Discus Suite

Users Guide

Version 5.14

written by

Reinhard Neder
Email: reinhard.neder@fau.de

http://tproffen.github.io/DiffuseCode

Document created: November 27, 2017

Preface

Disclaimer

The DISCUS SUITE 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 the DISCUS SUITE. No warranty is made with respect to this manual, or the program and functions therein. There are no warranties that the programs are free of error, or that they are consistent with any standard, or that they will meet the requirement for a particular application. The programs and the manual have been thoroughly checked. Nevertheless, it can not be guaranteed that the manual is correct and up to date in every detail. This manual and the DISCUS SUITE program may be changed without notice.
DISCUS SUITE 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 DISCUS SUITE

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

> NEDER, R.B. in preparation - check website.

More information

This users guide can only provide program specific details. A broader discussion of simulation techniques and some DISCUS SUITE 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 DISCUS_SUITE ?

The DISCUS SUITE is a combination of the components of the DISCUS program suite. It includes the stand alone programs DISCUS, DIFFEV, and KUPLOT under a common command language. In this combination, the DISCUS SUITE allows you to seamlessly switch between the individual sections. Its main advantage lies in the field of massive parallel computing. As the DISCUS SUITE includes the full functionality of DIFFEV, the DISCUS SUITE allows you to perform parallel refinements. In contrast to a stand alone DIFFEV, the DISCUS SUITE does have to perform nearly no I/O during the refinement.
See the DIFFEV manual for full details of the refinement process.

1.2 What is new ?

October, 2016
Starting with versions 5.7.0 new variables have been added to make the communication between the DIFFEV section and the DISCUS and KUPLOT sections more transparent. These variables automatically receive from the DIFFEV section the respective values. These variables are:

Entry	Description
REF_GENERATION	Current generation number
REF_MEMBER	Size of the population
REF_CHILDREN	Size of the child population
REF_DIMENSION	Number of parameters
REF_KID	Current child to work on
REF_INDIV	Current individual repetion of the current child
ref_para[*]	The trial values for the current child

A fully operational version of MPI distributed DISCUS SUITE is provided with the windows release as well.

1.3 Getting started

After the program *DISCUS_SUITE* has been installed properly and the environment variables are set, the program can be started by typing 'discus_suite' at the operating systems prompt.

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

Table 1.1: Used symbols

The program uses a command language to interact with the user. The command `exit` terminates the program and returns control to the shell. All commands of DISCUS SUITE consist of a command verb, optionally followed by one or more parameters. All parameters must be separated from one another by a comma ",". There is no predefined need for any specific sequence of commands. DISCUS SUITE is case sensitive, all commands and alphabetic parameters MUST be typed in lower case letters. If DISCUS SUITE has been compiled using the `-DREADLINE` option (see installation files) basic line editing and recall of commands is possible. For more information refer to the reference manual or check the online help using (`help command input`). Names of input or output files are to be typed as they will be expected by the shell. If necessary include a path to the file. All commands may be abbreviated to the shortest unique possibility. At least a single space is needed between the command verb and the first parameter. No comma is to precede the first parameter. A line can be marked as comment by inserting a "`#`" as first character in the line.
The symbols used throughout this manual to describe commands, command parameters, or explicit text used by the program DISCUS SUITE are listed in Table 1.1. There are several sources of information, first DISCUS SUITE has a build in online help, which can be accessed by entering the command `help` or if help for a particular command <cmd> is wanted by `help <cmd>`. This manual describes background and principle functions of DISCUS SUITE and should give some insight in the ways to use this program.
DISCUS SUITE is distributed as part of the diffuse scattering simulation software DISCUS. In contrast to, DIFFEV, DISCUS SUITE does not lend itself as efficiently as a general refinement program separate from the DISCUS program package. DISCUS SUITE handles a structure refinement via the internal DISCUS and KUPLOT sections.

1.4 Parallel execution

The typical application of the DISCUS SUITE is a run on a massive parallel system. You will have to check for details with your local administrator. In general you should expect to start DISCUS SUITE along the lines of:

```
mpiexec -n 192 discus_suite -macro suite.mac > /dev/null
```

The command `mpiexec` initiates the MPI system. Here in this example we request 192 CPUs `-n 192`. The MPI system then starts the DISCUS SUITE and read its input from the macro `suite.mac`. All output is discarded to `/dev/null` in order to minimize I/O. During development of your macro you might redirect the output to a logfile.
The example line above works fine on a single multi-core computer with Linux. The number of processes would be more reasonably placed around the number of CPUs on your computer. Within the Windows version, parallel execution of such a macro is performed by DISCUS SUITE command *parallel*:

```
parallel suite.mac
parallel 6, suite.mac
```

The first command version instructs DISCUS SUITE to run a parallel refinement via the macro *suite.mac*. The number of parallel threads is determined automatically. Alternatively you can tell DISCUS SUITE how many parallel threads you would like to run by adding the number of threads prior to the macro name. In the example above DISCUS SUITE is instructed to use 6 threads.

1.5 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. At the moment, DISCUS SUITE does not include any specific variables.

Chapter 2

Concepts

2.1 Quick overview

The purpose of the DISCUS SUITE is to run a combination of DIFFEV, DISCUS, and KUPLOT within a single program. It main application will be a run on a massive parallel system, but it will already pose an improvement on a multi-core laptop. DISCUS SUITE uses MPI to distribute a part of its workload onto several processes. The original DIFFEV starts DISCUS as a separate slave program. This means that at each refinement cycle DISCUS has to read all macros, and other relevant information over and over again. This causes a lot of unnecessary I/O, which puts a considerable burden onto the central disk system.
As DISCUS SUITE includes DIFFEV, DISCUS, and KUPLOT within one single program, all relevant input can be read once at the beginning of the refinement cycle. It will from there on be kept in internal memory, thus eliminating the need of further input. This includes the trial parameters as well as the result values. In the stand alone program DIFFEV, these values are passed from DIFFEV to and forth between DISCUS and KUPLOT via (small) files on the disk. Within DISCUS SUITE, the need to write these files onto the disk no longer exists and DISCUS SUITE instructs DIFFEV to store these values only internally.
As the main tasks are still performed within the sections DIFFEV, DISCUS, and KUPLOT, the actual set of commands within DISCUS SUITE is very limited. Besides the general command of the common command language the commands are just the names of the three sections, Table 2.1.
Within the three sections, a few details are slightly different compared to their stand alone versions. The main command syntax and typical command sequences are, however, identical.

Command	Description
diffev	Switch to the `diffev` section. in many cases this will be the only command used within the top menu of the suite.
discus	Switch to the `discus` section.
kuplot	Switch to the `kuplot` section.
parallel	Run a parallel refinement.

Table 2.1: DISCUS SUITE command verbs.

As DISCUS SUITE is a single program, the global variables `i[], r[], res[]` are identical throughout all sections. The same holds for user defined variables. Thus, for example, if `i[0]` is set to one value in any section, all other sections will see this new value. Be careful when you combine old DISCUS and KUPLOT macros. Their local variables my interfere with each other.
iBesides the three variables `i[], r[], res[]` and the user defined variables, each of the three section includes specific variables that are valid and meaningful witthin this section only. These variables specific to any of the sections are visible within that section only.

2.2 Modified concepts

In order to fully utilize the common programm architecture, a few details should be used in a modified way within the different sections. These details mostly concern disk I/O related parts.

2.2.1 DIFFEV

The stand alone DIFFEV communicates with DISCUS and KUPLOT via (small) files that contain the trial parameters and the cost function results. To avoid writing these to disk use:

```
init silent
```

The new `silent` parameter tells DISCUS SUITE not to write an initial set of trial parameters to disk. These values are automatically transfered to the slave sections.
Closely related is the modified comand:

```
trialfile silent
```

wich likewise tells DISCUS SUITE not to write trial files during the refinement cycles.
In both cases DISCUS SUITE will use the internal variable `ref_para[]` to communicate the trial parameters to the slave sections. Entries in this array range from 1 to the number of parameters to be refined, as defined by tt pop_dimx in DIFFEV. As of Versions 5.7 the data are still written to the array, `r` as well. As a deliberate break compared to the general flexibility in all DISCUS SUITE sections, the trial values are stored in entries starting at `r[201]`. Use of these entries is meant to preserve backwards compatiability but will be discontinued in future releases.
Another predefined section of the internal variables is the range:

Entry	Description
`i[201]`	Current generation number
`i[202]`	Size of the population
`i[203]`	Size of the child population
`i[204]`	Number of parameters

The use of this predefined section of the internal variables is discouraged and has been replaced by the variables:

Entry	Description
REF_GENERATION	Current generation number
REF_MEMBER	Size of the population
REF_CHILDREN	Size of the child population
REF_DIMENSION	Number of parameters
REF_KID	Current child to work on
REF_INDIV	Current individual repetion of the current child

To retain the backwards compatibility, the actual child number and the number of the individual repetition are transfered as parameters on the `run_mpi` command line.
In the DISCUS SUITE version of KUPLOT, a new internal variable is used to store and transfer the weighted R-value from KUPLOT to DIFFEV. Thus, the need to write and read the result file ceases to exist. Do this via the commands:

```
resultfile  silent
...                ! further initialization
run_mpi ...        ! Calculate with Discus und Kuplot
compare silent
```

The first and last lines instruct the DIFFEV section not to read the resultfile but to use the value that is transfered internally.
MPI has been implemented in the current Windows installation as well starting with Version 5.7. Use the icon `DISCUS_Suite Parallel_Version` for parallel refinements.
Even in the single session version of the DISCUS SUITE, the refinement shoud use the `run_mpi` command in an identical fashion.
In order to use the suite efficiently, the `branch` command is avaliable within DIFFEVas well, and you can branch to either DISCUSor KUPLOT . This command should replace the `system discus < discus_macro_name` construction that is used in a serial refinement set up:

```
 1  diffev
 2     ... SETUP ...
 3     init silent
 4     branch discus
 5        @discus_macro.mac
 6        exit
 7     branch kuplot
 8        @kuplot_macro.mac
 9        exit
10     compare silent
11  exit
```

In this simplified example, we switch to the diffev section with the `diffev` command in line 1. After initial setup done by DIFFEV and the initialization of generation zero in line 3, the DIFFEV section branches off to the DISCUS section in line 4. Now control is at DISCUS and the loop over all children and all calculations are performed by the macro `diffev_macro.mac` in line 5. The `exit` command in line 5 returns to the DIFFEV section. If you use a previous DISCUS macro, the `exit` line might be part of the DISCUS macro! Lines 7 to 9 do the same for KUPLOT. The `compare` command (line 10) at the DIFFEV level does the comparison between the generations, and finally we leave the DIFFEV section in line 11. In a full refinement there would be a loop over all refinement cycles that includes lines 4 to 10 of this example.

2.2.2 DISCUS

The performance of DISCUS is not directly affected by the DISCUS SUITE. To make full use of the DISCUS SUITE, a couple of concepts should be adhered to.
Read the trial values from the real valued array `ref_para[]`, where the trial values are stored in elements 1 and following. Use the variables REF_GENERATION, REF_MEMBER, REF_CHILDREN, REF_KID, REF_INDIV to obtain the current generation number, the size of the population, the number of children and the number of trial parameters. Do not open and read the file GENERATION. This file is still created within each refinement cycle, there is, however, no longer any need to read it.
If the initial asymmetric unit is not extremely long, it is best to build the initial asymmetric unit from scratch. To build a Wurtzite type asymmetric unit use the following command sequence:

```
1   read
2      free ref_para[1], ref_para[1], ref_para[2], 90.00, 90.00, 120.00, P63mc
3   insert Zn, 1./3.,  2./3, ref_para[3], ref_para[4]
4   insert S , 1./3.,  2./3, 0.00  , ref_para[4]
5   save
6      outf internal.wurtzite.cell
7      omit all
8      write scat
9      write adp
10     run
11  exit
12  read
13     cell internal.wurtzite.cell, 5,5,5
```

In this example it is assumed that ref_para[1] contains the a lattice parameter value, ref_para[2] the c lattice parameter, ref_para[3] the z-position of Zn and ref_para[4] a common atomic displacement parameter B. The last parameter in line 2 tells DISCUS that the structure belongs to space group P63mc. The asymmetric unit is stored internally into computer memory in lines 5 to 11, and then expanded into a block of 5x5x5 unit cells in lines 12 and 13. By providing the space group symbol as last parameter to the `free` command, the expansion in line 13 will ensure that the two atoms Zn and O are properly copied to create all atoms in the unit cell. This macro section does not perform any disk I/O and will be much more efficient on a large parallel system.
Do use the internal storage whenever you need to save a temporary crystal structure. Everytime a file name is preceded with the string `internal`, DISCUS will save / read the structure to / from internal memory.
To perform any further manipulations with the calculated data like multiplication by a scale factor, the addition of a background etc., KUPLOT needs to be executed once the DISCUS section is finished. To allow efficient use of massive parallel systems the new `branch` command has been added to DISCUS and KUPLOT. It allows to switch directly from the DISCUS section to the KUPLOT section within DISCUS SUITE.

```
0   ...                        ! Discus initialization
1   do indiv=1,nindiv          ! Repeat nanoparticle generation nindiv times
2      @discus.zno.mac kid     ! do the actual simulation
3   enddo
4   branch   kuplot            ! Switch to KUPLOT
5      @kup.diffev.mac ., kid  ! Contains a final exit
6   exit                       ! Finish DISCUS
```

In this DISCUS macro section the simulation of nanoparticles is repeated several (nindiv) times. This might be necessary to average out the defect distribution within the nanoparticles. Under these circumstances the individual calculated diffraction pattern / PDFs need to be averaged after the DISCUS section is finished. The branch to the KUPLOT section in line 4 allows DISCUS to do this while remaining on the identical slave process and thus on the identical compute node. On many massive parallel systems data transfer from a compute node to the central disk is time consuming and many of these systems have local fast (RAM) disks. The DISCUS section can write the individual diffraction pattern / PDF to this local disk without much of a penalty. But then, KUPLOT needs to operate on the same node in order to find the data. This is ensured by the `branch` construct.
In the alternative set up :

```
0   ...                          ! Diffev initialization
1   run\_mpi discus, discus\_macro.mac  " Discus calculation
2   run\_mpi kuplot, kuplot\_macro.mac  ! Kuplot calculation
```

DIFFEV will distribute the DISCUS jobs onto all nodes, and then do the same with the KUPLOT jobs. It is not straightforward to ensure that the two jobs for a given child are performed on the identical node.

2.2.3 KUPLOT

iIn the same fashion as DISCUS , the KUPLOT section should rely on the variables `ref_para[]` REF_GENERATION REF_MEMBER, REF_CHILDREN, REF_KID, ref_INDIV to determine the population size and if necessary the trial parameters. Do not read the GENERATION file.
Do not write a result file. The KUPLOT instruction:

```
rvalue 1, 2
```

stores the weighted R-value internally and instructs the DISCUS SUITE to transfer the value from the slave process back to the master process. The master process can then use this in the `compare silent` command.
The `branch` command is avaliable within KUPLOT as well, and you can branch to either DIFFEV or DISCUS . Most application will probably not need to do this.

Chapter 3

Example

This chapter takes you through all steps of a nanoparticle refinement. The example will use data published in Chory et al., which describes the refinement of ZnSe nanoparticles. We will quickly describe the main aspectes of the model that is used to describe the experimental data. The main focus will be on the use of the DISCUS SUITE and on the differences to a refinement with DIFFEV.
The ZnSe nanoparticles can be described as a disordered Zincblende structure. Stacking faults introduce a local Wurtzite structure. TEM images showed that the particles are of approximately ellipsoidal shape with one long and two short axes. Bulk ZnSe crystallizes in the Zincblende structure, and these nanoparticles show a typical high defect concentration. For this example we choose to describe the structure as a stack of layers based on a hexagonal metric. Thus the stacking faults are restricted to one of the symmetrically equivalent [111] axes of the bulk cubic structure. A perfect ABCABC sequence of these layers would correspond to the cubic Zincblende structure, a perfect ABAB sequence would yield the hexagonal Wurtzite structure. For further details see Chory et al. and Neder and Proffen (2007).

3.1 Parallel refinement

A main advantage of DISCUS SUITE compared to the stand alone programs is achieved in a parallel refinement. A stand alone refinement with DIFFEV needs to start DISCUS and KUPLOT as slave programs within each refinement cycle. Thus at each program start, the macros and the experimental data need to be read again. Furthermore, these separate programs need to communicate the results via files on the hard disk, which slows down the performance. As the DISCUS SUITE is a single program, internal communication is straighforward and the makes most disk I/O obsolete. The DISCUS SUITE performs its taks within the individual sections and thus most parts of the actual refinement macros do not need to be changed.
To run in parallel mode, the DISCUS SUITE must have been compiled with the MPI option. The suite will then be started with a command like:

```
mpiexec -n 192 discus_suite -macro suite.mac  > /dev/null
```

If you use the Windows operating system, start a parallel refinement via:

```
parallel suite.mac
```

see chapter 1.4 for further info on the invocation.
The main DISCUS SUITE macro can be very short:

```
1   diffev
2      @diffev.mac
3   exit
4   exit
```

In line 1 we switch to the DIFFEV section, perform the -slightly modified - macro `diffev.mac` and return to the main DISCUS SUITE menu via the `exit` in line 3. In line 4 we finally finish the suite itself.
This macro nesting is not a necessary feature. You could of course place all these lines into the `diffev.mac` macro as well.
The `diffev.mac` macro needs just few changes compared to a stand alone version:

```
 1    set error,exit
 2    @cleanup.mac
 3    #
 4    @setup.mac
 5    @diffev_setup.mac
 6    #
 7    init silent
 8    #
 9    do i[0]=1,100
10       echo "GENERATION %4d",REF_GENERATION
11       run_mpi discus, nano.znse.mac  , 0     , /dev/null ! LOGFILES/d
12       compare silent
13    enddo
```

In line 1 we instruct DISCUS SUITE to be picky with errors and to stop the execution is any errror should occur. I do not want to run a refinement for several hours just to find out that somewhere there there has been an erroneous calculation.
The *cleanup.mac* macro may be used to remove all files related to an earlier refinement.
The *setup.mac* macro is used to define the number of individual repetitions that DISCUS should perform for each member of the population.
Macro *diffev_setup.mac* defines the size of the population, the number and range of the parameters to be refined etc.
In line 7 we initialize the parameters to their starting values. The qualifiere *silent* instructs the DISCUS SUITE not to write these trial parameters onto the disk. They will instead be passed silently between the DIFFEV and DISCUS sections.
The *run_mpi* command in line 11 starts the simulation and the assesment of the current agreement between experimental and calculated data. The DIFFEV section will start the DISCUS section and within DISCUS operate the macro *nano.znse.mac*. The next parameter, here a zero, tells DIFFEV if the individual repetitions are to be carried out in parallel or if they are to be done in serial operation by the DISCUS section. If the parameters is zero, as in this example, any repetition is left to the DISCUS section. If the parameter is larger than one, DIFFEV will calculate the individual repetions in parallel on your system.
Finally lin line 12 we tell DIFFEV that the final aggrrement factor between experimental data and the calculated data shall not be read froma tiney file on your disk. Instead KUPLOT will pass this parameters silently back to DIFFEV.

Appendix A

SUITE commands

A.1 Summary

You can switch to the individual sections "discus", "diffev", and "kuplot" by typing the respective section name. To return to the suite type "exit" at the main menu of each section.
The variables i[*], r[*] and res[*] are global variables, a change in any section will be seen in any other section as well. The same holds for all user defined variables!
The section specific variables are local within each section.
If an output filename in "discus" starts with "kuplot", the data are written directly into the next available KUPLOT data set. This is available for Fourier output, powder, pdf.

A.2 News

2017_Sep

Throughout the program the internal calculation of random numbers was changed to the FORTRAN 90 intrinsic function.

2016_Dec

At a few select points colors are introduced into the output. Currently these are just the error messages.

2016_Oct

A new command 'parallel' has been added to the Windows version This allows to execute a macro in parallel.

2016_June

The SUITE may now be interruted gracefully with a CTRL-c. This will cause the DISCUS part to write the current structure, and DIFFEV to shut down MPI if active.

2015_December

The branch command within the sections discus, diffev, kuplot may now take the form : branch discus -macro macro_name par1, par2, ...

2015_June

Starting with Version 5.1, we have migrated to a X-Window environment for WINDOWS as well. As a small side effect, the technique to jump to the desired folder has changed slightly. See the help entry on "cd" in the general "Command_lang" section for further information. The process is described in the package manual as well.

A.3 diffev

Switches to the "diffev" section.
Within this section any standard DIFFEV command can be given. The behaviour of "diffev" is essentially the same as in the stand alone version. The 'diffev/run_mpi' command will start a discus/kuplot section. The syntax of the command is unchanged. All trial parameters are placed into array <r[]> at entries 201, 202, ... The values of generation, member, children and number of parameters are placed into <i[]>: i[201] = generation i[202] = member i[203] = children i[204] = parameter
Use an 'exit' to return to the suite.

A.4 discus

Switches to the "discus" section.
Within this section any standard DISCUS command can be given. The behaviour of "discus" is essentially the same as in the stand alone version.
Within the discus section you can use the command 'branch kuplot' to switch to the kuplot branch.
In contrast to the stand alone DISCUS version, one can write an output file directly into the KUPLOT data sets. The number of data sets in KUPLOT is atumatically incremented. Currently this is implemented for the PDF and the powder output. Single crystal diffraction pattern to follow shortly.

A.5 kuplot

Switches to the "kuplot" section.
Within this section any standard KUPLOT command can be given. The behaviour of "kuplot" is essentially the same as in the stand alone version.
Within the kuplot section you can use the command 'branch discus' to switch to the discus branch.

A.6 parallel

parallel {<**numprocs**>, },<**macro.mac**> {, <**para_1...**>}

Starts an MPI driven parallel calculation. See the diffev help on a full explanation of parallel processing.
The parallel refinement will execute file <macro.mac>, which must reside in the current directory. Make sure you have used cd <path> to change to the proper directory prior to the use of the 'parallel' command. The macro name must be given in full, including the ".mac" extension. If the macro requires parameters you must specify these following the macro name.
Optionally you can place the number or processes that MPI shall start prior tot he macro name. The numebr defaults to the value of the SHELL variable NUMBER_OF_PROCESSORS on your system. If this variable is not set, discus_suite will start 4 processes.

Bibliography

C. Chory; R. B. Neder; V. I. Korsunskiy; F. Niederdraenk; Ch. Kumpf; E. Umbach; M. Schumm; M. Lentze; J. Geurts; G. Astakhov; W. Ossau; G. Meuller. Influence of liquid-phase synthesis parameters on particle sizes and structural properties of nanocrystalline ZnO powders.

R.B. Neder; Th. Proffen. *Diffuse scattering and defect structure simulations - A cook book using the program DISCUS* . Oxford University Press, Oxford, UK 2007.