of loaded experimental PDFs
`np[s]`*	Number of data points of data set s
`pc[n,s]`	Value of calculated PDF point n of data set s for current phase
`po[n,s]`	Value of observed PDF point n of data set s
`pw[n,s]`	Value of weight ($1/\sigma^2$) of point n of data set s
`rmin[s]`	Minimum r-value for dataset s
`rmax[s]`	Maximum r-value for dataset s
`delr[s]`	Step size Δr for dataset s
`rang[i,s]`	Range in r to be calculated or refined ($i = 1, 2$ for lower and upper limit)
`delt[p]`	Value of δ for quadratic PDF peak sharpening for phase p
`gamm[p]`	Value of γ for linear PDF peak sharpening for phase p
`rcut[p]`	Value of r_{cut} below which the PDF peak width is multiplied with ϕ_0
`srat[p]`	Value of ϕ_0 (or σ-ratio)
`qmax[s]`	Maximum Q-value for data set s
`qalp[s]`	Value of $\alpha(s)$ (resolution broadening) for data set s
`qsig[s]`	Value of σ_Q (resolution dampening) for data set s
`dsca[s]`	Scale factor f_s for data set s
`bave[s]`	Average scattering length

Table 5.2: *PDFFIT* PDF related variables. Variables marked with * are read-only and can not be altered.

Logical expressions can be combined by the logical operators *.not.*, *.and.*, *.or.* and *.xor.*. 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 5.6.

5.4 Intrinsic functions

Several intrinsic functions are defined. Each function is referenced, as in FORTRAN, by its name **immediately** followed by a pair of parentheses (and) that include the list of arguments. Trigonometric and arithmetic functions are listed in table 5.4. Table 5.5 contains various random number generating functions.

PDFFIT offers some crystallographic functions to calculate bond length, bond angles and related values. These functions are summarized in Table 5.6.

Finally we have (currently only one) functions related to the refinement itself listed in Table 5.7.

Variable	Description
`n[7]`*	Maximum number of parameters
`n[8]`*	Number of used refinement parameters
`p[n]`	Value of refinement parameter p_n
`dp[n]`*	Standard deviation of `p[n]`
`pf[n]`	Refinement flag for p_n (1=refine, 0=fixed)
`cl[n,m]`*	Value of correlation between parameters p_n and p_m
`rw[1]`*	Expected R-value of refinement
`rw[2]`*	Resulting R-value of refinement
`rw[3]`*	Resulting weighted R-value of refinement
`dx[n]`	Standard deviation of `x[n]`
`dy[n]`	Standard deviation of `y[n]`
`dz[n]`	Standard deviation of `z[n]`
`du[i,n]`*	Standard deviation of `u[i,n]`
`do[n]`*	Standard deviation of `o[n]`
`drhoz[p]`*	Standard deviation of `rhoz[p]`
`dlat[i]`	Standard deviation of `lat[i]`
`ddelt[n]`*	Standard deviation of `delt[n]`
`dgamm[n]`*	Standard deviation of `gamm[n]`
`dsrat[n]`*	Standard deviation of `srat[n]`
`dcsca[n]`*	Standard deviation of `csca[n]`
`ddsca[s]`*	Standard deviation of `dsca[s]`
`dqsig[s]`*	Standard deviation of `qsig[s]`
`dqalp[s]`*	Standard deviation of `qalp[s]`

Table 5.3: *PDFFIT* refinement variables. Variables marked with * are read-only and can not be altered.

5.5 Loops

Loops can be programmed in *PDFFIT* 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

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 5.4: Trigonometric and arithmetic functions

Type	Name	Description
real	ran(r)	Uniformly distributed pseudo random number between 0.0 and 1.0. 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 5.5: Random number functions

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

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 5.6: Crystallographic functions

Type	Name	Description
real	rval(s [,rmi,rma])	Returns the weighted R-value for the given data set <s>. Optionally a the region in r taken into account can be restricted by giving the values <rmi> and <rma>.

Table 5.7: Refinement functions

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 *PDFFIT* 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 been equal to one, only the if block would have been interrupted, while the innermost do loop would have continued without break.

5.6 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 5.3. Enclosed within an if block any valid *PDFFIT* 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 5.5 for further explanations.

```
1  do i[1]=1,n[7]
2  if(dp[i[1]].gt.0.0) then
3  p[i[1]]=(1.0+0.1*(0.5-ran(0)))*p[i[1]]
```

```
4  endif
5  enddo
```

The example listed above illustrates the use of loops and conditional statements within *PDFFIT*. Again, the line numbers are given for easy reference and not part of the actual *PDFFIT* input. This little sequence of commands will add a $\pm 5\%$ random noise to all parameters that have a standard deviation greater than zero. In line 1 starts a do-loop over all possible parameters. The variable n[7] contains this information (see table 5.1 in section 5.1). In line 2 we check whether the standard deviation is greater than zero and if this is true, the value of `p[n]` is altered in line 3. Since the function 'ran' produces a uniformly distributed pseudo random number in the range 0.0 to 1.0, the factor the parameter is multiplied with (line 3) ranges from 0.95 to 1.05.

5.7 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, *PDFFIT* 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 characterised 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 parameter 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.pdf*':

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

The second example shows how filenames are changes within a loop. Here the structure files '*phase1.stru*' to '*phase3.stru*' will be loaded.

```
do i[1]=1,3
..
read stru,"data%d.calc",i[1]
..
enddo
```

5.8 Macros

Any list of valid *PDFFIT* commands can be written to an ASCII file and executed indirectly by the command '@<filename>'. Macro files can be written by any editor on your system or be generated by the `learn` command. The command `learn` starts to remember all the commands that follow and saves them into the given file. 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.

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.

If the program *PDFFIT* is started with command line parameters, e.g. `pdffit 1.mac 2.mac`, the program will execute the given macros in the specified order, in our example first `1.mac` then `2.mac`. If a macro is not found in the current working directory, a system macro director is searched. This system macro directory is located at `path_to_binary/sysmac/pdffit/`. Commonly used macro files might be installed in this directory.

5.9 Working with files

The command language offers the user several commands to write variables to a file or read values from a file. First a file needs to be opened using the command 'fopen'. An optional parameter 'append' allows one to append data to an existing file. Once the task is finished, the file must be closed via 'fclose'. In the standard configuration, the program can open five files at the same time. The first parameter of all file input/output related commands is the unit number which can range from 1 to 5. The commands 'fget' and 'fput' are used to read and write data, respectively. The following example illustrates the usage of these commands:

```
1  fopen 1,sin.dat
2  fput 1,'Cool sinus function'
3  #
```

```
 4  do i[1]=1,50
 5    r[1]=i[1]*0.1
 6    r[2]=sin(r[1])
 7    fput 1,r[1],r[2]
 8  enddo
 9  #
10  fclose 1
```

In line 1 we open the file '*sin.dat*' and write a title (line 2). If the file already exists it will be overwritten. Note that the text must be given in *single* quotes. Text and variables may be mixed in a single line. Next we have a loop calculating $y = \sin(x)$ and writing the resulting x and y values to the open file (line 7). Finally the file is closed (line 10). To read values from a file use simply the command 'fget' and the read numbers will be stored in the specified variables. In contrast to writing to a file, mixing of text and number is not allowed when reading data. However, complete lines will be skipped when the command 'fget' is entered without any parameters.

Chapter 6

Running a refinement

6.1 Refinement variable coding

When using *PDFFIT* we have to distinguish between refinement parameters, called p_i (or `p[i]`) and experimental and structural parameters listed in Table 6.1. The parameters that can be assigned to refinement parameters are a subset of all available variables discussed in section 5.1.

Structural	(Set phase using `phase` command !)
`x[n]`	x-position of atom n in fractional coordinates
`y[n]`	y-position of atom n in fractional coordinates
`z[n]`	z-position of atom n in fractional coordinates
`o[n]`	Occupancy for site n
`u[i,n]`	Anisotropic thermal parameter U_{ij} for atom n ($i = 1..6$ for $U_{11}, U_{22}, U_{33}, U_{12}, U_{13}$ and U_{23})
`lat[i]`	Lattice parameters ($i = 1..6$ for a, b, c, α, β and γ)
`delt[p]`	Quadratic correlation factor δ for phase p
`gamm[p]`	Linear correlation factor $gamma$ for phase p
`srat[p]`	Peak width ratio for $r < r_{cut}$ for phase p
`csca[p]`	Scale factor f_p for phase p
Experimental	
`dsca[s]`	Overall scale factor f_s for data set s
`qalp[s]`	Broadening factor α for data set s
`qsig[s]`	Resolution factor σ_Q for data set s

Table 6.1: *PDFFIT* experimental and structural refinement variables.

Each experimental or structural parameter that needs to be refined must be assigned to one or more refinement parameters via the command `par`. The maximum number of parameters p_i allowed in a single `par` command is defined by the variable MAXDPP in the file '*config.inc*' (see appendix C). For each pa-

rameter p_i that appears in the definition we have to also specify the derivative of the corresponding structural parameter with respect to all refinement parameters p_i. The value of i of the refinement parameters can be freely chosen by the user, however there is of course an upper limit. This limit together with other program related limits can be determined via the command `show config`.

Let us consider a simple case like in the example given in section 2. The lattice parameters a, b, c of Ni shall be refined using a single refinement parameter since Ni is cubic. The definitions we need to enter are listed below. In lines 1–3 refinement parameter p_1 is associated with the lattice constants a, b, c. It is important to remember, that one needs to assign a suitable starting value to p_1 as we have done in line 5 of our example.

```
1 par lat[1]=p[1],1.0
2 par lat[2]=p[1],1.0
3 par lat[3]=p[1],1.0
4 #
5 p[1]=lat[1]
```

Next we consider a slightly more complicated construction. It is again taken from the example in section 2. Assuming we want to refine a single isotropic temperature factor for all atoms within the unit cell, we can either type in all definitions explicitly of use a `do` loop as in the example below. The loop index is `i[1]` and `n[1]` contains the total number of atoms for the currently active phase. Note that *PDFFIT* replaces all variables in the arguments, i.e. enclosed in `[.]`, with the actual value at the time the command is processed. Try the example below and check the stored definitions using the command `show const`. Again we need to set the used refinement parameters to a suitable stating value (line 7).

```
1 do i[1]=1,n[1]
2   par u[1,i[1]]=p[2],1.0
3   par u[2,i[1]]=p[2],1.0
4   par u[3,i[1]]=p[2],1.0
5 enddo
6 #
7 p[2]=u[1,1]
```

As a final example we consider a (hypothetical) system containing a rigid molecule consisting of two atoms, one at $(\frac{1}{4}, 0, 0)$ and one at $(-\frac{1}{4}, 0, 0)$. Note that *DISCUS* supports the definition of molecules in its structure file. However, this extension is *not* supported by *PDFFIT*. The rotation of the molecule around the z-axis through the center of the molecule (conveniently located at $(0, 0, 0)$) shall be modelled by a single refinement parameter, the rotation angle ψ. This rotation can be described by the following matrix

$$R = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.1}$$

All we need to do now is to describe the new positions of the two atoms as a function of the rotation angle ψ and specify the corresponding derivatives. One should keep in mind that the atom positions are in *fractional*

coordinates. Furthermore one should *not* use definition of the type `par x[1]=x[1]*..` since `x[1]` is changing during the refinement. So we need to store the relevant information in some general variable `r[n]`. The parameter coding for our example is shown below:

```
 1 r[1]=x[1]
 2 r[2]=y[1]
 3 r[3]=x[2]
 4 r[4]=y[2]
 5 #
 6 par x[1]=    r[1]*cosd(p[5])+r[2]*sind(p[5]),-1.*r[1]*sind(p[5])+r[2]*cosd(p[5])
 7 par y[1]=-1.*r[1]*sind(p[5])+r[2]*cosd(p[5]),-1.*r[1]*cosd(p[5])-r[2]*sind(p[5])
 8 par x[2]=    r[3]*cosd(p[5])+r[4]*sind(p[5]),-1.*r[3]*sind(p[5])+r[4]*cosd(p[5])
 9 par y[2]=-1.*r[3]*sind(p[5])+r[4]*cosd(p[5]),-1.*r[3]*cosd(p[5])-r[4]*sind(p[5])
10 #
11 p[5]=4.5
```

In lines 1–4 the x and y position of both atoms is stored in the variables `r[1]` to `r[4]`. Note that z is not changing for our specific rotation. In lines 6–9 we give the equations according to (6.1) and the corresponding derivative with respect to parameter p_5. The intrinsic functions `sind` and `cosd` will expect their argument to be in degrees whereas `sin` and `cos` need the argument to be specified in radian. We chose to use degrees in our example. More information about intrinsic functions can be found in section 5.4. Currently the command language interpreter will not accept expressions that start with '-' like `-r[1]`, so one needs to specify more completely `-1.*r[1]`. The command `show const` will list all parameter definitions on the screen. The output for our example is listed below:

```
------------------------------------------------------------------------
PARAMETER DEFINITIONS :
------------------------------------------------------------------------

Definition  : x[1]=r[1]*cosd(p[5])+r[2]*sind(p[5])
Derivatives : d/d(p[   5]) = -1.*r[1]*sind(p[5])+r[2]*cosd(p[5])

Definition  : y[1]=-1.*r[1]*sind(p[5])+r[2]*cosd(p[5])
Derivatives : d/d(p[   5]) = -1.*r[1]*cosd(p[5])-r[2]*sind(p[5])

Definition  : x[2]=r[3]*cosd(p[5])+r[4]*sind(p[5])
Derivatives : d/d(p[   5]) = -1.*r[3]*sind(p[5])+r[4]*cosd(p[5])

Definition  : y[2]=-1.*r[3]*sind(p[5])+r[4]*cosd(p[5])
Derivatives : d/d(p[   5]) = -1.*r[3]*cosd(p[5])-r[4]*sind(p[5])
```

Basically this type of assigning refinement parameters to structural parameters allows the user to realize nearly all type of constraint models. In the next sections specific details of structural parameters as well as refinement setting will be discussed.

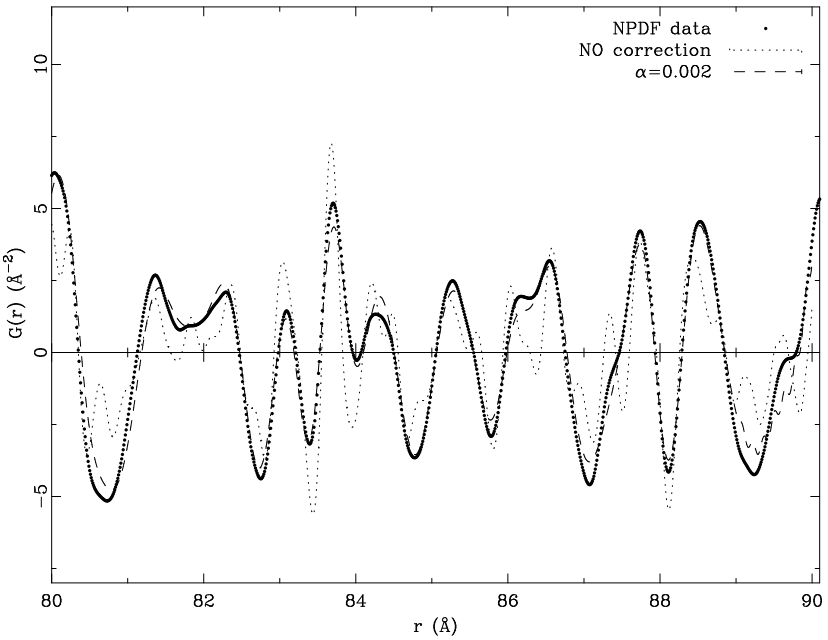

Figure 6.1: Result of PDF refinement of Ni showing the influence of the PDF peak broadening parameter, α. The filled circles are the data collected on the instrument NPDF at the Lujan Center at Los Alamos National Laboratory. The dotted line is a refinement without α and the dashed line is a refinement using the PDF broadening with $\alpha = 0.002$. Note that the range displayed extends from 80Å to 90Å.

6.1.1 Structural parameters

The structural parameters that can be refined using *PDFFIT* were already listed in Table 6.1 at the beginning of this chapter. Parameters `x[n], y[n]` and `z[n]` are the *fractional* coordinates of atom n of the currently active structural phase (see section 6.1.3). The site occupancy is given by `o[n]`. The anisotropic temperature factor U_{lm} is stored in the variable `u[i,n]` with $i = 1..6$ for $U_{11}, U_{22}, U_{33}, U_{12}, U_{13}$ and U_{23}. Note that $U_{lm} = \langle u_l u_m \rangle$ and in particular $U_{ll} = \langle u_{ll}^2 \rangle$.[1] Two other forms of the temperature factor commonly used in the literature are: $B_{lm} = 8\pi^2 U_{lm} = 8\pi^2 \langle u_l u_m \rangle$ and $\beta_{lm} = \frac{1}{4} \mathbf{a}_l^* \mathbf{a}_m^* B_{lm}$ where $\mathbf{a}^*$ are the reciprocal lattice parameters.

The remaining parameters are the scale factor `csca[p]` and the parameters `delt[p]` and `srat[p]` modelling the r dependence of the PDF peak width as discussed in the next section. Note that these parameters take the number of the corresponding structural phase p as argument.

6.1.2 R-dependence of the PDF peak width

The PDF peak width contains contributions from thermal and zero point displacements as well as static disorder. For large distances r the motion of the two contributing atoms is uncorrelated, for small distances, however, the motion can be strongly correlated leading to a sharpening of the first peak(s) in the observed

[1] The old *RESPAR* program was using $\langle u \rangle$ in its input file !

PDF (see Jeong et al. (1998)). The program PDFFIT provides three different correction terms for the PDF peak width. The final width is given by

$$\sigma_{ij} = \sqrt{\sigma'^2_{ij} - \frac{\delta}{r^2_{ij}} - \frac{\gamma}{r_{ij}} + \alpha^2 r^2_{ij}}. \tag{6.2}$$

Here σ' is the peak width without correlation given by the structural model. The first two terms correct for the effects of correlated motion. Details about these terms can be found in Jeong et al. (2002). Within the scope of the users guide, we just mention that the term δ/r^2 describes the low temperature behavior and the term γ/r describes the high temperature case. Since the two parameters are highly correlated, one will in practice choose which one to refine. The last term in equation 6.2 models the PDF peak broadening as a result of the Q resolution of the diffractometer. Details about this corrections can be found in Thorpe et al. (2002). In many cases this term will only be significant for refinements of wider ranges in r. Note that the Q resolution also results in an exponential dampening of the PDF peaks which is modelled using the parameter σ_Q. Complete details of the equations used in PDFFIT, refer to appendix A.

The last mechanism to sharpen PDF peaks at low r becomes necessary e.g. when a system has a static displacement component which is reflected by the structural model only up to a certain distance. In this case *PDFFIT* allows the user to sharpen the PDF peaks below a value of $r = r_{cut}$ by a factor $\phi < 1$. Thus below r_{cut} the temperature factor reflect only thermal and zero-point motion whereas at higher r a combination of static and dynamic displacements are modelled by the thermal parameters. The cutoff value is set using the variable `rcut[p]` and the ratio ϕ is set by `srat[p]` (read sigma ratio).

6.1.3 Refining multiple phases or data sets

PDFFIT is capable of refining multiple structural phases and multiple PDF data sets. Let us consider multiple data sets first. Simply read each data set using the command `read data` after starting the program or using the command `reset`. Assign a refinement parameter to the experimental variables of each data set. Setup the structural model as before and run the refinement. The calculated PDF for each data set needs to be saved using the command `save pdf,n,file` where `n` is the number of the data set. When refining a model with more than one structural phase read the corresponding structure using the command `read stru`. Next the definition of a structural model needs to be done for each phase. Use the command `phase` to make the phase active before entering the corresponding parameter definitions. Since the variables `delt[p], rcut[p], srat[p]` and `csca[p]` use the phase number as argument, the corresponding definitions can be entered without using the command `phase`. Again after running the refinement, the resulting structure of each phase must be saved separately via `save stru,p,file` where `p` is the number of the phase.

The new command `psel` allows the user to associate certain phases $p_i, i = 1..n$ with a given data set s. One extreme would be to have e.g. two data sets and one phase each completely separate which is not really different from running the refinements separately. However, by using identical refinement parameters one can constrain any structural parameters between the phases. The default is that all structural phases are associated with all loaded data sets.

When using multiple phases and data sets one needs to be careful who to use the scale factors `csca[p]` and `dsca[s]`. The program makes no internal normalization and the scale factors might be highly correlated.

6.1.4 Difference modeling

Sometimes it might be beneficial to refine the *change* of a PDF e.g. when crossing a phase transition rather than the PDF itself. One advantage is that systematic errors might cancel when calculating the difference. *PDFFIT* has the option to refine a difference PDF $D(r) = G(r) - G_r(r)$. The normal PDF $G(r)$ is derived from the model structure and subsequently a reference PDF $G_r(r)$ is subtracted.

Difference modeling is enabled by using the command `read diff` rather than `read data` when reading the PDF data. The command `read diff` requires an additional parameter containing the name of a file containing the reference PDF $G_r(r)$. The data file must now contain the difference between the observed and reference PDF. It is important that both PDFs cover identical points in r. The command `save pdf` will save the resulting difference PDF and `save dif` will save the difference between the calculated difference PDF and the experimental difference PDF, I hope this is not confusing.

6.2 Setting refinement options

There are only two settings the user can change before a refinement is started using the command `run`: the maximum number of iterations (`cycle`) and the magic number URF (some German: Unterer Relaxations Faktor, command `urf`). This value determines who 'fast' the fit will move to its minimum or how much the parameter values are changed in each cycle depending on the deviations. A small value (e.g. 0.1) might lead to a fast convergence but might also miss the minimum. A larger value (e.g. 100.0) will give a slow convergence which more certain finds the minimum, but might be caught in local minima rather than in the global one. Understood ? Well just try different values until your fit converges nicely to the global minimum. The convergence of a refinement can be judged by the final difference in the R-value displayed on the screen. Obviously a small difference is the goal. *PDFFIT* leaves the convergence criterion to the user. The command `show fit` will list the current refinement settings and results. If the number of actually computed iteration is equal to the maximum number of iterations, the refinement needs to be continued by simply issuing the command `run` again. One might also need to increase the maximum number of iterations using the command `cycle`.

When refining a model one can rarely refine all desired parameters from the start. So a good start is for example to refine the scale factor first followed by lattice parameters and so on. So one could add the parameter definitions gradually. However, *PDFFIT* offers refinement flags `pf[n]`. To refine a certain parameter n simply use `pf[n]=1`, to keep it fixed use `pf[n]=0`. The default is that all parameters that are used in any definition will be refined.

6.3 Saving results

After a refinement has been carried out, the results must be saved before *PDFFIT* is terminated, otherwise **all results will be lost**. The following sequence of command will save all results of a refinement:

```
1 save pdf,1,result.pdf
2 save dif,1,result.dif
3 save str,1,result.stru
4 save res,result.res
```

In lines 1–2 the calculated PDF and the difference between observed and calculated PDF are written to the files *result.pdf* and *result.dif*. The file formats were already discussed in section 3.1. The parameter `1` stands for the PDF and difference corresponding to the experimental dataset one. In cases where multiple datasets are refined, the results for each set must be saved separately. In line 3 the resulting structure is saved using the *PDFFIT* structure file format. Here the `1` stands for structural phase one. Again for refinements using multiple phases, each resulting structural phase needs to be save separately. Finally the a summary of the refinement settings and results is saved to the file *result.res* (line 4). The resulting structure can alternatively be saved in the *DISCUS* structure file format using the command `save disc,1,file.stru`. This allows one to used *DISCUS* for further analysis of the resulting structure. However, some information like the standard deviations and the anisotropic temperature factors U_{ij} will be lost. Thus it is recommended to save the resulting structure always in *PDFFIT* format as well.

Chapter 7

Examples

A simple example of the refinement of a single data set of Ni was already given in section 2. Here we will discuss three slightly more complicated refinement examples. Some of the example files are part of the tutorial found in the *PDFFIT* distribution.

7.1 Example 1: InGaAs

In this section we will use $In_{0.5}Ga_{0.5}As$ to discuss the question: What can be done when the first calculated PDF peak is not sharp enough compared to the data ?

Figure 7.1 shows the result of a straight forward refinement of $In_{0.5}Ga_{0.5}As$ data. These data were collected by V. Petvok and I.-K. Jeong at Cornell High Energy Synchrotron Source. The maximum Q value of these data is $Q_{max} = 45$Å^{-1}. The corresponding macro file is listed below. As usual we have added line numbers for convenience.

```
1 reset
2 read stru,in50.stru
3 read data,x,45.0,0.0,in50.data
4 #
5 urf 5.
6 cyc 100
7 range 1,1.5,10.0
```

First we reset *PDFFIT* (line 1), read the starting structure (line 2) and the data (line 3). Our starting structure includes static nearest neighbor displacements to account for the different In-As and Ga-As bond length. The atom coordinates of the starting structure are listed below:

```
GA         -0.00500000       -0.00350000       -0.00350000
IN          0.00550000        0.50550002        0.50550002
GA          0.49450001       -0.00550000        0.49450001
IN          0.50449997        0.50449997        0.00450000
AS          0.26499999        0.23500000        0.26499999
```

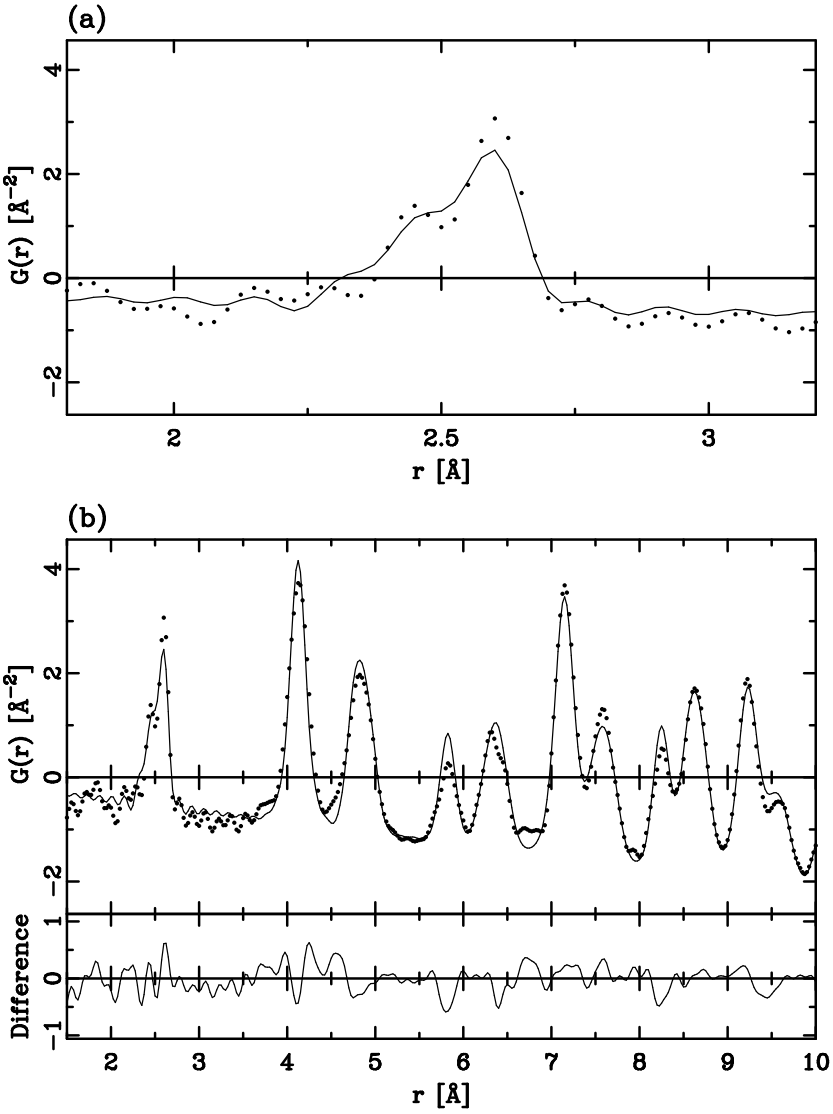

Figure 7.1: Result of PDF refinement A of $In_{0.5}Ga_{0.5}As$. The solid line is the calculated PDF, the filled circles are the data. Panel (a) shows the region around the split first PDF peak and (b) shows the complete refinement range. A difference curve is plotted below the data.

```
AS          0.26499999          0.76499999          0.73500001
AS          0.75599998          0.26199999          0.73600000
AS          0.73699999          0.76300001          0.23700000
```

As one can see, half of the metal sites are occupied by In, the other half by Ga. All atoms have been shifted from their ideal position to accommodate the static displacement. The positions shown could be refined using *PDFFIT*, however, this is beyond the scope of this section. In the next part we assign parameters for

the lattice parameters $a = b = c$ (line 13–15) and corresponding starting values (lines 17–19). Next the scaling factor, σ_Q and the dynamic correlation factor δ are assigned to their refinement parameters (lines 21–23) and again starting values for those parameters need to be given (lines 25–27).

```
 8 #
 9 ######################################################################
10 # Experimental and lattice parameters
11 ######################################################################
12 #
13 par lat[1]=p[1],1.0
14 par lat[2]=p[1],1.0
15 par lat[3]=p[1],1.0
16 #
17 p[1]=lat[1]
18 p[2]=lat[1]
19 p[3]=lat[1]
20 #
21 par csca[1]=p[20],1.0
22 par qsig[1]=p[21],1.0
23 par delt[1]=p[22],1.0
24 #
25 p[20]=0.40
26 p[21]=0.03
27 p[22]=0.10
```

In the next segment we assign two parameters for the isotropic temperature factors $U = U_{11} = U_{22} = U_{33}$ for the metal site occupied by In and Ga as well as for As. This is done in a similar fashion than for the Ni example given in chapter 2 by using `do` loops. A scan be seen from the structure file shown above, the first four atoms are the metals (In, Ga) and atoms 5 to 8 are As, so the first loop (line 33) loops over atoms 1–4, the second loop goes over numbers 5–8.

```
28 #
29 ######################################################################
30 # Temperature factors
31 ######################################################################
32 #
33 do i[1]=1,4
34   par u[1,i[1]]=p[4],1.0
35   par u[2,i[1]]=p[4],1.0
36   par u[3,i[1]]=p[4],1.0
37 enddo
38 #
39 do i[1]=5,8
40   par u[1,i[1]]=p[5],1.0
41   par u[2,i[1]]=p[5],1.0
42   par u[3,i[1]]=p[5],1.0
43 enddo
```

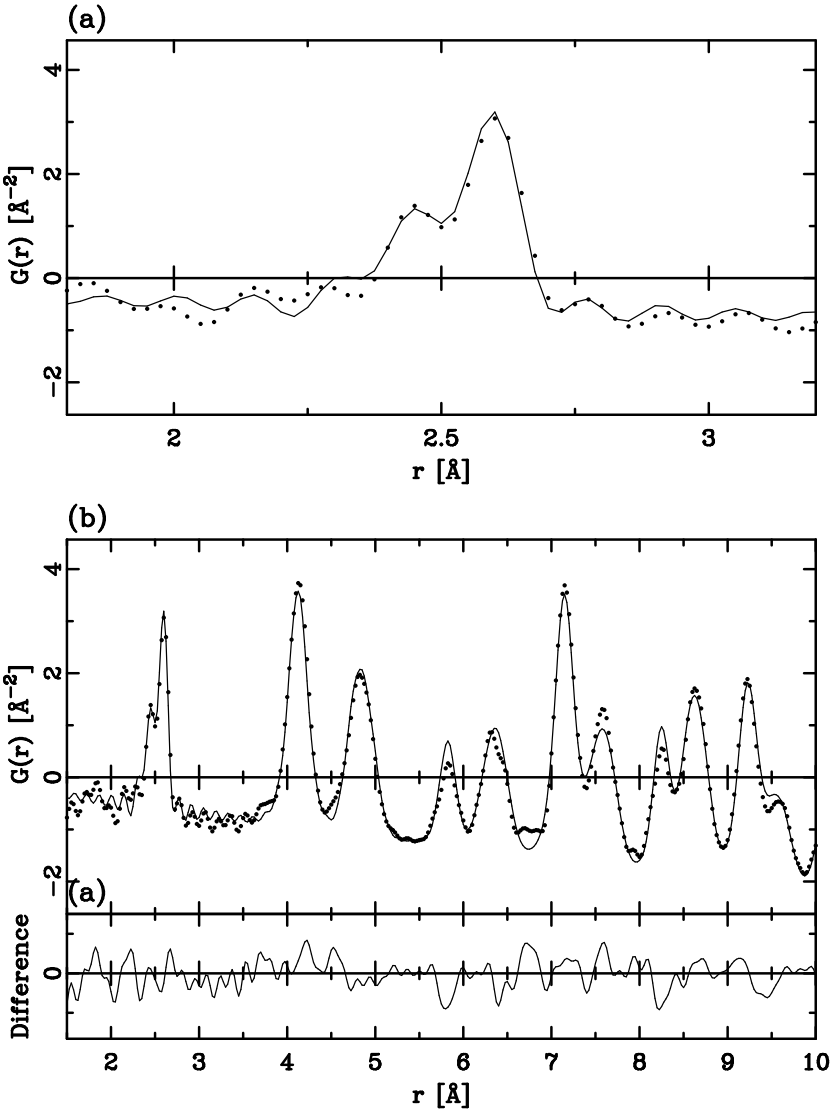

Figure 7.2: Result of PDF refinement B of $In_{0.5}Ga_{0.5}As$. Details see caption of Figure 7.1.

```
44 #
45 p[4]=u[1,1]
46 p[5]=u[1,5]
```

After all parameter coding is done, we need to start the refinement (line 47) and after the run is finished, we need to save the results to individual files (lines 49–52).

```
47 run
48 #
49 save pdf,1,in50.pdf
```

```
50 save dif,1,in50.dif
51 save stru,1,in50.rstr
52 save res,in50.res
```

Inspection of Figure 7.1 shows a reasonable agreement when taking the complete r range into account, however, the split first neighbour peak shown in panel (a) of Figure 7.1 agrees rather badly with the observed PDF. To understand this, we need to consider, that we have modeled static displacements only for the nearest neighbour and consequently only the width of the first PDF peak is completely determined by thermal broadening whereas the other peaks are determined by thermal as well as static displacements. The dynamical correlation factor δ describes the r dependence of the PDF peak with according to the following empirical equation:

$$\sigma = \sigma` - \frac{\delta}{r^2} \tag{7.1}$$

with σ standing for the actual PDF peak width and σ' the width for uncorrelated motion, the value σ converges to for high values of r. However, the parameter δ is designed to take the correlated nature of thermal motion into account. But in our case, we need a sharper transition from the first peak which is purely thermal in origin and the other peaks that contain thermal as well as static contributions. To solve this problem, we sharpen the PDF peaks below a given value r_{cut} by a factor ϕ (see section 6.1.2 for details. All we need to do, is to add the following lines to our example macro:

```
28a  par srat[1]=p[23],1.0
28b  p[23]=0.3
28c  rcut[1]=3.0
```

The ratio ϕ stored in the variable `srat[s]` is assigned to refinement parameter `p[23]` (line 28a) and a suitable starting value is set (line 28b). Furthermore, we need to define the value of r_{cut} or `rcut[s]` for the corresponding data set. This value determines below which value of r the PDF peak width is multiplied by ϕ. The result of this modified refinement can be seen in Figure 7.2. It is obvious that the refinement describes the first PDF peak much better, which is also reflected in the R-value of R=15.6% compared to R=19.1% for the refinement without additional sharpening.

7.2 Example 2: Manganites

As a next example we will discuss a slightly more complicated example, $LaMnO_3$. The structure is orthorhombic at room temperature, space group $Pbnm$. Assuming one has an experimental PDF, the first step is to obtain a starting structure. The data used in this example were measured by G.H. Kwei and S.J.L. Billinge on the special environment diffractometer (SEPD) at the IPNS, Argonne. Unfortunately space group $Pbnm$ is a nonstandard setting and the relations must be derived from space group $Pnma$ listed in the International Tables for Crystallography Wilson et al. (1983) by cyclic permutation of the elements of the relevant generators. Because we need this information later on, the symmetrically equivalent positions are listed in Table 7.1.

(8d)	x, y, z	$x+\frac{1}{2}, \overline{y}+\frac{1}{2}, \overline{z}$	$\overline{x}, \overline{y}, z+\frac{1}{2}$	$\overline{x}+\frac{1}{2}, y+\frac{1}{2}, \overline{z}+\frac{1}{2}$
	$\overline{x}, \overline{y}, \overline{z}$	$\overline{x}+\frac{1}{2}, y+\frac{1}{2}, z$	$x, y, \overline{z}+\frac{1}{2}$	$x+\frac{1}{2}, \overline{y}+\frac{1}{2}, z+\frac{1}{2}$
(4c)	$x, y, \frac{1}{4}$	$x+\frac{1}{2}, \overline{y}+\frac{1}{2}, \frac{3}{4}$	$\overline{x}, \overline{y}, \frac{3}{4}$	$\overline{x}+\frac{1}{2}, y+\frac{1}{2}, \frac{1}{4}$
(4b)	$0, \frac{1}{2}, 0$	$\frac{1}{2}, 0, 0$	$0, \frac{1}{2}, \frac{1}{2}$	$\frac{1}{2}, 0, \frac{1}{2}$

Table 7.1: Symmetrically equivalent positions in space group $Pbnm$ for Mn on (4b), La, O on (4c) and O on (8d).

Using the same procedure as described in section 3.2 we will use *DISCUS* to expand the structure using the symmetry generators in space group $Pbnm$. The following input file describes the structure as found in the literature (Rodríguez-Carvajal et al., 1998).

```
title LaMnO3 - La (4c) - Mn (4b) - O (4c & 8d)
spcgr Pbnm
cell   5.5367   5.7473    7.6929   90.000000   90.000000   90.000000
atoms
LA         -0.007800        0.049000        0.250000     0.3400
MN          0.000000        0.500000        0.000000     0.2100
O           0.074500        0.487400        0.250000     0.5000
O           0.725600        0.306600        0.038400     0.4300
```

Note that if we would use this structure file as input for *PDFFIT* only those four atoms would be used. It is a required step to use *DISCUS* to generate the other positions or add them by hand. The expanded structure file that is used for the refinement is shown below.

```
title LaMnO3 - La (4c) - Mn (4b) - O (4c & 8d)
spcgr Pbnm
cell   5.5367   5.7473    7.6929   90.000000   90.000000   90.000000
ncell        1,       1,       1,        20
atoms
LA           .992200         .049000         .250000      .3400
LA           .492200         .451000         .750000      .3400
LA           .007800         .951000         .750000      .3400
LA           .507800         .549000         .250000      .3400
MN           .000000         .500000         .000000      .2100
MN           .500000         .000000         .000000      .2100
MN           .000000         .500000         .500000      .2100
MN           .500000         .000000         .500000      .2100
O            .074500         .487400         .250000      .5000
O            .574500         .012600         .750000      .5000
O            .925500         .512600         .750000      .5000
O            .425500         .987400         .250000      .5000
O            .725600         .306600         .038400      .4300
O            .225600         .193400         .961600      .4300
O            .274400         .693400         .538400      .4300
O            .774400         .806600         .461600      .4300
```

```
O           .274400         .693400         .961600      .4300
O           .774400         .806600         .038400      .4300
O           .725600         .306600         .461600      .4300
O           .225600         .193400         .538400      .4300
```

This structure file now contains all 20 atoms per unit cell. Note that *DISCUS* could also have been used to create a larger model structure, e.g. 2x1x1 unit cells. Note that per convention *DISCUS* will transform the fractional coordinates of all atoms within a single unit cell to the range 0 to 1.

After having the required input files, i.e. an experimental PDF and a starting structure, one can start the refinement. In our example we want to refine isotropic temperature factors as well as positions, however, the symmetry constraints given by space group $Pbnm$ shall be applied. The following sequence of commands is used for the refinement. Obviously it is more convenient to prepare a text file with the commands and start it as a macro in *PDFFIT* using the `@` command. As before the line numbers in the example here are for easy reference only and not part of the actual input.

```
1 reset
2 read stru,lmo.stru
3 read data,n,27.0,0.03,lmo.data
4 #
5 urf 5.0
6 cyc 100
7 range 1.0,15.0
```

After *PDFFIT* has been reset (line 1), the starting structure (line 2) and PDF data (line 3) are read. The parameters in line 3 identify the data to be obtained from neutron scattering (`n`) with a maximum Q value of $Q_{max} = 27$Å^{-1}. The next parameters specifies the value for σ_Q followed by the name of the data file. For details about the file formats and conversion tools refer to chapter 3 of this manual. In line 5 the magic number for the refinement is set to 5.0 (see section 6.2). Next the maximum number of iterations is set to 100, which is of course much larger than the number of iteration we expect before the fit converges (or explodes). The command `range` (line 7) sets the range in real space used for the refinement to 1.0 to 15.0Å. The main part of the example macro contains refinement parameter definitions. We start with lattice parameters and some experimental parameters.

```
 8 #
 9 ################################################################
10 # Experimental and lattice parameters
11 ################################################################
12 #
13 par lat[1]=p[1],1.0
14 par lat[2]=p[2],1.0
15 par lat[3]=p[3],1.0
16 #
17 p[1]=lat[1]
18 p[2]=lat[2]
19 p[3]=lat[3]
```

```
20 #
21 par csca[1]=p[200],1.0
22 par qsig[1]=p[201],1.0
23 par delt[1]=p[202],1.0
24 #
25 p[200]=0.40
26 p[201]=0.03
27 p[202]=0.10
```

Lines 8–12 are simply a comment to make the macro file easier to read. It is generally recommended to include descriptive comments in refinement macros. Since our structure is orthorhombic, we assign three parameters `p[1],p[2]` and `p[3]` to refine the lattice parameters a, b and c stored in variables `lat[1], lat[2]` and `lat[3]` (line 13–15). The syntax of the command `par` was already discussed in detail in section 6.1. The starting values for the parameters `p[1]` to `p[3]` are set in lines 17–19. It is important to assign each used refinement parameter a proper starting value before starting the refinement. In lines 21–23 we assign parameters to the scale factor f_p, the resolution parameter σ_Q and the dynamic correlation factor δ. Again each parameter needs a proper starting value (lines 25–27). The choice which parameters `p[n]` is assigned to which experimental or structural parameter is completely up to the user.

```
28 #
29 ######################################################################
30 # Temperature factors
31 ######################################################################
32 #
33 do i[1]=1,4
34   par u[1,i[1]]=p[4],1.0
35   par u[2,i[1]]=p[4],1.0
36   par u[3,i[1]]=p[4],1.0
37 enddo
38 #
39 do i[1]=5,8
40   par u[1,i[1]]=p[5],1.0
41   par u[2,i[1]]=p[5],1.0
42   par u[3,i[1]]=p[5],1.0
43 enddo
44 #
45 p[4]=u[1,1]
46 p[5]=u[1,5]
47 #
48 do i[1]=9,16
49   par u[1,i[1]]=p[6],1.0
50   par u[2,i[1]]=p[6],1.0
51   par u[3,i[1]]=p[6],1.0
52 enddo
53 #
54 do i[1]=17,20
55   par u[1,i[1]]=p[7],1.0
```

```
56   par u[2,i[1]]=p[7],1.0
57   par u[3,i[1]]=p[7],1.0
58 enddo
59 #
60 p[6]=u[1, 9]
61 p[7]=u[1,17]
```

In out structure, we have atoms on four different positions: Mn on (4b), La, O on (4c) and O on (8d). The program *PDFFIT* assigns numbers to each atom in the same order they appear in the structure file. Inspecting the structure file listed above, we find that atoms 1–4 are Mn, 5–8 are La, 9–16 are the oxygens on position (4c) and finally atoms 17–20 are the oxygens on position (8d). The command sequence above assigns an individual isotropic thermal parameter $U = U_{11} = U_{22} = U_{33}$ to all four atom types (two for the different oxygens). In this example we use `do` loops for the parameter assignment (see section 6.1). The first loop (line 33) goes over atoms 1 to 4 and assigns parameters `u[1,n], u[2,n]` and `u[3,n]` a single refinement parameter `p[4]`. The same is done for the La atoms in lines 39–43. Next starting values are given to the parameters (lines 45–46). The last part of this segment assigns isotropic temperature factors to the two different oxygen sites in a similar way.

So far the example was quite straight forward and not really different from the simple macro given in chapter 2. Next we want to refine the atom positions, but restricted to the space group $Pbnm$. For Mn we have no free positional parameter, for La and oxygen on position (4c) there are two refinable parameters (x, y) and for the oxygen on position (8d) all three coordinates can be refined. The parameter coding for La is show below:

```
62 #
63 ####################################################################
64 # Positions (constrained)
65 ####################################################################
66 #
67 # La on (4c)
68 #
69 par x[ 1]= 1.0+p[21], 1.0
70 par y[ 1]= 0.0+p[22], 1.0
71 par x[ 2]= 0.5+p[21], 1.0
72 par y[ 2]= 0.5-p[22],-1.0
73 par x[ 3]= 0.0-p[21],-1.0
74 par y[ 3]= 1.0-p[22],-1.0
75 par x[ 4]= 0.5-p[21],-1.0
76 par y[ 4]= 0.5+p[22], 1.0
77 #
78 p[21]=-0.0078
79 p[22]= 0.0490
```

The x coordinate of atom one is assigned to parameter `p[21]` and y is assigned `p[22]`. In line 69 and 74 we have added 1.0 to conform to the *DISCUS* convention that fractional coordinates range from 0 to 1. The atom positions of the other three La atoms are calculated according to the relations given in Table 7.1.

Note the sign of the derivatives in this example ! The coding for the oxygens on site (4c) is exactly the same (apart from ± 1.0) and shown below:

```
 80 #
 81 # O on (4c)
 82 #
 83 par x[ 9]=0.0+p[23], 1.0
 84 par y[ 9]=0.0+p[24], 1.0
 85 par x[10]=0.5+p[23], 1.0
 86 par y[10]=0.5-p[24],-1.0
 87 par x[11]=1.0-p[23],-1.0
 88 par y[11]=1.0-p[24],-1.0
 89 par x[12]=0.5-p[23],-1.0
 90 par y[12]=0.5+p[24], 1.0
 91 #
 92 p[23]= 0.0745
 93 p[24]= 0.4874
```

The coding for the oxygen atoms on site (8d) is slightly longer since we change all three coordinates and have 8 symmetrically equivalent positions. However, following Table 7.1 there should be no difficulty to understand the following part of the *PDFFIT* macro.

```
 94 #
 95 # O on (8d)
 96 #
 97 par x[13]= 0.0+p[25], 1.0
 98 par y[13]= 0.0+p[26], 1.0
 99 par z[13]= 0.0+p[27], 1.0
100 par x[14]=-0.5+p[25], 1.0
101 par y[14]= 0.5-p[26],-1.0
102 par z[14]= 1.0-p[27],-1.0
103 par x[15]= 1.0-p[25],-1.0
104 par y[15]= 1.0-p[26],-1.0
105 par z[15]= 0.5+p[27], 1.0
106 par x[16]= 1.5-p[25], 1.0
107 par y[16]= 0.5+p[26], 1.0
108 par z[16]= 0.5-p[27],-1.0
109 par x[17]= 1.0-p[25],-1.0
110 par y[17]= 1.0-p[26],-1.0
111 par z[17]= 1.0-p[27],-1.0
112 par x[18]= 1.5-p[25],-1.0
113 par y[18]= 0.5+p[26], 1.0
114 par z[18]= 0.0+p[27], 1.0
115 par x[19]= 0.0+p[25], 1.0
116 par y[19]= 0.0+p[26], 1.0
117 par z[19]= 0.5-p[27],-1.0
118 par x[20]=-0.5+p[25], 1.0
119 par y[20]= 0.5-p[26],-1.0
```

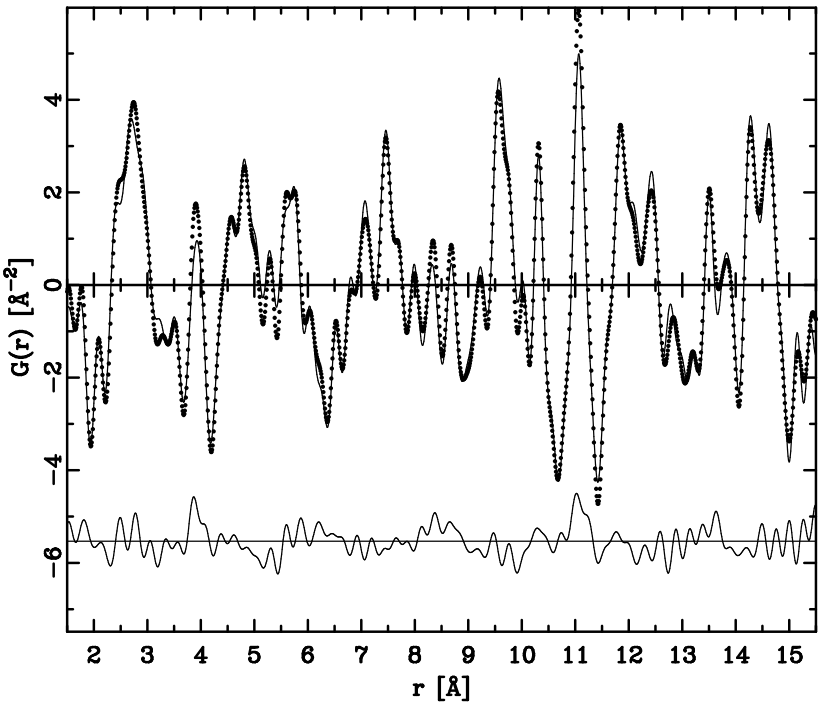

Figure 7.3: Result of PDF refinement of $LaMnO_3$. The solid line is the calculated PDF, the filled circles are the data. A difference curve is plotted below the data.

```
120 par z[20]= 0.5+p[27], 1.0
121 #
122 p[25]= 0.7256
123 p[26]= 0.3066
124 p[27]= 0.0384
```

Now the parameter coding is complete and the refinement can be started using the command `run` (line 125). After the fit has converged, the results have to be stored. In line 127 the resulting structure is saved to the file *lmo_ref.stru*, next the calculated PDF and its difference to the experimental data are written to a file (lines 128–129) and finally the refinement result is saved to the file *lmo_ref.res*.

```
125 run
126 #
127 save stru,1,lmo_ref.stru
128 save pdf,1,lmo_ref.pdf
129 save dif,1,lmo_ref.dif
130 save res,lmo_ref.res
```

The result of the refinement is shown in Figure 7.3. The next step one might be interested in is to refine the positions of the oxygen atoms without the symmetry restrictions imposed by space group $Pbnm$ since

the $Mn - O$ octahedra might have *locally* a different symmetry. One way of doing this is to construct different constraint equations based on geometrical considerations. Alternatively one could refine all oxygen coordinates individually. One possible parameter coding shown below is quite similar to the coding of the thermal factors above.

```
################################################################
# Positions O1 and O2 (free)
################################################################
do i[1]=9,12
  par x[i[1]]=p[i[1]*3+51],1.0
  par y[i[1]]=p[i[1]*3+52],1.0
#
  p[i[1]*3+51]=x[i[1]]
  p[i[1]*3+52]=y[i[1]]
enddo
#
do i[1]=13,20
  par x[i[1]]=p[i[1]*3+51],1.0
  par y[i[1]]=p[i[1]*3+52],1.0
  par z[i[1]]=p[i[1]*3+53],1.0
#
  p[i[1]*3+51]=x[i[1]]
  p[i[1]*3+52]=y[i[1]]
  p[i[1]*3+53]=z[i[1]]
enddo
```

Note that we are not refining the z value of the oxygen sitting on site (4c), although in principle one could try even that. It is important to understand, that the PDF probes the *local* structure. By selecting different ranges in r one has control over the length scale on which the refined *local* arrangement occurs. For further reading refer to the references given in the introduction of this users guide.

7.3 Example 3: Nickel using multiple data sets

In this section a combined refinement of two data sets for Ni will be shown. The first data set was already used in the Ni example shown in chapter 2 with a value of $Q_{max} = 22\text{Å}^{-1}$. The second data set is taken from the Billinge group (Michigan State University) diffractometer using a conventional X-ray tube as source. Using MoKα radiation a value of $Q_{max} = 16\text{Å}^{-1}$ could be reached. The refinement macro file is listed below:

```
reset
read stru,ni.stru
read data,x,22.0,0.0,ni-nsls.data
read data,x,16.0,0.0,ni-inhs.data
#
range 1,0.5,13.0
```

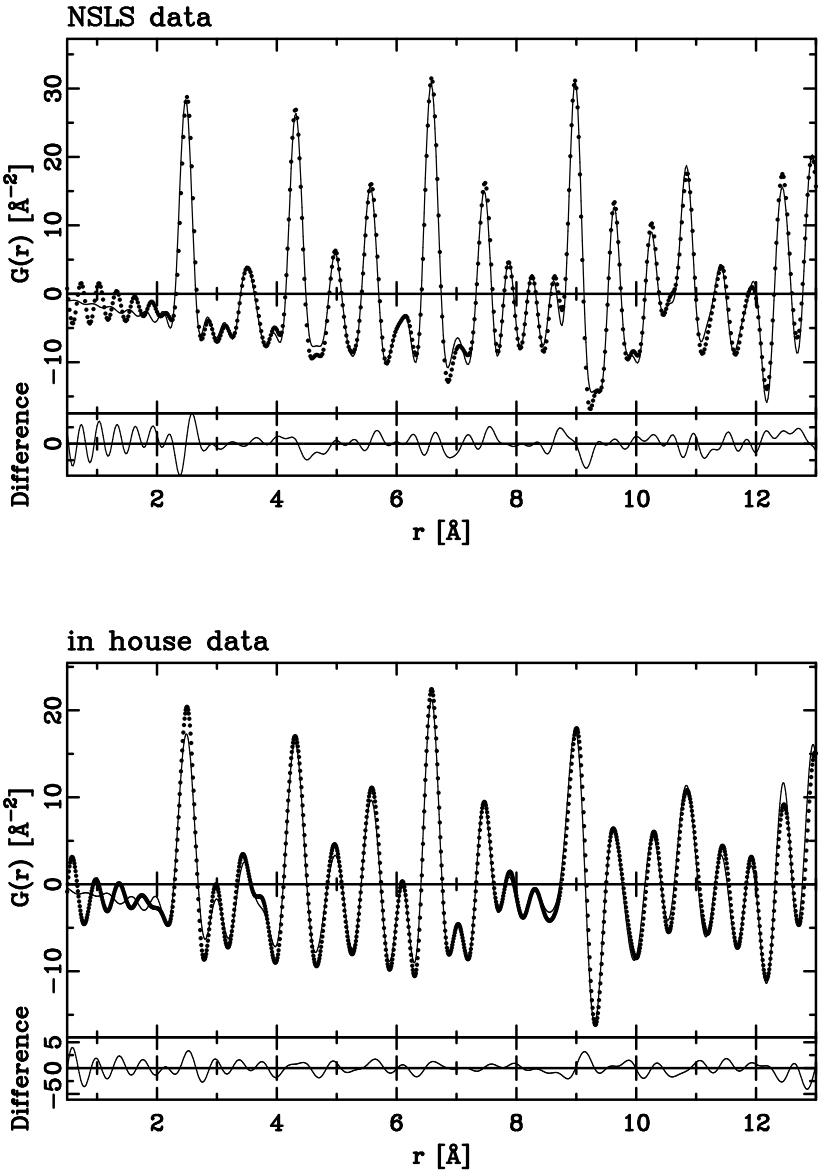

Figure 7.4: Result of PDF refinement of two data sets of Ni. The solid line is the calculated PDF, the filled circles are the data. The top panel shows synchrotron (NSLS) data, the bottom panel shows data from a in house X-ray source. A difference curve is plotted below the data.

```
 7 range 2,0.5,13.0
 8 #
 9 par dsca[1]=p[10],1.0
10 par dsca[2]=p[11],1.0
11 par qsig[1]=p[12],1.0
12 par qsig[2]=p[13],1.0
```

```
13 #
14 par delt[1]=p[14],1.0
15 #
16 p[10]=1.50
17 p[11]=1.00
18 p[12]=0.03
19 p[13]=0.03
20 p[14]=0.10
```

The first two lines are as before, resetting *PDFFIT* and reading the structure file. Now we need to issue a `read` command for each data set. Note the different file names and values for Q_{max} in lines 3 and 4. There are two data set dependent parameters, the scaling factor f_s and the resolution factor $\sigma_Q(s)$. One refinement parameter is assigned to each one for each of the two data sets (lines 9–12). Note that in previous examples we have used the scale factor f_p corresponding to the structural phase p. In cases where we have only *one* phase and *one* data set we are free to choose which scale factor to refine. Now the scale factors related to the different data sets must be used. The starting values for the different defined refinement parameters are set in lines 16–20.

```
21 #
22 ############################################################
23 # Refining lattice parameters and thermal factors
24 ############################################################
25 #
26 par lat[1]=p[1],1.0
27 par lat[2]=p[1],1.0
28 par lat[3]=p[1],1.0
29 #
30 do i[1]=1,n[1]
31   par u[1,i[1]]=p[2],1.0
32   par u[2,i[1]]=p[2],1.0
33   par u[3,i[1]]=p[2],1.0
34 enddo
35 #
36 p[1]=lat[1]
37 p[2]=u[1,1]
38 #
39 ############################################################
40 # Running the refinement and saving results
41 ############################################################
42 #
43 run
44 #
45 save pdf,1,ni-nsls.calc
46 save pdf,2,ni-inhs.calc
47 save dif,1,ni-nsls.dif
48 save dif,2,ni-inhs.dif
49 save stru,1,ni_ref.stru
```

```
50 save res,ni_ref.out
```

In this last part of the macro, we define refinement parameters for the lattice parameters $a = b = c$ and the isotropic thermal factor for the Ni atom. This is identical to the quick example given in chapter 2, the only difference is that one needs to save the calculated PDF and the difference between observed and calculated PDF for each data set separately (lines 45–48). The results for the synchrotron as well as the in-house data can be seen in Figure 7.4. Note that *PDFFIT* outputs only the total R-value of the refinement. Individual R-values for the different data sets or for a specific region in r can be obtained using the *PDFFIT* function `rval` (see section 5.4).

Chapter 8

Calculating structural properties

PDFFIT offers like *DISCUS* the intrinsic functions `blen` and `bang` to calculate the bond length and bond angles between atoms or general positions within the model crystal. However, *PDFFIT* allows the user to calculated bond length and bond angles with standard deviations propagated from the uncertainties of lattice parameters and positions obtained during the refinement

8.1 Calculating single bond lengths and angles

The bond length between any two atoms within the structure can simply be calculated using the command `blen` followed by the two atom numbers. The following example calculates the distance between the atoms `5` and `9`:

```
pdffit > blen 5,9
    MN (#    5) -    O (#    9)   =   1.965(5) A
```

The names of the two selected atoms and distance is given. If the standard deviations of the lattice parameters or the positions of the atoms are not zero, the standard deviation of the bond length is given as well. As usual the standard deviation is given in round brackets after the last significant digit. In our example the bond length is 1.965 ± 0.005Å. The bond angle defined by three atoms can be calculated in a similar way using the command `bang` as demonstrated below:

```
pdffit > bang 9,5,13
    O (#    9) -   MN (#    5) -    O (#   13)   =   74.1(9) degrees
```

It should be noted, that these functions calculates the direct distance between the given atoms, although the same atom might be closer to a given atom in a neighbouring unit cell. No periodic boundary conditions are applied. In order to obtain a specific bond length or angle it might be necessary to modify the atom coordinates by ± 1. This can simply be done using the corresponding variables, e.g. `x[1]=x[1]+1.0`. Note that this is actually modifying the current structure. Alternatively one can compute all bond length in a given interval as discussed in the next section.

8.2 Calculating multiple bond lengths

The command `blen` allows one to calculate all bond length between selected atom types in a given interval. This can be useful to monitor a specific bond length. In the following example all $Mn - O$ bonds within a range of 1.8 and 2.4Å are calculated.

```
pdffit > blen mn,o,1.8,2.4
 Bond lengths found in current phase :
   MN (#    5) -    O (#    9)   =          1.965(5) A
   MN (#    5) -    O (#   13)   =           1.93(1) A
   MN (#    5) -    O (#   18)   =           2.15(1) A
   MN (#    5) -    O (#   14)   =           2.15(1) A
   MN (#    5) -    O (#   17)   =           1.93(1) A
   MN (#    5) -    O (#   11)   =          1.965(5) A

   MN (#    6) -    O (#   13)   =           2.15(1) A
   MN (#    6) -    O (#   12)   =          1.965(5) A
   MN (#    6) -    O (#   18)   =           1.93(1) A
   MN (#    6) -    O (#   10)   =          1.965(5) A
   MN (#    6) -    O (#   14)   =           1.93(1) A
   MN (#    6) -    O (#   17)   =           2.15(1) A

   MN (#    7) -    O (#    9)   =          1.965(5) A
   MN (#    7) -    O (#   15)   =           1.93(1) A
   MN (#    7) -    O (#   20)   =           2.15(1) A
   MN (#    7) -    O (#   11)   =          1.965(5) A
   MN (#    7) -    O (#   16)   =           2.15(1) A
   MN (#    7) -    O (#   19)   =           1.93(1) A

   MN (#    8) -    O (#   10)   =          1.965(5) A
   MN (#    8) -    O (#   19)   =           2.15(1) A
   MN (#    8) -    O (#   20)   =           1.93(1) A
   MN (#    8) -    O (#   12)   =          1.965(5) A
   MN (#    8) -    O (#   15)   =           2.15(1) A
   MN (#    8) -    O (#   16)   =           1.93(1) A
```

As one can see around each of the four Mn in the unit cell we find six oxygens forming an octahedra. In this mode of the command `blen` periodic boundary conditions are applied in the same way as for the calculation of the PDF and as a result distances to atoms outside the structural box (or unit cell) are also found.

The command `blen` might also be used to identify which atom pairs contribute to a given PDF peak. Assume the peak in question is between 3.5 and 3.6Å. Simply enter the command `blen all,all,3.5,3.6` and *PDFFIT* will list all atom pairs that are separated by a distance between 3.5 and 3.6Å.

Appendix A

Computational Details

Here we give the details of the computation of the model PDF $G(r)$. In the introduction we defined $G(r)$ as follows:

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

As we discussed already in this manual, the sums go over all atoms within the model crystal and r_{ij} is the separation distance between atoms i and j. The value b_i is the scattering length for atom i and $\langle b \rangle$ is the average scattering length of the model crystal. Finally ρ_0 is the number density.

As we have seen in the introduction to this users guide, the experimental PDF is obtained by Fourier transform of the reduced structure factor. However, the accessible range in Q is limited by Q_{max}. This can be described by a multiplication of the structure factor up to infinity with a step function cutting off at $Q = Q_{max}$ resulting in the convolution of the PDF with the Fourier transform $S(r)$ of the step function. *PDFFIT* models the finite Q-range by convoluting the model PDF $G(r)$ with

$$S(r) = \frac{\sin(Q_{max} \cdot r)}{r} \tag{A.2}$$

In the following section we will omit this convolution and discuss how $G(r)$ and the partial derivatives with respect to the refinement parameters, $\partial G(r)/\partial p_n$, are actually calculated.

A.1 Calculating the PDF

The program *PDFFIT* refines the function $G(r)$ at discreet values of $r = r_k$ for an experimental dataset s. Thus our equation for the model PDF becomes:

$$G(r_k, s) = f_s B_k(s) \sum_{p=1}^{P} f_p G_p(r_k, s) \tag{A.3}$$

with

$$G_p(r_k, s) = \frac{1}{N_p r_k} \sum_i \sum_j [A_{ij}(p) \cdot T_{ij}(r_k, p)] - 4\pi r_k \rho_0(p) \quad \text{(A.4)}$$

$$B_k(s) = \exp\left[-\frac{(r_k \sigma_Q(s))^2}{2}\right] \quad \text{(A.5)}$$

$$A_{ij}(p) = \frac{c_i(p) c_j(p) b_i b_j}{\langle b \rangle^2} \quad \text{(A.6)}$$

$$T_{ij}(r_k, p) = \frac{1}{\sqrt{2\pi}\sigma_{ij}(p)} \exp\left[-\frac{(r_k - r_{ij}(p))^2}{2\sigma_{ij}^2(p)}\right] \left[1 + \left(\frac{r_k - r_{ij}}{r_{ij}}\right)\right] \quad \text{(A.7)}$$

where f_s stands for the overall scale factor and $B_k(s)$ is the experimental resolution factor for dataset s. The first sum in (A.3) is over the different structural phases p in a multi phase refinement. The relative abundance of each phase p is given by f_p. $G_p(r_k, s)$ is the model PDF for a single phase p given in (A.5). These values are actually stored in an array after the PDF is calculated. The indices i and j sum over all atoms within the structural phase p or in other words we sum over all atom *pairs* in that phase. The contribution of each pair of atoms i and j is weighted with a factor $A_{ij}(p)$ which depends on the scattering length b_i and concentration c_i of both atoms. The average scattering length is $\langle b \rangle = \sum_i c_i b_i / N_p$. The average number density is given by $\rho_0(p) = N_p / V_p$ where $N_p = \sum_i c_i$ is the number of atoms within phase p and V_p is the volume of the unit cell in Å^3. The calculation of V_p is presented later in this section.

The final term in (A.3) we need to discuss is $T_{ij}(r_k, p)$. 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 the program *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 (A.1) with a modified Gaussian $T_{ij}(r_k, p)$ accounting for the displacements. The deviation from a Gaussian shape is caused by anisotropic averaging and discussed in detail in Thorpe et al. (2002). The width of the function T_{ij} is given by the anisotropic thermal factors $U_{lm} = \langle u_l u_m \rangle$ of atoms i and j. Furthermore the $\sigma_{ij}(p)$ shows an r-dependence given in (A.8). Note that this definition has been extended since version 1.2 of PDFFIT. In some cases an additional sharpening of the PDF peaks below a user defined cutoff value, r_{cut} needs to be introduced (see 6.1.2 for details). This is done in (A.9).

$$\sigma_{ij}(p) = \phi(r_{ij}, p) \cdot \sqrt{\sigma'^2_{ij}(p) - \frac{\delta(p)}{r_{ij}^2(p)} - \frac{\gamma(p)}{r_{ij}(p)} + \alpha^2(s) r_{ij}^2(p)} \quad \text{(A.8)}$$

with

$$\phi(r_{ij}, p) = \begin{cases} \phi_0(p) & \text{for } r_{ij} < r_{cut} \\ 1.0 & \text{otherwise} \end{cases} \quad \text{(A.9)}$$

and

$$\sigma'_{ij}(p) = \frac{1}{|r_{ij}|}\sqrt{\sum_{l,m=1}^{3}\left[r^l_{ij}(p)r^m_{ij}(p)(U_{lm}(i)+U_{lm}(j))\right]} \tag{A.10}$$

with r^l_{ij} standing for the l-th component of the difference vector between atoms i and j. Note that so far r_{ij} referred to a scalar, the distance between the atoms without any directional information.

The last piece of the puzzle is the relation between $r_{ij}(p)$ and V_p and the lattice parameters a, b, c, α, β and γ. *PDFFIT* works with fractional coordinates, i.e. in units of the unit cell. However, to calculate the distance between two atoms, the difference vector must be converted to Å. A helpful concept when dealing with non-orthogonal coordinate systems is the so-called *metric tensor*, g_{ij}, which is defined as

$$g_{ij} = \mathbf{a}_i \cdot \mathbf{a}_j = \begin{pmatrix} a^2 & ab\cos\gamma & ac\cos\beta \\ ab\cos\gamma & b^2 & bc\cos\alpha \\ ac\cos\beta & bc\cos\alpha & c^2 \end{pmatrix} \tag{A.11}$$

with $\mathbf{a_i}$ describing the basis vectors of the crystal lattice in an orthonormal system. Using this metric tensor, the scalar product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is simply $\mathbf{uv} = \sum_{ij} u_i v_j g_{ij}$ or $V_p = abc[1 - \cos^2\alpha - \cos^2\beta - \cos^2\gamma + 2\cos\alpha\cos\beta\cos\gamma]^{\frac{1}{2}}$ which is simply the determinant of the metric tensor g_{ij}.

Let us assume we want to calculate the distance between the atoms i and j on positions $\mathbf{u}_i$ and $\mathbf{u}_j$ in the crystal system (We will omit the reference to phase p is this part). Now the difference vector still in fractional coordinates is simply $\mathbf{d}_{ij} = \mathbf{u}_j - \mathbf{u}_i$. In order to get the components r^l_{ij} needed to calculate σ'_{ij} in (A.10), we simply multiply with the corresponding lattice parameter:

$$r^1_{ij} = ad^1_{ij},\ r^2_{ij} = bd^2_{ij},\ r^3_{ij} = cd^3_{ij} \tag{A.12}$$

The length of the difference vector in Å is given by $r_{ij} = [\mathbf{d}_{ij} \cdot \mathbf{d}_{ij}]^{\frac{1}{2}}$ which can be written as follows using (A.11):

$$\begin{aligned} r_{ij} \quad = \quad & [a^2 d^2_{ij_x} + b^2 d^2_{ij_y} + c^2 d^2_{ij_z} + \\ & 2ab\cos\gamma d_{ij_x} d_{ij_y} + 2ac\cos\beta d_{ij_x} d_{ij_z} + 2bc\cos\alpha d_{ij_y} d_{ij_z}]^{\frac{1}{2}} \end{aligned} \tag{A.13}$$

Now we have all necessary relationships to calculate $G(r_k, s)$ from the experimental and structural parameters we might want to refine. The needed analytical derivatives are given in the next section and the actual parameter names used by *PDFFIT* are listed in the last section of this appendix.

A.2 Partial derivatives

It should be noted that the derivatives given in this section actually need to be convoluted with the function given in (A.2) as well since $\partial(G(r,p) * S(r))/\partial p = (\partial G(r,p)/\partial p) * S(r)$.

Scale factors

The derivatives with respect to the two scale factors f_s and f_p are straight forward and listed below:

$$\frac{\partial G(r_k,s)}{\partial f_s} = B_k(s)\sum_{p=1}^{P} f_p G_p(r_k,s) \tag{A.14}$$

$$\frac{\partial G(r_k,s)}{\partial f_p} = f_s B_k(s) G_p(r_k,s) \tag{A.15}$$

Experimental resolution factor

The experimental resolution σ_Q only appears in $B_k(s)$, thus the derivative is simply:

$$\frac{\partial G(r_k,s)}{\partial \sigma_Q(s)} = -r_k^2\sigma_Q(s)\ f_s B_k(s)\sum_{p=1}^{p} f_p G_p(r_k,s) \tag{A.16}$$

Quadratic dynamic correlation factor

The dynamic correlation factor $\delta(p)$ appears only in $T_{ij}(r_k,p)$ where $\sigma_{ij}(p)$ appears. Thus we can rewrite the derivative for a given phase p (we omit the reference to p in the following equations) as follows:

$$\frac{\partial G(r_k,s)}{\partial \delta} = \frac{f_s f_p B_k(s)}{N_p r_k}\sum_i\sum_j A_{ij}\frac{\partial T_{ij}(r_k)}{\partial \delta} \tag{A.17}$$

with

$$\begin{aligned}\frac{\partial T_{ij}(r_k)}{\partial \delta} &= \frac{\partial T_{ij}(r_k)}{\partial \sigma_{ij}}\frac{\partial \sigma_{ij}}{\partial \delta} \\ &= \frac{T_{ij}(r_k)}{\sigma_{ij}}\left[\frac{(r_k-r_{ij})^2}{\sigma_{ij}^2}-1\right]\cdot\frac{-\phi^2(r_{ij})}{2r_{ij}^2\sigma_{ij}}\end{aligned} \tag{A.18}$$

which gives the final result for the derivative of

$$\frac{\partial G(r_k,s)}{\partial \delta} = -\frac{f_s f_p B_k(s)}{N_p r_k}\sum_i\sum_j \frac{\phi^2(r_{ij})A_{ij}(p)T_{ij}(r_k)}{2r_{ij}^2\sigma_{ij}^2}\left[\frac{(r_k-r_{ij})^2}{\sigma_{ij}^2}-1\right] \tag{A.19}$$

Linear dynamic correlation factor

The linear correlation factor $\gamma(p)$ appears only where σ_{ij} appears. Similar to (A.17) we find using

$$\frac{\partial \sigma_{ij}}{\partial \gamma} = \frac{-\phi^2(r_{ij})}{2r_{ij}\sigma_{ij}} \tag{A.20}$$

which gives the resulting expression of

$$\frac{\partial G(r_k,s)}{\partial \gamma} = -\frac{f_s f_p B_k(s)}{N_p r_k} \sum_i \sum_j \frac{\phi^2(r_{ij}) A_{ij}(p) T_{ij}(r_k)}{2 r_{ij} \sigma_{ij}^2} \left[\frac{(r_k - r_{ij})^2}{\sigma_{ij}^2} - 1 \right] \tag{A.21}$$

Resolution broadening factor

The resolution broadening $\alpha(s)$ appears only where σ_{ij} appears. We omit the reference to s in the following equations for clarity. Similar to (A.17) we find using

$$\frac{\partial \sigma_{ij}}{\partial \alpha} = \frac{\alpha \phi^2(r_{ij}) r_{ij}^2}{\sigma_{ij}} \tag{A.22}$$

which gives the resulting expression of

$$\frac{\partial G(r_k,s)}{\partial \gamma} = \frac{f_s f_p B_k(s)}{N_p r_k} \sum_i \sum_j \frac{\alpha \phi^2(r_{ij}) r_{ij}^2 A_{ij}(p) T_{ij}(r_k)}{\sigma_{ij}^2} \left[\frac{(r_k - r_{ij})^2}{\sigma_{ij}^2} - 1 \right] \tag{A.23}$$

Peak with ratio

The parameter ϕ_0 describing the additional sharpening of the PDF peaks below r_{cut} appears only where σ_{ij} appears. For values of r above r_{cut}, the derivative is zero since no term depends on ϕ_0. For values of r below r_{cut}, we can write similar to (A.17)

$$\frac{\partial G(r_k,s)}{\partial \phi_0} = \frac{f_s f_p B_k(s)}{N_p r_k} \sum_i \sum_j A_{ij} \frac{\partial T_{ij}(r_k)}{\partial \phi_0} \tag{A.24}$$

with

$$\begin{aligned} \frac{\partial T_{ij}(r_k)}{\partial \phi_0} &= \frac{\partial T_{ij}(r_k)}{\partial \sigma_{ij}} \frac{\partial \sigma_{ij}}{\partial \phi_0} \\ &= \frac{T_{ij}(r_k)}{\sigma_{ij}} \left[\frac{(r_k - r_{ij})^2}{\sigma_{ij}^2} - 1 \right] \cdot \frac{\sigma_{ij}}{\phi_0}. \end{aligned} \tag{A.25}$$

Lattice parameters

The lattice parameters a_n (for now symbolic for a, b, c, α, β and γ) appear in $r_{ij}(p), \sigma_{ij}(p)$ and the unit cell volume V_p. Thus only $T_{ij}(r_k, p)$ and $\rho_0(p)$ depend on these parameters. Again we omit the reference to the phase p in the equations below. Let us consider the derivative of T_{ij} first.

$$\frac{\partial T_{ij}(r_k)}{\partial a_n} = \frac{\partial T_{ij}(r_k)}{\partial \sigma_{ij}} \frac{\partial \sigma_{ij}}{\partial a_n} + \frac{\partial T_{ij}(r_k)}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial a_n}$$

$$= \frac{T_{ij}(r_k)}{\sigma_{ij}} \left\{ \left[\frac{(r_k - r_{ij})^2}{\sigma_{ij}^2} - 1 \right] \frac{\partial \sigma_{ij}}{\partial a_n} + \left[\frac{r_k - r_{ij}}{\sigma_{ij}} - \frac{\sigma_{ij} r_k}{r_{ij}^2} \right] \frac{\partial r_{ij}}{\partial a_n} \right\} \quad \text{(A.26)}$$

Using Equation (A.8) we find for the derivative of σ_{ij} with respect to the lattice parameters:

$$\begin{aligned} \frac{\partial \sigma_{ij}}{\partial a_n} &= \frac{\partial \sigma_{ij}}{\partial \sigma'_{ij}} \frac{\partial \sigma'_{ij}}{\partial a_n} + \frac{\partial \sigma_{ij}}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial a_n} \\ &= \frac{\phi^2(r_{ij})}{2\sigma_{ij}} \left[\frac{\partial \sigma'_{ij}}{\partial a_n} \cdot 2\sigma'_{ij} + \frac{\partial r_{ij}}{\partial a_n} \left[2\alpha^2 r_{ij} + \frac{2\delta}{r_{ij}^3} + \frac{\gamma}{r_{ij}^2} \right] \right] \end{aligned} \quad \text{(A.27)}$$

The definition of σ_{ij} (A.10) shows that the lattice parameters appear in the distance r_{ij} as well as the components of the difference vector r_{ij}^l (A.12). Using these relations we get

$$\begin{aligned} \frac{\partial \sigma'_{ij}}{\partial a_n} &= \frac{\partial \sigma'_{ij}}{\partial r_{ij}^l} \frac{\partial r_{ij}^l}{\partial a_n} + \frac{\partial \sigma'_{ij}}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial a_n} \\ &= \frac{1}{r_{ij}^2 \sigma'_{ij}} \left[\sum_{m=1}^{3} a_m d_{ij}^n d_{ij}^m \left(U_{nm}(i) + U_{nm}(j) \right) \right] - \frac{\sigma'_{ij}}{r_{ij}} \frac{\partial r_{ij}}{\partial a_n} \end{aligned} \quad \text{(A.28)}$$

The first term in (A.28) is zero for $n = 4, 5, 6$, in other words for the derivatives with respect to the cell angles. Next we show the derivative of the distance r_{ij} with respect to the lattice parameters. Note that we now use a, b, c, α, β and γ explicitly rather than a_n as before.

$$\frac{\partial r_{ij}}{\partial a} = \frac{1}{r_{ij}} (a d_{ij_x}^2 + b \cos\gamma d_{ij_x} d_{ij_y} + c \cos\beta d_{ij_x} d_{ij_z}) \quad \text{(A.29)}$$

$$\frac{\partial r_{ij}}{\partial b} = \frac{1}{r_{ij}} (b d_{ij_y}^2 + a \cos\gamma d_{ij_x} d_{ij_y} + c \cos\alpha d_{ij_y} d_{ij_z}) \quad \text{(A.30)}$$

$$\frac{\partial r_{ij}}{\partial c} = \frac{1}{r_{ij}} (c d_{ij_z}^2 + a \cos\beta d_{ij_x} d_{ij_z} + b \cos\alpha d_{ij_y} d_{ij_z}) \quad \text{(A.31)}$$

$$\frac{\partial r_{ij}}{\partial \alpha} = -\frac{bc}{r_{ij}} (\sin\alpha d_{ij_y} d_{ij_z}) \quad \text{(A.32)}$$

$$\frac{\partial r_{ij}}{\partial \beta} = -\frac{ac}{r_{ij}} (\sin\beta d_{ij_x} d_{ij_z}) \quad \text{(A.33)}$$

$$\frac{\partial r_{ij}}{\partial \gamma} = -\frac{ab}{r_{ij}} (\sin\gamma d_{ij_x} d_{ij_y}) \quad \text{(A.34)}$$

Finally we need see who the number density $\rho_0 = N_p / V_p$ depends on the lattice parameters. Considering the definition of the unit cell volume, we get

$$\frac{\partial \rho_0}{\partial a_n} = -\frac{N_p}{V_p^2}\frac{\partial V_p}{\partial a_n} = \begin{cases} -\frac{\rho_0}{a_n} & \text{for } a_n = a,b,c \\ -\frac{\rho_0}{V_p^2} a^2b^2c^2(\sin\alpha\cos\alpha - \sin\alpha\cos\beta\cos\gamma) & \text{for } a_n = \alpha \\ -\frac{\rho_0}{V_p^2} a^2b^2c^2(\sin\beta\cos\beta - \sin\beta\cos\alpha\cos\gamma) & \text{for } a_n = \beta \\ -\frac{\rho_0}{V_p^2} a^2b^2c^2(\sin\gamma\cos\gamma - \sin\gamma\cos\alpha\cos\beta) & \text{for } a_n = \gamma \end{cases} \tag{A.35}$$

Using all equations given in this section, the desired derivative of the calculated PDF $G_p(r_k, s)$ with respect to the lattice parameters can be calculated using

$$\frac{\partial G_p(r_k,s)}{\partial a_n} = f_p f_s B_k(s) \left\{ \frac{1}{N_p r_k} \sum_i \sum_j \left[A_{ij} \frac{\partial T_{ij}(r_k)}{\partial a_n} \right] - 4\pi r_k \frac{\partial \rho_0}{\partial a_n} \right\} \tag{A.36}$$

Thermal parameters

Only the PDF peak width given by σ_{ij} is a function of the thermal parameters $U_{lm}(i)$ and σ_{ij} appears only in $T_{ij}(r_k)$. Thus with some of the work already done in the previous section, the desired derivative can be written as

$$\frac{\partial G(r_k,s)}{\partial U_{lm}(i)} = \frac{f_p f_s B_k(s)}{N_p r_k} \sum_i \sum_j \left[A_{ij}(p) \frac{\partial T_{ij}(r_k)}{\partial \sigma_{ij}(p)} \frac{\partial \sigma_{ij}(p)}{\partial \sigma'_{ij}(p)} \frac{\partial \sigma'_{ij}(p)}{\partial U_{lm}(i)} \right] \tag{A.37}$$

With $\partial T_{ij}/\partial \sigma_{ij}$ already calculated in (A.26) and $\partial \sigma_{ij}/\partial \sigma'_{ij} = \phi^2(r_{ij})/2\sigma_{ij}$ we only need

$$\frac{\partial \sigma'_{ij}}{\partial U_{lm}(i)} = \frac{1}{r_{ij}^2 \sigma'_{ij}} r_{ij}^l r_{ij}^m \tag{A.38}$$

Atomic positions

The atomic positions $\mathbf{v} = (v_1, v_2, v_3)$ of one specific atom i in phase p appear only in $T_{ij}(r_k)$ in form of σ_{ij} and r_{ij}. We can simply substitute a_n in equations (A.26) and (A.27) with the fractional coordinate and get

$$\begin{aligned} \frac{\partial T_{ij}(r_k)}{\partial v_n(i)} &= \frac{\partial T_{ij}(r_k)}{\partial \sigma_{ij}} \frac{\partial \sigma_{ij}}{\partial v_n(i)} + \frac{\partial T_{ij}(r_k)}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial v_n(i)} \\ &= \frac{T_{ij}(r_k)}{\sigma_{ij}} \left\{ \left[\frac{(r_k - r_{ij})^2}{\sigma_{ij}^2} - 1 \right] \frac{\partial \sigma_{ij}}{\partial v_n(i)} + \left[\frac{r_k - r_{ij}}{\sigma_{ij}} - \frac{\sigma_{ij} r_k}{r_{ij}^2} \right] \frac{\partial r_{ij}}{\partial v_n(i)} \right\} \end{aligned} \tag{A.39}$$

$$\frac{\partial \sigma_{ij}}{\partial v_n(i)} = \frac{\phi^2(r_{ij})}{2\sigma_{ij}} \left[\frac{\partial \sigma'_{ij}}{\partial v_n(i)} \cdot 2\sigma'_{ij} + \frac{\partial r_{ij}}{\partial v_n(i)} \left[2\alpha^2 r_{ij} + \frac{2\delta}{r_{ij}^3} + \frac{\gamma}{r_{ij}^2} \right] \right] \tag{A.40}$$

with

$$\frac{\partial \sigma'_{ij}}{\partial v_n(i)} = \frac{1}{r_{ij}^2 \sigma'_{ij}} \left[\sum_{m=1}^{3} a_n a_m d_{ij}^m (U_{nm}(i) + U_{nm}(j)) \right] - \frac{\sigma'_{ij}}{r_{ij}} \frac{\partial r_{ij}}{\partial v_n(i)} \tag{A.41}$$

and

$$\frac{\partial r_{ij}}{\partial v_1(i)} = \frac{\partial r_{ij}}{\partial d_{ij_1}} \frac{\partial d_{ij_1}}{\partial v_1(i)} = -\frac{1}{r_{ij}} (a^2 d_{ij_1} + ab\cos\gamma d_{ij_2} + ac\cos\beta d_{ij_3}) \tag{A.42}$$

$$\frac{\partial r_{ij}}{\partial v_2(i)} = -\frac{1}{r_{ij}} (b^2 d_{ij_2} + ab\cos\gamma d_{ij_1} + bc\cos\alpha d_{ij_3}) \tag{A.43}$$

$$\frac{\partial r_{ij}}{\partial v_3(i)} = -\frac{1}{r_{ij}} (c^2 d_{ij_3} + ac\cos\beta d_{ij_1} + bc\cos\alpha d_{ij_2}) \tag{A.44}$$

Note that $\partial T_{ij}(r_k)/\partial v_n(i) = -\partial T_{ij}(r_k)/\partial v_n(j)$ because $d_{ij} = -d_{ji}$.

Site occupancies

The site occupancies c_n for a given phase p appear in the terms A_{ij}, N_p and ρ_0. Thus the derivative is

$$\begin{aligned} \frac{\partial G_p(r_k, s)}{f_p \partial c_n} &= \frac{\partial G_p}{\partial N_p}\frac{\partial N_p}{\partial c_n} + \frac{\partial G_p}{\partial A_{ij}}\frac{\partial A_{ij}}{\partial c_n} + \frac{\partial G_p}{\partial \rho_0}\frac{\partial \rho_0}{\partial c_n} \\ & \frac{1}{N_p r_k} \left[\frac{1}{c_n} \sum_j A_{nj} T_{nj}(r_k) - \frac{1}{N_p} \sum_i \sum_j A_{ij} T_{ij}(r_k) \right] - \frac{4\pi r_k}{V_p} \end{aligned} \tag{A.45}$$

A.3 Calculating standard deviations

The derivatives presented in the last section were with respect to the experimental and structural parameters, which we will refer to as x_i in this section. *PDFFIT* allows the user to freely define their relationship to the refinement parameters p_i. However, for the least square refinement the derivatives with respect to those refinement parameters are needed. They can be obtained using the following relationship

$$\frac{\partial G(r_k, s)}{\partial p_i} = \sum_{j=1}^{N} \frac{\partial G(r_k, s)}{\partial x_j} \frac{\partial x_j}{\partial p_i} \tag{A.46}$$

The sum goes over the number of parameters x. The derivatives $\partial x_j / \partial p_i$ must be specified analytically by the user using the command `par`. These derivatives are also used to calculate the resulting standard deviations, Δx_i, of the parameters x_i from the errors, Δp_i of the refinement parameters using

$$\Delta x_i = \sqrt{\sum_n \left(\frac{\partial x_i}{\partial p_n} \Delta p_n \right)^2} \tag{A.47}$$

The sum goes over all refinement parameters p_i, however in practice each structural parameter x_i is only allowed to depend on a small number of refinement parameters, so only a few derivatives contribute to the sum.

Finally we will give the definitions of the R-values calculated in each cycle of the refinement. The expected R-value is defined as

$$R_{exp} = \sqrt{\frac{N-p}{\sum_{i=1}^{N} w(r_i) G_{obs}^2(r_i)}} \tag{A.48}$$

with G_{obs} being the experimental PDF, N the number of data points and p the number of (refined) parameters. The weight for each data point is given by $w(r_i)$. The weighted R-value is calculated as follows

$$R_w = \sqrt{\frac{\sum_{i=1}^{N} w(r_i)[G_{obs}(r_i) - G_{calc}(r_i)]^2}{\sum_{i=1}^{N} w(r_i) G_{obs}^2(r_i)}} \tag{A.49}$$

Here G_{calc} is the calculated PDF. The unweighted R-value is simply defined as in (A.49) just with the weights $w(r_i)$ set to unity.

Appendix B

List of commands

In this appendix, a list of all current *PDFFIT* commands is given. Note that commands marked with an asterix '*' branch to a sublevel of *PDFFIT* 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.

B.1 Alphabetical list of commands

Command	**Description**
#	Comment, the rest of the line will be ignored
@	Execution of a macrofile
alloc	Allocates space for calculating PDFs without reading data
bang	Calculates the bond angle and standard deviation
blen	Calculates bond length(s) and standard deviation
break	Interrupts a loop or conditional statement
calc	Calculates PDF from current structure
continue	Continues *PDFFIT* after 'stop' command
cycle	Sets the maximum number of refinement cycles
do	Start of a do loop
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
exit	Ends the *PDFFIT* program
fclose	Close file opened with 'fopen'
fget	Read information from file
fopen	Open file for 'fget' or 'fput'

fput	Write information to file
help	Gives on line help
if	Begin of an if construction
i/jdese	Selects atoms used for calculating the PDF
i/jsele	Deselects atoms used for calculating the PDF
learn	Starts a learn sequence
lend	Ends a learn sequence
occ	Sets occupancy for a given site
par	Sets a parameter definition
phase	Switches active structural phase
psel	Defines association between phases and data sets
range	Sets r-range used for refinement
read	Reads structure and PDF data files
reset	Reset *PDFFIT*
run	Starts a refinement
save	Saves resulting structure, PDF data and fit results
scat	Overwrites internal scattering factor for given element
show	Shows various *PDFFIT* settings
stop	Stops *PDFFIT* macro, resume with 'cont'
system	Executes a shell command
temp	Sets anisotropic temperature factor
urf	Sets URF (magic refinement number)
wait	Wait for user input
xray	Specifies Q-value for calculating X-ray form factors

B.2 Functional list of commands

PDFFIT related commands

Command	**Description**
alloc	Allocates space for calculating PDFs without reading data
bang	Calculates the bond angle and standard deviation
blen	Calculates bond length(s) and standard deviation
calc	Calculates PDF from current structure
cycle	Sets the maximum number of refinement cycles
i/jdese	Selects atoms used for calculating the PDF
i/jsele	Deselects atoms used for calculating the PDF
occ	Sets occupancy for a given site
par	Sets a parameter definition

phase	Switches active structural phase
psel	Defines association between phases and data sets
range	Sets r-range used for refinement
read	Reads structure and PDF data files
run	Starts a refinement
save	Saves resulting structure, PDF data and fit results
scat	Overwrites internal scattering factor for given element
show	Shows various *PDFFIT* settings
temp	Sets anisotropic temperature factor
urf	Sets URF (magic refinement number)
xray	Specifies Q-value for calculating X-ray form factors

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 *PDFFIT* 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 *PDFFIT* program
fclose	Close file opened with 'fopen'
fget	Read information from file
fopen	Open file for 'fget' or 'fput'
fput	Write information to file
if	Begin of an if construction
learn	Starts a learn sequence
lend	Ends a learn sequence
reset	Reset *PDFFIT*
stop	Stops *PDFFIT* macro, resume with 'cont'
system	Executes a shell command
wait	Wait for user input

Appendix C

Installation

In this appendix we will briefly describe the installation process of the program *PDFFIT*. *PDFFIT* is part of the *Diffuse* package which includes the programs *DISCUS* and *KUPLOT* as well as *PDFFIT*. Refer to the section corresponding to your operating system for installation information.

Windows

The Windows version of the *Diffuse* is distributed as a self-extracting installer. Simply download the file `Diffuse-4.1-win32.exe` (or the most current version) and run it by double clicking on the downloaded file. This will start the installation process and the dialog shown in Figure C.1 will appear. Simply follow the instructions on the screen and that is all, you are ready to use and of the programs that are part of the *Diffuse* package.

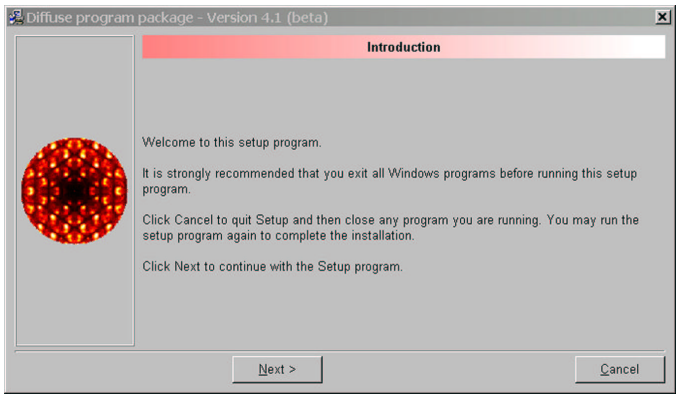

Figure C.1: Windows installer for *Diffuse*.

Unix / Linux

First one needs to obtain the complete 'diffuse' archive named `Diffuse-4.1.tar.gz` or the most recent version. Next you need to unpack the archive using the commands

```
gzip -d Diffuse-4.1.tar.gz
tar -xvof Diffuse-4.1.tar
```

This will create a directory 'diffuse' containing the distribution. Within this directory there are separate directories `pdffit`, `kuplot` and `pdffit` 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
```

After proceeding to the `pdffit/prog` directory you need to choose a `Makefile` that suits your needs and customize the `Makefile` as well as one configuration file '`config.inc`'. Next copy the UNIX style Makefile using the command

```
cp Makefile.unix Makefile
```

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 '`pdffit`' and '`pdffit.hlp`' in the same directory.

Before you actually can use the online help of the program, an environment variable `PDFFIT` 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 PDFFIT /path/to/pdffit` for the `csh` of `set PDFFIT=/path/to/pdffit; export PDFFIT` 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 `pdffit`. The program *PDFFIT* 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 *Diffuse* 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

S. J. L. Billinge. Real-Space Rietveld: Full Profile Structural Refinement of the Atomic Pair Distribution Function. In Billinge and Thorpe (1998) page 137.

S. J. L. Billinge; M. F. Thorpe, editors. *Local Structure from Diffraction* New York 1998. Plenum.

S.J.L. Billinge; 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** (1993) 14386–14406.

E. Dowty. ATOMS for Windows and Macintosh. Shape Software 1997. WWW: http://www.shapesoftware.com/.

T. Egami. PDF Analysis applied to Crystalline Materials. In Billinge and Thorpe (1998) page 1.

F. Frey. Diffuse Scattering from Disordered Crystals . Acta Cryst. B **51** (1995) 592–602.

I.-K. Jeong; R.H. Heffner; M.J. Graf; S.J.L. Billinge. Lattice dynamics and correlated motion from the atomic pair distribution function. cond-mat/0209603v1 (2002).

I.K. Jeong; Th. Proffen; F. Mohiuddin-Jacobs; S.J.L. Billinge. Measuring Correlated Atomic Motion using X-Ray Diffraction. J. Phys. Chem. B (1998). accepted.

V.M. Nield; D.A. Keen; R.L. McGreevy. The Interpretation of Single Crystal Diffuse Scattering using Reverse Monte Carlo Modelling . Acta Cryst. A **51** (1995) 763–771.

Th. Proffen; R.B. Neder. DISCUS, a Program for Diffuse Scattering and Defect Structure Simulations . J. Appl. Cryst. **30** (1997) 171.

Th. Proffen; T.R. Welberry. Analysis of Diffuse Scattering *via* Reverse Monte Carlo Technique: a Systematic Investigation . Acta Cryst. A **53** (1997) 202–216.

Th. Proffen; T.R. Welberry. Analysis of Diffuse Scattering from Single Crystals *via* Reverse Monte Carlo Technique: II. The Defect Structure of Calcium Stabilised Zirconia . J. Appl. Cryst. **31** (1998) 318–326.

H.M. Rietveld. A profile refinement method for nuclear and magnetic structures. J. Appl. Cryst. **2** (1969) 65–71.

J. Rodríguez-Carvajal; M. Hennion; F. Moussa; A.H. Moudden; L. Pinsard; A. Revcolevschi. Neutron-Diffraction Study of the Jahn-Teller Transition in stoichiometric $LaMnO_3$. Phys. Rev. B **57** (1998) R3189–R3192.

D.E. Sands. *Vectors and Tensors in Crystallography* . Dover Publications Inc. New York 1 edition 1995.

M.F. Thorpe; V.A. Levashov; M. Lei; S.J.L. Billinge. Notes on the Analysis of Data for Pair Distribution Functions. In S. J. L. Billinge; M. F. Thorpe, editors, *From Semiconductors to Proteins* page 105 New York 2002. Plenum.

B.H. Toby; T. Egami. Accuracy of Pair Distribution Function Analysis Applied to Crystalline and Non-Crystalline Materials. Acta Cryst. A **48** (1992) 336–346.

B.E. Warren. *X-ray Diffraction* . Dover Publications Inc. New York 1990.

Y. Waseda. *The Structure of Non-Crystalline Materials* . McGraw-Hill, New York 1 edition 1986.

T.R. Welberry; B.D. Butler. Diffuse X-ray Scattering from Disordered Crystals . Chem. Rev. **95** (1995) 2369–2403.

T.R. Welberry; Th. Proffen. Analysis of Diffuse Scattering from Single Crystals *via* Reverse Monte Carlo: I. Comparison with Direct Monte Carlo . J. Appl. Cryst. **31** (1998) 309–317.

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