PC-Mos II

Version 1.0

Manual

and

Program Documentation

Author: Gerhard Große

September 1993

PC-Mos II, Version 1.0
Manual and Program Documentation
First edition, September 1993
Written by Gerhard Große

This manual has been typeset with TEX

© 1993 by FAST ComTec / Gerhard Große
Grünwalder Weg 28
D-82041 Oberhaching
Germany

Trademark Acknowledgements:
The following are trademarks and registered trademarks of the companies listed:

IBM, AT, PS/2: Industrial Business Machines Corporation
Hercules: Hercules Computer Technology, Inc.
MS-DOS: Microsoft Corporation
INTEL: Intel Corporation
EPSON: Seiko Epson Corporation
Borland C++: Borland International, Inc.
TEX: American Mathematical Society

Contents

1 Introduction

1.1 About PC-Mos II

PC-Mos II is a software package for the analysis and evalutation of Mössbauer spectra. In general, the evaluation of Mössbauer data is done best by fitting a theoretically motivated curve to the measured spectrum. Such a least squares fit routine also lies in the heart of PC-Mos II. This routine is supplemented by miscellaneous options for data processing such as folding of crude data from sinusoidal or triangular Mössbauer drives, adding of two spectra, rejecting of corrupt data and many more.

The highlight of PC-Mos II is, however, its capability to fit more than one spectrum simultaneously. This means, for example, that you can measure spectra of one absorber at different velocity ranges and evaluate them without loss in the statistics. Another possible application is to exploit your knowledge of some physical property of two or more specimens being identical.

Great importance has been attached to making the program as flexible as possible. As a result the input and output of all kinds of data PC-Mos II processes is done through ASCII files, the formats of which are to a great extent programmable. Additionally there are many parameters that control the way PC-Mos II works. However, of course some technical restrictions remain:

PC-Mos II only runs on an IBM AT or PS/2 or a true compatible with an INTEL 80286 CPU (or later compatible versions), a 80287 math coprocessor (or later versions) and a fixed disk. Operating system is MS-DOS (version 3.0 or later). The graphics version requires a Hercules, EGA or VGA graphics adapter (unless you own a BGI-driver for another adapter). To compile user written theory programs (i.e., programs that provide the theoretical curves to be fitted to the spectrum) a C compiler of Borland's C++ series (Turbo C++ or Borland C++) is needed.

1.2 About this Manual

This manual is actually more than a manual. Having in mind that a scientist should know what happens to his data when he uses a computer program, I decided to include a lot of background information about the numerical algorithms applied in PC-Mos II. When it seemed appropriate, some technical information about the program structure has been included too. In order to distinguish sections containing such information from the actual manual they have been marked with the signs $\underline{\mathcal{M}}$ and $\underline{\mathcal{C}}$. (M stands for mathematical information; C is the programming language I used.)

To continue with typographic matters: Whenever this manual quotes some literal program input or output, a `teletype script style` is used. This style is also used for DOS commands, file names and similar items. A key of the computer keyboard is denoted by something like [ESC]. I.e., the phrase "Enter `Enter`" means that you should

type the letters `E`, `n`, `t`, `e` and `r`, whereas with "Enter [Enter]" I mean that you should press the [Enter] key of your keyboard.

For the most part (chapters 3 through 9) the organization of this manual parallels the menu structure of the program. This might help you to find information to a specific option. Additionally the appendix contains a tree graphic of the menu structure, in which with each option the number of the section describing it is given.

1.3 Installation and Disk Contents

To install PC-Mos II on your hard disk you only have to insert the PC-Mos II disk in your floppy drive, type `a:install` and follow the instructions. (Of course you can also use drive `b:`.) The installation program will create all necessary directories and copy all files to your hard disk.

After having run the installation program please supplement the DOS `PATH` command in your `autoexec.bat` file with the PC-Mos II directory that has been created during the installation and reboot your system to make this change come into force. Normally the name of this directory is `\pcmos`, but you could have changed this when running the installation program. In the following sections I will assume that you did not change this name.

To complete the installation, you have to connect the *hardlock* dongle that came with PC-Mos II to your printer port (parallel port LPT1, LPT2 or LPT3). This dongle is transparent to printer output, i.e. if you plug your printer to the open end of the connected hardlock, you can still use your printer in the normal way. Without this hardlock PC-Mos II will run only in a restricted mode, which is not capable of fitting and quality graphics output. However, this restricted mode might still be useful for computers dedicated only to data acquisition.

After you have run the installation program, the directory `\pcmos` will contain following files:

- `PCMOS.EXE`: the executable file of the graphics version of PC-Mos II,
- `PCTMOS.EXE`: the executable file of the text mode version of PC-Mos II,
- `MOSSETUP.EXE`: a menu controlled installation program which allows you to adjust PC-Mos II's parameters to your requirements (see appendix for more information),
- `MOSPLOT.EXE`: a menu controlled Mössbauer plot program which enables you to produce high quality plots of your fitted spectra (see appendix for more information),
- `MKTHEORY.EXE`: the PC-Mos II theory manager that must be used to compile theory programs (MkTheory requires Borland or Turbo C++ installed on your system; see appendix for more information),
- `PCMOS.CFG`: the configuration file of PC-Mos II containing the parameters adjustable with MosSetup and MkTheory,
- `PCMOSMT.???`: format files that PC-Mos II needs for input and output of data (see sect. 3.1; you can modify these files for your own purposes: see chapter 10),

- `NSING.EXE`, `NDOUB.EXE`, `NSEXT.EXE`, `NOCT.EXE`, `NTSING.EXE`, `NTDOUB.EXE`, `NTSEXT.EXE`, `NTOCT.EXE`, `MAGDIST.EXE` and `POLYNML.EXE`: ready-for-use theory programs which may serve both as examples and as analysis tools for your spectra.

The directory `\pcmos` will also contain two subdirectories:

- `\pcmos\theory` contains the source code of the enclosed theory programs,
- `\pcmos\examples` contains the files needed to go through the examples described in the next section as well as example parameter files for all enclosed theories.

1.4 Getting Started—An Example

Before going into a more formal description of PC-Mos II it is probably useful to illustrate its operation by a more or less typical example. To go through the example you first must change to the directory `\pcmos\examples` to make all necessary data available. (Type `cd \pcmos\examples` at the DOS prompt.)

To start PC-Mos II simply type `pcmos` at the DOS prompt. If PC-Mos II has been installed correctly, a graphics screen with some empty windows will appear and at the bottom of the screen a copyright statement will be displayed. (If your screen remains black, PC-Mos II probably couldn't recognize your graphics adapter correctly. You can use MosSetup to overcome this problem. See appendix for more information.)

The area at the bottom of the screen is the *message window*. Here the dialogoue between the user and PC-Mos II takes place. The blinking *cursor* indicates that PC-Mos II is waiting for input.

PC-Mos II can do very few things without having any data. So the first thing you have to do is to input some data from data files. At the top of the screen a menu line is displayed, the main menu of PC-Mos II. Now select the option `File` by pressing F and see what happens: A new menu is displayed and the text in the message window scrolls up. Mössbauer data for PC-Mos II is usually divided into two files, the first of which contains some general information about the spectrum (such as drive velocity and number of sampled channels). The second then contains the actual data. Now you will input both files to PC-Mos II: Because you want to read data from a file, now you select the option `Read` by pressing R. PC-Mos II now prompts for a file name. Please type `example.hdr` and press Return. The data from the file `example.hdr` will be read and some of the empty windows on the screen will be filled. The central window, however, is still empty, which means that PC-Mos II is still lacking of the actual Mössbauer data. So please press R again and now enter `example.asc` at the file name prompt: Now the familiar display of a not yet *folded* Mössbauer spectrum will appear.

Our goal is to fit a theoretical curve to the displayed spectrum. For this purpose PC-Mos II needs some additional information about how to calculate this theoretical curve. This information is contained in the file `example.par`. To make it available to PC-Mos II please once more select the option `Read` by pressing R and type `example.par` when PC-Mos II prompts for the file name.

Before the actual fit can start, some preparations must be made. We are done with the input of data, so first leave the *File Menu* by pressing X for `eXit`. From the

main menu now re-appearing on the top of the screen please select the option `Process`. If you look carefully at the left edge of the displayed spectrum, you will recognize a data point which is far below all others. This is the result of a not unusual error of the multi-channel-analyzer used to record the spectrum: the data in channel 0 is corrupt. The option `Reject` allows you to get rid of this data point. Select it and enter `0` when PC-Mos II prompts for a channel number.

The next step is to *fold* the spectrum, which has been measured with a sinusoidal Mössbauer drive. (Sect. 4.1 describes what it means to *fold a spectrum.*) First press P to invoke the option `FoldingPoint`, which determines the channel around which the spectrum must be folded. Then press F for `Fold`. You will recognize the changes in the display.

The last thing you have to do before the actual fit can be started is to calculate an estimate of the *baseline* of the spectrum, which is the avarage countrate in channels with no absorbtion (see sect. 5.3 for details). Please leave the *Process Menu* by pressing X and select the option `Assess` of the main menu. From the *Assess Menu* select the option `Statistic-Baseline` to calculate the baseline. (You can also use the option `Margin-Baseline`.)

You are ready to start the fit now. Please return to the main menu and press W to select the option `Weber-Fit`. The fit is started by pressing S. While the fit is running, you can monitor its progress by observing the continuous line in the graphics display approaching the data points, or—in a more quanitative way—by watching how the quantity χ^2 changes after each iteration. This quantity should approach one if the spectrum is well described by the theory curve.

When the fit is completed, PC-Mos II displays the message `Fit terminated after` x `iterations`, accompanied by some audible signal. The only thing left to do is to store the fit results in a file. This is done by selecting the option `Write` from the *File Menu* (you know how to get there?) and entering `example.res` when PC-Mos II prompts for a file name. If you want to see what results PC-Mos II has produced, you can select the option `DOS-shell` (in the *File Menu*) to return to the operating system without finishing PC-Mos II. There you can enter a command like `type example.res` to list the results on the screen. Typing `exit` at the DOS prompt takes you back to PC-Mos II.

You might be somewhat concerned about how many steps it took to fit a single spectrum. If you are, there is good news for you: All these steps can be combined in a *macro file* and then executed with one command. A macro file containing exactly the steps you have done before has already been prepared for you. To execute it press ESC, enter the command line `dofit example` followed by Return and see what happens ...

1.4 The Concept of Theory Programs

In order to understand how PC-Mos II works it is of great importance to understand the concept of *theory programs*. To provide a maximum of flexibility PC-Mos II assumes almost nothing about the theoretical curves it has to fit to a spectrum. This can only be achieved by a complete separation of the routine that calculates this curve from the main program. This routine is part of a so-called *theory program*, which I will often simply refer to by *theory*. A theory program is an executable EXE-file like any other DOS program. Whenever PC-Mos II needs a theory curve being calculated, it executes the theory program which will do this job.

In order to fit the theoretical curve calculated by a theory program to a spectrum, PC-Mos II must have a way to modify this curve. Therefore each theory program must provide some variable *parameters* that control the shape of the curve. (These parameters were part of the second file `example.par` you made PC-Mos II read in the preceding example.) Fitting a curve to a spectrum thus means for PC-Mos II to change these parameters until the coincidence of data and theoretical curve is optimum. (This will be made more quantitative in chapter 7.)

The desired flexibility is guaranteed by the fact that although many ready-for-use theories are enclosed with the PC-Mos II package, you can write your own theory programs designed to match your special requirements. You will need some basic knowledge of the programming language C to do this. More information on this item can be found in chapter 11.

2 Controlling PC-Mos II

2.1 Screen Organization

The graphics screen of PC-Mos II is essentially divided into three sections. At the top of the screen there is the menu line. A big framed graphics window follows and at the bottom five (text) lines are reserved for the message window.

The menu line lists all options that are available by one keystroke at the moment. At each displayed option one letter is underlined. By pressing the corresponding key (the *hot-key*) this option is invoked. The menu structure of PC-Mos II can be seen as a tree: An option of a menu can invoke sub-menus, whoose options might again be sub-sub-menus, and so on. A graphic visualizing this structure can be found in the appendix. One option that all menus, with the exception of the main and the fit menu, have in common is the option `eXit`. This option always takes you back to the main menu, regardless of where you are in the menu tree. The option `Quit` of the main menu finally finishes PC-Mos II and takes you back to the DOS level.

The graphics window is divided further. At the top there is place for a header normally containing some information about type and origin of the data. Below, an area follows in which the data and perhaps the theoretical curve are displayed graphically together with all captions. Further below two rows of smaller windows are sited. These windows contain some information about the state of the spectra and the program:

After the start of PC-Mos IIthe first of these windows (from left top to right bottom) contains the text `Spectrum 1: -----`. The number tells you which spectrum is selected at the moment. This is useful since PC-Mos II can manage more than one spectrum at one time. All displayed information and all options (the option `Weber-Fit` is an exception) only concern this selected spectrum. Another spectrum can be selected with the option `Choose`. The five dashes are flags—a dash means that the corresponding flag is not set. If one of the flags happens to be set, a letter is displayed instead of the dash. This is the meaning of the flags in the displayed order:

Flag	Meaning
`D`	Data (folded or not) is available.
`F`	Data is folded. This flag can only be set if also the Flag `D` is set.
`P`	Parameters are available.
`R`	Fit results are available. This flag can only be set if also the Flag `P` is set.
`S`	Simulation data is available.

The following three windows of the first row are file name windows. They contain the names of the files PC-Mos II has read from or written to most recently. From left to right the windows are assigned to the areas data, parameters and simulation data.

The lower row contains some information about the selected spectrum. If the data is not folded, from left to right: Folding point (in integral or half-integral channel numbers), followed by a letter `S` or `T` indicating sinusoidal or triangular folding mode,

then maximum velocity of the Mössbauer drive, the name of the theory program and the number of parameters. If the data is folded: Baseline, number of data points, still name of theory and still number of parameters.

Within the last five lines of the screen the dialogue between the user and PC-Mos II takes place. Like in the DOS-shell a blinking cursor indicates that PC-Mos II is waiting for some input. If PC-Mos II is waiting for a command (i.e., the choice of an option or the name of a macro file to pass control to), the prompt `>` is displayed followed by the blinking cursor. In the upper right corner the remaining free memory is displayed.

2.2 Invocation of Options and Macros

When PC-Mos II is waiting for a command, you have two alternatives to control PC-Mos II. One possibility is to rely on the user guidance of PC-Mos II and to select options by entering the underlined letters (the "hot-keys") of the menu line. Then PC-Mos II will prompt for all additional input it needs to do its job. However, you also have the possibility to control PC-Mos II with command lines. By pressing `ESC` you activate a line editor, which enables you to enter a whole command line. If you want to invoke an option this way, you have to enter the name of the option as it is spelled in the menu line. (It is sufficient to enter as many letters as needed to identify the option unambigously.) The command line may (but need not) include additional input PC-Mos II normally prompts for. These items must be separated from the option name and from each other by one or more blancs. If such an input is already given in the command line, PC-Mos II no longer prompts for it. If the command line only includes a part of the input PC-Mos II needs, the remaining input is still prompted for. On the other hand, if more input items are specified than PC-Mos II needs, a warning is issued. Normally you won't have to enter very long command lines. For this reason the line editor is rather poor: only the `Backspace`-key is provided for corrections.

Of course there is no reason to choose the command line mode, if you only want to select an option from the menu line. However, this mode provides additional possibilities, which are not available with hot-keys. In particular, you need the command line mode to invoke a macro file: Macro files are invoked by entering their name as command line, perhaps followed by some arguments. In the example of chapter 1 the name of the macro file was `dofit.pcm` (the file name extension `.pcm` was appended automatically) and `example` was an argument. When PC-Mos II begins the execution of a macro file, the message `control transfer to macro` x is output. Then PC-Mos II reads the macro file and treats its lines as command line just as described above. When a macro has been processed completely, PC-Mos II transfers control back to the user. You can recognize this by the message `control return from macro` and by the re-appearance of the blinking cursor. If you want to stop the processing of a macro file before its end is reached, you can enter `Ctrl` `Break`. Then, after the execution of the current command line, the message `Ctrl-Break detected - Press ESC` is displayed and you will regain control after you have pressed `ESC`. (If PC-Mos II is currently doing a fit itaration or a screen dump, the first `Ctrl` `Break` only aborts the current job but

not the macro processing. In this case you should press Ctrl Break a second time when above message is issued before pressing ESC.)

When PC-Mos II is parsing a command line, it first checks if the first token is the name of an option of the currently active menu. If this is true, the option is invoked and possible command line arguments are passed to that option. Otherwise PC-Mos II looks for a macro file with this name. PC-Mos II searches first in the current directory and then in all directories specified in the DOS environment variable `PATH`. (This variable is normally set by the `PATH` command in your `autoexec.bat` file.) If the search is successful, the macro is executed as described above. If the search fails, PC-Mos II issues the error message `Unknown command or macro`.

C Internally the program progress is always controlled by command lines. In the hot-key mode the command line parser only is preceeded by a kind of command line generator, which creates a line consisting of the name of the option the user selected with the hot-key. Then this line is output in the message window and is subsequently treated exactly as a command line directly entered by the user.

When you start PC-Mos II at the DOS prompt, you can also specify command line arguments. The command tail, which is the command line without the program name, is treated as normal command line of PC-Mos II. For example, if you type `pcmos dofit example` at the DOS prompt, PC-Mos II will immediatelly invoke the makro `dofit` with the argument `example`, just as if you had entered `dofit example` at the command prompt of PC-Mos II. This feature enables you to control PC-Mos II completely by DOS batch files and PC-Mos II macro files.

2.3 Writing Macro Files

As mentioned above, macro files simply consist of a sequence of PC-Mos II command lines, which are executed line by line. Like DOS batch files they are normal ASCII files and you can create and modify them with your preferred text editor.

A useful feature of macro files is that they can have arguments. Arguments are passed to macros with the command line that invokes the macro. They must be separated from the macro name and from each other by blancs. (As a consequence you cannot pass a text containing blancs as argument to a macro.) Within the macro file the argument are accessed by character sequences of the form `%`d, where d stands for a digit between 0 and 9. `%0` stands for the first command line argument, `%1` for the second, and so on. (Please note the difference to DOS batch files where `%1` denotes the first argument!) While processing a macro PC-Mos II replaces the character sequence `%`d with the text of the $(d+1)^{\text{st}}$ command line argument that has been specified when the macro has been invoked. If less than $d+1$ arguments have been specified, the replacement text is simply a blanc. An example might illustrate the mechanism of macro arguments: If you invoke a macro named `setfp.pcm` containing the line `Foldingpoint %0.5` with the command line `setfp 256`, PC-Mos II will actually encounter the instruction `Foldingpoint 256.5`.

Another useful feature of macro files is that they may be nested. This means that a command line of a macro file may result in the execution of another macro file.

When the processing of this second macro is completed, the execution of the first is continued. The number of active macro files only is restricted by the maximum number of files MS-DOS can open at one time. (You can check this number by writting a macro file that invokes itsself. If you find it to small, you can alter your `config.sys` file. Refer to your MS-DOS manual for details.) If a macro invokes another macro, it can pass arguments to this second macro in the same way as described above. Such arguments may be explicit arguments or *%d* tokens. This allows you to pass through an argument from one macro to another.

Macro files may contain comments. Such comments start with a semicolon and end with the end of the line (e.g. `; this is a comment`). Comments are simply skipped by PC-Mos II. They also do not appear in the message window. A consequence of this syntax is that the command lines in a macro file cannot contain semicolons. (%-signs in contrast can occur—if they are not immediatelly followed by a digit.)

The formal rules for writing macro files are complete now. However, there are a few suggestions you might want to follow to avoid the most common pitfalls.

Since a macro can be invoked any time PC-Mos II expects a command, you do not know the current context of the program when you write the macro file. If a macro that has been designed to be invoked from the main menu is started from any other menu level, an annoying sequence of error messages is the result. You can avoid this problem if you only write macros that operatate from the main menu and set the instruction `eXit` as the first command line at the beginning of the macro file. This `eXit` will take you back to the main menu wherever you are. If you are already in the main menu, the `eXit` is simply ignored. (This is the only exception of the rule that PC-Mos II looks for a macro file when it encounters a command that is not found in the current menu.)

If you want a macro to automate some process completely, you must pay attention to giving all input items an option needs in the command line that invokes the option. If you forget one, PC-Mos II will prompt for this input and wait while you might be away drinking a cup of coffee. You must also pay attention to the order of the input items in the command line, which is the same order as PC-Mos II prompts for them. An exception are the verify prompts that PC-Mos II issues when it is about to do something perhaps dangerous. (An example is the prompt `Overwrite existing file? (Y/N)`.) Input prompts of this kind are skipped when PC-Mos II is in macro mode (the assumed answer depends on the particular option). Thus you need (and should) not write a `Y` or a `N` at the end of a macro command line to answer a verify prompt.

3 The Options of the File and the Choose Menu

3.1 Input and Output of Data

For input and output of all kinds of data the options `Read` and `Write` of the *File Menu* are provided. PC-Mos II only reads and writes ASCII files. This means that you can edit these files with a normal text editor. Although PC-Mos II can read and write different formats of data files, only one command for input and output respectively is provided. This is possible because PC-Mos II recognizes the format of the file by its file name extension. You should therefore pay attention to your data files having the correct file name extension. In the original configuration PC-Mos II knows following formats:

Suffix	Description
`.hdr`	Header information (such as drive velocity and number of channels) of not yet folded spectra.
`.asc`	Countrates of not yet folded spectra.
`.dta`	Data points of a folded spectrum possibly together with measurement errors. With this format you can also input any other xy-data not obtained from Mössbauer spectroscopy.
`.par`	Parameters of the theory program (their names, values and attributes).
`.res`	Fit results. I.e., mainly the fitted values of the parameters of the theory program.
`.prs`	Again fit results, but in a different format intended for direct printer output (see option `Print` of the *LPT Menu*).
`.mpl`	Plot data. This format is dedicated to transfer data from PC-Mos II to Mos-Plot in order to produce publication quality Mössbauer plots (see appendix C).

The format `.prs` is a write-only format. I.e., PC-Mos II can only create files with this format, but it cannot read them.

As already mentioned in the introduction, you can both modify the existing formats and design new ones. This is explained in detail in chapter 10.

After you have selected one of the options `Read` or `Write`, PC-Mos II prompts for a file name. This name may also include a DOS path. As mentionend above, the file name extension is essential since it tells PC-Mos II what format the file will have. However, if you leave out the extension, PC-Mos II will append extensions from a list of default suffixes. There are separate suffix lists for the options `Read` and `Write` (and one for the option `File` of the *Edit Menu*), each of which can hold up to four entries. You can modify these lists with the installation program MosSetup. If the list that comes into force has more than one entry, PC-Mos II will read/write/edit also more than one file—one for each suffix from the list. In the original configuration the list for the option `Read` contains the suffixes `.hdr`, `.asc` and `.par`, whereas the list for the option `Write` only contains the extension `.res`. An example: If you select the option `Read` and enter `test` after the file name prompt, PC-Mos II will in succession (try to) read the files

`test.hdr`, `test.asc` and `test.par`. The order here is essential, since before PC-Mos II can read the countrates of a spectrum (contained in a file with extension `.asc`) it has to now the number of channels (contained in a file with extension `.hdr`).

Often you will find yourself in the situation repeatedly having to read or write files with the same file name (but perhaps different extensions). The repeated typing of the same name can mostly be avoided by using the asterisk (`*`) as abbrevation. The asterisk is an alias for the file name (without extension!) displayed in the first not empty file name window (from left to right). Since the file name windows do not include paths, it is most convenient to start PC-Mos II in the directory that contains your data.

If you instruct PC-Mos II to create a file with a name that is already existing, PC-Mos II will issue an according warning followed by the verify prompt `Overwrite? (Y/N)`. If you enter `N`, PC-Mos II won't create the file. If this happens while PC-Mos II is executing a macro file, only the warning is issued and the verify prompt is skipped. An error message is issued if you instruct PC-Mos II to read a file that does not exist.

C The internal course of the input or output of a file is as follows: First of all PC-Mos II parses the entered file name and replaces a possible asterisk with the first stored file name. If the file name extension is missing, the first suffix of the default list is appended. Then the completed file name is passed to a programmable I/O-routine, which searches for a file with the name `pcmosfmt` and the same suffix as the passed file name. This file contains the data format of the file to be read or written. The format is read and compiled into a form which is easier to process, and then the routine checks if all prerequisites for the data I/O are met. Only then the actual I/O starts. This concept provides a great flexibility concerning data formats since the user can modify the existing format files or create new ones for his own requirements.

3.2 Course Records

PC-Mos II provides the possibility to record the whole dialogue between the user and the program in a file. You can use such log files to check if a macro has been processed correctley, but log files might also be useful for the creation of macro files: Perform the actions you want to automate manually a last time and record the dialogue in a log file. After that this log file can easily be converted into a macro file by removing PC-Mos II's prompts and combining your inputs to complete command lines.

A course record is started with the option `Open-Log` of the *File Menu*. When you select this option, PC-Mos II prompts for a file name which may include a path and an arbitrary extension. If you omit the extension, PC-Mos II will automatically append `.log`. Then PC-Mos II creates a file with this name and records all input and output that appears in the message window in this file. If a file with this name already exists, PC-Mos II issues a warning followed by the verify prompt `Append? (Y/N)`. The new recordings are appended to the existing file if you type `Y`. Otherwise no recording is done. Again, when PC-Mos II processes a macro, the verify prompt is skipped and the recordings are appended in any case.

The course recording can be finished with the option `Close-Log`. However, if you don't invoke this option the recording finishes—of course—when you quit the program. The corresponding file is closed automatically then.

3.3 Working with the Operating System

Sometimes you will want to do some work with the operating system although you have not yet finished your work with PC-Mos II. The option `DOS-Shell` can help you in this case. It invokes a command line processor (that one that is specified in the DOS environment variable `COMSPEC`), which enables you to execute all DOS commands and also other programs without terminating PC-Mos II. When you have finished your work, simply type `exit` at the DOS prompt, and PC-Mos II will regain control.

If you want to execute only one DOS command (or other program), you can use the option `System`, which prompts for a DOS command line, executes this command and returns to PC-Mos II immediatelly. This option also facilitates the inclusion of DOS commands in PC-Mos II macro files.

3.4 Changing the Selected Spectrum

As you will already have noticed, PC-Mos II is able to manage more than one spectrum at one time. However, one of these spectra is always the *selected* spectrum. This is the spectrum that is displayed and that all options (with two exceptions) refer to. PC-Mos II never changes the selected spectrum automatically, you must explicitly instruct PC-Mos II to do this. (If you make PC-Mos II read two data files subsequently, the data of the first file will be overwritten—the second data is not automatically written to the second spectrum!)

The mentioned exceptions are the option `Weber-Fit` of the main menu, which fits all available spectra simultaneously, and the option `Add` of the *Process Menu*, which adds a spectrum whose number the user must specify to the selected spectrum.

You can the change selected spectrum with the options of the *Choose Menu*. The options `First`, `Last`, `Next` and `Previous` do exactly what you will guess: they change to the first, the last, the next or the previous spectrum respectively. The number of spectra PC-Mos II manages and hence the ordinal number of the *Last* spectrum can be set to any desired value with the installation program MosSetup. In the original configuration this number is four.

4 The Options of the Process Menu

4.1 Folding Spectra

In spectra measured with sinusoidal Mössbauer drives there is no linear relation between the channel number of the multi-channel-analyzer (MCA) and the corresponding velocity or Doppler shift. Before PC-Mos II can fit such a spectrum it needs a *velocity map* which assigns the correct Doppler shift to each countrate. Additionally, in one period of the sine each velocity appears twice. Therefore it is advantageous to add the countrates of channels corresponding to the same velocity. This both improves the statistics and eliminates the linear contribution to the *geometry effect* (see sect. 5.2). The creation of the velocity map and the adding of matching channels are summarized by the concept *folding* and done by the option `Fold` of the *Process Menu*.

If the spectrum has been recorded with a triangular drive, the relation between channel number and velocity is linear. However, still two channels correspond to one velocity, so also a triangular spectrum must be folded before it can be processed further.

PC-Mos II only is able to fold a spectrum if it knows two quantities: the amplitude (i.e., the maximum velocity) of the Mössbauer drive and its phase relative to the MCA. The maximum velocity is normally contained in the data files, but you can also enter it manually with the option `Velocity` of the sub-menu `Data` of the *Edit Menu*. The phase must be given in form of a *folding point* which corresponds to the (spatial) crossover of the drive in terms of integral or half-integral channel numbers. An estimate of this folding point should also be contained in your data or parameter files. Then you can (and should) improve this estimate with the option `FoldingPoint` of the *Process Menu* (see sect. 4.2). You also can enter (an estimate for) the folding point manually with the option `Foldingpoint` of the sub-menu `Data` of the *Edit Menu*. There you can also choose between sinusoidal and triangular folding mode with the option `Mode`.

Once a spectrum is folded, PC-Mos II changes its graphical display. A not yet folded spectrum is shown by plotting the countrates against the channel numbers, whereas the abscissa of a folded spectrum is the velocity. You can also recognize a folded spectrum by the flag `F` displayed in the first small window on the screen.

$\mathcal{M}$ According to the convention to assign the folding point to the time when the source is moving away from the absorber with its maximum velocity, the velocity map for sinusoidal spectra is calculated according to the formula

$$v_k = -v_{\max}\cos\Big(2\pi\frac{t_k - t_F}{T}\Big) = -v_{\max}\cos\Big(2\pi\frac{k - K_F}{N}\Big). \qquad (4a)$$

Here N denotes the number of channels and K_F is the folding point in terms of integral or half-integral channel numbers. In triangular mode one simply has

$$v_k = v_{\max}\Big(1 - 2\,\Big|2\frac{k - K_F}{N} - 1\Big|\Big). \qquad (4b)$$

4.2 Determination of the Folding Point

As mentioned in the previous section PC-Mos II must know the *folding point* of a spectrum in order to fold it. In general, the folding point of a spectrum is not known exactly in advance. Therefore PC-Mos II provides the option `FoldingPoint` (in the *Process Menu*) which improves an estimate of the folding point to an accuracy of a half channel. However, this option only is successful if the estimate is good enough (about ±2 channels). Normally PC-Mos II recognizes a too bad estimate—if it is not much too bad—and issues a warning (`Indefinite result`). In this case you can either try to repeat the option (often the "indefinite result" is a better estimate) or enter a folding point manually with the option `Foldingpoint` of the sub-menu `Data` of the *Edit Menu*. Of course you can also improve the manually entered value with the option `FoldingPoint`.

For sinusoidal Mössbauer drives the folding point is defined as the time when the source is moving away from the absorber with its maximum velocity. This means that before reaching the folding point the source is nearer to the absorber than it is in the second half period of its motion. If you prefer to have the folding point in the other half period, you can achieve this by a change of the sign of the "maximum" velocity. However, PC-Mos II can handle and determine also folding points at the edge of a spectrum without problems. So this is purely a matter of taste. Incidentally, PC-Mos II notices a folding point that is in the wrong half period when trying to improve it (or when calculating the geometry effect, sect. 5.2) and issues a warning (`Sign of velocity may be wrong`) if necessary. You can change the sign of the maximum velocity with the option `Sign-of-velocity` of the sub-menu `Data` of the *Edit Menu*.

$\mathcal{M}$ PC-Mos II determines the folding point ϕ_F by a minimization process, which exploits the fact that the statistical deviations of the countrates are not correlated with the channel number or the phase ϕ_k of the Mössbauer drive. In sinusoidal mode following integral (which of course is calculated as a sum over the discrete channels) is minimized:

$$\int_0^{\pi} \Big(I(\phi_F + \phi) - I(\phi_F - \phi) + 2\kappa \big(I(\phi_F + \phi) + I(\phi_F - \phi) \big) \sin\phi \Big)^2 \, d\phi \qquad (4c)$$

Here $I(\phi)$ denotes the countrate in the "channel" ϕ and κ is the geometry effect (see sect. 5.2). To see why $(4c)$ is a minimum if ϕ_F is the correct folding point one first notices that the countrate $I(\phi)$ of channel ϕ can be written (up to linear order in the geometry effect κ) as

$$I(\phi) = I_0\big(-v_{\max}\cos(\phi - \phi_F)\big) \cdot \big(1 - 2\kappa \sin(\phi - \phi_F)\big) + \delta I\big(\phi\big), \qquad (4d)$$

where $I_0(v)$ is some abstract theoretical countrate only depending on the velocity and free of geometry effect and statistical deviations, and $\delta I(\phi)$ is the statistical deviation of the countrate in channel ϕ, which should be not correlated with ϕ. Here ϕ_F is the true folding point of course. Now substituting $(4d)$ in $(4c)$ yields an integrand in which the terms with I_0 cancel if ϕ_F in $(4c)$ is also the correct folding point. Hence if ϕ_F in $(4c)$

deviates from the true folding point, additional (positive) contributions to the integral containing I_0 arise, whereas the terms not containing I_0 do not change the integral significantly, because the statistical deviations are not correlated with ϕ.

$\mathcal{M}$ In the case of a triangular drive, instead of $(4c)$ now following integral is minimized:

$$\int_0^{\pi} \Big(I(\phi_F + \phi) - I(\phi_F - \phi) - 2\kappa \big(I(\phi_F + \phi) + I(\phi_F - \phi) \big) \xi(\phi) \Big)^2 d\phi \qquad (4e)$$

where

$$\xi(\phi) = \big(2\frac{\phi}{\pi} - 1 \big)^2 - 1 \qquad (4f)$$

is the position of the source in a triangular drive as function of the phase, normalized to unity amplitude. $(4e)$ must be a minimum for the correct folding point for the same reason as $(4c)$ is a minimum for a sinusoidal drive.

$\mathcal{M}$ A parameter of this procedure is the number of trial folding points, for which the integral $(4c)$ or $(4e)$ is calculated. Normally PC-Mos II checks a range of five channels around the estimated folding point, but you can change this value with the installation program MosSetup. The calculated values of the integral $(4c)$ or $(4e)$ are plotted on a normalized logarithmic scale. The normalization is choosen to yield expectation value zero and variance unity for the integral with the correct folding point. In practice this value won't occur often since mostly the folding point cannot be represented exactly by an integral or a half-integral channel number.

4.3 Rejecting and Modifying Data Points

PC-Mos II provides two options to reject data points. The option `Cut-off` of the *Process Menu* rejects all data points of a folded spectrum that are beyond the displayed velocity range. You can modify this range with the option `Range` of the sub-menu `Graphics` of the *Edit Menu*. Since sinusoidal Mössbauer drives produce the most data points at the edges of the spectrum, you can reduce the computer time for a preliminary fit significantly with this option. Another reason to use the option `Cut-off` might be to adapt the velocity range of the data points to that of a simulated theory curve.

With the option `Reject` of the *Process Menu* you can reject the countrates of single channels in a not yet folded spectrum. When you select this option PC-Mos II prompts for a channel number; note that the first channel has the number 0. This option is mainly intended to get rid of corrupt data that some multi-channel-analyzers produce in edge channels.

$\mathcal{C}$ Rejected channels are internally marked by a countrate zero. You can exploit this fact by manually setting a countrate to zero in a data file, which will yield the same result. If you output a file with not folded data after having rejected a channel, the corresponding countrate in the data file will also be zero. Of course rejected channels lead to a reduced number of data points in the folded spectrum. Note that also a channel symmetric to a rejected one must be disregarded when PC-Mos II folds the spectrum.

PC-Mos II also allows the modification of the data points of a folded spectrum. When you invoke the option `Modify` of the *Process Menu*, PC-Mos II passes an array with the data points to the theory program, which may modify this array according to the current parameter values (more details in chapter 11). Of course the effect of this option depends on the used theory program. Except for the example theory `Polynml` the enclosed theory programs do not support the modifaction of data points. An application of this option might be to eliminate some disturbing background effect, which has been fitted to the data points before.

A simple modification of the countrates of a not folded spectrum is facilitated by the option `Overflow`. Some older multi-channel-analyzers don't have enough bits per channel to record a spectrum with high countrates. If you know how many times the MCA has overflowed, you can correct this error with this option: PC-Mos II will prompt for a overflow correction which will be added to all countrates of the spectrum.

4.4 Adding Countrates

There are three options which are concerned with adding countrates. The option `Add` of the *Process Menu* adds the countrates of two not folded spectra channel by channel. This is useful if you have had to interrupt a measurement, and now you have two spectra of the same absorber, each with reduced statistics. A mathematically nearly equivalent method would be to fit both spectra simultaneously with all parameters declared as `common` (see chapter 7). But this would be a waste of computer time.

In order to add two spectra you have two input them into two different spectra of PC-Mos II. (Please excuse the ambigous meaning of the word "spectrum" as I use it here!) When you invoke the option `Add`, one of these spectra must be the selected one (see sect. 3.4). Then PC-Mos II prompts for the ordinal number of the other spectrum. After PC-Mos II has added the spectra successfully, the sum spectrum is the selected one. The original data of both spectra is rejected. The adding of two spectra only is possible if following quantities match exactly: number of channels, maximum velocity and folding point.

With the option `Join` you can add the countrates of adjacent channels of one not yet folded spectrum and thus reduce the number of data points while simultaneously improving the statistics of the single data points. (Of course the overall statistics is not improved!) Like the option `Cut-off` this might be useful to reduce the computer time for preliminary fits. The option `Join` only works if the number of channels is even.

Finally, with the option `Dilute` you can reduce the number of data points of a folded spectrum. This option is mainly intended to make plots of sinusoidal spectra clearer, since normally the increased density of data points at the edges of the spectrum yields two non-resolved big black splodges. If you select this option, PC-Mos II prompts for a maximum (horizontal) distance between subsequent data points. PC-Mos II then continues to combine adjacent points by averaging their velocities and countrates and by appropriatelly reducing the statistical error of the combined data point until further combination would yield a distance exceeding your specified value. To preserve most

of the information, PC-Mos II does not combine data points with a vertical distance exceeding four standard deviations.

5 The Options of the Assess Menu

5.1 Calculation of the Effect

The *effect* of a Mössbauer spectrum is known as the maximum relative absorbtion at one point of the spectrum. From PC-Mos II's point of view the effect is not of interest for the further processing of the spectrum. Therefore it only is calculated if you order this explicitly with the option `Effect` of the *Assess Menu*. The effect is output as percentage. In principle the calculation of the effect requires the knowledge of the *baseline* (see sect. 5.3). However, if the baseline is not available, PC-Mos II uses the maximum countrate instead. This also facilitates the calculation of the effect for a not folded spectrum. If you need an accurate determination of the effect, you should nevertheless calculate it only after the spectrum has been folded and after the baseline has been estimated with one of the options described in section 5.3.

5.2 Estimation of the Geometry Effect

The option `Geometry-Effect` allows an estimation of the *geometry effect*, which is caused by the finite (spatial) amplitude of the Mössbauer drive. Thus the distance between source and detector aperture is not constant but varies correlated with the Doppler shift. If no other effects contribute, the geometry effect is given by the ratio of the amplitude of the drive and that distance. Internally an estimation of the geometry effect is of interest for the determination of the folding point (see sect. 4.2) and therefore is done automatically then. The geometry effect is output as percentage.

$\mathcal{M}$ The dependence of the measured countrate on the phase ϕ of a sinusoidal Mössbauer drive may be written as

$$I(\phi) = \frac{I_0\big(-v_{\max}\cos(\phi-\phi_F)\big)}{\big(1+\kappa\sin(\phi-\phi_F)\big)^2} + \delta I(\phi). \qquad (5a)$$

Here $I_0(v)$ denotes an abstract theoretical countrate only depending on the velocity and free of geometry effect and statistical deviations. ϕ_F is the folding point, κ is the geometry effect, and $\delta I(\phi)$ is the chance statistical deviation in the channel ϕ. Since κ is normally of order 10^{-3}, the right side of $(5a)$ may be expanded in a Taylor series and for the difference of countrates of channels symmetric to the foldingpoint one obtains:

$$I(\phi)-I(-\phi) = -4\kappa I_0\big(-v_{\max}\cos(\phi-\phi_F)\big)\sin(\phi-\phi_F) + \delta I(\phi) - \delta I(-\phi) + \mathcal{O}(\kappa^2) \qquad (5b)$$

Weighting of this difference with $\sin(\phi-\phi_F)$ and integration over all channels yields:

$$\int_0^\pi \big(I(\phi)-I(-\phi)\big)\sin(\phi-\phi_F)\,d\phi = -4\kappa\int_0^\pi I_0(-v_{\max}\cos\phi)\sin^2\phi\,d\phi \qquad (5c)$$

Here the fact has been exploited that $\delta I(\phi)$ is not correlated with $\sin\phi$ or $\sin^2\phi$. On the other hand, for the same reason:

$$\int_0^\pi \big(I(\phi)+I(-\phi)\big)\sin^2(\phi-\phi_F)\,d\phi = 2\int_0^\pi I_0(-v_{\max}\cos\phi)\sin^2\phi\,d\phi \qquad (5d)$$

Thus the ratio of the integrals (5c) and (5d), which can be calculated by a summation over all channels, yields the geometry effect κ.

$\mathcal{M}$ In the case of a triangular drive, in above formulas only the cosine function must be replaced by $\zeta(\phi) \equiv 2|\phi/\pi - 1| - 1$ and the sine function by $-\xi(\phi) \equiv (2\phi/\pi - 1)^2 - 1$. The arguments leading to (5c) and (5d) remain the same.

5.3 Determination of the Baseline

The *baseline* of a Mössbauer spectrum is known as the avarage countrate of data points with no absorbtion. In general, a transmission spectrum has the form $F(v) = I_0 \cdot \big(1 - f(v)\big)$, where I_0 denotes the baseline and $f(v)$ is the normalized absorbtion curve. Of course the theoretical curve to be fitted to the spectrum has the same form, and thus the baseline is necessarily one of the parameters of the theory program.

As it is the case with all parameters of the theory program, you need a reasonable estimate for the baseline before you can expect the fit to be successful. Since the baseline depends strongly on the measuring time and the absorber thickness, it is difficult to determine such an estimate in advance. PC-Mos II therefore provides two options to calculate the approximate baseline before fitting the spectrum.

PC-Mos II always identifies the first parameter of the theory program with the baseline. This is the only assumption PC-Mos II makes about these parameters. However, this assumption only has consequences if you invoke one of the options `Effect`, `Margin-Baseline` or `Statistic-Baseline`. You can use theories which interpret the parameters differently if you avoid using these option. If you instruct PC-Mos II to estimate the baseline before having input a file with parameters, PC-Mos II will create a dummy parameter list only containing the baseline and issue an according warning. Both options for the baseline estimation only work if the spectrum is folded.

5.3.1 Baseline Estimation from Marginal Channels

The standard method to estimate the baseline is to avarage over a certain number of countrates from the edges of the spectrum. The option `Margin-Baseline` uses this method. The method works well as long as the edges of the spectrum are not too much affected by absorbtion lines, i.e., as long as the scanned velocity range is large enough. A parameter of the method is the percentage of data points from the edges to be avaraged over. Normally PC-Mos II uses 5% of the data points but you can modify this value with the installation program MosSetup.

5.3.2 Baseline Estimation from Maximum Countrate Statistics

A second method for baseline estimation is provided with the option `Statistic-Baseline`. This method only assumes that there is a sufficient number of data points not affected by absorbtion lines, not necessarily sited at the edges. Again you can modify the number of data points to avarage over, which is normally 20%, with the installation program MosSetup.

$\mathcal{M}$ Assuming that the data points can rigidly be devided into points affected by absorbtion and points not affected by absorbtion, the value of the baseline I_0 is the avarage of the countrates of those data points that are not affected by absorbtion lines. According to the theory of the Poisson distribution the countrates I of these points should approximately be normally distributed with mean and variance equal to I_0:

$$\rho(I) = \frac{1}{\sqrt{2\pi I_0}} e^{-(I-I_0)^2/2I_0} \tag{5e}$$

If one only considers countrates above some limit $I_{\min}$, only the normalization of the distribution changes:

$$\rho'(I) = \frac{e^{-(I-I_0)^2/2I_0}}{\int_{I_{\min}}^{\infty} e^{-(x-I_0)^2/2I_0}\, dx} = \frac{e^{-(I-I_0)^2/2I_0}}{\sqrt{\frac{\pi}{2} I_0}\ \mathrm{erfc}\left(\frac{I_{\min}-I_0}{\sqrt{2I_0}}\right)} \tag{5f}$$

The mean of this distribution is given by

$$\langle I \rangle_{I \geq I_{\min}} = I_0 + \sqrt{\frac{2I_0}{\pi}}\, \frac{e^{-(I_{\min}-I_0)^2/2I_0}}{\mathrm{erfc}\left(\frac{I_{\min}-I_0}{\sqrt{2I_0}}\right)}. \tag{5g}$$

If one samples the, say, N maximum countrates of a spectrum, it is reasonable to assume that these countrates are not affected by absorbtion lines. Hence these countrates should by distributed according to $(5f)$ with $I_{\min}$ given by the smallest of the sampled countrates. Then the mean $(5g)$ can be approximated by the sample mean and the resulting equation can be iterated to yield the baseline I_0.

5.4 Further Options

With the option `Locate` you can locate a specific point of a displayed spectrum. If you invoke this option, a crosshair appears, which can be moved with the arrow keys [↑], [↓], [←] and [→]. You can change the speed with [+] and [−]: [+] doubles the speed, [−] halfes it. After you have moved the crosshair to the desired position, you can press [Return] and PC-Mos II will display the coordinates of the located point. The way PC-Mos II displays these coordinates naturally depends on whether the spectrum is folded or not, and whether a baseline is available.

To check the quality of a fit it is often useful to see the deviations of the data points from the theory curve directly. This is facilitated by the option `Deviations`, which displays these differences on the screen.

6 The Options of the Simulation Menu

6.1 Calculation of Simulations

To calculate a simulation here means to calculate the theoretical transmission curve for a given set of velocity values, perhaps together with some *sub-spectra*, which together assemble the overall transmission curve, the *envelope*. In general it is necessary to calculate some simulations before starting the fit in order to find some reasonable estimates for the parameters of the theory curve. If the start values of these parameters are too bad, the fit normally does not converge.

Simulations are calculated by the option `Update` of the *Simulation Menu*. This option actually does not much more than invoke the theory program and pass the current values of the parameters and a velocity map to it. Then the theory program returns the values of the theoretical transmission curve at the points of the velocity map. This means that before a simulation can be calculated, the name of the theory program and a parameter list must be available. Data need not be available, but if it is, it must also be folded, since PC-Mos II cannot display a not folded spectrum together with a transmission curve. If you calculate a simulation after a fit, PC-Mos II of course uses the fitted parameter values for the simulation rather than the estimated start values. The option `Update` is invoked automatically when you select the *Simulation Menu* and simulation data is not yet available. Whether simulation data is available or not can be seen by the flag `S` in the first of the small screen windows. If simulation data is already available, you must explicitly invoke the option `Update` to ensure that the perhaps changed parameter values take effect.

$\mathcal{M}$ Before PC-Mos II can invoke the theory program to calculate a simulation, it must set up a velocity map. This map is determined by four parameters: the minimum and maximum velocity, the number of simulation points and a so-called *non-equidistancy parameter*, which controls how much tighter the points will lie in the middle of the velocity range. (The density of points is normally higher in the middle because there most of the information is contained.) The velocity map is calculated according to the formula

$$v_k = v_{\min} + \left(k + \kappa \frac{\sin\frac{2\pi k}{n}}{4\sin\frac{\pi}{n}}\right)\frac{v_{\max} - v_{\min}}{n}; \qquad k = 0, \ldots, n. \tag{6a}$$

Here $n+1$ is the number of simulation points and κ is the non-equidistancy parameter. The distance between adjacent simulation points is given by

$$v_{k+1} - v_k = \left(1 + \frac{\kappa}{2}\cos\pi\frac{2k+1}{n}\right)\frac{v_{\max} - v_{\min}}{n}; \qquad k = 0, \ldots, n-1. \tag{6b}$$

From this it can be seen that $\kappa = 0$ yields an equidistant velocity map and that κ may take on values between -2 and $+2$. A moderate value is $\kappa = 1$, which also is the default value.

Minimum and maximum velocity of the simulation velocity map are identical with those of the graphical display and can be modified with the option `Range` of the sub-menu `Graphics` of the *Edit Menu*. The number of simulation points (normally 152) and above described non-equidistancy parameter can be modified with the installation program MosSetup or, temporarily, with the options `Resolution` and `Non-Equidistancy` of the sub-menu `Simulation` of the *Edit Menu*.

6.2 Sub-spectra

When a simulation is calculated, the theory program normally not only computes the overall transmission curve, the *envelope*, but also some partial transmission curves, the *sub-spectra*. The idea behind the concept of sub-spectra is that mostly the total resonance absorbtion of a specimen is due to some different physical effects and is composed additively of several groups of absorbtion lines therefore. (This only is true if the absorber is not too thick.) A transmission spectrum that would arise if only one of these effects contributed to the absorbtion is called sub-spectrum.

$\mathcal{M}$ Let I_0 be the baseline, $f(v)$ the normalized overall absorbtion curve and $f_i(v)$ a normalized partial absorbtion curve in above sense. Then the overall transmission curve $F(v)$ (the envelope) is given by

$$F(v) = I_0 \cdot \big(1 - f(v)\big) = I_0 \cdot \Big(1 - \sum_{i=1}^{N} f_i(v)\Big). \tag{6c}$$

Since the transmission curve of the sub-spectra is naturally defined by

$$F_i(v) = I_0 \cdot \big(1 - f_i(v)\big), \tag{6d}$$

following relation between the envelope and the sub-spectra can be deduced:

$$\sum_{i=1}^{N} F_i(v) = F(v) + (N-1)I_0. \tag{6e}$$

In above equations N denotes the number of sub-spectra that contribute to the envelope. If a theory program calculates sub-spectra that do not obey the relation (6e), the options of the *Simulation Menu* described in the next section will not work correctly.

A problem arises from the fact that PC-Mos II does not know in advance, how many sub-spectra a theory program will calculate. Therefore it assumes the worst case and reserves memory for as many sub-spectra as parameters are available, although the theory will probably calculate much less sub-spectra. If you are short of memory, you can reduce the number of sub-spectra PC-Mos II reserves memory for with the option `Subspectra` of the sub-menu `Simulation` of the *Edit-Menu*. Note, however, that by this means you cannot reduce the number of sub-spectra the theory program actually calculates. If it calculates more sub-spectra than PC-Mos II has reserved memory for, your system will inevitably crash.

6.3 Display of Simulations

The remaining options of the *Simulation Menu* control the way the simulation and the data are displayed. With the toggles `Envelope`, `Subspectra` and `Data` you can turn on and off the display of the envelope, the sub-spectra and the data. After you have entered the *Simulation Menu*, the state of the first two switches is always off, whereas the data toggle is on.

With the option `Light` you can make one of the sub-spectra blink. When you select this option, PC-Mos II will prompt for the number of the sub-spectrum. The blinking is finished by pressing any key.

The option `Cut-out` enables you to eliminate the contribution of a sub-spectrum to the envelope, in order to see how the envelope is affected by this sub-spetrum. When you select this option, PC-Mos II will prompt for the number of the sub-spectrum. You can eliminate the contributions of more than one sub-spectrum by invoking `Cut-off` several times subsequently. The option `Restore` cancels all previous eliminations and restores the original envelope.

7 The Marquardt-Levenberg Fit

7.1 Foundations

The central part of a Mössbauer program is a fit routine, which fits a model curve depending on some parameters to a set of data points, the spectrum. Many methods for fitting model curves to data points exist, but the Marquardt-Levenberg method is probably the most commonly used. PC-Mos II also uses this method, optionally with a slight modification that goes back to Markus Weber (hence the term `Weber-Fit`). This first section gives a short survey of the theoretical foundations of the Marquardt-Levenberg fit. If you only want to know how to use it at the moment, you will probably want to skip the remainder of this section.

$\mathcal{M}$ First of all one must settle what it means to fit a curve to some data. The Marquardt-Levenberg fit belongs to the class of *weighted least squares fits*, which minimize following merit function:

$$\Phi(\mathbf{b}) \equiv \sum_{i=1}^{N} \left(\frac{y_i - f(x_i; \mathbf{b})}{\Delta y_i} \right)^2 \qquad (7a)$$

Here $\mathbf{b}$ denotes a vector whose components are the parameters of the model function $f(x; \mathbf{b})$, y_i is the measured countrate belonging to the velocity x_i, and Δy_i is the statistical error (the standard deviation) of this countrate (normally but not necessarily $\Delta y_i = \sqrt{y_i}$). N is the number of data points. Intuitively it is clear that the minimization of $(7a)$ fits the curve to the data. A more mathematical reason for this merit

function is that the vector $\mathbf{b}_{\min}$ minimizing (7*a*) is a *maximum likelihood estimator* for the "true" parameter vector $\mathbf{b}_{\text{true}}$—if the measurement errors are normal deviates.

$\mathcal{M}$ The determination of the vector $\mathbf{b}_{\min}$ minimizing the merit function is possible only by iterative means. Given an estimate $\mathbf{b}'$ for $\mathbf{b}_{\min}$, there are two basic strategies to reach a better estimate $\mathbf{b}''$. Starting point for the first method is the fact that near its minimum every (not too pathological) function can be well approximated by a quadratic form:

$$\Phi(\mathbf{b}) \approx \Phi_0 - \mathbf{u}^T\mathbf{b} + \frac{1}{2}\mathbf{b}^T\mathbf{A}\mathbf{b} \tag{7b}$$

where

$$\Phi_0 \equiv \Phi(\mathbf{b}')\,, \qquad A_{ij} \equiv \frac{\partial^2\Phi(\mathbf{b}')}{\partial b_i \partial b_j} \qquad \text{and} \qquad \mathbf{u} \equiv \mathbf{A}\mathbf{b}' - \nabla\Phi(\mathbf{b}').$$

At the point $\mathbf{b}_{\min}$ the gradient of $\Phi(\mathbf{b})$ must vanish. Thus a reasonable correction vector $\Delta\mathbf{b} \equiv \mathbf{b}'' - \mathbf{b}'$ might be given by the linear equation system

$$\mathbf{A}\Delta\mathbf{b} = -\nabla\Phi(\mathbf{b}')\,, \tag{7c}$$

whose solution would yield $\mathbf{b}_{\min}$ in one step if the approximation by a quadratic form was exact. In terms of the data points and the model function the gradient $\nabla\Phi$ and the Hesse matrix $\mathbf{A}$ may be written:

$$\frac{\partial\Phi}{\partial b_k} = -2\sum_{i=1}^{N}\frac{y_i - f(x_i;\mathbf{b})}{\Delta y_i^2}\frac{\partial f(x_i;\mathbf{b})}{\partial b_k} \tag{7d}$$

$$\frac{\partial^2\Phi}{\partial b_k \partial b_l} = 2\sum_{i=1}^{N}\frac{1}{\Delta y_i^2}\left(\frac{\partial f(x_i;\mathbf{b})}{\partial b_k}\frac{\partial f(x_i;\mathbf{b})}{\partial b_l} - \big(y_i - f(x_i;\mathbf{b})\big)\frac{\partial^2 f(x_i;\mathbf{b})}{\partial b_k \partial b_l}\right) \tag{7e}$$

The second term on the right side of (7*e*) containing second derivatives of the theory function is normally skipped for practical calculations. This can be justified by the fact that for a correct model $f(x;\mathbf{b})$ the differences $y_i - f(x_i;\mathbf{b})$ are not correlated and tend to cancel in the sum. Incidentally, replacing the model function with its linear Taylor approximation and solving the thus linearized minimization problem yields the same equation system for $\Delta\mathbf{b}$ (without the term containing second derivatives).

$\mathcal{M}$ Since the approximation by a quadratic form only is sufficiently good near the minimum of $\Phi(\mathbf{b})$, a second strategy for the minimization is needed. This strategy is simply to take a correction step in the direction of the steepest descent of $\Phi(\mathbf{b})$. I.e.,

$$\Delta\mathbf{b} = -\alpha\nabla\Phi(\mathbf{b}').$$

However, the gradient alone contains not even information about the order of magnitude of the constant α. This problem is solved by the Marquardt-Levenberg method, which additionally allows to interpolate smoothly between both strategies. Now the correction vector $\Delta\mathbf{b}$ is given as solution of the equation system

$$\mathbf{A}'\Delta\mathbf{b} = \mathbf{g} \qquad \text{where} \qquad \mathbf{A}' \equiv \frac{1}{2}\mathbf{A} + \lambda\mathbf{1} \qquad \text{and} \qquad \mathbf{g} \equiv -\frac{1}{2}\nabla\Phi(\mathbf{b}')\,. \tag{7f}$$

Here **1** denotes the unity matrix and $\lambda \geq 0$ is the continuous inperpolation parameter. Obviously, setting $\lambda = 0$ again yields the equation system (7*c*). On the other hand, if λ is large enough, the matrix $\mathbf{A}'$ becomes diagonally dominant and the correction vector is approximately given by

$$\Delta\mathbf{b} = -\frac{1}{2\lambda}\nabla\Phi(\mathbf{b}')\,, \qquad (7g)$$

i.e., one essentially regains the gradient method without the problem of a seemingly arbitrary constant α. (Note that actually the elements of the Hesse matrix **A** determine α, since it depends on them, for which λ $\mathbf{A}'$ becomes diagonally dominant.)

$\mathcal{M}$ Now the basic algorithm for the Marquardt-Levenberg fit can be presented: First choose a moderate start value for the Marquardt parameter λ (PC-Mos II chooses 0.01 but you can change that). For this λ and a first estimate $\mathbf{b}'$ for the parameter vector calculate the merit function $\Phi(\mathbf{b}')$, the Hesse matrix **A** and the inhomogenity **g** and thus the correction vector $\Delta\mathbf{b}$. This yields a trial parameter vector $\mathbf{b}''$ for which again the merit function $\Phi(\mathbf{b}'')$ is evaluated. If the result is smaller than the previous value, accept $\mathbf{b}''$ as a better estimate and reduce λ by a factor 10. Otherwise stay with the old parameter vector and advance λ by a factor 10. This procedure is repeated until a stop criterion, which can be specified by the user, is met. In the original configuration the fit is stopped if the relative change of $\Phi(\mathbf{b})$ is smaller than 0.1 % two times subsequently, if the maximum relative change of one of the parameters is smaller than 0.1 % or if the number of iterations exceeds the limit 25.

$\mathcal{M}$ A small complication not mentioned so far is still left: Since the parameters of the model function are physical quantities, they mostly are also dimensional. In general, the components of the parameter vector will even have different dimensions. Hence, although the merit function itself is nondimensional, in general also the gradient $\nabla\Phi$ and the Hessian **A** will be dimensional, and adding the nondimensional parameter λ to the diagonal elements of **A** is at least something a physicist doesn't like. One can avoid this problem by rescaling the matrix **A** appropriatelly and modifying the equations accordingly. The scaling factors are essentially given by the diagonal elements of **A**. This procedure has the additional advantage of improving the numerical stability of the algorithm.

7.2 Fitting

7.2.1 Fit Preparations

In order to fit one or several spectra you have to invoke the option `Weber-Fit` of the main menu. This not only activates the *Fit Menu* but also orders all internal preparations for fitting all available spectra. In particular, memory is reserved for derivation matrices and similar items. (If at this point the message `ERROR - Not enough memory` is issued, you can perhaps save the situation by a method described in section 8.2.)

As mentioned before, the fit option is the only one that affects all available spectra rather than only the selected one. PC-Mos II always fits all available spectra being in a "fitable state" simultaneously. Fitable state means that data must be available, that

data must be folded and that start parameters but not fit results must be available. In terms of the spectrum flags displayed on the screen this means that the flags `D`, `F` and `P` must be set, whereas the flag `R` must be clear. (If you want to fit a spectrum although results are already available, you must either re-read the parameters or use the option `Upgrade-Parameters` of the *Dispose Menu* described in section 8.6.) If none of PC-Mos II's spectra meets these conditions, an according message is issued and the *Fit Menu* is not invoked.

One of the fit preparations PC-Mos II makes when you invoke the option `Weber-Fit` is to connect the parameters of all spectra to be fitted. The way these parameters are connected depends on their *attributes*: Each parameter is assigned one of the attributes `Standard`, `Common` or `Fixed`. Parameters with attribute `Fixed` are kept constant during the fit. They neither contribute to the memory requirements nor to the computing time. They only are passed to the theory program with their constant value each time the theory is invoked. Parameters with attribute `Standard` or `Common` are free parameters. During the fit PC-Mos II changes their values in order to minimize the merit function ($7a$). However, whereas standard parameters only concern the spectrum they belong to (i.e., the spectrum that was the selected one when you input the parameters) and are changed completely freely, the common parameters of all spectra are identified with each other and only are changed simultaneously. This means that if you want to fit several spectra simultaneously, you should assign the attribute `Common` to those parameters representing a physical quantity that is identical for all spectra.

The common parameters of all spectra are identified with each other according to their order in the parameter sets of each spectrum. I.e., the first common parameter of the first spectrum is identified with the first common parameter of the second one, and so on. This is the only criterion used. The common parameters need not occupy the same places in the parameter sets of the different spectra. The different spectra even need not be fitted by the same theory program. This allows a great flexibility for simultaneous fitting but has the drawback of easily leading you into making mistakes. You should always pay attention to which parameters will be identified by PC-Mos II. The only plausibility check PC-Mos II can make is to compare the numbers of common parameters of all spectra. If these numbers don't match, PC-Mos II issues an error message.

In the parameter files with format `.par` attributes are assigned to the parameters by putting a letter behind their values (separated by a blanc): `C` stands for `Common` and `F` stands for `Fixed`. Standard parameters need not be marked especially. These letters also appear in the result files with format `.res` and in the formatted printer output with format `.prs`.

7.2.2 The Options of the Fit Menu

Before the fit actually starts, the *Fit Menu* provides the options `Start`, `Info`, `Monte-Carlo` and `Abort`. The option `Abort` cancels all fit preparations and takes you back to the main menu. With `Monte-Carlo` you can replace the measured data points with simulated ones distributed statistically around the values of the theory curve. This option is intended as error control and is discussed in section 7.5.

With the option `Info` you can retrieve some information about the state of the fit routine. PC-Mos II displays following items:

- The number of spectra which will be fitted simultaneously,
- the total number of data point of all these spectra,
- the maxium number of data points of one spectrum (the memory requirements depend strongly on this number),
- the total number of free (standard or common) parameters (the second quantity contributing strongly to the memory requirements),
- the number of common parameters,
- the maximum number of free parameters of one spectrum
- the number of points calculated during one call of the theory program(s).

Finally, the option `Start` starts the actual fit. After that the menu line provides the options `Continue Info`, `Finish` and `Abort`, which, however, are accessable only after you have interrupted the fit with Ctrl Break. (If a macro file is being processed, you must press Ctrl Break a second time when the message `Ctrl-Break detected - Press ESC` is issued, in order to abort also the macro file processing.) You should note that you cannot regain control before the current fit iteration has been completed.

After an interruption you can continue the fit with the option `Continue`. The option `Info` again displays some information about the state of the fit routine:

- The number of iterations done so far together with its upper limit,
- the current value of the quantity χ^2, which is identical with the merit function ($7a$) except for a normalization constant,
- the percentage change of χ^2 during the preceeding iteration,
- the *fit flags*, which are discussed in the next subsection,
- the maximum percentage change of one of the parameters during the preceeding iteration,
- the current value of Marquardt's parameter λ (see sect. 7.1).

The options `Finish` and `Abort` both terminate the fit and take you back to the main menu. However, there is an important difference: `Abort` simply cancels the fit. All iterations done so far are disregarded and no fit results are available. The spectra are in the same state as before the option `Weber-Fit` has been invoked. On the other hand, `Finish` forces a regular end of the fit. The current values of the parametes are considered as results, and errors and check values are calculated as if the fit had terminated because of a regular stop criterion (see sect. 7.1).

C Actually, terminating the fit with `Finish` only works if the fact "user interruption" has been defined as a stop criterion (see sect. 8.2). However, this is the case in the original configuration, but you can change this. `Finish` does nothing more than set the fit flag `U` and continue the fit with the test of the stop criterions. If the flag `U` is not defined as stop criterion, the fit will go on. Terminating the fit with `Abort`, however, always works.

After the fit has terminated, regardless in which way, the main menu is invoked automatically. You need not leave the *Fit Menu* by an `eXit` command.

7.2.3 Monitoring the Fit; the Fit Flags

With each iteration PC-Mos II plots the theory curve belonging to the selected spectrum as it is defined by the current values of the parameters. (Of course also the data points are displayed.) By that you can see whether the fit converges or runs away because of too bad start values for the parameters. Additionally PC-Mos II issues some information in the message window after each iteration:

- The number of iterations done so far together with its upper limit,
- the current value of χ^2,
- the percentage change of χ^2 during the preceeding iteration,
- the *fit flags* discussed below.

The quantity χ^2 is identical with the merit function Φ defined in section 7.1, eq. (7*a*), except for the normalization: $\chi^2 \equiv \Phi/(N-M)$, where N is the total number of data points and M is the total number of free parameters. For a correct model χ^2 has the statistical expectation value 1. Hence this quantity tells you how good the curve is actually fitted to the data. On the other hand, the percentage change of χ^2 tells you how far the merit function has approached its minimum, regardless whether this is a good fit or not. Therefore the percentage change is one of the stop criterions of the fit.

Additional information is given by the *fit flags*. This is true especially after the end of the fit, and the fit flags are also contained in the result files with format `.res` and in the formatted printer output with format `.prs`. This is the meaning of the single fit flags:

Flag	Meaning
`U`	User interruption. This flag is set if the fit has been interrupted by the user. After the end of a fit it indicates that the fit has been terminated with `Finish`.
`I`	Iteration limit exceeded. The `I`-flag is set if the number of iterations has reached or exceeded a specific limit which is normally 25. Of course this value can be modified.
`P`	No significant change of the parameters. This flag is set if the maximum percentage change of one of the parameters has been less than 10^{-S} % during the preceeding iteration. The integral number S, the parameter significance, is normally 1 but can also be changed.
`C`	No significant change of χ^2. This flag indicates that the percentage change of the quantity χ^2 has been less than $10^{-S'}$ %. The number S', the χ^2-significance, is normally 1 but can be changed too.
`E`	Successful calculation of errors. This flag can be set only after the end of a fit and indicates that during the calculation of errors and check values (see sect. 7.3) no error has occured.
`M`	Monte-Carlo simulation. This flag is set if the fitted data points are not measured but simulated with the option `Monte-Carlo` (see sect. 7.5).

You can override the default values of the iteration limit, the parameter significance and the χ^2-significance by inserting your values in the parameter files, tagged with the

names `MaxIter`, `ParaSig` and `Chi2Sig` respectively, or by modifying them online with the option `Significance` of the sub-menu `Parameters` of the *Edit Menu*.

$\mathcal{C}$ In terms of the fit flags the fit stop criterions can be formalated easily. The stop criterion of the original configuration says: "Stop fitting if after an iteration one of the flags `U`, `I` or `P` has been set or if the flag `C` has been set two times subsequently." (It is useful to compare this with the formulation at the end of section 7.1.) You can define other stop criteria, which is done by providing a logical expression containing the fit flags. This is discussed in section 8.2.

7.3 Calculation of Errors and Check Values

7.3.1 The Covariance Matrix of the Parameters

Apart from the fitted values of the parameters of the theory curve PC-Mos II also calculates their covariance matrix, which yields the standard deviations (= "errors") of the parameters. In the result files with format `.res` these errors are output behind the fitted parameter values. You should note, however, that these errors are purely statistical errors due to the statistical variations of the countrates. They do not contain any information about systematic errors, which may result from an incorrect or inaccurate model. Statistical tests that check the quality of a model are discussed in section 7.4.

$\mathcal{M}$ Starting point for the calculation of errors is the assumption that the fitted model is correct and thus a set of "true" parameter values b_i does exist. Then the measured values y_i are samples of a statistical distribution with mean $f(x_i;\mathbf{b})$ and variance $(\Delta y_i)^2$. The parameter values b'_i determined by the fit are the solution of the minimization problem

$$\Phi(\mathbf{b}') \equiv \sum_i \left(\frac{y_i - f(x_i;\mathbf{b}')}{\Delta y_i} \right)^2 \to \min . \tag{7h}$$

Replacing the theory function $f(x_i;\mathbf{b}')$ with a linear Taylor approximation around the true parameter vector $\mathbf{b}$

$$f(x_i;\mathbf{b}') = f(x_i;\mathbf{b}) + \sum_j \frac{\partial f(x_i;\mathbf{b})}{\partial b_j}(b'_j - b_j) \tag{7i}$$

yields

$$\sum_i \left(\frac{y_i - f(x_i;\mathbf{b})}{\Delta y_i} - \sum_j \frac{1}{\Delta y_i} \frac{\partial f(x_i;\mathbf{b})}{\partial b_j}(b'_j - b_j) \right)^2 \to \min . \tag{7j}$$

Using a more compact notation one can write

$$\|\mathbf{g} - \mathbf{P}\Delta\mathbf{b}\|^2 \to \min, \tag{7k}$$

with

$$g_i \equiv \frac{y_i - f(x_i;\mathbf{b})}{\Delta y_i}, \qquad P_{ij} \equiv \frac{1}{\Delta y_i}\frac{\partial f(x_i;\mathbf{b})}{\partial b_j} \quad \text{and} \quad \Delta b_j \equiv b'_j - b_j . \tag{7l}$$

Thus the deviation vector $\Delta\mathbf{b}$ is the solution of the normal equations

$$\mathbf{P}^T\mathbf{P}\Delta\mathbf{b} = \mathbf{P}^T\mathbf{g} \quad \text{or} \quad \Delta\mathbf{b} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{g}\,. \tag{7m}$$

With this representation the statistical properties of the deviations Δb_i are easily determined. First one obtains:

$$E[g_i] = \frac{1}{\Delta y_i}\big(E[y_i] - f(x_i;\mathbf{b})\big) = 0, \tag{7n}$$

$$E[g_i g_j] = \frac{1}{\Delta y_i \Delta y_j} E[(y_i - f(x_i;\mathbf{b}))(y_j - f(x_j;\mathbf{b}))] = \delta_{ij}\,. \tag{7o}$$

Here E denotes the statistical expectation value. Equation (7o) follows from the fact that the deviations of the measured values from the true theory curve are not correlated. Using the compact notation $E[\mathbf{g}] = 0$, $E[\mathbf{g}\mathbf{g}^T] = \mathbf{1}$ one continues:

$$E[\Delta\mathbf{b}] = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T E[\mathbf{g}] = 0, \tag{7p}$$

$$E[\Delta\mathbf{b}\Delta\mathbf{b}^T] = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T E[\mathbf{g}\mathbf{g}^T]\mathbf{P}(\mathbf{P}^T\mathbf{P})^{-1} = (\mathbf{P}^T\mathbf{P})^{-1}. \tag{7q}$$

Thus $(\mathbf{P}^T\mathbf{P})^{-1}$ is the covariance matrix of the parameter values determined by the fit, and their errors or standard deviations are given by the square roots of the diagonal elements of this matrix.

$\mathcal{M}$ You have certainly noticed that for the determination of $(\mathbf{P}^T\mathbf{P})^{-1}$ actually the true parameter values b_i would be needed, which are, of course, unknown. Thus one has to use the fitted values b_i' instead and hope that the derivations of the theory function do not too critically depend on the parameters near the minimum. This assumption must also be made to justify the use of a Taylor approximation in (7i).

7.3.2 The Check Values

Apart from the standard deviation PC-Mos II calculates two further quantities for each parameter, the so-called *check values*. These quantities contain some information about the topology of the merit function (7a) at the point the fit terminated. In the ideal case both check values of all parameters take on the value 1. This would indicate that both the fit finished in a minimum of the merit function and the merit function is well approximated by a quadratic form at this point. In practice the check values won't be exactly 1 (at least for parameters that contribute to the theory function non-linearly). However, the values should be near to 1, and the check values of one parameter should be approximately equal. If this is not the case, you should at least be suspicious about the calculated errors, because then the assumptions of the error calculation seem to be not true. If one of the check values is even negative, the fit has not finished in a minimum but at best in a saddle point. In this case you should reject all results and try other start values for the parameters.

$\mathcal{M}$ To investigate the topology of the merit function near its minimum one again replaces the theory function with a linear Taylor approximation. This time the expansion point is the minimizing parameter vector $\mathbf{b}_0$:

$$f(x_i;\mathbf{b}_0 + \Delta\mathbf{b}) = f(x_i;\mathbf{b}_0) + \sum_j \frac{\partial f(x_i;\mathbf{b}_0)}{\partial b_j}\Delta b_j\,. \tag{7r}$$

Then the merit function $\Phi(\mathbf{b})$ can be written as

$$\Phi(\mathbf{b}_0 + \Delta\mathbf{b}) = \sum_i \left(\frac{y_i - f(x_i;\mathbf{b}_0)}{\Delta y_i} - \sum_j \frac{1}{\Delta y_i} \frac{\partial f(x_i;\mathbf{b}_0)}{\partial b_j} \Delta b_j \right)^2 \equiv \|\mathbf{g} - \mathbf{P}\Delta\mathbf{b}\|^2 , \quad (7s)$$

and the corresponding normal equations degenerate to $\mathbf{P}^T\mathbf{g} = 0$. Now one considers the change of $\Phi(\mathbf{b})$ if one moves away from the minimum by a step $\Delta\mathbf{b}$:

$$\begin{aligned} \Phi(\mathbf{b}_0 + \Delta\mathbf{b}) - \Phi(\mathbf{b}_0) &= \|\mathbf{g} - \mathbf{P}\Delta\mathbf{b}\|^2 - \|\mathbf{g}\|^2 \\ &= \mathbf{g}^T\mathbf{g} - \mathbf{g}^T\mathbf{P}\Delta\mathbf{b} - \Delta\mathbf{b}^T\mathbf{P}^T\mathbf{g} + \Delta\mathbf{b}^T\mathbf{P}^T\mathbf{P}\Delta\mathbf{b} - \mathbf{g}^T\mathbf{g} \\ &= \Delta\mathbf{b}^T\mathbf{P}^T\mathbf{P}\Delta\mathbf{b} . \end{aligned} \quad (7t)$$

Thus displacements $\Delta\mathbf{b}$ just resulting in a change $\Delta\Phi = 1$ obey the ellipsoidal equation $\Delta\mathbf{b}^T\mathbf{P}^T\mathbf{P}\Delta\mathbf{b} = 1$. For the determination of the check values one uses displacements whose direction is given by the column vectors of the covariance matrix $(\mathbf{P}^T\mathbf{P})^{-1}$:

$$\Delta\mathbf{b}_i = \lambda(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{e}_i . \quad (7u)$$

Here $\mathbf{e}_i$ denotes a unity vector directed along the i^{th} axis in the parameter space. The scaling factor λ can be obtained from the ellipsoidal equation:

$$1 = \lambda^2\mathbf{e}_i(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{e}_i = \lambda^2(\mathbf{P}^T\mathbf{P})^{-1}_{ii}. \quad (7v)$$

Thus λ is simply the inverse of the standard deviation of the i^{th} parameter. The check values calculated by PC-Mos II are the actual changes of the merit function Φ under the displacements $\Delta\mathbf{b}_i$ given by $(7u)$:

$$C_i^{\pm} \equiv \Phi(\mathbf{b}_0 \pm \Delta\mathbf{b}_i) - \Phi(\mathbf{b}_0) . \quad (7w)$$

Clearly, the check values take on the value 1 if the Taylor approximation $(7r)$ is exact.

7.4 Quality Control

One often wants to have some quantitative measure helping decide the question whether a model is correct or not; or, in other words, the question whether the inevitable deviations of the data from the theory curve only are due to statistical variations or also to an incorrect model. PC-Mos II calculates two such measures, the statistic χ^2 and the correlation of subsequent differences between data and theory. These measures are transformed into some kind of probabilities, which can be understood more intuitively. However, these probabilities can also be misunderstood. A correct definition of their meaning is the following: *If one designed a statistical test deciding whether a model is correct or not on the basis of one of these measures, and if he laid the decision limit exactly onto the observed value of this measure, then the computed probability would be the probability that this test rejects a model as incorrect although it is in fact correct.* (If you are familiar with the statistical theory of testing hypotheses, you will recognize

these probabilities as significance probabilities.) Unfortunately, some easier definitions for such probalities you might encounter are almost always wrong. This particularly is true for statements like "probality that the parameters are correct" or "probability that there is no better model". It should be clear that such probabilities, in a mathematical sense, cannot exist.

The two significance probabilities PC-Mos II calculates are output as percentage in the result files with format `.res`, tagged with the names `Chi2-Test` and `Corr-Test` respectively. If it is not clear from above definition: the larger the calculated probabilites, the better the model.

7.4.1 The χ^2-Test

$\mathcal{M}$ The merit function Φ defined by $(7a)$ is a random variable because it depends on the measured countrates y_i, which are also random. Assuming that the model is correct, Φ is distributed according to the χ^2-distribution with $(N-M)$ degrees of freedom. Here N is the number of data points and M is the number of free parameters. The density function of this distribution is

$$f(x) = \frac{1}{\Gamma\left(\frac{N-M}{2}\right)2^{(N-M)/2}} x^{(N-M)/2\ -1} e^{-x/2}. \tag{7x}$$

The mean of this distribution is $N-M$ and its variance is $2(N-M)$. Therefore the expectation value of the quantity $\chi^2 \equiv \Phi(y_i;\mathbf{b})/(N-M)$ is 1. The probability that the merit function takes on a value exceeding the observed value Φ is given by

$$P(\Phi) = \frac{1}{\Gamma(\frac{N-M}{2})2^{(N-M)/2}} \int_{\Phi}^{\infty} dx\, x^{(N-M)/2\ -1} e^{-x/2}. \tag{7y}$$

This is the probability calculated by PC-Mos II. It is the probability that a test rejecting a model if the merit function exceeds the value Φ fails to accept a correct model.

7.4.2 The Correlation Test

$\mathcal{M}$ For a correct model the residuals of subsequent data points from the theory curve are not correlated. I.e., if one data point lies above the theory curve, it is equally likely that the next data point lies above or below the curve. To check if this is true PC-Mos II calculates the linear correlation coefficient (*Pearson's r*) of the residuals of subsequent data points:

$$r = \frac{\sum_{i=1}^{N/2} \big(y_{2i-1} - f(x_{2i-1};\mathbf{b})\big)\big(y_{2i} - f(x_{2i};\mathbf{b})\big)}{\sqrt{\sum_i \big(y_{2i-1} - f(x_{2i-1};\mathbf{b})\big)^2}\sqrt{\sum_i \big(y_{2i} - f(x_{2i};\mathbf{b})\big)^2}}. \tag{7z}$$

If the residuals are not correlated, the correlation coefficient $(7z)$ for large N is normally distributed with mean 0 and variance $2/N$. Thus the probability that, for a correct

model, the modulus of the correlation coefficient exceeds the observed value $|r|$ is given by

$$P(r) = \mathrm{erfc}\Big(\frac{|r|\sqrt{N}}{2}\Big).$$

This is the probability issued by PC-Mos II. Again it is the probability that a test rejecting a model if the modulus of the correlation coefficient ($7z$) exceeds the value $|r|$ fails to accept a correct model.

7.5 Monte-Carlo Simulations

In order to check the reliability of the error estimation discussed in section 7.3 PC-Mos II provides the option `Monte-Carlo` of the *Fit Menu*. If you invoke this option before starting the fit, PC-Mos II replaces the measured data points with simulated ones, which are normally distributed around the corresponding values of the theory curve with the same variance as that of the measured data points. If you fit the theory curve to the simulated points, the resulting parameter values will differ from those obtained from fitting the measured data. You can compare the magnitude of the deviations with the errors calculated by PC-Mos II.

In practice you will want to proceed as follows: After the fit of the measured data select the option `Upgrade-Parameters` of the *Dispose Menu* in order to convert the fitted parameter values into start values for a new fit. (You should save the fit results before that because otherwise they will be lost.) Then again invoke the *Fit Menu* with `Weber-Fit`, but before starting the fit now select the option `Monte-Carlo`. After having output the results you can compare the differences between the start values (which are the results from the data fit) and the results with the error estimates.

8 The Options of the Edit and the Dispose Menu

8.1 Data Parameters

With the option `Data` of the *Edit Menu* miscellaneous variables belonging to the measured data can be modified or input manually. When you invoke this option, a new menu line is displayed, in which these parameters are listed. After you have selected one of these parameters, its current value is displayed and PC-Mos II prompts for a new value. If you don't want to change it, you can just press `Return`, which will leave the current value unchanged. Some of these parameters may be input before any data has been read. If the same parameter is also contained in a data file read later, PC-Mos II always uses the latest value. Thus the former value, that has been input manually, is overwritten.

With `Header` the header of the data can be entered or modified. This is the text displayed on top of the graphics window. The folding point may be entered with `Foldingpoint`. Note that your input will be rounded off to an integral or half-integral channel number. With `Mode` you can toggle the folding mode between sinusoidal and triangular. Which folding mode is currently active can be seen from the letter `S` or `T` that is displayed behind the folding point. Normally PC-Mos II is in sinusoidal mode when it is started, but you can change this with the installation program MosSetup. With `Velocity` you can enter the maximum velocity of the Mössbauer drive, whereas `Sign-of-velocity` only changes the sign of this velocity (to move the folding point to the other half period; see sect. 4.2). The options `Foldingpoint`, `Mode`, `Velocity` and `Sign-of-velocity` only are available if the spectrum is not folded.

When PC-Mos II reads data from a file, it must know the number of channels before it comes to read the first countrate (for reasons of memory management). Normally the data files specify this number before the first countrate appears. However, if, for any reason, you have to deal with data formats that do not contain this information, you can use the option `Channels` to enter this number manually before reading the data file.

8.2 Fit Parameters

The option `Parameters` of the *Edit Menu* facilitates modifying parameters of the fit routine. In the new menu line appearing after you have invoked this option sometimes several such parameters are summerized by one option. In this case PC-Mos II prompts for the single parameters subsequently. Again, pressing `Return` without input leaves the current value unchanged.

With `Header` you can enter or modify the header of the parameters. This header is the text in the first line of the parameter files with format `.par`. It only is displayed on screen if no data header is available.

The name of the theory program can be entered with `Theory`. This name may include a path and/or a file name extension, but if it does, PC-Mos II cannot recognize

an integrated theory any longer. If both an integrated and a standalone version of a theory program exist, you can use this feature to force PC-Mos II to use the standalone version. Normally PC-Mos II would use the integrated one. The description of the theory manager MkTheory in the appendix contains more information about integrated and standalone theories.

The option `Criterium` facilitates the modification of the fit stop criteria. This stop criterion is specified in form of a logical expression containing the fit flags discussed in section 7.2.3. The expression uses the so-called *reverse Polish notation* (RPN), which has been used because its interpretation is much easier to implement on a computer than the interpretation of a normal (*infix*) expression.

C Because it is not commonly known, it is useful first to discuss the RPN in some detail. An essential feature of this notation is that an oparator connecting two operands doesn't stand between but behind these operands. The operands can be either "elementary" or again a RPN (sub-)expression. The notation is unambigous if only the number of operands an operator connects is fixed. The RPN needs no parentheses; an operator always affects the operands standing immediatelly before it. Probably the RPN is best understood by studying some examples. In the following examples lower case letters serve as operands, `+` and `*` as binary operators and `-` as unary operator:

Infix	RPN
`-a`	`a-`
`a+b`	`ab+`
`a+(-b)`	`ab-+`
`(a+b)*c`	`ab+c*`
`a+(b*c)`	`abc*+`
`a+((-b)*c)`	`ab-c*+`
`(a*b)+(a*c)`	`ab*ac*+`
`(a*b)+(-(a*c))`	`ab*ac*-+`
`((a*b)+(b*(a+(-(c+d))+b)))*e`	`ab*bacd+-b++*+e*`

If you are familiar with the concept of a stack, you might like following "computer instruction": Whenever you encounter an elementary operand, push it onto the stack. Whenever you encounter an operator pop as many operands as needed, connect them accordingly and push the result back onto the stack. The result of the expression is the value finally remaining on the stack.

C Possible operators in the logical expression for the fit stop criterion are the binary logical *and*, denoted by `&`, the binary logical *or*, denoted by `|` and the unary logical negation, denoted by `!`. The elementary operands have the form Fn, where F stands for one of the fit flags (see sect, 7.2.3) and n is a number between 1 and 9. Such an operand yields the logical value *true* if the fit flag F has been set during the n preceeding fit iterations. Otherwise it yields the value *false*. The fit is terminated if the evaluation of the logical expression yields the value *true*.

C If you don't change it, PC-Mos II uses the expression `U1I1|P1|C2` as stop criterion. (It is useful to compare this with the formulation in section 7.2.3.) An example for a more complicated stop criterion is `U1I1C1!&|P1C1&|C2|`, whose infix equivalent is `U1`

`| (I1 & !C1) | (P1 & C1) | C2`, and that terminates the fit if one of the following conditions applies:

- The user tried to terminate the fit with `Finish`,
- the number of iterations has exceeded its limit and the change of χ^2 has been significant during the preceeding iteration,
- the change of χ^2 and the change of the parameters have not been significant during the preceeding iteration, or
- the change of χ^2 has not been significant during the two preceeding iterations.

With the option `Significance` you can modify the significance limit for changes of χ^2, the significance limit for changes of the parameters and the maximum number of iterations (all these parameters are discussed in sect. 7.2.3). The start value of Marquardt's parameter λ (see sect. 7.1) can be set with `Marquardt`.

A slight modification of the Marquardt-Levenberg fit method goes back to Markus Weber. This modification doesn't reduce Marquardt's parameter λ statically by a factor 10 after each successful iteration but by a factor depending on how χ^2 changed during the preceeding iterations. If χ^2 has changed drastically, λ is reduced only slightly; if χ^2 has not changed very much, λ is reduced faster. This modification sometimes improves the convergence of the fit. The method has two parameters that can be set with the option `Weber`. The first parameter, the relaxation factor, specifies the the minumum factor by which λ is reduced after each iteration. The number of preceeding changes of χ^2 that are taken into consideration is controlled by the second parameter. If this number is zero, the original static Marquardt-Levenberg algorithm is retrieved.

The option `Data-Step` provides a possibility to reduce the memory requirements of the fit at the expense of increasing the computing time. The parameter you can modify with this option specifies the maximum number of data points that are calculated by one execution of a theory program. By this you can reduce the number of columns of the derivation matrix, which has the main part of the memory requirements. If this number is smaller than the number of data points of one spectrum, the theory program must be executed more often in order to calculate all points. In the original configuration this number is 32767, which is large enough to allow the calculation of all data points by one theory call in any case.

When PC-Mos II reads parameters from a file, it must know their number before it encounters the first of them. Normally the parameter and result files specify this number before the first parameter appears. However, if, for any reason, you have to deal with file formats that do not contain this information, you can use the option `Number` to enter this number manually before reading the file.

8.3 Simulation Parameters

Some parameters that control the way PC-Mos II calculates simulations can be modified with the option `Simulation` of the *Edit Menu*. The menu line appearing after the invocation of this option contains the names of these parameters. As before, if after having selected one of these parameters you decide not to change its value, you can press `Return` without input.

The parameter `Resolution` specifies the number of points calculated with the simulation. The original configuration sets the value 152, which is sufficient for simple screen graphics. However, you will want to increase this number if you intend to produce high quality graphics by outputting the simulation data in a file (e.g. with the format `.plt`) and using a commercial graphics program to process this data. The velocity range for the simulations is always identical with the velocity range of the screen graphics. Therefore this range is treated as graphics parameter, and its modification is discussed in section 8.4.

The parameter `Non-Equidistancy` controls how the density of the calculated simulation points varies over the velocity range. If this parameter has a positive value (up to 2), the density is increased in the middle of the velocity range (because here normally most of the information is contained). The value 0 yields equidistant points, whereas a negative value increases the density at the edges. The formala that is used for the velocity map is discussed in section 6.1.

With the parameter `Subspectra` you can set an upper limit for the number of subspectra PC-Mos II expects a theory program might calculate in order to reduce memory requirements. Be careful with setting this parameter to a too small value. Its implications are discussed in section 6.1.

8.4 Graphics Parameters

The option `Graphics` of the *Edit Menu* provides access to parameters concerning the graphics display of PC-Mos II. The menu line invoked by this option contains the names of these parameters.

The first parameter of this kind is the displayed velocity range, which can be modified with the option `Range`. PC-Mos II will prompt for the minimum and the maximum displayed velocity. The value 0 for the minimum velocity is special: It results in a velocity range symmetric to zero, i.e., the minimum velocity is actually the negative of the maximum velocity. Note that the displayed velocity range also affects the option `Cut-off` of the *Process Menu* and serves as range for the velocity map of simulations.

The vertical display range is determined automatically from the highest and the lowest data or simulation points. However, you can modify the space between these extreme points and the margins of the window with the parameter `Overhang`, which specifies this gap as percentage of the range between lowest and highest point. If you set this parameter to zero, the highest and the lowest point will coincide with the upper and the lower margin respectively.

PC-Mos II displays data points as short vertical lines. The parameter `Bars` specifies the length of these lines in screen pixels. If you prefer points instead of lines, you will want to set this parameter to one. The value zero is special: Then PC-Mos II displays the data points as error bars with a length given by the standard deviation of this data point scaled according to the current display range. If measurement errors have not been read explicitely, PC-Mos II assumes that the errors are given by the square root of the countrates, according to the Poisson distribution.

With the option `Captions` you can modify the captions of folded spectra and simulations. PC-Mos II subsequently prompts for the caption of the x-axis and for two alternative captions of the y-axis, one that is used for absolute display and one for relative display. You can toggle between these two types of display with the option `Type-of-display`. However, the selection of the relative display only is possible if a baseline (see sect. 5.3) is available, since relative means relative to the baseline. Note that the display of not folded spectra is fixed. You can neither modify the captions of not folded spectra nor use a relative display type.

8.5 Invoking an Editor

PC-Mos II provides the possibility to execute an editor program of your choice without terminating PC-Mos II. The name of this editor must be specified with the installation program MosSetup (see appendix). If you select the option `File` of the *Edit Menu*, PC-Mos II will prompt for a file name and call the editor with this file name as command line argument. (If you press `Return` without input, the editor will not be executed.) Concerning file names, everything that has been discussed in section 3.1 also holds for the option `File` of the *Edit Menu*. In particular, there is a list with default suffixes also for this option, which comes into force if you don't specify an extension with the file name. If the list has more than one entry, the editor is executed several times subsequently. In the original configuration this list only contains the suffix `.par`.

After each execution of the editor PC-Mos II checks if the name of the edited file and one of the file names displayed on the screen match. If this is the case, PC-Mos II offers to reload this file. If you press `Y`, the file will be read as if you had selected the option `Read` of the file menu. The offer to reload a file is not issued when PC-Mos II is processing a macro file.

8.6 The Options of the Dispose Menu

The *Dispose Menu* provides options for deleting and rearranging data and parameters. The option `Data` deletes the data points (folded or not), the velocity map and the measurement errors. The flag `D` and, if necessary, the flag `F` are reset (see sect. 2.1).

The option `Parameters` deletes the start and the fitted values of the parameters of the theory curve as well as their names, the errors and the check values. PC-Mos II resets the flags `P` and `R`. Simulation data is deleted with the option `Simulation-Data`. The flag `S` is reset.

With the option `Upgrade-Parameters` you can convert fitted parameter values into start values for a new fit. The old start values are overwritten and the errors and check values are deleted. The flag `R` is reset.

9 The Options of the Printer Menu

9.1 Graphics Print-outs

With the options `Tiny-Dump`, `Medium-Dump` and `Huge-Dump` you can print out the graphics window of the screen on a matrix printer. The three options only differ in the size of the print-out. You can stop a running print-out by pressing `Ctrl` `Break`. A graphics print-out is not possible if neither data nor simulation data is available.

In the original configuration the print-out formats are adapted to an EPSON LQ printer and produce best results if a VGA graphics adapter is used with a resolution of 640×480 pixels. You can easily adapt the formats to an EGA or a Hercules adapter with the installation program MosSetup. It is also possible to use printers not completely compatible to the EPSON LQ, but the adaption is a bit more complicated. For details you are refered to the description of the installation program MosSetup in the appendix.

9.2 Formatted Printer Output

With the option `Print` of the *LPT Menu* you can produce formatted print-outs of data, parameters and fit results. As it is with the options `Read` and `Write` of the *File Menu*, the format is specified by a file name extension. Since a file name is superfluous in this case, PC-Mos II only prompts for a file name suffix. You can use the same formats as for file I/O (e.g. the enclosed formats `.dat`, `.dta`, `.par`, `.res` or `.plt`) but for print-outs of fit results a separate format is provided, which uses appropriate margins and script types. This format is specified with the suffix `.prs`. Before using this format, however, you should adapt it to your printer. This is done by editing the file `pcmosfmt.prs`, which can be found in the directory `\pcmos`. You will find two clearly marked lines, which should be adapted to your printer type.

C Formatted printer output works completely analogously to the input and output of files (see sect. 3.1). Internally the same I/O routine is invoked too. Thus you can create new formats for the formatted printer output in the same way as for file I/O. This is discussed in chapter 10.

9.3 Further Options

You can not only output course records to files, as described in section 3.2, but also directly to the printer. For this purpose the options `Open-Log` and `Close-Log` of the *LPT Menu* are provided. After you have invoked the option `Open-Log`, all input and output appearing in the message window on the screen is also output to the printer. The course record is finished with `Close-Log`.

With the option `Redirect` you can redirect all printer output to a file. Also graphics print-outs are included. If you invoke this option, PC-Mos II will prompt for a file name and send all output normally going to the printer to this file; the printer is accessed no more at all. Once you have set a printer redirection, it will remain active until you finish PC-Mos II. The corresponding file is closed automatically. Later you can use the DOS command `copy` to send the output to the printer. (You have to use the option `\B` with `copy` since the redirection file is binary.)

Normally PC-Mos II initializes the printer port when the program is started. If you want to avoid this, you can set up a permanent printer redirection with the installation program MosSetup. Then all printer output will be sent to a file with the name `PCMOS.LPT`. In this case the option `Redirect` will only use another file.

A linefeed or formfeed code can be sent to the printer with the options `LF` or `FF` respectively. I.e., the printer paper is moved forward by one line or one page respectively. (If a printer redirection is active, also these codes are output to the corresponding file.)

10 Programming Formats

C As already mentioned in section 3.1, before the actual input or output of a file (and also before a formatted printer output) PC-Mos II reads a format program that is contained in a file with the name `pcmosfmt` and a file name extension that identifies the desired format. This concept enables you both to modify the enclosed formats and to create new ones, as many as you like. However, programming of formats requires some familiarity with the concepts of structured programming and with some basic elements of the programming language C (namely with the format syntax of the `printf` standard library function). This will be assumed in the following sections.

C It is convenient to divide the data stream of the input and output of files into three logical levels. On the lowest level PC-Mos II reads or writes single characters (ASCII characters). The next level combines these characters to *tokens* and space, whereas on the highest level the tokens are converted into values of variables. The highest level is discussed in the next section, section 10.2 deals with the other levels.

10.1 PC-Mos II's Variables

C From the point of view of PC-Mos II's I/O routine all accessable variables are two-dimensional arrays, which, however, may degenerate to one-dimensional arrays or single variables if the corresponding array indices are restricted to a single value. The lowest (and perhaps the only) valid value of an array index is zero. The number of valid values is either one or the value of another variable. Each variable of PC-Mos II is assigned two own indices, which can be manipulated independently.

C The data type of the array elements is either an integral number (int), a floating point number (float) or a character array (string). Tokens are converted into integral or floating point numbers according to the syntax rules of the programming language C. However, this syntax is easy and almost indentical in most high-level programming languages, and therefore is not discussed further here. The conversion of tokens to strings is simply one-to-one. Only if the token is longer than the maximum length of the string, the token is truncated.

C The variable `attribute`, which is an one-dimensional array containing the parameter attributes (see sect. 7.2), is special in this respect. Its elements can take on three (integral) values corresponding to the attributes `standard`, `common` and `fixed`. The corresponding tokens are `" "` (a single blanc), `"C"` and `"F"` respectively (of course without the quotation marks).

C After a token has been converted, the obtained value is written to the array element that is currently addressed by the two indices assigned to the variable. The value of the indices is not changed automatically; there are special instructions for this purpose. If the value of one of the indices is beyond its valid range, PC-Mos II issues an according error message and terminates the I/O.

C The following table lists all accessable variables of PC-Mos II together with their data types and index ranges. In the descriptions you are sometimes refered to the sections in which the meaning of the variable is explained. The flags also listed in the table are discussed in the next section.

Name	Type	Index Ranges	Flags	Description
`dataheader`	string	1	1	Header of the data (8.1)
`paraheader`	string	1	1	Header of the parameters (8.2)
`extinfo`	string	8	1	External information; transparent to PC-Mos II
`theory`	string	1	1	Name of the theory program
`channels`	int	1	1	Number of channels of a not folded spectrum
`ndata`	int	1	1	Number of data points of a folded spectrum
`npara`	int	1	1	Number of parameters of the theory curve

nsubspect	int	1	1		Number of sub-spectra of a simulation including the envelope (6.1)
simpoints	int	1	1		Number of points of a simulation
foldpt	float	1	1		Folding point of the spectrum (4.2)
velocity	float	1	1		Maximum velocity of the Mössbauer drive
vmin	float	1	1		Minumum displayed velocity
vmax	float	1	1		Maximum displayed velocity
ymin	float	1	1		Minumum displayed countrate
ymax	float	1	1		Maximum displayed countrate
geomeff	float	1	1		Geometry effect of the spectrum in % (5.2)
effect	float	1	1		Effekt of the spectrum in % (5.1)
fitdate	string	1	1		String containing time and date of the fit
fitflags	string	1	1		String containing the fit flags (7.2)
fitspectra	int	1	1		Number of simultaneously fitted spectra (7.2)
stopcrit	string	1	1		String containing the fit stop criterion (8.2)
maxiter	int	1	1		Maximum number of fit iterations
niter	int	1	1		Actual number of fit iterations
parasig	int	1	1		Parameter significance (7.2)
chi2sig	int	1	1		χ^2-significance (7.2)
nchi2	int	1	1		Number of preceeding iterations considered for the determination of a new λ (8.2)
relax	float	1	1		Relaxation factor for the determination of a new λ (8.2)
lambda	float	1	1		Start value of Marquardt's λ (7.1)
chi2	float	1	1		Value of the quantity χ^2 (7.2)
chi2test	float	1	1		Probability of the χ^2-test in % (7.4)
corrtest	float	1	1		Probability of the correlation test in % (7.4)
counts	float	channels	1	D	Countrates of a not folded spectrum
data	float	ndata	1	DF	Countrates of a folded spectrum

derror	float	ndata	1	DF	Measurement errors of a folded spectrum
velmap	float	ndata	1	DF	Velocity map of a folded spectrum
simvelmap	float	simpoints	1	S	Velocity map of a simulation
simdata	float	nsubspect	simpoints	S	Simulation points of the envelope (1st index = 0) and of the sub-spectra (1st index > 0)
initpara	float	npara	1	P	Start values of the parameters of the theory curve
finalpara	float	npara	1	PR	Fitted values of the parameters of the theory curve
perror	float	npara	1	PR	Statistical errors of the parameters (7.4)
check1	float	npara	1	PR	1st check values of the parameters (7.4)
check2	float	npara	1	PR	2nd check values of the parameters (7.4)
attribute	-	npara	1	P	Attributes of the parameters (7.2)
paraname	string	npara	1	P	Names of the parameters

10.2 Format Statements

10.2.1 Syntax of Format Programs

C A format program consists of statements and comments. Comments start with the sign % and end with the end of the line. Apart from that line feed characters are treated exactly like blancs and tabulators (so-called *whitespace* characters). Statements consist of the name of a instruction (the *opcode*) and of a variable number of arguments. The first argument is separated from the opcode by whitespace characters; the arguments are separated from each other by commas. Each statement ends with a semicolon (;). Since line feed characters are treated as whitespace, a statement can extend over more than one line.

C The arguments of statements can be names of variables, integral numbers or strings. Strings must be specified using the syntax of the programming language C. In particular they must be bracketed by double quotation marks ("); note also the special meaning of the backslash \ as escape character (see any text on C for details). Most statements allow some of their arguments being omitted. They are replaced with specific default values. However, the commas may not be omitted; only if the missing argument is the last of the argument list, the statement can simply be finished early with the semicolon.

10.2.2 Management Statements

- `announce` ⟨*flagstring*⟩,⟨*I/O-mode*⟩;

C▌ In order not to impose unnecessary restrictions on the number of data points or parameters PC-Mos II manages the memory for all arrays dynamically. It is a consequence of this memory management that PC-Mos II must know in advance, which arrays are contained in a file it is about to read. Therefore the first statement in every format program is an `announce` statement which contains this information: ⟨*flagstring*⟩ is a string consisting of flags as they are listed in the table of section 10.1. (You have probably noticed that these flags are closely related to the spectrum flags displayed on the screen.) The rules are as follows: For every variable that will appear in the format program (and thus in the file to be read or written) the string ⟨*flagstring*⟩ must contain the flag sequence that is listed in the table with this variable. E.g., if you have the arrays `counts` and `ipara` in your format, ⟨*flagstring*⟩ must at least be `"DP"`. Additionally, variables that require the flag `"D"` and variables with the sequence `"DF"` must not appear in the same format file.

C▌ If PC-Mos II is about to read a file, old data stored in variables that are announced with the `announce` statement is lost since PC-Mos II expects new data for this array to be contained in the file to read. Note that as a consequence you cannot input arrays with the same flags in more than one step from several files. Of course the data is not lost if you write it to a file.

C▌ The string ⟨*I/O-mode*⟩ may contain the letters `R` and/or `W` standing for read and write respectively. This string tells PC-Mos II whether the format is intended for the input or the output of data, or for both.

- `alloc;`

C▌ With the statement `alloc` you can control when PC-Mos II allocates memory for the dynamic arrays when it reads data from a file. (Dynamic arrays are all variables that must be announced, i.e., all variables having some flags listed with them in the table of section 10.1.) None of these variables may appear before this statement. On the other hand, after the `alloc` statement none of the variables that serve as index range (i.e., that appear in the third or fourth column of the table) may appear any more. The `alloc` statement only has consequences if PC-Mos II reads a file. However, it must be contained also in write-only formats.

10.2.3 Transfer of Data

- `transfer` ⟨*variable*⟩,⟨*length*⟩,⟨*separators*⟩, ⟨*format*⟩,⟨*warn*⟩;

C▌ With this statement you can read one array element of the variable with the name ⟨*variable*⟩ from the file (or write it to the file respectively). The effect of the arguments of this instruction depends on whether PC-Mos II reads or writes a file. First the input is discussed:

C▌ ⟨*length*⟩ is the maximum number of characters that PC-Mos II reads and thus the maximum length of the token to be converted into a value for the variable. If this

argument misses, the value 255 is assumed, which also is the maximum length of any token. ⟨*separators*⟩ is a string containing characters that separate the token to be read from the next one. If this argument is an empty string, or if it misses, PC-Mos II always reads ⟨*length*⟩ characters, otherwise perhaps less: The token is finished as soon as one of these characters is encountered. (There is one exception: Line feed characters always serve as separators; no token can extend over more than one line.) The character that separated one token from the next is not part of the token. I.e., PC-Mos II will read this character again with the next read operation.

C The argument ⟨*format*⟩ has no effect when PC-Mos II reads a file. The argument ⟨*warn*⟩ is a number that tells PC-Mos II how to proceed if it encounters a missing data item. A data item is considered as missing if the corresponding token only consists of blancs, or if the value resulting from the conversion is no valid value of the variable unter consideration. (String variables cannot "miss", they only are empty strings perhaps.) If ⟨*warn*⟩ is zero, the missing data item is simply ignored. If it is one, a warning is issued, but PC-Mos II proceeds with reading the file. The value two forces the termination of the input with an according error message. The argument ⟨*warn*⟩ can be omitted, then the value one is assumed.

C When PC-Mos II is writing data to a file, besides of ⟨*variable*⟩ only the argument ⟨*format*⟩ is of interest. It is a string specifying the output format of the variable. It does this in the same way as the format strings passed to the C standard library function `printf`, but the character specifying the data type must be replaced with the character `@`. An example for a format string is `" %8@ mm/s"`, which outputs the variable in a eight characters wide field, with a preceeding blanc and some additional text as unit. For details about `printf` format strings you are refered to a text on the programming language C.

- `dummy` ⟨*length*⟩`,`⟨*separators*⟩`,`⟨*format*⟩`;`

C This statement does the same as `transfer`, but the read token is not converted into the value of a variable but is simply ignored. If PC-Mos II writes a file, it outputs an empty string with the format ⟨*format*⟩. This instruction is useful to output some fixed texts and to skip them when reading such a file.

10.2.4 Manipulating the Character Stream

- `skip` ⟨*length*⟩`,`⟨*separators*⟩`;`

C With this statement you can skip the space between two tokens. PC-Mos II reads at most ⟨*length*⟩ characters that are contained in the string ⟨*separators*⟩. If a character not contained in ⟨*separators*⟩ is encountered before ⟨*length*⟩ characters are skipped, this character is put back to the input stream and will be read again by the next read operation.

C The statement `skip` has no effect if PC-Mos II writes to a file. As a result you have to create space separating the tokens by means of the format string ⟨*format*⟩ of the `transfer` statement.

- `newline;`

C This statement skips all characters remaining in the current line (including the line feed character). If PC-Mos II writes to a file, a line feed character is output.

- `push` ⟨*string*⟩`;`

C The characters of the string ⟨*string*⟩ are put back to the input stream. The following read operations will first read the characters from this string (from left to right) before again reading from the file. This facilitates the "reading" of values that are not contained in the data files actually. If PC-Mos II writes to a file, the statement has no effect.

- `unget;`

C The token most recently read by one of the instructions `transfer`, `dummy` or `switch` (see sect. 10.2.6) is put back to the input stream. Thus it can be read again by a following read operation. This, for example, facilitates assigning the same value contained only once in a file to two variables. If PC-Mos II writes to a file, the statement has no effect.

10.2.5 Manipulating the Indices

- `reset` ⟨*variable*⟩`,`⟨*index*⟩`;`

C One of the two indices assigned to the variable with the name ⟨*variable*⟩ is set to zero. Which index is affected is specified with the number ⟨*index*⟩, which may be one or two. If ⟨*index*⟩ is omitted, the first index is reset.

- `increment` ⟨*variable*⟩`,`⟨*index*⟩`;`

C One of the two indices assigned to the variable with the name ⟨*variable*⟩ is incremented. Which index is affected is specified with the number ⟨*index*⟩, which may again be one or two. If ⟨*index*⟩ is omitted, PC-Mos II increments the first index. It is valid if a index leaves its valid range when it is incremented. (This is necessary to control loops; see sect. 10.2.6) However, it is an error if the index is already beyond its range when PC-Mos II tries to increment it.

10.2.6 Control Structures

- `loop` ⟨*repetitions*⟩`,`⟨*variable*⟩`,`⟨*index*⟩`;`
  ⟨*statement*⟩
  ...
  `endloop;`

C The execution of the instructions between the statements `loop` and `endloop` is repeated. ⟨*repetitions*⟩ is the maximum number of repetitions. A second stop criterion for the loop can be specified with the arguments ⟨*variable*⟩ and ⟨*index*⟩. Again ⟨*index*⟩ can be either one or two, denoting the first or the second index assigned to the variable with the name ⟨*variable*⟩ respectively. The loop is finished if this index is found to be beyond its valid range. The stop criterion is checked at the beginning of the loop,

i.e., it may happen that the loop is not executed at all. Loops may be nested to any depth.

C If the argument ⟨*repetitions*⟩ is omitted, only the stop criterion specified by ⟨*variable*⟩ and ⟨*index*⟩ can finish the loop. If ⟨*variable*⟩ and ⟨*index*⟩ are omitted, the loop is always executed ⟨*repetitions*⟩ times. If ⟨*variable*⟩ is present but ⟨*index*⟩ is omitted, the first index is checked. Of course you can not omit both ⟨*repetitions*⟩ and ⟨*variable*⟩.

- `break;`

C The statement `break` forces the immediate termination of the innermost active loop. The execution of the format program is continued with the first statement after the corresponding `endloop` statement. Normally you will only use `break` together with following conditional statements.

- `switch` ⟨*length*⟩`,`⟨*separators*⟩`,`⟨*format*⟩`;`
  ⟨*case-block*⟩
  ...
  `endswitch;`

C With the `switch` statement you can make the execution of instruction blocks depend on the appearance of specific labels in the file. If PC-Mos II is reading from a file, `switch` reads a token that is specified by the arguments ⟨*length*⟩ and ⟨*separators*⟩ in the same way as with the statement `transfer`. This token is saved for later comparisons in `case` statements (see below). If PC-Mos II writes to a file, `switch` only saves the format string ⟨*format*⟩ for later outputs in `case` statements. `switch` itsself writes nothing to the file. The syntax for the format string is the same as with the statement `transfer`.

- `case` ⟨*label*⟩`,`⟨*variable*⟩`;`
  ⟨*statement*⟩
  ...
  `endcase;`

C If PC-Mos II encounters a `case` statement when reading from a file, it compares the string ⟨*label*⟩ with the token that has been read by the preceeding `switch` statement. (The comparison is case independent.) If the strings match, the following instructions are executed until the `endcase` statement. Then the remaining `case` blocks are skipped and the execution is continued with the first instruction after the `endswitch` statement. If the strings don't match, PC-Mos II jumps to the next `case` block, if available, or to the instruction following the `endswitch` statement.

C If the `case` statement is encountered when PC-Mos II is writing to a file, the following instructions are executed if the variable with the name ⟨*variable*⟩ has a valid value, i.e., is initialized. Before the instructions are executed, the string ⟨*label*⟩ is output using the format string that has been saved by the preceeding `switch` statement. When the `endcase` statement is reached, the next `case` block is checked. (This is different from the course when PC-Mos II is reading from a file!) If the variable is not

initialized, PC-Mos II immediatelly continues with the next `case` block. If the argument ⟨*variable*⟩ is omitted, the instructions of the `case` block are executed always.

- `default;`
  ⟨*statement*⟩
  ...
  `endcase;`

C The `default` statements specifies a special type of `case` block. When PC-Mos II is reading from a file, the instructions of the `default` block are executed always; when PC-Mos II is writing to a file, the instructions are executed never. Note that normally a `default` block should be the last `case` block within a `switch` block, since when PC-Mos II is reading from a file, it will always skip all subsequent `case` blocks.

C `switch` statements may also be nested. I.e., within `case` or `default` block another `switch` statement may occur. Of course the corresponding `endswitch` must appear before the `endcase` that finishes the `case` block.

- `end;`

C The statement `end` signals PC-Mos II the end of a format program. The remainder of the format file is ignored. Of course `end` must not appear within any `loop`, `switch` or `case` block.

C Now all format statements are complete. After you have read these descriptions it is probably useful if you try to understand the enclosed format programs, which include some comments that might help you. `pcmosfmt.dat` is the simplest of them. The next step is to try slight modifications of these programs, before you create a completely new format.

11 Creating Theory Programs

11.1 General Remarks

C A theory program is a program or subroutine that basically calculates a theoretical curve, given the values of some parameters. However, in order to enable PC-Mos II to fit this curve to a spectrum, to provide the options of the *Simulation Menu* and perhaps to allow data modification with the option `Modify` of the *Process Menu*, the theory program must do a little bit more: Besides calculating this theoretical curve, the envelope, it must be able to calculate its derivatives with respect to all parameters, it must calculate sub-spectra and perhaps modify data points in a reasonable way.

C Theory programs must be written in the programming language C, and in this chapter it is assumed that you have at least some basic knowledge of this language. In order to compile them you need one of C compilers of Borland's C++ series (Turbo C++ or Borland C++). However, the theory programs are not compiled directly with this compiler but with the enclosed theory manager MkTheory, which in turn invokes the command line compiler of Borland C++ with all options set correctly. The operation of the theory manager MkTheory is discussed in the appendix.

C A theory program necessarily needs some interface which manages the communication with the main program. However, you need not bother about this interfaces, it has already been prepared for you and is contained in the files `theory.h` and `theory.c`, which can be found in the PC-Mos II directory. If you intend to write a theory that only deals with Lorentzian lines or a transmission integral over Lorentzian lines, even more work has already been done for you in the files `lorentz.c` and `lorti.c` respectively. (E.g., you need not calculate the derivatives.) Designing theories for Lorentzian lines is therefore discussed in an own section (sect. 11.3). You are strongly adviced not to make any changes in the files `theory.h` or `theory.c`. However, modifying `lorentz.c` or `lorti.c` is not critical and perhaps useful to serve your needs.

C There are some general rules which you should obey if you want your theories to work properly. One of them is that all global variables and all functions in a theory program must be declared `static` (i.e., with the key word `static` preceeding their declaration). That the theory must do a complete clean-up before returning (e.g., release all allocated memory, close open files) is another principle. You should not expect the operating system to do this for you (as it does with normal programs). Moreover, if any error occurs during the execution of the theory program, it should not terminate with the C standard library functions `abort`, `exit` or `_exit` since this could leave the system in an unstable state (with some interrupt vectors pointing to nowhere). For this purpose the file `theory.c` provides the function `error` with the prototype

```
static void error (int code, char *msg),
```

which terminates the theory properly. With `code` and `msg` you can pass some information about the reason for the termination to PC-Mos II: If `code` is not zero, PC-Mos II issues the warning `Non-zero return code from theory:` followed by the value of

`code`, and if the string `msg` is not empty, PC-Mos II issues the error message `Error in theory:` followed by the message you have passed. An example:

```
error(0,"Senseless parameter values");
```

A last rule concerns the screen output of messages. Since PC-Mos II normally is in graphics mode when executing a theory, you should not output text directly to the screen without any precautions. However, sometimes you will need to output some text in order to debug a theory. One possibility is to write it to the printer. However, the installation program MosSetup provides the possibility to make PC-Mos II switch to text mode whenever a theory is executed. Then you can also write to the screen. (You should use this feature only temporarily since video mode switching wastes a lot of time.)

11.2 Theories for General Line Shapes

C The user written part of a theory program for general line shapes must always start with the preprocessor instruction

```
#include <theory.c>
```

which includes the interface to the main program, some data types and prototypes for three functions. All you have to do to create a theory program is to define these three functions, each of which has to carry out a specific task. In the following subsections these functions are discussed in detail.

11.2.1 The Function `derive`

C The function `derive` must calcalute the theory curve and some of its derivatives for a given set of velocities. It is called when PC-Mos II is fitting a spectrum. This is the prototype of this function:

```
static void  derive  (int      npara,
                      int      npts,
                      double   para[],
                      attr_t   pattr[],
                      double   vel[],
                      double   env[],
                      double  *drv[]);
```

`para` is an array with `npara` entries containing the values of the parameters that determine the theory curve. The array `pattr` with also `npara` entries holds the attributes of the parameters in form of symbolic constants: `A_STD` stands for the attribute `standard`, `A_COM` for the attribute `common` and `A_FIX` for the attribute `fixed`. For the theory program it only is of interest if the parameter is fixed or not, since this determines whether it must calculate the derivative with respect to this parameter (see below). The array `vel` has `npts` entries and contains the velocity values for which the theory curve and the derivatives must be evaluated. (Perhaps, at this point it is worthwhile to remind you that arrays in C always have zero offset, i.e., `vel[0]` and `vel[npts-1]` are valid indirections but `vel[npts]` is not.) These are the input arguments of the function `derive`. They are read-only in the sense that their values must be unchanged when the function returns.

C The function `derive` must do following jobs: First it must calculate the theory curve for all velocities in `vel` and write their values to the array `env` (which provides space for exactly `npts` values). Additionally, it must calculate the derivatives of the theory curve with respect to all parameters whose attribute is not `fixed`. The `npts` values of the derivative with respect to the n^{th} not fixed parameter must be written to the array `drv[n-1]`. (`drv` is an array of arrays: Its elements are accessed by expressions like `drv[n][m]`; this element should hold the value of the derivative with respect to the $(n+1)^{\text{st}}$ not fixed parameter evaluated at the velocity `vel[m]`.) Note that `derive` not only needs not, it also must not calculate derivatives with respect to parameters with attribute `fixed`, since `drv` doesn't provide memory for additional derivatives. It may (and will) also happen that all parameters passed to `derive` have the attribute `fixed`. In this case `derive` must only calculate the theory curve and no derivative. How the derivatives are calculated, numerically or analytically, is up to you.

11.2.2 The Function `subspectra`

C The function `subspectra`, which is called when PC-Mos II is calculating a simulation, must calculate the theory curve and at least one sub-spectrum of this curve for a given set of velocities. This is the prototype of this function:

```
static int  subspectra  (int      npara,
                         int      npts,
                         double   para[],
                         double   vel[],
                         double   env[],
                         double  *sub[]);
```

Again, `para` is an array with `npara` entries containing the parameter values, and `vel` is an array with `npts` entries that hold the velocities for which the theory curve and its sub-spectra must be evaluated. `subspectra` must write the values of the theory curve (the envelope) to the array `env` which provides space for `npts` entries. It also must calculate at least one sub-spectrum and write the values of the n^{th} sub-spectrum to the array `sub[n-1]`. I.e., the array element `sub[n-1][m]` must hold the value of the n^{th} sub-spectrum evaluated at the velocity `vel[m]`. At most `npara` sub-spectra may be calculated. The actual number of calculated sub-spectra must be returned as the functions return value (i.e., with a statement like `return 5;`). Note that the options of the *Simulation Menu* will only work properly if the relation (6*e*) (sect. 6.2) between envelope and sub-spectra holds.

11.2.3 The Function `modify`

C The function `modify`, which is called when the option `Modify` of the *Process Menu* is selected, may in any way modify the data points (only their ordinate) of a spectrum, according to the current values of the parameters of the theory curve. This is the prototype of this function:

```
static void  modify  (int      npara,
                      int      npts,
                      double   para[],
                      attr_t   pattr[],
                      double   vel[],
                      double   data[]);
```

Again, `para` is an array with `npara` entries that contains the values of the parameters of the theory curve. `pattr` is an array of the same size with the attributes of these parameters. (The same symbolic constants are used as for the function `derive`.) `vel` is an array with `npts` entries holding the values of the velocity, and the array `data` of the same size holds the corresponding countrates (or any other ordinate data). The function `modify` may modify the entries of the array `data` (and only the entries of this array) in any way it likes. It may also modify the values not at all, or even fail the job with a statement like `error(0,"Data modification not supported")`, like the enclosed theories (except for `Polynml`) do.

11.3 Theories for Lorentzian Lines

C If you intend to design a theory program that only deals with Lorentzian lines, many of the work necessary to write general theories has already been done for you. Essentially you only have to define three functions that return the number of Lorentzians the theory curve is built of, their parameters (depth, width and position) and their assignment to specific sub-spectra. The remainder of the work is done by the routines in the file `lorentz.c`, which must be included in your source file with the preprocessor directive

```
#include <lorentz.c>
```

therefore. This file containes the functions `derive`, `subspectra` and `modify` as described in section 11.2 which, however, delegate a part of their job to three easier functions `no_lrtz`, `mk_lrtz` and `subspec`, which you have to define, and which are described in the following subsections. You need and must not include the file `theory.c` in theories for Lorentzian lines since this is already done in the file `lorentz.c`.

11.3.1 The Function `no_lrtz`

C Given the parameters of the theory curve, this function must return the number of Lorentzians the curve is built of. This is the prototype of this function:

```
static int  no_lrtz  (int  np, double  p[]);
```

Here `p` is an array with `np` entries containing the current values of the parameters of the theory curve. This function is called once for each execution of the theory program.

In all subsequent calls to the functions `mk_lrtz` or `subspec` the number of parameters and Lorentzians will be the same. Also the values of fixed parameters will not change. Hence `no_lrzt` may do any initialization of global (static!) variables that depends on the number of parameters or on the values of parameters that are always fixed. (It must at least set a variable to the number of parameters, since this number is not passed to the functions `mk_lrtz` and `subspec` again.)

C The function may also do a plausibility check of the parameter values, and it should return zero instead of the (positive) number of Lorentzians if this check failed. Then the theory program will be terminated with the error message `Inconsistent parameters`.

11.3.2 The Function `mk_lrtz`

C Given the parameters of the theory curve, this function must calculate the depth, width and position of the Lorentzian lines the curve is built of. This is the prototype of this function:

```
static void  mk_lrtz  (double  p[], lrtz_t  l[]);
```

Here `p` is an array containing the values of the parameters. The size of the array is the same as it was when `no_lrtz` has been called the last time and is therefore not passed again. `l` is an array consisting of structures of the type `lrtz_t`, which is defined in the file `lorentz.c` as

```
typedef struct {
  double  lpos;
  double  wdth;
  double  dpth;
  double  area;
} lrtz_t;
```

The function `mk_lrtz` must set the fields `lpos`, `wdth` and `dpth` of these structures to the position, width and depth of the Lorentzians the theory curve consists of. (The field `area` is not used and is intended for temporary storage.) The number of elements of the array `l` is therefore equal to the number of Lorentzians, i.e., the number returned by the function `no_lrtz`. The routines in the file `lorentz.c` calculate Lorentzian lines according to the formula

$$f(x) = \frac{I_0}{1 + \left(\frac{x - x_0}{\Gamma/2}\right)^2}. \qquad (11a)$$

x_0 is the position, I_0 the depth and Γ the width of the Lorentzian.

11.3.3 The Function `subspec`

$\mathcal{C}$ Given the ordinal number of a sub-spectrum, this function must specify which of the Lorentzians calculated by `mk_lrtz` belong to this sub-spectrum. This is the prototype of this function:

```
static int  subspec  (int  n, int  subl[]);
```

Here `n` is the ordinal number of the requested sub-spectrum. If this number is larger than the number of sub-spectra the theory curve consists of, `subspec` must return zero to indicate that no more sub-spectra are available. Otherwise `subspec` must return the number N of Lorentzians belonging to this sub-spectrum and set the N first elements of the array `subl` to the indices these Lorentzians have had in the array `l` returned by `mk_lrtz`.

11.3.4 Transmission Integrals

$\mathcal{C}$ Once you have defined the functions `no_lrtz`, `mk_lrtz` and `subspec` you can easily extend the theory for Lorentzian lines to a theory calculating a transmission integral. The only thing you have to do is to include the file `lorti.c` rather than `lorentz.c` at the beginning of your source code:

```
#include <lorti.c>
```

The parameters of the Lorentzians returned by `mk_lrtz` are, however, now interpreted as the position, width and depth of Lorentzian lines in the resonance absorbtion cross section.

$\mathcal{C}$ Theories calculating a transmission integral expect four additional parameters. These parameters are, however, invisible to `mk_lrtz`, so you really don't need to make any changes there. The additional parameters are:

- The fraction of resonant quanta in the detected single channel window. In general, this fraction is not the Lamb-Mössbauer factor since it is reduced by resonant self-absorbtion in the source and by all kinds of background radiation.
- The line width of the source spectrum, which is assumed to be Lorentzian-shaped.
- The amount by which the actual integration range exceeds the velocity range of the spectrum, in units of the source line width. Beyond this range the transmission is assumed to be unity and the wing integration is performed analytically. A proven value is 5.0, but you should increase it, if the measured velocity range does not include all absorbtion lines. (This parameter contributes linearly to the computing time.)
- The number of knots for the numerical integration in an interval of one source line width. A proven value is 12 knots. (Doubling this number reduces the magnitude of the error of the numerical integration by a factor 2^{-6}, but increases computing time by a factor 4.)

$\mathcal{M}$ The routines in `lorti.c` calculate a transmission integral spectrum according to the formula

$$I(v) = I_0\left\{1 - f_S\left[1 - \int_{-\infty}^{\infty} dv' \, \frac{\Gamma_S/2\pi}{(\Gamma_S/2)^2 + (v - v')^2} \exp\left(-\sum_i \frac{d_i(\Gamma_i/2)^2}{(\Gamma_i/2)^2 + (v' - v_i)^2}\right)\right]\right\},$$

where I_0 is the baseline, f_S is the fraction of resonant quanta detected, Γ_S is the width of the source spectrum (in mm/s), and v_i, Γ_i and d_i are position, width and depth of the i^{th} Lorentzian in the resonance absorbtion cross section. The integration method is based on Cranshaw's technique described in [Cra] and [Lin]. Additional information concerning transmission integrals can be found in [Lan] and [She].

Appendix

A The Text Mode Version PCTMos

PCTMos is a text mode version of PC-Mos II. It provides all options of the full screen graphics version described in this manual, and its operation is exactly the same, but all graphics output is omitted. This saves about 35 KBytes of memory, which might be valuable for fitting several spectra simultaneously. Additionally, the text mode version is a little bit faster since graphics output is a relatively slow process on personal computers.

It is probably most convenient to operate PCTMos with macro files that have been tested with the graphics version. Such macro files will have exactly the same effects with PCTMos.

A small difference concerning course records arises from the fact that PCTMos issues some additional text output to compensate for some loss of information caused by omitting all graphics. This additional output will also appear in course records, which will look a bit different from those created by the graphics version therefore.

B The Installation Program MosSetup

MosSetup is a menu controlled installation utility that enables you to modify some of PC-Mos II's internal parameters. You invoke it by typing `mossetup` at the DOS prompt.

B.1 Operating the Menu System

The menu system of MosSetup provides four types of menus or windows. The first is the main menu, which is always displayed in the uppermost line of the screen. Options of the main menu are selected either by pressing the highlighted key or by moving the selection bar onto the desired option (with the cursor keys [←] and [→]) and pressing [Return] or [↓].

The options of the main menu invoke either a pull-down menu or an input box. Options of a pull-down menu are selected either by pressing the highlighted key or by moving the selection bar onto the desired option (with the cursor keys [↑] and [↓]) and pressing [Return]. From a pull-down menu you can return to the main menu by pressing [Esc]. The cursor keys [←] and [→] invoke the pull-down menu or the input box left and right of the current one respectively.

Input boxes contain some parameters that can be edited. The parameters are selected by moving the selection bar forward and backward with the keys [Tab] and [Shift] [Tab] respectively. By pressing [↓] a line editor is invoked, which enables you to modify the selected parameter. (You can also simply start to enter a new value.) The line editor knows following commands:

[→]	Move cursor one character right
[←]	Move cursor one character left
[Ctrl] [→]	Move cursor one word right
[Ctrl] [←]	Move cursor one word left
[Home]	Move cursor to beginning of line
[End]	Move cursor to end of line
[Backspace]	Delete character left of cursor
[Del]	Delete character under cursor
[Ctrl] [T]	Delete word under cursor
[Ctrl] [E]	Delete from cursor to end of line
[Ctrl] [Y]	Delete whole line
[Ins]	Toggle between insert and overwrite mode
[Esc]	Cancel all changes and leave line editor
[Return]	Confirm all changes and leave line editor

The active line editor can be recognized by the blinking cursor, which appears instead of the selection bar. With [Esc] you can cancel all changes you have made in the input box (after a confirmation). To return to the main menu you press [Return]. If the input box has been invoked directly from the main menu, the cursor keys [←] and [→] invoke the pull-down menu or the input box left and right of the current one respectively.

Some of the options of pull-down menus invoke a further pop-up menu. Pop up menus are operated exactly as pull-down menus, but the cursor keys ← and → have no effect, and the menu is closed automatically after you have selected an option.

In the following sections all options of the main menu of MosSetup are discussed.

B.2 The Option Memory

The option `Memory` invokes an input box in which you can specify if and how much expanded and/or extended memory PC-Mos II may use for overlays. Normally PC-Mos II uses all available expanded memory and no extended memory.

The first parameter specifies the maximum number of 16 KB pages of expanded memory PC-Mos II may use. The usage of expanded memory is standardized, so you can safely set a large value here without the danger of interference with other programs.

The remaining parameters control the usage of extended memory. The first of them specifies how many KBytes above the 1 MB address PC-Mos II must reserve for other programs. The second determines the amount of extended memory in Kbytes PC-Mos II may use above this reserved range. Meanwhile also the usage of extended memory is standardized (at least if you have `himem.sys` loaded on your system), so you can also set a large value here.

B.3 The Option Graphics

The option `Graphics` invokes a pull-down menu with the options `Hardware`, `Size of bars`, `Display overhang` and `Captions`.

B.3.1 `Hardware`

Normally PC-Mos II detects your graphics hardware automatically and uses an appropriate video mode. If you think your graphics adapter is not recognized correctly, you can override this autodetection.

The option `Hardware` invokes a pop-up menu with the options `Autodetection`, `Select video mode` and `External driver`. If you select `Autodetection`, PC-Mos II will try to detect your graphics hardware. The option `Select video mode` invokes another pop-up menu, from which you can select a graphics mode for the EGA, VGA or Hercules adapter. Then PC-Mos II will set this mode without detecting the hardware. With `External driver` you can instruct PC-Mos II to use an external BGI file as graphics driver. This facilitates the usage of other graphics adapters—if you own a BGI driver for them. (Borland C++ provides some additional BGI files which can be used.) Selecting this option invokes an input box in which you must enter the name of the BGI file and the number of a graphics mode. (This number can normally be found in the documentation of the BGI driver. For the drivers provided by Borland C++ it is contained in the reference manual of the compiler.)

B.3.2 `Size of bars`

With `Size of bars` you can specify the size of the bars PC-Mos II uses to display data points. A pop-up menu is invoked with the options `Error bars` and `Fixed size bars`. If you select `Error bars`, PC-Mos II scales the bars according to the measurement errors. If you select `Fixed size bars`, an input box will be opened, in which you can enter the desired size in pixels.

B.3.3 `Display overhang`

Selecting the option `Display overhang` invokes an input box in which you can enter the value of the overhang parameter that determines the vertical display range. This parameter is explained in section 8.4.

B.3.4 `Type of display`

If you select this option, a pop-up menu appears, in which you can choose between absolute and relative display for folded data. Both types of display are explained in section 8.4.

B.3.5 `Captions`

The option `Captions` opens an input box in which you can modify the captions of folded spectra and simulations. The usage of the captions is explained in section 8.4.

B.4 The Option Fit

The option `Fit` opens an input box containing some parameters of the fit routine. These are the same parameters as provided by PC-Mos II's options `Criterium`, `Significance`, `Weber` and `Marquardt`, which can be found in the sub-menu `Parameters` of the *Edit Menu* of PC-Mos II. They are explained in section 8.2. The difference between modifying these parameters with PC-Mos II and MosSetup is that changes made during a run of PC-Mos II only concern this run. The next time you invoke PC-Mos II, the old values are restored. Changes made with MosSetup are permanent.

B.5 The Option Editor

The option `Editor` invokes an input box in which you can enter the name of your preferred editor. This editor will be called when you select the option `File` of PC-Mos II's *Edit Menu* as explained in section 8.5. Note that you also must make sure that this editor is in a directory that is listed in the DOS environment variable `PATH` (which is normally set by a `PATH` command in your `autoexec.bat` file).

B.6 The Option Suffixes

The option `Suffixes` opens an input box containing the default suffix lists for the options `Read` and `Write` of PC-Mos II's *File Menu* and for the option `File` of the *Edit Menu*. The meaning of these lists is explained in the sections 3.1 and 8.5. Note that the order of the entries in these lists might be important since if two subsequently read files contain some parameters twice, the latest value overwrites the previously read.

B.7 The Option Printer

The option `Printer` facilitates the modification of some printer parameters and a permanent printer output redirection. A pull-down menu is opened, whose options are discussed in following subsections.

B.7.1 `Destination`

The option `Destination` invokes a pop-up menu from which you can select a printer port or set a permanent redirection to the file `PCMOS.LPT`. The possibility to redirect the printer output to another file with the option `Redirect` from PC-Mos II's *LPT Menu*, however, still remains.

B.7.2 `Initialization`

The option `Initialization` opens an input box in which you can specify an initialization code that PC-Mos II sends to the printer before starting some text output. The code may be up to seven bytes long; the single bytes must be specified in hexadecimal notation. The end of the code is indicated by a zero byte.

B.7.3 `Format suffix`

The option `Format suffix` invokes an input box in which you can enter a file name suffix that identifies a format file. PC-Mos II will use this format file for formatted printer output with the option `Print` of the *LPT Menu* if you don't specify another format at PC-Mos II's prompt.

B.7.4 Screen Dump Formats

With the options `Tiny screen dump`, `Medium screen dump` and `Huge screen dump` you can modify the formats of graphics print-outs by the options `Tiny Dump`, `Medium Dump` and `Huge Dump` of PC-Mos II's *LPT Menu* respectively. If you select one of these options, first a pop-up menu with a list of all available printer modes appears. This menu only lists modes for the EPSON FX and LQ matrix printers and for the HP Laserjet printer, but even if you don't have one of these, it is very probable that your printer supports one of the listed modes: Most 9-pin matrix printers support the EPSON FX modes, most 24-pin matrix printers support both the EPSON FX and the EPSON LQ modes, and most laser and inkjet printers support the HP Laserjet modes.

After you have selected a printer mode, an input box appears, in which you can specify some parameters of the print-out format:

- First you can choose if the print-out should be *upright* or *tilted*, i.e., if horizontal screen lines remain horizontal on the print-out, or if they become vertical and vice versa.
- Next you can specify the approximate paper width in points (1/72"); this value is used to center the graphic on the print-out.
- The next four parameters determine the transmission of screen pixels to printer dots: In both directions, horizontally and vertically, a screen pixel can either

result in one or more printer dots, or several screen pixels can be combined in one printer dot. 1 : 2 means, for example, that one screen pixel will result in two adjacent printer dots on the print-out.

- The remaining parameters are printer control codes: You can specify both a code that is sent to the printer before the print-out and one that is transmitted afterwards. Use these codes to set and reset margin adjustments and similar settings. As with the option `Initialization` of the *Printer-Menu* (B.7.2) the codes must be specified in hexadecimal notation with a zero-byte indicating the end of the code.

Normally you don't need to think about the best values of all these parameters. The following table lists suggestions for printer modes and print-out parameters depending on your graphics adapter and printer type:

Hardware	Tiny Dump	Medium Dump	Huge Dump
VGA	EPSON FX 240x120	EPSON FX 120x120	EPSON FX 120x120
	upright	upright	tilted
and	576/72"	576/72"	576/72"
	1:1	1:1	1:2
9-pin	2:1	1:1	1:2
matrix printer	1B.6C.01	1B.6C.01	1B.6C.01
	1B.32	1B.32	1B.32
VGA	EPSON LQ 180x180	EPSON FX 120x120	EPSON LQ 120x180
	upright	upright	tilted
and	576/72"	576/72"	576/72"
	1:1	1:1	1:2
24-pin	1:1	1:1	1:3
matrix printer	1B.6C.01	1B.6C.01	1B.6C.01
	1B.32	1B.32	1B.32
VGA	HP Laserjet 150x150	HP Laserjet 100x100	HP Laserjet 150x150
	upright	upright	tilted
and	558/72"	558/72"	558/72"
	1:1	1:1	1:2
laser or	1:1	1:1	1:2
inkjet printer	1B.45	1B.45	1B.45
	-	-	-
EGA	EPSON FX 240x120	EPSON FX 120x120	EPSON FX 120x120
	upright	upright	tilted
and	576/72"	576/72"	576/72"
	1:1	1:1	1:2
9-pin	1:1	1:2	1:2
matrix printer	1B.6C.01	1B.6C.01	1B.6C.01
	1B.32	1B.32	1B.32

EGA	EPSON FX 240x180	EPSON LQ 120x180	EPSON LQ 120x180
	upright	upright	tilted
and	576/72"	576/72"	576/72"
	1:1	1:1	1:3
24 pin	1:1	1:2	1:3
matrix printer	1B.6C.01	1B.6C.01	1B.6C.01
	1B.32	1B.32	1B.32
EGA	HP Laserjet 150x150	HP Laserjet 100x100	HP Laserjet 150x150
	upright	upright	tilted
and	558/72"	558/72"	558/72"
	1:1	1:1	1:2
laser or	1:1	1:1	1:2
inkjet printer	1B.45	1B.45	1B.45
	-	-	-
Hercules	EPSON FX 240x120	EPSON FX 120x120	EPSON FX 90x120
	upright	upright	tilted
and	576/72"	576/72"	576/72"
	1:1	1:1	1:2
9-pin	1:1	1:2	1:2
matrix printer	1B.6C.01	1B.6C.01	1B.6C.01
	1B.32	1B.32	1B.32
Hercules	EPSON FX 240x180	EPSON LQ 120x180	EPSON LQ 120x180
	upright	upright	tilted
and	576/72"	576/72"	576/72"
	1:1	1:1	1:3
24-pin	1:1	1:2	1:2
matrix printer	1B.6C.01	1B.6C.01	1B.6C.01
	1B.32	1B.32	1B.32
Hercules	HP Laserjet 300x300	HP Laserjet 150x150	HP Laserjet 150x150
	upright	upright	tilted
and	558/72"	558/72"	558/72"
	1:1	1:1	1:3
laser or	1:2	1:2	1:2
inkjet printer	1B.45	1B.45	1B.45
	-	-	-

B.8 Further Options

The option `Options` invokes an input box with miscellaneous further parameters not matching one of the preceeding categories. First you can specify the maximum number of spectra PC-Mos II should be able to treat simultaneously. Only the size of the computer memory sets an upper limit for this number.

The following two parameters concern the methods for baseline estimation described in section 5.3. The parameters are also defined there. Next there are two

parameters concerning the folding of data: You can choose between sinusoidal and triangular folding mode by pressing [Space] after selecting this parameter, and you can enter the number of channels for the folding point optimization, a parameter that is is explained in section 4.2. You are refered to section 6.1 for a description of the following two parameters concerning the calculation of the velocity map of simulations.

The remaining parameters are toggles. Use [Space] to switch between their two values. The first switch specifies whether PC-Mos II may capture the interrupts triggered by [Ctrl][Break]. If, for any reason, you select `No` here, it is not possible to stop the processing of macro files and fit iterations. The second switch enables you to turn off the sounds of PC-Mos II, which you may find annoying. With the last switch you can force PC-Mos II to switch to text display mode whenever it is executing a theory program. This is intended to facilitate screen outputs of theory programs while you are debugging them (see sect. 11.1).

B.9 The Option Quit

The option `Quit` invokes a pull-down menu with the options `Save changes`, `reLoad configuration`, `Restore defaults` and `Quit`.

With `Save changes` you can confirm the changes you have made during this run of MosSetup and write them to PC-Mos II's configuration file. With `reLoad configuration` you can cancel all changes you have made since the last time you saved a configuration with `Save changes`. `Restore defaults` cancels all changes you have made ever since you received PC-Mos II. With `Quit`, finally, you leave the installation program.

C The Mössbauer Plot Program Mos-Plot

Mos-Plot is a menu controlled Mössbauer plot program which enables you to produce quality plots from your fitted spectra. The menu system of Mos-Plot is operated in the same way as that of MosSetup, so you are refered to appendix B.1 for its explanation.

Before you can use Mos-Plot, you must first fit the spectrum with PC-Mos II, calculate a simulation and output the data to a file with the extension `.mpl`. (`.mpl` is a data format designed specially for the communication between PC-Mos II and Mos-Plot.) Then you can invoke Mos-Plot by typing `mosplot` at the DOS prompt and input the created plot data file with the option `Read plot data` of Mos-Plot's *File-Menu*. With the option `Preview` of the *Action-Menu* you can check what the plot will look like before you actually create the plot with the option `Start print-out`.

With the other options of Mos-Plot you can change the format and design of the plots to your requirements and adapt Mos-Plot to your hardware. The menus and options of Mos-Plot are explaned in following subsections.

C.1 The File-Menu

The *File-Menu* of Mos-Plot provides the options `Read plot data`, `Load configuration`, `Save configuration`, `restore Defaults` and `Quit`.

Selecting the option `Read plot data` invokes an input box in which you must specify the name of the plot data file you have previously created with PC-Mos II. Since Mos-Plot uses the same I/O-routine as PC-Mos II (see sect. 3.1), you must also input the extension `.mpl` explicitely. However, the input box contains `.mpl` as default extension which can be preserved by pressing `Ins` before typing the filename.

Mos-Plot can maintain a configuration file `mosplot.cfg` to store changed values of the plot parameters. When Mos-Plot is invoked, it searches for a file with this name first in the current directory and then in all directories specified in the DOS variable `PATH`. If a configuration file is found, it is loaded and the default values of the plot parameters are superseded by the values stored in the configuration file. You can again load a configuration file with the option `Load configuration`; this time you will get an error message if Mos-Plot doesn't find a configuration file. You can save the current values of the plot parameters with the option `Save configuration`. If Mos-Plot finds an existing configuration file, it overwrites it, otherwise a new configuration file is created in the current directory. With the option `restore Defaults` you can restore the original values of all plot parameters.

The option `Quit` terminates Mos-Plot. Note that changed values of the plot parameters are not saved automatically when you leave Mos-Plot. You must save them explicitely with `Save configuration`.

C.2 The Action-Menu

The *Action-Menu* of Mos-Plot provides the options `Preview` and `Start print-out`. With `Preview` you can view a sketch of the plot on the screen. Note that because of the comparatively poor resolution of the screen graphics the plot might appear somewhat clumsy on the screen. You should use `Preview` only to check the approximate proportions and the style of the printout.

When you select `Start print-out`, the actual high resolution plot is successively built up on the screen and transmitted to the printer. Before you invoke `Start print-out` the first time, you should make sure that the correct printer type is selected with the option `Type` of the printer menu (see sect. C.7).

C.3 The Size-Window

In the *Size-Window* you can adjust parameters that control the size of the plot. The first two parameters are the horizontal and vertical size of the Mössbauer plot in inch. Note that these dimensions do not include the captions, enabling you to easily produce "compatible" plots with different captions. With the *zoom factor*, the next parameter, you can magnify or reduce the size of all scalable plot elements simultaneously.

The parameter `Printer dot overlap` specifies the size of the overlap of printer dots. A big value (2 or 3) yields thick smooth lines whereas fine lines are produced by a small value (0 or 1) for this parameter. The effect depends strongly on the printer resolution. For matrix printers (180 or 240 dpi) the value 1 is recommended; for laser and inkjet printers (300 dpi) too thin lines are avoided by a dot overlap of 2 or 3.

The limits of the ranges of abscissa and ordinate can be modified with the remaining four parameters titled `Velocity range` and `Transmission range`. Note that these limits are superseded whenever a new plot data file is read.

C.4 The Captions-Menu

The *Captions-Menu* enables you to modify all captions of the Mössbauer plot in text and style. Its options correspond to one type of caption each. `Title` is an optional heading for the plot. `Horizontal caption` and `Vertical caption` denote the texts that are printed along both axes of the plot. `Abscissa scale` and `Ordinate scale` denote the numbers at the scalings of the velocity and transmission axis.

Selecting one of these captions invokes another pop-up menu with the options `Text`, `Font`, `Position` and `Division`:

With `Text` you can specify the wording of the caption. This option is not available for `Abscissa scale` and `Ordinate scale` since these are numbers.

`Font` invokes an input box in which you can select the font style and specify the character size for the captions. Three fonts are available: Triplex, Sans Serif, and a plotter font. Choose between them by pressing `Space`. Additionally, each font is available in two styles: `bold` or `thin`.

With `Position` you can specify the position of the caption relative to some appropriate reference point. An input box is opened, which contains the horizontal and vertical shift in points (1/72") relative to this reference point.

The option `Division`, which is available only for `Abscissa scale` and `Ordinate scale`, enables you to set a minimum value for the number of axis divisions.

C.5 The Options-Window

Miscellaneous parameters of the plot can be modified in the *Options-Window*:

- You can specify if the lines of the plot frame should be bold or thin.
- You can specify if the markings on the abscissa and the ordinate should be bold or thin, and how far they should extend in and out of the plot area.
- You can specify the diameter of the circles representing data points and if they should be plotted bold or thin.
- You can choose whether you want error bars for all data point, a single error bar in the lower left corner or no error bars at all; you can specify the width of the error bars and if they should be plotted bold or thin.
- You can specify if the envelope should be plotted bold or thin.
- You can specify if the sub-spectra should be plotted bold or thin.

C.6 The Graphics-Menu

The graphics menu enables you to adapt Mos-Plot to your video hardware. It provides the options `Autodetection`, `Select video mode` and `External driver`. You should only select another option than `Autodetection` if you think Mos-Plot doesn't recognize your hardware correctly, or if you have a graphics adapter other than VGA, EGA or Hercules. The options of Mos-Plot's *Graphics-Menu* are exactly the same as the options of the sub-menu `Hardware` of the *Graphics-Menu* of the installation program MosSetup. So you are refered to section B.3.1 for more information.

C.7 The Printer-Menu

The printer menu provides options to adapt Mos-Plot to your printer. The option `Port` invokes a pop-up menu from which you can select a printer port or set up a printer redirection to the file `mosplot.lpt`. If you use a printer redirection, you can later send the created file to the printer with the copy command of MS-DOS (use the option `/b` behind the filename!). If you have only one printer, normally `LPT1` is the appropriate choice.

If you select the option `Type` of the *Printer-Menu*, a pop-up menu is opened, from which you can choose one of the three printer types Epson FX, Epson LQ and HP Laserjet. If you have one of these printers, the choice is clear. Otherwise you should select Epson FX if you have a (partly Epson compatible) 9-pin printer, EPSON LQ for a (partly Epson compatible) 24-pin printer and HP Laserjet for HP compatible laser and inkjet printers.

Selecting `Options` from the *Printer-Menu* invokes an input box in which you can specify the approximate paper width in points (1/72") (only to center the graphic) and

two printer codes that are sent to the printer before and after the graphics transmission. You can use these codes to set and reset margin adjustments and similar settings. The codes must be denoted in hexadecimal form. A zero byte indicates the end of the code.

D The Theory Manager MkTheory

With the menu controlled program MkTheory you can compile your theory programs and manage the theories integrated into PC-Mos II. The menu system is operated in the same way as that of MosSetup and is therefore not explained again here. You invoke MkTheory by typing `mktheory` at the DOS prompt, perhaps followed by the name of the theory source file you want to compile. In the following subsections the options of MkTheory displayed in its menu line are discussed.

D.1 The Option Paths

The option `Paths` opens an input box in which you can specify the directories in which the Borland / Turbo C++ compiler searches for include files and libraries. You can also specify the directory in which PC-Mos II can be found. (This might be useful if you have several versions of PC-Mos II in different directories.) The last item of the input box is the name of a temporary directory, which MkTheory will create to store some temporary files in it. Normally you need not change this name.

D.2 The Option Names

The option `Names` opens an input box in which you can specify the name of the Borland or Turbo C++ command line compiler and the name of your preferred editor. Modifying the first name also enables you to ues the protected mode compiler BCCX instead of BCC, which is somewhat faster. The protected mode compiler, however, needs about 750 KBytes of extended memory and only works if no other protected mode program is running on your system. If you use Turbo C++ you have to input TCC instead of BCC. The editor you can specify in this input box is the editor called by the option `Edit`, with the name of your source file as command line argument.

D.4 The Option Source

The option `Source` opens an input box in which you can specify the name of the theory source file you want to compile. The file name extension `.C` is appended automatically. If you have invoked MkTheory with a command line argument, the input box will contain the name you have specified there.

D.6 The Option Edit

The option `Edit` invokes the editor you have specified with the option `Names`. The name of your source file is passed as command line argument. `Edit` is only accessable if both names are available.

D.7 The Option Compile

The option `Compile` invokes the command line compiler of Borland C++ to compile your theory. All messages of the compiler (as well as some messages of MkTheory) are displayed in the output window on the screen. The compiler messages are also written to a file which has the same name as your source file but the extension `.log`. This enables you to trace syntax errors reported by the compiler.

D.8 The Option OS-Shell

The option `OS-Shell` invokes a DOS command line processor enabling you to work with the operating system without terminating MkTheory. You can use all DOS commands. To return to MkTheory simply type `exit` at the DOS prompt.

D.9 The Option Quit

The option `Quit` terminates MkTheory. Since before returning to DOS MkTheory deletes all temporary files it has created, you will sometimes have to wait a few seconds until the DOS prompt appears.

E The Enclosed Theories

This appendix briefly describes all enclosed theory programs. The source code of all theories can be found in the subdirectory `\theory`. Example parameter files for all theories can be found in the subdirectory `\examples`. With the exception of `Polynml`, all enclosed theories are of the form described in section 11.3, i.e., deal with Lorentzian lines or a transmission integral for Lorentzian lines.

E.1 NSing

The theory NSing simply builds a theory curve from N single independent Lorentzians. The number of Lorentzians is determined from the number of parameters passed to the theory.

General parameters:

P_0	Baseline
P_1	Total area (sum of the products of width and depth of all Lorentzians)

Parameters of the first Lorentzian:

P_{2+0}	Isomer shift (mm/s)
P_{2+1}	Width (mm/s)

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ Lorentzian:

P_{1+3i+0}	Isomer shift (mm/s)
P_{1+3i+1}	Width relative to the first Lorentzian
P_{1+3i+2}	Partial intensity, i.e. relative contribution to the total area

E.2 NTSing

The theory NTSing calculates a transmission integral for a Lorentzian shaped source spectrum and an resonance absorbtion cross section consisting of N single independent Lorentzian lines. Note that the line widths occuring in the parameter lists of theories for transmission integrals now are the widths of the Lorentzians in the cross section rather than of those in the transmission spectrum. The latter are wider at least by the amount of the source line width. More information on transmission integrals can be found in section 11.3.4.

General parameters:

P_0	Baseline
P_1	Fraction of resonant quanta in the single channel window
P_2	Line width of the source spectrum (mm/s)
P_3	Range beyond the velocity range of the spectrum for which the transmission is not yet assumed to be unity, in units of the source line width. A value of 5.0 has proved to yield good results in most cases; this parameter must always be fixed.
P_4	Number of knots per source line width for the numerical integration. A value of 12 has proved to yield good results; this parameter must always be fixed.
P_5	Total area (sum of the products of width and depth of all Lorentzians)

Parameters of the first Lorentzian:

P_{6+0}	Isomer shift (mm/s)
P_{6+1}	Width (mm/s)

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ Lorentzian:

P_{5+3i+0}	Isomer shift (mm/s)
P_{5+3i+1}	Width relative to the first Lorentzian
P_{5+3i+2}	Partial intensity, i.e. relative contribution to the total area

E.3 NDoub

The theory NDoub builds its theory curve from N (quadrupole) doublets of Lorentzians. Normally such doublets arise from a $1/2 - 3/2$ transition in a crystal field with a non-vanishing electric field gradient tensor. The number of doublets is determined from the number of parameters passed to the theory. Note that you can reduce such a doublet to a single line by fixing its quadropole splitting parameter with the value zero.

General parameters:

P_0	Baseline
P_1	Total area (sum of the products of width and depth of all Lorentzians)

Parameters of the first doublet:

P_{2+0}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{2+1}	Isomer shift (mm/s)
P_{2+2}	Width (mm/s)

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ doublet:

P_{1+4i+0}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{1+4i+1}	Isomer shift (mm/s)
P_{1+4i+2}	Width relative to the first doublet
P_{1+4i+3}	Partial intensity, i.e. relative contribution to the total area

E.4 NTDoub

The theory NTDoub is the transmission integral extension to the theory NDoub (see E.3). The parameters of the transmission integral are described in more detail in E.2 and in section 11.3.4.

General parameters:

P_0	Baseline
P_1	Fraction of resonant quanta in the single channel window
P_2	Line width of the source spectrum (mm/s)
P_3	Excess integration range in units of the source line width
P_4	Number of integration knots per source line width
P_5	Total area (sum of the products of width and depth of all Lorentzians)

Parameters of the first doublet:

P_{6+0}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{6+1}	Isomer shift (mm/s)
P_{6+2}	Width (mm/s)

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ doublet:

P_{5+4i+0}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{5+4i+1}	Isomer shift (mm/s)
P_{5+4i+2}	Width relative to the first doublet
P_{5+4i+3}	Partial intensity, i.e. relative contribution to the total area

E.3 NSext

The theory NSext builds its theory curve from N (magnetic) sextets of Lorentzians. Such sextets normally arise from the Mössbauer transition of the nucleus ^{57}Fe in a (internal or external) magnetic field. The theory assumes the g-factors of this nucleus. An additional electric quadrupole interaction can be taken into account as long as the interaction energy is small compared with the magnetic energies. (Note that here the deciding component of the electric field gradient tensor is the zz-component along the direction of the magnetic field.) A small distribution of magnetic fields contributing to one sextet can also be taken into account by a correlated widening of the Lorentzian lines. The relative intensities of the Lorentzians are calculated under the assumption of a poly-crystalline specimen.

Additionally M doublets (as described in E.3) may contribute to the theory curve. The number of sextets and doublets is specified by two extra parameters, which, of course, always must be assigned the attribute `fixed`.

General parameters:

P_0	Baseline
P_1	Total area (sum of the products of width and depth of all Lorentzians)
P_2	Number of sextets (≥ 1)
P_3	Number of doublets (≥ 0)

Parameters of the first sextet:

P_{4+0}	Magnetic hyperfine field (Tesla)
P_{4+1}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{4+2}	Isomer shift (mm/s)
P_{4+3}	Width of the outer lines (mm/s)
P_{4+4}	Sets the intensity ratio of the sextet lines to $3 : 2x : x^2$ if $x > 0$ and to $3 : -2x : 1$ if $x < 0$ respectively
P_{4+5}	Natural line width (mm/s)

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ sextet:

P_{3+7i+0}	Magnetic hyperfine field (Tesla)
P_{3+7i+1}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{3+7i+2}	Isomer shift (mm/s)
P_{3+7i+3}	Width of the outer lines (mm/s)
P_{3+7i+4}	Sets the intensity ratio of the sextet lines to $3 : 2x : x^2$ if $x > 0$ and to $3 : -2x : 1$ if $x < 0$ respectively
P_{3+7i+5}	Partial intensity, i.e. relative contribution to the total area
P_{3+7i+6}	Natural line width (mm/s)

Parameters of the first doublet (if any) (assume n sextets):

P_{3+7n+0}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{3+7n+1}	Isomer shift (mm/s)
P_{3+7n+2}	Width (mm/s)
P_{3+7n+3}	Partial intensity, i.e. relative contribution to the total area

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ doublet:

$P_{3+7n+4i+0}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{3+7n+4i+1}$	Isomer shift (mm/s)
$P_{3+7n+4i+2}$	Width relative to the first doublet
$P_{3+7n+4i+3}$	Partial intensity, i.e. relative contribution to the total area

E.6 NTSext

The theory NTSext is the transmission integral extension to the theory NSext (see E.5). The parameters of the transmission integral are described in more detail in E.2 and in section 11.3.4.

General parameters:

P_0	Baseline
P_1	Fraction of resonant quanta in the single channel window
P_2	Line width of the source spectrum (mm/s)
P_3	Excess integration range in units of the source line width
P_4	Number of integration knots per source line width
P_5	Total area (sum of the products of width and depth of all Lorentzians)
P_6	Number of sextets (≥ 1)
P_7	Number of doublets (≥ 0)

Parameters of the first sextet:

P_{8+0}	Magnetic hyperfine field (Tesla)
P_{8+1}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{8+2}	Isomer shift (mm/s)
P_{8+3}	Width of the outer lines (mm/s)
P_{8+4}	Sets the intensity ratio of the sextet lines to $3 : 2x : x^2$ if $x > 0$ and to $3 : -2x : 1$ if $x < 0$ respectively
P_{8+5}	Natural line width (mm/s)

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ sextet:

P_{7+7i+0}	Magnetic hyperfine field (Tesla)
P_{7+7i+1}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{7+7i+2}	Isomer shift (mm/s)
P_{7+7i+3}	Width of the outer lines (mm/s)
P_{7+7i+4}	Sets the intensity ratio of the sextet lines to $3 : 2x : x^2$ if $x > 0$ and to $3 : -2x : 1$ if $x < 0$ respectively
P_{7+7i+5}	Partial intensity, i.e. relative contribution to the total area
P_{7+7i+6}	Natural line width (mm/s)

Parameters of the first doublet (if any) (assume n sextets):

P_{7+7n+0}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{7+7n+1}	Isomer shift (mm/s)
P_{7+7n+2}	Width (mm/s)
P_{7+7n+3}	Partial intensity, i.e. relative contribution to the total area

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ doublet:

$P_{7+7n+4i+0}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{7+7n+4i+1}$	Isomer shift (mm/s)
$P_{7+7n+4i+2}$	Width relative to the first doublet
$P_{7+7n+4i+3}$	Partial intensity, i.e. relative contribution to the total area

E.7 NOct

The theory NOct builds its theory curve from N octets of Lorentzians. Such octets can be generalized from the sextets desribed above by allowing quadropole interaction energies comparable to the magnetic energies. (The arising pertubation Hamiltonian is diagonalized numerically.) The relative intensities of the Lorentzians are calculated under the assumption of a poly-crystalline specimen, but the theory can easily be modified to allow also single-crystalline absorbers (see source file `octet.c` and [Hag]).

Additionally M sextets (as described in E.5) and K doublets (as described in E.3) may contribute ot the theory curve. The number of octets, sextets and doublets is specified by three extra parameters, which, of course, always must be assigned the attribute `fixed`.

General parameters:

P_0	Baseline
P_1	Total area (sum of the products of width and depth of all Lorentzians)
P_2	Number of octets (≥ 1)
P_3	Number of sextets (≥ 0)
P_4	Number of doublets (≥ 0)

Parameters of the first octet:

P_{5+0}	Magnetic hyperfine field B (Tesla)
P_{5+1}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{5+2}	Isomer shift (mm/s)
P_{5+3}	Asymmetry parameter η
P_{5+4}	Polar angle θ between B and V_{zz}
P_{5+5}	Azimuth angle ϕ between B and V_{zz}
P_{5+6}	Width of the octet lines

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ octet:

P_{4+8i+0}	Magnetic hyperfine field B (Tesla)
P_{4+8i+1}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{4+8i+2}	Isomer shift (mm/s)
P_{4+8i+3}	Asymmetry parameter η

P_{4+8i+4}	Polar angle θ between B and V_{zz}
P_{4+8i+5}	Azimuth angle ϕ between B and V_{zz}
P_{4+8i+6}	Width of the octet lines
P_{4+8i+7}	Partial intensity, i.e. relative contribution to the total area

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 0)$ sextet (assume n octets):

$P_{4+8n+7i+0}$	Magnetic hyperfine field (Tesla)
$P_{4+8n+7i+1}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{4+8n+7i+2}$	Isomer shift (mm/s)
$P_{4+8n+7i+3}$	Width of the outer lines (mm/s)
$P_{4+8n+7i+4}$	Sets the intensity ratio of the sextet lines to $3 : 2x : x^2$ if $x > 0$ and to $3 : -2x : 1$ if $x < 0$ respectively
$P_{4+8n+7i+5}$	Partial intensity, i.e. relative contribution to the total area
$P_{4+8n+7i+6}$	Natural line width (mm/s)

Parameters of the first doublet (if any) (assume n octets and m sextets):

$P_{4+8n+7m+0}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{4+8n+7m+1}$	Isomer shift (mm/s)
$P_{4+8n+7m+2}$	Width (mm/s)
$P_{4+8n+7m+3}$	Partial intensity, i.e. relative contribution to the total area

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ doublet:

$P_{4+8n+7m+4i+0}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{4+8n+7m+4i+1}$	Isomer shift (mm/s)
$P_{4+8n+7m+4i+2}$	Width relative to the first doublet
$P_{4+8n+7m+4i+3}$	Partial intensity, i.e. relative contribution to the total area

E.8 NTOct

The theory NTOct is the transmission integral extension to the theory NOct (see E.7). The parameters of the transmission integral are described in more detail in E.2 and in section 11.3.4.

General parameters:

P_0	Baseline
P_1	Fraction of resonant quanta in the single channel window
P_2	Line width of the source spectrum (mm/s)
P_3	Excess integration range in units of the source line width
P_4	Number of integration knots per source line width
P_5	Total area (sum of the products of width and depth of all Lorentzians)
P_6	Number of octets (≥ 1)
P_7	Number of sextets (≥ 0)
P_8	Number of doublets (≥ 0)

Parameters of the first octet:

P_{9+0}	Magnetic hyperfine field B (Tesla)
P_{9+1}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{9+2}	Isomer shift (mm/s)
P_{9+3}	Asymmetry parameter η

P_{9+4}	Polar angle θ between B and V_{zz}
P_{9+5}	Azimuth angle ϕ between B and V_{zz}
P_{9+6}	Width of the octet lines

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ octet:

P_{8+8i+0}	Magnetic hyperfine field B (Tesla)
P_{8+8i+1}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{8+8i+2}	Isomer shift (mm/s)
P_{8+8i+3}	Asymmetry parameter η
P_{8+8i+4}	Polar angle θ between B and V_{zz}
P_{8+8i+5}	Azimuth angle ϕ between B and V_{zz}
P_{8+8i+6}	Width of the octet lines
P_{8+8i+7}	Partial intensity, i.e. relative contribution to the total area

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 0)$ sextet (assume n octets):

$P_{8+8n+7i+0}$	Magnetic hyperfine field (Tesla)
$P_{8+8n+7i+1}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{8+8n+7i+2}$	Isomer shift (mm/s)
$P_{8+8n+7i+3}$	Width of the outer lines (mm/s)
$P_{8+8n+7i+4}$	Sets the intensity ratio of the sextet lines to $3 : 2x : x^2$ if $x > 0$ and to $3 : -2x : 1$ if $x < 0$ respectively
$P_{8+8n+7i+5}$	Partial intensity, i.e. relative contribution to the total area
$P_{8+8n+7i+6}$	Natural line width (mm/s)

Parameters of the first doublet (if any) (assume n octets and m sextets):

$P_{8+8n+7m+0}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{8+8n+7m+1}$	Isomer shift (mm/s)
$P_{8+8n+7m+2}$	Width (mm/s)
$P_{8+8n+7m+3}$	Partial intensity, i.e. relative contribution to the total area

Parameters of the $(i+1)^{\text{st}}$ $(i \geq 1)$ doublet:

$P_{8+8n+7m+4i+0}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{8+8n+7m+4i+1}$	Isomer shift (mm/s)
$P_{8+8n+7m+4i+2}$	Width relative to the first doublet
$P_{8+8n+7m+4i+3}$	Partial intensity, i.e. relative contribution to the total area

E.9 MagDist

The theory MagDist builds its theory curve from a distribution of N magnetic sextets (as described in E.5) arising from an equidistant distribution of magnetic fields. The sextets in this distribution are only allowed to differ in their relative intensities. Additionally M single sextets (as described in E.3) and K doublets (as described in E.2) may contribute to the theory curve. The number of sextets in the distribution as well as the number of single sextets and doublets are specified by three extra parameters, which, of course, always must be assigned the attribute `fixed`.

General parameters:

P_0	Baseline
P_1	Total area (sum of the products of width and depth of all Lorentzians)

P_2	Number of sextets in the distribution (≥ 2)
P_3	Number of single sextets (≥ 0)
P_4	Number of doublets (≥ 0)

Parameters of the distribution:

P_{5+0}	Minimum magnetic hyperfine field (Tesla)
P_{5+1}	Maximum magnetic hyperfine field (Tesla)
P_{5+2}	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
P_{5+3}	Isomer shift (mm/s)
P_{5+4}	Width of the outer lines (mm/s)
P_{5+5}	Sets the intensity ratio of the sextet lines to $3 : 2x : x^2$ if $x > 0$ and to $3 : -2x : 1$ if $x < 0$ respectively
P_{5+6}	Natural line width (mm/s)
P_{5+7+i}	Partial intensity of the $(i+1)^{\text{st}}$ ($0 \leq i \leq n-2$) sextet in the distribution relative to the total area. The intensity of the last sextet (that with strongest field) is determined by the normalization constraint.

Parameters of the $(i+1)^{\text{st}}$ ($i \geq 0$) single sextet (assume n sextets in the distribution):

$P_{11+n+7i+0}$	Magnetic hyperfine field (Tesla)
$P_{11+n+7i+1}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{11+n+7i+2}$	Isomer shift (mm/s)
$P_{11+n+7i+3}$	Width of the outer lines (mm/s)
$P_{11+n+7i+4}$	Sets the intensity ratio of the sextet lines to $3 : 2x : x^2$ if $x > 0$ and to $3 : -2x : 1$ if $x < 0$ respectively
$P_{11+n+7i+5}$	Partial intensity, i.e. relative contribution to the total area
$P_{11+n+7i+6}$	Natural line width (mm/s)

Parameters of the first doublet (if any) (assume n sextets in the distribution and m single sextets):

$P_{11+n+7m+0}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{11+n+7m+1}$	Isomer shift (mm/s)
$P_{11+n+7m+2}$	Width (mm/s)
$P_{11+n+7m+3}$	Partial intensity, i.e. relative contribution to the total area

Parameters of the $(i+1)^{\text{st}}$ ($i \geq 1$) doublet:

$P_{11+n+7m+4i+0}$	Quadrupole splitting ($eQV_{zz}/2$ in mm/s)
$P_{11+n+7m+4i+1}$	Isomer shift (mm/s)
$P_{11+n+7m+4i+2}$	Width relative to the first doublet
$P_{11+n+7m+4i+3}$	Partial intensity, i.e. relative contribution to the total area

E.10 Polynml

The theory curve calculated by the theory `Polynml` is a polynomial of arbitrary degree. This theory is mainly intended as example for a theory for general "line shapes", but maybe you can use it to fit some (non-Mössbauer) data to a straight line. The parameters of the theory are the polynomial coefficients in the order of increasing powers. The degree is determined from the number of parameters.

`Polynml` is the only of the enclosed theories which supports the option `Modify` of the *Process Menu*. It simply eliminates the highest power term of the polynomial from the data points (just as example).

F The Enclosed Formats

The following sections contain short descriptions of PC-Mos II's input formats. In the tables following conventions have been used:

- A text that appears literally in the data files is typeset in teletype style.
- Lines that are optional (i.e., that may be omitted) are preceeded by an asterisk (*).
- Names of variables typeset in italics and enclosed in angle brackets mean that at this position the value of this variable is expected.
- If such a variable is additionally enclosed in brackets, the value may be omitted.
- The column ruler at the top of each format is intended to make the location of the items easier.

F.1 The Format .HDR

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

⟨*dataheader*⟩

*	`velocity`	⟨*velocity*⟩
*	`vmax`	⟨*vmax*⟩
*	`foldingpoint`	⟨*foldpt*⟩
	`channels`	⟨*channels*⟩

This format contains some header information for a spectrum but not the spectrum itself. Usually it is used together with data files with the format `.asc`:

F.2 The Format .ASC

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

⟨*counts*(0)⟩
⟨*counts*(1)⟩
⟨*counts*(2)⟩
...

This format is the simplest of all, it simply contains all countrates in sequence, separated by blancs or linefeeds. Note that since such raw data files don't contain the number of channels, you must either enter this number manually with the option `Channels` of the sub-menu `Data` of the *Edit-Menu* before or read a `.hdr`-file prior to reading a file with this format.

F.3 The Format .PAR

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

⟨*paraheader*⟩

*	`foldingpoint`	⟨*foldpt*⟩
*	`vmax`	⟨*vmax*⟩
*	`theory`	⟨*theory*⟩
*	`maxiter`	⟨*maxiter*⟩
*	`parasig`	⟨*parasig*⟩

```
*  chi2sig          ⟨chi2sig⟩
   fitparameter     ⟨npara⟩
   ⟨paraname(0)⟩    ⟨initpara(0)⟩ [⟨attribute(0)⟩]
   ⟨paraname(1)⟩    ⟨initpara(1)⟩ [⟨attribute(1)⟩]
   ...
```

F.4 The Format .RES

```
123456789012345678901234567890123456789012345678901234567890
```

```
   ⟨paraheader⟩
   fitresults of data :
   ⟨dataheader⟩
*  foldingpoint     ⟨foldpt⟩
*  geometryeffect   ⟨geomeff⟩
*  effect           ⟨effect⟩
*  theory           ⟨theory⟩
*  date             ⟨fitdate⟩
*  flags            ⟨fitflags⟩
*  fitspectra       ⟨fitspectra⟩
*  chi2             ⟨chi2⟩
*  chi2-test        ⟨chi2test⟩
*  corr-test        ⟨corrtest⟩
*  fitcycles        ⟨niter⟩
*  maxiter          ⟨maxiter⟩
*  parasig          ⟨parasig⟩
*  chi2sig          ⟨chi2sig⟩
   fitparameter     ⟨npara⟩
   ⟨paraname(0)⟩    ⟨ipara(0)⟩ ⟨fpara(0)⟩ ⟨attr(0)⟩ ⟨error(0)⟩ ⟨chk1(0)⟩⟨chk2(0)⟩
   ⟨paraname(1)⟩    ⟨ipara(1)⟩ ⟨fpara(1)⟩ ⟨attr(1)⟩ ⟨error(1)⟩ ⟨chk1(1)⟩⟨chk2(1)⟩
   ...
```

The variables *initpara*, *finalpara*, *attribute*, *perror*, *check1* and *check2* have been abbreviated with *ipara*, *fpara*, *attr*, *error*, *chk1* and *chk2* respectively.

F.5 The Format .dta

```
123456789012345678901234567890123456789012345678901234567890
```

```
   ⟨dataheader⟩
*  geometryeffect ⟨geomeff⟩
*  effect         ⟨effect⟩
*  vmin           ⟨vmin⟩
*  vmax           ⟨vmax⟩
   channels       ⟨ndata⟩
   ⟨velmap(0)⟩  ⟨velmap(1)⟩  ⟨velmap(2)⟩  ⟨velmap(3)⟩  ⟨velmap(4)⟩
   ⟨velmap(5)⟩  ⟨velmap(6)⟩  ⟨velmap(7)⟩  ⟨velmap(8)⟩  ⟨velmap(9)⟩
   ...
   ⟨data(0)⟩    ⟨data(1)⟩    ⟨data(2)⟩    ⟨data(3)⟩    ⟨data(4)⟩
```

$\langle data(5)\rangle$ $\langle data(6)\rangle$ $\langle data(7)\rangle$ $\langle data(8)\rangle$ $\langle data(9)\rangle$
...
* $\langle derror(0)\rangle$ $\langle derror(1)\rangle$ $\langle derror(2)\rangle$ $\langle derror(3)\rangle$ $\langle derror(4)\rangle$
* $\langle derror(5)\rangle$ $\langle derror(6)\rangle$ $\langle derror(7)\rangle$ $\langle derror(8)\rangle$ $\langle derror(9)\rangle$
* ...

With this format you can also input some non-Mössbauer data to PC-Mos II.

F.6 The Format .MPL

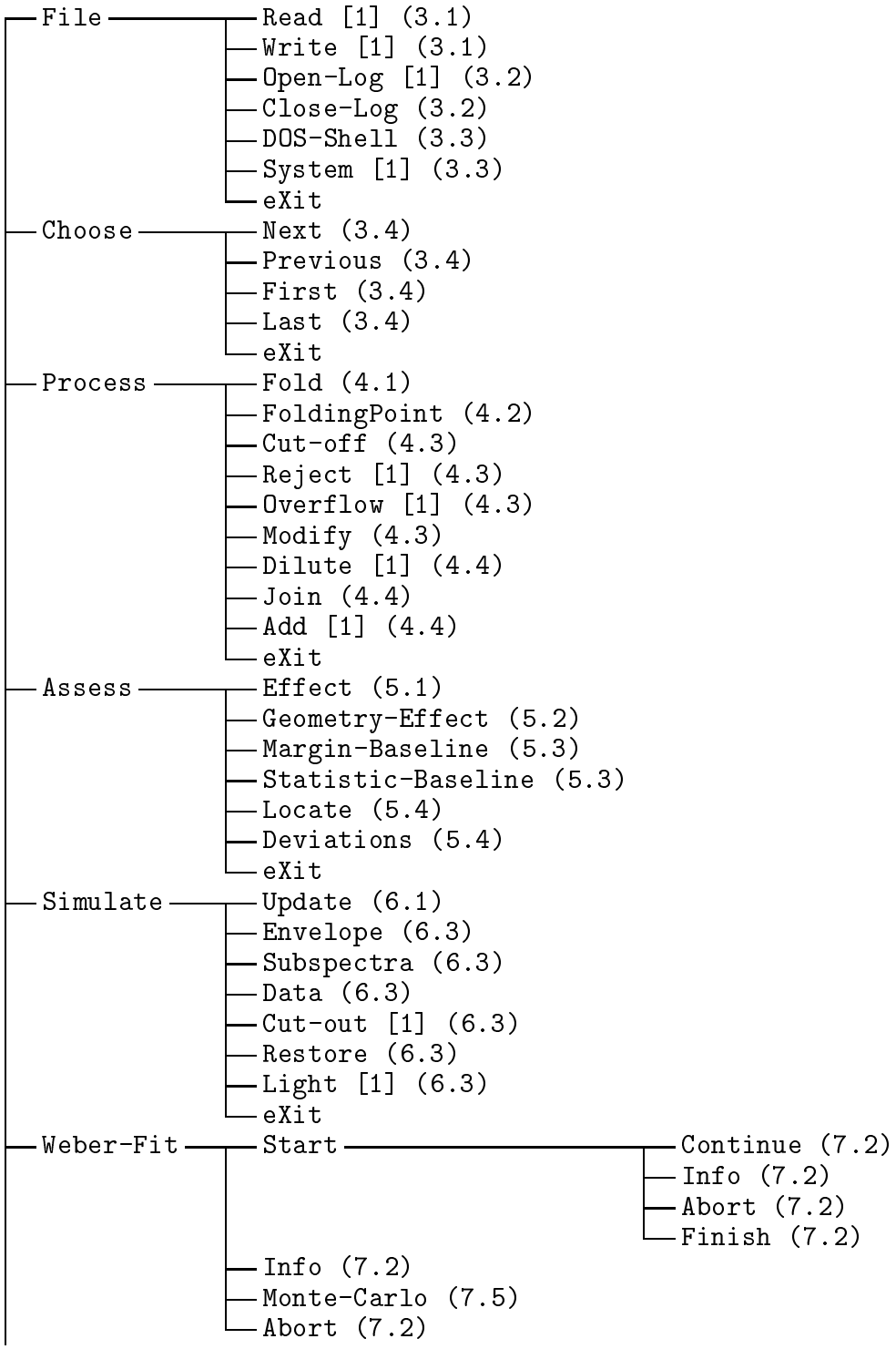
```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

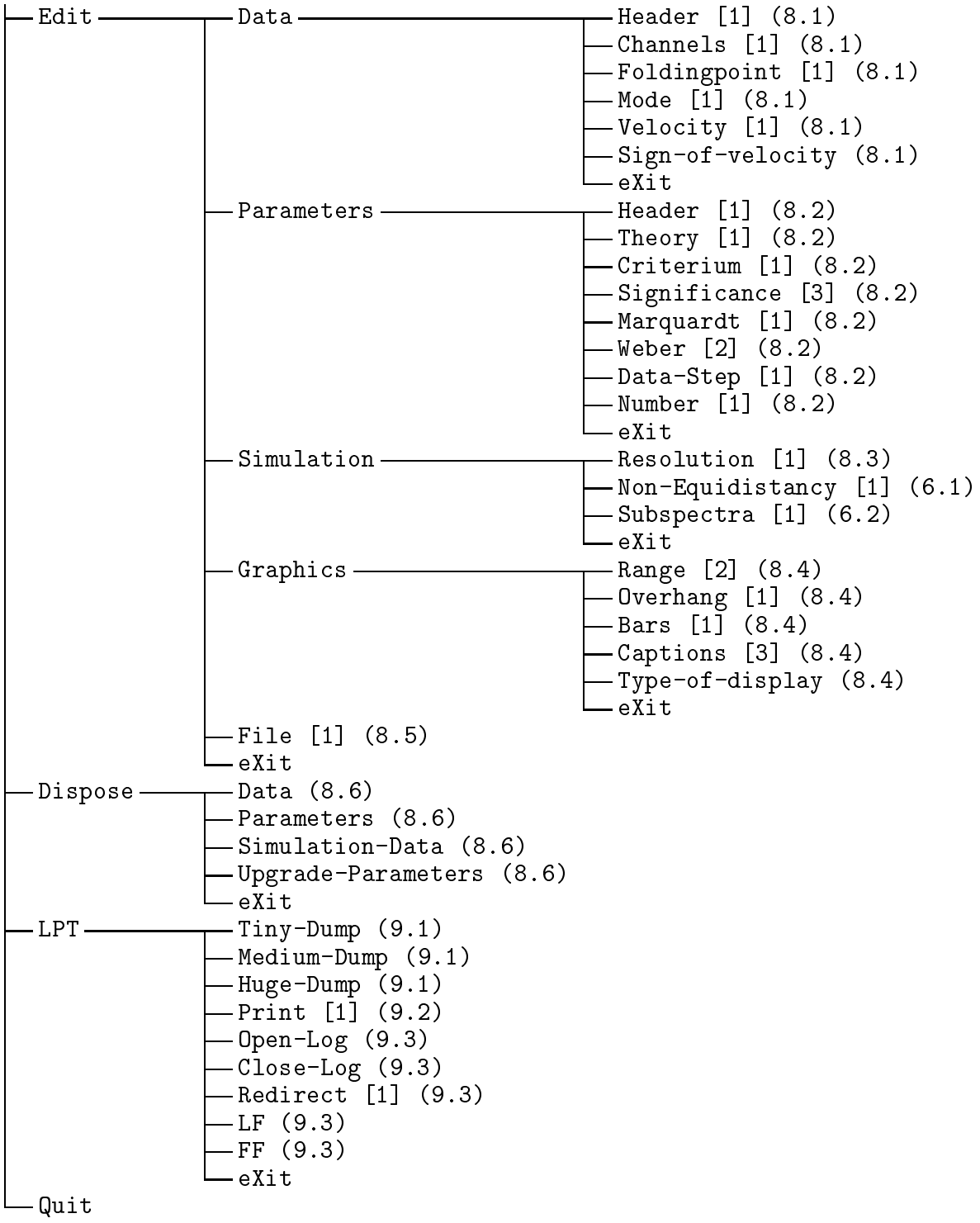
$\langle dataheader\rangle$
`data points:` $\langle ndata\rangle$
`theory points:` $\langle simpoints\rangle$
`sub-spectra:` $\langle nsubspect\rangle$
`minimum velocity:` $\langle vmin\rangle$
`maximum velocity:` $\langle vmax\rangle$
`minimum countrate:` $\langle ymin\rangle$
`maximum countrate:` $\langle ymax\rangle$
`baseline:` $\langle finalpara(0)\rangle$
`data points with errors:`
$\langle velmap(0)\rangle$ $\langle data(0)\rangle$ $\langle derror(0)\rangle$
$\langle velmap(1)\rangle$ $\langle data(1)\rangle$ $\langle derror(1)\rangle$
...
`theory points with sub-spectra:`
$\langle simvelmap(0)\rangle$ $\langle simdata(0,0)\rangle$ $\langle simdata(1,0)\rangle$ ...
$\langle simvelmap(1)\rangle$ $\langle simdata(0,1)\rangle$ $\langle simdata(1,1)\rangle$ ...
...

This is the preferred format for the communication between PC-Mos II and MosPlot.

G The Menu Tree of PC-Mos II

In the following grafic all options of PC-Mos II are listed together with the the number of input items they expect (in brackets) and together with a reference to the section of the manual in which they are explained (in parentheses).




Edit
Data
Header [1] (8.1)
Channels [1] (8.1)
Foldingpoint [1] (8.1)
Mode [1] (8.1)
Velocity [1] (8.1)
Sign-of-velocity (8.1)
eXit
Parameters
Header [1] (8.2)
Theory [1] (8.2)
Criterium [1] (8.2)
Significance [3] (8.2)
Marquardt [1] (8.2)
Weber [2] (8.2)
Data-Step [1] (8.2)
Number [1] (8.2)
eXit
Simulation
Resolution [1] (8.3)
Non-Equidistancy [1] (6.1)
Subspectra [1] (6.2)
eXit
Graphics
Range [2] (8.4)
Overhang [1] (8.4)
Bars [1] (8.4)
Captions [3] (8.4)
Type-of-display (8.4)
eXit
File [1] (8.5)
eXit
Dispose
Data (8.6)
Parameters (8.6)
Simulation-Data (8.6)
Upgrade-Parameters (8.6)
eXit
LPT
Tiny-Dump (9.1)
Medium-Dump (9.1)
Huge-Dump (9.1)
Print [1] (9.2)
Open-Log (9.3)
Close-Log (9.3)
Redirect [1] (9.3)
LF (9.3)
FF (9.3)
eXit
Quit

H References

[Cra] T. E. Cranshaw, *The Deduction of the Best Values of the Parameters from Mössbauer Spectra*, J. Phys. E **7** (1974) 122

[Hag] L. Häggström: *Determination of Hyperfine Parameters from* 1/2 – 3/2 *Transitions in Mößbauer Spectroscopy*, Institute of Physics, University of Uppsala, Uppsala, Sweden SW ISSN 0042–0263

[Ker] B. W. Kernighan, D. M. Ritchie: *The C Programming Language* (second edition), New Jersey (Prentice Hall) 1988

[Lan] G. Lang: *Interpretation of Experimental Mössbauer Spectrum Areas*, Nucl. Instr. Methods **24** (1963) 425

[Lin] T.-M. Lin, R. S. Preston, *Comparison of Techniques for Folding and Unfolding Mössbauer Spectra for Data Analysis*, in: J. Gruverman, C. W. Seidel, D. K. Dieterly (ed.): *Mössbauer Effect Methology, Volume 9*, New York, London 1974

[Pre] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling: *Numerical Recipes in C*, Cambridge (Cambridge University Press) 1988

[Sch] G. Schatz, A. Weidinger: *Nukleare Festkörperphysik*, Stuttgart (Teubner) 1985

[She] G. K. Shenoy, J, M. Friedt, H. Maletta, S. L. Ruby: *Curve Fitting and the Transmission Integral: Warnings and Suggestions*, in: J. Gruverman, C. W. Seidel, D. K. Dieterly (ed.): *Mössbauer Effect Methology, Volume 9*, New York, London 1974

[Sto] J. Stoer: *Einführung in die Numerische Mathematik I*, Berlin, Heidelberg, New York, Tokyo (Springer) 1983

[Web] M. Weber: *Turbo Pascal Tools*, Braunschweig (Vieweg) 1988