REFINE

Users Guide

Version 5.99

written by

Reinhard Neder
Email: reinhard.neder@fau.de

http://tproffen.github.io/DiffuseCode

Document created: October 17, 2019

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 *DIFFEV*. 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

1 Introduction **3**
1.1 What is REFINE ? . . . 3
1.2 Getting started . . . 3
1.3 Command language . . . 4

2 Least Squares Refinement **5**
2.1 Refinement via least squares refinement . . . 5

3 Example refinements **7**
3.1 Simple noisy data function . . . 7

A REFINE commands **9**
A.1 Summary . . . 9
A.2 Example . . . 9
A.3 data . . . 10
A.4 fix . . . 10
A.5 newparam . . . 10
A.6 set . . . 11
A.7 sigma . . . 11

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 **?**. 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 manual or check the online help using (`help command input`). Names of input or output files

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

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. Currently there are no REFINE specific variables.

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.

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.

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 from 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 !1.1
 4: newparam, P_lin  , value:1.01  !-1.2
 5: newparam, P_quad , value:1.49  !1.3
 6: newparam, P_trip , value:0.31 !0.3
 7: set  cycle,   5
 8: set conv, dchi:0.50, pshift:0.005, conf:0.10, chi2:0.5
 9: run triple.mac
10: 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.

```
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
```

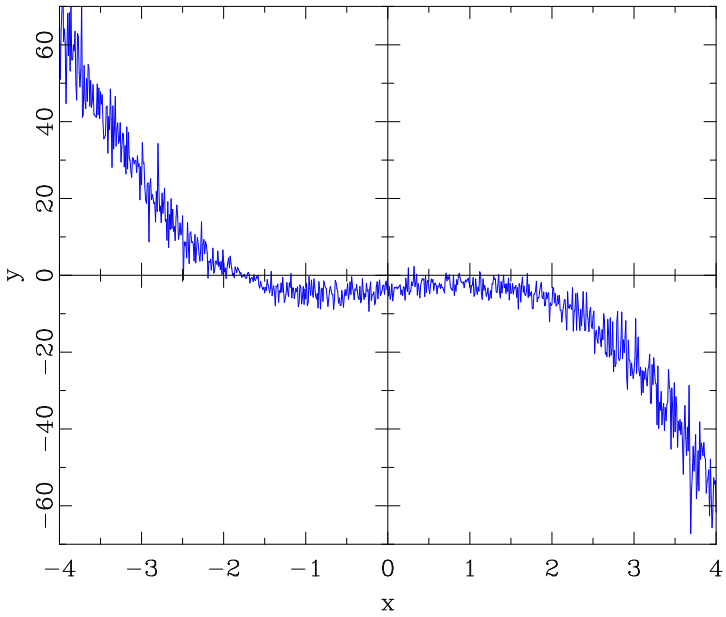

Figure 3.1: Experimental polynomial function

```
  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
```

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.

```
News          : Information on program updates
```

A.2 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.3 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.4 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.5 newparam

newparam <**parname**> **[,"value:"**<**start**>**] [,"status:"**<**flag**>**]**
[,"range:["<**lower**>**,**<**upper**>**"]"]**

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. 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.

A.6 set

```
set cycle,<maxc>
set conver [,"dchi:"<delta>] [,"pshift:"<max>] [,"conf:"<level>]
```

set cycle,<**maxc**>

Sets the maximum number of refinement cycles
set conver [,"dchi:"<**delta**>**] [,"pshift:"**<**max**>**] [,"conf:"**<**level**>**]**
[,"chi2:"<**level**>**]**

Allows the user to define convergence criteria.
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.

A.7 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.