REFINE

Users Guide

Version 6.03

written by

Reinhard Neder
Email: reinhard.neder@fau.de

http://tproffen.github.io/DiffuseCode

Document created: November 27, 2020

Preface

Disclaimer

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

Using REFINE

More information

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

> NEDER, R.B. & PROFFEN, TH. "Diffuse Scattering and Defect Structure Simulations - A cook book using the programs DISCUS", *IUCr Texts on Crystallography*, Oxford University Press, 2007.

Contents

Chapter 1

Introduction

1.1 What is REFINE ?

REFINE is the direct Least Squares Refinement section of the DISCUS SUITE program. Its internal engine is a Levenberg-Marquardt type least squares fit that can be applied to 1D or 2D data.
See the DIFFEV section on a more general generic evolutionary refinement program that implements the differential evolutionary algorithm Price et al. (2005). Evolutionary or genetic refinement algorithms allow the refinement of models, functions, or more generally speaking the parameters of a cost function to obtain a good solution.
A least squares based refinement of a function $y = F(p_0, p_1, ..., p_n)$ requires the calculation of all partial derivatives $\partial y/\partial p_i$, either from an analytical or a numeric solution. The refinement that is described in this section is intended to work with disordered crystal structures as build by DISCUS . To achieve this REFINE uses a macro that is provided by the user to build a crystal and to calculate a diffraction pattern or a PDF. As the details of a calculation are hidden within the source code of the DISCUS and KUPLOT section, REFINE generally relies on a numerical calculation of the derivative.
Since this macro could calculate any cost function, REFINE is not limited to the refinement of a particular physical problem.

1.2 Getting started

After the program *DISCUS_SUITE* is installed properly and the environment variables are set, the program can be started by typing 'discus_suite' at the operating systems prompt.
The section uses the identical command language to interact with the user as is used throughout the DISCUS SUITE. The command `exit` terminates the section and returns control to the top level of the DISCUS SUITE. All commands of REFINE 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. REFINE is case sensitive, all commands and alphabetic parameters MUST be typed in lower case letters. If REFINE 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 man-

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 DIFFEV. It could, for example be the actual name of a file, or a numerical value.
`text`	Text in single quotes exclusively refers to REFINE 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

Variable	Description
F_DATA	Kuplot data set that holds the experimental data.
F_SIGMA	Kuplot data set that holds the experimental uncertainties.
F_XMIN	Minimum 'x'-value of the experimental data set.
F_XMAX	Maximum 'x'-value of the experimental data set.
F_XSTP	Interval size along the 'x-axis of the experimental data set.
F_YMIN	Minimum 'y'-value of the experimental data set. xyz data only
F_YMAX	Maximum 'y'-value of the experimental data set. xyz data only
F_YSTP	Interval size along the 'y-axis of the experimental data set. xyz data only

Table 1.2: REFINE variables. Variables marked with * are read-only and cannot be altered.

ual 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 REFINE are listed in Table 1.1. There are several sources of information, first REFINE has a build in online help, which can be accessed by entering the command `help` or if help for a particular command <cmd> is wanted by `help <cmd>`. This manual describes background and principle functions of REFINE and should give some insight in the ways to use this program.
REFINE is distributed as part of the diffuse scattering simulation software DISCUS. However, REFINE can be used as general refinement program separate from the main purpose of the DISCUS program package.

1.3 Command language

The program includes a FORTRAN style interpreter that allows the user to program complex modifications. A detailed discussion about the command language, which is common to all DISCUS package programs can be found in the separate DISCUS package reference guide which is included with the package.
Table 1.2 shows a summary of REFINE specific variables. All variables are read/write.

Chapter 2

Least Squares Refinement

2.1 Refinement via least squares refinement

Every time we measure some physical effect and wish to understand how this effect works, we want to determine the parameters of a model function that will replicate the observations. The term refinement refers to the process by which the parameters of the function are tuned such as to give the best agreement between the observed and calculated values. The term *best agreement* merits careful definition, for right now it is sufficient to say that the sum over all squared differences between the observations and the calculations shall be minimized. Thus, refinement is but a special case of general optimization. A very different example for an optimization could be the task to place as many integrated circuits into a chip and simultaneously achieve the fastest computations. Quite well known is the traveling salesman problem. Here the optimization task requires to find the shortest route that visits a number of spots distributed in space.
By far the fastest refinement technique is a least squares algorithm. Such an algorithm can always be applied if we can describe the physical effect as a function of parameters:

$$y \; = \; F(p_0, p_1, ..., p_n), \tag{2.1}$$

and all the partial derivatives $\partial y / \partial p_i$ can be calculated, either analytically or numerically. For each observed value y_{obs}, we calculate a value y_{calc} and minimize the value of a weighted residual wR:

$$wR = \sqrt{\frac{\sum_i w_i (y_{obs}(i) - y_{calc}(i))^2}{\sum_i w_i y_{obs}(i)^2}} \tag{2.2}$$

Here each difference is multiplied by a weight w_i that reflects the uncertainties of the experimental values. In case of crystal structure analysis, the observed values would be the observed intensities in a diffraction pattern and the calculated values those intensities that were calculated based on a structural model. Model parameters will be the lattice parameters, the positions of the atoms in the unit cell, atomic displacement parameters etc. as well as experimental parameters, such as the background. Under the assumption that we have a periodic crystal, the partial derivatives of the intensity with respect to lattice parameters, atom positions etc., can all be derived analytically. This is the concept you will find within any single crystal structure refinement program or a Rietveld program.

For disordered structures, the situation becomes more complicated. Except for a few special cases like stacking faults or short-range order problems, no general analytical function straightforwardly links the disorder parameter to the intensity. The intensity can still be calculated from structural models. The simulation, however, usually involves the application of random choices to generate part or all of the atom positions, and the analytical derivative of the intensity with respect to the order parameter is no longer available. A numeric calculation of the derivatives involves the repeated simulation of a new model for each parameter and is fairly time consuming.

2.2 Algorithm in the REFINE section

The REFINE section uses a Least-Squares algorithm based on the software in the Numerical Recipes Press et al. (1989). This software is based on the Levenberg-Marquart algorithm. In any least-squares algorithm the derivatives are used to determine a new estimate for each parameter. While the refinement is far from the final solution, each parameter can be modified by a large step in order to quickly approach the global minimum. Close to the minimum the steps need to be smaller not to miss the minimum. Essentially the step size is roughly propertional to the derivative. The Levenberg-Marquardt algorithm, optimizes the steps to be taken.
Within the REFINE section of the DISCUS SUITE the derivatives are calculated numerically. The program runs the simulation several times for each parameter: at the current parameter value and at slightly larger and slightly smaller parameter values. The resulting R-values at each parameter value are analyzed to calculate the derivative for this parameter. The REFINE section calculates the R-value for a given parameter at:

P	R-value(P)
P+h	R-value(P+h)
P-h	R-value(P-h)
P+2h	R-value(P+2h)
P-2h	R-value(P-2h)

The REFINE section calculates a polynomial of order two through these three points and uses this polynomial to derive the value of the derivative a the current value of the Parameter `P`.
In any numerical determination of the derivative it is not straightforward to know what is the best value of the deviation `h` from the current parameter value `P`. If the function whose derivative we need to obtain is a smooth function, a very small value of `h` is best, as this is likely to yield a good approximation to the analytical derivative. The value `h` must, however, be large enough to produce a R-value that differs significantly from the original R-value and whose calculation is not affected by rounding errors that are unavoidable for very small numerical values. Within the REFINE section the value of the absolute shift `h` of the parameter `P` is calculated as `P * shift`, where the value of `shift` defaults to 0.002.
As REFINE does not know the significance behind a user supplied parameter the **newparam** command comes with the optional parameter **shift:** that lets you set the relative parameter shift for each individual parameter.
A special difficulty is encountered for refined parameters that cause a stepwise change in the simulated structure. The diameter of a (small) nanoparticle falls into this category. The number of atoms inside a small nanoparticle is a discrete integer number. Increasing the formally real

valued diameter results in no structural change until the diameter is big enough to add one or more atoms. If such a parameter is modified by a small fraction or by a small absolute amount, no change in the structure might be encountered and the numerical derivative would appear to be zero. Only for a larger shift of the parameter will a significant structureal modification occur, which will in turn affect the R-value. A good value for diameters seems to be around 0.05, i.e. the current diameter is modified by `h= P*0.05`.
A second aspect to consider while using a Least-Squares algorithm that is based on the numerical calculation of the derivatives is the presence of local minima.
A refinement within REFINE typically consists of two user supplied macro. In the main macro the refinement parameters, the input data and convergence criteria are defined. The actual calculation of the model function, respectively the model structure and its diffraction pattern, are carried out by the second macro.
The REFINE section uses four parameters to determine is the refinement has reached convergence. These are based on the absolute value of χ^2, and on the change of χ^2, the relative parameter change and a confidence level.
Convergence is considered to have been reached if either the absolute value of χ^2 falls below a user defined leve, or if all three other criteria are met. These three criteria are:

- The value of χ^2 changes less than a user defined value between two cycles
- The largest relative parameter shift is less than a user defined value. The relative parameter shift is defined as the abolute value of the parameter change between two cycles divided by the parameter uncertainty.
- The confidence level reaches a user defined level.

The correct values of χ^2, of the parameter uncertainties and the confidence value depend on the correct values of the data uncertainties. Often the raw data do not provide an accurate estimate of the data uncertainties. In this case the best strategy to start with is to assign unit wheights to all data points. As this implies that the optimum χ^2 and the confidence level will be unknown, REFINE also requires the user to set a maximum number of refinement cycles.
The REFINE section defines the parameters that shall be refined as DISCUS SUITE variable names. These variable names and their current values are passed on to the user macro that is used to simulate the model structure. To tune the refinement, an initial value, a valid parameter range, and a status flag that indicates if this parameter is to be refined or to be kept fixed can be provided.

Chapter 3

Example refinements

3.1 Simple noisy data function

In this first example the refinement to a simple function is illustrated. The data were calculated as a simple polynomial, see Fig. 3.1:

$$y(x) = P_{const} + P_{lin} * x + P_{quad} * x^2 + P_{trip} * x^3 \tag{3.1}$$

A Gaussian distributed random error was added to each data point with a sigma equal to the absolute y-value of each data point. If y was close to zero, the minimum sigma was set to 0.001. As y depends linearly on each of the parameters P_i, we can expect that this refinement will run smoothly. Essentially arbitrary starting values could be used to perform the refinement.
Two macro files are needed for the refinement, the main refinement macro and the macro that caculates the theoretical function. The main refinement macro that was used for this simple introductory example is:

```
 1: refine
 2: data xy, DATA/triple.noise
 3: newparam, P_const, value:-8.01
 4: newparam, P_lin  , value:1.01
 5: newparam, P_quad , value:1.49
 6: newparam, P_trip , value:0.31
 7: set  cycle,   5
 8: set conv, dchi:0.50, pshift:0.005, conf:0.10, chisq:0.5
 9: set relax, start:0.02, success:0.5, fail:16.0
10: run triple.mac, plot:k_inter.mac
11: exit   ! back to SUITE
```

In line 1 we switch from the main DISCUS SUITE level to the REFINE section. The data are loaded in line 2 as a simple "xy" file. All formats that are supported by the KUPLOT `load` command are available. REFINE stores the data set within its on memory. Thus no harm is done if KUPLOT resets the data sets during its calculations. During a refinement in which you need to calculate and average multiple powder or PDF data sets such a reset migh be the norm rather than the exception.
Lines 3 to 6 define the four parameters we wish to refine. For each parameter a suitable name needs to be defined. This name has to be a variable name that is valid within the DISCUS SUITE. If the variable name does not yet exist, it will be created. The command offers three

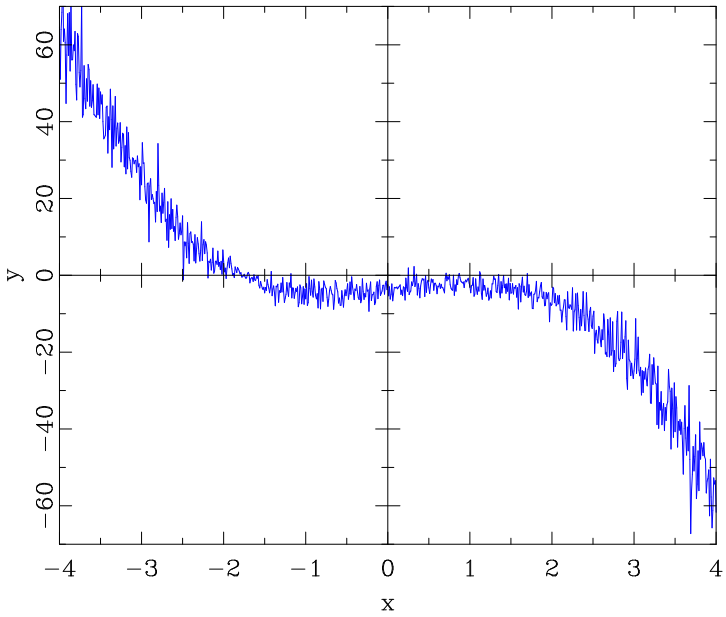

Figure 3.1: Experimental polynomial function

optional parameters, only the "value" parameter is used in this example. it initiates the parameter to the user supplied value, which can be a numerical expression. As further optional parameter you can limit the range over which a parameter is valid by the parameter "range:[<lower>,<upper>]. This will be helpful if you need to exclude negative numbers or if the physical range of a parameter is limited. The last optional parameter allows to change the status of a parameter with "status:fix" or "status:fixed" the parameter will be fixed. If a "value:" parameter is given, the parameter will be fixed to this value, otherwise it takes its current value. If the optional parameter "status:" is omitted or if its value is set to "refine" or "free", its value will be refined.
Line 7 defines the maximum number of refinement cycles that the REFINE section will perform. If the convergence criteria are met in an earlier cycle, the refinement will stop at that point. Several criteria are used to determine if the refinement has reached convergence. In this example they are all specified on line 8. The four criteria are:

- chisq:<value> Convergence has been reached if the value of chi square drops below the user specified value.
- conf:<value> If the confidence level is larger than this value and
- pshift:<value> the largest parameter shift defined as the absolute value of the pareameter shift divided by the abolute value of the parameter uncertainty and
- dchi:<value> the change in chi2 is less than this value then convergence is assumed.

If either the `chisq` or all three other criteria `conf`, `pshift` and `dchi` are met, onvergence is reached and the refinement stops. Keep in mind that the value of chisq and the parameter uncertainty wil depend on the uncertainties of the input data values.

The Levenberg-Marquardt algorithm uses a parameter lamda to adjust the change of a parameter from one refinement cycle to the next. A small value results in a large parameter change and vice versa. REFINE will dynamically adjust this parameter during the refinement. Starting with a value of `0.02` the value of lambda is decreased by multiplication with `success:0.5` if the refinement succeeded i.e. chi2 decreased. The underlying assumption is that as long as the algorithm decreases chi2, we can take larger parameter steps. Once the algorithm does not find a better solution, its time to take smaller parameter steps in order to find the minimum, and lambda is increased with `fail:16.0`. The values in the xample are the default values. The main parameter you may have to play with is the start parameter. A larger value up to 2 seems to help difficult refinements.
The "run" command on line 10 will start the refinement using the macro "triple.mac" to calculate the theory data set. The optional parameter `plot:` allows to specify a user supplied macro that will display any data at each successful refinement step.

```
 1: branch kuplot
 2: #
 3: if(F_DATA==0) then
 4:    reset
 5:    func (P_const)+(P_lin)*r[0]+(P_quad)*r[0]**2+(P_trip)*r[0]**3, F_XMIN, F_XMAX, F_XSTP
 6: else
 7:    func (P_const)+(P_lin)*r[0]+(P_quad)*r[0]**2+(P_trip)*r[0]**3, F_DATA
 8: endif
 9: exit
10: finished
```

The refinement begins within the REFINE section of the DISCUS SUITE. Since we want to calculate a simple x-y data set, the KUPLOT section is the best choice. With the `branch kuplot` command in line 1 REFINE switches to KUPLOT. In line 3 the macro tests if REFINE loaded the data into a KUPLOT data set with data set number `F_DATA`. The REFINE build in variable `F_DATA` defines the data set number within KUPLOT into which REFINE was able to load the data set with the `data` command, line 2 in the main fit macro. The actual limits and the step width that REFINE detects are stored in the REFINE variables `F_XMIN`, `F_XMAX` and `F_XSTP`. Thes variables are all Read/Write but change their values with caution.
In KUPLOT the `func` command will calculate a user supplied function over a user defined range. The first parameter here the string "(P_const)+(P_lin)*r[0]+(P_quad)*r[0]**2+(P_trip)*r[0]**3" defines the function to be calculated. If three parameter follow, they define x-min, xmax and the x-step for the function. The DISCUS SUITE variable r[0] takes the role of the functions x-coordinate. If only one parameter is supplied, it defines the data set number whose range and step width are to be used for the function calculation. In the current example macro lines 5 and 7 will thus calculate the theory function over the identical range.
A reason to deveate from the limits of the data set occures when you reifne against a powder diffraction data set. In order to take the effect of Bragg reflections slightly above the upper 2Theta limit into account, it is necessary to calculate the powder pattern to a larger 2Theta value. This will ensure that the low 2Theta tail of these Bragg reflections will b e present within the calculated powder patttern, even if the peak position of the Bragg reflections is not.
The user macro needs to ensure that the last data set loaded into KUPLOT is the final theory curve.
At line 9 the user macro returns to REFINE . Finally in line 10, the REFINE command `finished` tells REFINE that the user macro is finished.

For the current macro the output produced during the refinement will be the following lines:

```
Cyc Chi^2/(N-P)   MAX(dP/sig)   Conf          Lambda       wRvalue      Rexp
  0   1.0222      0.26239      0.34233      0.50000E-03  0.25018      0.24745
  1   1.0219      0.14187E-02  0.34404      0.25000E-03  0.25015      0.24745

Convergence reached

 Information about the fit :
   Chi^2      :  814.484          Chi^2/N :  1.01683
   Conf. level: 0.344038      Chi^2/(N-P) :  1.02194
   No.Data    :          801     No.Params:            4
   MRQ final  : 0.250000E-03
   wR value   : 0.250150          R exp   : 0.247451

 Correlations larger than 0.8 :
   ** none **

   P_const          :   -3.92623      +-    0.107205
   P_lin            :    2.16236      +-    0.772162E-01
   P_quad           :   0.512809      +-    0.314855E-01
   P_trip           :   -1.11685      +-    0.134130E-01
```

The optional plot macro needs to start within the KUPLOT section and return with a final exit to REFINE. For the current example you could use:

```
 1: load xy, DATA/triple.noise
 2: kccal sub, 2,1
 3: r[0] = min(ymin[1],ymin[2]) - ymax[3]
 4: ccal add, wy, 3, r[0]
 5: skal
 6: mark
 7: rval 2,1, dat
 8: reset
 9: exit
```

3.2 Nanoparticle PDF

In this section we will introduce the refinement of a nanoparticle PDF. This initial example is fairly straightforward. The data set is the calculated PDF of a spherical ZnO 45 Ådiameter nanoparticle without any defects, Fig. 3.2. Thus we can, of course expect a rather perfect fit result.
The main parts of the main refinement macro are identical to the example in section 3.1.

```
 1: refine
 2: #
 3: data xy, DATA/zno.grobs
 4: newparam  P_lata   , value:3.220, range:[2.90,3.50], shift:0.0020 , points:5
 5: newparam  P_latc   , value:5.050, range:[4.70,5.30], shift:0.0020 , points:5
 6: newparam  P_z_zn   , value:0.360, range:[0.30,0.42], shift:0.0020 , points:5
 7: newparam  P_biso   , value:0.400, range:[0.00,]    , shift:0.0001 , points:5
 8: newparam  P_ab_dia , value:45.00, range:[20.0,80.0], shift:0.0100 , points:5
 9: #
10: newparam  P_eta    , value:0.600, range:[0.00,1.00], shift:0.0020 , points:5
11: newparam  P_eta_l  , value:0.000, status:fixed
12: newparam  P_u      , value:0.000, status:fixed
13: newparam  P_v      , value:0.000, status:fixed
```

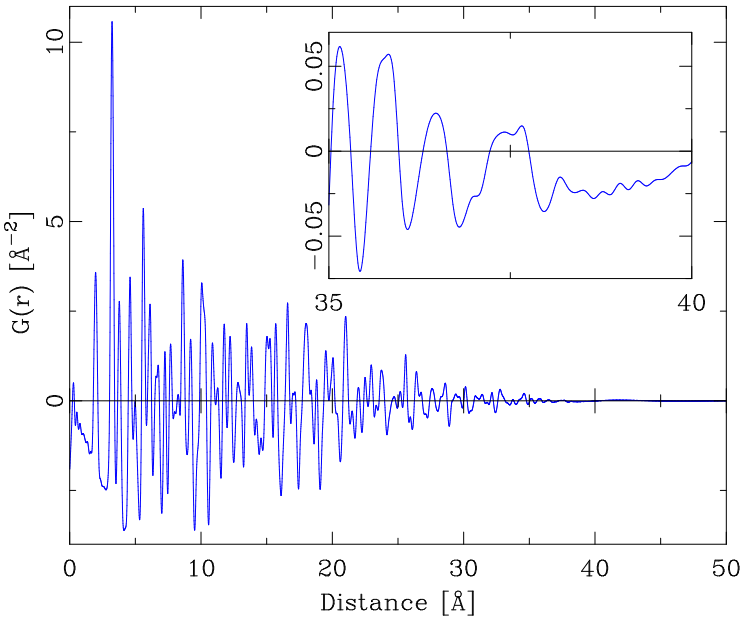

Figure 3.2: Experimental nanoparticle PDF. The nanoparticle has the perfect ZnO structure with a diameter of 45 Å. The insert shows enlarged the longer distances.

```
14: newparam  P_w      , value:0.001, range:[0.00,0.10], shift:0.0020 , points:5
15: newparam  P_qmax   , value:24.00, status:fixed
16: newparam  P_scale  , value:1.000, range:[0.10,10.0], shift:0.0020 , points:5
17: set cycle,  25
18: set relax, start,2.00
19: set conver, dchi:0.5, pshift:0.005, conf:0.90, chisq:1.1
20: #
21: run discus_main.mac, plot:k_inter.mac
22: #
23: exit
```

As before, the data set is loaded as a simple x-y ASCII data set in line 3. The parameters are defined in lines 4 to 16. The main difference is the use of the additional optional parameters `range`, `shift` and `points`.

With `range` a range of allowed parameter limits can be defined. You can use this to restrict the refinement range to a sensible range, limited by intuition or by physical constraints. The optional parameter takes two values, the lower and upper boundary. One of these two values may be omitted, as in the case of `P_biso`. The lower limit is defined as 0.00, as a atomic displacement parameter cannot assume negative values. The upper limit is absent, allowing arbitrarily large values.

The next optional parameter `shift` allows to specify a multiplicative term that is used to calculate parameter points at `p-2*delta`, `p-1*delta`, `p-2*delta`, `p-2*delta`. These additional points are needed to calculate a numerical derivative of the function value with respect to the parameter at hand. An accurate derivative will be calculated at the current parameter value if the shifts `delta` are small. In a crystal structure simulation there are, however, a number of parameter types that need a rather large shift `delta`. The most prominent example of this class are nanoparticle diameters. If these are increased by a small amount, no atom might be added

to the structure at all, as the boundary surface might cut in between the positions of all atoms. In the example a value of `shift:0.01` works well.
The last optional parameter `points` specifies how many points are used to calculated the derivative. Possible values are 3 or 5, which include the parameter value itself. The larger number of points will provide a more accurate derivative, but will of course require more computing time.
Another issue to be considered is the determination and transfer of fixed parameters from the main fit macro to the slave macro. Two different styles might be adopted. You can create variables and set their values within a separate macro, which you use at all appropriate macros. The other option, chosen in this example, is to use the `newparam` command but to fix the parameter value. It is mostly a matter of personal choice which style you prefer. As an argument in favor of the `newparam` style, consider the profile parameters `P_u`, `P_v`. It will often be necessary to refine all three profile shape parameters. It makes sense though to start with the simplest model, a profile function of constant width, determined by parameter `P_w`. If, later on, you decide that a full profile function is needed, all you have to do is to free parameters `P_u` and `P_v`. For the style of a separate macro, you would have to remove the parameters from the variable defining macro, remove all initilization lines in any macro and add new parameters to the main fit routine. This is likely more error prone.
The main DISCUS macro DISCUS_main.mac has to build a spherical nanoparticle based on the current refinement parameters, calculate the necessary diffraction data and provide these as final data set within the KUPLOT section.

```
 1: branch discus
 2: variable integer, ncellx
 3: variable integer, ncellz
 4: #
 5: read
 6:   stru CELL/zno.cell
 7: lat[1] = P_lata
 8: lat[2] = P_lata
 9: lat[3] = P_latc
10: z[1]   = P_z_zn
11: b[1]   = P_biso
12: b[2]   = P_biso
13: ncellx = 2.00*int(P_ab_dia/lat[1]) + 2
14: ncellz = 2.00*int(P_ab_dia/lat[3]) + 2
15: save
16:   outf internal.zno.cell
17:   omit all
18:   run
19: exit
20: read
21:   cell internal.zno.cell     , ncellx, ncellx, ncellz
22: surface
23:   boundary sphere, P_ab_dia/2.
24: exit
25: purge
26: @powder.mac
27: @output.mac TEMP/zno.grcalc
28: branch kuplot
29: rese
30: load xy, TEMP/zno.grcalc
31: skal
32: ccal mul, wy, 1 , P_scale
33: ksav 1
34:   form xy
```

```
35:   outf   TEMP/zno.grcalc
36:   run
37: exit
38: exit
39: finished
```

In lines 5 to 12 a template asymmetric unit is read, its lattice parameters atom position and ADP values are adapted. The known lattice parameters are used to calculate the number of unit cells required to fit a spherical object into a block of unit cells in lines 13 and 14. The modified asymmetric unit is saved to an internal file (lines 15 to 19) and a block of unit cells is build using the current information, lines 20 and 21. Within the `surface` menu (lines 22 to 24) a sphere is cut with radius `P_dia/2` and all voids are removed from the structure, line 25. The PDF is determined by calculating the powder diffraction pattern through the Debye Scattering Equation, macro `powder.mac` line 26. This PDF is saved with the `output` menu in line 27. In the remainder of the macro the PDF is scaled and its final form saved to the hard disk. The `exit` in line 37 returns to the DISCUS section and the `exit` in line 38 returns to the REFINE section. The slave macro is terminated by the special REFINE command `finished`.

```
 1: powder
 2: xray
 3: set axis,q
 4: set calc,debye
 5: set disp,off
 6: set delta,0.00
 7: set qmin, 0.5000
 8: set qmax, P_qmax
 9: set dq,   0.0001
10: set profile, pseudo
11: set profile, eta, P_eta, P_eta_l
12: set profile, uvw, P_u, P_v, P_w
13: set profile,  width, 11
14: set temp,use
15: set wvle,0.20000
16: set four,four
17: set lpcor,bragg,0.500
18: set scale, 1.000
19: run
20: exit
```

The powder menu uses the Debye Scattering Equation (line 4) to calculate the powder pattern for the finite sized spherical nanoparticle. While the lower limit Q_{min} is fixed in this example to 0.5, the upper limit is determined by the (fixed) parameter `P_qmax`. A Pseudo Voigt profile function is set up in lines 11 to 13.

```
 1: output
 2: value PDF
 3: outf \$1
 4: form powder, r, 0.01,  50, 0.01
 5: run
 6: exit
```

In order to generate the PDF, the output menu is instructed to write the data with value `PDF`, line 2. Correspondingly, the format is set to `r` with user supplied limits in line 4. In this example, the limits are fixed number, if necessary this can be changed to use variable names. This refinement proceeds with the following initial output:

```
Cyc Chi^2/(N-P)   MAX(dP/sig) Par   Conf          Lambda       wRvalue      Rexp
  0   2273.9      0.31748E-02   5   0.0000      0.10000E-01   1.0180      0.21348E-01
  1   1476.8      0.15617       5   0.0000      0.50000E-02  0.82039      0.21348E-01
  2   1194.3      0.56799E-01   5   0.0000      0.25000E-02  0.73777      0.21348E-01
  3   862.37      0.59080E-02   6   0.0000      0.12500E-02  0.62690      0.21348E-01
  4   557.78      0.13358E-01   5   0.0000      0.62500E-03  0.50418      0.21348E-01
  5   329.67      0.14493E-01   5   0.0000      0.31250E-03  0.38761      0.21348E-01
  6   171.28      0.10872E-01   5   0.0000      0.15625E-03  0.27939      0.21348E-01
  7   138.25      0.17644E-01   6   0.0000      0.78125E-04  0.25101      0.21348E-01
  8   51.188      0.99726E-02   5   0.0000      0.39062E-04  0.15274      0.21348E-01
  9   26.416      0.29501E-01   6   0.0000      0.19531E-04  0.10972      0.21348E-01
 10   8.3212      0.51764E-01   6   0.0000      0.97656E-05  0.61581E-01  0.21348E-01
 11   6.7917      0.62021E-02   5   0.0000      0.48828E-05  0.55635E-01  0.21348E-01
 12   6.7917       0.0000       1   0.0000      0.78125E-04  0.55635E-01  0.21348E-01
 13   6.7917       0.0000       1   0.0000      0.12500E-02  0.55635E-01  0.21348E-01
 14   6.7917       0.0000       1   0.0000      0.20000E-01  0.55635E-01  0.21348E-01
 15   6.7917       0.0000       1   0.0000      0.32000      0.55635E-01  0.21348E-01
 16   5.8721      0.16346E-02   5   0.0000      0.16000      0.51731E-01  0.21348E-01
 17   4.7877      0.26307E-02   5   0.0000      0.80000E-01  0.46711E-01  0.21348E-01
 18   3.7652      0.35205E-02   5   0.0000      0.40000E-01  0.41424E-01  0.21348E-01
 19   2.8465      0.19826E-02   5   0.0000      0.20000E-01  0.36017E-01  0.21348E-01
 20   1.6043      0.16427E-02   5   0.0000      0.10000E-01  0.27039E-01  0.21348E-01
 21   1.2085      0.17484E-03   5  0.12624E-21  0.50000E-02  0.23468E-01  0.21348E-01
Convergence reached
```

As you can see the inital agreement is not very well, nevertheless the algorithm manages to find and refine into the perfect solution in cycle 21. Note that in cycles 12 to 15 the algorithm does not find a better solution and increases `lambda`. At cycle 21 the shift in chi squared is less than the user defined threshold and the refinement terminates.
The final output lists the convergence criteria, the final agreement values and the refined parameters.

```
 Data loaded as     : xy, DATA/zno.grobs
 Sigma not defined
 Refinement macro   : discus_main.mac
   Convergence 1    : dP/sigma < AND conf >     AND dChi^2 <
                      0.500E-002     0.900          0.500
   Convergence 2    : dChi^2 <   AND dP/sigma > 0.0
                      0.500
   Convergence 3    : Chi^2 <
                       1.10

 Information about the fit :
   Chi^2      :  6032.82          Chi^2/N :  1.20656
   Conf. level: 0.126237E-21  Chi^2/(N-P) :  1.20850
   No.Data    :         5000     No.Params:            8
   MRQ final  : 0.500000E-02
   wR value   : 0.234680E-01      R exp   : 0.213478E-01

 Correlations larger than 0.8 :
   P_w              - P_eta            : -0.953

 Refined parameters
   P_lata           :    3.24883      +-    0.126421E-04    [   2.90000     ,   3.50000     ]
   P_latc           :    5.20379      +-    0.296431E-04    [   4.70000     ,   5.30000     ]
   P_z_zn           :   0.367053      +-    0.108764E-04    [  0.300000     ,  0.420000     ]
   P_biso           :   0.462104      +-    0.327584E-03    [   0.00000     ,               ]
   P_ab_dia         :    39.7863      +-    0.129176E-01    [   20.0000     ,   80.0000     ]
   P_eta            :   0.867988      +-    0.174564E-01    [   0.00000     ,   1.00000     ]
   P_w              :   0.520064E-003 +-    0.197551E-04    [   0.00000     ,  0.100000     ]
   P_scale          :   0.986814      +-    0.837900E-03    [  0.100000     ,   10.0000     ]
```

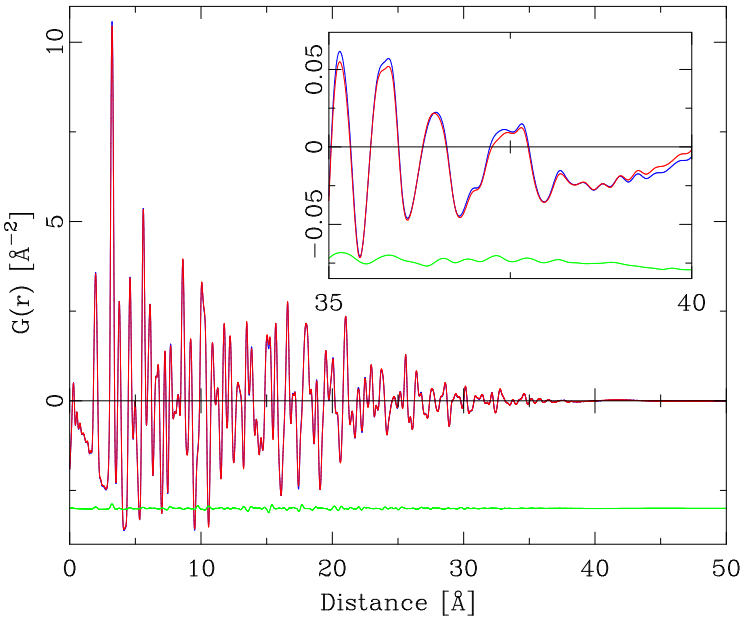

Figure 3.3: Experimental nanoparticle PDF overlayed with the final result. The difference curve in green is shifted down for clarity.

```
Fixed   parameters
  P_eta_1           :    0.00000
  P_u               :    0.00000
  P_v               :    0.00000
  P_qmax            :    24.0000
```

Fig 3.3 shows the excellent agreement between *observed* and calculated datai, even at very large distances near the nanoparticle diameter.

Chapter 4

Mathematical Background

4.1 Least squares algorithm

This chapter describes the mathematical background to the least squares algorithm implemented in REFINE. The description largely follows the discussion in the Numerical recipes Price et al. (2005).
A least squares algorithms works on the basic assumption that the we are able to describe our data by a model function that depends on M parameters:

$$y \; = \; F(x; p_0, p_1, ..., p_M) \tag{4.1}$$

If the uncertainties of the data are subject to a Gaussian distribution, the highest probability for any model parameters is given if the weighted residual. The weights are the inverse of the variance (= squared uncertainties) of the data points. $w_i = 1/\sigma_i^2$:

$$wR = \sqrt{\frac{\sum_i w_i (y_{obs}(i) - y_{calc}(i))^2}{\sum_i w_i y_{obs}(i)^2}} \tag{4.2}$$

is at its lowest possible value. In this equation for the weighted residual the sums run over all observed data points `i`. These data points might be the individual data points in a powder pattern, in a PDF or a two or three dimensional data set. The data points may likewise include several data sets and possible the need for special functions for a subsection of the data set. This latter situation occurs if we perform a simultaneous refinement of a structural model aginst a neutron and X-ray data set. Note that there is no need for the data points to be arranged in any special sequence of `i`.
Often the weighted residual as defined in Eq. 4.2 is replaced by the term "chi-squared", defined as:

$$\chi^2 = \sum_i w_i (y_{obs}(i) - y_{calc}(i))^2 = \sum_i \frac{(y_{obs}(i) - y_{calc}(i))^2}{\sigma_i^2} \tag{4.3}$$

A necessary condition for a minimum of the residual is that all derivatives of `wR` with respect to any parameter is zero. This condition holds of course for the squared residual as well:

$$wR^2 = \frac{\sum_i w_i (y_{obs}(i) - y_{calc}(i))^2}{\sum_i w_i y_{obs}(i)^2} \tag{4.4}$$

Thus, for each parameter j we have the partial derivative:

$$\frac{\partial wR^2}{\partial P_j} = \frac{2}{\sum_i w_i y_{obs}(i)^2} \sum_i w_i \left(y_{obs}(i) - y_{calc}(i)\right) \frac{\partial (y_{obs}(i) - y_{calc}(i))}{\partial P_j} = 0 \tag{4.5}$$

Since the observed values are independent of the parameters the equation simplifies to:

$$\frac{\partial wR^2}{\partial P_j} = -\sum_i w_i \left(y_{obs}(i) - y_{calc}(i)\right) \frac{\partial y_{calc}(i)}{\partial P_j} = 0 \tag{4.6}$$

To simplify the notation we will replace $y_{obs}(i)$ by y_i and $y_{calc}(i)$ by f_i. This change in notation has our equations for the partial derivatives read:

$$\frac{\partial wR^2}{\partial P_j} = -\sum_i w_i \left(y_i - f_i\right) \frac{\partial f_i}{\partial P_j} = 0 \tag{4.7}$$

A total of M such equations must be solved for all parameters P from 1 to M.

4.1.1 Linear least squares algorithm

A reasonable straightforward solution to find the optimum parameters P_j exists if the function f_i can be written as a linear combination of the parameters:

$$f_i = \sum_j P_j g_{ij} \tag{4.8}$$

Here g_{ij} can be any function of the data points `i`. As an example of such a linear combination take any polynomial function:

$$y_i = P_0 + P_1 * x_i^1 + P_2 * x_i^2 + P_3 * x_i^3 + ... \tag{4.9}$$

Here the functions g_{ij} are powers of x at point `i`. We are now ready to substitute f_i and its derivative with respect to parameter `j` in Eq. 4.7 by the function definition in 4.8 and obtain:

$$0 = -\sum_{i=1}^{N} w_i \left(y_i - \sum_{k=1}^{M} P_k g_{ik} \right) g_{ij} \tag{4.10}$$

This equation must hold for all parameters j=1 to M, and we can write all these equations as:

$$\sum_{k=1}^{M} \sum_{i=1}^{N} w_i \, g_{ik} \, g_{ij} P_k = \sum_{i=1}^{N} w_i \, y_i \, g_{ji} \qquad \text{j=1 to M} \tag{4.11}$$

This set of equations can be written concisely if we introduce a few matrices. The first is the so called design matrix A. Its elements are the values of the theory functions g_{ij} divided by the experimental uncertainties at data point `i`, with $\sigma_i = \sqrt{w_i}$:

$$\mathbf{A} = \begin{pmatrix} \frac{g_{11}}{\sigma_1} & \frac{g_{12}}{\sigma_1} & \cdots & \frac{g_{1M}}{\sigma_1} \\ \frac{g_{21}}{\sigma_2} & \frac{g_{22}}{\sigma_2} & \cdots & \frac{g_{2M}}{\sigma_2} \\ \vdots & & & \\ \frac{g_{N1}}{\sigma_N} & \frac{g_{N2}}{\sigma_N} & \cdots & \frac{g_{NM}}{\sigma_N} \end{pmatrix} \tag{4.12}$$

Each row corresponds to the M functions g_{ij} evaluated for data point `i`, while each column corresponds to a specific function evaluated at all data points. Like wise we can group the parameters P_k into a column matrix:

$$\mathbf{a} = \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_M \end{pmatrix} \tag{4.13}$$

Next we define a column matrix for the N observed data points as :

$$\mathbf{b} = \begin{pmatrix} \frac{y_1}{\sigma_1} \\ \frac{y_2}{\sigma_2} \\ \vdots \\ \frac{y_N}{\sigma_N} \end{pmatrix} \tag{4.14}$$

This allows to write set M set of Eqs. 4.11 as:

$$\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{P} = \mathbf{A}^T\mathbf{b} \tag{4.15}$$

Since $\mathbf{A}^T\mathbf{A}$ is a square MxM matrix, we can solve the equation to obtain the parameters as:

$$\mathbf{P} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} \tag{4.16}$$

The inverse matrix $\left(\mathbf{A}^T\mathbf{A}\right)^{-1}$ is called the *covariance matrix* $\mathbf{C}$. Its elements allow to estimate the uncertainties of the parameters P_k and any correlations between two parameters. Each element of the column matrix P in equation 4.16 is:

$$P_k = \sum_{j=1}^{M} C_{kj}\left(\sum_{i=1}^{N}\frac{y_i g_{ij}}{\sigma_i^2}\right) \tag{4.17}$$

The variance (= squared uncertainty) of parameter k can be calculated as:

$$\sigma^2(P_k) = \sum_{i=1}^{N}\sigma_i^2\left(\frac{\partial P_k}{\partial y_i}\right)^2 \tag{4.18}$$

Note that in Eq. 4.17 the elements g_{ij} are independent of y_i. Furthermore in the sum over `i` the derivative with respect to y_i is non-zero for element `i` only. Thus the derivative in Eq. 4.18 is:

$$\frac{\partial P_k}{\partial y_i} = \sum_{j=1}^{M} C_{kj}\frac{g_{ij}}{\sigma_i^2} \tag{4.19}$$

We can insert this derivative in squared form into Eq. 4.18, interchange the sequence of the sums to write:

$$\sigma^2(P_k) = \sum_{j=1}^{M}\sum_{l=1}^{M} C_{kj}C_{jl}\left(\sum_{i=1}^{N}\frac{g_{ij}g_{il}}{\sigma_i^2}\right) \tag{4.20}$$

The right hand term corresponds to the elements of $\left(\mathbf{A}^T\mathbf{A}\right)$, which in turn is the invariant matrix to $\mathbf{C}$. Thus the equation reduces to:

$$\sigma^2(P_k) = C_{kk} \tag{4.21}$$

The diagonal element of matrix $\mathbf{C}$ are the variances of the parameters k.

4.1.2 Non-linear least squares algorithm

Most of the times, the function f_i cannot be written as a linear combination of the parameters. As an example take a function like:

$$f_i = P_1\cos(P_2 \cdot x_i) \tag{4.22}$$

Under these circumstances, the individual parameters cannot be separated from each other. Iterative strategies have been developed to find the minimum. Independend of the complexity of the function f_i, in close proximity to the minimum one can expect to approximate the value of χ^2 as a quadriatic function of the parameters P_i:

$$\chi^2(P_j) = const - d_jP_j + \frac{1}{2}P_jD_{jk}P_k \tag{4.23}$$

Here, we can write this equation in more concise form if we use a matrix formalism. With matrix **a** from Eq. 4.13, a vector of length M (= number of parameters), and an MxM matrix **D**:

$$\chi^2(a) = const - \mathbf{d}\,\mathbf{a} + \frac{1}{2}\mathbf{a}^T\mathbf{D}\mathbf{a} \tag{4.24}$$

Very close to the minimum, we can hope for the approximation to be a good one which allows us to find the best parameters with a single step:

$$\mathbf{a}_{best} = \mathbf{a}_{current} + \mathbf{D}^{-1}\cdot\left(-\nabla\chi^2(\mathbf{a}_{current})\right) \tag{4.25}$$

Unfortunately, most of the time we cannnot know how good our current approximation will be. The best one can do is to calculate the gradient of the squared residual and take a small step into the direction of the steepest decent. At this new point in the parameter space one needs to calculate the gradient again and take further steps towards the minimum. At any iteration during this procedure, the new parameter set can be calculated as:

$$\mathbf{a}_{next} = \mathbf{a}_{current} - constant \times \nabla\chi^2(\mathbf{a}_{current}) \tag{4.26}$$

The tricky part is the determination of the value of *constant*. It must be small enough to take a step downhill, and not beyond the minimum. A very small value will, however, increase the number of iterations it takes to reach the minimum. A further risk with a very small value of

the *constant* is to step into a local minimum. Thus different values of the *constant* and a dynamic adapration are needed.
As for the linear model we have out function:

$$y \;=\; F(x; p_0, p_1, ..., p_M) \tag{4.27}$$

and we need to minimize chi squared:

$$\chi^2 = \sum_i w_i (y_{obs}(i) - y_{calc}(i))^2 = \sum_i \frac{(y_{obs}(i) - y_{calc}(i))^2}{\sigma_i^2} \tag{4.28}$$

As in the introductory part we will reduce the terminology to:

$$\chi^2 = \sum_i w_i (y_i - f_i)^2 \tag{4.29}$$

In order to take steps towards the minimum we need to calculate the gradient of χ^2 with respect to each parameter k:

$$\frac{\partial \chi^2}{\partial P_k} = -2 \sum_{i=1}^{N} \frac{(y_i - f_i)}{\sigma_i^2} \frac{\partial f_i}{\partial P_k} \tag{4.30}$$

A further derivative with respect to another parameter give the cross terms needed to a direct decent:

$$\frac{\partial^2 \chi^2}{\partial P_k \partial P_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left(\frac{\partial f_i}{\partial P_k} \frac{\partial f_i}{\partial P_l} - (y_i - f_i) \frac{\partial^2 f_i}{\partial P_k \partial P_l} \right) \tag{4.31}$$

The individual equations 4.31 can be considered to be matrix elements of a matrix α with:

$$\alpha_{kl} \;=\; \frac{1}{2} \frac{\partial^2 \chi^2}{\partial P_k \partial P_l} \tag{4.32}$$

and a vector β:

$$\beta_k \;=\; -\frac{1}{2} \frac{\partial^2 \chi^2}{\partial P_k} \tag{4.33}$$

Here the factors $1/2$ have been introduced for convenience sake. With these definitions the parameter shift can be written as a vector δ and the equation to be solved becomes:

$$\mathbf{A}\delta \;=\; \beta \tag{4.34}$$

In the algorithm implemented in REFINE, as taken from Price et al. (2005) the second derivatives in Eq. 4.31 is omitted, making the matrix elements α_{kl}:

$$\alpha_{kl} \;=\; \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left(\frac{\partial f_i}{\partial P_k} \frac{\partial f_i}{\partial P_l} \right) \tag{4.35}$$

The reasoning behind this omission is that the second derivatives tend to be much smaller, tend to destabilize the algorithm due to the noise in the experimental data and the final parameter values do not depend on the second derivatives, just the path towards the final parameters.

In terms of the matrices introduced in Eqs. 4.32 and 4.33, the steepest decent in Eq. 4.26 corresponds to

$$\delta a_l = constant \times \beta_l \tag{4.36}$$

Far from the minimum, this decent is the optimum method, while close to the minimum a solution according to 4.34 serves best. The commonly used Levenberg-Marquardt method provides an algorithm to vary between these two algorithms. The algorithm is described by Marquardt (1963) based on earlier suggestions by Levenberg (1944).
There are two parts to the Levenberg-Marquardt algorithm. First of all, a scale factor must be included into equation 4.36 to convert the dimensions of β into those of the parameters α. This conversion factor will likely be different for each parameter α. The diagonal elements of matrix **A** in Eq. 4.32 do provide this vonversion factor. The Levenberg-Marquadrt algorithm provides an additional dimensionless factor λ to scale the parameter shift, turning Eq. 4.36 into

$$\delta a_l = \frac{1}{\lambda \alpha_{ii}} \times \beta_l \tag{4.37}$$

The second part to the Levenberg-Marquardt algorithm provides a transition between the steepest decent and the direct solution. The matrix α is replaced by a new matrix α', where the diagonal elements are replaced according to:

$$\alpha'_{jj} = \alpha_{jj}\ (1\ +\ \lambda) \tag{4.38}$$

while the off diagonal elements remain the same.
The steepest decent, Eq. 4.36, and the direct solution close to the minimum, Eq. 4.34, are combined into a single new equation:

$$\boldsymbol{\alpha'} \delta \boldsymbol{a} = \boldsymbol{\beta} \tag{4.39}$$

The implementation of the Levenberg-Marquardt algorithm used in REFINE is taken from the numerical recipes Press et al. (1989) and is initialized by the following steps:

- Provide initial guesses for all parameters **a**.
- Provide initial guesses for λ.
- Calculate the corresponding value of χ^2.

The program then carries out the iterations by:

- Solving Eq. 4.39 for the parameter shifts.
- Calculate χ^2 for the modified parameters $a + \delta a$.
- If the iteration succeeds i.e. the new χ^2 is less than the previous one, the new parameter values are accepted and λ is decreased by a suitable factor.
- If the iteration did not succedd, i.e. the new χ^2 is higher than the previous one, the old parameter set is retained and λ is increased by a suitable factor.

REFINE enables the user to provide a starting value for λ and individual factors for the decrease and increase.

Appendix A

REFINE commands

A.1 Summary

Here is a list and brief description of valid REFINE commands. Further help can be obtained by typing the corresponding command name at the help prompt.

```
data     ! Defines the data set
finished ! Terminates the slave macro
fix      ! Fixes a parameter, free with newparam
newparam ! Defines and sets a new parameter, which is to be refined
reset    ! Performs a global reset to system start conditions
run      ! Starts the refinement
show     ! Shows settings
set      ! Sets control values like cycles, convergence criteria
sigma    ! For 2D data; defines a file that contains experimental uncertainties
```

A.2 News : Information on program updates

This is the initial release of the REFINE section.

A.3 Example

A simple fit might refine parameters of a straight line to observed data. Lets assume that the data are in a simple x/y file "observed.data". A suitable macro to refine is:

```
refine                        ! Switch from SUITE to REFINE section
data xy, observed.data        ! Load data set "observed.data"
                              ! The load command creates user variables
                              ! F_DATA = Number of Data set in Kuplot
                              ! F_XMIN, F_XMAX, F_XSTP
                              ! F_YMIN, F_YMAX, F_YSTP
                              ! That contain the data limits and step size.
newparam P_inter, value:1.0   ! Define y-axis intersept as first parameter
newparam P_slope, value:1.0   ! Define slope as second parameter
set cycle, 10                 ! Define maximum number of cycles
run fit_work.mac              ! Start the fit with user macro fit_work.mac
exit                          ! Back to SUITE
```

The user macro fit_run should be something like:

```
branch kuplot                ! Step into the KUPLOT section
                             !   or for structures into DISCUS
func P_inter + P_slope*r[0], F_DATA
                             ! Calculate the function,
                             !   limits are set automatically from
                             !   data set F_data that was loaded by refine.
exit                         ! Back to REFINE
finished                     ! Special keyword signals end of user macro
```

A.4 data

```
data "kuplot", <number>
data <type>, <infile>
data "kuplot", <number>
```

The calculation will refine parameters versus the observed data that are stored in the KUPLOT data set number <number>

```
data <type>,<infile>
```

The calculation will refine parameters versus the observed data that are loaded from file <infile>.
See the ==> 'kuplot/load' command for details on proper ways to load a data set.
Refer to the KUPLOT section for instructions on loading data.

A.5 finished

```
finished
```

This command must be the last instruction in any slave macro. The command instructs refine to terminate execution of the slave macro. REFINE will then evaluate the results and proceed to the next refinement cycle.

A.6 fix

```
fix <parname> [,"value:"<number>]
```

Fixes a parameter. Its value will remain at its current value or at <number>, if the optional parameter "value:" is used.
A fixed parameter can be freed with a ==> 'newparam' command.

A.7 newparam

```
newparam <parname> [,"value:"<start>] [,"status:"<flag>]
                   [,"range:["<lower>,<upper>"]"]
                   [,"shift:"<value>] [,"points:"<nderiv>]
```

The command performs three tasks: - Defines a new parameter named <parname> - Frees a parameter <pname> that had previously been fixed by the ==> 'fix' command - Defines a constant <pname> that you may need to use in the user macro.
Defines a new parameter name <parname>. This should be any valid user defined variable name, limited to a length of 16 characters. The user variable is allowed to have been defined previously, and its current value will be used if the "value:" option is omitted.
See the general command line section for details on the definition of variables.
Optional parameters are: value:<start> The Parameter will be initialized to the <start> value. If omitted, the parameter value will take on its current value.
status:refine status:free status:fix status:fixed With the flags "refine" or "free" the parameter will be refined during the refinement cycles. With the flags "fixed" or "fix", the parameter will remain fixed at its current value. Be careful that the user macro does not change the parameter value! Default flag is "refine".
range:[<lower>, upper>] Defines a lower and upper boundary for the parameter. The fit will ensure that the parameter does not move outside the specified range. <lower> must be less than <upper>. If either <lower> or <upper> is absent, but the comma is present: [,<upper>] or [<lower>,] the missing side is not constrained. If the "range:" parameter is omitted, the default behavior is to assume no boundaries. In order to turn the boundaries off, simply state the 'newparam' command again for the refinement parameter without the "range:" option.
shift:<value> To calculate the numerical derivatives REFINE needs to calculate the cost function at the current parameter value and at points at slightly smaller and slightly larger values. REFINE calculates the shift of the original parameter values as par*<value>. It is not straightforward to recommend good values. For parameters that can be expected to give a smooth variation of the cost function like scale factors a value of 0.003 seems to be good. For parameters like nano particle diameters larger values like 0.1 to 0.01 seem to be better. Default: shift:0.003
points:<nderiv> REFINE estimates the derivative by calculating the cost function at either: the 3 points p-h; p; p+h or at the 5 points p-2h; p-h; p; h+h; p+2h. The five point estimate is more accurate and is recommended for parameters like nanoparticle diameters. Default: points:3

A.8 reset

```
reset
```

Puts the REFINE section back into the state at system start.

A.9 run

```
run <work.mac> [, "plot:"<kuplot_macro.mac>]
```

Starts the actual refinement using the macro <work.mac> to simulate the structure and to evaluate the cost function.
The optional parameter "plot:" instructs REFINE to display the observed and calculated data at each cycle. The macro <kuplot_macro> must be a pure KUPLOT macro. The last line in this macro must be an 'exit' command that returns to REFINE.

A.10 show

```
show
```

Shows the current refinement status.

A.11 set

```
set cycle,<maxc>
set conver, ["status:on" | "status:off"]
set conver ["dchi:"<delta>] [,"pshift:"<max>] [,"conf:"<level>]
           ["chisq:<value>]
set relax  ["start:"<lamda_s>] [,"fail:"<lamda_f>]
           [,"success:"<lamda_g>]
```

cycle

```
set cycle,<maxc>
```

Sets the maximum number of refinement cycles

convergence

```
set conver [,"dchi:"<delta>] [,"pshift:"<max>] [,"conf:"<level>]
           [,"chi2:"<level>] [,"status:on" | "status:off"]
```

Allows the user to define convergence criteria.
If the status is set to "on", the convergence criteria are used. Otherwise the refinement will run for the cycles defined by ==> 'set cycle'.
dchi:<delta> If the value of Chi**2/(Ndata-Npara) decreases by less than <delta> convergence is reached. Defaults to 0.5 pshift:<max> If all refinement parameters change by less then |DeltaP/Sigma| convergence is reached. Defaults to 0.005 conf:<level> If the confidence level is greater than <level> convergence is reached. Defaults to 0.010 chi2:<level> If the value of Chi**2/(Ndata-Npara) falls below <level> convergence is reached. Defaults to 0.500
Convergence is reached, if: Either (dchi AND pshift AND conf) or chi2 are met.

relax

```
set relax  ["start:"<lamda_s>] [,"fail:"<lamda_f>]
           [,"success:"<lamda_g>]
```

Allows to fine tune the relaxation behavior. The initial Levenberg-Marquardt lamda parameter is defined with ["start:"<lamda_s>]. Default is 0.001
If a cycle was successful, the value of lamda is multiplied by <lamda_g>, default is 0.5.
If a cycle was not successful, the value of lamda is multiplied by <lamda_f>, default is 4.0.
A larger value of lambda implies smaller steps for the refinement parameters.

A.12 sigma

```
sigma "kuplot", <number>
sigma <type>,<infile>
sigma "kuplot", <number>
```

The calculation will refine parameters versus the observed data and use sigmas that are stored in the KUPLOT data set number <number>

```
sigma <type>,<infile>
```

The calculation will refine parameters versus the observed data and use sigmas that are loaded from file <infile>. See the ==> 'kuplot/load' command for details on proper ways to load a data set.
Refer to the KUPLOT section for instructions on loading data.

Bibliography

Levenberg. A method for the solution of certain problems in least squares. Quart. Appl. Math. **2** (1944) 164 – 168.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11** (1963) 431–441.

W. H. Press; B. P. Flannery; S. A. Teukolsky; W. T. Vetterling. *Numerical Recipes; The art of Scientific Computing*. Cambridge University Press, Cambridge 1989.

K. V. Price; R. M. Storn; J. A. Lampinen. *Differential Evolution; A Practical Approach to Global Optimization*. Springer, Berlin 2005.