Introduction to datreat

Content

Datreat Purpose and Philosophy

Datreat is a command-based multi purpose system that allows to deal with collections of (x,y)-data (data-records) each of which has an individual block of parameters. Data may stem from any field ranging e.g. from SANS diffraction results, NSE-S(Q,t) curves, magnetic field values along a path, BSS-spectra etc.. Datreat allows to plot, manipulate and fit these data-records. Theoretical models may be assembled from existing "theories" or new theories may be formulated (which, however, requires some Fortran programming). Simultaneous fitting of collections of data-records that are distinguished by one or several parameters is supported as well as coupling of fit parameters.

Some automation of data-evaluation is possible with the help of "makros". These are files that contain a collection of genuine commands including conditional and unconditional jumps that allow also the construction of loops.

Interaction with the program is effected by commands. Each command line consists of a ***command-name*** and a number of ***items*** that are ***separated by blanks***. ***Command line item***s may be ***names***, ***numbers*** or ***expressions*** that evaluate to a number.
In general ***expressions*** may be used at any place in ***command lines*** and ***input files*** where a ***number*** would be appropriate.
The system classifies all items that start with one of the following characters as ***expressions***:
(+-.0123456789
any item that would start with one of these characters and would not lead to a valid ***expression*** leads to an error.

Despite some preliminary mechanisms to specify a ***path*** the recommended use of Datreat is to call the program from the current data-subdirectory and keep the specific makros and other auxiliary files there too.
RB: But it is usefull to discriminate between raw data and evaluated data plots ... See **path** at the end

To ***exit the program***: **→ q**

Hint1: Running datreat from an Emacs-shell offers a number of advantages, in particular the availability of command-line history (use ALT-p, ALT-n) to retrieve previous/next command inputs. In addition all editor functions are available on the whole output produced by the current session.
A simple history (20 commands) exists. Accessible by "_xxx" the last command starting with xxx ist executed again.

Hint2: Command hierarchy: Commands are first used as datreat commands than as makro name. After both failed (eg no makro with name pwd) command is given to a new shell instance and executed. This means you can use any shell command as you are used to it, as long as no makros has the same name. (see history for a nice example how to use this feature)
Here we will write always MAKRO with k because the makros are detected by the German „makro" in the first line.

Hint3: If a makro "**initdatr**" exists within datreat/makro or in your local path this will be executed right after start of datreat. Its a normal makro and you can define your own path, standard data treatment plotting .... whatever you can imagine.

Data Structure

Files

Data files: contain a number of data records each of which consists of a header, a parameter section and a list of x,y values with an optional third column of y-errors.
Theory definition file: always has the value **lastth.** To save previous versions copy it onto another name.
Makro files: contain a collection of command lines their names follow the same rules as the data files. To qualify as a makro the name must also be different from any genuine datreat-command and the first line must start with the keyword: **makro**.
Shell commands are allowed. **Lines are limited to 1024 characters**.

Internal Representation

Data and theory definitions are stored internally (when loaded) in common blocks with fixed (may be changed by recompiling the program with other limits) maximum capacity.
The data on these structures are largely accessible by predefined system functions.

Datreat Commands

Input/Output

Input of Data / Display of List of Loaded Data

in **/input**	filename
	reads data (x, y, y-error) from <filename>. <filename> may not contain one of the fllowing characters: '+-0123456789.('. <filename> can be a complete path prepended by variable <datapath>. If <filename> starts with ./ or / (./path/name or /path/name) <datapath> is NOT prepended. **Hint:** if data stem form a different computer system, make sure that they do not contain tab's and extra CR-characters etc., these cause occasionally observed errors during reading. The last read data record is automatically selected. All previous selections are removed. **NEW Data format** Simpler format rules: **parameters and comment ahead of Data** parameter : any line starting with a name followed by numbers (parameter name; first number is set as value; second number as parlev (output control with plot) data : line with minimum 2 values (x,y); 3rd is used as error; others are ignored comment: everything not data or parameter; first comment line is stored; other comment lines are ignored ignored lines: start with "#" as first character; "values" and "paramete" also ignored new set: initiated by non Data after Data (parameters, comment or empty line) or "#nxt" **New** **in** now reads also data in the 'inx'-format (e.g. spectra vs. meV or tof-channel from IN5). In order to reckognize this format the files must have the extension '.inx'. Further if reading inx data the option 'GHz' may be specified. In that case the meV scale of x is translated to circular frequency ω in GHz. This option is ignored for tof-channel data.

Special functionality added to the input routines:

- variables **read1** and **readlast** are created/updated to give the frist and last addresses of the last read-in contents for automatic addressing purpose.
- a variable **numor0** may be set prior to input of inx files, The automatically generated numor values then start at numor0+1.

OLD Data format for input files:

```
q= 0.161 dpeo25pmma75 no argon :comment
j04n019 ncounts vs omeg/Ghz 10308  :name xname vs yname <numor>
parameter                :keyword=parameter
normal  0.10000000E+01   :f_parnam <value>
lambda   0.62710000E+01 : „ „ „
q        0.16070952E+00
vzg_velo 0.19110000E+05
temp     0.39580000E+03
chan_v0  0.12700000E+03
d_energ  0.17000000E+02
v_5v     0.50000000E+01
: blank line
values :keyword=values
-0.25345777E+02 0.64695009E-02 0.17290469E-02 : <x> <y> <y-error>
-0.25129146E+02 0.63254744E-02 0.19072023E-02
......
0.24479254E+02 0.88531187E-02 0.18874913E-02
0.24695885E+02 0.78431373E-02 0.16721625E-02
: blank line
#nxt  :keyword=#nxt
q= 0.002 dpeo25pmma75 no argon
j04n019 ncounts vs omeg/Ghz 10308
parameter
normal 0.10000000E+01     0
...
v_5v 0.50000000E+01


values
-0.26212299E+02 0.46082949E-02 0.46082952E-02
-0.25995668E+02 0.14817254E-02 0.49390846E-03
-0.25779038E+02 0.16228497E-02 0.72576046E-03
...
0.23829363E+02 0.74321813E-03 0.52553458E-03
0.24045993E+02 0.79744817E-03 0.56388101E-03
0.24262624E+02 0.78369906E-03 0.55415892E-03


#eod :keyword=#eod
```

Output and manipulating of Data

save	[<isel>] **to** filename
	saves data from selected respectively <isel> internal record to <filename>. If <filename> starts not with "." or "/" savepath is prepended. Current **restriction**: <filename> may not begin with one of the fllowing characters: '+-0123456789('. **Hint:** behind the **data** the saved file will contain the theory setting definition which was active at the time when the save command was issued. Note that this will only be useful if save is performed before a different theory setting which does not represent the situation of the saved data is prepared! **Hint2**: To restore the theory including parameters use : > csplit <saved_filename> %theory% -f <theory_filename> resulting in a file < theory_filename00 > which you can cp to lastth, activate it by > acl and recalculate theory by > thc
msave	filename
	saves all selected data records to <filename>. If <filename> starts not with "." or "/" savepath is prepended. Current **restriction**: <filename> may not begin with one of the fllowing characters: '+-0123456789('. **Hint:** behind the **data** the saved file will contain the theory setting definition which was active at the time when the save command was issued. Note that this will only be useful if save is performed before a different theory setting which does not represent the situation of the saved data is prepared!

	Use: get_th <filename> to load the th-setting only; in <filename> to load the data only; or the macro inplus <filename> to load both. NEW: virtually no length restriction for filenames.
clip	[[from <n1>] [to <n2>]] [last <n>] [errmax <val>] [rel]
	removes points from a loaded data record. Either by specifying the range of data point numbers <n1> ..<n2>, resp. last <n> or by setting a limit to the maximum error. **rel** modifies the error condition to hold for relative errors. **Note:** the action only affects the loaded data record and not the data in the input file. The userfunction indxval(ibuf,x) or intxval(ibuf,x) to extract point numbers ni for x-values.
rerange	<x1> <x2> [y]
	creates a copy that contains only the points within the given range <x1>…<x2>; if the option **y** is specified the range pertains the y axis, elswhere x is considered.
copy	[x1 <x1> x2 <x2>]
	copy selected records (selection is transferred) to new records at the end of the dir list. If x1 AND x2 are given only data from that interval are copied.
sequence	
	replace x-values by sequence numbers, applies to all selected records
swapxy	
	exchages x and y columns for all selected recors. Errors are zeroed.
edit	<n> [sc <numor>]
	allows to edit data of data-record <n> or data-record with <numor>. The contents is written to a file ***datbuf*** which is laded to an editor and reread after the editor is left. **Note:** the action only affects the loaded data record and not the data in the input file
putpar	f_parname <value>
	adds a parameter or changes a parameter value of the selected data-record.
parlev	f_parname <value>
	modifies the display level (plot) of parameter (the higher, the more supressed), display can be controlled with: plot parlev <lev>
paraout	
	lists parameters of selcted record
numorchg	<newnumor>
	assigns new numor to first item in the selected list.
dir	[**clength** <displayed-comment-length>] [f_parnam1] [f_parnam2]…

	displays a list of the loaded data records. Selected records are marked by a '**!**'. The **clength** parameter allows to control the number of characters that are dsplayed from a comment. If f_parname is given only data records that contain a parameter with that name are listed and the parameter value is displayed. It is possible to specify more than one f_parname.
z	
	zeroes data-record counter,. logically clears data-record list.

Associated variables:

nbuf	evaluates to the number of loaded items, i.e. points to the end of the dir-list.
num(j)	evaluates to the "numor" of the j-th loaded data record.
nv(j)	evaluates to the length (number of points) of the j-th loaded data record.
xv(j,i)	evaluates to the x-value of the i-th point of the j-th loaded data record.
yv(j,i)	evaluates to the y-value of the i-th point of the j-th loaded data record.
ye(j,i)	evaluates to the y-value error of the i-th point of the j-th loaded data record.
sumx(j,i1,i2)	evaluates to the sum of x-values from the i1-th point to the i2-th point of the j-th loaded data record.
sumy(j,i1,i2)	evaluates to the sum of y-values from the i1-th point to the i2-th point of the j-th loaded data record.
indxval(j,x)	evaluates indes number I of first point with x(I) > x (integer value)
intxval(j,x)	evaluates interpolated floating point "address" of x

Selecting dataset

In general actions triggered by datreat commands are performed on all data records that are selected.
In special cases a certain sequence of selections assigns data records to different roles in command performance (see command descriptions).

sel	<n1> <n2> ……….<nx> [add]
	selects exactly those data records with (internal) numbers <n1>, <n2> ….<nx>. The option **add** causes that the previous selctions are kept and the new ones are added to the list.
sel	<n1> -<nx> [add]
	selects all data records with (internal) numbers <n1>...<nx>
sel	all f_parnam <value> band <range> [add]
	selects all data records which have a parameter with name **f_parnam** which has a value within the range: <value>±<range>.
sel	next f_parnam <value> band <range> [add]
	selects next data record (after actually selected) which hase a parameter with name **f_parnam** which has a value within the range: <value>±<range>.

sel	[fits] [fit+]
	selects fitted data, or fits associated to selected data (fit+).
sel	narrow f_parnam <value> band <range>
	narrows selections.
sel	exclude f_parnam <value> band <range>
	exclude selections.
sel	exclude numor mod <n>
	exclude records with run numbers mod(#,n) = 0
sel	
	removes selections.
dsl	
	displays a list of the current selections. (see also **dir**)
purge	<n1> <n2> ……….<nx> [all]
	removes data records <n1> ..<nx> or all data records from the dir-list. **Note**: all selection will be removed!

Associated variables:

sel	evaluates to the last selected data record number.
nsel	evaluates to the number of selected data records (length of selected list).
isel(i)	evaluates to the data record number of the i-th selected list item.

Plotting

Use xmgrace as plotting interface (gplot) solely
if the Grsoftware supporting the plot command is not installed !

Try to use plot instead!

gplot	<?/help><nf><fp><ne><nl><cl><cn><na> <k0><com><gr><jp><sv><so><cleanio>
	gplot,gp to plot selected files in connected xmgrace gp plots into same grace until connection is closed without open connection a new grace process is created common parameter were extracted and a legend is shown build up from uncommon params' in grace: C-t write text C-d delete text; C-M moves objekt C-L moves legend

	all parameters were transferred to set comments in grace to identify datasets save your plots in the file menu as grace file (?.agr or ?.xgr) if you want to modify later Print it in menu File/Print Setup (Device setup opens) choose format and size (press apply) (for .eps choose Tight box in device options) printed in menu point File/Print
Options: help, h, ? switches **on/off'** fits, fi fitpara, fp error, er legend, le jointpar,jp close, cl clearplo,clear autoscal, as	stored for this help text hide fits (but transfered) write last fit parameters no error bars (but transfered); can be changed in Set Appearance hide legend; if parameters were appended they build up legend; e.g. “gplot le q” hides parameters in legend except q values write joint parameters' close connection to grace after plotting; with “save name cl” in a script a new plot is saved and then closed gp starts always with an empty clean plot; works only for graph 0; datasets 0-99 are deleted autoscale switched off/on, default on
normal: closenow, cn kill, ki command, com graph, gr start save, sv setoffset, so cleanio	closes connection without plotting of new data kills given sets in grace k,l deltes sets k l ; n -m kills from n to m; -m kills from 0 to m rest of input line as direct command to grace see xmgrace help or grace files as example' plot data in graph i, a new graph inside grace is build,' number i is stored for next dataset arrange them in EDIT/ARRANGE Graphs' The rest of the line will be sent directly to grace'; used to customise the startup behaviour of xmgrace. Its a semicolon separated list of xmgrace commands e.g (see saved *.agr file) ##> gp start title "NSE data";subtitle "\xt\N is a fouriertime";xaxis label "\xt\N / ns"' save graceplot with following name (.agr is added)'; if name is “on” updated plots are saved always to “lastgrace.agr”; without name its saved to “lastgrace.agr” and switches save off. offset for numbering new data, default 0 data with smaller numbers are preserved' deletes iofile.tmpg?s? if exists'
	hints to grace
gplot com <grcom>	grcom is a direct command to grace some usefull examples: *page size 400, 300* resizes page to 400, 300 pixels (try what you like most) *xaxes scale Normal* scale set to lin title “this is the main title” obvious yaxis label "S(q,\xt\f{}) / S(q,0)" obvious, see xmgrace help for text *yaxes scale Logarithmic* scale set to log *redraw* redraw plot needed sometimes do see the effect of previous commands the full command in datreat is “gp com redraw” to install the .grace directory with default configuration files of xmgrace copy the directory config.grace to your home directory as .grace To change the default startup layout of your gracewindow open .grace/templates/Default.agr and save it again after you changed whatever you wanted.

	*To change the default behaviour look in grace.user to use something like this input filter for .dat files. accept only lines starting with +,- or a number for .dat files DEFINE IFILTER "egrep '^ *([+,-,0-9,\.]|$)' %s" PATTERN "*.dat"*
gplot start xxxx	xxx is the rest of the line after start. It is a semicolon separated list of xmgrace commands like above and will be used for all of your following new xmgrace windows in this session. To have a permanent behaviouzr use initdatr initialization file in your makro path.

Plotting

plot	[xmin <xmin>] [xmax <xmax>] [ymin <ymin>] [ymax <ymax>]
	plots selected data records (including fit/thc results if present) using the (optionally) specified range. Default is *symbols* for data and *lines* for fit/thc results.
plot	[*option1*] …….[*option_n*]

	OPTIONS:
	`log_x` : log scale on x-axis (range given by xmin, xmax must be > 0)
	`lin_x` : resets the default linear x-scaling
	`log_y` : log scale on y-axis (range given by ymin, ymax must be > 0)
	`lin_y` : resets the default linear y-scaling
	`symb <s1> [...<sn>]` : assigns symbols to plotted curves
	`icolo <c1> [...<cn>]`: assigns colors to plotted curves
	`errplo` : plot with error bars
	`noerrplo` : plot without error bars (default)
	`parplo` : plot with written parameters at the side (default)
	`noparplo [f_parnam1] [..[f_parnam_n]]`
	: plot without parameters at the side except those listed: [f_parnam1…..
	`framx <fx>`
	`framy <fy>`
	`frlux <ux>`
	`frluy <uz>` : changes size and shape of plot area (axes) on the screen <fx> <fy> dimension
	: of plot window, <ux> <uy> location of lower left (values in cm relative A4)
	`legsize <ls>`
	`legx <lx>`
	`legy <ly>` : text size and relative location of block of written parameters.
	`txsize <ts>`
	`font <n>` : text size <ts> (cm) and font (by number <n>)
	`parlev <lev>` : level up to which parameters are shown in plot (default 0), increase to see more
	`text`
	`notext` : allow or suppress text plotting completely.
	`o <option>` : GR-software option specifier string (graxs-subprogram). the format is e.g.
	`o` : ox x=1 {shows x-axes below your graph}{2 puts the axes above} and {3 on both sides. -1/-2/-3 shows a grid}
p0	
	sets/changes options only without producing a plot.

tit	<title-string>
	sets a title string that occurs as title on the plot. tit 1 to 1 copies the input string, title “allows for strings and evaluated items” x
rename	[**xaxis** <new-string>] [**yaxis** <new-string>] [**name** <new-id>]
	sets new names (labels) for xaxis, yaxis or ident of the selected data record.

The plot command is based on the call of the FZJ/ZAM GR-software package. It asks for an output device number to poduce the plot. For X-terminals (and that is the standard now for all workplaces) the number to be given is **211.**
How to get a printed output?
If the plot shall be printed or stored as eps-file, the output device number is **62.** This causes the plot command to write a file named *gliXX.eps* with *XX* the number of the plot invocation during the current session. This file may be copied to another name and/or printed using the UNIX **lpr** command.

Fitting

Setting-up Theories and Parameters

theos	**<string>**
	lists names of all available theory types. (short: th) <string > If string is given only theos starting with string are listed. The always available pseudo-theory **eval** is not quoted. It is intended to allow for the ad-hoc fitting of simple expressions that have to be enterd using the **yfitformel** command.
ac	theoryname [**multiply**] [**range** <datparname> **min** <p1> **max** <p2>]
	activates one instance of theory theoryname which is appended to the current list. **Options:** **multiply** if given the value from this theory instance is multiplied to the result of the previous list, **range** if given the name of the data parameter datparname and min and max values are to be specified. The theory will only be evaluated if the value of <datparname> as parameter of a dataset under consideration is present and its value is in the interval between <p1> and <p2>. **Hint:** the range option is intended to be used for fits of datasets of a series with different values of an experimental parameter (like an increasing Temperature stored as <datparname>). A theory instance may be activated for each dataset and assigned to it by appropriate specification of the range interval. Those theory parameters that shall be common to the whole data sets may be coupled (see below). Any individual scalings etc. may simultaneously be effected by keeping the corresponding parameters as individual fit parameters,
acl	activates last theory stored in file lastth
	HINT: To reuse a complete theory including parameters (how to restore a theory setting from saved data see **save**) copy the lastth to a different file name and copy it back to file name lastth if needed like this: >ac strexpo : activates theory strexpo >emacs lastth : opens emacs to modify parameters close it.or use **chgthpar** >cp lastth lasth_strexpo : copy to new filename (backup of your settings) to reuse it: >cp lasth_strexpo lastth >acl
gth	gth <filename>
	synonym: **get_th** reads the automatically stored theory specification that is appended to files written by the msave command (filename) and reactivates them.

	The actual theory setting is replaced by that.
dac	
	desactivates all theory instances. After **dac** the theory definition is completely cleared.
al	
	lists current theory definition .on console and into file ***lastth***.
chgthpar	theoryname [<instance>] t_parname **par** <p1> **scale** <s1>
	sets parameter value <p1> and scale <s1> for parameter _t_parname of the <instance>-th instance of theory theoryname. If scale <s1>=0 the parameter is fixed during fits, otherwise <s1> should set to a value of the same order of magnitude as the expected parameter value.
label	theoryname [<instance>] t_parname label
	assigns label label to parameter _t_parname of the <instance>-th instance of theory theoryname. The label may contain of a maximum of 4 characters. The labels are used to establish parameter couplings between different theory instances.
couple	theoryname [<instance>] t_parname label <factor>
	installs a coupling of a labelled parameter onto a parameter as quoted in this command, The coupling factor is <factor>. All changes with respect to the start values of the labelled parameter are multiplied by facor <factor> and added to the parameter quoted in this command. Coupling to several lablelled parameters is possible. Parameters with couplings will not be fitted, the corresponding scale **must be zero.**
yfitform;	(xx*p(1)+p(2)*p(3).......)
	definition of the special theory **eval.** xx means the actual x-value, p(1)…p(9) are parameters of the theory. Note the necessary '**;**' between yfitformel and the expression in brackets. f_parnams may be used.
cth	change theory parameter
	Makro: allows editing of the complete theory definition list.

Data format of theory list:

```
theory zimm                                        1. instance of theory zimm
a1 intensit 0.9765E+00 0.1E+00 0.39E-02
   eta_solv 0.8220E-03 0.1E-02 0.18E-04
   epsilon  0.1000E-06 0.0E+00 0.00E+00
   temp     0.3000E+03 0.0E+00 0.00E+00
   com_diff 0.0000E+00 0.0E+00 0.00E+00
theory zimm                                        2. instance of theory zimm
   intensit 0.9765E+00 0.0E+00 0.39E-02 a1 -1
   eta_solv 0.8220E-03 0.1E-02 0.18E-04
   epsilon  0.1000E-06 0.0E+00 0.00E+00
   temp     0.3000E+03 0.0E+00 0.00E+00
   com_diff 0.0000E+00 0.0E+00 0.00E+00
theory rouse range q min 0.1 max 0.3               1. instance of theory rouse
   intensit 0.0000E+00 0.0E+00 0.00E+00
   xi_frict 0.1000E+09 0.0E+00 0.00E+00
   b_segmnt 0.5000E+01 0.0E+00 0.00E+00
   epsilon  0.0000E+00 0.0E+00 0.00E+00
   temp     0.3000E+03 0.0E+00 0.00E+00
   com_diff 0.0000E+00 0.0E+00 0.00E+00
   q_width  0.0000E+00 0.0E+00 0.00E+00
end
```

label par scale couplings

	All numeric entries may also be expresssions which will be converted to their numerical values upon reading. Remember: expressions must be given in brackets () or begin by +.

Theory Computations and Fittings

thc	[**n** <num> **x1** <x1> **x2** <x1>] [**convolute xc1** <xc1> **xc2** <xc2>]
	computes theory values for the selected data using the activated theory list. Default is theory ***evaluation at the location of data points*** only. If **n** is specified the values are not computed at the data point positions but on ***<num> equidistant points in the interval between* x1 and x2**. If the value ***<num>*** is ***negative*** the interval is divided in equidistant steps on a ***logarithmic scale***. The option convolute applies for SANS data only and requires special parameters in the data sets (see SANS). A first call of thc must contain x1 and x2 values.
fit	[**x1** <x1> **x2** <x1>]
	runs a fit of all selected data records with the list of activated theories. By <x1> , <x2> a (sub) interval for data comparison may be specified. If any parameters are give go is needed to get an immediate start of fit, otherwise a second call of fit without parameters starts the execution. Further **options**: writes one line into the opened parameter file according to the actual selection and/or theory status. fit [sc scan1 [scan2 ...]] [x1 startvalue] [x2 endvalue] .. and other thc opts. [auto] [ngood est_no. of valid digits in the theory] [maxfn] max_fun_calls] [relerr] [abserr] (default) [wrtfit] writes fitted curve data (x,y,yth) at each step onto file ***fitdat.tmp*** [nowrtfit] (default) [errors] [map mgrid] [div ndiv] [parwght] if nonzero the scale parameter deviation from their start values is include into the error signal, this may be used to restrict the parameter variation. Default is 0. Wth the **fit** -command the parameters of the activated theories are fitted to your selected data (more than one data-record may be selected !) x1 specifies the lower x-value of the fit-interval, x2 the upper value. If **auto** is given x1 and x2 values are ignored and all data are respected in the fit. the next time x1 or x2 is given again a corresponding limit is again established. See also the command **clip** . With **maxfn** you can set the maximum number of function calls that take place during the fit-procedure. The **go**-option from previous version has beeen removed. Convolution with a resolution function (SANS) may be performed; for details see **fit** ! Another way to deal with resolution convolution is decribed in the **BSS** part. **map** option causes **fit** to writes files ***map.xy*** and ***map.ssq*** that contain the landscape of ssq vs paramaters (**max. no of free pars.=3**) The parameters varied are those with nonzero fitscale. The stepwidth used is fitscale/**div** (**div** may be given as parameter in the command line, default is **20**). **map** must be followed by the ***number of grid points*** in each direction. Use ***ssq.gli*** to display the resulting files (2D).If map is given **NO fitting** will be performed ! The comman dline must be finished by the parameter: **go**
	Using **parameter values for output or in makros** is possible by taking advantage of one of several mechanisms: 1: **fit** and **thc** add a number of paramters to the resulting curve buffers these may be addressed in other

	functions or viewed in the plot. These parameters describe the theory parameters and errors. In the moment we are still restricted to 8 characters for the names therefore the theory parameters are coded as follows: first 2 characters = first two charactres of theory name 3rd character = one digit (1..9) telling the number of the theory instance in the activation list. 4..8 character = 1..5 charcater of the th-parameter name The corresponding names for errors are build by putting 'e' in front of the coded parameter name and truncating the result to 8 chars; those parameters may be used in expressions. See **plot noparplo** to get a plot with a non crowded parameter part 2. The last instance of a parameter name is used to create a user defined variable of that name and the value of the parameter. The corresponding error has the same name with a trailing '***e.***'. This mechanism is only viable if one or a few instances of different theories are defined. 3. The internal vectors that hold the parameters are accessible via **+th_par(i,j)**, **+th_err(i,j)** are evaluated to the value of the i-th parameter (parameter-error) of the j-th instance of any theory.
ga_fit	[**npop** <np> **ngen** <ng> **mutation** <mr> **bits** <nb> **trace** <-2..2>]
	simple genetic algorithm to find the optimum parameters. The algorithm is useful to fit simple (fast to compute) functions with many parameters. The parameters are: **npop** = size of 'population' [100] **ngen** = number of generations [100] the total number of function calls is: **npop** × **ngen** **mutation** = mutation rate [0.001] **bits** = resolution of parameter coding in bits [10] **trace** = controls amount of output [-1] The theory definition is the same as for 'fit', however, the meaning of the scale parameters is slightly different, except for scale 0, which still means fixed parameter. For parameters to be fitted the scale parameter codes the range of parameter search which is **[startvalue-scale/2, startparameter+scale/2]** After a **ga_fit** a following call of **fit** is recommended to increase the accuracy.
th_init	not yet implemented
	initialises the theories if they were changed during execution of datreat After changes in a theory or adding a new theory you have to compile it and linkit by invoking "make pur"in the main directory of datreat. this will look for new theories and compile the shared library libtheospur.so in datreat/lib which is used for the theories. All this is done automatically with the comand **th_make** below compiles the theos library afterwards you need to reinitialise the theories with **th_init**

Saving Parameters and Fit Results

Output/Edit Theory Parameters

al	

activlst	
	lists current theory definition .on console and into file ***lastth***.

Write a List of Any Parameters/Variables

print	filename expr1 [expr2] [expr3] …………..[expr9]
	appends one line of numbers to file <filename> expr1..9 denote expressions that compute to a number. They may parameter names, user-defined variables, automatic variables or expressions containing numbers and/or these variables. **Remember:** only strings that start with one of the following characters : '+-0123456789.(' will compute to a number, an open bracket '(' requires a closing counterpart ')', blanks are not allowed within an expression. Refer to section ***PredefinedVariables/Functions*** for a list of useful links to internal information.

Cast Parameters into a Datreat Input File

open	filename parnam1 [no.theo1] parnam2 [no.theo2]
	will prepare a file named file <filename> in datreat readable input format for output. This file will be filled with the values of the parameters with names <parnam1> as x-values and <parnam2> as y-values. The parameters may be taken from the parameter-block associated with the currently selected file (1st file of selection list) or from the parameters associated with the currently activated theory setting, in the latter case it may be specified, by giving the number <no.theo>, that the parameter is to be taken from the <no.theo>-th instance of the theory's parameter name in the activated list. Open write a header.
write	
	writes one line into the opened parameter file according to the actual selection and/or theory status. **Tip:** to use any other values make them a parameter of the selected file by the command putpar.
close	
	closes the open parameter collecting file and writes **#eod** mark.

Data Manipulation

General Purpose

arit	**f1** <f1> **f2** <f2> **to** <numor> [**mult**] [**div**] [**norm**|**nonorm**]
	linear combination of two selected data-records with factors <f1> and <f2>, the result gets the by **to** assigned <numor>. Default is addition, the options **mult** or **div** select multiplication or division of y-data. norm or nonorm relate to monitor normalization, this is only effective if the data-records contain a corresponding parameter ***monitor***.
addsels	
	adds all selected records and creates a new record with the sum that inherits the parametyer of the middle record of the series (to be improved). It assumes equal structure of the slected records.

combine	**raster** <xstart> <dx> <n> **to** <numor> [**norm**|**nonorm**]
	combines contents of selected data records and interpolates them to a common grid (raster). norm or nonorm relate to monitor normalization, this is only effective if the data-records contain a corresponding parameter ***monitor***.
average	**xcatch <dx>**
	error weighted average of different selected records, xcatch is the relative distance below which the data points are combined. Typical use: select all q <qx>; average
fun	function [<value>]
	applies a function to x or y-values f selected data records. See also: **xformel** and **yformel** as alternatives. function: **x : x ---> x** **log(x) : ln(x) ---> x** **exp(x) : exp(x) ---> x** **x**2 : x^2 ---> x** **x* <f1> : x* f1 ---> x** **x+ <s1> : x+ s1 ---> x** **sqrt(x) : sqrt(x)---> x** **rouse : q^2*√(wl4*x) --> x (q from parameter, wl4 from command or parameter) Diffusion correction see y-values below.** **zimm : (q**3*kT/(6*pi*eta)*x)**(2/3) --> x (q, temp, eta_solv from parameter, or command)** **y : y ---> y** **log(y) : ln(y) ---> y** **exp(y) : exp(y) ---> y** **y**2 : y^2 ---> y** **y* [f1] : y* f1 ---> y** **y+ [s1] : y+ s1 ---> y** **sqrt(y) : sqrt(y)---> y** **deff : y/x^2 ---> y** **rouse: y*exp(diff*q^2t) → y (q from parameter, diff from parameter in units cm**2/s (default))** **Diffusion correction is only performed if commond is** **→ fun rouse usediff <diff> [unit <fc>]** **where <diff> is the parameter name which holds the diffusion value and unit is the conversion factor of the unit of diff to A**2/ns, default is 10^7. If usediff is not given no diff-scaling of y values is performed.** with appropriate calculation of errors The results are written to newly created data records.
xformel	;(xx*3+q…….)
yformel	;(sin(yy)*xx^3…….)
	allows to specify expression by which the x and y-values (denoted xx and yy within the expressions) of the selected data record are transformed. The action is initiated by the command funfun. Note the semicolon ; which is necessary here. The expression is given within brackets, no blanks.
funfun	[immediate] [op]
	treats selected data-records according to the x- and y formulae which are either taken from internal storage as specified by the **xformel** and **yformel** commands if the option immediate is given **OR** are taken as

	consecutive lines (max length 132) in the file ***formdat***. If the option **op** is given the data transformation id performed “on place” otherwise new data records are created. Errors are treated appropriately.
seterr	
	creates an error column for the selected data-records by using the expression specified by the command **yformel**.

SANS Related

invers	[bkgr <b>]
	creates a data-record with $x \rightarrow x^2$ and $y \rightarrow 1/(y-b)$.
mirror	<k0> <k1> <k2>
	generates a mirror-image of a selected data-record **k0** is the estimated x-center **k1** and k2 specify the range that shall be used to determine the center.
qc	
	converts x-values (channels) of the first selected data-record to q-values. For that purpose the data-record has to have the following f_parameters: xk0, lambda, detdis, bklen ; i.e. channel of zero angle, wavelength, detector distance, size of one detector channel.
spline	[iequal <i>] [nneu <n>] [auto|noauto smpar <s>]
	spline interpolation of the fiirst selected data-record. **nneu** specifies the number of points of the splined data. auto selects automatic parameter determination, noauto allows to specify smpar <s> to control the amount of smoothing.
des	[nneu <n>] [qmax <qm>] [errabs <ea>] [errrel <er>]
	infinite slit height desmearing. **nneu** specifies the number of points of the treated data. Needs spline immediately before beeing called. Needs f_parameter: delqv
mux/ dmx	trans <tr> thick <t> xmax <xmax> nfft <n>
	SANS multiple scattering computation / removal.

```
c yi <--- i0 * sigma(x) * [unit sample thickness:d] =
c lim[d-->0] ( i-measured(x,d) / d )
c the output data on y will the be scaled such that they represent
c yo <--- i0 * i-measured(x,d) / d (note: no limit !!)
c --------------------------------
c i.e. the scattering intensity including all orders of multiple
c elastic small angle scattering will be calculated for a sample
c of unit thickness assuming the same primary intensity factor,i0,
c as for the input data.
c -------------------------------------------------------------------
c input variables:
c t ....... : transmission reduction factor due to small angle sc.
c xmax .... : largest scattering angle (or q) to be considered
c dx ...... : increment of x
c nx ...... : no. of points to be generated
c nfft .... : fft no. of points (optimal choice nfft=2**m)
c output variables:
c x(1..nfft/2) : output x-values
c y(1..nfft/2) : output y-values
c -------------------------------------------------------------------
```

Backscattering/Spectrometer Related

uni_ft	
	Fourier trans for of selected spectrum. If spectrum and resolution are selected (observe sequence of selection!) then also the 'deconvoluted' time function is produced. The results (forurier 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 opserved: micro-eV, meV, GHz, omega (→ GHz) (GHz means giga rad/s)

Obsoletes

out-gli	<filename>
	writes data in simple x y column form.
inscn	<filename>

makro Language

Call

A makro is a file (name max. 8 characters) which contains a header line and a collection of command lines.
The filename must be different from the names of genuine commands.
At call a number of values may be given after the name that replace the parameter-variables in the makro.
I.e. call:
makname 1.23 op3 (expression1) .xx.
invokes a makro with name maknam the first parameter evaluates to 0.123000e+00 the second is a string op3 the third gets the value of (expression1) and the last one is again a string xx. Also shell commands are involved, so you can also change file names etc.

Nested calls up to a nesting level of 10 are supported. However, all variables have global scope!

It is possible to use normal shell commands in a makro eg to copy a fit result or rename a file.
To load a list of files with unknown filenames (perhaps in a makro to load all b___???? files inside a directory (with a line "q 0.05" use grep))
Some commands to generate this makro (do it inside of datreat):
echo makro > newmakro :writes newmakro with line "makro"
ls b___41* | sed 's#^#in #' >> newmakro :sed appends „in „ in front of filenames
echo sel 1 2 3 4 select some data
echo gp > newmakro : gp will plot selected data

start it with newmakro
think about the possibilities

Header and Parameters

The first line of a makro must start with the keyword: **makro .**
Then a number of arguments **_arg1_** …may follow which generally give the heading line the following structure:

makro _a1_ _a2_ _this_ _another_

in the makro body the arguments are ***string replaced*** at any place where they occur by the values given when the makro is called. Expressions are first evaluated.

Expressions

Expressions evaluate to a (real) number and may be used throughout ***in commands and input files*** wherever a number is expected. The system identifies an expression by the occurrence of one of the following characters at the beginning of a blank-separated item:
+-.0123456789(
they may contain the following binary operators:
+ - * / ^
they may use the ***system specific, user-defined or automatically defined variables and functions*** as well as the following standard mathematical functions:
sqrt(x),sin(x),cos(x),tan(x),asin(x),acos(x),atan(x),ln(x),exp(x),abs(x),int(x)

Expression may ***not contain any blanks***, since these are considered as item separators throughout the system.

Loops

Loops must be formulated explicitly using **if** and **goto**.

Commands that relate to the makro language

set	v_name <(valuelexpression)>
	creates uservariable v_name with value or value of expression
vars?	
	lists all defined variables and values
??	(expression)
	displays the value of an expression, internal parnames from datasets can be used (value from first of multiple selected sets used) ?? +q evaluates the internal data variable q to its value expression to be evaluated are indicated by brackets () or +- or contain +*/- like a+b*q set q2 q^2 to calculate the square of q and access it latter as uservariable q2

clr	v_name
	removes a variable. **Hint:** a variable of the same name will mask the visibility of am f_parameter. To access the f_parameter the variable must be cleared. Vice versa creation of a corresponding variable may deliberately be used to perform this making.
if	(expression1) **<|>|=|<=|>=** (expression2) **then** cmd|**goto** :l
	realizes an if condition by comparing the values of two expressions.
goto	:label
	jumps to label. The label has to start with a colon '**:**' and must be the only item of a line.
cms	<system~command> **obsolete** non commands or makros used as shell commands
	transmits the given system command to the shell. All blanks within the system command may be replaced by '~'. By using this mechanism errors may be avoided that would occur if the command contains items beginning with '+','-' or '.' which normally would be considered as expressions to be evaluated by the datreat input-communication module.

Predefined Variables/Function

nbuf	evaluates to the number of loaded items, i.e. points to the end of the dir-list.
num(j)	evaluates to the "numor" of the j-th loaded data record.
nv(j)	evaluates to the length (number of points) of the j-th loaded data record.
xv(i,j)	evaluates to the x-value of the i-th point of the j-th loaded data record.
yv(i,j)	evaluates to the y-value of the i-th point of the j-th loaded data record.
ye(i,j)	evaluates to the y-value error of the i-th point of the j-th loaded data record.
xx	evaluates to the currently appropriate x(i,j)-value (in xyformel expressions).
yy	evaluates to the currently appropriate y(i,j)-value (in xyformel expressions).
ye	evaluates to the currently appropriate y(i,j)-error (in xyformel expressions).
maxx / minx	evaluates to the maximum/minimum (x) of the current data-record.
maxy / miny	evaluates to the maximum/minimum (y) of the current data-record.
centerx	evaluates the "center-of-mass" of the selected data record using the data in the fit-range [x1,x2]
widthx	evaluates the distribution rms-width of the selected data record using the data in the fit-range [x1,x2]
sumx(j,i1,i2)	evaluates to the sum of x-values from the i1-th point to the i2-th point of the j-th loaded data record.
sumy(j,i1,i2)	evaluates to the sum of y-values from the i1-th point to the i2-th point of the j-th loaded data record.

sel	evaluates to the last selected data record number.
nsel	evaluates to the number of selected data records (length of selected list).
isel(i)	evaluates to the data record number of the i-th selected list item.
sc(i)	evaluates to the data-record address (number in list) for data with numor=i. (inverse of num(j))
th_par(i,j)	evaluates to the value of the i-th parameter of the j-th activated theory instance.
th_err(i,j)	evaluates to the error of the i-th parameter of the j-th activated theory instance.
f_parnam(i)	evaluates to the value of parameter f_parnam of the i-th data-record.
f_parnam	evaluates to the parameter of the currently considered data record (e.g. in xyformel expressions)..

Automatically Created Variables

Main

ssq0	chi-squared from last fit / thc.
t_parnam	(fit) parameter of theory **only unique if there is just one theory instance** with this parameter name.
e.t_parnam	evaluates to the error of t_parnam.

Theories create a number of user-variables and/or f_parameters, see there to get the corresponding descriptions.

Interaction with the system

History

history **hist**	shows history of last 20 commands writes them to file history
"_???" underscore	if the command has an underscore as first character the last command starting with the same characters is redone immediately.
hist clear	clears internal history list
	to create a makro from history: echo makro > newmakro : creates a file newmakro with `first line „makro‟` histo clear :clear history .... : do something histo : shows history and writes it without line numbers to file history cat history >> newmakro :appends history to newmakro ..... newmakro : use newmakro whenever you need

Path

You can give presets in the „initdatr“ makro somewhere in your datreat makro path.

path	shows path definitions
macrpath	<newpath> change to newpath defailt "./" additional path to search fpr makros
datapath	<newpath> change to newpath defailt "./" path to search for input `datafiles`

savepath	<newpath> change to newpath defailt "./" path where something is stored

Active working directory

cd	<path> changes active working directory. ".." is working
sys	a system command

Collection of Standard and Example makros

cth obsolete but there

Editing of theory parameters:

makro
al
cms emacs lastth
acl

Programming

Installation

Prerequesities

Intel Fortran Compiler above 8.0 (needed Fortran 90)
LAPACK -- Linear Algebra PACKage
from http://www.netlib.org/lapack/
BLAS (Basic Linear Algebra Subprograms)
from http://www.netlib.org/blas/
should be installed on your system
(if you have problems with blas (missing functions) ask your system administrator to install from the original package.
In lapack is a subset of blas included but it is only a subset)
Graphical output
xmgrace should be installed
http://plasma-gate.weizmann.ac.il/Grace/
other libraries are included in datreat

Compile:

- Datreat is a local operating program -> userspace you dont need root for datreat itself.
- uncompress the tar.gz or your source file
- In datreat main directory you find the "Makefile", type in a shell
>>make
- If you have a $HOME/bin directory a executable script will be placed there, which calls datreat from the correct place, if bin is in your PATH variable you can call it from everywhere..
- Afterwards you can reduce the used space on disc with "make clean".
To delete all compiled files use "make cleanall" This is necessary after release of a new version or if you have problems with changes of names
- **Test:** After compile start with datreat
change dir with "cd test"
start macro "testmac" as example macro

make <Options>"
Options:" builds datreat
#update show new/changed files on nse/local/datreat Source
#do_update synchronize local files with nse datreat only update of the original files
clean deletes object files executables were not touched
cleanall deletes everything exept source, for a new start
cleanlibs cleans only lib libs (if compiler changed) or new theories were installed/written
help shows this "
distribution make a tar.gz to distribute to others with pure source files and documentation

Linking theories

The theories section is or should be a **standalone library** which can , in near future , be used without datreat. The corresponding theos Makefile is within src/theos . theories are linked automatically with a limitation to 80 theories. Please select your needed theories and move the unneeded theories to another folder. The old version theories in f77 are still working. Please use only free available non-commercial libraries.
Naming convention is: th_give_a_usefull_descriptive_name.f90
Additional helper function :
without_a_th_in_front_and_a_descriptive_name.f90
It is planned to make it possible to use also C instead of Fortran90.
Perhaps at some point we will use Python as command interpreter with the old Fortran theories. But till now this is only my plan --Ralf

Creating New Theories

In utilities a tiny program: **th_template_generator** is supplied (src/UTILITIES).
Using this the salient names parameters and variables can be entered and converted to a fortran template that conatins all the theory framework.
Subroutines and more compuations can be included by editing.
The final theory shall be copied from out.f90 to src/theos/th_theoryname.f90 and activated by make.
Example
###
th_template_generator < template1
will create out.f90 from

Input template1:

```
#THEORY thtester
        a test comment to check the th_template generator
        and another line for it.
        With complicated formulae.
#CITE
        ref. Plisch und Plum in Grimms Maerchen
#PARAMETERS         (allow 8 chars for name, start only beyond 8 with
default)
        ampli            ! prefactor
        scope            ! scope scope scope
        knurrrer         ! a knurrer
#RECIN-PARAMETERS    (allow 8 chars for name, start only beyond 8 with
default)
        q           0.1   ! Q-VALUE
        temp        300.0 ! Temperatur der Probe
#RECOUT-PARAMETERS
        qq               ! square of q
#VARIABLES
```

```
     double precision :: tau
     integer          :: myint
#IMPLEMENTATION
     tau = x
     qq  = q
#SUBROUTINES
  function s2(q)
    double precision :: s2, q
    s2 = 1
  end function s2
#END
```

The created .f90 th_template:

out.f90:

```
---------
FUNCTION th_thtester(x, pa, thnam, parnam, npar,ini, nopar
,params,napar,mbuf)
!=========================================================================
======
!  a test comment to check the th_template generator and another line for
it. With complicated formulae.
!  ref. Plisch und Plum in Grimms Maerchen
      use theory_description
      implicit none
      real    :: th_thtester
      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 :: scope      ! scope scope scope
     double precision :: knurrrer   ! a knurrer
! the recin parameter representation
     double precision :: q          ! Q-VALUE
     double precision :: temp       ! Temperatur der Probe
! the reout parameter representation
     double precision :: qq         ! square of q

     double precision :: tau
     integer          :: myint
!
! ----- initialisation -----
    IF (ini.eq.0) then
       thnam = 'thtester'
       nparx =        3
       IF (npar.lt.nparx) then
           WRITE (6,*)' theory: ',thnam,' no of parametrs=',nparx,' exceeds
current max. = ',npar
          th_thtester = 0
          RETURN
       ENDIF
       npar = nparx
! >>>>> describe theory with >>>>>>>
       idesc = next_th_desc()
       th_identifier(idesc)   = thnam
       th_explanation(idesc)  = " a test comment to check the th_template
generator and another line for it. With complicated formulae."
       th_citation(idesc)     = " ref. Plisch und Plum in Grimms Maerchen"
!       --------------> set the parameter names --->
        parnam ( 1) = 'ampli    '  ! prefactor
        parnam ( 2) = 'scope    '  ! scope scope scope
        parnam ( 3) = 'knurrrer'  ! a knurrer
! >>>>> describe parameters >>>>>>>
        th_param_desc( 1,idesc) = "prefactor" !//cr//parspace//&
        th_param_desc( 2,idesc) = "scope scope scope" !//cr//parspace//&
        th_param_desc( 3,idesc) = "a knurrer" !//cr//parspace//&
! >>>>> describe record parameters used >>>>>>>
        th_file_param(:,idesc) = " "
        th_file_param(  1,idesc) = "q        > Q-VALUE"
        th_file_param(  2,idesc) = "temp     > Temperatur der Probe"
! >>>>> describe record parameters creaqted by this theory >>>>>>>
        th_out_param(:,idesc)  = " "
        th_out_param(  1,idesc) = "qq       > square of q"
!
        th_thtester = 0.0

        RETURN
     ENDIF
!
! ---- transfer parameters -----
      ampli    =      pa( 1)
      scope    =      pa( 2)
      knurrrer =      pa( 3)
! ---- extract parameters that are contained in the present record under
consideration by fit or thc ---
      iadda = actual_record_address()
! >>> extract: Q-VALUE
      xh =     0.1
      call parget('q       ',xh,iadda,ier)
      q        = xh
! >>> extract: Temperatur der Probe
      xh =     300.0
      call parget('temp    ',xh,iadda,ier)
      temp     = xh
!
! ------------------------------------------------------------------
! ----------------------- implementation ---------------------------
! ------------------------------------------------------------------
!
     tau = x
     qq  = q
     th_thtester = ! INSERT RESULTING VALUE OF TH EVALUATION HERE

! ---- writing computed parameters to the record >>>
      call parset('qq      ',sngl(qq),iadda,ier)

 CONTAINS

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

  function s2(q)
    double precision :: s2, q
    s2 = 1
  end function s2
 end function th_thtester
```

##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##
##

```
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
######################################################################
```

```
############################################################################
############################################################################
############################################################################
############################################################################
############################################################################
############################################################################
############################################################################
############################################################################
############################################################################
############################################################################
```

Creating New Commands

????? If you want??

Available Theories

Use theos
more is coming soon

OLD datreat help

```
***** datreat information *****
symbols:
<,> are used to mark a datreat-command (usually the short form)
[,] mark optional parameters
commands available: <command/synonym/synonym..>
<activate/ac> : activates a theory
<aclast/acl> : reactivates the last theory used
<activlst/al> : lists active theories & current parameters
<chgthpar> : change single theory parameter value or scale
<label> : supply a theory parameter with a label
<couple> : install a coupling of a
theory parameter with a label
<arit> : allows arithmetic operations on your data
<combine> : interpolates and combines data from
several datasets onto one raster
<clearsel/cs> : clears list of selected datarecords
<cms> : executes a cms command (xedit,list,...)
<desactiv/dac> : deactivates all theories
<dir> : lists all loaded and generated datarecords
<dsl> : lists all selected datarecords
<edit> : allows editing of a specified datarecord
<clip> : remove points (by numer or error-limit)
<save> : allows saving of a specified datarecord
<msave> : multiple data-record save (all selected+fits)
<fit> : starts a fit of the selected data with the
activated theory
<yfitform> : enters a fitformula for eval-theory !
<open> : opens a file for the collection of
parameters(fitparametres) vs parameters(f.
<write> : writes the actual parameter(fitp.) into
the opened file
<close> : close the opened parameter collection file
<help/?> : gives infos
<input/in> : reads a datafile
<invers> : converts i vs q-data to i-1 vs q**2
<iout> : sets output-level
<mirror/m> : mirrors the data
<numorpls> : sets the increment of scan-numbers when
the data is treated. (default 10000)
<plot/p> : plots selected datarecords
<plot0/p0> : allows to set plot-parameters. no plot!
<purge> : erases specified datarecords from buffer
<q-conv/qc> : converts channel-numbers to q
<fun> : treat x and/or y -values by some functions
<funfun> : x-formula and y-formula from file fromdat
<fft> : experimental small angle multiple scatt.
<quit/q> : leaves the program
<sel> : selects specified datarecords
<sym> : makes data symmetric
<spline> : generates spline coefficients smooth data
<des> : infinte slit height desmearing
<thc> : computes data according to activated theo-
ries (no fit !)
<theos> : lists all theories available
<title/tit> : defines a title for the plots
<rename> : renames axis of selected items
<putpar> : puts a parameter into the list of selected
<zero/z> : clears databuffer
<set> : set/create a user-defined variable
<clr> : remove user-defined variables
```

```
<setdeg> : set angle-units to degree
<setrad> : set angle-units to rad
<??> : display the value of a formula
<if> : if-construct for makros
<goto> : goto construct for makros
<vars?> : display all userdefined variables
<ref> : reference to internal data.
adressing a specific datarecord:
there are two ways to access a specific set of data: either use
the
linenumber of the databuffer-directory or use the 'scan'-code of
your
data, which is defined in your datafile or generated by some
operations
of the program. nearly all of the commands require the keyword
'sc',
when you want to use the scan-code.
makro-facilities: you can define a makro containing a series of
com-
mands which may use parameters. it is possible
to call makros recursively!!
to generate a makro use the <cms> xedit-command.
the first line must be:
makro [par1] [par2] [...]
now you can simply type your commands - one per line!
to use the parameters simply type the parameter-name.
example: <plot> sc par1 xmax par2 ymax 60
!!!> makros allow for the use of a few basic like
programming features as set, if and goto.
any numerical parameters may be a formula.
to identify a parameter as number or formula
the parameterexpression has to begin with
one of the following characters: '(+-.0..9'.
there are two standard makros, which, of course, can
be altered to suit your purposes: doit fname scannr
and cth. the first makro requires a filename and its
scancode as inputs and will read the file, find the
mirror-axes, symmetrize and convert channel-numbers
to q. the second makro calls xedit with the last
activated theories and parameters. this allows easy
theory-parameter-changes.
*
? ------- general remarks --------
to execute datreat simply type : datreat
after a while you will see a list of all available theories and
then
the request-prompt: '----->'. always when this symbol appears, you
can type a command. datreat will try to execute the command or -
if
there is no such command - it will look for a makro with that name.
if there is no command and no makro with that name, no action will
take place.
a command line has the following syntax:
<cmd> parname1 value1 [value2] [..] [parname2] [value1] [...]
cmd is the command-name, parameters are adressed by names and
seperated with a blank. with certain parameters you can set more
than one value. these have to be seperated by blanks, too.
usually every parameter once defined is kept until it is
explicitly
changed. one important exeption from this rule is the list of
selec-
ted datarecords. <m>,<sym> and <qc> allow only one selected item.
if
```

```
there is more than one in the list, only the first datarecord is
used!
the maximum no of names and of numerical values that may be given
at
once is 20, the maximum length of an input line is 80 characters.
one
input line may contain several commands seperated by ';'.
once you activated a theory and fitted your data, the fit-curve
will
automatically be plotted. you can recognize a fit-curve in the
data-
buffer (use <dir> to see) by a negative scan-number. if you don't
want
the fit further displayed, use <purge> to erase it.
------- additional remarks -------
new theoretical curves for fitting may be programmed by use of one
of
the subroutines thx1 ... thxn as templates (near the end of the
listing). if you look at the predefined theories, you will
immediately
see, what you have to do.
to save a generated datarecord, use the <edit>-command. when you
enter
the fullscreen-editor, simply type 'file file xxxx' into the
command-
line, where xxxx is the name, you want to give your file.
*
*
* ------- detailed explanations -------
ac <ac> theoryname [par1] [scale1] [par2] [scale2] [...]
or <ac> theoryname parname par1 [scale1] [parname...]
or <ac> theoryname multiply parname par1 [scale1] [parname...]
the theorynames can be listed with the <al>-command.
the number of parameters to be passed depends on the number you
defined for this theory (see general remarks).
the scale is used in the fit-procedure. for sensible results it
should be ten to the power of the corresponding parameter.
you can activate more than one theory by calling <ac> several
times. all activated theories will be added or multiplied
respectively. i.e. if the multiply flag is given the
corresponding theory result is multiplied with the
result from the preceeding theories (sum or product)
instead of beeing added.
acl <acl>
theories that were activated before are reactivated with the
parameters as stored in file lastth. this file is rewritten
every times when <al> (list activated theories) is executed
(this takes also place during the fit-procedure). you can edit
file lastth using the <cms> x-command.
typing of multiply (blank separated) behind a theory name
flags this as multiplicative theory, the result of which
is multiplied with the result from the previous theories
to yield the new (intermediate) result.
the command sequence al, <edit>, acl is combined in the
cth makro, which thereby provides a tool for easy change
of theory(fit)-parameters.
al <al>
all activated theories and their actual parameters are listed.
this command rewrites the file lastth.

chgthpar <chgthpar> theoryname [number] parametername [par
new_value] [scale new_value]
```

changes the parameter given by parametername in the specified theory to
a new value and/or the scale of the parameter.
par and scale may be specified both or individually.
If the theoryname is not unique (e.g. the same theory is active several times)
the optional <number> specifies the (sequence) number of it in the list of
activated theories.
label <label> theoryname [which occurence] parametername label
supplies a theory parameter with a label. the label may contain
up to 4 characters, it may not be a number nor may its first
character be a number. the label may be referenced if linear
couplings of parameters are to be employed during a fit.
see command couple.
cth <cth>
is a standard makro to faciliate the changing of theory-
(fit) parameters. it loads the actual parameter set into
the editor, thereby allows for editing and rereads the
(changed) parameters after filing of the editor input.
couple <couple> theoryname [which occurence] parametername label factor
installs a coupling of a labeled parameter onto a parameter
quoted in this command. labels may be assigned to parameters
by the label command. the whole procedure may also be done
by editing the fitparameter-file.
see command label.
arit <arit> [normflag] [typ] [sc scan1 [scan2 ...]] [f1 factor1] [f2 factor2] [to scan]
this command requires two selected scans; these can be selected
in the command with the sc-parameter.
If sc is not give arit expects two curves to be selected,
they are used as source in the seqence they were selected.
normflag must have one of
the values 'norm' or 'nonorm' - if not specified, norm is de-
faulted.

typ may be div or mult (if not give add is assumed)

the command performs arithmetic operations with your data:
factor1 * scan1-data + factor2 * scan2-data is written into
the destination-scan datarecord. if not specified by to, the
destination is defaulted.
the norm-option normalises by the monitor-parameter given in
your data-file. if this parameter is not specified, the program
will give a message.

errors are preserved
combine <combine> [sc scan1 [scan2 ...]] [raster xstart dx n] [to scan]
this command requires two or more selected scans;
these may be selected in the command with the sc-parameter.
the data are collected on a new scan, which has x-values
that are created by the raster directive. y-values are
generated by interpolation from the input data sets.
if the input data sets are provided with a monitor
parameter, data and monitor values are summed for each
overlapping channel. finally all data are normalized
to the summed monitor values. if no monitor-data are
given, 1 is assumed. errors are evaluated using
the 1/sqrt(n) ansatz. x-points that are not present
in the set of input points are set to zero.

cs <cs>
this command clears the list of selected data-records.
cms <cms> cms-command
with <cms> it is possible to use most of the cms-commands, especially the xeditor. when a cms-command is executed, it will return to the program.
dac <dac> [number1] [number2 ...]
the theories are deactivated according to their number of acti-vation. if you use <dac> without a parameter, all theories are deactivated.
you can reactivate these theories by <acl>.
dir <dir> [<clength> value]
this command shows you all the datarecords presently loaded. it is recommended to press the pa2-button before executing this command.
The optional parameter <clength> followed by a value=1..80 set the number of characters of the comment to be displayed.
dsl <dsl>
lists all selected datarecords.
edit <edit> [number] [sc scan]
with <edit> you can change the datarecords that are loaded or generated by the program. this command will call the xeditor.
<edit> also gives you the possibility to save these records on your disk: you only have to specifiy a filename when leaving the xeditor by file or ffile. if you only type file, your datarecord is stored in file datbuf and your original file (if existing) will not be affected.
clip <clip> [[<from> n1] [<to> n2]] [<last> n] [<errmax> val] [<rel>]
removes points from n1 to n2 or the last n points or all points with error > errmax wit option rel relative error is checked

save <save> to filename SEE ALSO: msave
<save> dirnum to filename
<save> n name to filename
<save> sc numor to filename
<save>
store a datarecord on disk permanently with the name :
file filename a. if no item is specified the first datarecord in the selection table will be saved. by giving dirnum the dirnum-th record as shown in the dir-list may be saved.
by specifying n name a file with the internal name <name> will be save and by sc <numor> a file with the corresponding numor will be saved. if no destination <filename> is given the data will be saved onto file lastsave.

msave <msave> filename

store a datarecords on disk permanently with the name :
file filename a.
Records that are selected are stored (associated fit results are also stored).
<open> <open> filename parnam1 [no.theo1] parnam2 [no.theo2]
will prepare a file named file <filename> a for output.
this file will be filled with the values of the parameters with names <parnam1> as x-values and <parnam2> as y-values.
the parameters may be taken from the parameter-block associated with the currently selected file (1st file of selection list) or from the parameters associated with the currently activated theory setting, in the latter case it may be specified by giving <no.theo> that the parameter is to be taken from

```
the <no.theo>-th theory that is activated.
<write> <write>
write one line into the opened parameter file according to the
actual selection and/or theory status.
<close> <close>
close the open parameter collecting file.
<fun> <fun> optionx optiony
treat all selected files by applying some functions to the x
and/or y-values.
option : op or np : item ---> item or item ---> new item
optionx: x : x ---> x
log(x) : ln(x) ---> x
exp(x) : exp(x) ---> x
x**2 : x**2 ---> x
x* [f1] : x* f1 ---> x
x+ [s1] : x+ s1 ---> x
sqrt(x) : sqrt(x)---> x
rouse : q**2*sqrt(wl4*x) --> x (q from parameter, wl4 from command
or parameter)
zimm : (q**3*kT/(6*pi*eta)*x)**(2/3) --> x (q, temp, eta_solv from
parameter)
(optional eta from command)
optionx: y : y ---> y
log(y) : ln(y) ---> y
exp(y) : exp(y) ---> y
y**2 : y**2 ---> y
y* [f1] : y* f1 ---> y
y+ [s1] : y+ s1 ---> y
sqrt(y) : sqrt(y)---> y
deff : y/(x**2) -> y

with appropriate errors
funfun x-values and y-values of all selected items are treated
by the formula lines that are given in file formdat a
as first and second line. the lines may not contain
any blanks within the formula. they may be written
as usual formulas, the x-values are to be referred
by xx, the y-values by yy.
exponentiation is indicated by ^ eg. xx^2, xx may
not be negative!
log is ln.
functions available: sin,cos,tan,asin,acos,atan,ln,exp,
sqrt,int,abs
the trig. function operate with rad or degree, which may
be switched by the setrad or setdeg command.
the output may be reduced by iout -5 .
fit <fit> [sc scan1 [scan2 ...]]
[x1 startvalue] [x2 endvalue] .. and other thc opts.
[auto]
[maxit max._number_of_iterations]
[ngood est_no. of valid digits in the theory]
[maxstep max._step_between_to_values]
[trustreg trustregion]
[maxfn max_fun_calls]
[relerr]
[abserr] (default)
[wrtfit]
[nowrtfit] (default)
[errors]
[map mgrid] [div ndiv] go
[go]
with the <fit>-command the parameters of the activated theories
```

are fitted to your selected data (more than one data-record may be selected !)
x1 specifies the lower x-value of the fit-intervall, x2 the upper value.
if auto is given x1 and x2 values are ignored and all data are respected in the fit. the next time x1 or x2 is given again a corresponding limit is again established.
with maxfn you can set the maximum number of function calls that take place during the fit-procedure.
the (volatile) errors option causes the computation of statictical errors for the fit-parameters, provided errors for the data are given. the errors option must be combined with the go option.
the go-option must be explicitely stated every time the <fit>-command is used. if not given, <fit> will only find the para-meters for the activated theory. the curve-fitting can then be done by once more typing <fit> without any parameters.
convolution with a resolution function may be performed.
for details see thc !

fit and thc add a number of paramters to the resulting curve buffers
these may be addressed in other functions or viewed in the plot.
These parameters describe the theory parameters and errors.
In the moment we are still restricted to 8 characters for the names therefore the theory parameters are coded as follows:

first 2 characters = first two charactres of theory name
3rd character = one digit (1..9) telling the number of the theory in the activation list.
4..8 character = 1..5 charcter of the th-parameter name

the corresponding errors are build by puuting 'e' in front of the coded parameter name and truncating the result to 8 chars.

See --> plot noparplo to get a plot with a noncrowded parameter part
those parameters may be used in expressions.

OPTION: map
----
if --> fit map writes files map.xy and map.ssq that contain the landscape of ssq vs paramateters (max. no of free pars.=3)
The parameters varied are those with nonzero fitscale.
The stepwidth used is fitscale/div (div may be given as parameter in the command line, default is 20).
Map must be followed by the number of grid points in each direction.
Use ssq.gli to display the resulting files (2D).
If map is given NO fitting will be performed !
The commandline must be finished by the parameter: go

yfitform enter a formula that may be used as fitting function by invoking eval as theory. the formula is entered as a pseudo-resline after a ;
```
yfitform ;p(1)+xx*p(2)+xx*xx*p(3)
```
where xx is the independent variable and p(i) are the parameters as given with eval.
! this type of function evaluation is quite inefficient !
however it may be used for a quick check of simple ideas

without changing the program code.
help with the <help>-command you get these information or, if you use
<help> <command>, you will get specific information about one datreat-command.
in <in> filename
this command allows you to read a datarecord from disk. the data must be stored in the following format:
(the uppercaes words are keywords !)

```
line 1: arbitrary comments
line 2: filename y-label vs x-label scancode
line 3: {must be empty !}
line 4: parameters
line 5: par1 value1
.
.
line x: parxy valuexy {10 is the maximum number of parameters}
{let's call it line 10}
line 11: {must be empty !}
line 12: values
line 13ff.: [x] x-value [y] y-value [e] error
or x x-value1 x-value2 x-value3 ...
y y-value1 y-value2 y-value3 ...
e error1 error2 error3 .....
{always one line x-values and one line y-values !}
line y: {must be empty !}
line y+1: #eod
```

there are two files that can be linked to your data by the get-command of the xeditor: file kopf must be linked to the beginning of your data and provides the header, file eod gives an empty line and the #eod-keyword.
invers <invers> [bkgr background]
this command converts the first selected datarecord to i-1 vs q**2 - format. the original data must be i vs q.
with bkgr you can subtract a constant background-level before inverting.
iout <iout> [number]
with <iout> you can set your output-level.
m <m> est_mid x1 x2
the <m>-command generates a mirror-image of a selected data-record (it must be selected before !).
in est_mid you estimate the center-x.
with x1 and x2 you specifiy the range that shall be used to determine the center.
numorplos <numorpls> offset
this command should only be used in the beginning of a session, it sets the offset, that is added to the scan-code, when any operation is performed by the program.
the default-value is 10000. this means, that usually you can identify your data as follows:

```
00xxx : raw-data i(ntensity) vs kanal
10xxx : mirror-data dto. {see <m>}
20xxx : symmetric-data sym-i vs kanal {see <sym>}
30xxx : converted-data i vs q {see <qc>}
40xxx : "inverted"-data i-1 vs q**2 {see <invers>}
-xxxxx : fit-curve {see <fit>}
```

to keap your head clear of garbage, you should use the cycle-numbers of your experiment as scan-numbers. to distinguish "horizontal"-data from "vertical"-data use 1xxx for horizontal and 0xxx for vertical data. if this definition becomes popular, it will be easier to exchange data with other users.
p <p> [sc scan1 [scan2 ...]]
[fsc fitscan1 [fitscan2 ...]]

```
[xmin minimum_x_value]
[xmax maximum_x_value]
[ymin min_y] [ymax max_y]
[framx length_of_x_axes] [framy length_of_y_axes]
[frlux beginning_of_x_axes] [frluy beg_of_y_axes]
[symb symbol1 [symbol2 ...]]
[icolo color1 [color2 ...]]
[o... option-specifier]
[fitflag]
[textflag]
[show_parameter_flag]
[txsize textsize]
[font fontnumber]
[legsize legend_size]
[legx relative_x_position_of_legend]
[legy rel_y_pos_of_legend]
[errplo / noerrplo]
[parplo / noparplo]
[log_x / lin_x]
[log_y / lin_y]
```

the <plot>-command is one of the most powerful commands of this program. once you know how to use the parameters, you can do nearly every plot you like. after execution you are asked, whether you want to print the plot elsewhere.
with sc you can select certain data-records as usually.
fsc may be used to select several fit-curves. if you performed a fitting, this fit is automatically selected. after selecting other curves by sc or a <sel>-command, you have to specify the fit-curves that shall be plotted. this is especially useful, when you want to have different fits for different intervals.
attention: you must give the corresponding positive scan-number to select a fit! this allows it to select fit-curves automatically in a makro.
choose the borders of your plot with xmin, xmax, ymin and ymax.
the frxxx-parameters can be used to size and locate your plot.
symb chooses the symbols that are used to represent a data-point in the plot. the first number will be used for the first selected item, the second for the second, ... for detailed information see the gr-software-handbook or (if you read a print of this guide to datreat) the appendix.
icolo chooses the colors for your plot. only selected data-records can be colored. the codes are:

0	black (default for all curves)
1	red
2	blue
3	green
4	purple
5	yellow
6	zyan.

o... specifies the options for your axes. for detailed information see the gr-software-handbook (graxs-subprogram). the format is (example): ox x=1 {shows x-axes below your graph}.
{2 puts the axes above and 3 on both sides. -1/-2/-3 shows a grid}
fitflag can be fits or nofits and says if fit-curves are to be presented.
textflag is text or notext and is used to show title and legend or not.
show_parameter_flag is parplo or noparplo and determines whether (when text is set) all parameters are shown or only the legend.
txsize determines the size of the text on the axes and the title.
with font you can choose a graphic font. detailed information is

```
again given in the gr-software-handbook or in the appendix.
legsize sets the size in which the legend-text and parameters
are plotted.
legx and legy can be used to move the legend elsewhere, when
it would cover your data (or if you prefer a different place).
{if you want to set plot-parameters only, use <plot0>}
errplo / noerrplo activates/desactivates plotting of
errorbars

parplo / noparplo may be use to switch (on/off) the writing
of the parameters associated to each file at the right border
of the plot. If noparplo is in effect one may select a number
of parameters that is plotted anyway by specifying the name
of those parameters in the plot command (volatile).

log_x, log_y yields log-scaling of the x/y-axes. Use
lin_x, lin_y to go back to linear scaling.
p0 <p0> [... see p(lot) ... ]
<plot0> sets parameters for following plots, but does not plot
anything. this command is especially useful for defining an
initial setting according to your taste. the parameters are
described in <p(lot)>.
purge <purge> [dir#1 [dir#2 ...]] | [all]
this command is used to delete datarecords from your directory.
use the <dir>-command before to get the right numbers.
all selections will be removed.
qc <qc>
<qc> converts the symmetric i vs kanal -data into i vs q. this
command operates on the first selected item.
q <q>
leaves the program. if you send plots to external printing-media
it may be useful to quit datreat once a while, because the plot
is only started when the program is quitted.
sel <sel> nadd1 nadd2 ... [fit+]
<sel> add nadd3 nadd4 ...
<sel> sc scan1 [scan2 ...]
<sel> sc+ scan-n+1 ...
<sel> fits
the <sel>-command can be used to select certain datarecords for
further operations. The same effect is achieved when using sc
as keyword in several commands.
If direct addressing is used, the OPTION add allows to keep NEW
the previous selction and to add further recods to the selection.
when an operation generates a new datarecord this new record is
automatically selected.

Option: fit+ adds the corresponding fits to the selection NEW
given as list of sequence numbers NEW


parameter sc selects scans according to their numors
sc+ adds numor selections to present list
only a number list selects entries according to their
sequence number in the dir-list
OPTION fits searches for old fitted items and selects them
as fits of the selected items if the numors do match.
sym <sym>
this command calculates the mean-values of the data on the left
and right side of the center and creates a new datarecord with
only one side.
spline <spline> [auto]/[noauto] [nneu nneu] [smpar smpar]
spline approximation of scattered data on the selected scan.
the smoothing parameter smpar is automatically determined if
```

the option auto is given. otherwise (noauto) smpar must be
specified (by try and error, look at the plot!). smpar will
influence the degree of smoothing. nneu specifies the number
of points (density) the smoothed synthetic new scan should
get within the x-range of the original scan.
the spline polynomial coefficients of the last spline call
are stored in an internal common block for subsequent use,
i.e. in des or fft.
des des [qmax qmax] [nneu nneu]
infinite slit height desmearing up to a q-value of qmax.
the new dataset will contain nneu points.
the data to be desmeared must be splined immediately before
des is invoked.
thc <thc> [n] n [x[c]1 x1] [x[c]2 x2] [auto] [convolute] [off]
with <thc> you can recalculate fit-data. n is the number
of points that are used for calculation; 0 means: take value
from last fit.
NEGATIVE n will cause an even distribution x-values on a
logarithmic scale, if x1=0 or x2=0 or x1>=x2 this option
is ignored.
if auto is given x1 and x2 values are ignored and all data are
respected in the calcul. the next time x1 or x2 is given again
a corresponding limit is again established.
if convolute the external routine datconv is taken to
convolute the calculated data before leaving thc.
the number of points and range before convolution is
specified by n, xc1 and xc2 (instead of n, x1 and x2 !).
the result gets the x-values of the selected template(s).
the x-ranges before and after convolution may be different
depending on the transformation that is implicit in the
convolution kernel (e.g. q-values ---> scattering angles).
usually x belongs is a physical varaible before and an
experiment varaiable after convolution. xc1/2 refer to the
physical variable range. the selected item has to
be in experiment space.
all convolution settings are also valid during fit.
convolution is valid until convolution off is specified.
any parameters needed to specify the convolution kernel
are to be given as parameters in the selected data-item,
they are extracted by datconv using the getpar routine.
theos <theos>
if you want to know what theories are available in your version
of datreat, simply type theos and you will see.
tit <tit> titlestring
with the <title>-command you can define a title that is dis-
played with your plots if the text-option is set.
rename <rename> [xaxis <new-string>] [yaxis <new-string>] [name
new]
the xaxis and/or yaxis and/or id-names of all selected items
are replaced by the strings given here.
putpar <putpar> <parname> <value>
a parameter with name <parname> will be created and set
to the value <value> in all selected items. if the
parameter already exists, its value is updated.
z <z>
this command is used to clear (<zero>) the buffer. no datare-
cords can be selected afterwards. the <dir>-command will show
an empty list.
makro {see general information also}
there are some standard-makros which simplify the use of this
program:
doit filename scancode loads the specified data-record
and does everything up to the

```
q-conversion automatically.
cth calls the xeditor to edit the
last activated theorie-para-
meters and then activates this
theorie.
plnorm sets standard values for plot
i vs q.
plzimm sets standard values for plot
i-1 vs q**2.
set set/create a user-defined variable.
usage is set <name> <value> [ <name> <value> ...
variables may be listed by the command: vars?
some commands transfer their parameters to the
uservariable-stack, so that these variables can
be referred to.
clr remove user-defined variables
usage clr all removes all uservariables.
clr <name1> <name2> ... <namex> removes the
variables with <name1> .. <namex>
setdeg set angle-units to degree for the commandline
trigonometric functions.
setrad set angle-units to rad for the commandline
trigonometric functions.
?? display the value of a formula.
?? <expression>
displays the evaluated value of <expression>
if if-construct for makros
usage if <expression1> = <expression2> then <commandline>
or if <expression1> > <expression2> then <commandline>
or if <expression1> < <expression2> then <commandline>
or if <expression1> <= <expression2> then <commandline>
or if <expression1> >= <expression2> then <commandline>
or if <expression1> <> <expression2> then <commandline>
goto goto construct for makros
usage goto :label
the label :label has to start with ':', the
line containing the label may not contain any
other commands.
vars lists all userdefined variables
ref internal data may be referenced within expressions.
xx --> current x-value (funfun)
yy --> current y-value (funfun)
xv(i,j) --> j-th x-value of i-th databuffer
yv(i,j) --> j-th y-value of i-th databuffer
ye(i,j) --> j-th yerror-value of i-th databuffer
sumx(i,j1,j2)
sumy(i,j1,j2) --> sums
sel --> first selected buffer
nbuf --> number of loaded buffers
maxx
maxy
minx
miny --> max and min of the first selected curve
iout --> current outputlevel
<parname> --> parametervalue of selected curve
<parname(n)-> parametervalue curve n
<fitpar> --> name of fitparameter, only the
last of equally named fit-parameters
is accessible
th_par(ip,it) --> value of the ip-th parameter of the NEW
it-th activated theory
th_err(ip,it) --> error of the ip-th parameter of the NEW
it-th activated theory
```

```
Michael Monkenbusch,
IFF, Forschungszentrum Juelich, D-52428 Juelich
Tel.: +49-2461-61-4314/2799
Fax.: +49-2461-61-2610
E-m.: m.monkenbusch@fz-juelich.de
```

new

sel all [parna] [val] band [val]
sel next [parna] …..
sel add +…..
theory kohl with gaussian resolution parameters for folding
(makros resit …)

theor

BSS-Data Converter

convert_dat2.f

NEW DEVELOPMENT: Thinktank

create a new entry in the theory header line (same level as 'multiply') to specify a parameter range in which the theory is evaluated, outside a zero contribution is added. The parameter value stems from the parameter block of the actual data-record (iadda).
E.g.: theory XYZ [multiply] [range <parname> <val-min> <val-max>]
default: full range.

Programming Interface by multiple include files/preprocessor (see McStas as example).
Intro-section
Closing-section
???
Preprocessor/ vs Description