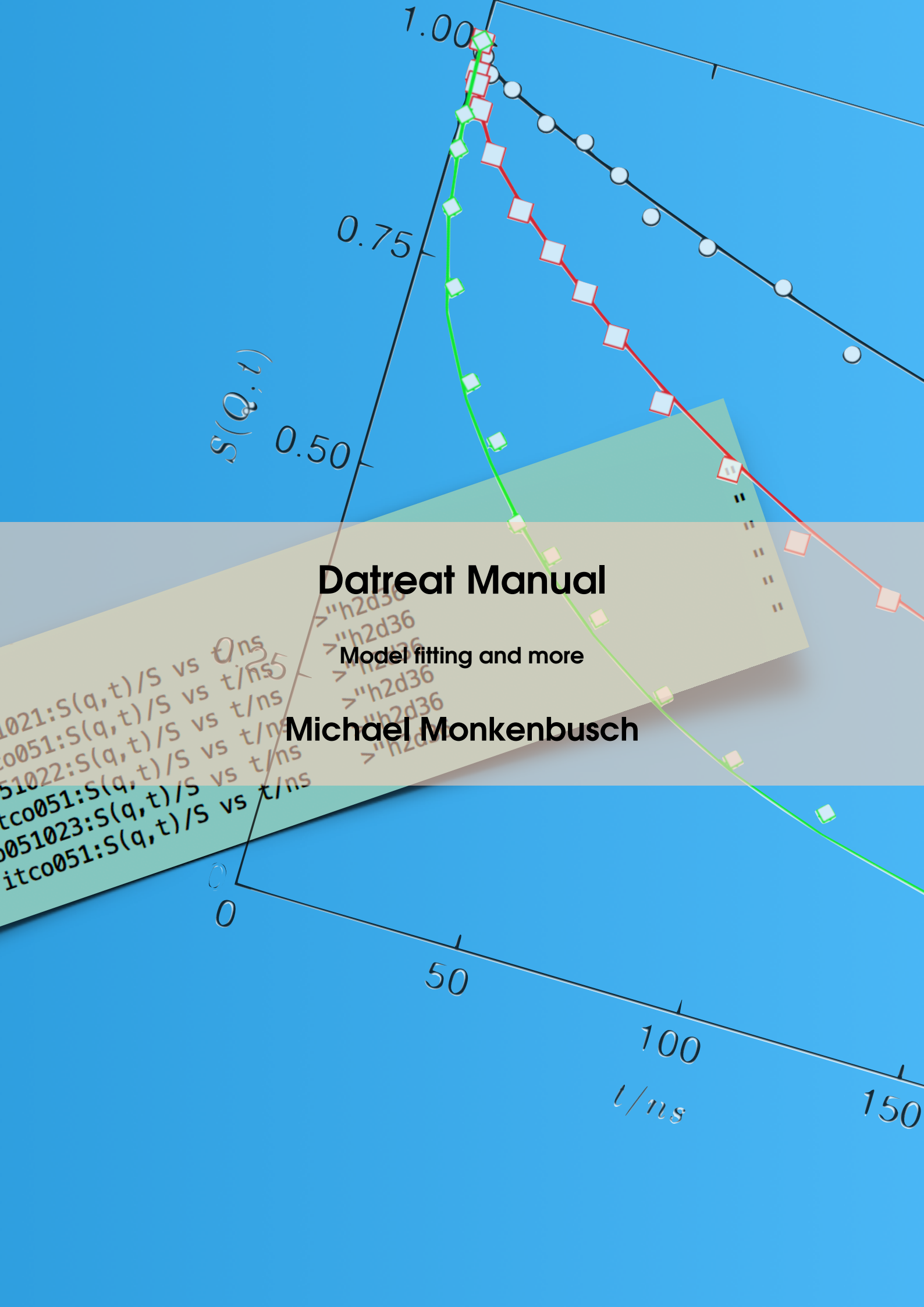


Datreat Manual

Model fitting and more

Michael Monkenbusch

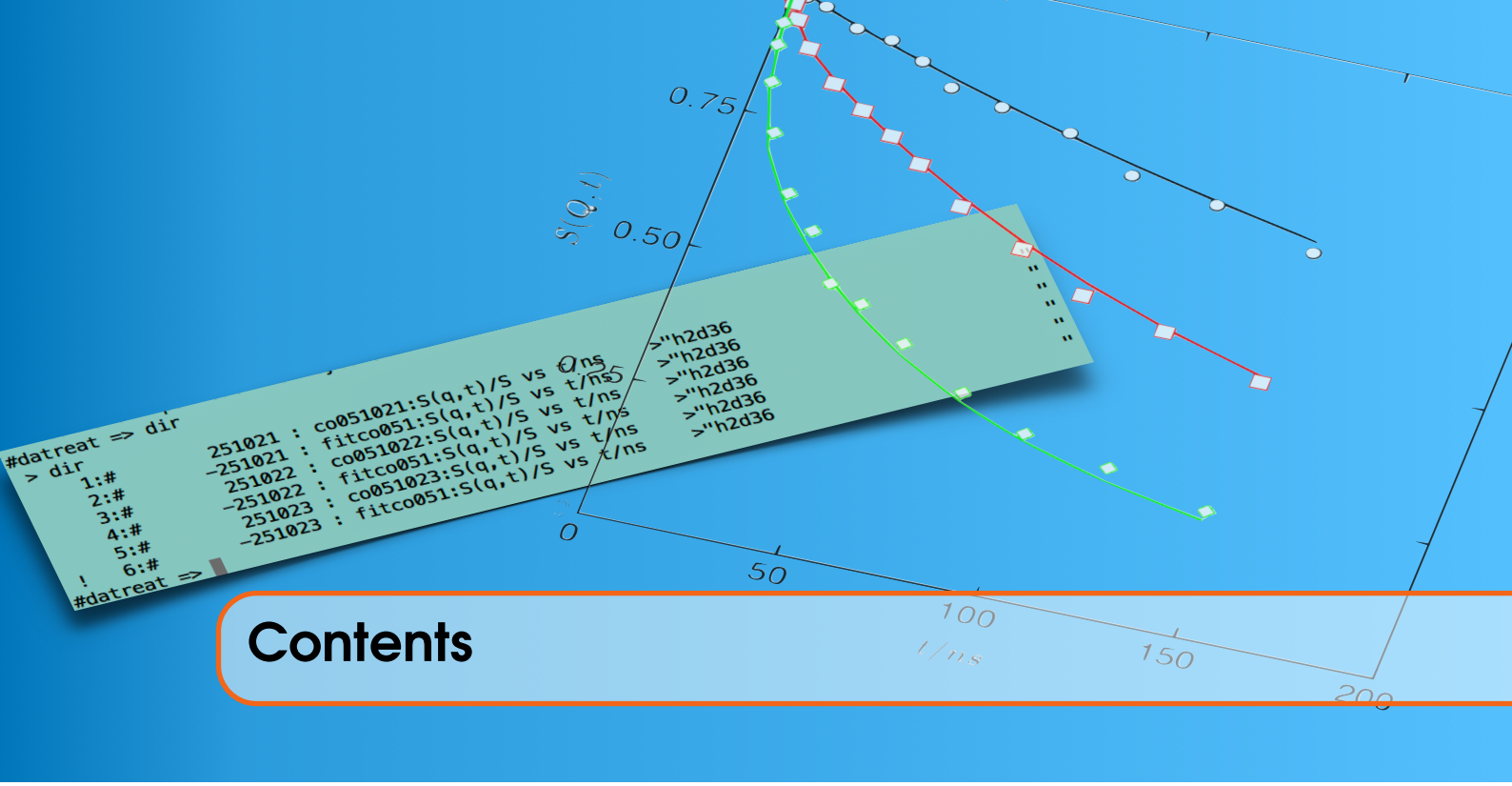
Copyright © 2019 M.M, Forschungszentrum jülich

PUBLISHED BY PUBLISHER

BOOK-WEBSITE.COM

Licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License (the "License"). You may not use this file except in compliance with the License. You may obtain a copy of the License at `http://creativecommons.org/licenses/by-nc/3.0`. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

First printing, October 23, 2023



Contents

I General

1 Introduction: read this first! 11

1.1 Purpose and general modes of use 11

1.2 Manual Notation conventions **11**

1.3 Basics of use data structure and operations **11**

1.4 The command line components of the command input **11**

1.5 Datreat data structure, variables and built-in functions 12

1.5.1 Variables and functions user defined variables, datreat variables and functions 13

1.5.2 Global functions 13

1.5.3 Expressions 14

1.6 Customisation (Makros) How to combine commands **15**

1.6.1 Makros specific commands 15

1.7 Necessary and useful UNIX commands (Mac) 16

2 First steps 17

2.1 Starting and quitting the program 17

II Input, records and display

3 Data management I/O 21

3.1 input *filename* input files and data format **21**

3.2 dir (pname) listing data holding records 21
3.3 purge (all | rest | sel) 22
3.4 msave *filename* save data and fit results 22
3.5 inplus *filename* read msave-stored data with theory model 23
3.6 putpar adding/setting data-record associated parameters 23
3.7 copy copy records in memory 23
3.8 allsave save current datreat state to subdirectory 24
3.9 allresto restore cdatreat state from a subdirectory 24

4 Select 25
4.1 sel (spec) select data records for next actions 25
4.2 dsl display list of selected records 26
4.3 activate theory model activation 27
4.4 acl Loading previous theory model 28

5 Data inspection and plotting 29
5.1 plot basic data display 29
5.2 plot (styling) Sizes, colors and legends 30
5.3 rename xaxis *new-label-x* | yaxis *new-label-y* rename axes (plot +) 31
5.4 title *Any title* set a title (plot) 31
5.5 plot (save and print) 31
5.6 edit (j) Editing data in a memory stored record 32

III Data reduction

6 Combination, averaging and clipping 35
6.1 average xcatch δx (absolute) (log_y) 35
6.2 rebin nk 37
6.3 rerange r_1 r_2 (y) 37
6.4 alingny (offset o yscale ys ostep do) ystep dy automatically adjust scales 37

7 Transformation: applying functions to x and y 39
7.1 fxy (op) x=*(expressionX)* y=*(expression)* 39

8 Arithmetic between data sets (records) 41
8.1 arit f1 *(factor1)* f2 *(factor2)* (mult|div|vid) (to *numor*) 41
8.2 addsels 42
8.3 recsum from *rec1* to *rec2* 42

9 Special function scaling (spatcorr) 45
9.1 spqscal 45

IV Fitting

10 Model fitting 49

10.1 th (name) display information on available theories 49

10.2 activate *name* (multiply | insert) (range ...) Add a theory to the active list 50

10.3 cth Change/set (edit) theory model parameters 51

10.4 thc compute the theory model 52

10.5 fit Fitting the model parameters to data 53

11 Advanced fitting topics 55

11.1 Coupling of parameters and record selective theories 55

11.2 theory range *pnam* min *p1* max *p2* selective application to records 57

11.3 chgthpar *var p* scale *fs* command line fit parameter setting 57

11.4 label & couple command line setting parameter couplings 58

11.5 putpar adding/setting data-record associated parameters 58

11.6 extthpar *var p* extract value of a specified model fit-parameter 59

11.7 print *filename par1* ... *par9* saving parameters etc. of a file 59

11.8 table *tabname* (open | | close) *par1* ... *par...* creating a tex-table containing some parameters 59

11.9 serfit vs *parname* (par1) ... serial fitting of records 61

11.10 mexp Automatic fitting of a sum of exponential 62

12 Creating and modifying theory models 63

12.1 Selecting among available theories from the UNUSED collection 63

12.2 Create a NEW theory: Define the problem 63

12.3 Programming Language 64

12.4 Use the template generator 64

12.5 Conversion to the proper Fortran theory function 65

12.6 Test and use How to use a new theory 66

12.7 Code Generated illustration for this example 66

12.8 Hints How to efficiently create and debug theories 67

V Special Issues

13 SANS and multiple scattering 73

14 TOF and BSS spectra 75

14.1 uni_ft Fourier transform of spectra 75

14.2 restran transfer of resolution function for $S(q,\omega)$ fitting 75

15 NSE instrumentation/design 77
15.1 echocurv simulated phase scan 77

16 Development tools 81
16.1 noise add Gaussian noise and errors to a record 81

VI Working Examples

17 Recommended workflows 85
17.1 Open a terminal from there datreat may be started 85
17.2 Directories "where" to do the work 85

18 Workflows: examples 87
18.1 A simple start basic usage example 87
18.2 Automation and makros prepmak: a more complex makro example 88
18.2.1 The shell script cotxtdtr 91

VII Theories

19 strexpo generic streched exponential 95

20 gauss, gauss2, gaussft Gaussians 97

21 Spectral functions in ω General aspects for resolution broadened spectra. 99
21.1 Determination of Gaussian resolution parameters 100

22 kohlfast Spectrum of Kohlrausch fktn. 109

23 sqt_om1 S(q,t) for system with labelled crosslinkers 111

24 nrouse Rouse single chain structure factor 113

25 ring2b Ring single chain structure factor (Goossen,PRL 2014) 115
25.1 TO BE MODIFIED... 117

VIII Installation

26 Installing datreat (on Mac) 121
26.1 Preliminaries: compiler and gr-software system preparation 121
26.2 Directories and paths 121

26.3 Datreat making datreat, build **122**
26.3.1 Copying 122
26.3.2 Compiling 122
26.3.3 Choosing a favorite editor 122

Bibliography 127
Articles **127**
Books **127**

Index 129

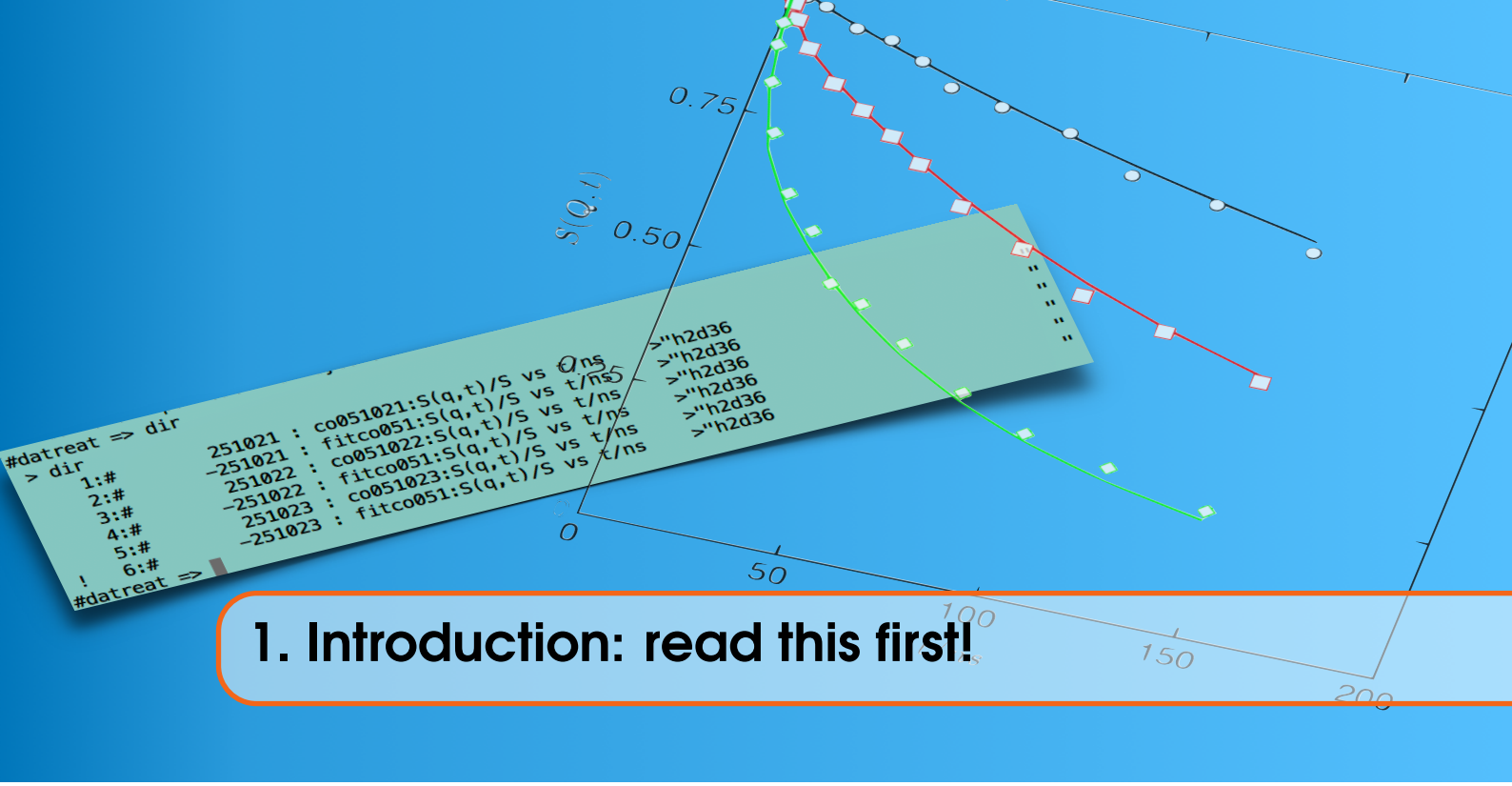
General

1 Introduction: read this first! 11

1.1 Purpose and general modes of use
1.2 Manual Notation conventions
1.3 Basics of use data structure and operations
1.4 The command line components of the command input
1.5 Datreat data structure, variables and built-in functions
1.6 Customisation (Makros) How to combine commands
1.7 Necessary and useful UNIX commands (Mac)

2 First steps 17

2.1 Starting and quitting the program



1. Introduction: read this first!

1.1 Purpose and general modes of use

1.2 Manual Notation conventions

Any input or parts of **input** that is **literal** is written **like this**.
Any input of *variable* information/values is written *like this*.
Any **volatile** inputs of information/values is written **like** *this*.
Output from datreat is written **like this**.
Optional inputs are indicated [like this].

1.3 Basics of use data structure and operations

The data "atoms" (called **records**) are curves/tables of (x,y,y-error) values augmented by a comment, x- and y-axis labels and a set of parameters that contain metadata. Each **record** occupies one slot j in the internal data storage and can be addressed by this slot number. In order to perform some *action* of processing, display etc. the corresponding **records** must be **selected**. Depending on the required action **one or several records** must be **selected**. Depending on the performed action the *selection* either is kept or is transferred to newly created **result records**. This should be mentioned in the execution message issued by the processing command. The memory contents and the selection state can be checked using the commands **dir** () and/or **dsl** (see their description for more detail).
Fitting requires the specification of a **theory** model, which can be combined form a number of available models, more advanced modelling can be achieved by adding own models. For details on model fitting see chapter *Fitting*.
Inspection of **records**, fit results etc. is bets done using the command **plot** .

1.4 The command line components of the command input

If datreat is ready it shows a prompt: #datreat => , then in can be controlled by entering a command line. Command lines start with a command followed by a number of keywords and values.

- **cmd**
- **cmd keyword** *value*
- **cmd option**
- **cmd** *value*
- **cmd** *value value*

cmd	command name, it is the first item of the command line
" "	one or several blanks separate any items
keyword	name like item that serves as identifier for options or following value(s)
Value	numerical or literal value (depends on keyword)
&	command separator allows to enter several command in one line

> **plot xmin** *x1* **xmax** *x2*

1.5 Datreat data structure, variables and built-in functions

data source	j_s	index of active source data record
result dest.	j_r	index of resulting data record
selected set	S	set of active source data records (indices)
x-values	$x_j(i)$	i-th x-value of record j
y-values	$y_j(i)$	i-th y-value of record j
y-errors	$\delta y_j(i)$	i-th y-error-value of record j
parameters	p_j^l	l-th parameter associated with record j
variables	v^n	n-user defined/system variable/function (global)
numor	#	separate associated id number to each record
xname	xname_j	x-label (max 80 char) of record j
yname	yname_j	y-label (max 80 char) of record j

Internal data representation (example)

record j	name	xname	yname	#	$\{p_j^l\}$	$\{x_j(i), y_j(i), \delta y_j(i)\}$
1	dat1	q	s(q)	123	$\{temp_1, conc_1, \cdots\}$	$\{i = 1, n_1 \vert q_i, I(q_i), \delta I(q_i)\}$
2	dat2	q	s(q)	124	$\{temp_2, conc_2, \cdots\}$	$\{i = 1, n_2 \vert q_i, I(q_i), \delta I(q_i)\}$
3	nse1	t	s(q,t)	700	$\{q_3, temp_3, \cdots\}$	$\{i = 1, n_3 \vert t_i, S(q, t_i), \delta S(q, t_i)\}$
4	...	...	...	...	...	...
...	...	...	...	...	...	...
nbuf	xy	x	y	999	$\{p1_j, p2_j, \cdots\}$	$\{i = 1, n_j \vert x_i, y_i, \delta y_i\}$

Table 1.1: Internal representation of data records.

record j	actual sequence number (address) of one $(y, y, \delta y)$ table including metadata
name	short (max. 8 characters) name-id of that table
xname	label of x-axis (max. 80 characters) ,it may contain latex elements if it starts with "$$"
yname	label of y-axis (max. 80 characters) ,it may contain latex elements if it starts with "$$"
#	numeric id code (numor), mostly for information only

p_j^l metadata, l-th parameter (**name(8)**, *value*) of this, j, record
$\{x_j(i), \cdots\}$ n_j table values $(y, y, \delta y)$

1.5.1 Variables and functions user defined variables, datreat variables and functions

With the **set** command one can create and modify (floating point) variables that can be referred in expressions, commands and input.
Note: Defining a *user variable* with the same name as a *parameter* of a data record makes the parameter invisible, all references in expressions yield the value of the defined *user variable* until it is removed by **clr varname**.

1.5.2 Global functions

Mathematical functions

sin(..)	$\sin(x)$ std sine
cos(..)	$\cos(x)$ std cosine
tan(..)	$\tan(x)$ std tan
atan(..)	$\arctan(x)$ std arctan
asin(..)	$\arcsin(x)$ std arcsin
acos(..)	$\arccos(x)$ std arccos
ln(..)	$\ln(x)$ std natural log
exp(..)	$\exp(x)$ std exp
sqrt(..)	$\sqrt{x}$ std square root
abs(..)	$\lvert x \rvert$ std absolute value
int(..)	$floor(x)$ std integer
erf(..)	$\mathrm{erf}(x)$ error function
J0(..)	Bessel function $J_0(x)$
J1(..)	Bessel function $J_1(x)$

Datreat functions and variables

X	the actual x-value implicitly runs over (j,i) in actions on records (note: capital letter!)
Y	the actual y-value "
ERR	the actual y-value "
x	the actual x-value (same as X but will be kept in axis name as x)
y	the actual y-value (same as Y but will be kept in axis name as y)
err	the actual y-value (same as ERR) "
nbuf	total number of records in memory
nsel	total number of **selected** records
sel	address j of the **first** selected record
maxx	maximum x-value of the first selected record
minx	minimum x-value of the first selected record
maxy	maximum y-value of the first selected record

miny	minimum y-value of the first selected record TBD:all sel recs
MAXX	maximum x-value of all selected records
MINX	minimum x-value of all selected records
MAXY	maximum y-value of all selected records
MINY	minimum y-value of all selected records TBD:all sel recs
centerx	"center of mass" of first selected record using data from present fit range only
widthxx	width obtained for the distribution around centerx
xv(j,i)	i-th x-value of record j (requires no selection)
yv(j,i)	i-th y-value of record j (requires no selection)
ye(j,i)	i-th y-error of record j (requires no selection)
nv(j)	number of table points in record j
indxval(j,x)	point index i (float) of x-value = x in record j
intxval(j,x)	point index i (integer) of x-value = x in record j
sumx(j,l,m)	sum of x-values l⋯m of record j : $\sum_{i=l}^{m} y_j(i)$
sumy(j,l,m)	sum of y-values l⋯m of record j : $\sum_{i=l}^{m} y_j(i)$
sumyerr(j,l,m)	error of sum of y-values l⋯m of record j : $\sum_{i=l}^{m} y_j(i)$
isel(n)	address j of the n-th selected record
num(j)	id-number (numor) of record at address j
sc(num)	record address j with id-number=numor (first occurance)
th_par(ip,it)	value of the ip-th parameter of the it-th activated theory
th_err(ip,it)	error for th_par
om(j,i)	frequency of the TOF data in point i of record j (requires TOF parameters)
en(j,i)	frequency of the TOF data in point i of record j (requires TOF parameters)

1.5.3 Expressions

The notion **expression** is used for parts of input/command lines that are to be converted to numerical values upon reading. The simplest **expression** is just a numerical value. Genuine expressions are arithmetic expression that can contain functions, user defined variables and datreat specific global functions and variables. Genuine expressions must be given in **brackets (...)** and may not contain

any blanks! A fast check to what value an expression will evaluate one can us the command **??**

expression

An expression can occur at any place, where a numerical input is expected. **Examples:**

```
#datreat =>  set myvar1 1.23 ↵
#datreat =>  set myvar2 0.77 ↵
#datreat =>  ?? (exp(-myvar1+3*myvar2)) ↵
> ?? (exp(-myvar1+3*myvar2))
```

```
2.9446795510655241
```

1.6 Customisation (Makros) How to combine commands

Sometimes one notices that complex commands or command sequences have to be repeated frequently. To save typing and increase the efficiency of *datreat* usage **makros** can be used. This may start with just associating another name to a command (for better memorisation) to more complex constructions with conditions and loops etc.

Makros are simple text-files that contain a sequence of datreat commands or other makro calls. The only special syntax rules pertain the **first line** of the makro, which looks like this:

```
makro ↵
makro _arg1_ ↵
makro _arg1_ _arg2_ ··· ↵
```

The keyword **makro** is necessary to confirm to *datreat* that this file is a makro. Optionally one or several arguments: **_arg1_ _arg2_** ··· can be present.
A simple example is a makro **pstyle** that plots selected records of NSE data in publication ready style:

```
makro
rename yaxis $$S(Q,t)
rename xaxis $$tns
plot ymax 1.05 txoff flinewd 1.2 axtxsize 1.3 xlegdist 0.1
```

Or slightly more complex a makro with one argument (here a filename) that reads the record table content of saved data **in filename** and then tries to extract theory **get_th filename**. The filename of the makro, i.e. the makro name is **inplus**, its contents:

```
makro _f_ ! input file (as created by msave) and activate the theory definition contained in _f_
get_th _f_
```

the filename variable *_f_* is inserted into the command lines present in the makro by **string replacement**. Lines starting with **!** are comments and ignored by *datreat*.
To use the makro in datreat:

```
#datreat => inplus myMsavedData ↵
```

For the system to be able to find a makro it should either be **stored in the current working directory**, such that it is strictly associated to the evaluation done with the data there. Or it may be put to the standard makro directory (*datreat/makros/...*, then it should be visible for any *datreat* instance. Makros in the current working directory will mask those with the same name in the standard makro directory.
Note: Makros with names that match genuine *datreat* command names will not be recognized.

1.6.1 Makros specific commands

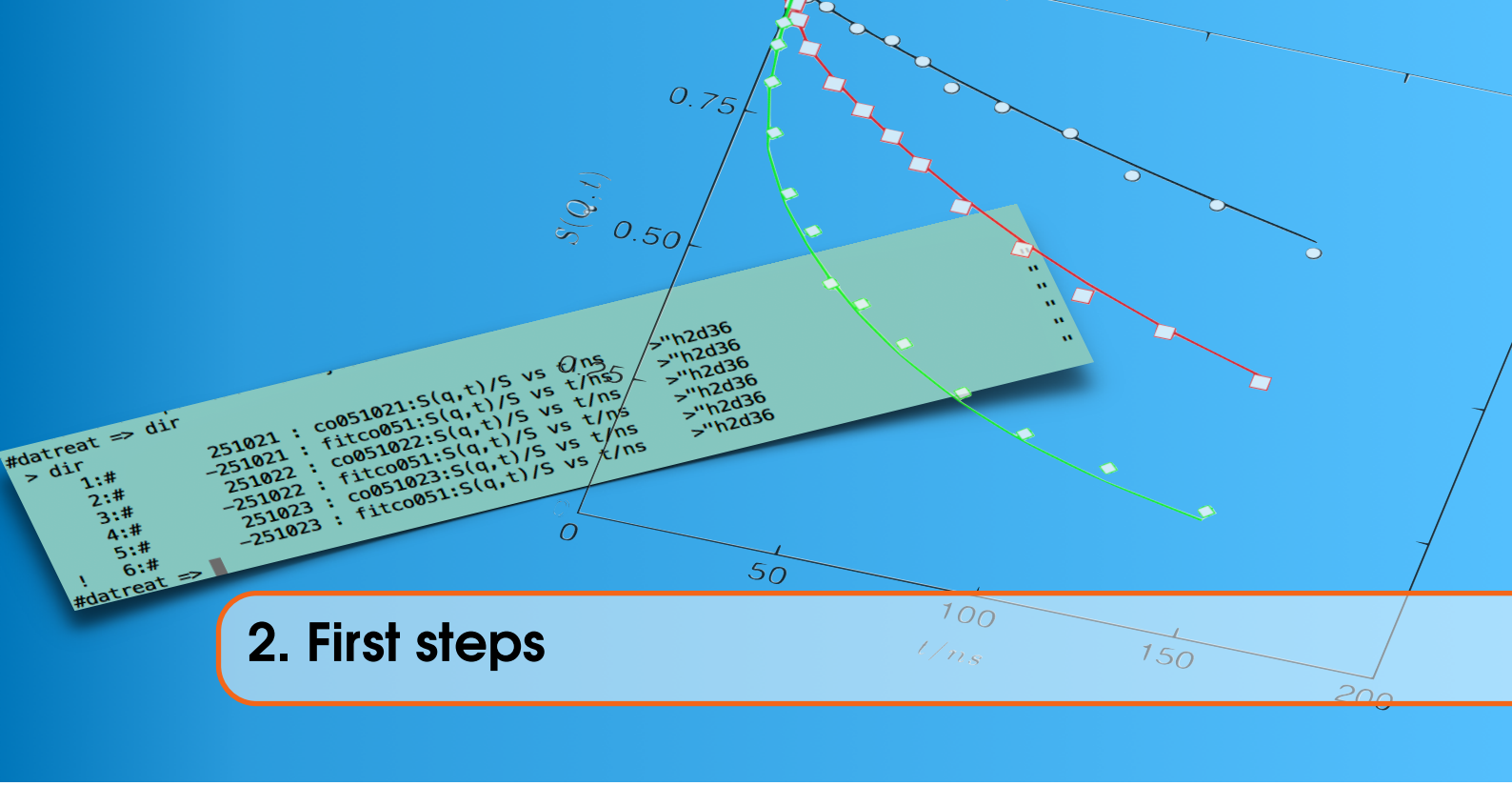
!	line content from here is comment only
:lab	label, the rest of the line is ignored
say message	display (speak) a message

if (expr1) <= (expr2) then goto :lab	conditional jump to label :lab
goto :lab	unconditional jump to label :lab

1.7 Necessary and useful UNIX commands (Mac)

cd	*directory*	change directory
ls -ltr		list files in directory
pwd		show the current directory path
cp	*file1 file2*	copy file1 to file2
mv	*file1 file2*	move/rename file1 to file2
rm	*file1*	remove file1 (delete)
cat	*file1*	lists file content
>	*cmd > file1*	redirection: writes output of cmd to file1
>>	*cmd >> file1*	redirection: APPENDS output of cmd to file1
<	*cmd < file1*	redirection: inputs contents of file1 to cmd
|	*cmd1 | cmd2*	pipe: output of cmd1 is used as input for cmd2

Note: File operations on this level cannot be undone!



2. First steps

2.1 Starting and quitting the program

After installation according to the procedure described in **??** the program should be ready to use. This is done by opening a **terminal** and change the directory to where the work shall be performed. Then start **datreat** by typing *datreat* ↵.

```
...> mkdir Myworkingdirectory ↵
...> cd Myworkingdirectory ↵
...> datreat ↵#datreat =>
```

After use *datreat* by be exited by the commands:

```
#datreat =>  quit     Leaves datreat and saves last state
or
#datreat =>  exit     Leaves datreat WITHOUT saving
```

Note 1:

```
...> datreat 0     Starts datreat without restoring the last state
```

However, the last plot setting is kept. Remove file *last_plotsetting* to return back to default prior (re)start of datreat.

Note 2: The last state is stored in the following files:

- last_datreat_content data records
- lastselections actual selections, requires unique nonzero numors !
- lastusv uservariable with contents
- lastth theory settings
- last_plotsetting plot settings, always kept

These settings are stored in the current working directory and therfore are "private" to that topic. Treating another problem in another working directory will have its own last-state.

Note: It is recommended to do the work on one topic in a dedicated *subdirectory*, which the will contain the comment history, the last state of the program and any plots or other output. The used data may be copied to that *subdirectory* or addressed by using the full path specification (see also the **path** command).

II Input, records and display

3 Data management I/O 21

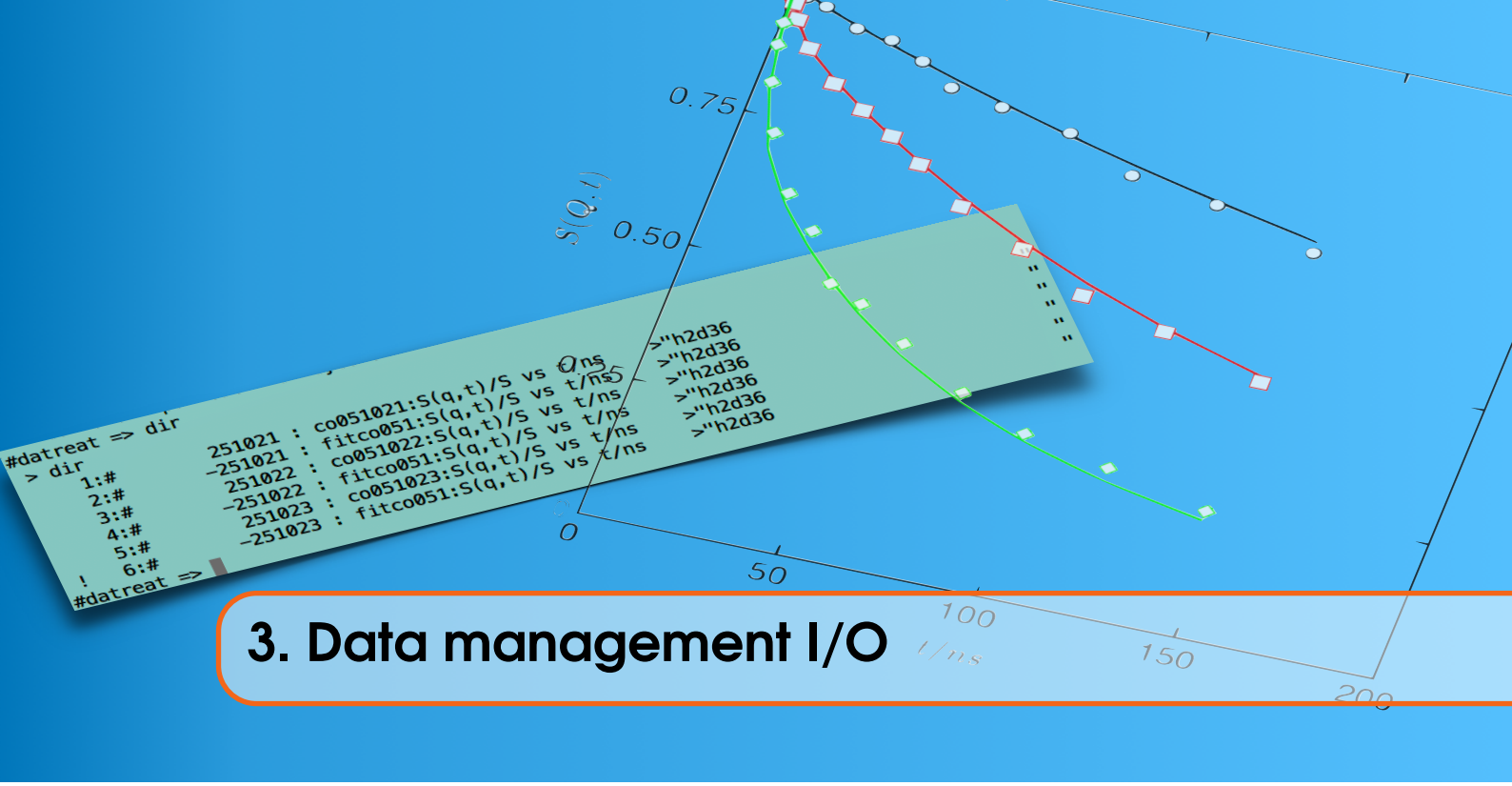
3.1 input *filename* input files and data format
3.2 dir (pname) listing data holding records
3.3 purge (all | rest | sel)
3.4 msave *filename* save data and fit results
3.5 inplus *filename* read msave-stored data with theory model
3.6 putpar adding/setting data-record associated parameters
3.7 copy copy records in memory
3.8 allsave save current datreat state to subdirectory
3.9 allresto restore cdatreat state from a subdirectory

4 Select 25

4.1 sel (spec) select data records for next actions
4.2 dsl display list of selected records
4.3 activate theory model activation
4.4 acl Loading previous theory model

5 Data inspection and plotting 29

5.1 plot basic data display
5.2 plot (styling) Sizes, colors and legends
5.3 rename xaxis *new-label-x* | yaxis *new-label-y* rename axes (plot +)
5.4 title *Any title* set a title (plot)
5.5 plot (save and print)
5.6 edit (*j*) Editing data in a memory stored record



3. Data management I/O

3.1 input *filename* input files and data format

Synonym: **in** Reading data from ASCII files that may contain a header of metadata followed by a

table containing the proper (x, y) data with an optional 3rd column with y-errors. The data are read into datreat memory, each (x, y)-table group is represented by one **record**, the record addresses *j* are filled in the sequence of reading. File names can be arbitrary, except that a few extensions indicate special deviating data formats (e.g. inx or ins).
The default *subdirectory* (i.e. *path*) where input files are expected is the *current working directory*. This may be changed either by explicitly specifying a path in front of the filename in the **input command** or by using:

```
#datreat =>  path          lists paths setting in this session ↵
#datreat =>  datapath PATH-TO-DATA     sets data-path for this session ↵
```

```
#datreat =>  in filename ↵
```

Note: After **in** ALL records with *positive numor* (i.e. omitting fit results) that have been read are *SELECTED*!

3.2 dir (pname) listing data holding records

Displays the present contents of the *datreat* memory in terms of a list of loaded records.

```
#datreat =>  dir parname1, parname2, ... ↵
```

Example:

```
#datreat =>   dir q temp ↵

 > dir q temp
   1:#  974101 : 6SB520 :fqt  vs t/ns  >"6SB520C 6SB5 2mm "  q   0.041468 temp  295.29
   2:#  974103 : 6SB520 :fqt  vs t/ns  >"6SB520C 6SB5 2mm "  q   0.058920 temp  295.29
   3:#  974104 : 6SB520 :fqt  vs t/ns  >"6SB520C 6SB5 2mm "  q   0.082602 temp  295.29
!  4:#  974105 : 6SB520 :fqt  vs t/ns  >"6SB520C 6SB5 2mm "  q   0.098593 temp  295.29
!  5:#  974106 : 6SB520 :fqt  vs t/ns  >"6SB520C 6SB5 2mm "  q   0.122080 temp  295.29
!  6:#  974107 : 6SB520 :fqt  vs t/ns  >"6SB520C 6SB5 2mm "  q   0.137806 temp  295.29
!  7:#  974110 : 6SB520 :fqt  vs t/ns  >"6SB520C 6SB5 2mm "  q   0.157455 temp  295.29
!  8:#  974112 : 6SB520 :fqt  vs t/ns  >"6SB520C 6SB5 2mm "  q   0.179635 temp  295.29
```

The example output contains the following information:

column	example	description
1	**!**	an exclamation mark in column 1 indicates that the record is **selected**
2	**1:, 2: ..**	record address *j*
3	**# 97**···	numeric-id (numor) of this record (mainly information)
4	**6SB520**	name-id
5	**fqt**	y-axis name
6	**vs**	fix keyword (versus)
7	**>"..."**	first n-characters of the record **comment**
8	**q**	name of first specified parameter to be listed (volatile option)
9	**0.04**···	value of parameter *q* for this record
10	**temp**	name of 2nd specified parameter to be listed (volatile option)
11	**295**···	value of parameter *temp* for this record

3.3 purge (all | rest | sel)

Removes all **selected** or *all* data from the record address list. Or *rest* = all not selected records.

```
...>  purge rest ↵   clear record memory but keeps selected records
...>  purge all ↵   clear record memory completely
...>  purge sel ↵   removes all selected records
...>  purge fits ↵   removes all fits/calculated records (and those with negative numor
```

Clear the stored record list.

Note: Record addresses may (will) change if some are removed.

3.4 msave *filename* save data and fit results

Save all selected records (including fitted/computed curves) and the currently active theory model to file *filename*

```
...>  msave filename ↵
```

TBD(datreat src):consolidate plotting actions by this... title setting etc.

3.5 inplus *filename* read msave-stored data with theory model

Read records that have been save by **msave** including the stored active theory model from file *filename*.
Note: The record addresses will most probably different form those when stored! **Note:**Any actual theory model definition will be replaced by that store in file *filename*.

```
...> inplus filename ↵
```

TBD(datreat src):make it a genuine command

3.6 putpar adding/setting data-record associated parameters

Add a new parameter (with value) or change value of record associated parameters in **all selected records**. Ideally with *input* all relevant parameter associated with the respective data record should also be read-in by the *input* command. Often, however, this is not the case and additional information needed by some *theories* or as plain information in plot legends or for selecting a subset of loaded records are still missing. The **putpar** offers a means to add this information. ■

```
#datreat => sel selection list      first select the records to be affected ↵
#datreat => putpar parname value    assign value to parname ↵
```

Example:

```
#datreat => sel 1 2 3 ↵     select records 1 and 2 and 3
#datreat => putpar temp 373 ↵     set/create parameter temp to value 373
```

Note1: In the present version of datreat the name *parname* has a length limit of 8 characters.
Note2: The parameters are displayed in the plot legend.
Note3: The parameters can be inspected by **dir** *parname1 parname2* ⋯ or for selected records only by **dsl** *parname1 parname2* ⋯ .
Note4: The parameters may be used in selection criteria by **sel all** *parname value* **band** *value*

Attention! In order to preserve the added parameter information select all relevant records and store a copy using the **msave** *newname* command. And restore it using **input** *newname* or **inplus** *newname*. ■

Note5: The value of a parameter assoiated with the **first** selected record can be addressed in any expressions by (*parname*) that is evaluated to the parameter value.
Note6: Of course parameters can (permanently) be added to record data files by directly editing and adding lines to the parameter section in the beginning of the data files.

3.7 copy copy records in memory

Copy *selected* records to the end of the record list (selection moves to copy).

```
...> copy ↵     copy selected records
```

3.8 allsave save current datreat state to subdirectory

Copy all information needed to restore currently loaded data, theories, selection, plotsettings etc . to a subdirectory.This is a *makro*.

```
...> allsave DIRNAME ↵    copy present state to subdir DIRNAME
```

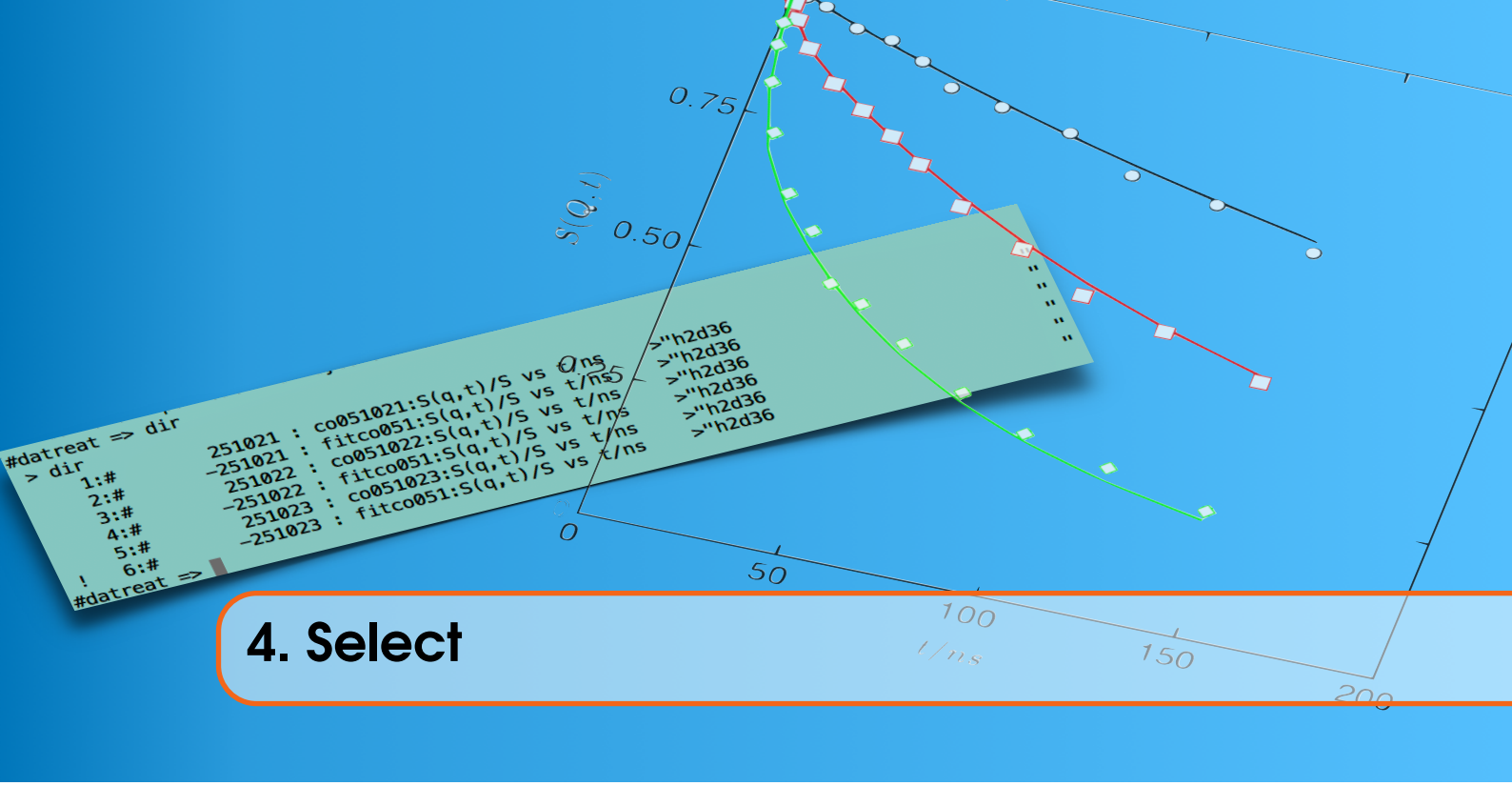
Note: Use (makro) *allresto* to restore!

3.9 allresto restore cdatreat state from a subdirectory

Restore datreat state (loaded data, theories, selection, plotsettings etc .) from a subdirectory.This is a *makro*.

```
...> allresto DIRNAME ↵    restore state from subdir DIRNAME
```

Note: Use (makro) *allsave* to save state!



4. Select

Selection of records is the way to control on which data the following command actions are to be applied. The main command **sel** besides the plain selection according to record address j a number of powerful options. **Note:** The sequence of selection matters!

4.1 sel (spec) select data records for next actions

There are a number of methods to select records. Since selection is a genuine to control the evaluation in *datreat* in the following the methods ([spec]) are explained one by one.

#datreat => **sel** j_1 j_2 $j_3 \cdots$ ↵

Selects record addresses j_1, j_2, etc..

#datreat => **sel** j_1 $-j_2$ ↵

Selects a continuous range of record addresses $j_1 \cdots j_2$, note $-j_2$ means the negative of the address, i.e. no blank between "-" and the address value is allowed.

#datreat => **sel** j_1 $-j_2$ $-j_s$ ↵ e.g. select all odd or even addresses in range ...

Selects a continuous range of record addresses j_1, j_1+j_s, $j_1+2j_s \cdots j_2$, note $-j_2$ means the negative of the address, i.e. no blank between "-" and the address value is allowed. A further negative number $-j_s$ allows to specify a step between the address selection in the continuous range between j_1 and j_2. This allows, e.g. the simple selection of all odd or all even addresses etc..

#datreat => **sel add** j_1 j_2 $j_3 \cdots$ ↵

Appends record addresses j_1, j_2, etc.. to the current selection list.

#datreat => **sel** j_1 j_2 $j_3 \cdots$ **fit+** ↵ selects records including fitted curves

Selects j_1 j_2 $j_3 \cdots$ and associate corresponding fitted curves to the selection. **Note:** this is automatically done by the fitting process but is lost by following **sel** commands.

#datreat => **sel next** *parname* p band Δp ↵

Selects the next (scanning record addresses starting form the last selected record) that has a parameter *parname* with value equal to $p \pm \Delta p$.

#datreat => **sel all** *parname* p band Δp ↵

Selects all records that have a parameter *parname* with value equal to $p \pm \Delta p$.

#datreat => **sel narrow** *parname* p band Δp ↵

Selects all records form the currently selected list that have a parameter *parname* with value equal to $p \pm \Delta p$.

#datreat => **sel exclude** *parname* p band Δp ↵

Remove all records form the currently select list that have a parameter *parname* with value equal to $p \pm \Delta p$.

#datreat => **sel exclude numor mod** m ↵

Remove all records form the currently select list that have a numeric-id parameter numor N that matches $\mod(N, m) = 0$.

#datreat => **sel** ↵

removes all selections.

4.2 dsl display list of selected records

Display a list of the selected records (similar to **dir** but restricted to selected records. endcorollary

#datreat => **dsl** *parname1, parname2, ...* ↵

Example:

#datreat => dsl q temp ↵

```
 > dsl q temp
3 [ 0]:# 974104 : _6SB5_20:t/ns vs fqt  >"_6SB5_20C _6SB5 2mm " q 0.082602 temp 295.29
4 [ 0]:# 974105 : _6SB5_20:t/ns vs fqt  >"_6SB5_20C _6SB5 2mm " q 0.098593 temp 295.29
5 [ 0]:# 974106 : _6SB5_20:t/ns vs fqt  >"_6SB5_20C _6SB5 2mm " q 0.122080 temp 295.29
```

In analogy to the dir command, in [brackets] the addresses of the associated fitted curve records are shown (0=no fit associated).

4.3 activate theory model activation

Activate (synonym **ac** is used to set up a model assembled from the current collection of available theories. The *activated* model may the be fitted or compared by calculation with data records.

A list of available theories (names only) is obtained by the command **th** . More information on one of these theories can be displayed using

#datreat => **th theoryname** ↵

The (fit)-model can then be assembled by single or multiple activation commands:

#datreat => **ac** *theoryname1* ↵#datreat => **ac** *theoryname2* ↵#datreat => **ac** ··· ↵

The results of the thus *activated* theories are *added* to yield the final result.

#datreat => **ac** *theoryname3* **multiply** ↵

will add a further definition the result of which will be multiplied to the result of the precedent activated theories. **Note:** starting with a *multiplied* theory is nonsense.

#datreat => **ac** *theoryname* **range** *parname* **min** *value1* **max** *value2* ↵

Causes that instance of the theory only to be evaluated if the value of the parameter /varparname of the record under actual comparison lies within the $min \cdots max$ range.

To **set the parameters** of the activate theory model use **cth** , which opens an editor that allows to set numbers or otherwise modify the model, choose which parameters shall be fitted and possibly establish parameter couplings (see. sec....).

Note: The actual theory model definition as e.g. modified using **cth** is always stored on file *lastth* in the current working directory. It also will be appended to the file written by **msave filename**.

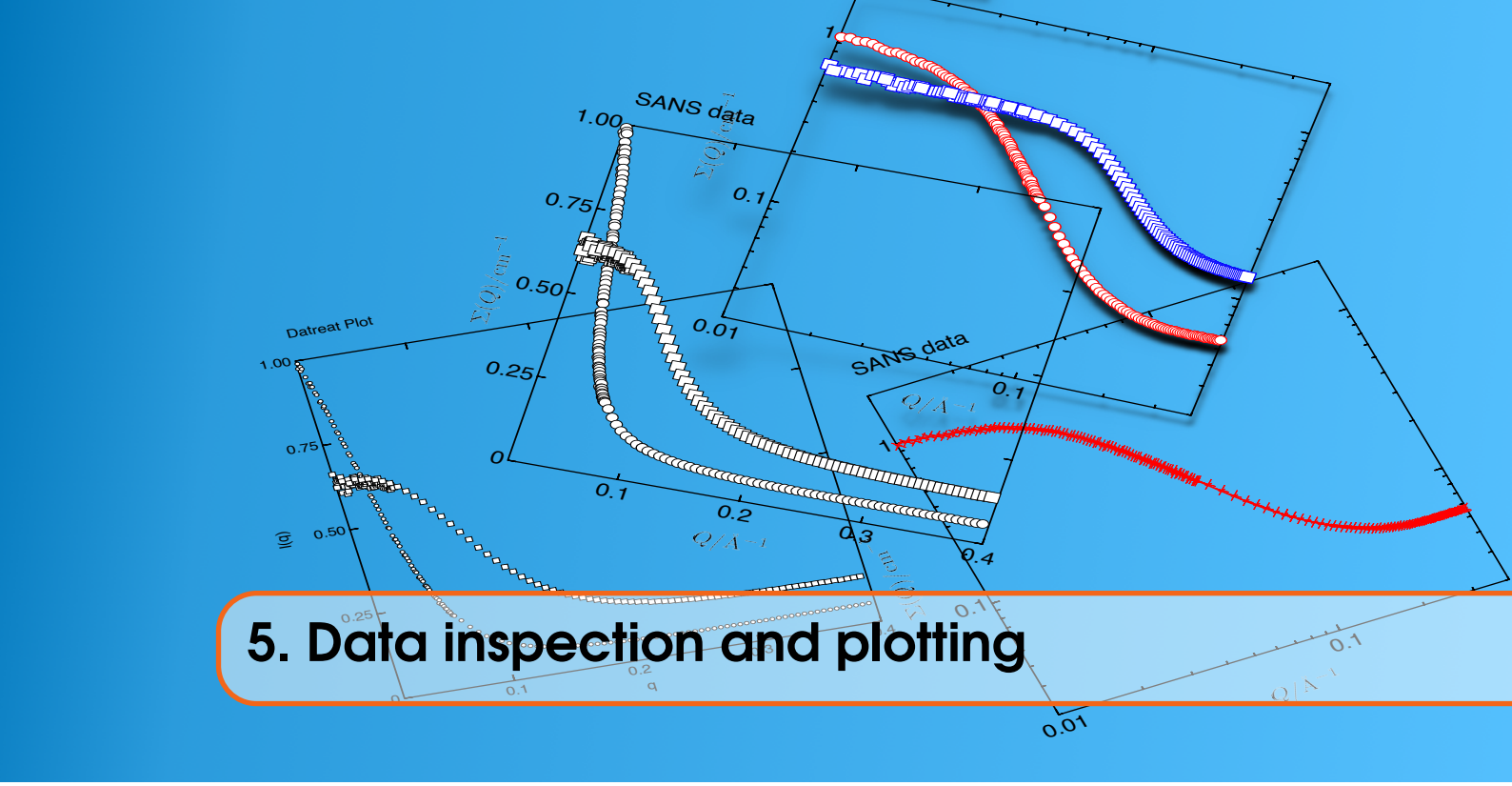
#datreat => **al** ↵

displays the actually loaded theory model (normally this is implicitly executed upon loading/modifying).

4.4 acl Loading previous theory model

```
#datreat => acl ↵
```

loads the last theory model, which is on file *lastth*. **Note:** *lastth* is updated each time a theory model is update or modified, i.e. by fitting.



5. Data inspection and plotting

5.1 plot basic data display

Plotting of the *selected* records and (if present) associated fitted curves.

```
#datreat =>  plot ↵
```

Plots the selected records using the current axis and style settings.

```
#datreat =>  plot xmin xmax ymin ymax ↵
```

Plots the selected records after setting the x- and/or y- limits. **Note:**

```
#datreat =>  plot xmin (minx) xmax (maxx) ymin (miny) ymax (maxy) ↵
```

Realizes and auto scaled plot.

```
#datreat =>  plot log_x log_y ↵
```

Plots the selected records using log-scaled axes (either x or y or both).

```
#datreat =>  plot lin_x lin_y ↵
```

Plots the selected records using (back to normal) lin-scaled axes (either x or y or both).

```
#datreat =>  plot scaled f1 f2 ··· ↵
```

Plots after scaling the N selected records (in the sequence of selection) by factors $f_1 \cdots f_N$. The scaling factors are reported in the legend part of the plot. **Note: scaled** is a volatile option. It has to be specified each time when is shall be applied. The selected records are not changed.

#datreat => **plot clip** ↵

Plots with exact clipping of curves. **Note: clip** is a volatile option. It has to be specified each time when is shall be applied. The selected records are not changed.

5.2 plot (styling) Sizes, colors and legends

To yield publication style plots or make things better visible there are a number of style modification options.

#datreat => **plot sysize** *sz* **symb** *S1 s2* ⋯ ↵

Plots the selected records using symbols with numbers *S1*, *S2* ⋯ assigned in the sequence of selection. Symbol type number 0 yields a line. *sz* is (global) symbol size scaling (default is 0.7). Symbol types are: **0:line, 1:dots, 2:open circles, 3:open squares, 4:open diamonds, 5:triangle_up, 6: triangle_down, 7:diabolo, 8:solid_triangle_up, 9:solid_triangle_down, 10:solid_square, 11:plus, 12:star, 13:cross, 14:solid_circle, 15:solid_diabolo, 16:hourglass, 17:solid hourglass, 18:solid diamond, 19:star2, 20:solid star2, ...**.

#datreat => **plot lwid** *lw1 lw2* **...** **ltype** *lt1 lt2* **...** ↵

Plots the selected records using line widths *lw1*, ... and line types *lt1* ... . Line types are: **1:solid, 2:dashed, 3:dotted, 4, 5, ...:dash-dotted**.

#datreat => **plot colo** *C1 C2* ⋯ ↵

Plots the selected records using colors with numbers *C1*, *C2* ⋯ assigned in the sequence of selection. Colors are: **0:black, 1:red, 2:green, 3:blue, 4:cyan, 5:yellow, 6:magenta, 7:black, ...**.

#datreat => **plot errors | noerrors** ↵

Plots the selected records including error bars (**errors**) or without error bars (**noerrors**).

#datreat => **plot parplo | noparplo** [*pn1 pn2* **...**] ↵

Plot with all record parameters aside (legend) (**parplo**) or with no or a limited selection { *pn1*, *pn2* ... } of parameters. **Note:** if the number of parameters or selected records is too large for the legend to contain all parameters use **noparplo** or **txsize** *tscale* to reduce the letter size.

```
#datreat => plot txsize tscale axtxsize ascale ↵
```

Plot with a scaled side-legend text size (**txsize**) and/or a scaled axis legend size (**axtxsize**).

```
#datreat => plot xlegdist xd ylegdist ascale ↵
```

Plot with shifted x- and or y- legend positions.

```
#datreat => plot tit_x tx tit_y ty ↵
```

Plot with a shifted location of the title, *tx* is the title distance from the left axis limit and *ty* from the upper limit of the y-axis.

```
#datreat => plot texfak f ↵
```

Plot with a size scale factor *f* for tex-type axis names (applies to texts that start with $ or $$ to trigger tex decoding).

5.3 rename xaxis *new-label-x* | yaxis *new-label-y* rename axes (plot +)

Sets new labels for x- and y-axis (maximum length is 80 characters) to all selected records. Labels starting with **$$** may contain LaTex type constructions.

```
#datreat => rename xaxis t/ns ↵
#datreat => rename yaxis S(Q,t) ↵
```

or for LaTexed labels

```
#datreat => rename xaxis $$\tau/ns ↵
#datreat => rename yaxis $$S(Q,t) ↵
```

5.4 title *Any title* set a title (plot)

Sets the title for the NEXT plot.

```
#datreat => title the title ↵
```

Note: The titel setting is "consumed" by the next plot, after that the title will be empty in order to avoid accidental overwriting of the plot by the next plotting action!

5.5 plot (save and print)

Plot always creates pdf-file *last_datreat_plot*, which, however, is overwritten each time. In addition copies will be generated with names containing the title and (if activated by specifying **plot # 1** once per session) a numbered copy with auto incremented number (example here 123) *dtrplot123.pdf* or if a title is present *dtrplot_thetitle123.pdf*. The number is also shown in the small text in the lower left corner of each plot. The last created plot is automatically opened in a pdf-viewer, form there it

may also be printed or copied. The viewer may be left open on screen or can be closed without affecting the further datreat operation.

5.6 edit (*j*) Editing data in a memory stored record

Edit writes the contents of a record (selected or specified by *j*) on a buffer file *datbuf* and loads this to a text editor. One may change entries or just inspect the contents. After leaving the editor the (modified) data are read back to a **new** record address, which then is selected.

```
#datreat =>  edit j ↵
```

TBD(datreat src):Allow for many selected records, collect them in datbuf ...

Note: the default editor may be changed (for the present datreat session) by the command **editor editor system call command**. I.e. to choose *text editor* (Mac default)

```
#datreat =>  editor open -Wa textedit ↵     set editor call to textedit
#datreat =>  editor open -Wa emacs ↵     set editor to emacs (if installed)
#datreat =>  editor open -Wa Aquamacs ↵     set editor to Aquamacs (if installed)
#datreat =>  editor open -Wa MacVim ↵     set editor to MacVim (if installed)
```

the (mac) option **-W** lets the execution wait until the editor is left again, which is required for proper function of the datreat editor use. Proper function of the selected editor may be checked by the datreat command **ed filename**, which uses the chosen editor to open the specified file.

III Data reduction

6 Combination, averaging and clipping 35

6.1 average xcatch δx (absolute) (log_y)

6.2 rebin nk

6.3 rerange r_1 r_2 (y)

6.4 alingny (offset o yscale ys ostep do) ystep dy automatically adjust scales

7 Transformation: applying functions to x and y 39

7.1 fxy (op) x=*(expressionX)* y=*(expression)*

8 Arithmetic between data sets (records) 41

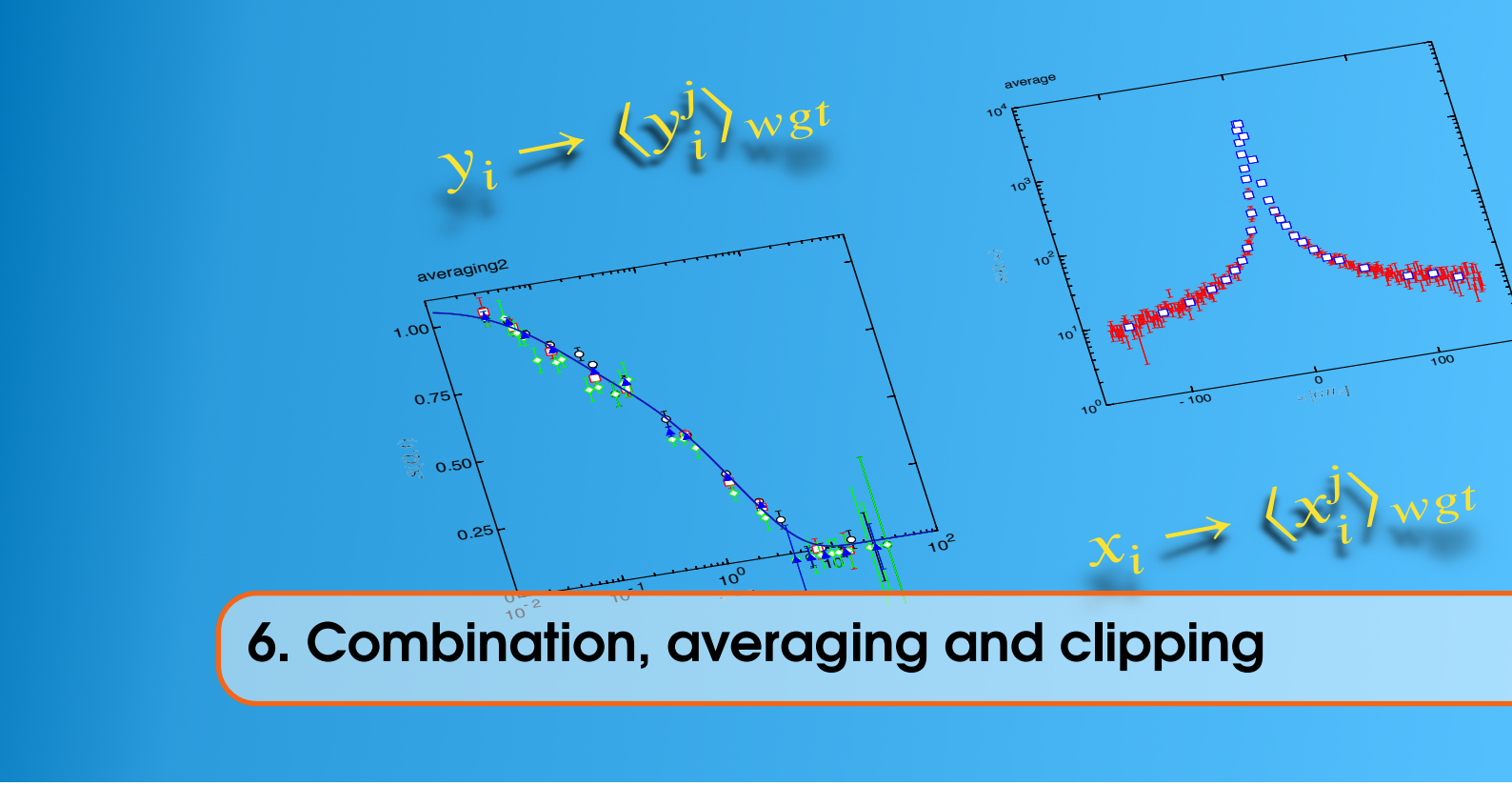
8.1 arit f1 *(factor1)* f2 *(factor2)* (**mult|div|vid**) (**to** *numor*)

8.2 addsels

8.3 recsum from *rec1* to *rec2*

9 Special function scaling (spatcorr) . . . 45

9.1 spqscal



6. Combination, averaging and clipping

6.1 average xcatch δx (absolute) (log_y)

Merge close points from one or several selected records to create a sorted consolidated result records with smaller error bars and fewer points. This may be used to get cleaner plots or **reduce the number of points to be computed if fitting models with long computing times** in order to speed up the fitting process (intermediate steps).
Usage variations:

```
#datreat =>  average ↵      use default/last xcatch
#datreat =>  average xcatch δx ↵
#datreat =>  average xcatch δx absolute ↵
#datreat =>  average xcatch δx log_y ↵
```

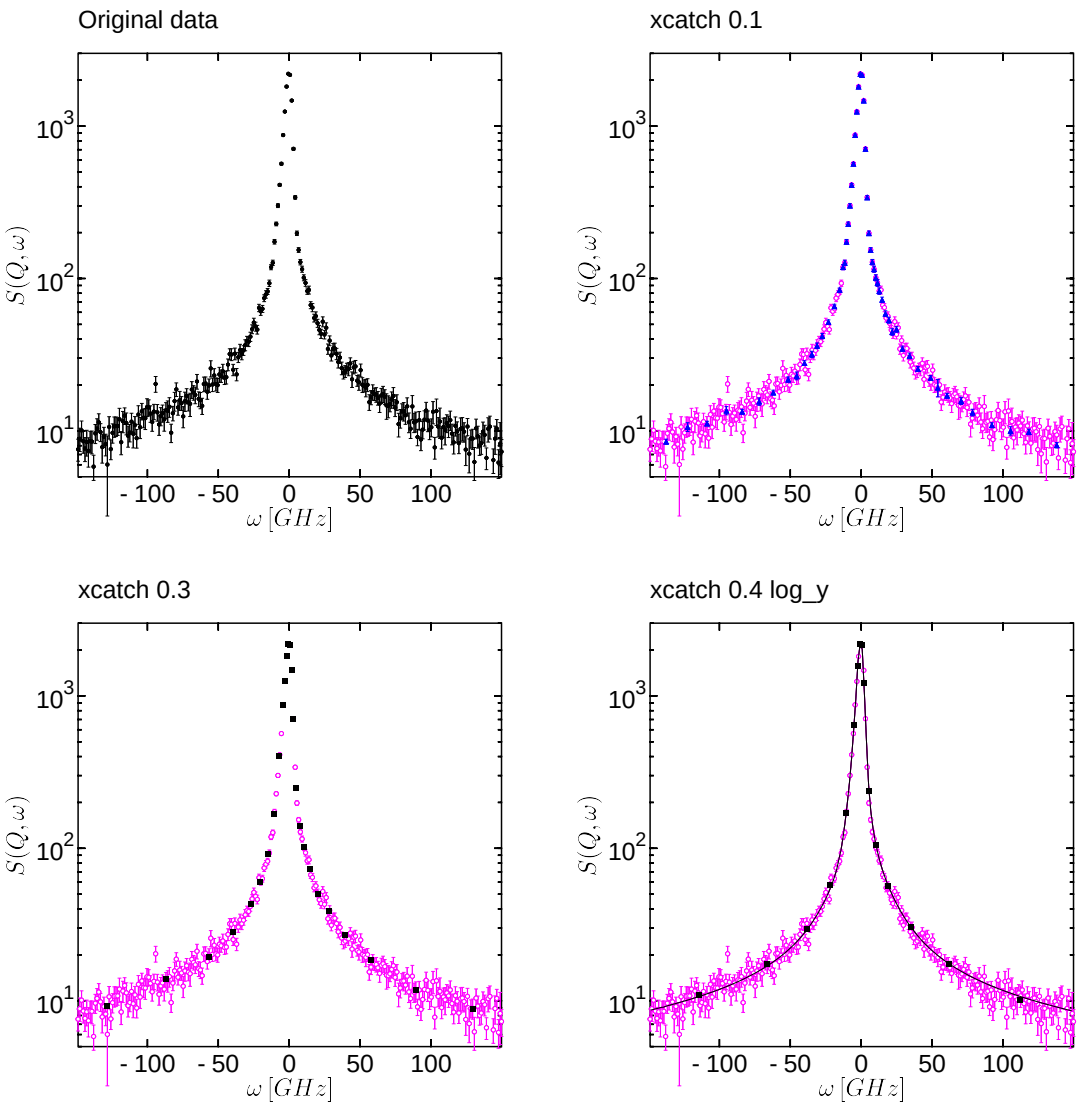
1. Collect all data points from one or several records into one linear array (x,y,error).
2. Sort that array to increasing error.
3. Start with the (still left *not used*) lowest error entry and look for all other points within a x-distance given by *xcatch* (relative to x or absolute) (index j).
4. collect the information to create a new average point (index i) using error-weighting: $w_i = [\sum_j 1/y_{err,j}^2]$
   $x_i = [(1/w_i)\sum_j x_i/y_{err,j}^2]$
   $y_i = (1/w_i)[\sum_j y_j/y_{err,j}^2]$
   $y_{err,i} = \sqrt{1/w_i}$.
5. flag the selected initial and the selected *neighbouring* points as *used*.
6. Iterate by returning to point 3 until all points are used.
7. Finally sort result array according to x.
8. If several records contribute determine the best guess of the metadata parameters of the resulting record.
9. Select the resulting record.

The collection range is controlled by the value δx of *xcatch* and the volatile options *absolute* or *relative* (=default).

For NSE type or other relaxation function type data normally *relative* is the appropriate choice, δx may typically be between 0.05 and 0.5.

Example:

```
#datreat =>  sel (original) ↵   #datreat =>  average xcatch 0.1  ↵
#datreat =>  sel (original) ↵   #datreat =>  average xcatch 0.3  ↵
#datreat =>  sel (original) ↵   #datreat =>  average xcatch 0.4  ↵
```



Illustrates the merging (and reduction in number) of points from a backscattering spectrum, always starting from the original data and the applying **average** with **xcatch** = 0.1, 0.3, 0.4, which yields the solid symbols in the plots. A fitted curve to the average results is shown for xcatch = 0.4.

Note 1: The x-values of the result are different form the originals and generally not spaced evenly (even if the originals were) but come out sorted in ascending order (even if the originals were not).

Note 2: Parameters of the result are weighted averages of the parameters of the contributing records (if more than one were selected). **Attention** in that case **average** will only yield sensible results if the selected records correspond to close by physical states and the range of x-values covers approximately the same region.

6.2 rebin *nk*

Rebins the x-axis by adding the contents of *nk* adjacent x-values (channels) and replacing it by the **average** values of *x* and *y*, errors are treated accordingly. The contraction is put to a new record. To perform it *on place* use the otherwise equal command **kz** (with loss of the original content).

```
#datreat => rebin nk ↵    averages over nk x-channels.
```

Note: If data are non evenly spaced or have very different error bars on points with close by x-values: use **average** instead!

6.3 rerange r_1 r_2 (y)

Removal of points outside the range $r_1 \cdots r_2$ from the selected records. If no option **y** is given the range pertains the x-axis, with **y** it pertains the y-axis. The reduced data are stored with new record addresses.

```
#datreat => rerange r1 r2 ↵
```

Examples:

```
#datreat => sel (j0) ↵    select original record
#datreat => rerange x1 x2 ↵    copy only x-range x1 x2 to new record
#datreat => set (nbuf) j1 ↵    save the created address to j1
#datreat => sel (j0) ↵    select original record again
#datreat => rerange x12 x22 ↵    copy another x-range x12 x22 to new record
#datreat => sel add (j1) ↵    select also the previous range restricted record
#datreat => average xcatch 0 ↵    recombine both ranges
```

The above sequence is an example to prepare a more complex selection of x-values, e.g. for fitting. The finally selected record contains the selected regions. Note: for fitting it would also suffice to select all the partial ranges.

```
#datreat => rerange 0 (maxy+1) y ↵    keep only positive y-values
```

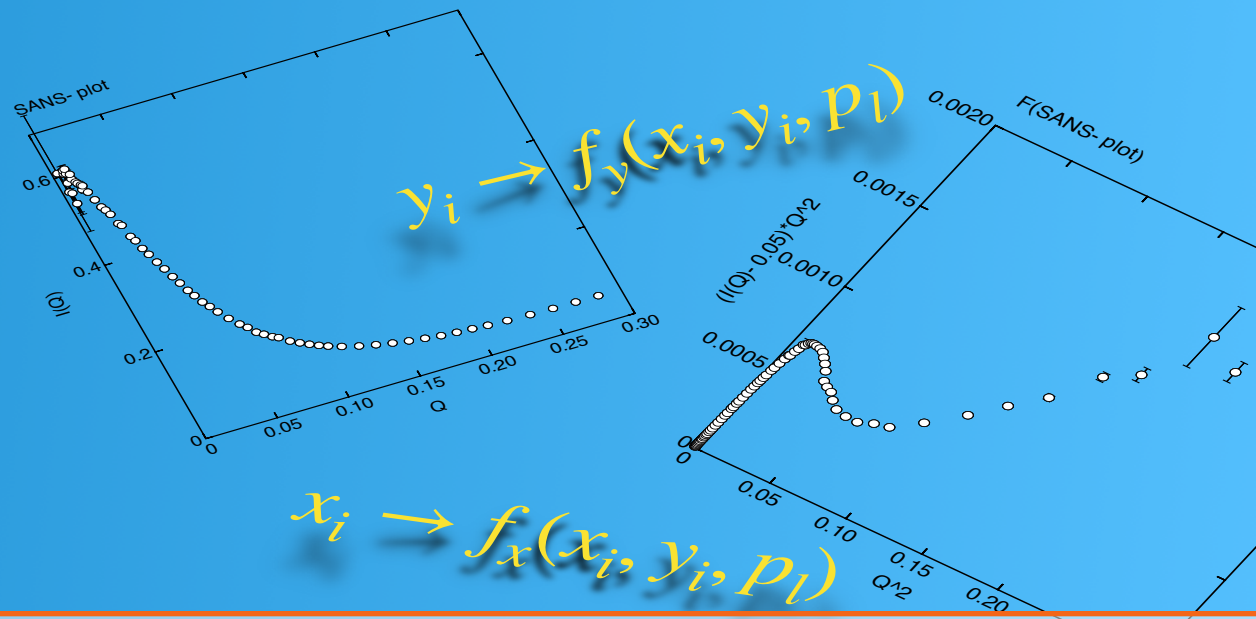
6.4 alingny (offset *o* yscale *ys* ostep *do*) ystep *dy* automatically adjust scales

To determine offset and scale $(y+o)\times ys \rightarrow y$ of second selected record in order to match the first selected record use command **aligny**! The scaled result is copied to a new record this and the "master" record are selected such that one may use **average** to merge them. Applied scale and offset are put to the record parameters of the scaled second record.

```
#datreat => alingny [offset o yscale ys ostep do ystep dy]
```

The optional specification of *offset*, *yscale*, *ostep* and *ostep* enable explicit specification of start values and step width for the Nelder-Mead simplex minimisation procedure used here.

Hint: Excluding marginal points in the overlap region (**rerange**) may improve the accuracy of the result.



7. Transformation: applying functions to x and y

7.1 fxy (op) x=*(expressionX)* y=*(expression)*

Allows to transform the x-values and the y-values by functions. Errors in y are transformed accordingly. The transformation is done for all selected records (is the sequence of selection), finally the resulting transformed data records are selected.

```
#datreat =>   fxy [op] x=(expressionX) y=(expression) ↵
```

Examples:

```
#datreat =>   fxy x=X y=Y*X*X ↵
#datreat =>   fxy x=q^2*sqrt(wl4*X) y=Y ↵
```

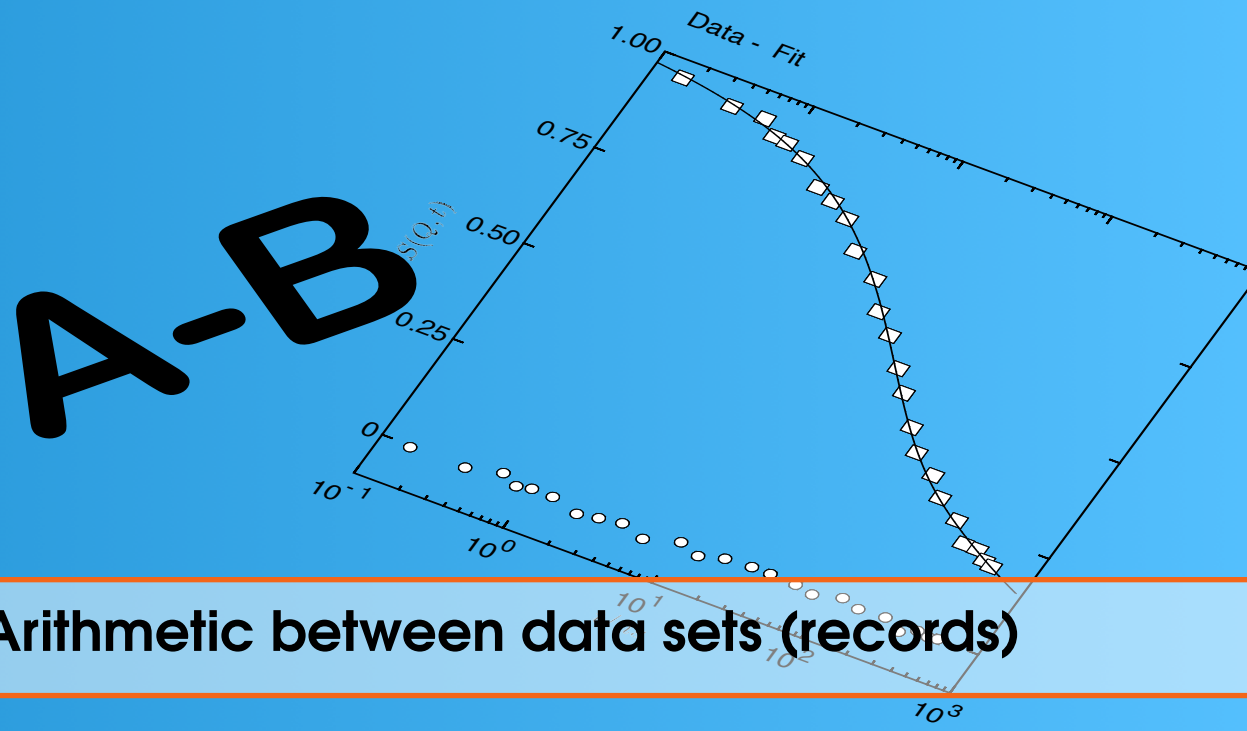
where in the last line the *parameter*, p^l, **wl4** is used.

cmd item	type	volatile	# arg	descriptions
op	option	yes	0	If present the results will replace the original data instead of creating new result records.
x=	keyword	no	(1)	Expects an expression for the transformation of x-values to follow.
(expressionX)	value string	no	–	is an expression that refers to the x-values as **X** and y-values as **Y** (and y-errors as **ERR**) also values of parameters, user- and system-variables can be used. The values are always extracted form the actually processed record. It can contain binary operators **+, -, *, /, ^** and functions: **sqrt, exp, sin, cos, tan, atan, erf, J0, J1**. The result replaces the x-values.
y=	keyword	no	(1)	Expects an expression for the transformation of x-values to follow. May not occur prior to **x=**.
(expressionY)	value string	no	–	is an expression as (expressionX) but the result replaces the y-values and corresponding y-errors.

Actions:

key	source	dest	values	param.	numor	names
op	$j_S \in \mathcal{S}$	$j_R = j_S$	–	–	–	–
x=	$j_S \in \mathcal{S}$	$j_R \to \mathcal{S}$	$x_j(i) = f_x(X, Y, ERR, P^l, \{V^n\})$; $X = x_j(i), Y = y_j(i), ERR = \delta y_j$; $P^l = p^l_j$ and V^n	–	–	xname $\to f_x$(xname)
y=	$j_S \in \mathcal{S}$	$j_R \to \mathcal{S}$	$y_j(i) = f_y(X, Y, ERR, P^l, \{V^n\})$ and $\delta y_j(i) = \lvert f_y(\ldots, Y + \delta y/2) - f_y(\ldots, Y + \delta y/2)\rvert$; with $X = x_j(i), Y = y_j(i), ERR = \delta y_j$; $P^l = p^l_j$ and V^n	–	–	xname $\to f_y$(yname)

Note: It is discouraged to use the older commands: *xformel, yformel, fun, funfun*, just use fxy instead!



8. Arithmetic between data sets (records)

8.1 arit f1 *(factor1)* f2 *(factor2)* (mult|div|vid) (to *numor*)

Arithmetic combination of (exactly) 2 selected data-records. Default is ADDITION, options are also MULTIPLICATIONS and DIVISION. Before combination the data are scaled by factor1 and factor2 respectively. The values of the seconds selection are interpolated to the x-values of the first selection. The resulting record gets the x-values of the first selected record. Errors are combined and transferred to the result.

```
#datreat =>   arit f1 factor1 f2 factor2 [mult |div|vid] [to numor] ↵
```

Examples:

```
#datreat =>   arit ↵
#datreat =>   arit f1 1 f2 -1 ↵
#datreat =>   arit f1 1 f2 1 div ↵
#datreat =>   arit f1 1 f2 1 to 7777 ↵
```

The first command (initial default still there) just adds selected records 1 and 2.
Next a subtraction selected record 1 - selected record 2 is executed.
Next a division record1/record2 is shown.
Finally the result is provided with a numor=7777.

cmd item	type	volatile	# arg	descriptions
f1	keyword	no	1	Set the factor for the (y)-data of the first selected record, initial default is 1.
(factor1)	value string	no (value)	0	value or expression yielding the value of factor1.
f2	keyword	no	1	Set the factor for the (y)-data of the first selected record, initial default is 1.
(factor2)	value string	no (value)	0	value or expression yielding the value of factor1.
mult	option	yes	0	Instead of adding multiply the y-data of the selected records.
div	option	yes	0	Instead of adding divide the y-data of the first records by those of the second record.
vid	option	yes	0	Instead of adding divide the y-data of the second records by those of the first record.
to	keyword	no	1	Set the result record numor to the following value.
numor	value string	no	–	Numor to the record of the result values.

Actions:

$$x_{j_1}(i) \rightarrow x_{j_d}(i) \tag{8.1}$$

$$\left\{f_1 \times y_{j_1}(i)\right\} + | * |/ \left\{f_2 \times \mathcal{I}(\{y_{j_2}\}, x_{j_1}(i))\right\} \rightarrow y_{j_d}(i) \tag{8.2}$$

where $\mathcal{I}(\cdots, x)$ denotes an interpolation.
Note: The determination of errors in the result record assumes that the errors pf the combined records are statistically independent!

8.2 addsels

Addition of data from several (many) selected records. In this case the x-values must be the same for all records. Example use: adding spectra from detectors of a TOF instrument.

```
#datreat => addsels ↵
```

Now the result record is selected. Parameters of the summed records are averaged and the averages and their variances (indicated by prefix 'V') and the number of contributing parameters (indicated by prefix 'N') are updated/added to the parameter section of the result record.

8.3 recsum from *rec1* to *rec2*

Addition of data from several (many) selected records – as addsels – but including the selection of a continuous range of record numbers. In this case the x-values must be the same for all records. Example use: adding spectra from detectors of a TOF instrument.

#datreat => recsum from *rec1* to *rec2* ↵

$y_{ij} \rightarrow y_i^j$

$x_{ij} \rightarrow K_t(x_i^j)q$

9. Special function scaling (spatcorr)

9.1 spqscal

Select any number of $S(q,t)/S(q)$ vs. t spectra and combine these into a scaling plot using parameters for a time dependent displacement function $K_1(t) = \xi(t)$ [*PRE 104, 024503 (2021)*] that immediately prior to spqscal have been fitted.

$$\xi^2 = r^2 \left[\left\{ \left(\frac{t}{\tau} \right)^{-\nu_1} \right\}^a + \left\{ \left(\frac{t}{\tau} \right)^{-\nu_2} \right\}^a \right]^{-1/a} \tag{9.1}$$

with exp1 = ν_1 and exp2 = ν_2. ■

Examples:

```
#datreat =>  in in15runs ↵
#datreat =>  sel 1 -20 ↵
#datreat =>  ac spatcor2 ↵
#datreat =>  cth ↵
#datreat =>  ... enter start parameters ↵
#datreat =>  fit ↵
#datreat =>  spqscal ↵
#datreat =>  fxy x=X*X y=Y ! scale vs (K1*Q)**2 ↵
```

IV Fitting

10 Model fitting 49

10.1 th (name) display information on available theories

10.2 activate *name* (multiply | insert) (range ...) Add a theory to the active list

10.3 cth Change/set (edit) theory model parameters

10.4 thc compute the theory model

10.5 fit Fitting the model parameters to data

11 Advanced fitting topics 55

11.1 Coupling of parameters and record selective theories

11.2 theory range *pnam* min *p1* max *p2* selective application to records

11.3 chgthpar *var p* scale *fs* command line fit parameter setting

11.4 label & couple command line setting parameter couplings

11.5 putpar adding/setting data-record associated parameters

11.6 extthpar *var p* extract value of a specified model fit-parameter

11.7 print *filename par1* ... *par9* saving parameters etc. of a file

11.8 table *tabname* (open | | close) *par1* ... *par...* creating a tex-table containing some parameters

11.9 serfit vs *parname* (par1) ... serial fitting of records

11.10 mexp Automatic fitting of a sum of exponential

12 Creating and modifying theory models 63

12.1 Selecting among available theories from the UNUSED collection

12.2 Create a NEW theory: Define the problem

12.3 Programming Language

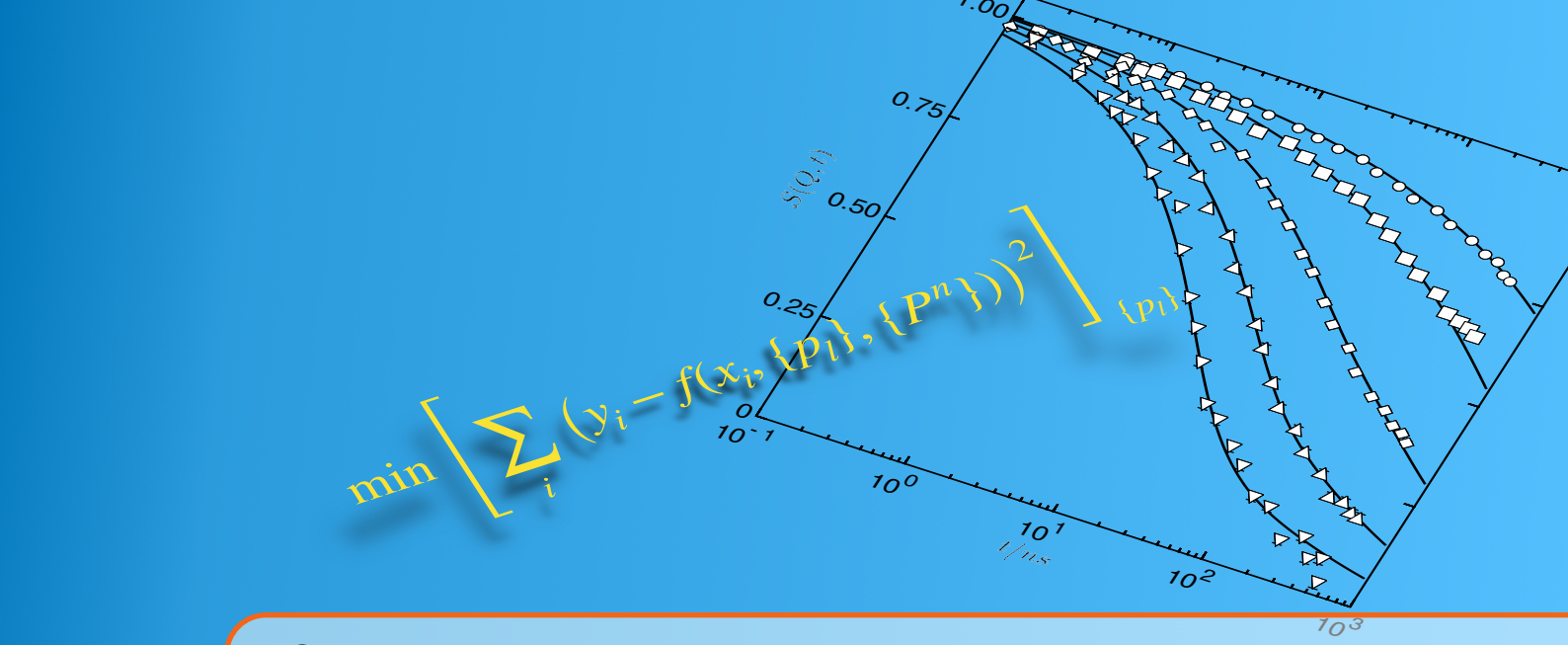
12.4 Use the template generator

12.5 Conversion to the proper Fortran theory function

12.6 Test and use How to use a new theory

12.7 Code Generated illustration for this example

12.8 Hints How to efficiently create and debug theories



10. Model fitting

One of the central purposes of *datreat* is model fitting, either to a combination of available (predefined) model function or to custom made new model functions as described in chapter 12. It is possible to fit several data-records simultaneously, the applied function may then be discriminated by the metadata parameters p_j^m associated with each record j.

10.1 th (name) display information on available theories

The first step needed is the specification of the model to be fitted. Here we describe how this is achieved by using the actual predefined *theories*. To get an overview of the available *theories*

(alphabetically ordered list of names):

```
#datreat =>  th ↵
> th
 ***** available theories available *****
 ---------------------------------------------------------------------------------
  anisodi   beaucage  benoitmf  bkgr      d2sphere  ddiavo    debstar   debye2
  degenne2  degennea  degennes  diffusio  dirosp    dolgushe  echo      echotri2
  eval      gauss     gauss2    gaussft   hetdif    hetdifq   hyperbra  kohl_q
  kohlfast  likhtman  ln_bss1   locrep    locrep2   locrep3   lrep2nr   moderato
  ndebye    ndendpy   ndendri   nmrdiff   nmrdiff2  nmrdiff3  nnvilgis  nrouconf
  nrous_22  nrous_s2  nrous_s4  nrousefn  nrousefy  nrousep   nrouserr  nvilgis
  nzimm_pn  nzimm_s   nzimmaz   peryev    q**n      repmode2  riggch    ring2
  ring2a    roufbead  rouse     rpalin    rpastar2  rpastar3  sansfac   shell3_f
  sinhafra  sofq      srouse    star      strexpo   strexpsq  strexsqq  thickco2
  thickco4  zgbic2    zglam1    zimm
 ---------------------------------------------------------------------------------

  .... for more details on one of these: --> th <theoryname> !
```

In order to get more details on a specific theory use:

```
#datreat =>  th theoryname ↵
```

Example:

```
#datreat =>  th nrousep ↵
 > th nrousep
-------------------------------------------------------------------------------
theory id-name: nrousep
-------------------------------------------------------------------------------
 Rouse by discrete summation with mode restriction
                 ACCELERATED (OMP+IMPOVED ALG.) VERSION!
-------------------------------------------------------------------------------
Parameters( 8):
  1: amplitu  > prefactor
  2: wl4      > rouse rate
  3: n_segmen > effective N for summation
  4: re       > end-to-end radius of Gaussian coil
  5: temp     > temperature if not in parameters
  6: com_diff > center of mass diffusion (if 0, use Rouse default)
  7: pmin     > minimum mode to be included
  8: pmax     > maximum mode to be included
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
INPUT: Parameters that are extracted from the actual considered data records:
..there may be default assumptions, but better make sure that these parameters are set properly!
  1: q        > momentum transfer
  2: temp     > temperature
===============================================================================
OUTPUT: Parameters that are computed and added to the records parameters as information:
  1: diff     > diffusion in cm**2/s
  2: l        > effective segment length
  3: w        > rouse rate: W
  4: wl4      > inferred value of Wl4
.............................................................................
cite:  Rouse, Doi_Edwards !
-------------------------------------------------------------------------------
```

10.2 activate *name* (multiply | insert) (range ...) Add a theory to the active list

Add a new (instance) of an available theory to the model definition. Normally the resulting value from the new activated theory is added to the result of the result of the preceding theories. ■

```
#datreat =>  ac theoryname ↵
```

If the option [multiply] is specified the resulting value from the new activated theory is **multiplied** to the result of the result of the preceding theories instead.

```
#datreat =>  ac theoryname multiply ↵
```

If the option [insert] is specified the resulting value from the **previous** activated theories is **inserted** as independent variable into this theory.

```
#datreat =>  ac theoryname insert ↵
```

The current state of the modeled definition can be displayed using **a**ctive**l**ist:

```
#datreat => al ↵
```

Any present theory model definition can be cleared by deactivate theory: ■

```
#datreat => dac ↵
```

Example:

```
#datreat => dac ↵
#datreat => ac strexpo ↵
#datreat => ac strexpo multiply ↵
#datreat => ac bgr ↵

...
...
 > al
  ***** activated theories *****
 current parametersetting:
 theory   strexpo
      amplitu     0.000000       0.00       0.0000
      tau         0.000000       0.00       0.0000
      beta        0.000000       0.00       0.0000
      qexp        0.000000       0.00       0.0000
 theory   strexpo    multiply
      amplitu     0.000000       0.00       0.0000
      tau         0.000000       0.00       0.0000
      beta        0.000000       0.00       0.0000
      qexp        0.000000       0.00       0.0000
 theory   bkgr
      level       0.000000       0.00       0.0000
      slope       0.000000       0.00       0.0000
 end
```

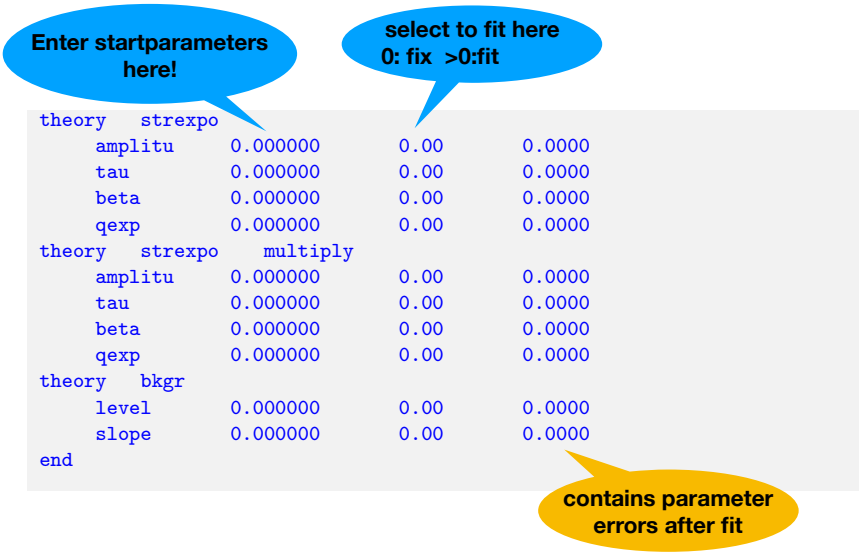
Here the defined model function (with no physical sense) would be:

$$f_{\text{model}} = a_1 \exp(-(x/\tau_1)^{\beta_1}) \times a_2 \exp(-(x/\tau_2)^{\beta_2}) + (b_{\text{level}} + x b_{\text{slope}}) \tag{10.1}$$

as you may note here all parameters still have value 0. So in the next step these have to be given (start) values and selection which are to be fitted.

10.3 cth Change/set (edit) theory model parameters

The command (makro call) **c**hange**th**eory can be used to set up the theory model parameters and to control which parameters shall be fitted. It opens an editor which allows editing: ■




```
theory   strexpo
     amplitu     0.000000       0.00       0.0000
     tau         0.000000       0.00       0.0000
     beta        0.000000       0.00       0.0000
     qexp        0.000000       0.00       0.0000
theory   strexpo    multiply
     amplitu     0.000000       0.00       0.0000
     tau         0.000000       0.00       0.0000
     beta        0.000000       0.00       0.0000
     qexp        0.000000       0.00       0.0000
theory   bkgr
     level       0.000000       0.00       0.0000
     slope       0.000000       0.00       0.0000
end
```

The first numerical column represents the (start) parameters of the theory. The second column selects parameters for fitting. A zero value means that the parameter shall be held fixed, if it shall be fitted is must be set to a value in the order of magnitude of the expected parameter value.
The third column may contain parameter errors determined by a (previous) fit step, here they can be ignored.

Note 1: Numerical input here may also be an *expression*, it will be evaluated upon rereading the model definition and the replaced by the corresponding value.

Note 2: Theories may be removed or added at this level by deleting or inserting the text blocks starting with *theory*.

Note 3: **cth** is a makro the opens an editor with file *lastth* thus allowing editing. After leaving the editor then it executes the command **acl** to read the contents of lastth. The preferred editor can be set in datreat/makros/cth.

TBD(datreat src):make editor selection global!

Note 4: Check that the new theory is correctly displayed in the *datreat* window. If not directly reissue *cth* and correct typos or other errors.

Note 5: For more advanced fitting options like parameter coupling, record selective theories, restriction of fitted points etc. see chapter 11!

10.4 thc compute the theory model

theorycompute computes the theory at the x-values of the selected records or on a specified x-range. ■

```
#datreat =>  thc ↵
#datreat =>  thc  n np x1 x1 x2 x2 ↵
```

the second line is the form to specify th x-range to be computed, *np* denotes the number of points per curve (record) to be computed and *x1* ··· *x2* is the x-range to be covered. If $n < 0$ then the x-axis division is logarithmic (use this if *log_x*-plots are intended.

Note 1: The computed curves are stored with new record addresses that are internally associated with the selected data-records. Plotting automatically displays the computed curves as lines with the same color as the pertaining data.

Note 2: The numerical-id *numor* of the computed curve is the negative of the parent data record numor.

Note 3: The automatic association of data and fit is removed as soon as a new sel has been issued.

10.5 fit Fitting the model parameters to data

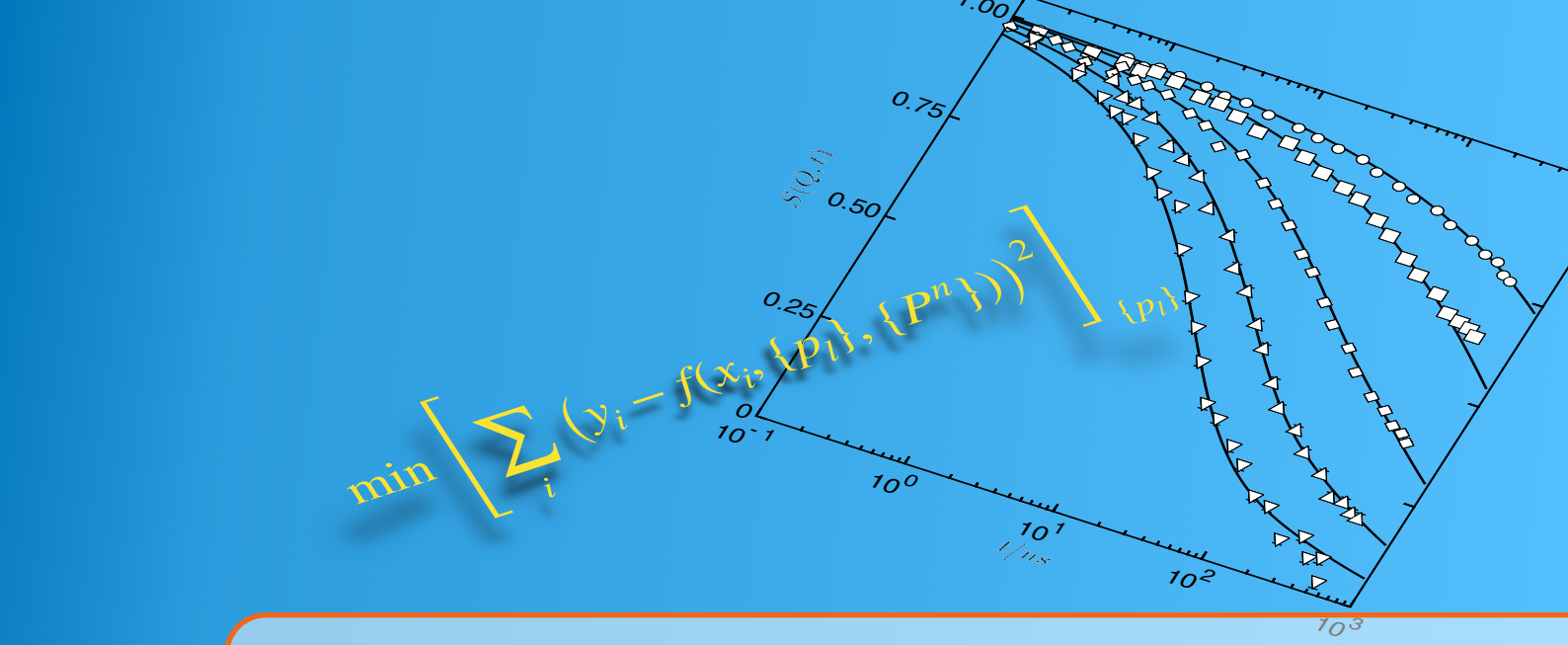
Fits the theory model (setup see above) to the selected records at the x-values over a specified x-range. ■

Fits the theory model (setup see above) to the selected records at the x-values over a specified x-range.

```
#datreat =>  fit ↵
#datreat =>  fit maxfn m ↵
#datreat =>  fit  x1 x1 x2 x2 ↵
```

the first line initiates fitting with the (current | default) range settings. The second form allows to modify the maximum (approximately) allowed number of model evaluations *m*. The third line shows how to restrict the x-range for fitting (**permanent** until changed.
The model with the resulting parameters is automatically computed at the end of the fit and all remarks and notes to **thc** (section: 10.4) apply.

Finally use **plot** to view the result.
Note: More on fitting in the next chapter and in chapter 18.



11. Advanced fitting topics

11.1 Coupling of parameters and record selective theories

It may be desired that some fit-parameters in a *theory model* have a **fixed** relation, i.e. are coupled. Further it may be desired to assign specific instances of a theory to a subset of all data-records that are fitted simultaneously. The records are to be discriminated by a record related parameter (in the following example this parameter has the name **_xwidth**). First (max. 4 characters wide) **labels**

are assigned to each of the lead parameters (those that are fitted). The dependent parameters then are **coupled** by a sequence of **label coefficient**. During fit a dependent parameter p_d then gets the value:

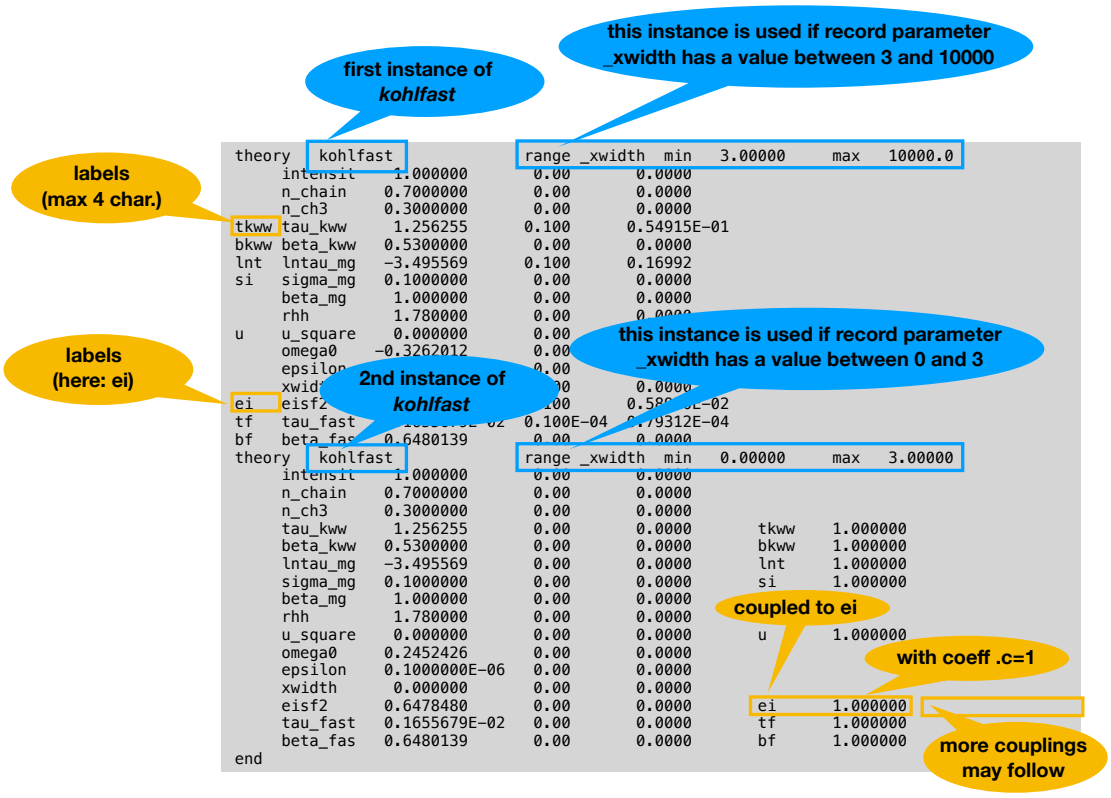
$$p_d = p_d^0 + \sum_{\{i\}} c_{i,d}\,(p_i - p_i^0) \tag{11.1}$$

where p^0 means the starting value of a theory parameter, p_i any (labelled) lead parameter, $c_{i,d}$ the coupling coefficient and $\{i\}$ the set of parameters (indices) that are coupled to p_d. The easiest way

to setup parameter coupling is to *activate* the required theories and do the further editing using **cth** (command line/makros methods: see below). In the following an example is shown to illustrate the structure of the required input.

```
theory   kohlfast                 range _xwidth  min    3.00000      max    10000.0
     intensit     1.000000         1.00        0.0000
     n_chain     0.7000000         0.00        0.0000
     n_ch3       0.3000000         0.00        0.0000
tkww tau_kww      1.256255        0.100       0.54915E-01
bkww beta_kww    0.5300000         0.00        0.0000
lnt  lntau_mg    -3.495569        0.100       0.16992
si   sigma_mg    0.1000000         0.00        0.0000
     beta_mg      1.000000         0.00        0.0000
     rhh          1.780000         0.00        0.0000
```

```
u     u_square     0.000000        0.00        0.0000
      omega0      -0.3262012       0.00        0.0000
      epsilon      0.1000000E-06   0.00        0.0000
      xwidth       0.000000        0.00        0.0000
ei    eisf2        0.6478480       0.100       0.58930E-02
tf    tau_fast     0.1655679E-02   0.100E-04   0.79312E-04
bf    beta_fas     0.6480139       0.00        0.0000
theory    kohlfast                range _xwidth  min    0.00000      max    3.00000
      intensit     1.000000        1.00        0.0000
      n_chain      0.7000000       0.00        0.0000
      n_ch3        0.3000000       0.00        0.0000
      tau_kww      1.256255        0.00        0.0000        tkww     1.000000
      beta_kww     0.5300000       0.00        0.0000        bkww     1.000000
      lntau_mg    -3.495569        0.00        0.0000        lnt      1.000000
      sigma_mg     0.1000000       0.00        0.0000        si       1.000000
      beta_mg      1.000000        0.00        0.0000
      rhh          1.780000        0.00        0.0000
      u_square     0.000000        0.00        0.0000        u        1.000000
      omega0       0.2452426       0.00        0.0000
      epsilon      0.1000000E-06   0.00        0.0000
      xwidth       0.000000        0.00        0.0000
      eisf2        0.6478480       0.00        0.0000        ei       1.000000
      tau_fast     0.1655679E-02   0.00        0.0000        tf       1.000000
      beta_fas     0.6480139       0.00        0.0000        bf       1.000000
end
```



The example illustrates the case that spectra form two instruments (backscattering and tof) with possibly different intensity scaling, different resolution (given by record related parameters) are to be fitted simultaneously. The *range* selection discriminates the instruments by the channel width

record parameter *_xwidth* and in the theory the sample related physical parameters are kept equal in both instances of the theory *kohlfast* by coupling. To achieve that these parameters are provided with *labels* in the first instance and these labels are used in the second instance to specify the couplings. In this case the parameters shall always have the same value, therefor there is one **coupling coefficient = 1** and the corresponding *starting values must be the same*!

Note 1: in cases like this (coupling = 1) *datreat* will issue a *warning* if the starting values differ. (Use coupling = 0.99999 to avoid the warning if the unequal starting point setting is (exceptionally) intended!

Note 2: in cases where the sum of two parameters shall be kept (i.e. total amplitude) use (coupling(s) = -1)!

11.2 theory range *pnam* min *p1* max *p2* selective application to records

If a record parameter associated range shall be associated to a theory instance this has to be specified when the theory is activated.

#datreat => **ac thnam [multiply] range rparname min** *value* **max** *value* ↵

Activates a new instance of theory *thnam* (optionally as *multiply* and active if the record parameter *rparname* has a value between the specified min-max values.

11.3 chgthpar *var p* scale *fs* command line fit parameter setting

chgthpar allows the specification of single theory parameters via command (particular useful in makros). ■

#datreat => **chgthpar thnam** *instance* **parname par** *value* **scale** *value* ↵

Example:

```
#datreat => dac ↵      clear theory definition
#datreat => ac bkgr ↵      activate theory bkgr
#datreat => ac ndebye ↵      activate theory ndebye
#datreat => chgthpar bkgr 1 level par 0.1 scale 0.01 ↵      set parameter and fit scale
#datreat => chgthpar bkgr 1 slope par 0.0 scale 0 ↵
#datreat => chgthpar ndebye 1 ampli par 0.01 scale 0.01 ↵
#datreat => chgthpar ndebye 1 n par 333 scale 0.00 ! do not fit ! ↵
#datreat => chgthpar ndebye 1 l par 10 scale 0.100 ↵
#datreat => chgthpar ndebye 1 nu par 0.4 scale 0.100 ↵
#datreat => chgthpar ndebye 1 chi par 1e-3 scale 1e-3 ↵
#datreat => chgthpar ndebye 1 withsq par 1 scale 0.0 ↵
#datreat => al ↵
 ***** activated theories *****
 current parametersetting:
 theory   bkgr
```

```
    level     0.1000000     0.100E-01   0.0000
    slope     0.000000      0.00        0.0000
theory   ndebye
    ampli     0.1000000E-01  0.100E-01   0.0000
    n         333.0000      0.00        0.0000
    l         10.00000      0.100       0.0000
    nu        0.4000000     0.100       0.0000
    chi       0.1000000E-02  0.100E-02   0.0000
    withsq    1.000000      0.00        0.0000
end
end of data
```

Note: This command is mainly intended to allow parameter settings by a *makro*.

11.4 label & couple command line setting parameter couplings

Specify linear couplings between fit-parameters in the model definition via command. ■

#datreat => **label thnam** *instance* **parname** *label* assign *l*abel to parname ↵

#datreat => **couple thnam** *instance* **parname** *label coeff* couple parameter with label *l*abel to parname with coeff. ↵

Example:

```
#datreat =>  label bkgr 1 level abc ↵     assign label abc to level
#datreat =>  couple ndebye 1 ampli abc -1 ↵     couple parameter ampli to level wit c=-1
```

11.5 putpar adding/setting data-record associated parameters

Add a new parameter (with value) or change value of record associated parameters in **all selected records**. These may be used to transfer individual parameter values (for records selected prior to **fit** or **thc**) to the active theories. Thereby supplying a mechanism to simultaneously fit data with various external parameters as e.g. *temperature*, *concentration*, ⋯ . An often used parameter of this kind is e.g. q, the momentum transfer value (usually should be already present in data, if not: use *putpar*). ■

#datreat => **sel** ***selection list*** first select the records to be affected ↵
#datreat => **putpar** *parname value* assign *v*alue to parname ↵

Example:

```
#datreat =>  sel 1 2 3 ↵     select records 1 and 2 and 3
#datreat =>  putpar temp 373 ↵     set/create parameter temp to value 373
```

Note1: In the present version of datreat the name *parname* has a length limit of 8 characters.
Note2: The parameters are displayed in the plot legend.
Note3: The parameters can be inspected by **dir** *parname1 parname2* ··· or for selected records only by **dsl** *parname1 parname2* ··· .
Note4: The parameters may be used in selection criteria by **sel all** *parname value* **band** *value*
Note5: Similar the parameters may be used in the **range** specification of theory validity (see above).

11.6 extthpar *var p* extract value of a specified model fit-parameter

chgthpar allows the extraction specification of a single theory parameters and its error via command (particular useful in makros). ■

```
#datreat => extthpar thnam i parname ↵
```

writes the value of parameter *parname* of the i-th instance of theory *thnam* on a variable *lastpar* and the associated error to *lastperr*.

11.7 print *filename par1* ... *par9* saving parameters etc. of a file

Append the values of (up to 9) expressions to a file. ■

```
#datreat => print myfile (expr1) (expr2) ...      Append a line with values of (expr..) to myfile
↵
```

Note1: To use recent fit results (parameters), use the **extthpar** *thname instance parname* command. It writes the parameter value and error to variables *lastpar* and *lastperr* respectively. If necessary copy these to some suitable name (**set myname1 (lastpar)**, ...) and extract more parameters. The *myname1*... variables then can be used in the expressions to be printed.

Note2: To use recent fit results (parameters), you may also use the user-functions **th_par(ipar,ith)** and/or **th_err(ipar,ith)**. Note that the addressing depends on the position in the model definition.

Note3: The parameters of the last (only!) instance of the theory in the model definition are also available as variables (use **vars?** to display! Errors with prefix **e.**) these may be used in the expressions to be *print*ed. The *ssq*-parameter of the recent fit is on **ssq0**.

11.8 table *tabname* (open | | close) *par1* ... *par...* creating a tex-table containing some parameters

Create a table (*tabname.tex*) than can be inculdes into latex documents. In a first line (open) the table header is written and the columns are defined. In following N-lines the line entries are added. Finally (close) the table environment is closed. ■

#datreat => **table mytab open** *col1* **..** *coln* Write the table header ↵
#datreat => **table mytab [F:format]** *val1* **..** *valn* one line of columns ↵
#datreat => **table mytab [F:format]** *val1* **[E:** *val1_err* **] .. [F:...]** *valn* more line of columns, E: ipeceeds an associated error ↵
#datreat => **table mytab close** append closing latex statements to mytab.tex ↵

Example :

#datreat => **dotab2 mytab** call makro dotab2 (see below) to create to mytab.tex ↵

Makros dotab2 and tebenter The macro dotab2 makes header and closing and in between calls the makro tabenter for each table line. The header and the table-line commands must match!

```
c ==============================================================================
c dotab2
c ==============================================================================
makro _tabnam_

table _tabnam_ open  polymer $Wl^4\,[\si{\angstrom}^4/\mathrm{ns}]$  b    ....
$R_\mathrm{e}\,[\si{\angstrom}]$  $d_\mathrm{NSE}\,[\si{\angstrom}]$      ....
$d_\mathrm{eff}\,[\si{\angstrom}$] $t_b\,[\mathrm{ns}]$                   ....
$t_w\,[\mathrm{ns}]$ $\alpha_0$ $\chi^2$

tabenter _tabnam_  pe190_fitneux
tabenter _tabnam_  pe190_fitneux_ng
tabenter _tabnam_  pe190_fitneux_w
...
tabenter _tabnam_  spp_fitneux
tabenter _tabnam_  spp_fitneux_ng
tabenter _tabnam_  spp_fitneux_ng_tw

table _tabnam_ close


c ==============================================================================
c tabenter
c ==============================================================================
makro _tn_ _fn_

purge all
inplus _fn_
purge fits
sel 1 -(nbuf)
fit

table _tn_  _fn_ F:f12.0 +wl4 E: +e.wl4 F:f8.2 +b E: +e.b                 ...
F:f7.1 +re E: +e.re +dnse +deff  +t0_locrep +tw_locrep F:f8.2             ...
+alpha0 E: +e.alpha0 F:f7.1 +ssq
```

Note1: To use recent fit results (parameters), use the **extthpar** *thname instance parname* command. It writes the parameter value and error to variables *lastpar* and *lastperr* respectively. If necessary copy these to some suitable name (**set mynameI (lastpar)**, ...) and extract more parameters. The *mynameI*... variables then can be used in the expressions to be printed.

Note2: To use recent fit results (parameters), you may also use the user-functions **th_par(ipar,ith)**

and/or **th_err(ipar,ith)**. Note that the addressing depends on the position in the model definition.

Note3: The parameters of the last (only!) instance of the theory in the model definition are also available as variables (use **vars?** to display! Errors with prefix **e.**) these may be used in the expressions to be *print*ed. The *ssq*-parameter of the recent fit is on **ssq0**.

Note4: The items: *col1* etc. (in the opening line) may contain latex but without blanks.

Note5: The items: *col1* etc. (in the add column lines) may be preceeded by (fortran) format specifiers that are valid for the following items, these are identfied by the prefix **F:** the rest of the item (containing no blanks) is interpreted as format.

Note5: The items error indicator **E:** must immediately preceed the corresponding error value, any format specifier for the error must be given in front of **E:** ! Value and error pertain to one column in the header!

11.9 serfit vs *parname* (par1) ... serial fitting of records

Fits selected records one after the other and creates additional records with the resulting fit parameter values as function of parameter *parname*. ■

The model specification is done exactly as for the normal fitting procedure. During the series the fitted parameters of the previous step serve as start parameters for the next one.

```
#datreat => serfit vs parameter    Execute series of fits, create records with results vs
parameter ↵
```

Note: There is no guarantee that the fits will stay on track during the series. Have a look to check!

Example (see TRAINING/DENDRIMER_datreat):

```
#datreat => purge all ↵    clear memory (optional)
#datreat => dac ↵    clear theory models
#datreat => in report7741_6.5.dtr ↵    input data (here: series of NSE spectra)
#datreat => ac strexpo ↵    activate stretched theory stretched exp.
#datreat => cth ↵    set start parameters
#datreat => serfit vs q ↵    perform series of fits an organize results as function of parameter
q
#datreat => dir ↵    display records (see the results)

 39:#  77410101 : st1ampli :st1ampli vs q       >"_6SB5_20C _6SB5 serf"
 40:#  77410102 : st1tau   :st1tau   vs q       >"_6SB5_20C _6SB5 serf"
 41:#  77410103 : st1beta  :st1beta  vs q       >"_6SB5_20C _6SB5 serf"
 42:#  77410104 : st1qexp  :st1qexp  vs q       >"_6SB5_20C _6SB5 serf"
 43:#  77410105 : ssq      :ssq      vs q       >"_6SB5_20C _6SB5 serf"
```

Position 39 to 42 now contain fit results for the **st**rexpo instance**1** parameters **ampli**tude, **tau**, **beta**,

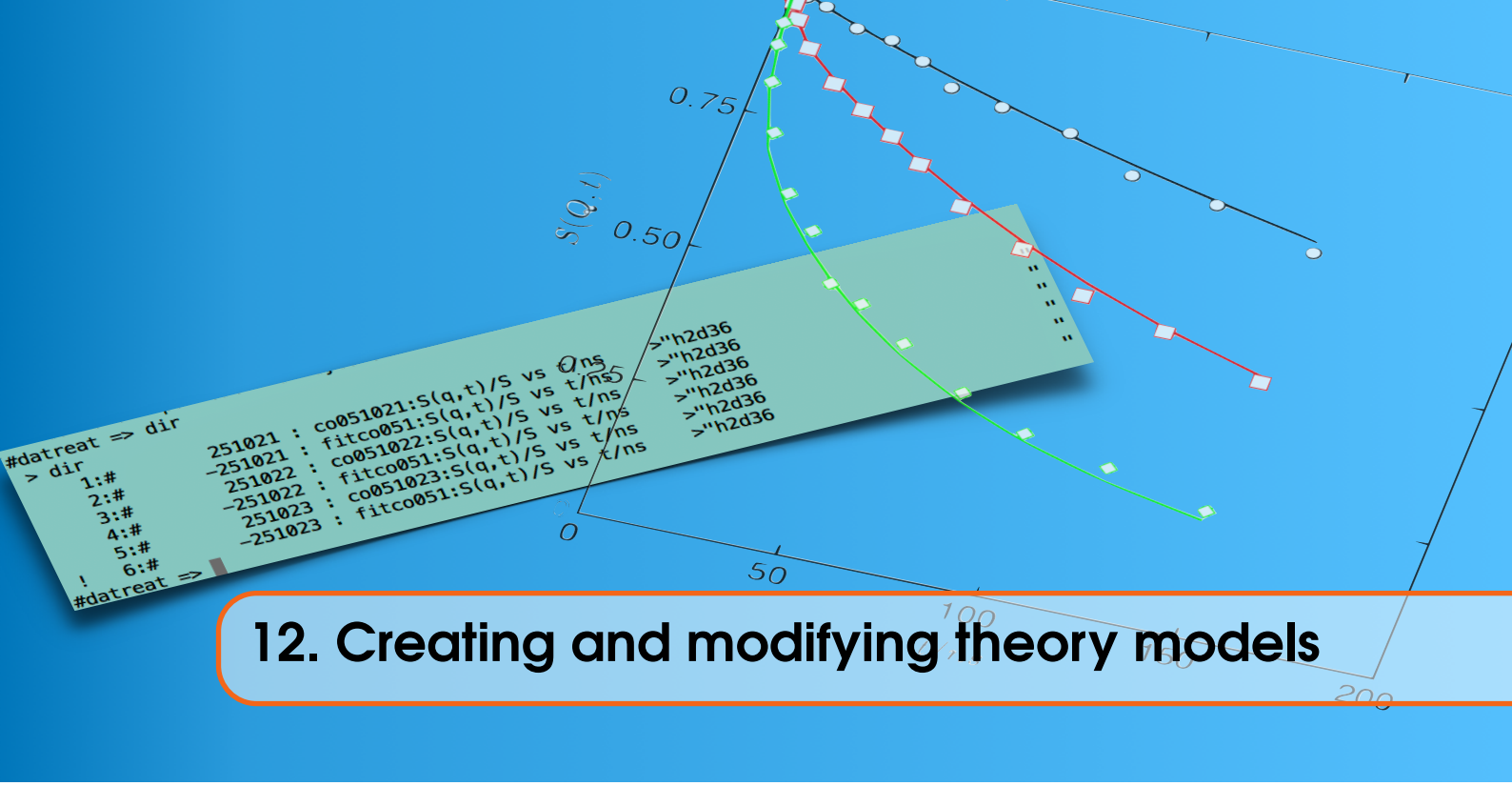
and **qexp** (created names contain truncated IDs to fit into the 8 character restriction, hope its unique!) as a function of q. Finally 43 contains the fits **s**um of **sq**uares values. All in terms of standard datreat records that -after selection- may be plotted or saved (msave).

11.10 mexp Automatic fitting of a sum of exponential

Automatically creates a m-exp model for the record data.

#datreat => **mexp n** *m* **[rms** *rmslevel***]** Determine a n≤m-exp model ↵

Note: After **mexp** is finished the current theory model contains the N-exp model definition that has been found by mexp. Use **thc** and **plot** to check this against the initial data! Any previous theory model definition will be overwritten!



12. Creating and modifying theory models

12.1 Selecting among available theories from the UNUSED collection

Datreat can offer a limited number of theories at one time (currently 80) that can be listed by the command **th** . However, there may be more theories present in the stock (of sources for theories) or there may be newly created *theories* (see following sections). The sources that are actually used are located in subdirectory *~/Desktop/datreat/src/theos*, those that are in stock but currently not active are stored in subdirectory *~/Desktop/datreat/src/unused_theos*. To be able to change the selection of theories offered by *datreat* during runtime perform the following steps in a *terminal*.

```
...> cd ~/Desktop/datreat/src/theos ↵      change directory to the theos subdir
...> mv th_xyz.f90 ../unused_theos ↵       move unwanted theory xyz to unused
...> mv th_xyz1.f90 ../unused_theos ↵      possibly repeat this with further theories
...> mv ../unused_theos/th_abc.f90 ./ ↵    get the now desired theories from unused
...> make clean ↵      remove the old compiled theories
...> make ↵      compile the actual theory collection
...> cd .. ↵      change directory to ~/Desktop/datreat/src
...> make ↵      compile and link datreat
```

If the last *make* was executed without errors, quit any running instances of *datreat* and start in new such that the modified theory collection becomes active.

Note: Here we assumed that *datreat* is installed in *~/Desktop/datreat*. The character ~, is short for your home directory (Mac German keyboard: option-n).

12.2 Create a NEW theory: Define the problem

The model to be used for data-fitting has to be selected and expressed in terms of some formula or procedures.

- Identify parameters that are to be fitted, they are common to all data records that enter the analysis. Select names.

- Identify parameters that are associated to the data records and possibly discriminate them, e.g. q-value, temperature, ⋯.
- What is the ordinate (x-value)?

12.3 Programming Language

The programming language for *datreat* theories is *fortran*. It is recommended (but not mandatory) to use modern (2003/2008) language features.
Most of the work to setup the scaffold of the theory function in *fortran* is done by using the template generator (see below). However, the contents in terms of a proper formula expression or more sophisticated procedure must be supplied as input to the template generator or edited at a later stage into the theory function source code.

12.4 Use the template generator

The template generator converts a template input containing the information on parameters and names etc. into a theory function source code. The *template generator* some examples are in subdirectory ~/Desktop/datreat/src/UTILITIES. To start defining a theory:

```
...> cd ~/Desktop/datreat/src/UTILITIES ↵
...> cd cp template_template my_new_template ↵    copy a template template to a chosen name
...> edit my_new_template ↵    open the new template with an editor and the edit
```

Note that names are restricted to a length of 8 characters. The template input is explained using the following example (**#KEYWORD** introducing input groups for the template generator. The **template template** has the following form:

```
#THEORY name
        hear a short description of what the theory with name (within datreat) does.
        It may contain several lines.
        It will be displayed by datreat when the command "th name" is issued.
#CITE
        Put here any relevant literature references for information, displayed upon "th name"
#PARAMETERS
        nam_p1           ! first theory parameter (possibly fittable) named nam_p1 (typically here ampli=
        nam_p2           ! second parameter with given name nam_p2
        ...              ! further parameters may follow
        ...
#RECIN-PARAMETERS    (allow 8 chars for name, start only beyond 8 with default)
        nam_pjn1         defval1 ! name of parameter that is taken from the parameters section of each record
        nam_pjn2         defval2 ! name of parameter that is taken from the parameters section of each record
        ...           ...        ! name of parameter that is taken from the parameters section of each record
#RECOUT-PARAMETERS
        pjout1               ! name of any record j associated parameters that shall be set/modified by the theory
        pjout2               ! next, if applies
        ...                  ! until finished
#VARIABLES
        double precision :: list of fortran variables that shall be used
        integer          ::  "
#IMPLEMENTATION
     xvar   = x              ! extract the x-value and copy it to the variable name you want to use for x
     ...                     ! valid fortran staements
     ...                     ! that compute the desired function result stored say on a variable y
     th  = y                 ! this will be returnde as function value

#SUBROUTINES

    here add any subroutines and functions to be used to compute results needed to arrive at the value for
    y.

    Note: these will be appended to the theory function by a CONTAINS statement. This means that
    variables that are defined above (i.e. defined in the proper th-function that emerges) are also
    known to the here located subroutines (like former COMMON) without the need of any specification.
    HOWEVER, if a variable with the same name is defined locally in one of the subroutines it is
    local to that subroutine and NOT shared with the other parts of the program.

    Further CONTAINS avoids name conflicts with equally named subroutines/functions used in other
    theories. As a consequence the here entered subroutines are only visible to this theory.
#END
```

Example:

given name of theory

any comment short description

what to cite

theory (fit) parameters ! comment

parameter from records

this is returned as theory value

any specific subroutines and functions to be called in the main implementation

```
001  #THEORY locrep
002          generalized local reptation expression
003          along the lines of deGennes, but with finite summation of integrals
004          and lenthscale, timescale and fluctuation ratio as parameters
005  #CITE   P.G. De Gennes. Coherent scattering by one reptating chain.
006          Journal de Physique, 1981, 42 (5), pp.735-740. <10.1051/jphys:01981004205073500>
007  #PARAMETERS
008          ampli            ! prefactor
009          b                ! fluctuation intensity (relative)
010          a                ! length scale
011          tau              ! timescale
012          lz               ! total length
013  #RECIN-PARAMETERS    (allow 8 chars for name, start only beyond 8 with default)
014          q            0   ! q-value    default value
015  #RECOUT-PARAMETERS
016  #VARIABLES
017       double precision   :: t
018  #IMPLEMENTATION
019       t  = x              ! since we prefer to call the independent variable t, x must be copied to t
020       th = ampli * local_reptation(q*a, t/tau, lz)
021
022  #SUBROUTINES
023    function local_reptation(q, t, L) result(val)
024      implicit none
025      double precision, intent(in)   :: q, t, L
026      double precision               :: val
027      double precision, parameter    :: sqp = sqrt(4*atan(1d0))
028
029      val = 0.72D2 * (sqrt(t) * q ** 4 * exp((-0.2D1 * L * q ** 2 * t -
030       #0.3D1 * L ** 2) / t / 0.12D2) / 0.36D2 + sqrt(0.3141592653589793D1
031       #) * (q ** 2 * t / 0.3D1 + L) * q ** 4 * exp(t * q ** 4 / 0.36D2) *
032       # (-erfc((q ** 2 * t + 0.3D1 * L) * t ** (-0.1D1 / 0.2D1) / 0.6D1)
033       #+ erfc(sqrt(t) * q ** 2 / 0.6D1)) / 0.72D2 - sqrt(t) * q ** 4 / 0.
034       #36D2) * B / q ** 4 * 0.3141592653589793D1 ** (-0.1D1 / 0.2D1) / L
035       #+ 0.72D2 * (A * exp(-q ** 2 * L / 0.6D1) * sqrt(0.3141592653589793
036       #D1) + (A * L * q ** 2 / 0.6D1 - A) * sqrt(0.3141592653589793D1)) *
037       # 0.3141592653589793D1 ** (-0.1D1 / 0.2D1) / q ** 4 / L
038
039
040    end function local_reptation
041  #END
```

The #KEYWORD lines start the different input groups.

- line 001: #THEORY locrep. Give a name (here: *locrep*) to the theory.
- lines 002-004: description of the theory
- lines 005-006: citation
- lines 007-012: the fit parameters, list of names and ! explanations
- lines 013-014: parameters associated with the data records, here q. Name default value ! Explanation.
- line 015: output parameters that are added to the record parameter section (here empty)
- lines 016-017: fortran variable definition for any additional variables to be used in the theory (not needed for parameters, they are automatically created).
- lines 018-020: implementation: fortran code to compute the theory
- lines 022-040: any function and subroutines that are needed to do the computation

12.5 Conversion to the proper Fortran theory function

```
...> th_template_generator < my_new_template ↵      processes the template to out.f90
...> cp out.f90 ../theos/th_mynewthname.f90 ↵      copy source to theos
...> cd ../theos ↵      change directory to ~/Desktop/datreat/src/theos
...> make ↵      compile
```

..in most cases some residual fortran (syntax) errors will pop up (or not?!), in that case edit the file *~/Desktop/datreat/src/theos/th_mynewthname.f90*, correct it and try *make* again until no error messages remain (see HINTS for strategies to avoid errors!). The theory inclusion into **datreat** can be finished.

```
...> make clean ↵    make sure that no old residues interfere
...> make ↵    in subdirectory theos this now compiles the theories
...> cd ../ ↵    change to subdirectory ~/Desktop/datreat/src
...> make ↵    rebuild datreat including the new theory.
```

12.6 Test and use — How to use a new theory

After successful *make* the new theory should be listed if datreat is started and the commend **th** is entered. **th thname** then will yield the information on parameters etc. (here thname=*locrep*). To use it give the command **ac thname** and then use **cth** to enter parameter values. Finally (after selecting some suitable record) use **thc** to compute and then **plot** and/or **edit** to check the resulting values.

12.7 Code Generated — illustration for this example

The above example yields the following *Fortran* source code.

```
 FUNCTION th_locrep(x, pa, thnam, parnam, npar,ini, nopar ,params,napar,mbuf)
!==============================================================================
!  generalized local reptation expression along the lines of deGennes, but with finite summation of integrals and lenthscale, timescale and fluctuation ratio as
!  Journal de Physique, 1981, 42 (5), pp.735-740. <10.1051/jphys:01981004205073500>
      use theory_description
      implicit none
      real    :: th_locrep
      character(len=8) :: thnam, parnam (*)
      real    :: pa (*)
      real    :: x , xh
      integer :: mbuf, nparx, ier, ini, npar, iadda
      integer, intent(inout) :: nopar
      character(len=80), intent(inout) :: napar(mbuf)
      real, intent(inout) :: params(mbuf)

      double precision, parameter :: Pi = 4*atan(1d0)
      integer                     :: actual_record_address

! the internal parameter representation
     double precision :: ampli      ! prefactor
     double precision :: b          ! fluctuation intensity (relative)
     double precision :: a          ! length scale
     double precision :: tau        ! timescale
     double precision :: lz         ! total length
! the recin parameter representation
     double precision :: q          ! q-value    default value
! the reout parameter representation

     double precision :: th

     double precision   :: t
!
! ----- initialisation -----
    IF (ini.eq.0) then
       thnam = 'locrep'
       nparx =        5
       IF (npar.lt.nparx) then
           WRITE (6,*)' theory: ',thnam,' no of parametrs=',nparx,' exceeds current max. = ',npar
          th_locrep = 0
          RETURN
       ENDIF
       npar = nparx
! >>>>> describe theory with >>>>>>>
       idesc = next_th_desc()
       th_identifier(idesc)   = thnam
       th_explanation(idesc)  = " generalized local reptation expression along the lines of deGennes, but with finite summation of integrals and lenthscale, time
       th_citation(idesc)     = " Journal de Physique, 1981, 42 (5), pp.735-740. <10.1051/jphys:01981004205073500>"
!       --------------> set the parameter names --->
        parnam ( 1) = 'ampli   '  ! prefactor
        parnam ( 2) = 'b       '  ! fluctuation intensity (relative)
        parnam ( 3) = 'a       '  ! length scale
        parnam ( 4) = 'tau     '  ! timescale
        parnam ( 5) = 'lz      '  ! total length
! >>>>> describe parameters >>>>>>>
        th_param_desc( 1,idesc) = "prefactor" !//cr//parspace//&
        th_param_desc( 2,idesc) = "fluctuation intensity (relative)" !//cr//parspace//&
        th_param_desc( 3,idesc) = "length scale" !//cr//parspace//&
        th_param_desc( 4,idesc) = "timescale" !//cr//parspace//&
        th_param_desc( 5,idesc) = "total length" !//cr//parspace//&
! >>>>> describe record parameters used >>>>>>>
        th_file_param(:,idesc) = " "
```

```
        th_file_param(  1,idesc) = "q        > q-value    default value"
! >>>>> describe record parameters creaqted by this theory >>>>>>>
        th_out_param(:,idesc)  = " "
!
        th_locrep = 0.0

        RETURN
     ENDIF
!
! ---- transfer parameters -----
      ampli    =      pa( 1)
      b        =      pa( 2)
      a        =      pa( 3)
      tau      =      pa( 4)
      lz       =      pa( 5)
! ---- extract parameters that are contained in the present record under consideration by fit or thc ---
      iadda = actual_record_address()
! >>> extract: q-value    default value
      xh =      0
      call parget('q       ',xh,iadda,ier)
      q        = xh
!
! ----------------------------------------------------------------
! ---------------------- implementation --------------------------
! ----------------------------------------------------------------
!
     t  = x              ! since we prefer to call the independent variable t, x must be copied to t
     th = ampli * local_reptation(q*a, t/tau, lz)

     th_locrep = th

! ---- writing computed parameters to the record >>>

 CONTAINS

! subroutines and functions entered here are private to this theory and share its variables

  function local_reptation(q, t, L) result(val)
    implicit none
    double precision, intent(in)   :: q, t, L
    double precision               :: val
    double precision, parameter    :: sqp = sqrt(4*atan(1d0))

    val = 0.72D2 * (sqrt(t) * q ** 4 * exp((-0.2D1 * L * q ** 2 * t -
     #0.3D1 * L ** 2) / t / 0.12D2) / 0.36D2 + sqrt(0.3141592653589793D1
     #) * (q ** 2 * t / 0.3D1 + L) * q ** 4 * exp(t * q ** 4 / 0.36D2) *
     # (-erfc((q ** 2 * t + 0.3D1 * L) * t ** (-0.1D1 / 0.2D1) / 0.6D1)
     #+ erfc(sqrt(t) * q ** 2 / 0.6D1)) / 0.72D2 - sqrt(t) * q ** 4 / 0.
     #36D2) * B / q ** 4 * 0.3141592653589793D1 ** (-0.1D1 / 0.2D1) / L
     #+ 0.72D2 * (A * exp(-q ** 2 * L / 0.6D1) * sqrt(0.3141592653589793
     #D1) + (A * L * q ** 2 / 0.6D1 - A) * sqrt(0.3141592653589793D1)) *
     # 0.3141592653589793D1 ** (-0.1D1 / 0.2D1) / q ** 4 / L


  end function local_reptation
 end function th_locrep
```

12.8 Hints How to efficiently create and debug theories

Debugging and verification and optimisation of a new *theory* may be challenging. The most effective approach to quickly yield reliable results is to first write a procedure (function or subroutine) that contains the core implementation of the new *theory* on its own. To test, it is should be embedded in (or called from) a short main program that serves for testing only. After the *theory* works satisfactory in that environment, its genuine source code can be copied into either an appropriate *template* or directly into a previously created *theory* scaffold.

```
!!!
```

```
   t = 2*i
   call NNvilgis(q,t,wl4,l,rmesh,N,sqt,sqti)
   write(*,*)q,t,sqt0,sqt/sqt0, sqti/sqt0
 enddo

CONTAINS

!!!
```

```
end subroutine NNvilgis

!!!
```

V Special Issues

13 SANS and multiple scattering 73

14 TOF and BSS spectra 75

14.1 uni_ft Fourier transform of spectra

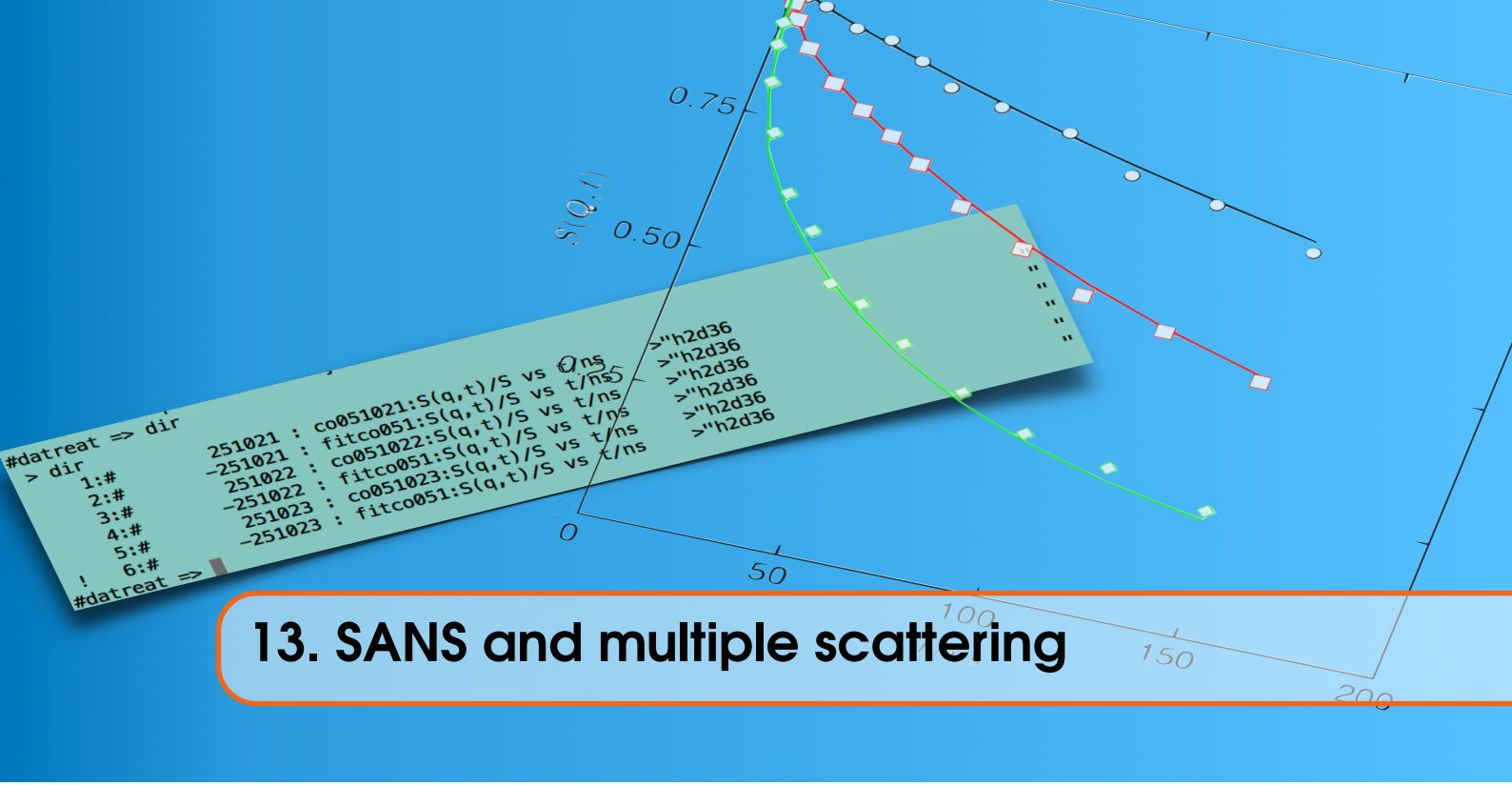
14.2 restran transfer of resolution function for $S(q,\omega)$ fitting

15 NSE instrumentation/design 77

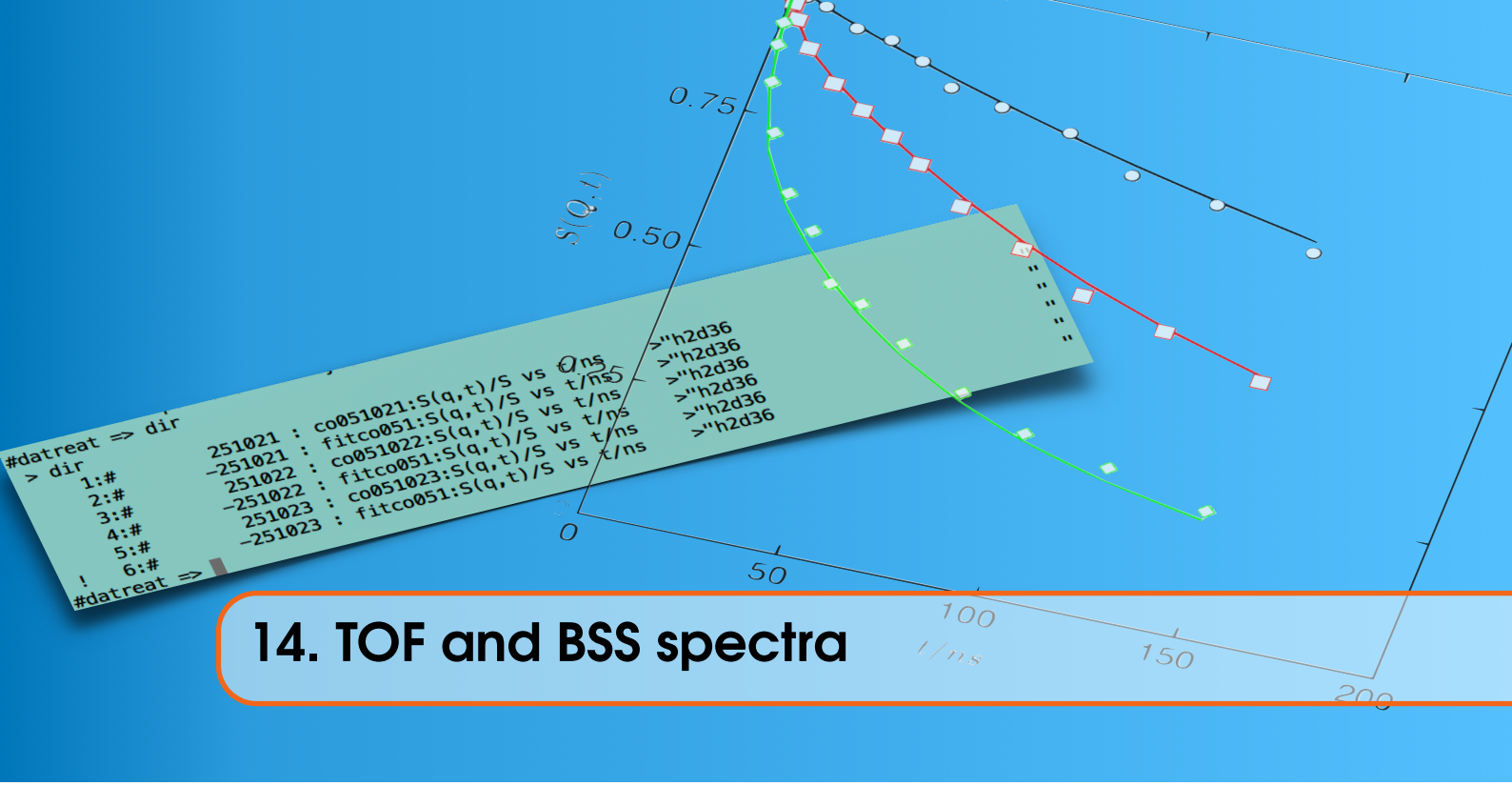
15.1 echocurv simulated phase scan

16 Development tools 81

16.1 noise add Gaussian noise and errors to a record



13. SANS and multiple scattering



14. TOF and BSS spectra

14.1 uni_ft Fourier transform of spectra

Fouriertransform of one selected spectrum from energy or frequency to the time domain. If one spectrum AND a resolution are selected (observe sequence of selection!) then also the 'deconvoluted' time function is produced. ■

The results (fourier transforms of data and resolution as well as the 'deconvoluted' time function is found in newly created records at the end of the list.
The algorithm is that from Reiner Zorn's uni_ft program.

The energy units given as x-axis are observed: **micro-eV, meV, GHz, omega (GHz)** (GHz means Giga rad/s). ■

First select a spectrum at second a resolution spectrum (if available/desired).

#datreat => **uni_ft** [resnorm] [reslim] *rlim* ↵ fourier transform of spectrum

(Further) options:
resnorm if present use resolution also to normalize the result (use when cold sample is resolution!)
reslim limit resolution deconvolution (=division) to values greater *rlim*.
rexpand factor to get finer t-spacing.
Uni_ft uses the *temp* parameter of the spectrum to perform a detailed balance based symmetrising of the spectrum and the energy gain side of the spectrum is used to (mirror) complete the energy loss side of the spectrum. The conventions used (Marshall-Lovesey) are $\omega > 0$ **is energy loss** of the neutron.

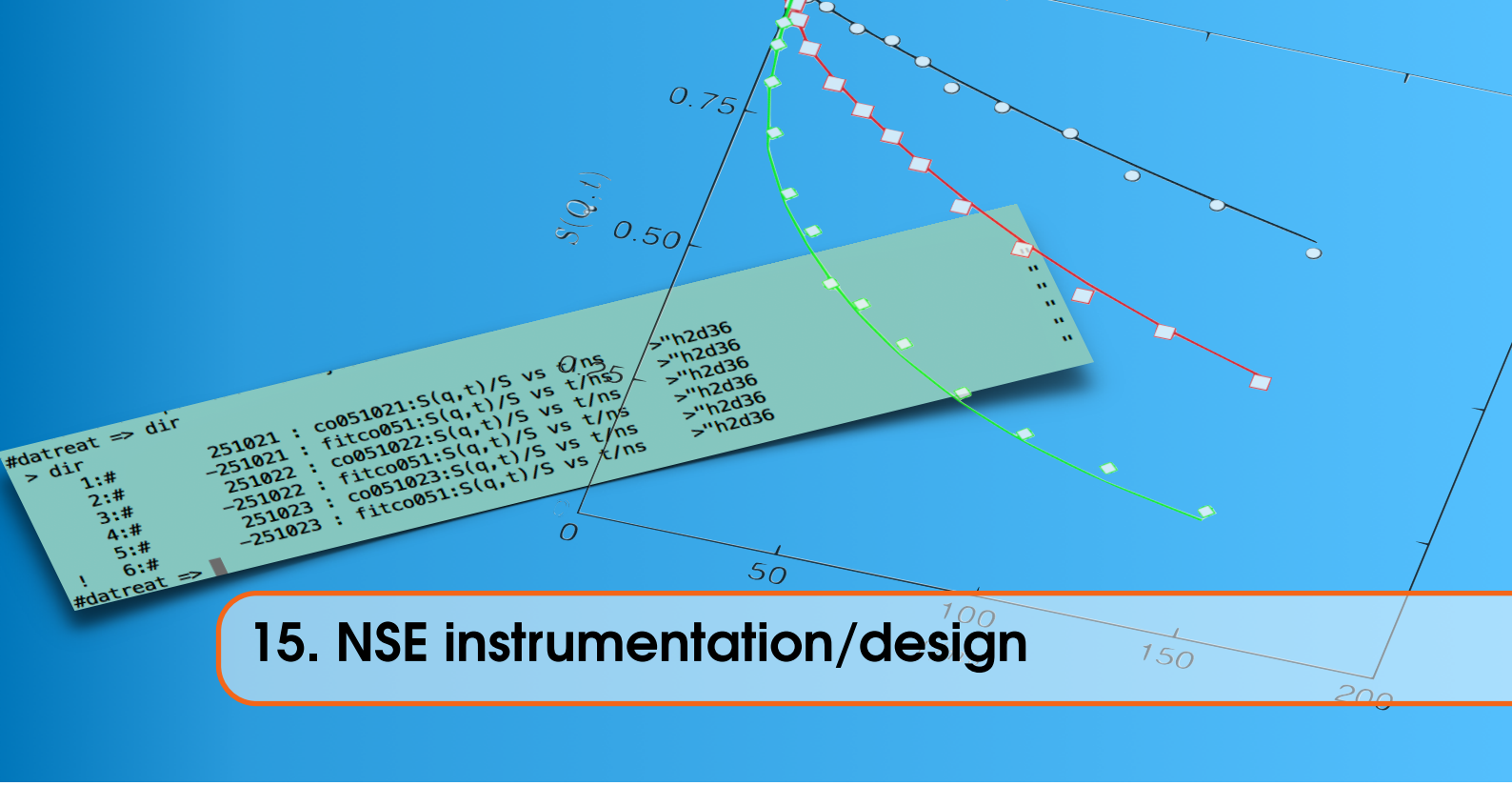
14.2 restran transfer of resolution function for $S(q,\omega)$ fitting

Fitting of $S(q,\omega)$ data from instruments like time-of-flight(TOF)- or backscattering(BSS)- instruments implies a resolution convolution. Starting from a model for the intermediate scattering function $S(q,t)$ this convolution can advantageously be combined with the necessary Fourier-transform of $S(q,t)$. In order to do so a representation of the reolution function by a sum (up to 8) Gaussians is needed. These may be obatined e.g. by fitting spectra of the cold sample with a number of Gaussians (example in makro TOFRES). The obtained parameters have to be transferred to the pertaining data records prior to fitting with a theory of the *sqt_omega* family (e.g. kohl, kohklfast, ..., sqt_om1), see 21. This transfer is supported by the **restran** command. ■

Once the resolution parameters *ga1inten, ga1width, ga1cente, ga2inten, ...* are introduced in the resolution data set by fitting they can/must be transferred to the corresponding data records prior to fitting with a model of the *sqt_omega* family. This may be done by the following command sequence:

```
#datreat =>  purge all ↵
#datreat =>  in resolution-file ↵
#datreat =>  in data-file ↵
#datreat =>  sel nq ↵
#datreat =>  set qnext (q) ↵
#datreat =>  sel all q (qnext) band 0.01 ↵
#datreat =>  restran ↵
```

restran extracts the resolution parameter from the record in the selection list and transfers them to all further selected records. Going through all records (nq) of the reolution file the reolution parameters or all q-values can be transferred.



15. NSE instrumentation/design

15.1 echocurv simulated phase scan

Creates phase scan intensity data that would be obtained from a NSE experiment using a Gaussian distributed wavelength band of width $\Delta\lambda$ from a scatterer with a simple relaxation with time constant τ. The field integral J, and the nominal wavelength λ_0 determine the associated Fourier time.
A subsequent **noise** command can be use to add simulated counting noise. ■

```
      if(comand.eq.'echocurv') then
!                    ---------
       if(found('help    ')) then
          write(6,*)'=================================================================
          write(6,*)'= echocurv j <j> ......
          write(6,*)'=    computation of simulated phase scan of NSE for a simple Lorenzia
          write(6,*)'=    ft by integration assuming gaussian distribution functions
          write(6,*)'=    parameters:
          write(6,*)'=      j   <val>        : field integral
          write(6,*)'=      dj  <val>        : asymmetry (phase0)
          write(6,*)'=      j0delta <val>    : inhomogeneity offset (external fields..)
          write(6,*)'=      ddj <val>        : phase scan step (field integral)
          write(6,*)'=      n <val>          : number of points in scan
          write(6,*)'=      cdelta <val>     : relative inhomogeneity
          write(6,*)'=      lambda0 <val>    : wavelength
          write(6,*)'=      dlambda <val>    : wavelength width
          write(6,*)'=      tau <val>        : relaxation time --> spectrum
          write(6,*)'=      errabs <> errel <> maxint  : integration parameters
          write(6,*)'=================================================================
          goto 2000
       endif
```

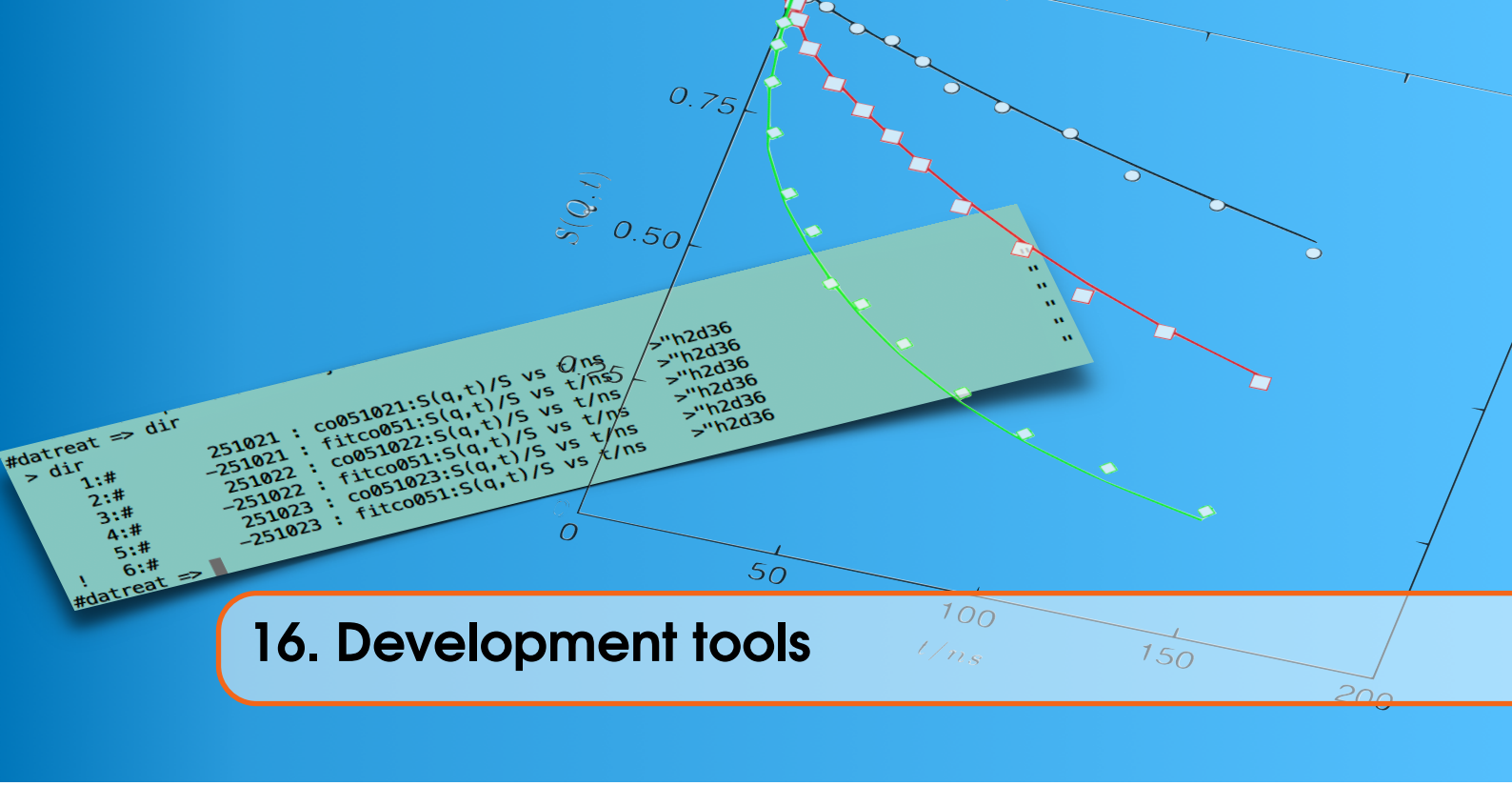
```
        alim(1) = alam0 - 3*dalam
        blim(1) = alam0 + 3*dalam

        alim(2) = -5.d0/tau
        blim(2) = -alim(2)



        n0 =     nech/2
        n0     = intval('n0      ',n0    ,inew)
                                                        !! aix.sp
        call     parset('n0      ',(real(n0)) ,ia1)

        nwert(ia1) = nech
        do i=1,nech
          j2echo = j1echo + (i-n0) * ddj + dj
          xwerte(i,ia1) = j2echo-j1echo
! --- imsl mehrdimensionale integration ---
          call  qand(f,2,alim,blim,erraec,errrec,maxint,              &
     &                result, errest )
          write(6,*)'result =',result,'     errest=',errest
          ywerte(i,ia1) = result
          yerror(i,ia1) = 0
        enddo
        xname(ia1) = 'dj/gm'
        yname(ia1) = 'counts'
        name(ia1)  = 'echo'
        coment(ia1)= 'echo'
        numor(ia1) = intval('numor   ',1,inew)


      function wlambda( alam )
!     ------------------------
! --- repraesentiert die wellenlaengenverteilung
       use wlntran
       implicit none
       real*4 arg, alam, wlambda
         arg     = ( (alam-alam0)/dalam )**2
         if(arg.lt.50.e0) then
           wlambda =  exp( -arg )
         else
           wlambda = 0.e0
         endif
      return
     END
```

```
      function sofqom( omega )
! --- repraesentiert die streufunktion, omega in s**-1
      use sqtran
      implicit none

      real*4 x, omega, sofqom
      x      = omega * tau
      sofqom = tau /(1.e0+x*x)
      return
     END


      function f(n,x)
! --- integrand ---
      use partran
      implicit none

      integer n
      real*4 x
      dimension x(n)
!
!     x(1) = lambda
!     x(2) = omega
!
! ---- larmorkonstante in rad/s/gauss
! ---- neutronenmasse durch h in 2/m/angstroem

      real*4 :: gamma=18303.33e0, xmh = 2.50607e-4, zpi = 6.283185307e0
      real*4 a,b,f,del, wlambda, sofqom
      a  = gamma*xmh*x(1)/(1.d0+xmh*x(1)*x(1)*x(2)*1d-10/zpi)
      b  = gamma*xmh*x(1)
      del= j0delta**2 + ( (j1echo+j2echo)*0.5e0*cdelta )**2
      f  = ( 1.d0 +  exp(-(a**2+b**2)*del*0.25e0 )*                    &
    &               cos( a*j1echo - b*j2echo )                         &
    &      ) * sofqom(x(2)) * wlambda(x(1))
      return
     END


 theory   echo
      amplitu     239.9750       1.00       1.4213
      average     242.4903       1.00       1.1182
      derotat     0.000000       0.00       0.0000
      lambda0     8.000000       0.00       0.0000
      lamfwhm    0.8316469      0.100      0.18950E-01
      ishift      0.000000       0.00       0.0000
      nturns      82.04076       1.00      0.52094E-01
      rcoil      0.3100000       0.00       0.0000
```

```
     lcoil      0.4700000E-01   0.00       0.0000
     acoil      -1.154000       0.00       0.0000
     bcoil       1.700000       0.00       0.0000
     a2coil     -3.600000       0.00       0.0000
theory   bkgr
     level       0.000000       0.00       0.0000
     slope       0.000000       0.00       0.0000
end
```

Note: The presently used 2D-integration may fail in cases with very low echo or other extremes!

TBD(datreat src):Replace 2 integragtion scuhre by a better method (adapint recursive)!



16. Development tools

16.1 noise add Gaussian noise and errors to a record

After scaling the y-values by a such that they (virtually) correspond to direct counting results a Gaussian random variable is added to them in order to simulate counting statistics noise. The associated errors are set accordingly. If the option [errors] is given, only the errors are set but no noise is added to the y-values. ■

All selected records are treated with the same a and options, finally the results are on new records, which are selected. The noise application is notified by a new parameter *noise_a*.

#datreat => **noise a** *a* **[errors] [iseed]** *seed* ↵

Note: The factor a must explicitly be specified and chosen such that it reconverts the y-data to (virtual) counting results, i.e. $\sigma(a\,y) = \sqrt{a\,y}$. The results may be scaled back to the original gauge by **fxy x=X y=Y/noise_a**.

VI Working Examples

17 Recommended workflows 85

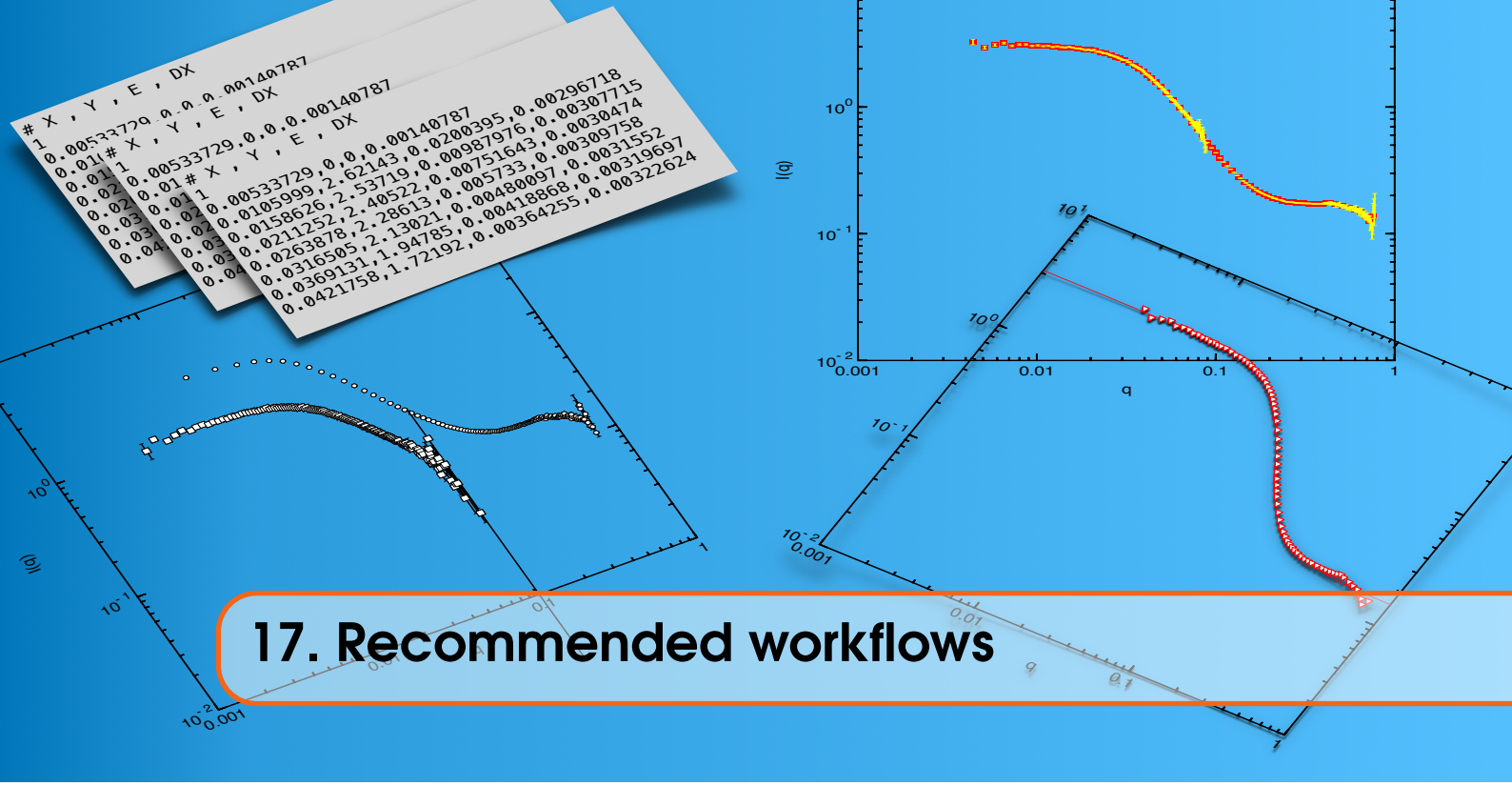
17.1 Open a terminal — from there datreat may be started

17.2 Directories — "where" to do the work

18 Workflows: examples 87

18.1 A simple start — basic usage example

18.2 Automation and makros — prepmak: a more complex makro example



17. Recommended workflows

17.1 Open a terminal from there datreat may be started

Datreat is command driven, in order to use it one has to open a *terminal* (or a *shell* within e.g. an editor like *emacs*). On a *Mac OSX* computer you may do this by typing (key)s:

> press: **(command)+(space)**
> type: **terminal** ↵

or if there is already a terminal open, click into that terminal and

> press: **(command)+n**

and a new terminal will open.
Within that terminal switch to the desired subdirectory:

> ...> **cd ~/Desktop/myworkingdirectory** ↵

Note 1: On a Mac *terminal* a protocol of a session can readily be stored by either pressing the keys **command+s** or using the *shell* pull down menu and click on *text exportieren*.

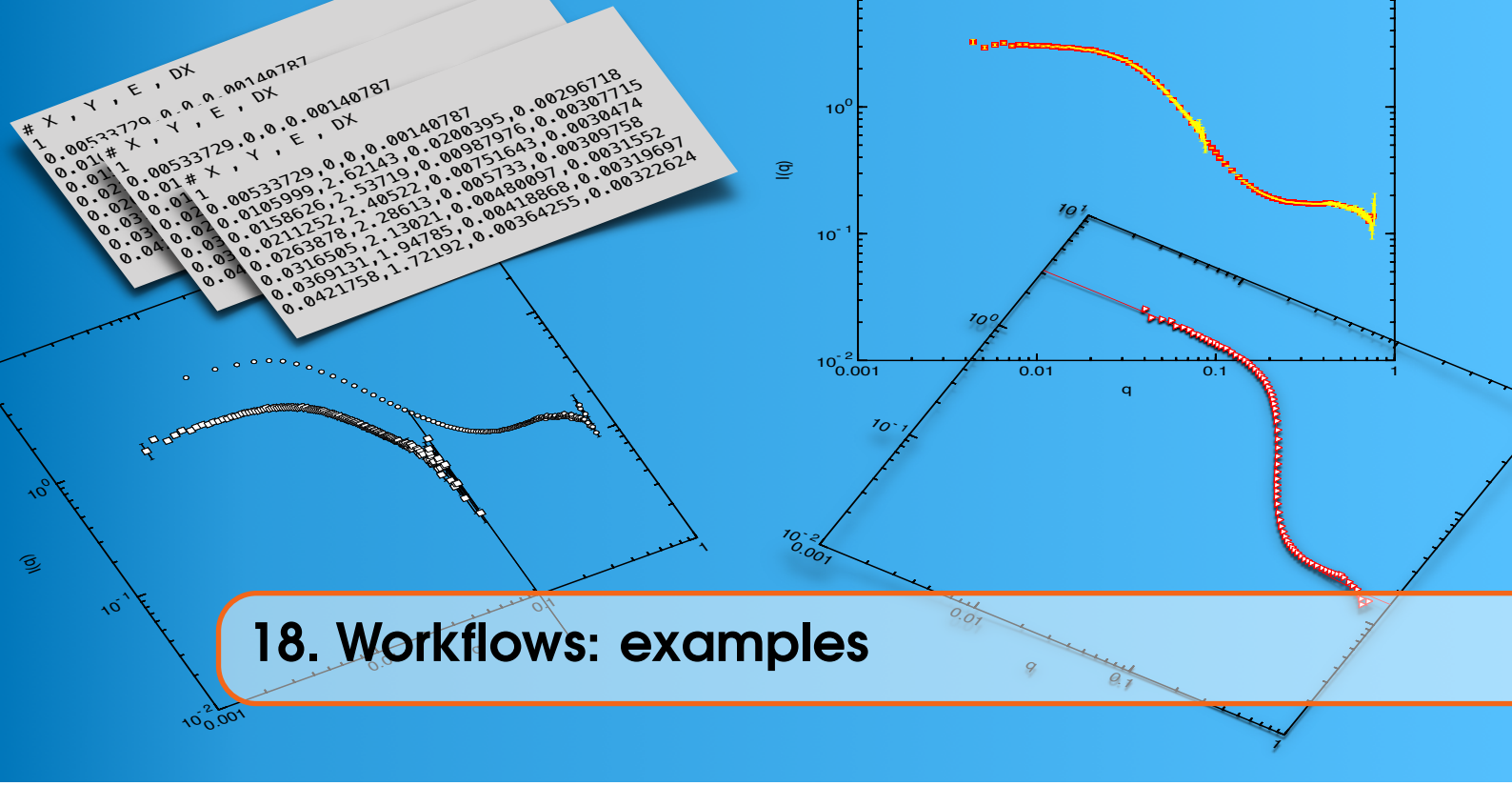
Note 2: Datreat keeps a history file: **.last_dtr_commands** in the working directory that allows to repeat previous command lines by pressing the **arrow**-keys **/up** or **down**. The history is *private* to the respective working directory and is kept between several invocations of datreat.

17.2 Directories "where" to do the work

It is recommended to select/create a working directory, switch to that directory and possibly copy the data to that directory. The latter is not mandatory. The start datreat from that directory.

```
...> cd ~/Desktop/myworkingdirectory ↵
...> datreat
```

Output, plots, command line-history and primary source of makros is THIS directory.



18. Workflows: examples

18.1 A simple start basic usage example

To make first steps and get an idea of the datreat philosophy the following is recommended:
First create a (subdirectory) to work for this problem, copy the data file(s) to it and start *datreat*

```
...> mkdir Myworkingdirectory ↵
...> cd Myworkingdirectory ↵
...> cp /Desktop/datreat/TRAINING/DENDRIMER_datreat/SB5q0.dtr ./ ↵   copy an example data file
...> datreat ↵   start datreat
```

now the datreat prompt appears and will accept datreat commands (it is further transparent for non ambiguous shell commands). First create a (subdirectory) to work for this problem, copy the data file(s) to it and start *datreat*

```
#datreat => in SB5q0.dtr ↵   read the data into the record memory
#datreat => sel 1 ↵   select the 1st record in memory
#datreat => dir ↵   check by listing the contents of record memory
#datreat => aplot errors ↵   autoplot (makro) with errors
#datreat => ac strexpo ↵   activate theory strexpo
#datreat => cth ↵   open theory specification for parameter editing
#datreat => thc ↵   first just calculate the theory to check start parameters etc.
#datreat => plot ymin 0 ↵   using plot to examine
#datreat => fit ↵   now fit
#datreat => plot ymin 0 ↵   check plot now with ymin set to 0
#datreat => msave myfirstfit ↵   save data fit and model
#datreat => ...
#datreat => quit ↵   exit datreat
```

To resume work you may simply use:

```
...>  datreat ↵     start datreat and restores last state
#datreat =>  ...
#datreat =>  quit ↵     exit datreat
```

```
...>  datreat 0 ↵     start datreat without restoring
#datreat =>  inplus myfirstfit ↵     read save data and fit into
#datreat =>  sel 1 ↵     select the 1st record in memory
#datreat =>  dir ↵     check by listing the contents of record memory
#datreat =>  ...
#datreat =>  quit ↵     exit datreat
```

18.2 Automation and makros prepmak: a more complex makro example

As example we examine the *makro* **prepmak**, which is used in the TRAINING/DENDRIMER context to prepare consolidated data for datreat from a certain set of raw data. To invoke it:

```
#datreat =>  prepmak 0.1 ↵     makro call preparing, scaling and averaging SANS data
```

The makro, Preliminaries:

```
1 makro _x_      first line telling it is a makro with one argument _x_
2
3 set xc (_x_)     store argument value to variable xc
4 ?? (xc)     show value of xc (for check only)
```

next define some preset variables, if change is needed it can be done here.

```
5 set clp1 8     set var clp1 to 8
6 set clp2 20     set var clp2 to 20
7
8 set xfit1 0.002     set xfit1 to 0.002
9 set xfit2 0.25     set xfit2 to 0.25 10 vars?     display all variables for (check only)
```

here we set variables for clipping *clp1, clp2* and fit range *xfit1,2*.

```
11 ! run the collecting and transform of data (shell script)
12 sys cotxtdtr     run shell script on the operating system level
```

This is an example how data preparations beyond the scope of datreat commands may be included. **cotxtdtr** is the name of a specific shell script that (in this example) contains calls of data-conversion routines to prepare datreat compatible data formats for the here treated data. At the end the *datreat input file* **eqsans_dendrimer** is written. In the following course of the makros it will be read using the datreat command **in**.
More details on *cotxtdtr* will be given below.
But for now we proceed with the proper makro.

```
14 purge all      clear the record memory completely
15 in eqsans_dendrimer      read the data
16 dir      list the contents of the record memory
```

Now SANS data for different detector distances with different scales and for different samples as was converted and written to the datreat output file *eqsans_dendrimer* (and augmented by auxiliary information) by running *cotxtdtr*, after reading using the command **in**, occupies a number of records in the memory.

```
17 set naly 0      create and initialise a user variable naly
18 set fak 0      " fak = factor
18 set ofs 0      " ofs = offset
19 set fakq 0      " fakq
21 set ofsq 0      " ofsq
22
23 set irun -1      " irun = run index variable
24
```

Here a number of user defined variables (chosen names) are initialised. They serve to determine averages and variances for the linear scaling parameters *fak, ofs* used to align data from the two detector distance. Finally a counter/pointer to the records to be processed (*irun*) is set.

Now the first pair of records is processed:

```
26 ! treat first pair of detector distances 27 set irun (irun+2)      increment record counter
28 sel (irun)      select first record of this pair
29 clip from 1 to (clp1)      remove the first clp points (beam stop)[1]
30 sel (irun+1)      select the other record of this pair
31 clip from (nv(irun+1)-clp2) to nv(irun+1)      remove the last clp2 points[2]
32 sel (irun) (irun+1)      select both (now clipped) records
33 set nn (num(isel(1)))      remember the numor of the first selected record
34 aligny      use overlap between selected records for scaling
35 set naly (naly+1)      count the number of scaling actions
36 set fak (fak+aligny_s)      add the present factor aligny_s to fak
37 set ofs (ofs+aligny_o)      same for offset
38 set fakq (fakq+aligny_s2̂)      ..and collect squares for the variances
39 set ofsq (ofsq+aligny_o2̂)      "
40 average xcatch (xc)      finally average and combine the two scaled records
41 putpar avof (nn)      put information of the original numor to the average
```

[1] the variable *clp1* was set in line 6 of the macro, note the *clip* acts on place.
[2] the variable *clp2* was set in line 6 of the macro, note the *clip* acts on place.

This now is repeated for all present pairs of detector distance data. Thus the sequence an number is *customized* to the available data for this measurement.

```
43 ! sel 3 4
44 set irun (irun+2)      increment record counter
... ...
```

Then we do final adjustments to the parameters etc. and store the consolidated results.

```
218 sel all avof 2000 band 4000      now all results that have the parameter avof are selected
219 rename xaxis  $$\rm\bf~q\,~\slash~\,\AA{^-1}      tex-style "pretty" x-axis
220 rename yaxis I(q)/[a.u.]      y-axis name
221
222 p0 xmin 0.001 xmax 1 ymin 0.01 ymax 10 log_x log_y colo 0 1 2 3 4 5 6 7 8 9 errors
prepare plot
223
224 set fak (fak/naly)      normalise averages ...
225 set ofs (ofs/naly)      "
226 set fakq (sqrt(fakq/naly-fak*fak))      ... and compute variances
227 set ofsq (sqrt(ofsq/naly-ofs*ofs))      "
228 putpar fakq (fakq)      ..and put the results to the parameters of the records
229 putpar fak (fak)      "
230 putpar ofsq (ofsq)      "
231 putpar ofs (ofs)      "
232 msave allsansav      store (all) final result records to file allsansav
233 purge all      clear record memory
234 in allsansav      read the final records in a cleared memory, i.e from position 1
```

So now the data are prepared for further analysis, e.g. **fitting**. So first we set up a model, here: background (**bkgr**) + extended Debye model with A2 and discrete summation with fractal chain expansion (**ndebye**).

```
242 dac      clear any previous model definitions
243 ac bkgr      add one background model instance
243 ac ndebye      add one Debye model instance
244
245 chgthpar ndebye 1 ampli par 0.01 scale 0.01      set start parameters ...
246 chgthpar ndebye 1 n par 333 scale 0.00 ! do not fit !
247 chgthpar ndebye 1 l par 10 scale 0.100
248 chgthpar ndebye 1 nu par 0.4 scale 0.100
249 chgthpar ndebye 1 chi par 1e-3 scale 1e-3
250 chgthpar ndebye 1 withsq par 1 scale 0.0
251
252 al      finally display the model setting (and write lastth)
```

Now we are ready for some fitting:

```
257 ! fit and determine background 258 set nn (nbuf)      put number of datasets (records) to nn
259 set i 1      initialize loop counter
260 :j1      label (loop start)
261 sel (i)      select i-th record 262 rename xaxis  $$\rm\bf~q\,~\slash~\,\AA{^-1}
tex-style "pretty" x-axis
263 rename yaxis I(q)/[a.u.]      y-axis name
264 fit x1 (xfit1) x2 (xfit2)      do the fitting for record i
265 thc n -200 x1 0.001 x2 0.5      compute fit result over the extended q-range
266 plot # 1      ...and plot this
267 fun y+ (-level)[1]      subtract the fitted background level
```

```
268 putpar bgrsub (level)      and add the (subtracted) level to the parameters
267 plot      plot the background subtracted version
268 set i (i+1)      increment the (intended) loop index
269 if (i) <= (nn) then goto :j1      jump to loop start if i ≤ nn
```

...the loop serve mainly as illustration here, usually fitting is done more interactively.

18.2.1 The shell script cotxtdtr

In the above example makro we use the shell script *cotxtdtr* called by the command **sys cotxtdtr**. The name is freely chosen (here meaning **co**nvert**txt**to**dtr**). The explicit use of **sys** ensures that the name is not mixed up with any *makro* or *genuine datreat command*.

cotxtdtr

```
 1: gfortran rp.f90
 2: a.out sHB2_1o3m.txt   0.020  1.03  3021 >   eqsans_denrimer
 3: a.out sHB2_4o5m.txt   0.020  4.05  3022 >>  eqsans_denrimer
 4: a.out sHB3p5_1o3m.txt 0.035  1.03  3031 >>  eqsans_denrimer
 5: a.out sHB3p5_4o5m.txt 0.035  4.05  3032 >>  eqsans_denrimer
 6: a.out sHB5_1o3m.txt   0.050  1.03  3051 >>  eqsans_denrimer
 7: a.out sHB5_4o5m.txt   0.050  4.05  3052 >>  eqsans_denrimer
 8: a.out sL2_1o3m.txt    0.020  1.03  1021 >>  eqsans_denrimer
 9: a.out sL2_4o5m.txt    0.020  4.05  1022 >>  eqsans_denrimer
10: a.out sL3p5_1o3m.txt  0.035  1.03  1031 >>  eqsans_denrimer
11: a.out sL3p5_4o5m.txt  0.035  4.05  1032 >>  eqsans_denrimer
12: a.out sL5_1o3m.txt    0.050  1.03  1051 >>  eqsans_denrimer
13: a.out sL5_4o5m.txt    0.050  4.05  1052 >>  eqsans_denrimer
14: a.out sSB2_1o3m.txt   0.020  1.03  2021 >>  eqsans_denrimer
15: a.out sSB2_4o5m.txt   0.020  4.05  2022 >>  eqsans_denrimer
16: a.out sSB3p5_1o3m.txt 0.030  1.03  2031 >>  eqsans_denrimer
17: a.out sSB3p5_4o5m.txt 0.030  4.05  2032 >>  eqsans_denrimer
18: a.out sSB5_1o3m.txt   0.050  1.03  2051 >>  eqsans_denrimer
19: a.out sSB5_4o5m.txt   0.050  4.05  2052 >>  eqsans_denrimer
20: cat L1.dat    >>    eqsans_denrimer
21: cat L10.dat   >>    eqsans_denrimer
22: # cat S1.dat    >>    eqsans_denrimer
23: # cat S10.dat   >>    eqsans_denrimer
24: cat HB1.dat   >>    eqsans_denrimer
25: cat HB10.dat  >>    eqsans_denrimer
```

The intention of this shell-script is to convert raw data that come in a different form (files **.txt*) to datreat format and at this occasion include further missing metadata. The results in datreat format are all collected in one single file: *eqsans_denrimer*. The method chosen to do this here is to use a custom written small fortran program **rp.f90** to perform the task. Besides the raw file names it also accepts further metadata numbers enabling to put q, detector distance and a given id-number (numor). The program **rp.f90** is freshly compiled in line 1. The compiler call in this form creates an executable **a.out**, which in the following lines is called to do the conversion. The UNIX redirection operators > (write output to file) and >> (append output to file) are used to push the converted data to the target file *eqsans_denrimer*. Finally in lines 20: - 25: the UNIX *cat* command is used to also append some data that are already present in datreat format.

rp.f90

```
! read SANS DATA:
! use gfortran rp.f90
! a.out filename <conc> <collen> <numor> > datfile
program rp
implicit none
```

```
double precision :: q, iq, err, dx, conc, l2
integer          :: ios
integer          :: numor
CHARACTER(len=255) :: cmd, lin, fil

     CALL get_command_argument(1,fil)  ! the first argument (to a.out) is the filename
     open(10, file=trim (fil))         ! open the file
     CALL get_command_argument(2,cmd)  ! the next argument holds the concentration
     read(cmd,*) conc                  ! convert the concentration from string to number
     CALL get_command_argument(3,cmd)  ! the next argument is the detectro distance l2
     read(cmd,*) l2                    ! convert it to a number
     CALL get_command_argument(4,cmd)  ! the last (4-th) argument is the numot
     read(cmd,*) numor                 ! convert it to a number
     write(*,'(a)')trim(fil)                                  ! write filename as comment to *=stdout (> eqsans_denrimer)
     write(*,'(a,a,i9)')fil(1:8),'  "I(q)"  vs q  ', numor  ! write the "versus" line for datreat
     write(*,'(a,f12.6)') "conc ", conc                       ! put parameter conc
     write(*,'(a,f12.6)') "l2 ", l2                           ! put parameter l2
     write(*,*)"values"                                       ! optional values indicator
     do                                                       ! loop over lines in txt-file
       read(10,'(a)',iostat=ios)lin                             ! read a line
       if(ios .ne.0 ) exit                                      ! exit=finish if no longer successful
       read(lin,*,iostat=ios) q, iq, err, dx                    ! convert to numbers
       if(ios .ne. 0) cycle                                     ! skip if conversion did not succeed
       if(iq  == 0d0) cycle                                     ! skip if inensity (empty data)
       write(*,'(3f14.7)') q, iq, err                           ! write q, intensity and error to stdout
     enddo                                                    ! end of loop
     write(*,'(a)')"#nxt"                                     ! write a end group label
     close(10)
end program rp
```

Used to read data in the form:

sHB2_1o3m.txt

```
# X , Y , E , DX
1
0.00533729,0,0,0.00140787
0.0105999,2.62143,0.0200395,0.00296718
0.0158626,2.53719,0.00987976,0.00307715
0.0211252,2.40522,0.00751643,0.0030474
0.0263878,2.28613,0.005733,0.00309758
0.0316505,2.13021,0.00480097,0.0031552
...
...
...
0.763157,0.168901,0.0193701,0.0215116
0.76842,0.142007,0.0259377,0.0216196
0.773683,0.17035,0.0376949,0.0108332
0.778945,0,0,0
0.784208,0,0,0
0.789471,0,0,0
```

VII Theories

19 strexpo generic streched exponential 95

20 gauss, gauss2, gaussft Gaussians . . 97

21 Spectral functions in ω General aspects for resolution broadened spectra. 99

21.1 Determination of Gaussian resolution parameters

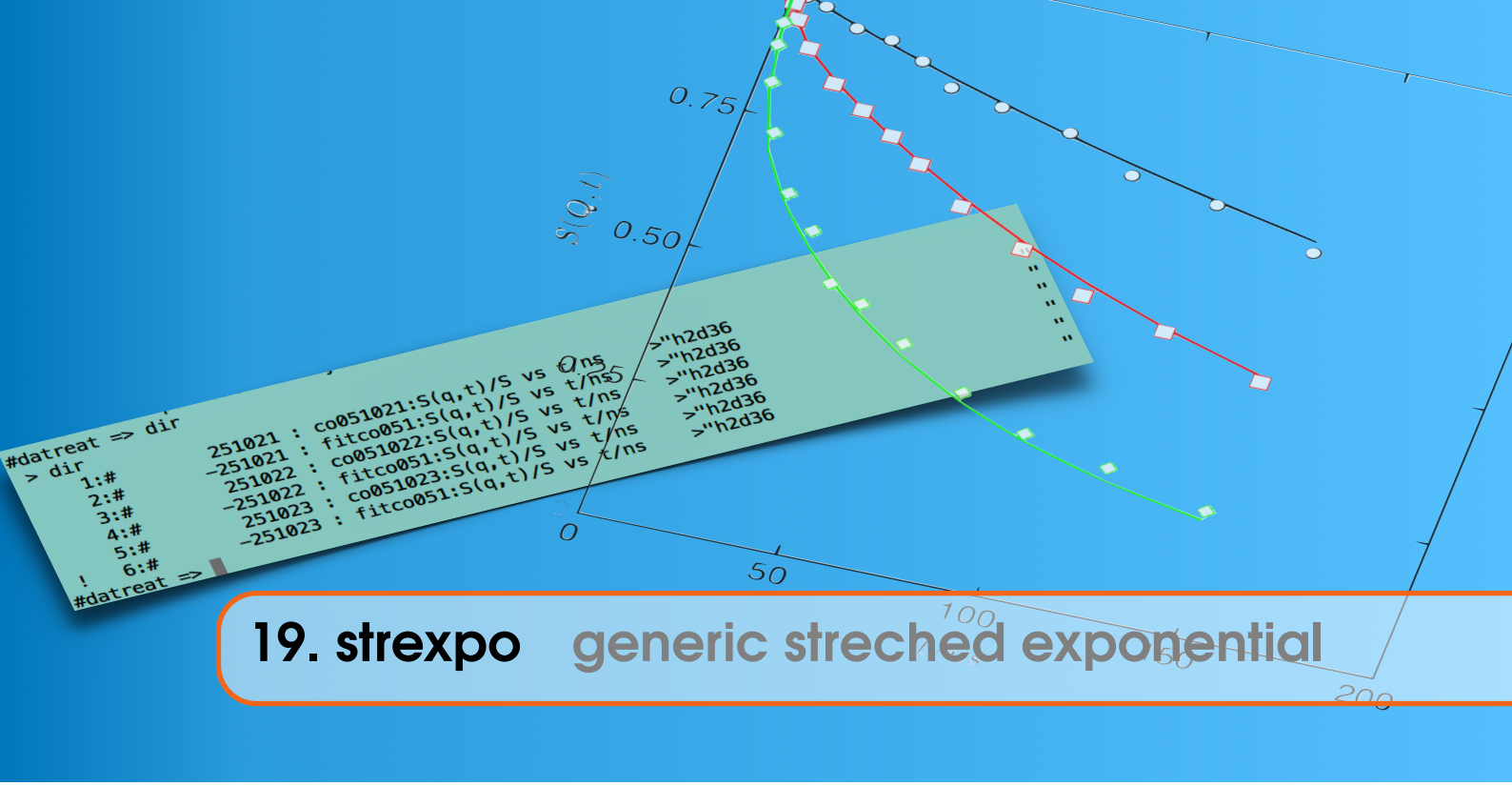
22 kohlfast Spectrum of Kohlrausch fktn. 109

23 sqt_om1 S(q,t) for system with labelled crosslinkers 111

24 nrouse Rouse single chain structure factor 113

25 ring2b Ring single chain structure factor (Goossen,PRL 2014) 115

25.1 TO BE MODIFIED...



19. strexpo generic streched exponential

$$f_{th}(x = t; \{q\}) = A \exp\left(-\left[\frac{|t|}{\tau_0 q^n}\right]^\beta\right) \tag{19.1}$$

Direct theory parameters ((fit)parameters in model definition):

Parameter	meaning	pretreat	fit
amplitu	A, standard pre factor	–	yes
tau	τ_0 time constant	$\lvert\cdots\rvert$	yes
beta	β stretching exponent	$\lvert\cdots\rvert$	yes
qexp	n exponent of q-dependence	–	yes

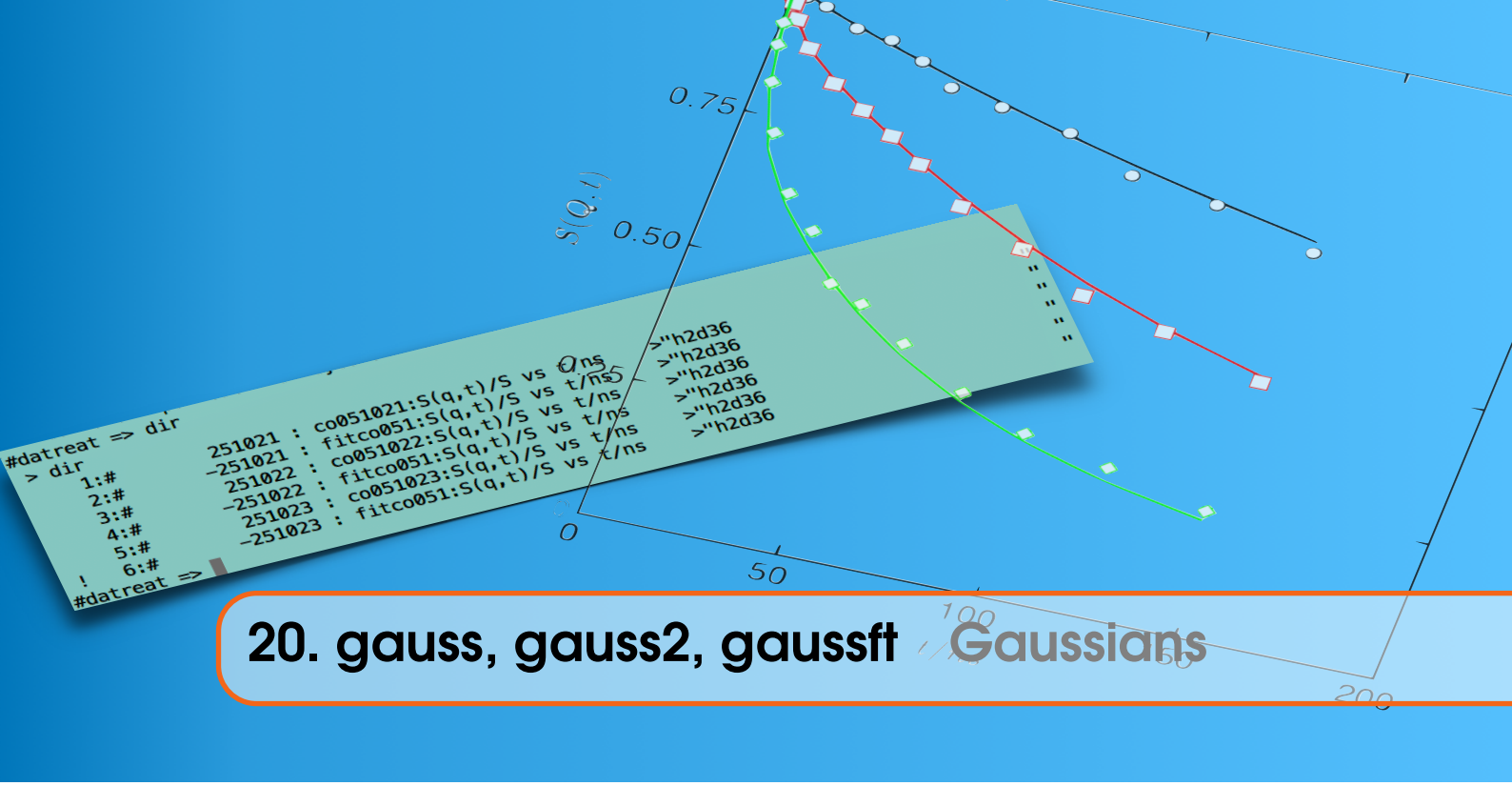
Used record parameters:

q	q, q-value for this record

Parameters pushed to records:

tauave0	$\tau_0\Gamma(1\beta)/\beta$, average tau
tauave	$\tau\Gamma(1\beta)/\beta$, average tau(q)
deff	$1/[q^2\tau\Gamma(1\beta)/\beta]$, effective diffusion

Note: Fix $n = 0$ *(qexp)* to get a simple, q-independent stretched exponential!



20. gauss, gauss2, gaussft Gaussians

gauss : plain Gaussian

$$f_{th}(x) = A \exp\left(-\left[\frac{x - x_0}{w}\right]^2\right) \tag{20.1}$$

Direct theory parameters ((fit)parameters in model definition):

Parameter	meaning	pretreat	fit
intensit	A, standard pre factor	–	yes
width	$w = \sqrt{2}\sigma$, width	–	yes
center	x_0, center	–	yes

Used record parameters:

-none-

Parameters pushed to records:

-none-

Note: this function is intended to accurately model (resolution) contribution with width in the range or smaller than the bin-width of histogramed data.

gauss2 : Gaussian integrated over (one fixed) channel-width

$$f_{th}(x) = A(\sqrt{\pi/4}w/\Delta x))\left(\mathrm{erf}\left[\frac{2(x-x_0)+\Delta x}{2\Delta x}\right] - \mathrm{erf}\left[\frac{2(x-x_0)-\Delta x}{2\Delta x}\right]\right) \tag{20.2}$$

Direct theory parameters ((fit)parameters in model definition):

Parameter	meaning	pretreat	fit
intensit	A, standard pre factor	–	yes
width	$w = \sqrt{2}\sigma$, width	–	yes
center	x_0, center	–	yes
xbinwid	Δ_x channel width	–	no

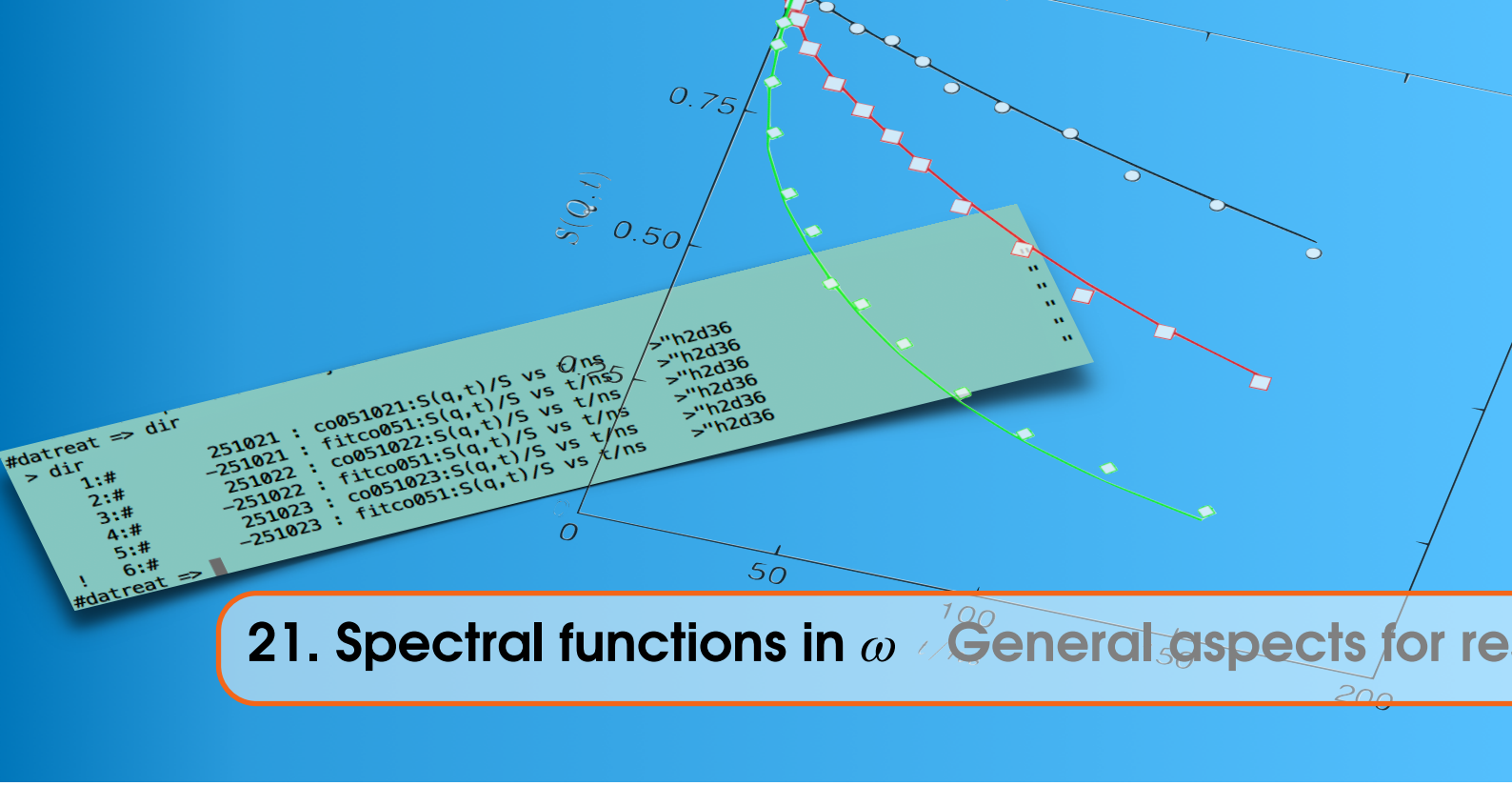
Used record parameters:

-none-

Parameters pushed to records:

-none-

gaussft : Gaussian for resolution model ?OBSOLETE?



21. Spectral functions in ω General aspects for re

Resolution convolution is performed by multiplication of the intermediate scattering function with the Fourier transform of the resolution function. This Fourier transform is obtained by using a representation of the resolution function in terms of Gaussians

$$\mathcal{R}(Q,\omega) = \sum_{i=1}^{N} a_i \, e^{-(\omega-\omega_i)^2/w_i^2} \tag{21.1}$$

with the Fourier-representation

$$\mathcal{R}(Q,\omega) = \sum_{i=1}^{N} \int \frac{a_i \, e^{-(w_i t)^2/4}}{2\pi^{3/2} \, w_i} \cos([\omega-\omega_i]\,t) d\omega \tag{21.2}$$

typically the components of the Gaussian representations have a rather limited extension in the time domain such that the numerical Fouriertransform of the product with the intermediate scattering function can be restricted to a corresponding maximum time, thus yielding an efficient computation of the expected experimentally observed spectrum

$$I(Q,\omega) = \sum_{i=1}^{N} \int_0^{t_{\max}=W_t/w_i} \frac{a_i \, e^{-(w_i t)^2/4}}{2\pi^{3/2} \, w_i} \cos([\omega-\omega_i]\,t) \, S(Q,t) \, dt \tag{21.3}$$

where the coefficient W_t used for the upper time limit is selected such that the integral contributions beyond that point are sufficiently reduced, here a value of 5 is used sich that the integrand is reduced with a factor of less than 2×10^{-3} due to the resolution factor (if reqired the factor may easily be increased without causing too much extra computational effort). The procedure as outline above captures the salient features use for fitting the results, however, to cope with effects pertaining to the finite width of one energy/frequency bin $\Delta\omega$, which will play a role if broadening effects in the range of $\Delta\omega$ are to be considered, the final form of the method takes account for these effects by the following modifications. 1. the resolution is fitted with the integral of the Gaussian over the

channel width

$$\sum_{i=1}^{N} a_i \{\sqrt{\pi/4}\, w_i/\Delta\omega\} (\mathrm{erf}([2\omega+\Delta\omega]/2w_i) - \mathrm{erf}([2\omega-\Delta\omega]/2w_i)) \tag{21.4}$$

.

2. when computing the estimation of the measured intensity $I(Q,\omega)$ the same integration over a channel width is performed, which effectively is achieved by replacing Eq. 21.5 by

$$I(Q,\omega) = \sum_{i=1}^{N} \int_0^{t_{\max}=W_t/w_i} \frac{a_i\, e^{-(w_i t)^2/4}}{2\pi^{3/2}\, w_i} \frac{\sin(\{-[\omega-\omega_i]+\Delta\omega/2\}\, t) + \sin(\}[\omega-\omega_i]+\Delta\omega/2\}\, t)}{t\,\Delta\omega} S(Q,t)\, dt \tag{21.5}$$

This ensures proper treatment of spectral features with variations within the bin width of the histogramm of count results. For virtually any conceivable model for the intermediate scattering function $S(Q,t)$ the numerical integration using an adaptive method is reliable and fast. Due to the inclusion of the resolution convolution any cutoff effects are avoided as well as the typical artefacts of replacing the integration by an FFT sum.

> The resolution parameters a_i, w_i, and ω_i for $i = 1 \cdots N$ ($N \leq 8$) have to be determined from a suitable resolution experiment (e.g. the vvery cold sample) and then set a parameters in the data records in order to be available the spectrum fit-theories. ■

Parameter	meaning
ga1inten	a_1, amplitude of first Gaussina resolution component
ga1width	w_1, width of first Gaussina resolution component
ga1center	ω_1, center of first Gaussina resolution component
...	...
...	...
ga8inten	a_1, amplitude of first Gaussina resolution component
ga8width	w_1, width of first Gaussina resolution component
ga8center	ω_1, center of first Gaussina resolution component

Note: One may specifiy less than 8 gaussian components, however, 8 is the maximum.

> Use the command **restran** to transfer fitted resolution parameters from the resolution files to the pertaining data! (replaces makro *putres*. ■

21.1 Determination of Gaussian resolution parameters

TBD(datreat src):device more automatic procedure!

Examples for the resolution determinetion and data preparation are found in the makros: **makres..**, **getres..** and **putres..**.

makresm8

```
makro _q_

! Makro for bulk PEP TOFTOF
```

```
purge all

set n_container +nbuf
 in EmptyCan135k.i96.dat

c in EmptyCan45k.i96.dat

set n_res +nbuf
 in b10k135.i96.dat

c in b10k45.i96.dat

sel +n_res+1

sel next q (_q_) band 0.001

set n_data_s (sel)

c goto :ex

c
c subtract empty container from resolution
c

sel 0

sel next q (_q_)
sel next q (_q_) add
c goto :ex
set result (sel)

if +nsel = 2 then goto :ok0

echo "NO DATA WITH THAT Q FOUND"
goto :ex

:ok0

sel 0
sel next q (_q_)
sel next q (_q_)
sel add (result)

dsl

arit f1 1 f2 -1  to 101235

c msave file1

xformel ;(xx*1000*1.5193)
yformel ;(yy)
funfun immediate op
rename xaxis GHz


c plot xmin -1000 xmax 1000 ymin 0.001 ymax 5000 log_y

c goto :ex

c
c do the 4G + bgr resolution fitting
c
c
c Sichern des letzten Theoriesettings auf lastth_x

c goto :ex
c
c
c
c dummy action to set the range of peak detection as [x1,x2]
c dac
 ac bkgr
 chgthpar bkgr 1 level  par 1  scale 1
 fit x1 -1000 x2 1000 go
c
c
c
c
c In der (neuen) Datreat Version sind maxy, centerx, widthx
c "Extract" variablen, die das Maximum und den "Schwerpunkt" und Weite (in [x1,x2])
c  der Daten des selktierten Records enthalten. Damit kann man gute Startwerte fuer den
c  Gaussian Fit finden.
c
set ym (maxy)13
set xc -centerx
set w1 +widthx
set w2 +w1*1.5
c
c
c leeren der Theorieliste und aktivierenm von 2 Gaussians
c
dac
ac gauss
ac gauss
```

```
c
c Inline Setzten von Parametern (interaktiv mit cth).
c

chgthpar gauss 1 intensit  par 1*ym        scale 1
chgthpar gauss 1 width     par +w1         scale 0
chgthpar gauss 1 center    par 0-xc        scale 0

chgthpar gauss 2 intensit  par 0.2*ym     scale 1
chgthpar gauss 2 width     par +w2        scale 0
chgthpar gauss 2 center    par 0-xc       scale 0


c
c Fit in einem eingeschraenkten Bereich um den Peak
c

fit x1 -5*w1 x2 5*w1 go
set  ssqfit1 +ssq

c
c Jetzt folgt im Wechselspiel von Parameter setzen/freigeben und
c Hinzufuegen weitere Gaussians oder background definitionen der
c Aufbau und (hoffentlich) robuste Fit des Aufloesungsmodells
c


chgthpar gauss 1 width     par +w1      scale 0
chgthpar gauss 2 width     par +w2      scale 0




fit
set  ssqfit2 +ssq


chgthpar gauss 1 center    par 0-xc        scale 0
chgthpar gauss 2 center    par 0-xc        scale 0



fit
set  ssqfit3 +ssq



c ac gauss
c chgthpar gauss 3 intensit  par 0.1*ym           scale 1
c chgthpar gauss 3 width     par (ga1width)/2     scale 0
c chgthpar gauss 3 center    par (ga1center)      scale 0


fit
set  ssqfit4 +ssq

c chgthpar gauss 3 width     par (ga1width)/2     scale 1


fit
set  ssqfit5 +ssq

c if +plotyes = 0 then plot

chgthpar gauss 1 intensit  par (ga1inten)        scale 1
chgthpar gauss 1 width     par (ga1width)        scale 0
chgthpar gauss 1 center    par (ga1cente)        scale 0

chgthpar gauss 2 intensit  par (ga2inten)        scale 1
chgthpar gauss 2 width     par (ga2width)        scale 0
chgthpar gauss 2 center    par (ga2cente)        scale 0

c chgthpar gauss 3 intensit  par (ga3inten)        scale 1
c chgthpar gauss 3 width     par (ga3width)        scale 0
c chgthpar gauss 3 center    par (ga3cente)        scale 0

c ac gauss
c chgthpar gauss 4 intensit  par (ga1inten)*0      scale 1
c chgthpar gauss 4 width     par (ga1width)*2.5    scale 0
c chgthpar gauss 4 center    par (ga1cente)        scale 0



 ac bkgr
chgthpar bkgr 1 level      par 1                  scale 1
chgthpar bkgr 1 slope      par 0                  scale 1


fit x1 -1000 x2 1000 go
set  ssqfit6 +ssq

c chgthpar gauss 4 intensit  par (ga1inten)*0.05   scale 1
c chgthpar gauss 4 width     par (ga1width)*2.5    scale 0
c chgthpar gauss 4 center    par (ga1cente)        scale 0
```

```
fit
set  ssqfit7 +ssq

c chgthpar gauss 4 width     par (ga1width)*2.5    scale 1

fit
set  ssqfit8 +ssq

c chgthpar gauss 4 center    par (ga1cente)        scale 1

fit
set  ssqfit9 +ssq

c chgthpar gauss 4 intensit  par (ga4inten)        scale 1
c chgthpar gauss 4 width     par (ga4width)        scale 0
c chgthpar gauss 4 center    par (ga4cente)        scale 0

fit
set  ssqfit10 +ssq


chgthpar gauss 1 intensit  par (ga1inten)        scale 1
chgthpar gauss 1 width     par (ga1width)        scale 0
chgthpar gauss 1 center    par (ga1cente)        scale 0

chgthpar gauss 2 intensit  par (ga2inten)        scale 1
chgthpar gauss 2 width     par (ga2width)        scale 0
chgthpar gauss 2 center    par (ga2cente)        scale 0

c chgthpar gauss 3 intensit  par (ga3inten)        scale 1
c chgthpar gauss 3 width     par (ga3width)        scale 0
c chgthpar gauss 3 center    par (ga3cente)        scale 0

c chgthpar gauss 4 intensit  par (ga4inten)        scale 1
c chgthpar gauss 4 width     par (ga4width)        scale 0
c chgthpar gauss 4 center    par (ga4cente)        scale 0



chgthpar bkgr 1 level                        scale 0
chgthpar bkgr 1 slope                        scale 0

fit x1 -5*w1 x2 5*w1 go
set  ssqfit11 +ssq


chgthpar gauss 1 intensit  par (ga1inten)        scale 1
chgthpar gauss 1 width     par (ga1width)        scale 0
chgthpar gauss 1 center    par (ga1cente)        scale 0

chgthpar gauss 2 intensit  par (ga2inten)        scale 1
chgthpar gauss 2 width     par (ga2width)        scale 0
chgthpar gauss 2 center    par (ga2cente)        scale 0

c chgthpar gauss 3 intensit  par (ga3inten)        scale 1
c chgthpar gauss 3 width     par (ga3width)        scale 0
c chgthpar gauss 3 center    par (ga3cente)        scale 0

c chgthpar gauss 4 intensit  par (ga4inten)        scale 1
c chgthpar gauss 4 width     par (ga4width)        scale 0
c chgthpar gauss 4 center    par (ga4cente)        scale 0



 chgthpar bkgr 1 level                        scale 0
 chgthpar bkgr 1 slope                        scale 0

fit x1 -5*w1 x2 5*w1 go
set  ssqfit12 +ssq
c if +plotyes = 1 then plot



c Q 1.0 1.1 1.2 1.3 1.4 1.5
 chgthpar gauss 1 intensit  par 100        scale 1
 chgthpar gauss 1 width     par 36        scale 1
 chgthpar gauss 1 center    par 0        scale 1

 chgthpar gauss 2 intensit  par 0.04        scale 1
 chgthpar gauss 2 width     par 226        scale 1
 chgthpar gauss 2 center    par -122        scale 1

c Q 0.9
c chgthpar gauss 1 intensit  par 65        scale 1
c chgthpar gauss 1 width     par 36        scale 1
c chgthpar gauss 1 center    par 0        scale 1

c chgthpar gauss 2 intensit  par 0.035        scale 0
c chgthpar gauss 2 width     par 204        scale 0
c chgthpar gauss 2 center    par -56        scale 0

c chgthpar gauss 3 intensit  par (ga3inten)        scale 1
c chgthpar gauss 3 width     par (ga3width)        scale 1
c chgthpar gauss 3 center    par (ga3cente)        scale 1
```

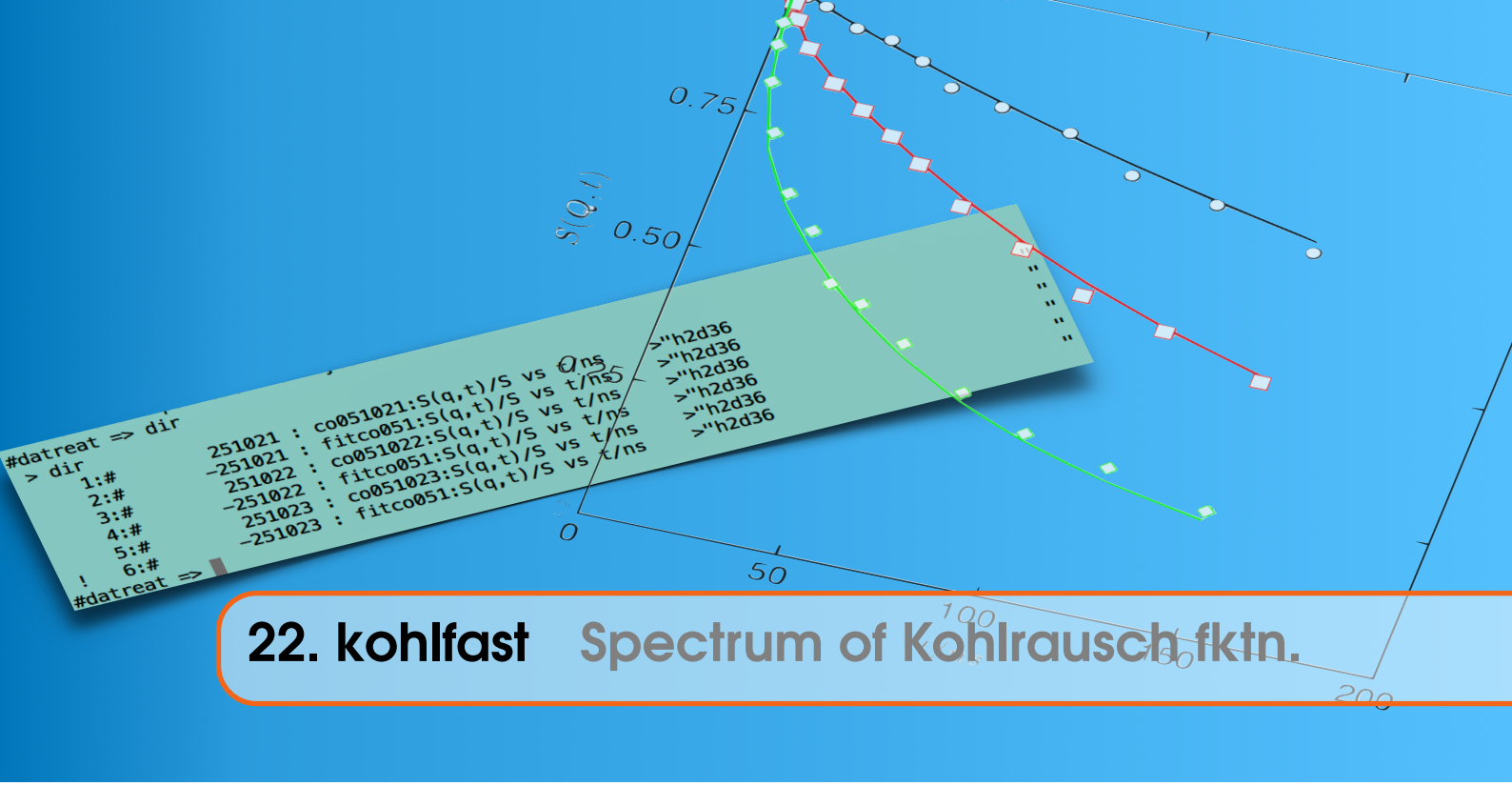
```
c chgthpar gauss 4 intensit  par (ga4inten)        scale 1
c chgthpar gauss 4 width     par (ga4width)        scale 1
c chgthpar gauss 4 center    par (ga4cente)        scale 1

 chgthpar bkgr 1 level par 0.0029 scale 1
 chgthpar bkgr 1 slope par 0 scale 1

fit x1 -1000 x2 1000 go
set  ssqfit13 +ssq

c if +plotyes = 1 then plot ymin -0.05*maxy ymax 0.1*maxy


if +ssq < 2 then goto :ok

echo "CHECK ssq Fit seems to be bad"

fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;fit;

c plot xmin -3000 xmax 3000 ymin 0.1 ymax 100 log_y


:ok

c plot log_y xmin -2000 xmax 2000 ymin 0.001 ymax 100 log_y

c
c transfer resolution parameters to uservariables
c
c
c Read parameters for 4 gaussian type resolutino
c and store it on user defined variables
c
c before calling select the fitted resolution
c
set ga1 +ga1inten
set gw1 +ga1width
set gc1 +ga1cente
c
set ga2 +ga2inten
set gw2 +ga2width
set gc2 +ga2cente
c
c set ga3 +ga3inten
c set gw3 +ga3width
c set gc3 +ga3cente
c
c set ga4 +ga4inten
c set gw4 +ga4width
c set gc4 +ga4cente
c
 set bglevel +level
 set bgslope +slope
c

c goto :ex

gp

c
c Ab hier werden jetzt die Daten selbst aufbereitet und mit den
c Aufloesungsparam,etern gespickt
c
c
c

purge all

set n_container +nbuf

in EmptyCan135k.i96.dat

c in EmptyCan45k.i96.dat

set n_dat +nbuf

 in b300k135.i96.dat

c in b325k135.i96.dat

c in b350k135.i96.dat

c in b375k135.i96.dat

c in b400k135.i96.dat

c in b425k135.i96.dat

c in b450k135.i96.dat

c in b475k135.i96.dat

c in b492k135.i96.dat

c in b300k45.i96.dat
```

```
c in b325k45.i96.dat

c in b350k45.i96.dat

c in b375k45.i96.dat

c in b400k45.i96.dat

c in b425k45.i96.dat

c in b450k45.i96.dat

c in b475k45.i96.dat

c in b492k45.i96.dat

sel +n_dat+1
sel next q (_q_) band 0.001
set n_data_s (sel)

c plot xmin -2 xmax 2 ymin 0.001*maxy ymax (1.2*maxy) log_y

c
c subtract empty container from data
c

sel 0
sel next q (_q_)
sel next q (_q_) add


if +nsel = 2 then goto :ok2

echo "NO DATA WITH THAT ANGLE FOUND"
c goto :ex

:ok2

set result (sel)

sel 0
sel next q (_q_)
sel next q (_q_)
sel add (result)
dsl

arit f1 1 f2 -1  to 101237

msave file2

xformel ;(xx*1000*1.5193)
yformel ;(yy)
funfun immediate op
rename xaxis GHz

plot xmin -2000 xmax 2000 ymin 0.001*maxy ymax (1.2*maxy) log_y

c goto :ex

c
c Push parameters for 4 gaussian type resolution
c to the selected dataset
c use getres to create the necessary variables ga1..gc4
c
c before calling select the fitted resolution
c
c set qd  +q
c if (abs(qd-qr)) < 1d-2 then goto :j1
c Q-values do not match !
 putpar ga1inten +ga1
 putpar ga1width +gw1
 putpar ga1cente +gc1
c
 putpar ga2inten +ga2
 putpar ga2width +gw2
 putpar ga2cente +gc2
c
c putpar ga3inten +ga3
c putpar ga3width +gw3
c putpar ga3cente +gc3
c
c putpar ga4inten +ga4
c putpar ga4width +gw4
c putpar ga4cente +gc4
c
 putpar bgrlevel +bglevel
 putpar bgeslope +bgslope
c

dac

plot xmin -3000 xmax 3500 ymin 0.001*maxy ymax (1.2*maxy) log_y
```

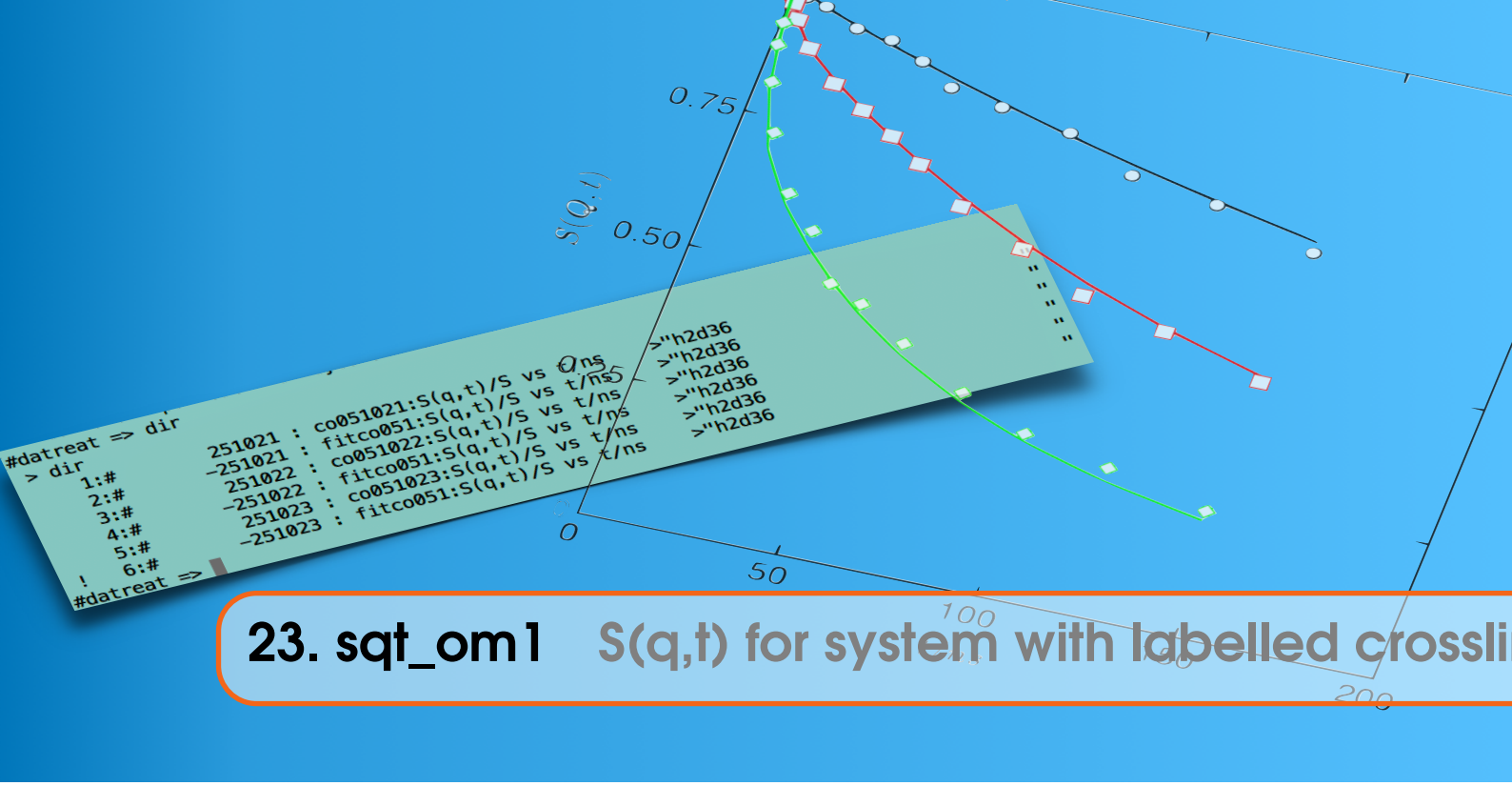
```
c cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c NEXT SECTION
c
c MODEL FITTING (may be a new makro)
c
c
c cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


c dac

c ac kohl_q

!    parnam(1) = 'intensit'          ! prefactor
!         parnam(2) = 'tau0'              ! KWW time-constant, prefactor in front of q-dependence
!         parnam(3) = 'beta'              ! streched exp, prefactor in front of-q-dependence
!         parnam(4) = 'epsilon '          ! accuracy parameter for FT-integrations (DO NOT FIT)
!         parnam(5) = 'omega0'            ! omega scale zero shift
!         parnam(6) = 'u_sqr'             ! <u**2> value for Debye-Waller-Factor
!         parnam(7) = 'j0'                ! jump length (if applicable)
!         parnam(8) = 'beta0'             ! beta offset
!         parnam(9) = 'qexp_t0'           ! q-exponent for tau0
!         parnam(10)= 'qexp_bet'          ! beta-exponent
!         parnam(11)= 'bkgr'              ! constant background
!        parnam(12)= 'n_mg'              ! fraction of hydrogen in side chains

c chgthpar kohl_q  1 intensit  par    0.5        scale 1
c chgthpar kohl_q  1 tau0     par    0.7      scale 1
c chgthpar kohl_q  1 beta      par    0.53   scale 0
c chgthpar kohl_q   1 epsilon   par    0          scale 0
c chgthpar kohl_q   1 omega0    par    0          scale 0
c chgthpar kohl_q   1 u_sqr  par    0          scale 0
c chgthpar kohl_q  1  j0         par    0          scale 0
c chgthpar kohl_q  1  beta0      par    0   scale 0
c chgthpar kohl_q  1 qexp_t0      par    0   scale 0
c chgthpar kohl_q  1 qexp_bet     par    0   scale 0
c chgthpar kohl_q  1  bkgr    par    0         scale 0
c chgthpar kohl_q  1  n_mg     par    0          scale 0

c ac kohl_q

!    parnam(1) = 'intensit'          ! prefactor
!         parnam(2) = 'tau0'              ! KWW time-constant, prefactor in front of q-dependence
!         parnam(3) = 'beta'              ! streched exp, prefactor in front of-q-dependence
!         parnam(4) = 'epsilon '          ! accuracy parameter for FT-integrations (DO NOT FIT)
!         parnam(5) = 'omega0'            ! omega scale zero shift
!         parnam(6) = 'u_sqr'             ! <u**2> value for Debye-Waller-Factor
!         parnam(7) = 'j0'                ! jump length (if applicable)
!         parnam(8) = 'beta0'             ! beta offset
!         parnam(9) = 'qexp_t0'           ! q-exponent for tau0
!         parnam(10)= 'qexp_bet'          ! beta-exponent
!         parnam(11)= 'bkgr'              ! constant background
!        parnam(12)= 'n_mg'              ! fraction of hydrogen in side chains

c chgthpar kohl_q  2 intensit  par    0.5        scale 0
c chgthpar kohl_q  2 tau0     par    0.007      scale 1
c chgthpar kohl_q  2 beta      par    1   scale 0
c chgthpar kohl_q   2 epsilon   par    0          scale 0
c chgthpar kohl_q   2 omega0    par    0          scale 0
c chgthpar kohl_q  2 u_sqr  par    0          scale 0
c chgthpar kohl_q  2  j0         par    0          scale 0
c chgthpar kohl_q 2  beta0      par    0   scale 0
c chgthpar kohl_q 2 qexp_t0      par    0   scale 0
c chgthpar kohl_q  2 qexp_bet     par    0   scale 0
c chgthpar kohl_q  2  bkgr    par    0         scale 0
c chgthpar kohl_q  2  n_mg     par    0          scale 0

c cth

c fit x1 -3000 x2 3000 go



ac kohlfast

c         parnam(1) = 'intensit'          ! prefactor (should be 1)
c         parnam(2) = 'n_chain'           ! chain protons fraction
c         parnam(3) = 'n_ch3'             ! ch3-protons fraction
c         parnam(4) = 'tau_kww'           ! main-chain tau
c         parnam(5) = 'beta_kww'          ! main-chain beta
c         parnam(6) = 'lntau_mg'          ! log of center of methyl tau's (here we enter ln(tau0) !! )
c         parnam(7) = 'sigma_mg'          ! width of lognormal distribution (sigma of the paper)
c         parnam(8) = 'beta_mg'           ! beta of mg rotation (paper ==> 1)
c         parnam(9) = 'rhh'               ! radius of ch3-rotation (paper ==> 1.78 Angstroem)
c         parnam(10)= 'u_square'          ! <u**2> value for Debye-Waller-Factor
c         parnam(11)= 'omega0'            ! omega scale zero shift
c         parnam(12)= 'epsilon'           ! accuracy parameter for FT-integrations (DO NOT FIT)
c         parnam(13)= 'xwidth'            ! channelwidth
c        parnam(14)= 'eisf2'             ! "eisf" for the fast local motion
c         parnam(15)= 'tau_fast'          ! tau of fast component
c         parnam(16)= 'beta_fast'         ! beta of fast

chgthpar kohlfast  1 intensit  par    1        scale 0
```

```
chgthpar kohlfast  1 n_chain   par    0.7     scale 0
chgthpar kohlfast  1 n_ch3     par    0.3        scale 0
chgthpar kohlfast  1 tau_kww   par    0.01       scale 1
chgthpar kohlfast  1 beta_kww  par    0.53       scale 0
chgthpar kohlfast  1 lntau_mg  par   -2.8      scale 0
chgthpar kohlfast  1 sigma_mg  par    0.18        scale 0
chgthpar kohlfast  1 beta_mg   par    1       scale 0
chgthpar kohlfast  1 rhh       par    1.78       scale 0
chgthpar kohlfast  1 u_square  par    0.88          scale 0
chgthpar kohlfast  1 omega0    par    0          scale 0
chgthpar kohlfast  1 epsilon   par    1e-7          scale 0
chgthpar kohlfast  1 xwidth    par    0.1e-3          scale 0
chgthpar kohlfast  1 eisf2     par    1         scale 1
chgthpar kohlfast  1 tau_fast  par    0.3642e-3         scale 1
chgthpar kohlfast  1 beta_fast par    1          scale 0

cth

goto: ex


c fit x1 -1000 x2 1000 go

c ac kohl0ch3

c       parnam(1) = 'intensit'          ! prefactor (should be 1)
c parnam(2) = 'n_chain'           ! chain protons fraction
c       parnam(3) = 'n_ch3'             ! ch3-protons fraction
c       parnam(4) = 'tau_kww'           ! main-chain tau
c       parnam(5) = 'beta_kww'          ! main-chain beta
c       parnam(6) = 'lntau_mg'          ! log of center of methyl tau's (here we enter ln(tau0) !! )
c       parnam(7) = 'sigma_mg'          ! width of lognormal distribution (sigma of the paper)
c       parnam(8) = 'beta_mg'           ! beta of mg rotation (paper ==> 1)
c      parnam(9) = 'rhh'               ! radius of ch3-rotation (paper ==> 1.78 Angstroem)
c      parnam(10)= 'u_square'          ! <u**2> value for Debye-Waller-Factor
c      parnam(11)= 'omega0'            ! omega scale zero shift
c      parnam(12)= 'epsilon'           ! accuracy parameter for FT-integrations (DO NOT FIT)
c      parnam(13)= 'xwidth'            ! channelwidth


c chgthpar kohl0ch3  1 intensit  par    1        scale 0
c chgthpar kohl0ch3  1 n_chain   par    0.7     scale 0
c chgthpar kohl0ch3  1 n_ch3     par    0.3        scale 0
c chgthpar kohl0ch3  1 tau_kww   par    0.01         scale 1
c chgthpar kohl0ch3  1 beta_kww  par    0.5       scale 0
c chgthpar kohl0ch3  1 lntau_mg  par   -3.02      scale 1
c chgthpar kohl0ch3  1 sigma_mg  par    0.2        scale 1
c chgthpar kohl0ch3  1 beta_mg   par    1       scale 0
c chgthpar kohl0ch3  1 rhh       par    1.78       scale 0
c chgthpar kohl0ch3  1 u_square  par    0.95          scale 1
c chgthpar kohl0ch3  1 omega0    par    0          scale 0
c chgthpar kohl0ch3  1 epsilon   par    1e-7          scale 0
c chgthpar kohl0ch3  1 xwidth    par    0.1e-3          scale 0




 c parnam(1) = 'intensit'          ! prefactor
 c        parnam(2) = 'tau0'              ! KWW time-constant, prefactor in front of q-dependence
 c        parnam(3) = 'beta'              ! streched exp, prefactor in front of-q-dependence
 c        parnam(4) = 'epsilon '          ! accuracy parameter for FT-integrations (DO NOT FIT)
 c        parnam(5) = 'omega0'            ! omega scale zero shift
 c        parnam(6) = 'u_sqr'             ! <u**2> value for Debye-Waller-Factor
 c        parnam(7) = 'j0'                ! jump length (if applicable)
 c        parnam(8) = 'beta0'             ! beta offset
 c        parnam(9) = 'qexp_t0'           ! q-exponent for tau0
 c        parnam(10)= 'qexp_bet'          ! beta-exponent
 c        parnam(11)= 'bkgr'              ! constant background


c chgthpar kohl_q  1 intensit  par    0.5        scale 1
c chgthpar kohl_q  1 tau0      par    1.53e-2   scale 1
c chgthpar kohl_q  1 beta      par    0.5    scale 0
c chgthpar kohl_q  1 epsilon   par    0        scale 0
c chgthpar kohl_q  1 omega0    par    0    scale 0
c chgthpar kohl_q  1 u_sqr     par    0    scale 0
c chgthpar kohl_q  1 j0        par    0        scale 0
c chgthpar kohl_q  1 beta0     par    0    scale 0
c chgthpar kohl_q  1 qexp_t0   par    0    scale 0
c chgthpar kohl_q  1 qexp_bet  par    0        scale 0
c chgthpar kohl_q  1 bkgr      par    0        scale 0

c  ac kohl_q

c chgthpar kohl_q  2 intensit  par    0.3        scale 1
c chgthpar kohl_q  2 tau0      par    0.61     scale 1
c chgthpar kohl_q  2 beta      par    1    scale 0
c chgthpar kohl_q  2 epsilon   par    0        scale 0
c chgthpar kohl_q  2 omega0    par    0    scale 0
c chgthpar kohl_q  2 u_sqr     par    0    scale 0
c chgthpar kohl_q  2 j0        par    0        scale 0
c chgthpar kohl_q  2 beta0     par    0    scale 0
c chgthpar kohl_q  2 qexp_t0   par    0    scale 0
c chgthpar kohl_q  2 qexp_bet  par    0        scale 0
```

```
c chgthpar kohl_q  2 bkgr      par    0        scale 0

c ac bkgr
c chgthpar bkgr 0.5 level par 1  scale 1

c goto :ex

c fit x1 -2000 x2 2000 go

c fit x1 -2 x2 2 go

c saverg (int(temp)) (int(q)) 1

c plot xmin -2000 xmax 2000 ymin 0.001*maxy ymax (1*maxy) log_y

c chgthpar bkgr 1 level  par (bgrlevel)     scale 1
c chgthpar bkgr 1 slope  par (bgrslope)     scale 0


c echo "lengthy fit started: Crtl-C (twice when run in emacs) interrupts calc at next brkpt."

c fit x1 -2 x2 2 go
c saverg (int(temp)) (int(angle)) 2
c plot


:ex
```



22. kohlfast Spectrum of Kohlrausch fktn.



23. sqt_om1 S(q,t) for system with labelled crossli

The theory applies the Fourier transform Eq. 21.5 to teh following $S(q,t)$:

$$
\begin{aligned}
S(q,t) &= EISF(q) + (1 - EISF(q)) \exp[-(t/\tau_{inc})^{\beta_{inc}}] \\
&+ A_1(q) \exp[-(t/\tau_{coh})^{\beta_{coh}}] \\
&+ S(q)_{cross} \exp\left(-(qu_{cross})^2 (1 - \exp[-(t/\tau_{cross})^{\beta_{cross}}])\right)
\end{aligned} \tag{23.1}
$$

with

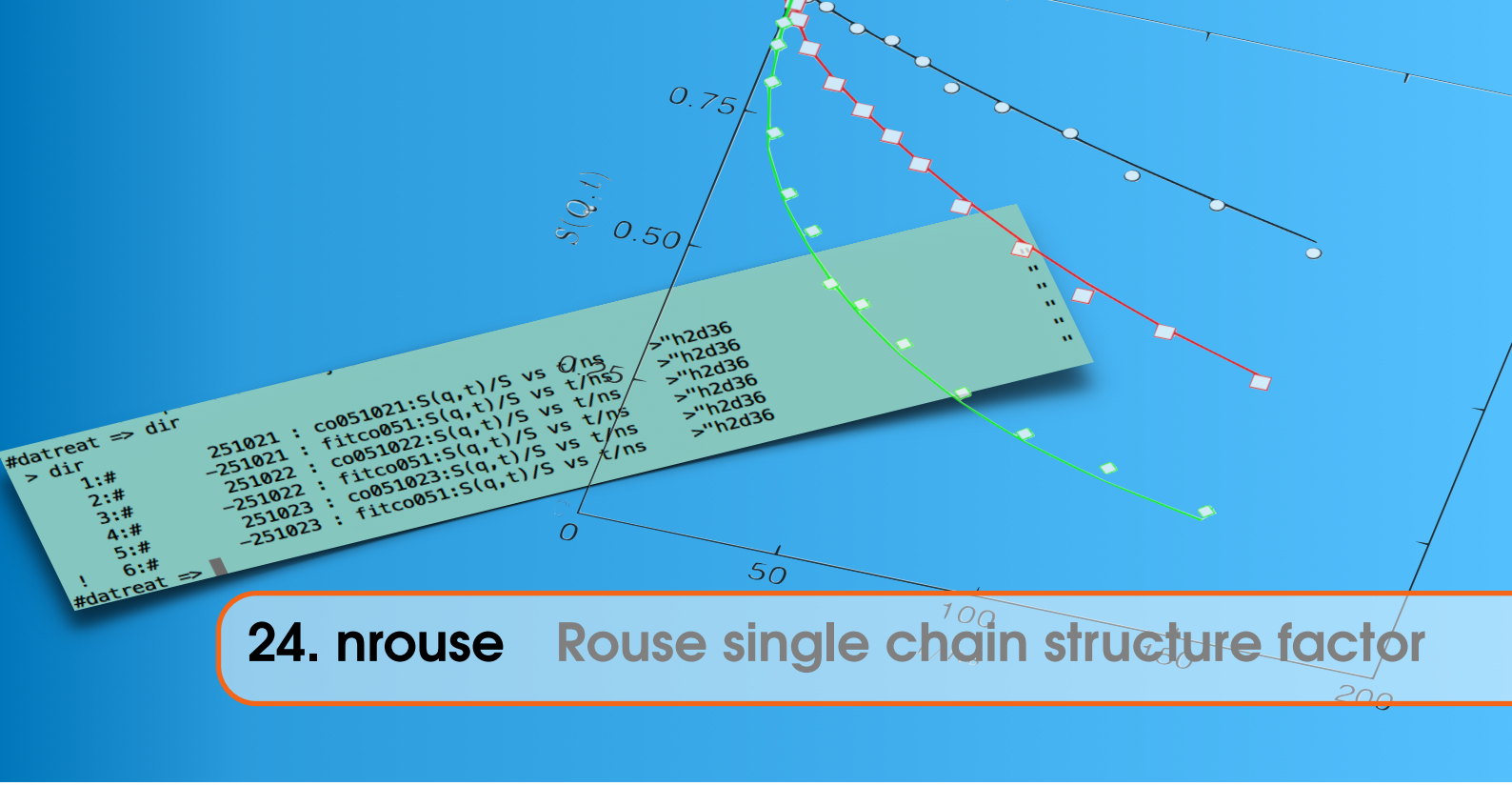
$$\tau_{inc,coh} = \tau_{q;inc,coh}\, q^{\nu_{inc,coh}} + \tau_{0;inc,coh} \tag{23.2}$$

$$\beta_{inc,coh} = \beta_{q;inc,coh}\, q^{\mu_{inc,coh}} + \beta_{0;inc,coh} \tag{23.3}$$

$$EISF(q) = [3 * j_1(q\, r_{jump})/(qr_{jump})]^2 \tag{23.4}$$

$$A_1(q) = a1_0 + a1_{q,\nu} q^{\nu_{a1}} + a2_{q,\nu} q^{\nu_{a2}} \tag{23.5}$$

$$S(q)_{cross} = B_{cross} \exp[-\frac{1}{3}(q\, Rg_{cross})^2] \tag{23.6}$$



24. nrouse Rouse single chain structure factor

$$f^{\text{int}}_{\text{Rouse}}(Q,t) = \sum_{\text{m,n}}^{N} e^{-\frac{1}{6}|n-m|Q^2 l^2}\, e^{-\frac{4Nl^2Q^2}{6\pi^2}\sum_{p=p_{min}}^{p_{max}} \frac{1}{p^2}\cos\left(\frac{p\pi n}{N}\right)\cos\left(\frac{p\pi m}{N}\right)\left(1-e^{-p^2 t/\tau_R}\right)} \tag{24.1}$$

with

$$S(Q,t) = A\, e^{-D_{com}Q^2 t}\, f^{\text{int}}_{\text{Rouse}}(Q,t) \tag{24.2}$$

with $l = \sqrt{R_{ee}^2/N}$ and $D_{com} = W l^4/(3R_{ee}^2)$ if the parameter was set to **zero**, otherwise it gets the fit-parameter value (units cm^2/s).

Direct theory parameters ((fit)parameters in model definition):

amplitu	a, standard pre factor	–	yes
wl4	Wl^4, Rouse rate (units $\text{Å}^4/\text{ns}$)	abs	yes
n_segmen	N, (effective) number of segments/chain	int	no
re	R_{ee}, end-to-end radius of chain, yields l	abs	yes
temp	T, temperature, obsolete not used	abs	no
com_diff	D_{com} in cm^2/s, if $= 0$ Rouse value is assumed)	abs	yes
pmin	p_{min} lowest mode included	int	no
pmax	p_{max} highest mode included	int	no

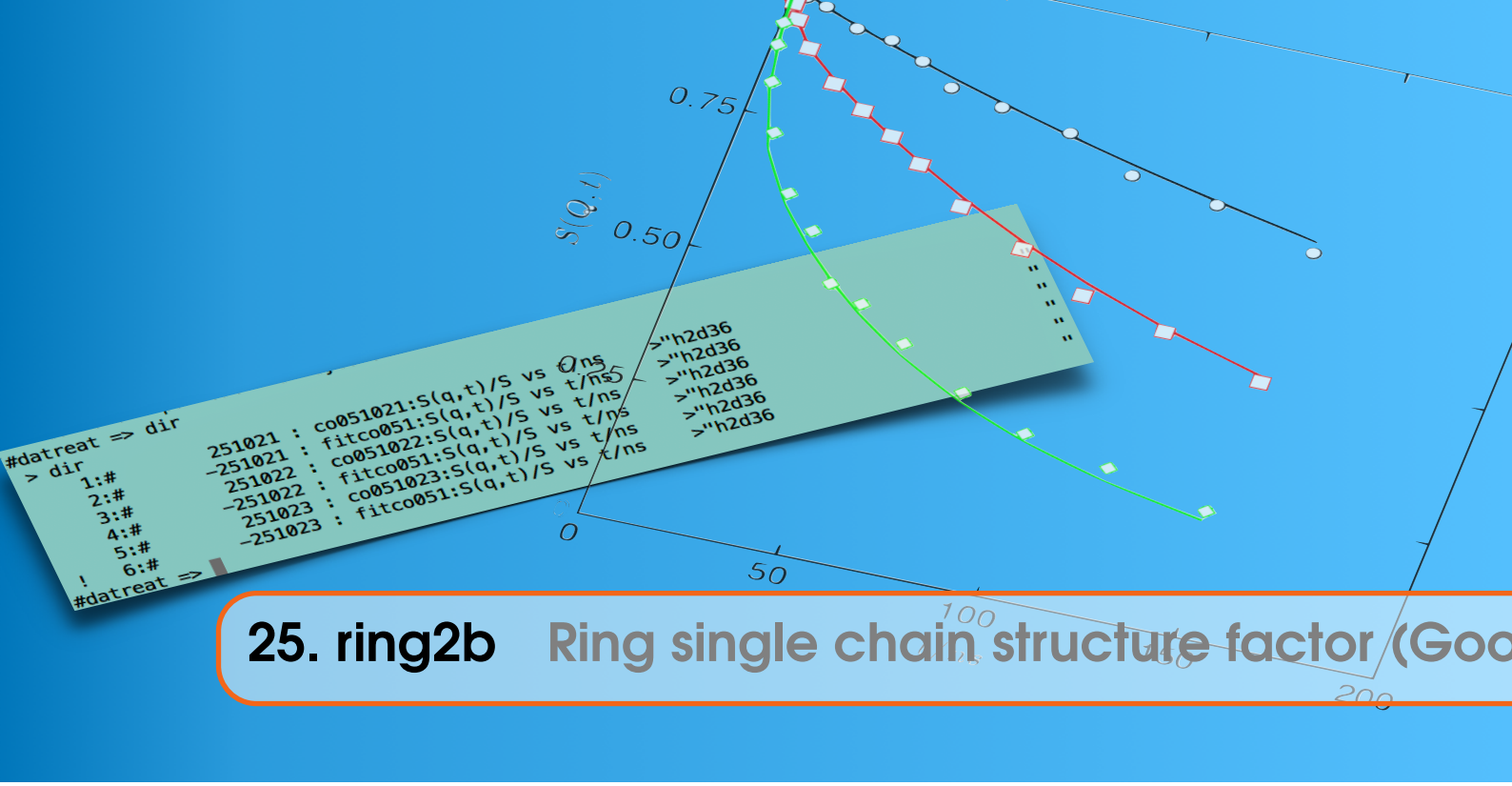
Used record parameters:

q	q momentum transfer of record data
temp	temperature (obsolete)

Parameters pushed to records:

diff	D_{com} in cm^2/s use diffusion coefficient (Rouse)
l	l effective segment length as inferred from N and R_{ee}
w	w Rouse rate for this effective segment length

Note: The *temp* parameter is without function, it was only relevant in theory versions that used the segmental friction as parameter.



25. ring2b Ring single chain structure factor (Goo

For more information: *S. Goossen et al., PRL 2014, 113, 168302*.

$$S(q,t) = exp\left(-\langle r^2(t)\rangle q^2/6\right) \frac{1}{N}\sum_{i,j}^{N} \exp\left[\frac{(ql)^2}{6}(|i-j|\{N-|i-j|\}/N)^{2\nu} - B_{i,j}(t)\right] \quad (25.1)$$

with

$$B_{i,j}(t) = A \sum_{p,even} \frac{F(p)}{p^{1+2\nu}} \cos(p\pi[i-j]/N)[1-\exp(-t\Gamma(p))] \quad (25.2)$$

with the mode p dependent amplitude $F(p)$ and rate $\Gamma(p)$ parameters. Probably $p^{1+2\nu}$ is not adequate and should rather stay at: p^2 (compare ring2a)! Here

$$A = 2N^{2\nu}(lq)^2/(3\pi^2) \quad (25.3)$$

and

$$F(p) = F_0[1-T_f(p)] + F_\infty T_f(p) \quad (25.4)$$

and

$$\Gamma(p) = [1-T_f(p)]p^{colorred4\nu}/\tau_R + T_f(p)p^\mu/\tau_\infty \quad (25.5)$$

Probably $p^{4\nu}$ is not adequate and should rather stay at: p^2 (compare ring2a)! the transition function is

$$T_f(p) = \{1 + \exp([p - p_{min}]/p_{width})\}^{-1} \tag{25.6}$$

and the Rouse time is computed as

$$\tau_R = N^2/(W\pi^2) \text{ with } W = Wl^4/l^4 \tag{25.7}$$

Scaling (by factor η) works **approximately** with $N \to \eta N$ and $l \to l/\eta^\nu$!

■

And the subdiffusion part:

Diffusion factor:

$$exp\left(-\langle r^2(t)\rangle q^2/6\right) \tag{25.8}$$

with

$$\langle r^2(t)\rangle = \left[\left(e^{\frac{1}{\beta}\left((-\alpha+\beta)\ln\left(\frac{r_1{}^2}{r_2{}^2}\right)+\beta\alpha\left(-\ln\left(\frac{r_2{}^2}{6D_0}\right)\right)\right)} r_2{}^2 t^\alpha\right)^a + \left(e^{\left(-\ln\left(\frac{r_2{}^2}{6D_0}\right)\right)\beta} r_2{}^2 t^\beta\right)^a + \left(6D_0 t\right)^a\right]^{1/a} \tag{25.9}$$

describing sublinear-linear center-of-mass diffusion with a normal long time diffusion constant D_0 and a sublinear diffusion with exponent α until the first kink at the msd level r_1^2 and with an exponent β until the second kink at the msd level r_2^2. The sharpness of the transitions is controlled by , the larger the value the sharper is the transition. Intended use assumes that $0 < \alpha < \beta < 1$ and $r_1^2 < r_2^2$. This is NOT automatic ! ■

Direct theory parameters ((fit)parameters in model definition):

ampli	A, standard pre factor	–	yes
diff	D_0, long time diffusion	abs	yes
r021	r_1^2, MSD at first kink in subdiffusion	abs	yes
alpha	α, initial subdiffusion exponent	abs	yes
r022	r_2^2, MSD at 2nd kink in subdiffusion	abs	yes
beta	β, 2nd subdiffusion exponent	abs	yes
a_cross	a, sharpness of kinks	abs	(yes)
n	N, (effective) number of segments/chain	abs	no
l	l, (effective) segment length	abs	yes
nue	ν, chain expansion exponent	abs	yes
wl4	Wl^4, Rouse rate (units Å^4/ns)	abs	yes
pmin	p_{min}, transition mode number between simple ring-Rouse and large p modification	abs	yes
pwith	p_{width}, sharpness of transition	abs	yes
f0	F_0, pre factor $F(p)$ limit for small p values (default 1)	abs	yes
finf	F_∞, pre factor $F(p)$ limit for large p values	abs	yes
tauinf	τ_∞,for large p: $\tau(p) = \tau_\infty/p^\mu$	abs	yes
pexinf	μ, for large p: $\tau(p) = \tau_\infty/p^\mu$	abs	yes

Determination of R_g by comparing the Guinier expression and the Gaussian path sums at low q: The ring $P(q)$ follows from the common expression with Gaussian distribution of distances according to $|n-m|^\nu$:

$$P(q) = \frac{1}{N^2}\sum_{i,j}^{N,N} e^{-1/6(ql)^2(S_{i,j})^{2\nu}} \simeq 1 - \frac{1}{N^2}\frac{1}{6}\sum_{i,j}^{N,N}(ql)^2(S_{i,j})^{2\nu} + \cdots\cdots \tag{25.10}$$

with N=total number of segments and

$$S_{i,j} = (|i-j|\{N-|i-j|\}/N) \tag{25.11}$$

as in the ring $S(q,t)$ expression comparing to the Guinier expression:

$$P_{\text{Guinier}} = e^{1/3(qR_g)^2} \simeq 1 - q^2R_g^2/3 + \cdots \tag{25.12}$$

leads immediately to

$$R_g = \sqrt{\frac{l^2}{2}\frac{1}{N^2}\sum_{i,j}^{N,N}(S_{i,j})^{2\nu}} \tag{25.13}$$

This is what the program reports as Rg. ■

Used record parameters:

q q - value of record data

Parameters pushed to records:

Rg R_g as derived form Eq. 25.13

25.1 TO BE MODIFIED...

Unify amplitude ==> yscale as pa(1) automatic in template !
Unify xscale ==> xscale as pa(2) automatic in template ! What are the best names for these (xscale, yf, ampli, ...; xscale, xf, strech, ...)??

VIII Installation

26 Installing datreat (on Mac) 121

26.1 Preliminaries: compiler and gr-software system preparation

26.2 Directories and paths

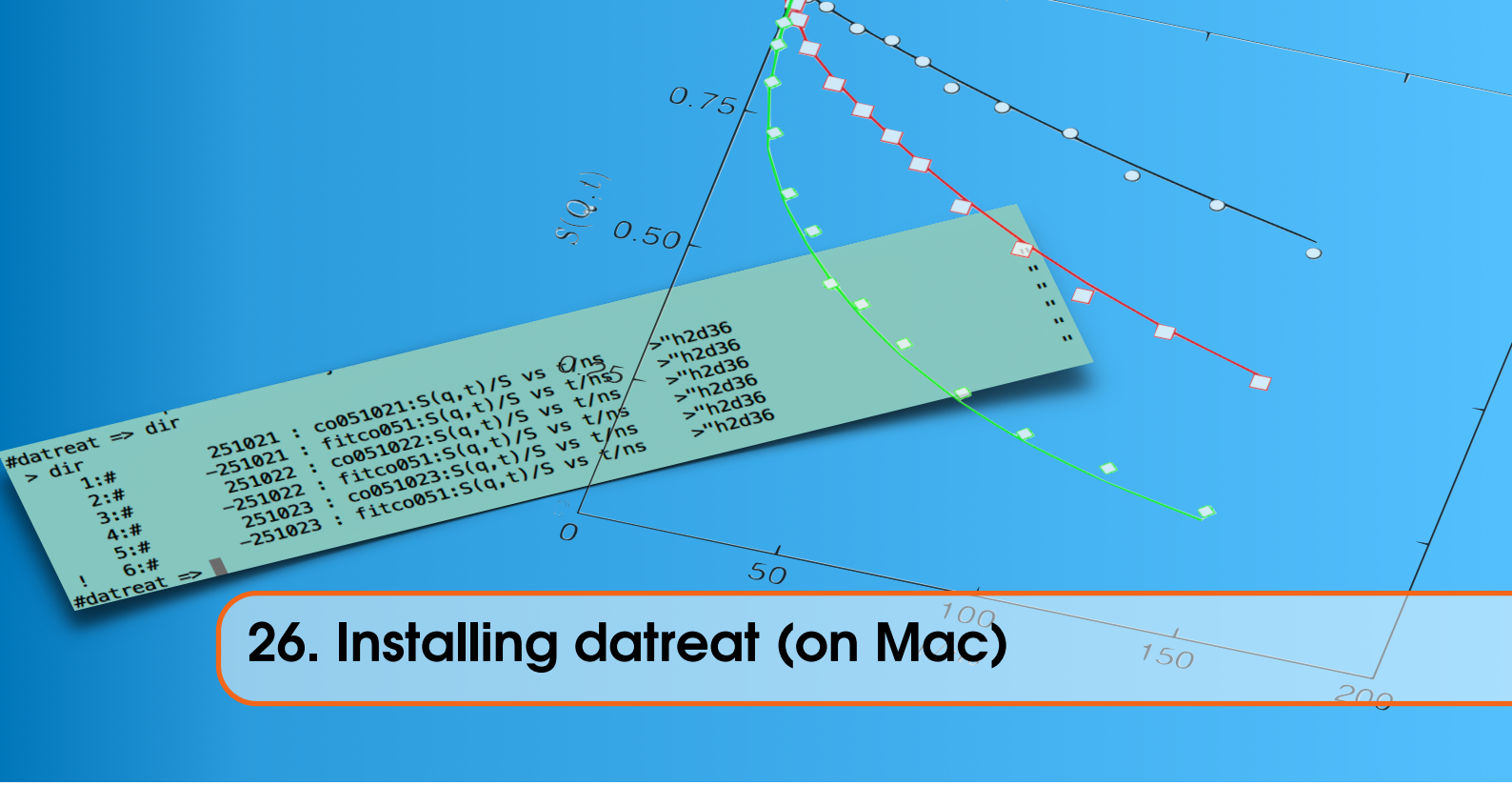
26.3 Datreat making datreat, build

Bibliography 127

Articles

Books

Index 129



26. Installing datreat (on Mac)

Here we describe the installation procedure for **Mac OSX**.

26.1 Preliminaries: compiler and gr-software system preparation

Installation of datreat requires the prior installation of a **gfortran** compiler and the **grframework** library (J. Heinen).

For gfortran goto https://gcc.gnu.org/wiki/GFortran

For grframework goto https://gr-framework.org/c.html

Note: The FZJ mac-setup should already contain these packages. Use **macu** to get the latest versions!

26.2 Directories and paths

Before installing datreat make sure that you have a **~/Desktop/bin** directory, where the *datreat* executable will be stored.

```
...> mkdir ~/Desktop/bin ↵    Create bin directory
```

This directory should also be included into the **PATH** environment variable:

```
...> cd ↵
...> echo 'export PATH="$HOME/bin:$PATH"' >> ~/Desktop/.profile ↵
```

TBD(datreat src):Put this into an install procedure!

26.3 Datreat making datreat, build

Now we can install datreat itself. First choose a directory where it should be stored. In principle any location should work, however, it is recommended to copy the *datreat* directory to **~/Desktop**, i.e. "everything" is in **~/Desktop/datreat/...**. This is assumed in this manual.

26.3.1 Copying

```
...> cp -r MYSOURCE/datreat mkdir ~/Desktop/datreat ↵
```

or copy it by any other means (drag symbol to Desktop ...).

26.3.2 Compiling

After copying open a **terminal** goto the **src** subdirectory and use **make**:

```
...> cd ~/Desktop/datreat/src ↵
...> make cleanall ↵
...> make ↵
...> cd UTILITIES ↵
...> gfortran -o th_template_generator th_template_generator.f90 ↵
```

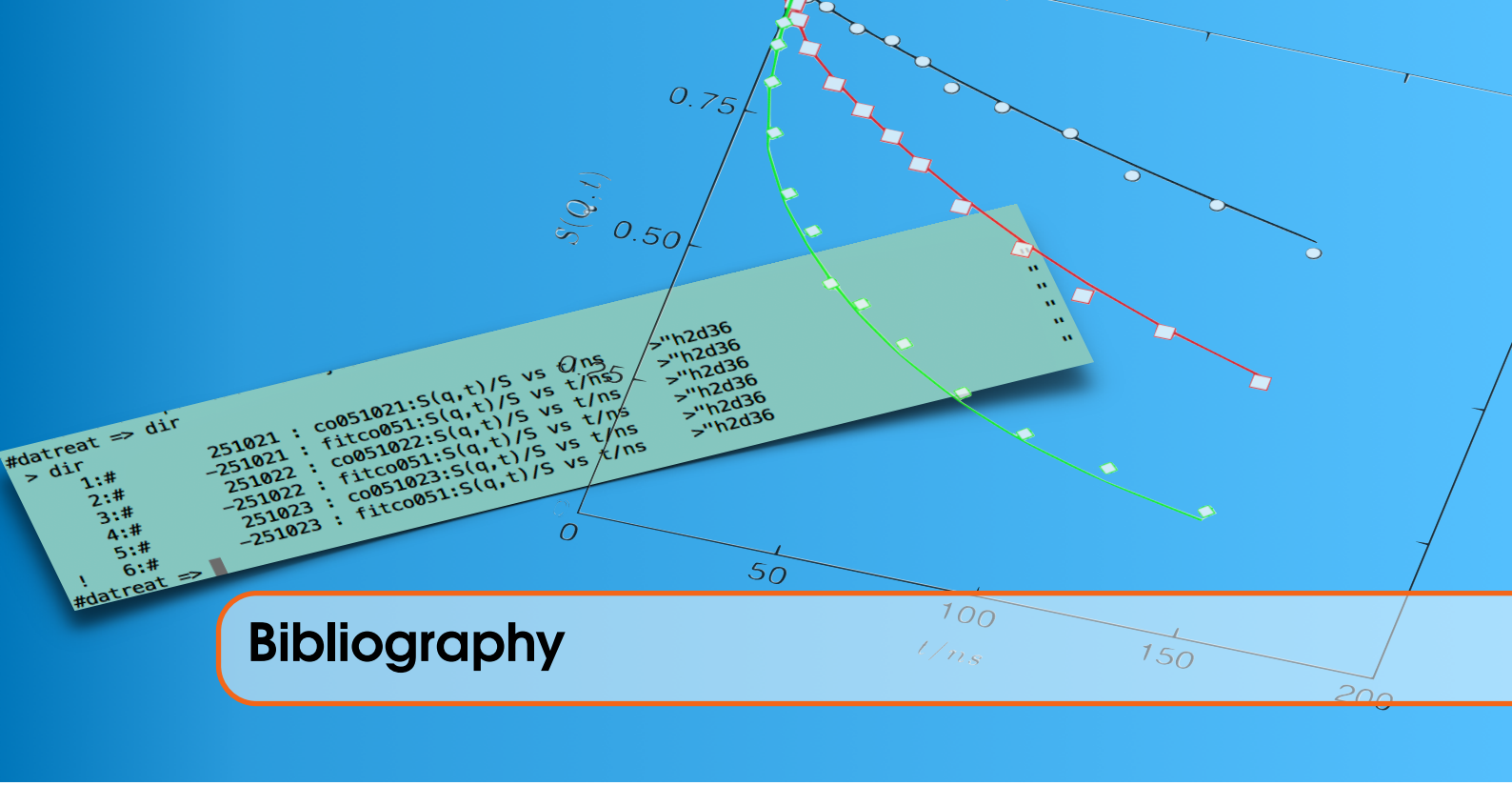
26.3.3 Choosing a favorite editor

TBD(datreat src):Establish a consistent mechanism to set editors!

```
command name                parameters:
addval
==============================================================================
= addval <x> <y> <err>
=    appends one data point to selected record
=    only the first selected record is treated
==============================================================================
clip   (obsolete use rerange)
combine  (better use average , keep it ??)
copy
creatser
cs        (synonym) clearsel
des       (synonym) desmear
detsens
dmx
echocurv
ercorrc   (synonym) ecc
fft
fftmx     (synonym) fft-ms
fftrand
fit
fit_plus
fun       (synonym) function
funfun
fxy                          & x=(formel) y=(formel)
ga_fit
gaiscale
gdos_q
gen.res
hiqextra
ia        (synonym) i-abso
in        (synonym) input
inscn
integrat
interpol
invers
kz
label
m         (synonym) mirror
mux
noise
numorchg
numorpls
fopen
out_gli   (synonym) gli
paraout
parextra
oplot    plot to ocular
```

```
prompt
putpar   !!!!!!!!!!!!!!!!!!!!!!!
parlev
qc        (synonym) q-conv
rebin   (use average)
scale
sequence
serfit
seterr
sp        (synonym) spline
swapxy
sym
th_init
tofdos
tracorr
uni_ft
write
yformel
z         (synonym) zero
zero
```

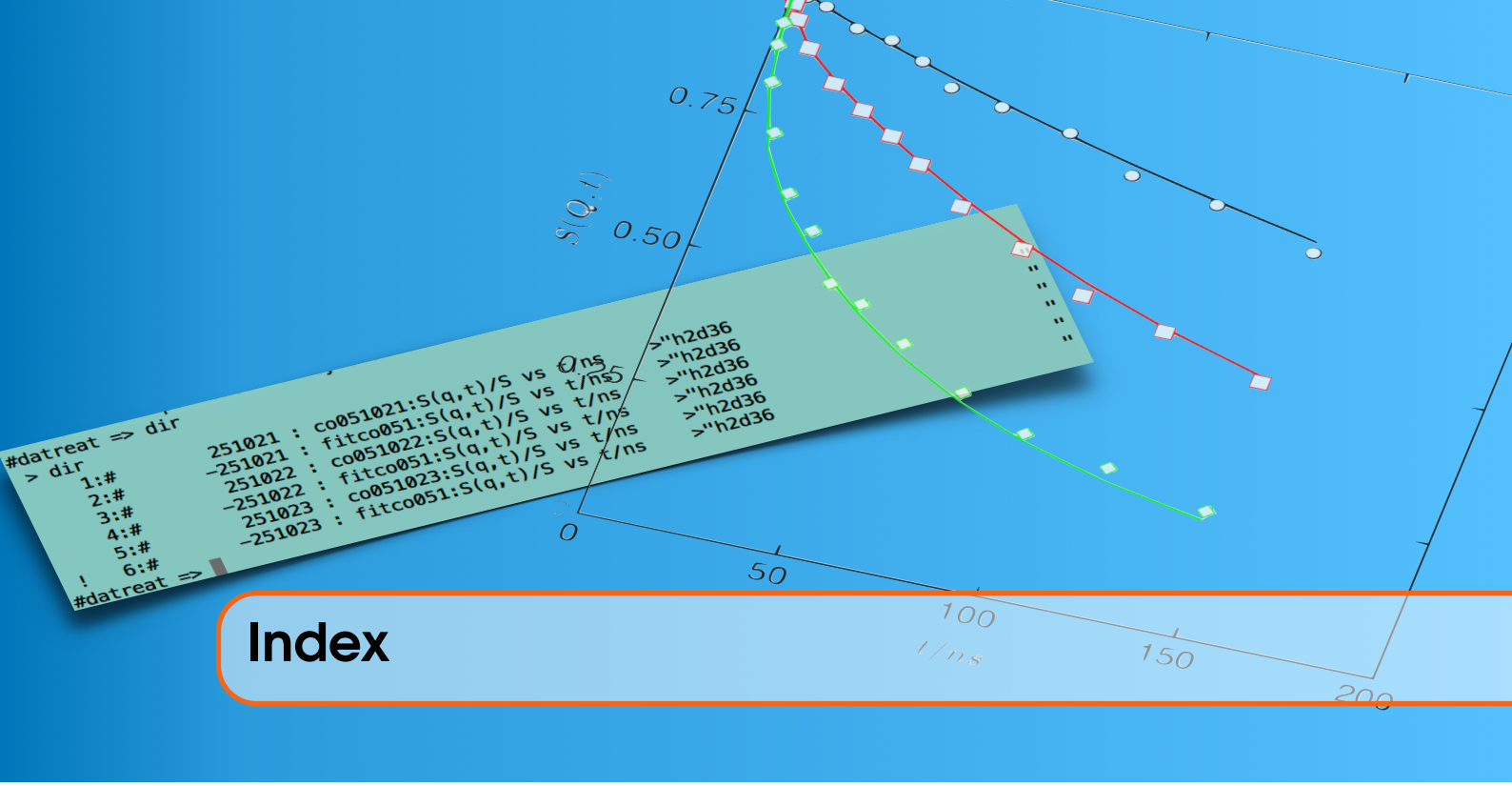
```
      if(comand.eq.'integrat') then
!                    --------           approximates integral from binned data
!                                       by simple summation of data times bin-width
!                                       there may be better interpolation based schemes
!                                       but this is clear an simple
       if(found('help    ')) then
          write(6,*)'=====================================================================
          write(6,*)'= integrat x1 <x1> x2 <x2>
          write(6,*)'=        approximates integral from binned data over the given range
          write(6,*)'=        by simple summation of data times bin-width
          write(6,*)'=        there may be better interpolation based schemes
          write(6,*)'=        but this is clear an simple
          write(6,*)'=====================================================================
          goto 2000
       endif
        if(nsel<1) then
          call errsig(200,"ERROR: integrat slect at least one record$")
          goto 2000
        endif
```



Bibliography

Articles

Books



Index

<, unix redirection, 14
>, unix redirection, 14
>>, unix redirection, 14
[..], 9
(dir), 20

(dir), 20

activate a theory, 25
activate theories, 46
add many consecutive records (recsum), 40
add many records (addsels), 40
add two records, 39
aligny, 35
apply functions on x- and y-values, 37
arguments (makro), 13
automation and makros, 82
average, 33
axes latex, 29

bin, 113
build (datreat), 114

cat, unix file content, 14
cd, unix change directory, 14
change fitting parameters (cth), 47
change theory parameter, command, 53
chgthpar, 53
clearing records, 20
combine records, 35
combine theories, 46
combining records, 33
command line, 10
compiler, 113
copy records, 21
cotxtdtr, 85
coupling (fit parameters), 51
cp, unix file copy, 14

data conversion example, 85
data structure, 9
deconvolution, uni_ft, 69
deleting records, 20
display data, 27
display memory content (records), 19
display selected records, 24
divide y-values of two records, 39
duplicate records, 21

echo phase scan, 71
echocurv, 71
edit data, 30
edit fitting parameters (cth), 47
editor, 114
editor, choice, 30
enter fitting parameters (cth), 47
errors (creation), 75
exclude points from a record, 35
exit, 15
expression, variables and functions, 11
expressions, 12

extract theory parameter, 55
extract theory parameters, 55
extthpar, extract theory parameter, 55

f(x)->x, 37
f(y)->y, 37
fit (coupling parameters), 51
fit (parameter coupling), 54
fit (record selective theory), 53
fitting, 45
fitting a series of records, 55
fitting a sum of exponentials, 56
fitting multiple records discriminated by parameters, 21, 54
format (input files), 19
fortran, 85
fourier transform of spectra, 69
functions, 10, 11
functions and variables in datreat, 11

ga1center, 94
ga1inten, 94
ga1width, 94
gauss, 91
gauss2, 91
gaussft, 91
Gaussian noise, 75
graphics software, 113

inplus, 20
input of data from files, 19
installation, 113

kohlfast, 103
kz channel contraction, 35

label (fit parameter coupling), 54
latex axes, 29
libraries (program), 113
list data, 30
list selected records, 24
listing loaded files, 19
load previous theory definition, 26

MAC-OS, 14
make (datreat), 114
makro, example, 82
makros, 13
match record scales, 35
mexp, 56
model definition, 25, 46
model fitting, 45
msave, 20
msave (read stored data), 20
multiply y-values of two records, 39
mv, unix file move (rename), 14

narrow data range of a record, 35
noise, add noise to records, 75
notation, 9
nrouse, 105

offset, 35
operations, 9

path, 113
path (input files), 19
phase scan echo, 71
plot, 27
plot (colors), 28
plot (comparing curves with individual scales), 27
plot (legends), 28
plot (letter and symbol sizes), 28
plot (lin, log axes), 27
plot (publication ready), 28
plot (save and print), 29
plot (style), 28
plot (title), 29
prepmak, makro example, 82
print, 55
program (language), 58
purge, 20
putpar, 21, 54

quit, 15

range (theory), 53
rebin x-channels, 35
record, 9
record parameter, 21, 54
reducing number of points, 33
rename (axes), 29
rerange, 35
resolution convolution, 93
resolution parameters, 94
resolution, uni_ft, 69
restore, 22
restore saved content, 21
restqore state, 22
restrict data range of a record, 35
ring-model, direct summation, 107

ring-model, Goossen, 107
ring-model, subdiffusion, 107
ring2, 107
rm, unix file remove, 14
Rouse model, direct summation, 105
rp.f90, 85

save state, 21
saving result values/parameters to a file, 55
saving results and parameters on a file, 20
scale, 35
sel, 9, 23
selecting records for actions, 23
selection, 9
selection (for next actions), 23
serfit, 55
serial fitting, 55
set theory parameter, command, 53
shell script, 85
show data, 30
simple start, 81
sort x-values, 33
spectra, 93
start, 79, 81
starting datreat, 15
store, 21
strexpo, 89
subtract two records, 39
sum of exponentials, 56

target records, 23
template (theory creating), 58
terminal, 79
theories, 57
theory (code structure), 60
theory (create a new one), 57
theory (select form unused), 57
theory (template generator), 58
theory (test and use of new th), 60
theory define (activate), 25
theory parameter extract, 55
theory parameter extraction, 11
title, 29
transform x- and y-values, 37

uni_ft, fourier transform of spectra, 69
UNIX, 14
user defined, 10

value extraction, 11
variables, 10, 11
vs (dir), 20

workflow, 79
working directory, 79