MRUI Manual v96.3

A user's guide to the
Magnetic Resonance User Interface
software package

by

Aad van den Boogaart

1 May 1997

This manual was written by

Dr. ir. Aad van den Boogaart, Eur. Ing.
Katholieke Universiteit Leuven
Faculteit Geneeskunde, Onderwijs en Navorsing
Biomedical NMR Unit
Gasthuisberg
3000 Leuven
BELGIUM
tel + 32 - 16 - 34 59 10
fax + 32 - 16 - 34 59 91

Manual version 96.3 from 23 April 1996 to 1 May 1997

The author welcomes feedback on this manual, in the form of comments, suggestions, or simply fan mail. Be aware that although the MRUI software package is the result of a joint EC project, this manual was written by me mainly in my spare time. Recently, Leentje Vanhamme added chap. 14 & 15.

Aad van den Boogaart
Spinnerwei 4
5551 PS Valkenswaard
THE NETHERLANDS
e-mail `a.boogaart@aranea.nl`
home page `http://mrui-web.uab.es/mruiwww/aad_html/index.html`

The author thanks the following persons for valuable contributions to this manual (listed in alphabetical order):

Ron de Beer	Technische Universiteit Delft	The Netherlands
Tiziano Binzoni	Université de Genève	Switzerland
Sophie Cavassila	Université Claude Bernard – Lyon I	France
Miquel Cabañas	Universitat Autònoma de Barcelona	Spain
Yvonne Heidkamp	Academisch Ziekenhuis Nijmegen	The Netherlands
Arno Knijn	Istituto Superiore di Sanita Rome	Italy
Dirk van Ormondt	Technische Universiteit Delft	The Netherlands
Leentje Vanhamme	Katholieke Universiteit Leuven	Belgium
Paul Van Hecke	Katholieke Universiteit Leuven	Belgium

N.B. In some figures in this manual, text in windows may be poorly displayed, and some parts of menus or graphics item may seem to overlap with other parts of the window. This is due entirely to the way the MRUI windows have been included in this manual document, and does ***NOT*** *occur in the MRUI software itself.*

ISBN: 90-9010509-3

EC Project HCM

From April 1st, 1995 to March 31st, 1997, the development of the MRUI software package, and its official distribution, was part of, and funded by, the **EC Human Capital & Mobility / Networks** program, project number **HCM-CHRX-CT94-0432**, *Advanced Signal Processing for Medical Magnetic Resonance Imaging and Spectroscopy*.

EU supervisors:	M. Rubin, (Ges.)
	R. Rahders, (Neg.)
	Dr. Shiel-Tscherny, (R.Sc.)
Project co-ordinator:	Dr. Dirk van Ormondt — Delft University of Technology

Dr. D. van Ormondt
Applied Physics Department
Delft University of Technology
P.O. Box 5046
2600 GA Delft
THE NETHERLANDS

Dr. D. Graveron-Demilly
Lab de RMN, CNRS UPRESA Q5012
Université Claude Bernard Lyon I, Bât. 308
43, Boulevard du 11 Novembre 1918
69622 Villeurbanne Cédex
FRANCE

Prof. S. Van Huffel
Dept. Electrical Engineering, ESAT / SISTA
Katholieke Universiteit Leuven
Kardinaal Mercierlaan 94
3001 Heverlee
BELGIUM

Prof. P. Van Hecke
Biomedical NMR Unit
Katholieke Universiteit Leuven
Gasthuisberg
3000 Leuven
BELGIUM

Prof. Dr. D. Michel
Fakultät für Physik und Geowissenschaften
Universität Leipzig
Linnestrasse 5
04103 Leipzig
GERMANY

Prof. B.G. Mertzios
Automatic Control Systems Laboratory
Dept. Electrical and Computer Engineering
Democritus University of Thrace
67100 Xanthi
GREECE

Prof. J-P. Antoine
Unité de Physique Théorique et Physique
Dept. Physique - Faculté des Sciences
Université Catholique de Louvain
1348 Louvain-La-Neuve
BELGIUM

Prof. G. Carayannis
Computer Science Division
Dept. Electrical Engineering
National Technical University of Athens
9 Iroon Polytechniou St.
15773 Zographou
GREECE

Prof. J.M. Dereppe
Chimie Physique Moleculaire et Cristallographie
Dept. Chimie - Faculté des Sciences
Bât. Lavoisier, Place Louis Pasteur 1, Bte 3a
Université Catholique de Louvain
1348 Louvain-La-Neuve
BELGIUM

For more information, please visit the HCM *Advanced Signal Processing for Medical Magnetic Resonance Imaging and Spectroscopy* WWW home pages at:
`http://azur.univ-lyon1.fr/HCM/hcm.html`http://azur.univ-lyon1.fr/HCM/hcm.html

Contributions to the MRUI software

The following persons/groups have developed the following algorithms of the MRUI software:

VARPRO	Jan-Willem van der Veen, Arno Knijn, Chris van der Voort	Technische Universiteit Delft (NL)
	Aad van den Boogaart	Katholieke Universiteit Leuven (B)
	Leentje Vanhamme	Katholieke Universiteit Leuven (B)
AMARES	Leentje Vanhamme	Katholieke Universiteit Leuven (B)
(Filter)HLSVD	Willem Pijnappel, Aad van den Boogaart	Technische Universiteit Delft (NL)
(Filter)HSVD	Herbert Barkhuijsen, Ron de Beer, Aad van den Boogaart	Technische Universiteit Delft (NL)
Matlab HSVD	Frank Wajer	Technische Universiteit Delft (NL)
HTLS	Vlastimil Vondra	Academy of Sciences Brno (CZ)
	Sabine Van Huffel	Katholieke Universiteit Leuven (B)
EPLPSVD	Aboubakar Diop, Danielle Graveron-Demilly, Sophie Cavassila	Université Claude Bernard - LYON I (F)
LPSVD(CR)	Jens Totz, Wolfgang Kölbel, Ralf Heidler	Universität Leipzig (D)
	Danielle Graveron-Demilly, Aboubakar Diop	Université Claude Bernard - LYON I (F)
ER_Filter	Sophie Cavassila, Danielle Graveron-Demilly	Université Claude Bernard - LYON I (F)
MRUI interface	Aad van den Boogaart	Katholieke Universiteit Leuven (B)
	Ron de Beer	Technische Universiteit Delft (NL)

Contributions to other functions are acknowledged from:

Mika Ala-Korpela	University of Kuopio (FIN)
Ronald Behling	Bristol Myers-Squibb New Jersey (USA)
Tiziano Binzoni	Université de Genève (CH)
Koen Bruynseels	Katholieke Universiteit Leuven (B)
Miquel Cabañas	Universitat Autònoma de Barcelona (SP)
Katie Bingham	University of Nottingham (UK)
Blaise Frederick	McLean Hospital Belmont (USA)
Nanna Gillis	Katholieke Universiteit Leuven (B)
Yvonne Heidkamp	Academisch Ziekenhuis Nijmegen (NL)
Thomas Hoess	Universität Tübingen (D)
Franklyn Howe	St. George's Hospital Medical School (UK)
Jyrki Keisala	University of Oulu (FIN)
Reino Laatikainen	University of Kuopio (FIN)
Philippe Lemmerling	Katholieke Universiteit Leuven (B)
Ross Maxwell	Skejby University Hospital Arhus (DK)
Cheryl McCoy	St. George's Hospital Medical School (UK)
Dominick McIntyre	St. George's Hospital Medical School (UK)
Sarka Mierisova	Slovak Technical University (SK)
Pedro Miranda	University of Lisbon (P)
Jim Murdoch	Picker International Ohio (USA)
Richard Nolde	Yale MR Center (USA)
Takashi Ogino	Institute of Neuroscience Tokyo (J)
Alessandro Ponti	University of Milan (I)
C. Santos	The Mathworks, Inc. Natick (USA)
Manoj Saranathan	University of Washington (USA)
Jeff Stanley	University of Pittsburgh (USA)
Drea Thomas	The Mathworks, Inc. Natick (USA)

Table of Contents

1. Welcome to the MRUI manual

1.1 Dear reader,

Finally you see in front of you the first version of the official MRUI manual. This manual will be updated when new software versions are released. The main purpose of it is to present all basic knowledge necessary to handle the MRUI software package; it is impossible to go into all details of all possible combinations of ways to analyze a given data set. The manual will explain the most necessary basics of some of the signal processing methods, but in general it will refer to the appropriate publications in which you can find the theory of the implemented methods, and perhaps examples of real applications, comparisons to other methods etc. For those of you who are not too familiar with the implemented signal processing methods, and wish to find out a little more about how for example HLSVD and VARPRO work, and what their respective properties are, we recommend as an excellent starting point the lectures notes of Dr. Ron de Beer, Delft University of Technology. You can obtain these lecture notes in electronic format from the anonymous ftp server `130.161.188.140`, where you will find a file called `c59.ps.Z` in the `pub/si` directory. The file is a compressed Postscript file, which you have to uncompress after ftp using the `uncompress` command. These lecture notes can also be viewed using your Web browser, open location:

```
http://dutnsic.tn.tudelft:8080/c59_to_html/c59.html
```

You will also find a wealth of information in the MRUI WWW home pages, which are being maintained by Miquel Cabañas of the Universitat Autònoma de Barcelona, Spain:

```
http://mrui-web.uab.es/mruiwww/mrui_hom.html
```

These pages are continuously being updated and expanded and will always contain the latest information, so check these regularly. It also contains graphical examples of MRUI sessions, theoretical background information on signal processing, literature lists, Frequently Asked Questions with answers, a bug fix library for the current software version, software & hardware requirements, etc.
Further information about the software, including bug reports and fixes, release announcements, etc. will be distributed using the MRUIMAIL electronic user group platform. You can submit questions and suggestions, or share useful information, using this platform, which will reach all MRUI users worldwide. Send an e-mail to:

```
mrui-list@mrui.uab.es
```

To subscribe, unsubscribe or simply change your e-mail address, send a message to:

```
mrui-list-request@mrui.uab.es
```

Mention any problems or questions related to the list to:

```
owner-mrui-list@mrui.uab.es
```

1.2 Conventions

The following conventions will be used throughout this manual (note that all characters used for filenames, directory names, prompts, paths, commands, questions, answers etc. are in `courier` font):

characters used	meaning
`>>`	prompt in Matlab command window
`%`	user's prompt in operating system shell in UNIX this is likely to be something like `myhost%` where `myhost` is your computer's hostname (and `myuser` your username) in DOS this is likely to be something like `C:\MATLAB>`

`#`	super user's prompt in operating system shell in UNIX this is likely to be something like `#` in DOS this is likely to be something like `C:\MATLAB>`
`command arg`	a command to be typed (possibly with argument), whether in the Matlab command window or in an operating system
`question`	a question posed by a software program, to which an answer must be entered
`answer`	an answer to be entered to a question posed by a software program (identical format to command to be typed, see above)
`<name>`	the base name of a data file this base name will be used for all accompanying result files etc., which all will have a different extension
`filename.ext`	file name
`directory`	directory name
Peakpick	MRUI menu item (Arial font)

For a single-user installation, the operating system prompt is most likely to be something like `%`, or `myuser%`, or `myhost%`, whereas for a multi-user installation, usually carried out by the super-user ('root') it is most likely to be a `#`. However, for common instructions, a `#` will be used in this manual.

World Wide Web addresses, e-mail addresses, host IP addresses and variables will be in `courier` font.

When typing a command to the operating system or Matlab prompt, or typing an answer to a question posed by a program, you must press the `[Enter]` (or `[Return]`) key to execute the action. This is not explicitly printed in the text of this manual.

2. Installing the MRUI software

2.1 Software and hardware requirements

2.1.1 Required system configuration

The MRUI software runs both on UNIX workstations and IBM-compatible PC's. Because MRUI is run in the Matlab environment, the minimum system requirements are those of the Matlab package; check your Matlab manual / installation guide or the Mathworks website, see section 2.1.2. As an extreme minimum amount of RAM memory to compile and run the software, one should consider 12 Mbytes. However, the recommended amount of RAM memory is 32 Mbytes, to avoid excessive disk-paging. The amount of Mbytes in the swap partition/file should at least equal the amount of RAM. Especially on a PC it is recommended to create a large swap file (see section 2.4.6). For further information on MRUI's use of memory, see Appendix E. The amount of disk space required for a full MRUI installation depends on your operating system and compilers. On a Sun SPARC 5, running Solaris 2.5, using the SPARCworks Fortran compiler and the GNU C compiler, both without optimization, the MRUI package takes up just under 17 MB of disk space.

2.1.2 Software requirements

The signal processing algorithms in the MRUI package are Fortran 77 source code, with some auxiliary programs in C, thus a Fortran 77 compiler and a C compiler are required. If you do not have these compilers on your system, you can download the public domain GNU compilers (`gcc` and `g77`) from the net (for more information see the MRUI WWW home pages; see page 1), and install them on your local host/server. The GNU C compiler `gcc` is available as a ready-to-run binary executable for many different platforms, but the GNU Fortran 77 compiler `g77` has to be compiled on your local system using the `gcc` compiler. Alternatively, you can download the signal processing algorithms as a pre-compiled set of binary executables; these are available for a number of platforms such as Sun SPARC, SGi, HP, DEC and PC.
The MRUI graphical interface is written as a set of Matlab M-scripts and functions. This requires Matlab version 4.0 or higher, with a small number of features only available in Matlab version 4.2 or higher. For those features, MRUI will detect the running version of Matlab first and for older versions call alternative Matlab commands to avoid syntax errors.
As a complete MRS data analysis package, the MRUI software package requires both the Fortran signal processing algorithms and the Matlab-based graphical user interface.
In any case, this manual is geared towards the complete MRUI package, with the emphasis on its user-friendly graphical user interface. If you do not have the Matlab software on your system, we seriously recommend purchasing it. With an educational discount it is very cheap for such a complete and powerful mathematics and graphics package. To run the MRUI software under Matlab the basic Matlab toolbox is sufficient: you do not need to purchase any of the extra toolboxes. For all details on the Matlab software, browse through The Mathworks Web site at

```
http://www.mathworks.com/
```

or contact:

The Mathworks, Inc.
24 Prime Park Way
Natick, MA 01760-1500
USA

tel	+ 1 - 508 - 647 7000
fax	+ 1 - 508 - 647 7201
e-mail	`info@mathworks.com`

2.1.3 Supported architectures

The MRUI program has been extensively tested on the following platforms:

Sun SPARC	Solaris 1.1 - 2.5	32 MB RAM
Silicon Graphics Indy	IRIX 5.3	64 MB RAM
Hewlett Packard 712/80	HP-UX 9.05	64 MB RAM
PC Pentium 150	Windows 95	24 MB RAM
PC 486DX2/66	Windows 3.1	32 MB RAM

In principle, all UNIX and DOS/Windows platforms are supported, as long as you have the corresponding Matlab software and compilers. Computer-specific commands and separators are built in for a number of platforms (Sun, SGi, HP, IBM-RISC, DEC and PC). Some commands are also provided for VAX workstations running VMS. If your type of computer falls outside this group, you can make the software compatible to it by editing the `compat.m` script, which resides in the `$MRUIDIR/mrui/utils` directory (see section 2.3, 2.4). Do the following:

1. Type at the Matlab prompt:
```
>> computer
```
2. Make a note of the first two characters of the name that appears, e.g. `'SU'` for a Sun workstation. Upper/lower case is essential!

3. For every command in the `compat.m` script, add a set of lines:
```
elseif strcmp(c(1:2),'AB') == 1
    compretn = 'mycomnd';
```
where `AB` are your first two characters from point 2. and `mycomnd` is your computer-specific version of the commands, separators, extensions, etc. that are listed in the script.

4. Please let me know of your additions (address on page i), so that we can incorporate them in the next software upgrade. You will be acknowledged as a contributor.

2.2 Obtaining the software

To be able to obtain the software, we recommend that you use the on-line form available at the MRUI website (follow the link to “How do I get MRUI?”). In brief, you should fill out the on-line form presented there, and then submit it by pressing the “Submit License Request” button. When you do this, your request will be e-mailed for evaluation, and you will receive a copy of the message for your information. Also, the MRUI web server will send back to your browser a personalized letter, which you should print, have it signed by the head of your research group; then send it to the address in the header of the letter. Note that your request cannot be approved unless we receive this signed letter. If you are in a hurry getting the software, you can fax a copy of this letter in advance, but please indicate that the original letter is in the post. If you do not have access to the MRUI web pages, you should send an e-mail message to the MRUI distribution manager at `mrui-mgr@proton.uab.es`, after which you will receive further instructions.

It is in your own benefit if you also send details of the computer(s) you intend to install the software on. Two examples:

Sun SPARC 5;	Solaris 2.5;	32 MB RAM;	Matlab 4.2c
486DX2/66 PC;	MS-DOS 6.0 & Windows 3.1;	16 MB RAM;	PC-Matlab 4.0

If your request has been approved your software may be installed to serve an arbitrary number of computers as long as these fall under your specified research environment, and as long as it is inaccessible to persons outside that specified research environment, or to persons who could use it for industrial or commercial purposes. Section 2.3.3 explains how to set permissions to take care of this.

2.2.1 FTP

To obtain the software, the use of ftp is preferable as it is the fastest, easiest and cheapest way. You can "`get`" the software as a (compressed) tar-file, from an anonymous ftp site (managed by Miquel Cabañas, Universitat

Autònoma de Barcelona). You can obtain the password for this account from Miquel (send request to `mrui-mgr@mrui.uab.es`), which will be valid for one day. Once you know the password, you can load the tar-file containing the package `mrui963.tar` (PC users without appropriate Fortran compiler should also download `mruipcex.tar`) as follows:

```
% ftp ftp.mrui.uab.es
Connected to carbon.uab.es.
220 carbon FTP server (UNIX(r) System V Release 4.0) ready.
Name (ftp.mrui.uab.es:myuser): anonymous
331 Guest login ok, send ident as password.
Password: <enter your e-mail address here>
230 Guest login ok, access restrictions apply.
ftp> ls
200 PORT command successful.
150 ASCII data connection for /bin/ls (158.109.12.160,32922) (0 bytes).
bin
dev
etc
MRUI
usr
226 ASCII Transfer complete.
126 bytes received in 0.074 seconds (1.7 Kbytes/s)
ftp> cd MRUI
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 ASCII data connection for /bin/ls (158.109.12.160,32922) (0 bytes).
00_INDEX.TXT
DOCS
incoming
mruibugs
mruigoodies
private
ftp> cd private/<password-of-the-day>
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 ASCII data connection for /bin/ls (158.109.12.160,32922) (0 bytes).
00_INDEX.TXT
BINARIES
F77_C_SOURCE
LATEST
MATLAB
ftp> cd LATEST
250 CWD command successful.
ftp> bin
200 Type set to I.
ftp> get mrui963.tar
200 PORT command successful.
150 Binary data connection for mrui963.tar (158.109.12.160,39329) (3781120
bytes).
226 Binary Transfer complete.
local: mrui963.tar remote: mrui963.tar
3781120 bytes received in 3.6 seconds (1e+03 Kbytes/s)
ftp> bye
221 Goodbye.
%
```

This example was carried out on the local host, so transfer times will generally be slower when accessing the ftp account from anywhere else. If your experience problems with the ftp command, you can instead of the host name address `ftp.mrui.uab.es` also use the direct IP address:

```
% ftp 158.109.12.160
```

You can also ftp a compressed version of the tar-file to reduce the ftp download time. You have a choice between a compressed tar-file (`mrui963.tar.Z` or `mruipcex.tar.Z`) and a GNU gzipped tar-file (`mrui963.tar.gz` or `mruipcex.tar.gz`).
After downloading a compressed tar-file, you should uncompress it before starting the installation:

```
% uncompress mrui963.tar.Z
```

After downloading a gzipped tar-file, you should unzip it before starting the installation:

```
% gunzip mrui963.tar.gz
```

In the following, it is assumed that you have got the tar-file `mrui963.tar` onto your computer.

2.2.2 Magnetic medium

If you received the software on a magnetic medium, such as diskette, 4mm DAT tape or 1/4" cartridge, use your computer's tools to read the software off the magnetic medium and onto your computer's hard disk. For the location of the software, see section 2.3.1 or 2.4.1, depending on your operating system.

2.3 Installing the software on your computer - UNIX

2.3.1 Unpacking the MRUI tar-file

In the following, it is important to assume one system directory where you will store the MRUI program files, starting with the downloaded tar-file. For a single-user installation this could be your home directory, e.g. `/home/myuser`. For a multi-user installation, you should best use a generally accessible system directory, such as `/usr/local` We will refer to this directory as `$TARDIR`. Whenever you encounter `$TARDIR` in an instruction, replace it by the directory name you had selected. Alternatively, you can activate an environment variable `$TARDIR`, by setting it to the directory name you had selected, as follows (e.g. your directory of choice is `/usr/local`):

```
# setenv TARDIR /usr/local
```

After activating the environment variable, you can leave the `$TARDIR` in the instructions as it is, the UNIX operating system will replace it by the directory name you had entered for the environment variable.

You should extract the MRUI program files from the downloaded tar-file as follows:

```
# cd $TARDIR
# tar -xvf mrui963.tar
```

This will create a new directory under `$TARDIR`, called `MRUI_96.3`, with various subdirectories underneath. The directory `$TARDIR/MRUI_96.3` will be referred to as `$MRUIDIR`. Whenever you encounter `$MRUIDIR` in an instruction, replace it by the complete `MRUI_96.3` pathname. Alternatively, you can activate an environment variable `$MRUIDIR`, by setting it to the complete `MRUI_96.3` pathname, as follows (e.g. your `$TARDIR` of choice was `/usr/local`):

```
# setenv MRUIDIR /usr/local/MRUI_96.3
```

After activating the environment variable, you can leave the `$MRUIDIR` in the instructions as it is, the UNIX operating system will replace it by the directory name you had entered for the environment variable (not within `Makefile`, see section 2.3.2, or `.cshrc`, see section 2.3.4).

If you have no Fortran and/or C compiler on your computer system and you have downloaded statically linked binary executables programs for your specific configuration instead, you should extract the binary executables from the tar file, and store the files in the directory `$MRUIDIR/bin`. In this case you can skip the next section.

2.3.2 Compiling the Fortran and C programs

All Fortran and C compilations have been arranged in so-called Makefiles. These Makefiles organize all the necessary compilation and linking for you, but rely on you to edit a number of system-specific variables, such as the command name for the Fortran and/or C compiler, compiler options, and the target directory for the binary executables. The required changes are summarized in each sub-section, and detailed within the header of each Makefile, which can be edited as a normal ASCII text file.

Compile and distribute the conversion programs

```
# cd $MRUIDIR/lib/fortio
```

In the `Makefile`, edit the variables `FCOMP` (default is `f77`) and `FOPTS` (default is none).

```
# make all
# cd $MRUIDIR/a2b
```

In `Makefile`, edit the variables `CCOMP` (default is `gcc`), `COPTS` (default is `-traditional`), `FCOMP` (default is `f77`), `FOPTS` (default is none), `STD_LIB` (default is `-lF77`) and IEEE_LIB (default is `-lc`). For a multi-user installation, edit the variables `TARG_DIR` (default is `$HOME/MRUI_96.3/bin`) and `IOIB_DIR` (default is `$HOME/MRUI_96.3/lib/fortio`).

```
# make all
# make install
# make clean
```

Compile and distribute the LAPACK library

```
# cd $MRUIDIR/lib/lapack
```

In the `Makefile`, edit the variables `FCOMP` (default is `f77`) and `FOPTS` (default is none). For a multi-user installation, edit the variables `SVTARG_DIR` (default is `$HOME/MRUI_96.3/lib/svd`), `TLTARG_DIR` (default is `$HOME/MRUI_96.3/lib/tls`) and `CRTARG_DIR` (default is `$HOME/MRUI_96.3/lib/cr`)

```
# make all
# make install
# make clean
```

Compile and distribute the H(L)SVD and HTLS programs

```
# cd $MRUIDIR/hsvd
```

In the `Makefile`, edit the variables `FCOMP` (default is `f77`), `FOPTS` (default is none) and `STD_LIB` (default is`-lF77`). For a multi-user installation, edit the variables `TARG_DIR` (default is `$HOME/MRUI_96.3/bin`), `IOLIB_DIR` (default is `$HOME/MRUI_96.3/lib/fortio`), `SVLIB_DIR` (default is `$HOME/MRUI_96.3/lib/svd`) and `TLLIB_DIR` (default is `$HOME/MRUI_96.3/lib/tls`).

```
# make all
# make install
# make clean
```

Compile and distribute the (EP)LPSVD(CR) programs

```
# cd $MRUIDIR/lpsvd
```

In the `Makefile`, edit the variables `FCOMP` (default is `f77`), `FOPTS` (default is none) and `STD_LIB` (default is`-lF77`). For a multi-user installation, edit the variable `TARG_DIR` (default is `$HOME/MRUI_96.3/bin`).

```
# make all
# make install
# make clean
```

Compile and distribute the VARPRO programs

```
# cd $MRUIDIR/varpro
```

In the `Makefile`, edit the variables `FCOMP` (default is `f77`), `FOPTS` (default is none) , `HOPTS` (default is none) and `STD_LIB` (default is `-lF77`). For a multi-user installation, edit the variables `TARG_DIR` (default is `$HOME/MRUI_96.3/bin`) and `IOLIB_DIR` (default is `$HOME/MRUI_96.3/lib/fortio`).

```
# make all
# make install
# make clean
```

Compile and distribute the AMARES programs

```
# cd $MRUIDIR/amares
```

In the `Makefile`, edit the variables `FCOMP` (default is `f77`), `FOPTS` (default is none) , `HOPTS` (default is none) and `STD_LIB` (default is none). For a multi-user installation, edit the variables `TARG_DIR` (default is `$HOME/MRUI_96.3/bin`) and `IOLIB_DIR` (default is `$HOME/MRUI_96.3/lib/fortio`).

```
# make all
# make install
# make clean
```

Compile and distribute the tool programs

```
# cd $MRUIDIR/tools
```

In the `Makefile`, edit the variables `FCOMP` (default is `f77`), `FOPTS` (default is none) and `STD_LIB` (default is `-lF77`). For a multi-user installation, edit the variables `TARG_DIR` (default is `$HOME/MRUI_96.3/bin`), `CRLIB_DIR` (default is `$HOME/MRUI_96.3/lib/cr`) and `IOLIB_DIR` (default is `$HOME/MRUI_96.3/lib/fortio`).

```
# make all
# make install
# make clean
```

Now that you have done all the compilation, you may want to delete the remaining object files, which are no longer needed:

```
# cd $MRUIDIR/lib
# rm -f fortio/*.o cr/*.o svd/*.o tls/*.o
```

Distribute auxiliary programs

In the directory `$MRUIDIR/bin`, four programs were already present: `convkit`, `machead1`, `machead3` and `mruiusersetup`. These program scripts should be distributed to the directory where the compiled executable programs have been stored (via the `TARG_DIR` variables in the respective Makefiles). For a single-user installation, no action is needed here if you did not change the default target directory in the Makefiles, because then all binary executable program files are automatically stored in `$MRUIDIR/bin`. However, if you changed the `TARG_DIR` variables in the Makefiles, to store the binary executable program files in a generally accessible directory (e.g. for a multi-user installation), then you must now also copy these three program files to that target directory.

2.3.3 Setting the file permissions

Under the MRUI license, you are expected to install the software on your computer in such a way, that users from outside your group cannot freely access or copy the files (see "Obtaining the software" on page 4). You should protect the MRUI files against access from outside your user group by setting the so-called UNIX file permissions. This is done using the UNIX command `chmod`. For full information on this command, type:

```
% man chmod
```

If you perform a single-user installation, with the `MRUI_96.3` directory under your home directory, you can set the file permissions such that only you can access the MRUI files:

```
% cd
% chmod -R go-rxw MRUI_96.3
% chmod go-rxw mrui963.tar
```

For a multi-user installation, the super-user should perform this action in the `$TARDIR` directory:

```
# cd $TARDIR
# chmod -R o-rxw MRUI_96.3
# chmod o-rxw mrui963.tar
```

2.3.4 Preparing the environment variables for users

For users to be able to run MRUI under Matlab, the compiled binary executable programs from section 2.3.2 must be included in the UNIX path, and the Matlab scripts under the `$MRUIDIR/mrui` directory must be included in the Matlab path. This is best done in the `.cshrc` file in the user's home directory. For a multi-user installation the super-user should do this for all users that wish to use the MRUI software package. To include the `TARG_DIR` directory, containing the compiled binary executable programs, in the UNIX path variable, add the following line in the `.cshrc` file (in this case you really have to insert the name of the `$MRUIDIR` directory, see section 2.3.1):

```
      set path = ( $MRUIDIR/bin $path )
```

To include all directories under `$MRUIDIR/mrui` in the Matlab path variable, it is easiest to use the files `$MRUIDIR/info/mrui.set` and `$MRUIDIR/info/mrui.profile` as templates. First make sure that the general directory name in these files corresponds to your `$MRUIDIR` directory. Then copy the template files `mrui.set` and `mrui.profile` to the home directory (of each user when performing a multi-user installation) under the name `.mrui` and `.mrui.profile`, respectively. You can then 'activate' all these directories for Matlab by adding the following line in the `.cshrc` file:

```
      source $HOME/.mrui
```

Each user can do this automatically using the `mruiusersetup` script. Before using it, the installer must make sure that the directory pointer in the script (`/usr/local/MRUI_96.3/work_dir`) corresponds to where you have installed the MRUI program files. For a first-time MRUI installation, each user can then activate her/his own MRUI settings by running (also works under the Bourne shell):

```
% mruiusersetup
```

If you upgrade from a previous version of MRUI, and you already have an existing `.mrui` file, or a set of lines in your `.cshrc` or `.profile` file(s) that activate the Matlab MRUI directories, then check with the template files in `$MRUIDIR/info` whether there have been any additions or modifications.

The `mruiusersetup` script also creates for each user their personal `$HOME/matlab/work_dir` directory, from which the Matlab program is best started up. The `work_dir` directory should contain the following working files, ***WHICH SHOULD NEVER BE DELETED***:

```
            barc.awk                fftcheck.fil
            barc_par.awk            mrui.ini
            default.gra             mrui.def
```

If the `mruiusersetup` script has not been used by a user, (s)he can accomplish this by copying the template `work_dir` directory from the `$MRUIDIR` directory. The super-user can also do this for each user (remember to set the ownership correctly after copying or creating new directories as 'root'). It is assumed that each user already has a directory called `matlab` under her/his home directory, as recommended by the Matlab Installation Guide.

```
% cd
% cd matlab
% cp -r $MRUIDIR/work_dir .
```

For each upgrade, check if the list above contains any new entries that do not appear in your `work_dir` directory. It will be notified in new versions of the manual or information files if one of the files has been modified and requires re-copying. Two files that will definitely change with each upgrade are `mrui.ini` and mrui.def. The file `$MRUIDIR/work_dir/mrui.def` should be copied over the older version in your personal `work_dir` directory for each MRUI upgrade. After the introduction of the customization menus in MRUI_96.1, with the settings saved in `mrui.ini`, the format and contents of that file will change in version upgrades according to additions or modifications to the customization menus. To preserve your personal settings that you have saved to your `$HOME/matlab/work_dir/mrui.ini` file in a previous MRUI version, you should convert it to the new version format, so that only the new features are added to it at the appropriate places. This can be done fully automatically by typing to the Matlab prompt:

```
>> opt_upg;
```

You ***MUST*** do this ***BEFORE*** you start your the new MRUI version for the first time, because MRUI needs a correct `$HOME/matlab/work_dir/mrui.ini` to run.

2.3.5 Automatic MRUI startup in Matlab

If you want MRUI to start up automatically when Matlab starts up, create a file called `startup.m` in your personal `~/matlab` directory; it should contain the following line:

```
mrui
```

You could also use this Matlab startup script to bring you to your preferred directory before you start using MRUI, e.g.:

```
cd ~/matlab/work_dir/my_data
mrui
```

2.4 Installing the software on your computer - Windows / DOS

2.4.1 Storing the software on your hard disk

In the following, it is important to assume one system directory where you will store the MRUI program files. It is assumed that Matlab is installed in `C:\MATLAB`, with a toolbox directory below, called `C:\MATLAB\TOOLBOX`. If you installed Matlab in a different directory, replace the directory names in the following with the appropriate directory names. It is most convenient to install MRUI as if it were just another Matlab toolbox, called MRUI_96.3:

```
% cd \matlab\toolbox
% mkdir mrui_96.3
```

If you used ftp to download the (compressed) tar-file, use your own PC's software to (uncompress and) extract the files from the tar-file, and store the directory structure under `C:\MATLAB\TOOLBOX\MRUI_96.3`. If you downloaded the binary executables, store them in `C:\MATLAB\TOOLBOX\MRUI_96.3\BIN`. If you received the MRUI files on diskettes, store them on your hard disk under the directory `C:\MATLAB\TOOLBOX\MRUI_96.3` so that you get the tree structure as shown in Figure 1.

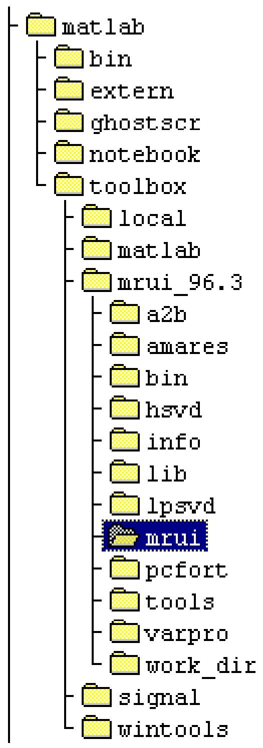


Figure 1. The MRUI directory tree structure on your hard disk.

2.4.2 Compiling the Fortran and C programs yourself

MRUI for PC comes with ready-to-run binary executables (`*.EXE`) for most programs in MRUI, so that you do not have to compile. Once the files have been copied to the appropriate locations, and the environment variables have been properly set (see section 2.4.4), you can immediately start using MRUI. However, if you have a good Fortran compiler for your PC (we recommend Microsoft Fortran PowerStation), you will benefit in terms of computation times and memory usage if you compile the source files yourself.

One drawback of the PC version of MRUI, is that we have not found a satisfactory solution for those programs that link Fortran to C code. Should you have a solution, please contact the author of this manual (see page i) as soon as possible. The programs that can currently not be used on PC:

program name	relates to MRUI function
`a2b`	conversion `*.dat`/`*.mat` (all others functions in `a2b` intact)
`bru2sun`	conversion of Bruker ASPECT files
`fel2mat`	conversion Matlab to Felix format (necessary for running (EP)LPSVD(CR))
`mat2fel`	conversion Felix to Matlab format (necessary for running (EP)LPSVD(CR))

Especially the `a2b` compilation may cause worries, because it carries out the vital conversion between `*.dat` and `*.mat` files, which is often during a typical MRUI session. However, the Matlab scripts have been adapted so that for PCs, the `*.dat`/`*.mat` conversion is automatically taken over by a Matlab script, so that you do not have to rely on linking the Fortran and C code for the program `a2b`.
The compilers used to create the executable programs that are distributed with MRUI were:

Microsoft Fortran 77 PowerStation Compiler	version 1.0
Microsoft C Optimizing Compiler	version 6.00

The program `NMAKE` used for the Makefiles in this instruction was supplied with the Fortran compiler.

All Fortran and C compilations have been arranged in so-called Makefiles (`*.mak`). These Makefiles organize all the necessary compilation and linking for you, but rely on you to edit a number of system-specific variables, such as the command name for the Fortran and/or C compiler, compiler options, and the target directory for the binary executables. The required changes are summarized in each sub-section, and detailed within the header of each Makefile, which can be edited as a normal ASCII text file.

Finally, note that for the Microsoft compiler, all Fortran files must be called `*.for` instead of `*.f`, before PC compilation can be successful.

Compile and distribute the conversion programs

```
% cd \matlab\toolbox\mrui_96.3
% cd lib\fortio
% ren *.f *.for
```

In the Makefile `fortio.mak`, edit the variables `FCOMP` (default is `fl32`) and `FOPTS` (default is `/G4 /Ox /nologo`).

```
% nmake /NOLOGO /F fortio.mak all
% cd ..\..\a2b
% ren *.f *.for
% copy ..\pcfort\a2b\*.for .
% copy ..\pcfort\a2b\*.c .
```

In the Makefile `a2b.mak`, edit the variables `CCOMP` (default is `cl`), `COPTS` (default is `/AH /G2 /Ot /nologo`), `CLINK` (default is `cl`), `CLOPTS` (default is `/nologo`), `FCOMP` (default is `fl32`), `FOPTS` (default is `/G4 /Ox /nologo`), `FLINK` (default is `fl32`) and `FLOPTS` (default is `/nologo`). You may also want to edit the variables `TARG_DIR` (default is `..\bin`) and `IOLIB_DIR` (default is `..\lib\fortio`).

```
% nmake /NOLOGO /F a2b.mak all
% nmake /NOLOGO /F a2b.mak install
% nmake /NOLOGO /F a2b.mak clean
```

Compile and distribute the LAPACK library

```
% cd ..\lib\lapack
% ren *.f *.for
```

In the Makefile `lapack.mak`, edit the variables `FCOMP` (default is `fl32`) and `FOPTS` (default is `/W0 /nologo`). You may also want to edit the variables `SVTARG_DIR` (default is `..\svd`), `TLTARG_DIR` (default is `..\tls`) and `CRTARG_DIR` (default is `..\cr`).

```
% nmake /NOLOGO /F lapack.mak all
% nmake /NOLOGO /F lapack.mak install
% nmake /NOLOGO /F lapack.mak clean
```

Compile and distribute the H(L)SVD and HTLS programs

```
% cd ..\..\hsvd
% ren *.f *.for
% copy ..\pcfort\hsvd\*.for .
```

In the Makefile `hsvd.mak`, edit the variables `FCOMP` (default is `fl32`), `FOPTS` (default is `/G4 /Ox /nologo`), `FLINK` (default is `fl32`) and `FLOPTS` (default is `/nologo`). You may also want to edit the variables `TARG_DIR` (default is `..\bin`), `IOLIB_DIR` (default is `..\lib\fortio`), `SVLIB_DIR` (default is `..\lib\svd`) and `TLLIB_DIR` (default is `..\lib\tls`).

```
% nmake /NOLOGO /F hsvd.mak all
% nmake /NOLOGO /F hsvd.mak install
% nmake /NOLOGO /F hsvd.mak clean
```

Compile and distribute the (EP)LPSVD(CR) programs

```
% cd..\lpsvd
% ren *.f *.for
% copy ..\pcfort\lpsvd\*.for .
```

In the Makefile `lpsvd.mak`, edit the variables `FCOMP` (default is `fl32`), `FOPTS` (default is `/G4 /Ox /W0 /nologo`), `FLINK` (default is `fl32`) and `FLOPTS` (default is `/nologo`). You may also want to edit the variable `TARG_DIR` (default is `..\bin`).

```
% nmake /NOLOGO /F lpsvd.mak all
% nmake /NOLOGO /F lpsvd.mak install
% nmake /NOLOGO /F lpsvd.mak clean
```

Compile and distribute the VARPRO programs

```
% cd ..\varpro
% ren *.f *.for
% copy ..\pcfort\varpro\*.for .
```

In the Makefile `varpro.mak`, edit the variables `FCOMP` (default is `fl32`), `FOPTS` (default is `/G4 /Ox /nologo`), `FLINK` (default is `fl32`) and `FLOPTS` (default is `/nologo`). You may also want to edit the variables `TARG_DIR` (default is `..\bin`), `CRLIB_DIR` (default is `..\lib\cr`) and `IOLIB_DIR` (default is `..\lib\fortio`).

```
% nmake /NOLOGO /F varpro.mak all
% nmake /NOLOGO /F varpro.mak install
% nmake /NOLOGO /F varpro.mak clean
```

Compile and distribute the AMARES programs

```
% cd ..\amares
% ren *.f *.for
```

In the Makefile `varpro.mak`, edit the variables `FCOMP` (default is `fl32`), `FOPTS` (default is `/G4 /Ox /nologo`), `FLINK` (default is `fl32`) and `FLOPTS` (default is `/nologo`). You may also want to edit the variables `TARG_DIR` (default is `..\bin`), `CRLIB_DIR` (default is `..\lib\cr`) and `IOLIB_DIR` (default is `..\lib\fortio`).

```
% nmake /NOLOGO /F amares.mak all
% nmake /NOLOGO /F amares.mak install
% nmake /NOLOGO /F amares.mak clean
```

Compile and distribute the tool programs

```
% cd ..\tools
% ren *.f *.for
% copy ..\pcfort\tools\*.for .
```

In the Makefile `tools.mak`, edit the variables `FCOMP` (default is `fl32`), `FOPTS` (default is `/G4 /Ox /nologo`), `FLINK` (default is `fl32`) and `FLOPTS` (default is `/nologo`). You may also want to edit the variables `TARG_DIR` (default is `..\bin`), `CRLIB_DIR` (default is `..\lib\cr`) and `IOLIB_DIR` (default is `..\lib\fortio`).

```
% nmake /NOLOGO /F tools.mak all
% nmake /NOLOGO /F tools.mak install
% nmake /NOLOGO /F tools.mak clean
```

Now that you have done all the compilation, you may want to delete the remaining object files, which are no longer needed:

```
% cd ..\lib
% del fortio\*.obj
% del cr\*.obj
% del svd\*.obj
% del tls\*.obj
```

2.4.3 Installing the working directory

If this is your first MRUI installation on PC, you must make a copy of the provided `WORK_DIR` directory under the `MATLAB` directory:

```
% cd \matlab
% mkdir work_dir
% cd work_dir
% copy \matlab\toolbox\mrui_96.3\work_dir\*.* .
```

This directory would further be a good position to store your data for MRUI analysis, in different subdirectories, see also section 4.1.3.

The following files in the `WORK_DIR` directory ***SHOULD NEVER BE DELETED***:

BARC.AWK	FFTCHECK.FIL
BARC_PAR.AWK	MRUI.INI
DEFAULT.GRA	MRUI.DEF

For each upgrade, check if the list above contains any new entries that do not appear in your `WORK_DIR` directory yet, and if there are copy them from the distributed `MRUI_96.3\WORK_DIR` directory.

If this installation is an upgrade of a previous MRUI version on your PC, and you already have a `MATLAB\WORK_DIR` directory, you only need to copy the `MRUI.DEF` file and upgrade your `MRUI.INI` file:

```
% cd \matlab\work_dir
% copy \matlab\toolbox\mrui_96.3\work_dir\mrui.def .
```

After the introduction of the customization menus in MRUI_96.1, with the settings saved in `MRUI.INI`, the format and contents of that file will change in version upgrades according to additions or modifications to the customization menus. To preserve your personal settings that you have saved to your `\MATLAB\WORK_DIR\MRUI.INI` file in a previous MRUI version, you should convert it to the new version format, so that only the new features are added to it at the appropriate places. This can be done fully automatically by typing to the Matlab prompt:

```
>> opt_upg;
```

You ***MUST*** do this ***BEFORE*** you start your the new MRUI version for the first time, because MRUI needs a correct `\MATLAB\WORK_DIR\MRUI.INI` to run. If you already have a `\MATLAB\WORK_DIR` directory but you do not have a previous version of an `MRUI.INI` file, just copy the one provided with the software (`MRUI_96.3\WORK_DIR\MRUI.INI`) to your `\MATLAB\WORK_DIR` directory.

2.4.4 Setting the environment variables

To be able to run MRUI under Matlab, the compiled binary executable programs from sections 2.4.1 and 2.4.2 must be included in the DOS path, and the Matlab scripts under the `MRUI_96.3\MRUI` directory must be included in the Matlab path. To include the `MRUI_96.3\BIN` directory, containing the compiled binary executable programs, in the DOS path variable, add the full directory name (e.g. `C:\MATLAB\TOOLBOX\MRUI_96.3\BIN`) to the `PATH` line in the `C:\AUTOEXEC.BAT` file, e.g.:

```
      PATH C:\MATLAB\TOOLBOX\MRUI_96.3\BIN;C:\DOS;C:\WINDOWS
```

In general, it is recommend that the `MRUI_96.3\BIN` directory be at the beginning of the path. Placing the `MRUI_96.3\BIN` at the beginning of your DOS path will not affect any of your system's properties while using your normal programs.

To include all directories under `MRUI_96.3\MRUI` in the Matlab path variable, you need to add the settings in `MRUI_96.3\INFO\MRUI_PC.SET` to `C:\MATLAB\MATLABRC.M` or `C:\MATLAB\TOOLBOX\LOCAL\STARTUP.M`. Use any text editor for this (e.g. Windows Notepad or Wordpad). You can immediately insert the contents of `MRUI_PC.SET` into the `MATLABRC.M` (below the section where the variable `MATLABPATH` is defined) or `STARTUP.M`, if you installed the MRUI software in `C:\MATLAB\TOOLBOX\MRUI_96.3`). If you installed `MRUI_96.3` in a different directory, you must edit the corresponding lines in `MRUI_PC.SET` accordingly.

If you upgrade from a previous version of MRUI, and you already have an existing set of lines in your `C:\MATLAB\MATLABRC.M` or `C:\MATLAB\TOOLBOX\LOCAL\STARTUP.M` file that activate the Matlab MRUI directories, then check with the template file `MRUI_96.3\INFO\MRUI_PC.SET` whether there have been any additions or modifications.

In C:\AUTOEXEC.BAT, you must insert a HOME setting for the directory above your MATLAB\WORK_DIR directory. If you have, as suggested, C:\MATLAB\WORK_DIR, add in C:\AUTOEXEC.BAT (omit the drive specification C:):

```
set HOME=\
```

2.4.5 Enabling 32 bit Fortran programs

The binary executable *.EXE programs that come with MRUI (you may skip this point if you have Microsoft Fortran PowerStation installed on your PC) are 32 bit programs. To enable their use on your PC, the programs DOSXMSF.EXE and DOSXNT.386 are provided in the MRUI_96.3\BIN directory. You must add a line in your C:\WINDOWS\SYSTEM.INI file to enable their use under your Windows software. Under the [386Enh] section in that file, add the entry:

```
device=C:\MATLAB\TOOLBOX\MRUI_96.3\BIN\DOSXNT.386
```

2.4.6 Increasing the size of the swap file

Especially on a PC it is a good idea to have a large swap file. As a rule of thumb, create a swap file three times the size of your physical RAM memory. Use the Windows "Control Panel" for this, section "386 Enhanced", "Virtual Memory". Use a permanent swap file. Even though the recommended size may be smaller than the maximum size, the surplus of the swap file can be used by Matlab if not by Windows, so create a swap file as large as your system allows (given the amount of free space you wish to reserve for data etc.).

2.4.7 Tweaking Matlab's Windows / MS-DOS interaction

Matlab under Windows has one irritating property: any shell escape (a command run directly under the operating system) brings up a DOS window. And there are *many* shell commands in MRUI. The default Matlab installation brings up the DOS window and halts there. For MRUI to continue you would need to close every window as it came along. To make sure the DOS windows appear automatically as soon as Matlab is done with them, edit the files C:\MATLAB\BIN\ML_BANG.PIF and C:\MATLAB\BIN\ML_DOS.PIF. Under Windows 3.1 and 3.11 you should do this using the PIF editor (most likely in the "Main" group). In the PIF files, there is a check box in the lower left corner marked "Close window on exit", which will be unchecked by default. Checking this box and saving the PIF file will automatically close the inactive DOS windows and make MRUI continue as it should. Under Windows 95, use "My computer" to go to C:\MATLAB\BIN. There you will see two files with the MSDOS icon: Ml_bang and Ml_dos. For both, the "Properties" should be changed. So select the Ml_bang, go to the "File" menu item and select "Properties". You will see the window "Ml_bang Properties", in which you must select the second tab-page, titled "Program", and there at the bottom is a box saying "Close on exit" which must be checked. Repeat this for Ml_dos.

2.4.8 If you use home-built software to read MRUI files

There is one strange difference of Matlab under MS Windows as compared to Matlab under UNIX. It does not harm the MRUI scripts or Fortran programs, but you should be aware of it in case you have home-built software that reads either the PEAKPICK.PAR file (or <NAME>.PKP, see chapter 11), the <NAME>.HNS file (see chapter 10) or the <NAME>.VNS (see chapter 13) file. It is the fact that numerical values get written into these files by Matlab scripts, in the exponential format. This means that a value of e.g. 15,000 gets written as 1.500e+04. At least, that is what you would expect, and that is what happens on all UNIX systems. However, Matlab for PC writes to the value to the file in the format 1.500E+004, i.e. with three digits for the exponent, instead of two. When contacting The Mathworks about it, they were most intrigued and thanked me for pointing this out to them. It had not occurred to them yet. They assured us that it got their full attention, and that they would see into possibilities of new variables and options that will allow control over the exponential format in future Matlab versions. In the mean time, you may want to check your scripts that read the files concerned, to make sure that their performance is not impeded by this difference.

2.4.9 Automatic MRUI startup in Matlab

If you want MRUI to start up automatically when Matlab starts up, create a file called `startup.m` in your personal `C:\MATLAB` directory; it should contain the following line:

```
mrui
```

You could also use this Matlab startup script to bring you to your preferred directory before you start using MRUI, e.g.:

```
cd \matlab\work_dir\my_data
mrui
```

3. Starting MRUI; the Graphical User Interface

3.1 Starting MRUI: the base window

To start MRUI, you have to start up Matlab first. From the Matlab command prompt, call `mrui`. If you organize your data under the `$HOME/matlab/work_dir` directory (see section 4.1.3), you might want to change directory there first:

```
% cd $HOME/matlab/work_dir
% matlab
>> mrui
```

This will bring up the MRUI base window, the center of the Graphical User Interface (GUI), from which you call the various signal processing functions. For an example of the MRUI base window, see Figure 2. The figures in this manual were produced running MRUI on a PC running MS-Windows 3.1, and the Matlab graphical features, such as menus and buttons, may look slightly different on an X Windows platform.

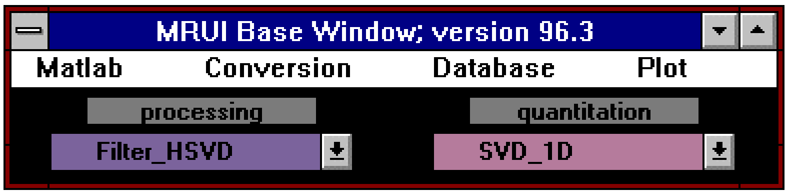


Figure 2. The MRUI base window, containing the main menu.

The various signal processing functions will be described in the following chapters. The two main popup menus, called Processing and Quantitation contain the main function for quantitative data processing:

Popup menu	Function	see
processing	Filter_HSVD	chapter 9
	FID Maths	chapter 8
	Simulate	chapter 6
quantitation	SVD_1D	chapter 10
	Peakpick	chapter 11
	Inputvp VARPRO	chapter 12
	Inputnv AMARES	chapter 14
	VARPRO AMARES	chapter 13

The menus at the top of the MRUI base window are for miscellaneous functions. The Matlab menu will be described in section 3.2, the Conversion menu in chapter 4, the Database menu in chapter 7 and the Plot menu in chapter 16.

In the MRUI graphical interface, all interaction is controlled by windows, menus, buttons, etc. However, it is recommended not to iconize the basic Matlab command window, as it may contain error messages, output messages from programs running in the shell (UNIX or DOS) or sporadically even ask for input from the user (the routines in which this is still necessary are under development to avoid this). Run Matlab from a window that allows scrolling, so you can read the contents of the screen even if these are longer than the default number of lines for your window.

3.2 The Matlab menu

The Matlab menu under the MRUI base window contains a set of Matlab commands that help you modify the look of the interface according to your personal favor, set defaults for all input parameter to be used in any signal processing function, navigate through your data directories, and make things behave if they do not. The submenus and menu items are explained in the following.

3.2.1 customize

The customize menu helps you make MRUI behave the way you want to, including window sizes and positions, colors, preferences and input parameters to signal processing functions. All these settings are stored in the `mrui.ini` file in the `work_dir` directory. Originally this will be the `mrui.ini` distributed with the MRUI package; you had copied it from `$MRUIDIR/work_dir` (see section 2.3.4) together with the file `mrui.def`, which has the same structure as `mrui.ini`, but contains all the default settings as distributed with the package. Once you start changing the default settings with the customize menu, your personal settings will be stored in your `matlab/work_dir/mrui.ini` file. The contents of `mrui.ini` have then changed, but `mrui.def` is still in its original state. This allows you to restore the package default settings at any time, by selecting in the customize menu package "default & exit". Once you start making changes in the settings, a third version of the settings file will be created in the `work_dir`: `mrui.bak`. This file provides a backup copy of your previous settings. When you run customize, `mrui.ini` is automatically copied to `mrui.bak`, so that despite the changes you make, you can always fall back on your previous situation via:

```
% cd ~/matlab/work_dir
% cp mrui.bak mrui.ini
```

This may also be useful, should you accidentally delete your `mrui.ini` file. Should you for some reason have lost all `mrui.*` files in your `work_dir` directory, your only option is to copy a new set from the `$MRUIDIR/work_dir` directory (see section 2.3.4) and start again from the package defaults (unless you can retrieve parts of your personal mrui.ini from a system backup you or the system manager made earlier).
When changing settings in any of the customize functions, you have three options to exit the function:

save & exit	Saves your changes in the `mrui.ini` file, with a backup copy of your previous settings still available in the `mrui.bak` file.
exit without save	Annihilates all the changes you made in the current customize function, and returns to the previous settings.
package defaults & exit	Copies the `mrui.def` over `mrui.ini`, thereby restoring the original MRUI default settings. A backup copy of your previous personal settings still exists in `mrui.bak`.

3.2.1.1 customize windows

The customize windows function consists of a popup menu containing a list of all windows used in the MRUI graphical interface, see Figure 3.

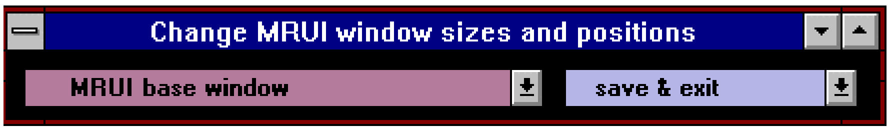


Figure 3. The customize window function with its popup menu

Select the name of the window of which you want to change default size and position, and that window will appear, inactive (i.e. you cannot use it for its normal functions, it appears only so that you can reposition it according to your preference). Should a window already be open (e.g. the base window and the customize window will necessarily be open), then it will remain active. Next a message window will appear, saying that you can now reposition the window according to your wishes, and then press the OK button to save the new positions in the `mrui.ini` file. When you press the OK button, both the repositioned window and the message window will disappear (not if the window you wanted to change was already open, see above). You can repeat this process for as many window on the list as you wish.
Notice that the window you wish to reposition will appear first, and then the message window, which will at that time be the current window (active). Hence you will have to click on the bottom window to make it the current window, before you can reposition it. This may cause some confusing overlap. If you want to reposition a large window, then making it the current window will bring it to the front, and hide the message window. After repositioning your window, you must press the OK button, for which the message window needs to come to the front. You can either do this using your X-Windows or MS-Windows task manager, or simply iconize the repositioned window by clicking the corresponding button in the top right corner of the window. Iconizing your

window will not affect the position parameters that the customize function stores, even though the icon will be placed at the border of your screen. If you wish to permanently change the default size and position of the message window, or one of the other question windows with the same size and position, the top message window will fully overlap the window to be repositioned. In this case it is easiest to temporarily move the top message window a little out of the way. Beware that you can only press the OK button on the top message window (the one displayed for repositioning is inactive).

3.2.1.2 customize colors

The customize colors function brings up a window with three columns of color settings. One for the background colors (bcol*), one for the foreground colors (fcol*), and one for the miscellaneous colors (acol*). See Figure 4.

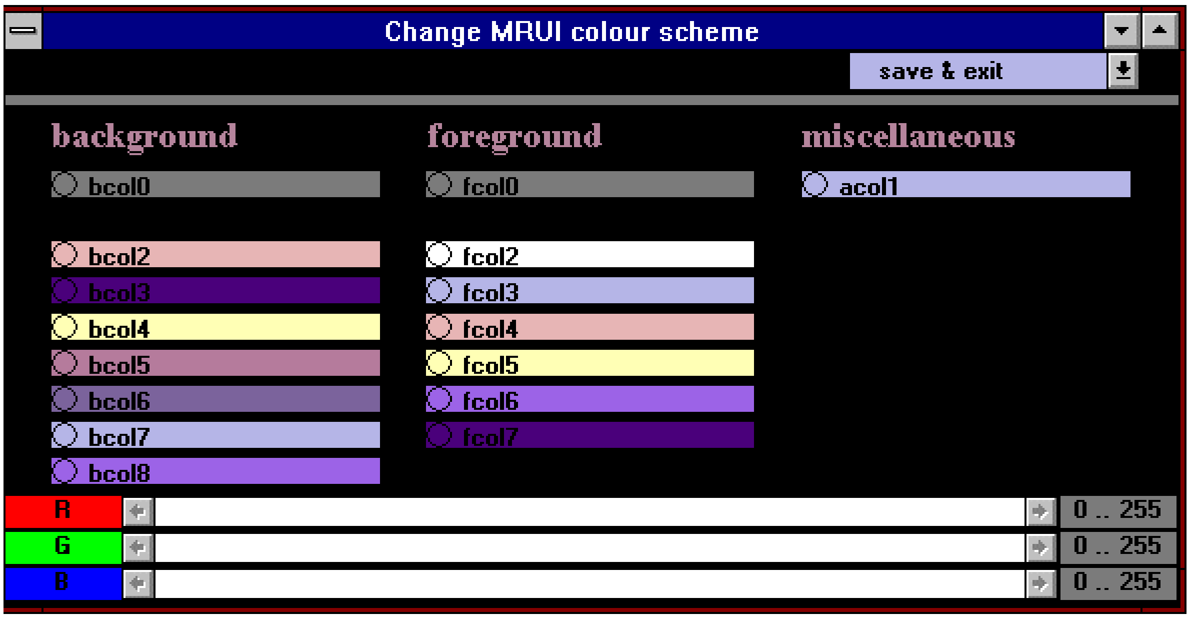


Figure 4. The customize colors window with its RGB sliders

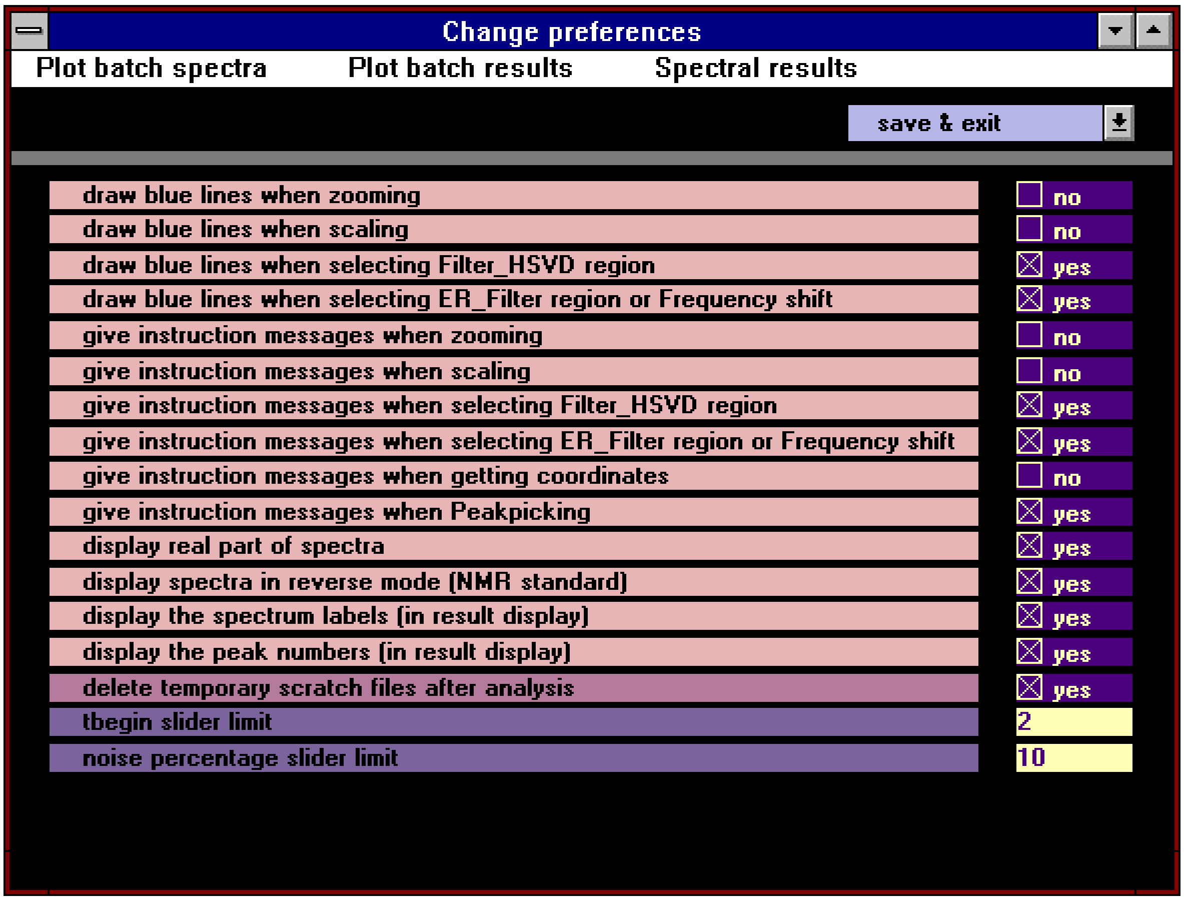
The individual color settings are displayed as a Matlab-type radiobutton, with the color of their current setting. Changing one of these colors will change all occurrences of that (background, foreground or miscellaneous) color in the MRUI software. At the bottom of the window, three sliders are displayed, initially disable (inactive), one for each red (R), green (G) or blue (B) fraction of a color. This is the default color scheme format on most computer systems. To change one of the MRUI colors, just click on its radiobutton, upon which the RGB sliders will become enable (active), and the current RGB values of the color (values between 0 and 255) you clicked will be listed in the text boxes on the right of the sliders, and represented by the position of the respective sliders. Note also the appearance of a new button labeled apply. Now you can change the selected color using the three sliders. There are three ways of using a slider in Matlab, and hence in MRUI:

1. By clicking on an arrow sign at either end of a slider, you make small steps in that direction.
2. By clicking on the bar between the slider and the arrow sign, you make larger steps in that direction.
3. By positioning the mouse cursor over the actual slider, then clicking the left mouse key and keeping it depressed, you can drag the slider along in either direction over an arbitrary distance.

As you move the RGB sliders along, the color of the selected radiobutton will change accordingly, and the RGB values in the text boxes will be updated as well. Once you are happy with the new color, press the apply button. The new color settings will then be put in a format to be stored in the `mrui.ini` file once you exit this function. You can similarly selected other radiobuttons for different colors, and use the RGB sliders to change their color settings. Remember to press the apply button each time you have finished changing a color, otherwise its new settings will not be stored in the `mrui.ini` file, and hence be forgotten when you exit this function.

3.2.1.3 *customize preferences*

The customize preferences function brings up a list of various options that occur throughout the MRUI interface, see Figure 5.



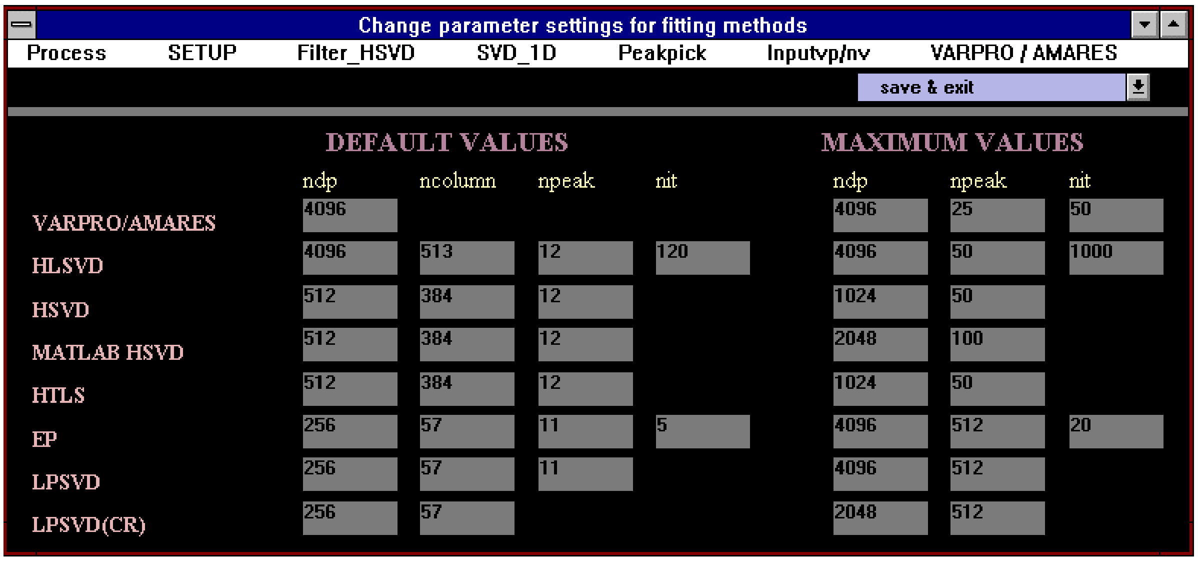
draw blue lines when zooming	☐ no
draw blue lines when scaling	☐ no
draw blue lines when selecting Filter_HSVD region	☒ yes
draw blue lines when selecting ER_Filter region or Frequency shift	☒ yes
give instruction messages when zooming	☐ no
give instruction messages when scaling	☐ no
give instruction messages when selecting Filter_HSVD region	☒ yes
give instruction messages when selecting ER_Filter region or Frequency shift	☒ yes
give instruction messages when getting coordinates	☐ no
give instruction messages when Peakpicking	☒ yes
display real part of spectra	☒ yes
display spectra in reverse mode (NMR standard)	☒ yes
display the spectrum labels (in result display)	☒ yes
display the peak numbers (in result display)	☒ yes
delete temporary scratch files after analysis	☒ yes
tbegin slider limit	2
noise percentage slider limit	10

Figure 5. The customize preferences window.

Here you can change their default settings. The preferences for the plot functions (Plot batch spectra and Plot batch results) can be changed using the menus as described in their corresponding sections in this manual, and the settings of the Spectral results menu are described in section 5.2.1. Most other default preferences are controlled by the checkboxes on the right, which you can switch between yes and no by clicking on them. The bottom two settings contain default values for sliders and should be changed by editing the edit/text box. Most settings are totally up to the preference and experience of each individual user, but it may be useful to know, once you get more experienced with the software, that *not* displaying auxiliary blue lines and instruction messages (in the text box in the bottom right corner of the spectral window, see e.g. section 5.1.2.1) accelerates the corresponding functions. See the corresponding sections for more detailed information on the various options.

3.2.1.4 *customize fitting parameters*

The customize fitting parameters function controls default settings of input parameters for the various signal processing functions, and also their maximum values. See Figure 6.



	DEFAULT VALUES				MAXIMUM VALUES		
	ndp	ncolumn	npeak	nit	ndp	npeak	nit
VARPRO/AMARES	4096				4096	25	50
HLSVD	4096	513	12	120	4096	50	1000
HSVD	512	384	12		1024	50	
MATLAB HSVD	512	384	12		2048	100	
HTLS	512	384	12		1024	50	
EP	256	57	11	5	4096	512	20
LPSVD	256	57	11		4096	512	
LPSVD(CR)	256	57			2048	512	

Figure 6. The customize fitting parameters window lets you 'streamline' your functions.

Here you can change the default input parameter settings for the methods you use, in order to reduce the operator interaction with these functions to almost zero if your applications are very similar in nature. You can also change the default settings for various options via the corresponding menus. It is even worthwhile to use this customize function to change the default input parameters for each new type of experiment you are analyzing, so that you can virtually process all signals from that particular type of experiment simply by loading the signal and selecting the go! menu item (see e.g. sections 8.2.5, 10.7 and 13.4). This certainly helps toward full automation of routine MRS data analysis. You should change the maximum values of the input parameters here *only* if you changed their constants in the corresponding Fortran programs (see Appendix E). If you changed the constants in the Fortran programs, it suffices to change the maximum values here in order to make MRUI incorporate the new settings in the Fortran algorithms, and it will then automatically update the relevant functions, such as e.g. the size of the peak number submenu in the Peakpick function (see section 11.5). See the corresponding sections for more detailed information on the various input parameters and menu options.

3.2.1.5 customize absolute quantitation

The customize absolute quantitation function is disabled as it is still under development.

3.2.2 directories

The directories menu lets you browse through, change to, and create directories. If your version of Matlab allows it, it will use a familiar file manager window. For older versions of Matlab, MRUI will display a get name window in which you have to type the relevant directory name.

3.2.2.1 current dir?

Selecting this menu item will display the full path name of the current directory in the Matlab command window. This is equivalent to (but with cleaner output than) to the Matlab pwd command.

3.2.2.2 cd work_dir

This will change the current Matlab directory to your `$HOME/matlab/work_dir` directory and display its full path name in the Matlab command window.

3.2.2.3 select dir

This menu item is disabled (inactive) when running MRUI on a PC. This is because it uses Matlab properties of the file manager function that are only valid under UNIX and not under DOS. When running MRUI on a UNIX platform, you can use this function to bring up the file manager window in which you can browse through your directories by double-clicking on them. Once your directory of choice is the current directory in this window, and its contents displayed on the right, press the Done button, and Matlab will change directory to it (make it the current directory), and display its full path name in the Matlab command window. If you have Matlab version 4.0, you will see a get name popup window, in which you must type the *full* path name of the directory you want to change to.

3.2.2.4 create dir

This menu item is disabled (inactive) when running MRUI on a PC. This is because it uses Matlab properties of the file manager function that are only valid under UNIX and not under DOS. When running MRUI on a UNIX platform, this menu item allows you to create a new directory in your existing directory structure. Browse through your directories (by double-clicking on the directory names) until you reach the directory under which you wish to create the new one, type the name for your new directory in the appropriate text box and click on the Done button.

3.2.3 list files

This menu item displays a list of files in the current directory. If you have Matlab version 4.1 or higher, this will be with the help of the familiar file manager window, in which you can even browse through an arbitrary number of directories (*without* Matlab actually changing the current directory) and see lists of files there, complete with scroll bars. For more information about the use of the file manager, see section 5.1.1.1.1. If you have Matlab version 4.0, the list of files will be that produced by the Matlab `dir` command.

3.2.4 workspace

This menu item displays all variables currently active in Matlab workspace (memory). If you use MRUI as an MRS data analysis tool, it is unlikely that you will benefit from this function. It is meant to be useful for regular Matlab programmers that wish to manipulate certain variables, or understand how things are working. You can choose between a short variable list (equivalent of Matlab `who` command) and a detailed list which also lists the properties (size, density, complexity, etc.) of each variable (equivalent of Matlab `whos` command).

3.2.5 save

Again, this function is included for the use of software development or independent manipulation of variables and data. This menu item saves all variables in the Matlab workspace (see section 3.2.4) to a binary Matlab file, of which you can enter the name (equivalent to Matlab `save <name>` command).

3.2.6 pack

Pack clears and re-organizes Matlab's memory usage. If you tend to get "memory full" messages or unexpected behavior (such as error messages which cannot be reproduced after a complete re-start of Matlab), you may find it useful to select this menu item regularly to avoid Matlab memory errors. This menu item is nothing more than a callback to the Matlab `pack` command. `pack` consolidates workspace memory; it performs memory garbage collection. Extended Matlab sessions may cause memory to become fragmented, preventing large variables from being stored. `pack` is a command that saves all variables on disk, clears the memory, and then reloads the variables.

3.2.7 clear screen

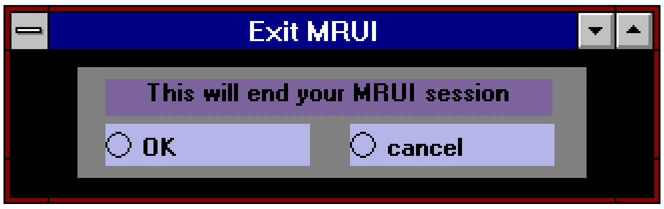
Selecting this menu item clears the Matlab command window (equivalent to the Matlab `clc` command).

3.2.8 restart

This function quits MRUI, clears all variables and windows, re-organizes memory (using the Matlab `pack` command, see section 3.2.6) and automatically restarts MRUI. You may find this useful when you find after an error message that other functions no longer work, or that you get a cascade of error messages. In many cases you can use this function without having to quit and restart Matlab, which is much more cumbersome. Also, when issuing a restart, Matlab keeps the current directory, whereas after a Matlab restart you will return to the directory from which you called the Matlab program (see section 3.1).

3.2.9 quit MRUI

This menu item quits MRUI, clear all variables and windows, and leaves Matlab running, with a cleared command window at your disposal. It brings up a final window in which you must confirm your exit action. See Figure 7.

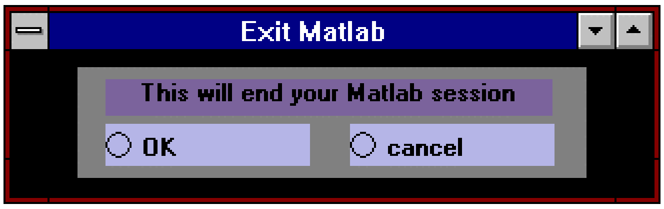

Exit MRUI

This will end your MRUI session

Figure 7. Exit window to confirm 'exit MRUI'.

3.2.10 quit Matlab

This menu item terminates Matlab (and consequently MRUI) in one go. Equivalent to the Matlab `quit` command. You can use this to end your session (once you have finished running MRUI and you no longer need Matlab it is recommend to exit Matlab as this clears a large portion of RAM memory), or as a last resort in case you had errors that even prohibited the MRUI restart. It brings up a final window in which you must confirm your exit action. See Figure 8.


Exit Matlab

This will end your Matlab session

Figure 8. Exit window to confirm 'exit Matlab'.

3.3 Strategy for MRS data analysis with MRUI

So how exactly do you approach this software package? Which steps do you take when you wish to get numbers out of your experimental data? What do you pay attention to when selecting on of the available methods? This section will hand you a short instruction list of steps to take from having an MRS signal to getting the table of numerical results. The methods named in the various steps will be described in detail in the following chapters. Right now we stress that this manual aims at explaining the MRUI software package, and the *use of* the signal processing methods contained in it, such as HLSVD and VARPRO. This manual is *not* the right place to learn about the theory of time domain fitting and the algorithms used, see also section 1.1. However, at times you may find it useful to understand more of the underlying theory of the methods you use, especially if you need to research a special strategy when the routine instructions in this manual do not suffice. This manual will give as much information and tips as possible, but it can be very rewarding to read some of the cited references, or browse through Dr. Ron de Beer's lecture notes on the internet (see page 1).

But now, in short, the hitch-hiker's guide to MRS data analysis with MRUI. You will find a fully worked out example of this step-by-step list in the `work_dir/TEST` directory. Using the provided files, you can perform this entire sequence of signal processing steps on your own computer, which is really not a bad way to get to know your MRUI.

step 1	First of all, get yourself a signal. In most cases, you will have a measured MRS signal in a spectrometer-format file, which must be changed to a format readable by the MRUI programs, so start with Conversion.	see chapter 4
	Otherwise, you can create yourself an FID, using the MRUI Simulation tool	see chapter 6
step 2	Make sure the spectral display is according to your expectations (and your spectrometer software display). Use Database - SETUP: experimental.	see chapter 7
step 3	If you wish to apply any preprocessing to your signal before you carry out the numerical parameter estimation, use FID Maths.	see chapter 8
step 4	To remove unwanted components from your signal (such as residual water, lipid peaks or solvent peaks), use Filter_HSVD before parameter estimation.	see chapter 9
step 5	If your signal is well-conditioned (high SNR, low to moderate overlap), 'black-box' parameter estimation using an SVD-based method is fast and easy. SVD_1D will already complete your numerical data analysis.	see chapter 10
step 6	However, with *in vivo* MRS data, the SNR is usually low and an SVD-based method will not yield most accurate results. For most accurate parameter estimation you should use a non-linear least squares fitting technique that can implement biochemical prior knowledge: VARPRO or its new improved variant AMARES. Here you must first supply a set of starting values for the nonlinear least squares routine. You can do this using Peakpick.	see chapter 11
step 7	Next you can instruct the VARPRO or AMARES program with a number of optional settings (such as the lineshape to fit with), and impose constraints known from biochemical prior knowledge (e.g. multiplet structures) using, depending on the type of prior knowledge required, and whether you are going to run the conventional VARPRO method or the new improved AMARES program, the Inputvp or Inputnv function, respectively.	see chapters 12,14
step 8	Once you have provided starting values and prior knowledge, you can run VARPRO / AMARES, which does not need any further input.	see chapter 13

This list is illustrated below in a diagram. Solid-bordered boxes are necessary processing functions (or the most likely of two alternatives); dashed-bordered boxed optional (or the less likely of two alternatives).

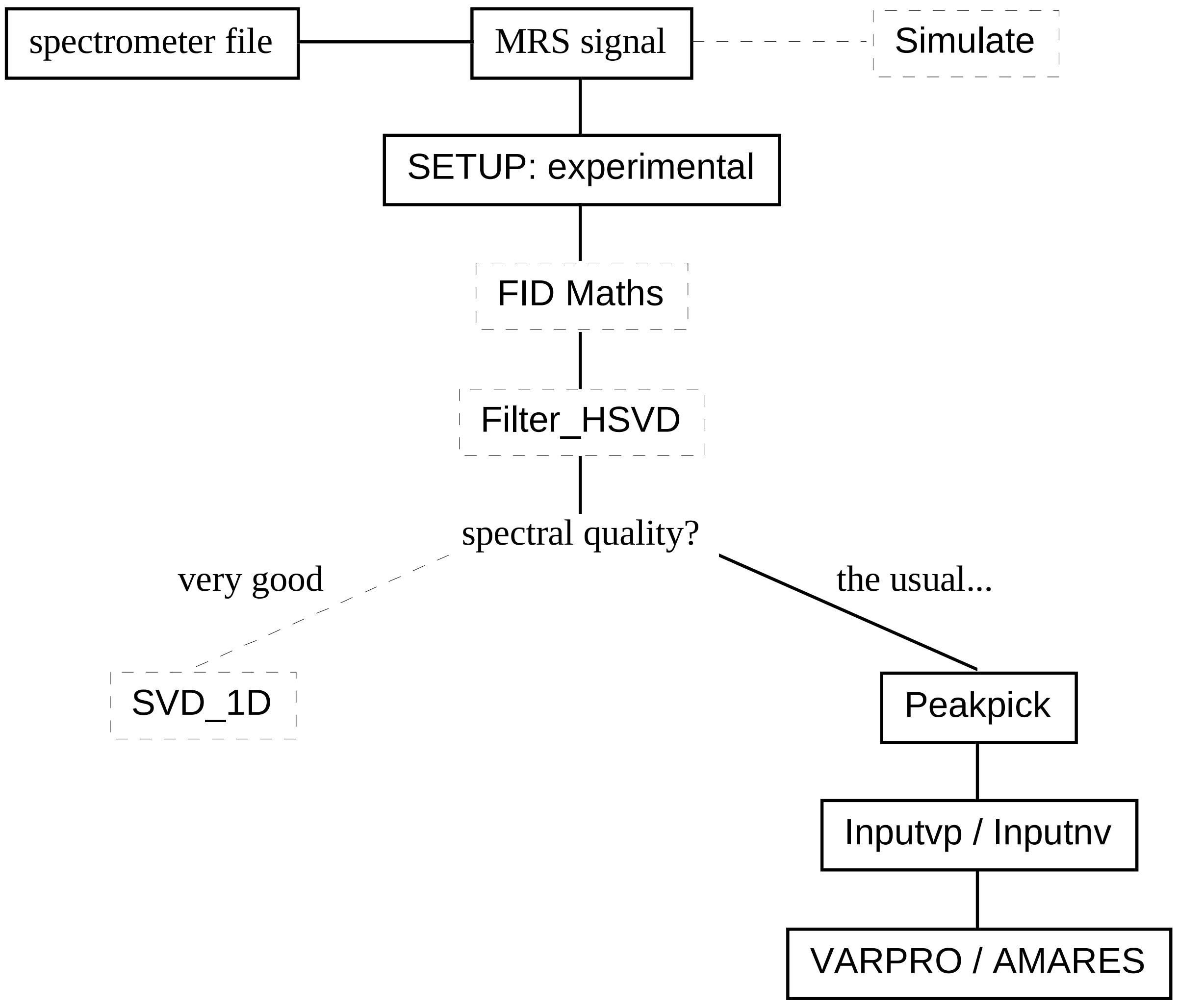


Figure 9. The hitch-hiker's guide to the MRUI functions.

You are now ready to embark on your journey through the MRUI signal processing functions, but first we will describe the Conversion menu with its features in chapter 4, and illustrate the spectral windows to you and explain their general functions in chapter 5.

Throughout the main course of this manual, we will focus on one particular example of a measured MRS signal, and follow it through the various relevant signal processing procedures. For signal processing functions that are not relevant to the quantitation of this *in vivo* ^{31}P MRS signal, other signals will be used for illustrative purposes. The example signal, named `exam`, is an *in vivo* ^{31}P MRS signal, measured from a human calf muscle on a Picker modified (SMIS, Guildford, UK) 1.5 T NMR spectrometer (the SMIS format data file `p31ischb.mrd` must first be converted into `ischr.dat` and `ischr.mat`, see section 4.6). The human subject, a healthy volunteer, was placed in a supine position, and a 5 cm radio-frequency surface coil positioned under the right calf muscle. ^{31}P MR spectra were collected continuously with a temporal resolution of 96 s during three ischaemia & reperfusion cycles of 5 & 5 minutes each[1]. We are grateful to Tiziano Binzoni, E. Hiltbrand and P. Cerretelli from the departments of Physiology and Radiology of the University of Geneva (Switzerland) and the Centro Interuniversitario Grandi Apparecchiature Biomediche nelle Neuroscienze (CIGABIN) at the University of Padova (Italy), for providing this example data set, and allowing its use for demonstration purposes in this manual, to which is has an important contribution in illustrating the various functions of the MRUI software.

4. Spectrometer file conversion

MRUI is equipped with a number of conversion routines to read files from different spectrometers. The purpose of the conversion is to create data files in such a format that they can be read by the Fortran programs that run the signal processing, and by the Matlab scripts that run the MRUI interface. Also, all available and relevant experimental parameters read from the spectrometer file will be stored in a small ASCII file, so that they can be used for MRUI's spectral display. It is stressed that the available conversion routines are *not* provided centrally from the MRUI development, but originate from MRUI users with spectrometer files of a 'new' format (i.e. not yet included in the MRUI program), who provide us with the necessary manufacturer information to read those files (upon which we will write the conversion script) and sometimes even a ready-to-run Matlab conversion script (and then we only have to incorporate it into MRUI using the general file management and spectral display format). So these conversion menus are constantly updated according to the requirements of the MRUI users. Should you have written your own conversion tools for a particular type of spectrometer file, please share it with the rest of the MRUI community!

4.1 The two MRUI formats

Whenever you convert a spectrometer data file, you will see that for *each* signal, *two* binary data files will be produced: the `<name>.dat` file in binary rdb format, and the `<name>.mat` file in Matlab format. This may sound like a waste of disk space, but it speeds up file reading into the graphical interface. The origin of this duality lies in the history of signal processing before the MRUI interface. Initially, HLSVD and VARPRO existed only as Fortran 77 programs. Data input was performed using Fortran routines, and for maximum speed, data files were read in a binary format. This format, initially introduced by Dr. Ron de Beer, was dubbed the "rdb format", of which an interchangeable binary and ASCII version existed. Next, with the introduction of the MRUI interface, the signals from the data files also had to be read in and written from the Matlab environment. The obvious format for these data files was of course the Matlab binary format, though this meant the necessity for a second copy of the data file in the Matlab format. Could both types of program not use the same binary file? Well, if we would stick to the binary rdb format, it would be possible to read in those files into Matlab, and fill the necessary variables with the right values. However, this would much slower than Matlab's file I/O for files in its own format, and require more run-time memory. On the other hand, can the Fortran programs not be taught to read the Matlab binary files? The reading time difference would be minimal. It is indeed possible to write a simple Fortran subroutine that reads or writes a Matlab format binary file, but this would require system-specific Matlab-provided file I/O libraries when compiling the Fortran programs. This might create a compatibility problem for the distribution of MRUI over a large number of groups with different computer systems and Matlab versions, and would require parallel copies of all Fortran programs for MRUI users with Matlab, and users of the Fortran programs that do not have Matlab. This situation would be difficult to maintain over a number of software upgrades and is therefore undesired. Finally, would it not be possible to write a complete Fortran routine for reading and writing of Matlab format binary files, without the need for the Matlab-provided libraries? Well, all Matlab owners can check their "Matlab external interface guide", of which the appendix "MAT-File Structure" lists the required information. I have tried to write a Fortran subroutine that would achieve this, but I did not get far and suffered from time pressure of other tasks. It is still on the list of things-to-do, but I cannot judge when we will get it done. Should anyone succeed in writing an appropriate Fortran routine, then we would be most happy to incorporate it in the MRUI programs, thus avoiding the necessity of dual data files.

However, for the moment MRUI will keep working with the double copies of each data file, the `<name>.dat` file for reading into the Fortran programs, and the `<name>.mat` file for reading into the MRUI interface. Any MRUI function that creates new files, will automatically produce both types. In the rest of this chapter on conversion, when we refer to one type, e.g. `<name>.dat` being produced by conversion of SMIS MRD files (see section), it is implicitly meant that the corresponding `<name>.mat` file is produced as well, and vice versa. If for some reason only one of the two is being found at any time, the corresponding file will be automatically created. This means that after you have analyzed a set of data, you can remove all the `*.dat` files, or all the `*.mat` files, so you can clear some disk space, without losing the data. You can also use the Conversion menu to convert between the two file formats, should you want to do this manually, see section ***.

The following two sections describe the rdb and Matlab format files, which store the number of data points (`ndp`), the dwell time (`step`), delay time (`begin`) and of course the data points themselves. The number of data points of the signal written to the MRUI data file is always taken as the number of data points present in the signal in the spectrometer data file, with a restriction that `ndp` in the MRUI data file cannot be larger than the absolute maximum number of data points allowed in the MRUI software. This maximum value protects the Fortran read and write routines in the signal processing and conversion programs from overflowing the data arrays, which would incur uncontrolled loss of data points and lead to aberrant results. The default maximum in the MRUI package is set at 16384, but you can change this value in the maxndp submenu item of the SETUP menu in the customize fitting parameters window. However, you should do this *after* you have increased the maximum array sizes in the corresponding Fortran routines, see Appendix E. In the example, the data files are called `test.asc`, `test.dat` and `test.mat`, respectively.

4.1.1 The rdb file format (`*.dat`)

The rdb format files are characterized by a single precision buffer of 64 elements (ASCII format: 16 rows, 4 columns), followed by a the real parts of the complex data points (`ndp`/4 rows, 4 columns), in turn followed by the imaginary parts of the complex data points (`ndp`/4 rows, 4 columns). According to the Fortran protocol, the rdb format ASCII file is written as:

```
      open(100,file='test.asc',status='unknown')
      write(100,10) (buf(i),i=1,64)
   10 format(4e15.5)
      write(100,20) (yr(i),i=1,buf(1)/2)
      write(100,20) (yi(i),i=1,buf(1)/2)
   20 format(4e18.10)
      close(100)
```

The parameters contained in the buffer are as follows:

`buf(1) = 2*ndp`	number of complex data points times two
`buf(2) = step`	sample time step of FID in milliseconds
`buf(3) = begin`	begin (delay) time of FID in milliseconds

The VNMR format conversion program `bsis-vp` (see section ***) also writes (although these are not automatically used) the following variables:

`buf(4) = phzero`	zero order phase of FID in degrees
`buf(5) = lb*pi/1000`	line broadening times π in kilohertz

Using the same variable names, the rdb format binary file is written as:

```
      open(100,file='test.dat',form='unformatted',status='unknown')
      write(100) (buf(i),i=1,64)
      write(100) (yr(i),i=1,ndp)
      write(100) (yi(i),i=1,ndp)
      close(100)
```

4.1.2 The Matlab file format (`*.mat`)

The Matlab format data files contain the same parameters as the rdb format files, written using the Matlab `save` command:

```
save test ndp step begin signal
```

All variables can be read back into the Matlab workspace (memory) using the Matlab `load` command:

```
load test
```

In the Fortran programs it is not essential to use these names for the variables. However, in the Matlab file writing and reading, you should adhere to the variable names as above, as subsequent scripts expect variables with such names to exist in memory. Also, since the Matlab files need to be converted to the rdb binary files, the order in which the variables are stored in the Matlab files is also important and should not be changed (e.g. if `step` and `begin` are interchanged, the Fortran programs will later read the value of `begin` into the `step` variable, and vice versa).

4.1.3 Converted binary data files: names and locations

Whenever you convert a spectrometer file using one of the MRUI Conversion menus, you will be asked what the name (or base name) of the binary data file will be. These file names in MRUI may contain up to 32 valid characters, including the extension (such as `.dat` or `.mat`). PC users with Windows 3.1 or 3.11 (or with Win-

dows 95 using Matlab for Windows 3.1 or 3.11; the true Windows 95-compatible Matlab version is expected soon), should of course comply to the DOS convention that file names can have a maximum of 12 characters: a file name base of maximally 8 characters, then a separation dot and an extension of maximally three characters. If you are asked to enter the base name of a series ('batch') of files, make sure that the base name does not end in a numerical character, otherwise it may upset the numbering of the series as maintained by MRUI. Do not use dots in the file names other than the one to separate the file name base from its `dat` or `mat` extension. Also, do not use underscore _ characters near (within 4 positions of) the end of your file names, as these are preserved for data files that have undergone some sort of FID processing (see chapter 8). For trouble-free use of the Plot batch results function (see section 16.3), file names *must* start with a non-numerical character.
You can also specify the directory in which the converted files need to be stored, either by browsing through the directories in the File manager (Matlab versions 4.1 or higher) or by typing the full path name of the file (Matlab 4.0). For an example of this interaction for batch processing, see section 4.4.1, for an example of this interaction when converting single spectrometer files, see section 4.6. In sections 2.3.4 and 2.4.3 you read how your personal `$HOME/matlab/work_dir` is the central point of your MRUI file system, and it is recommended, although not crucial, that you create subdirectories (see e.g. section 3.2.2.4) for your data files under this `work_dir` directory. You can have your data directories on any arbitrary position in your directory structure, but in some situations it has proven a little easier if the data files to be processed by MRUI are in subdirectories under your `work_dir` directory. It is however *not* advised to store data files directly *in* the `work_dir` directory, as that will result in cluttering of the directory with data and result files; it is better to leave the `work_dir` directory for the dedicated MRUI files as described in sections 2.3.4, 2.4.3 and 3.2.1.

4.2 FID scaling

When you convert a spectrometer file, the FID in it may have a very large intensity. When gain factors in your spectrometer are high, and no automatic correction for the number of summed blocks takes place, FID data points may be of the order of one million to one billion (arbitrary units). This means that when doing the parameter estimation with MRUI, the estimated amplitudes have values of a similar order of magnitude. This is undesired, as some parameter estimation algorithms may not perform optimally for very large numbers, and besides, it will cause tables with results to misalign. Therefore, all conversion routines contain a built-in function to normalize the converted FID to norm 1000. This means that the maximum (real or imaginary) value in the FID is scaled to be 1000, and all other data points are scaled analogously. This properly organizes the tables in MRUI, and optimizes performance of VARPRO. However, the scaling factors need of course be taken into account later on, if absolute quantitation is to be performed. Therefore, for each converted FID, apart from the `<name>.dat` and `<name>.mat` file, a third file is created: the `<name>.scl` file, which contains the relevant parameters:

FID scaling parameter		package default
`sfrq`	transmitter frequency in Hz	63860000
`scans`	number of FID blocks summed	N/A
`scale`	true maximum (real or imag) value of raw FID	N/A
`norm`	maximum value for normalization	1000

Whenever necessary (combining two signals into one, calculate absolute concentrations, etc.), MRUI will automatically correct the (amplitudes of) the scaled signals in the data file by reading the `scale` and `norm` values from the `<name>.scl` file, and multiply the (amplitudes of) the scaled signals by the `scale` value and divide it by the `norm` value. This will make the whole scaling feature rather transparent, as the final results presented to you will still be in terms of the original intensities of the FIDs. One particular point of interest is the values of the estimated amplitude values that are presented in the parameter result table in the result spectral window after an SVD_1D or VARPRO fit (see e.g. Figure 25 in section 5.2). By default, the presented values in this column are those estimated by SVD_1D or VARPRO from the time domain signal in the data file, i.e. the scaled signal, so the amplitude values will be in the order of 10 to 1000 (because the maximum intensity of the entire FID was set to 1000). However, if you wish to have the original intensities (i.e. the amplitude values corrected by the `scale` and `norm` factors) displayed in this table, you can change that default setting using the amplitudes in result table menu item under the SVD_1D and VARPRO menus of the customize fitting parameters menu from the Matlab menu under the MRUI base window (see section 3.2.1.4).

4.3 Storage of experimental parameters

At conversion time, not only the variables described in sections 4.1.1 and 4.1.2 are read from the spectrometer file; all other relevant experimental parameters available are read and stored in an ASCII file called `default.xpt`. This file is consulted by MRUI whenever it needs some experimental parameter for the spectral display or absolute quantitation. The experimental parameters stored in `default.xpt` are:

experimental parameter		package default
`nucleus`	measured nucleus	'h'
`b0`	static magnetic field in Tesla	1.5
`sfrq`	transmitter frequency in Hz	63860000
`xun`	x-axis units	'ppm'
`khzref`	reference resonance in kHz	0
`ppmref`	reference resonance in ppm	4.7
`ndpfit`	number of complex data points to be used in fit	`ndp`
`ndft`	number of data points for FT spectrum	4096
`fresign`	indicator for correct display of spectrum	1

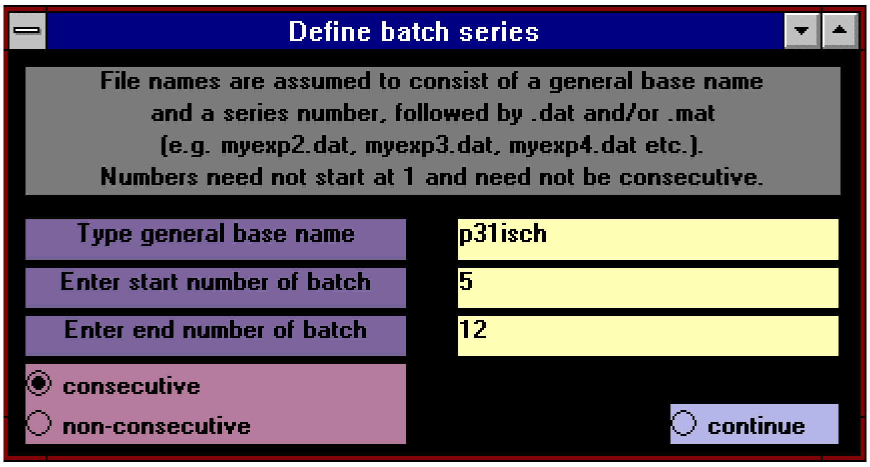
Each conversion routine tries to read as many of the above parameters from the spectrometer file as possible, and for the ones it could not obtain from the spectrometer file, it will take a set of default values. When starting conversion of data files in a new, empty directory, these default values come from the `mrui.ini` file in the `work_dir` directory. The default values distributed with the MRUI package are listed above (corresponding to an *in vivo* ^{1}H MRS application at 1.5 T), but these can be set at any different value using the Customize fitting parameters function, see also section 3.2.1.4. However, if a `default.xpt` already exists in a directory (e.g. from an earlier conversion), then the conversion routine will take the values for the parameters it could not read from the spectrometer file, from that `default.xpt` file. This means that once you have done one conversion (and created the first `default.xpt` in that directory), you can optimize the settings using the Database menu, item SETUP: experimental (see chapter 7), and these new values will then keep existing in the `default.xpt` file in that directory, even though it gets overwritten at each new conversion. Thus the parameters that are read from the data file can be different after each conversion (which is desired for e.g. the exact offset transmitter frequency), whereas parameters not obtainable from the data file always take the values as set by yourself using the SETUP: experimental function. So for each directory with data files, one `default.xpt` file will exist, containing experimental parameters that customize your spectral display. This fact makes it recommendable to store data files from different experiments in different directories, which is a good idea anyway.

4.4 Conversion Options

The first menu item under the Conversion menu shows two related options: Batch conversion and display results. Selecting either of these options simply toggles their setting between off and on.

4.4.1 batch conversion

Any conversion routine can be run in batches. This means that you can indicate a series of spectrometer files, such that they will be all be automatically converted to their binary rdb and Matlab format files. By default, this option is switched off. By selecting the menu item you can toggle this setting. Once the batch conversion item is switched on, any conversion routine run will start by displaying the Define batch series window (see Figure 10), in which you should type the general base name of the files in the batch, the start serial number of the batch and the end serial number of the batch. The start serial number of the batch does not have to be 1.


Define batch series

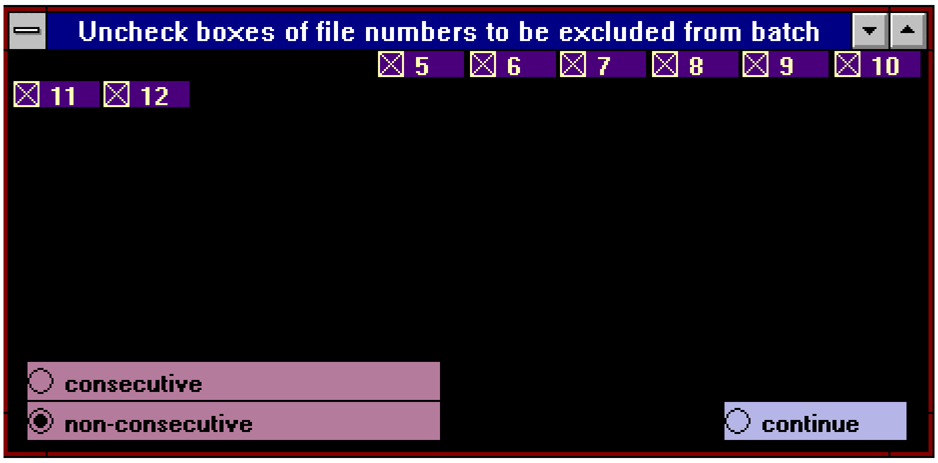
File names are assumed to consist of a general base name and a series number, followed by .dat and/or .mat (e.g. myexp2.dat, myexp3.dat, myexp4.dat etc.). Numbers need not start at 1 and need not be consecutive.

Type general base name	p31isch
Enter start number of batch	5
Enter end number of batch	12

[✓ consecutive] non-consecutive

Figure 10. The batch series definition window

The continue button in the lower right corner of the window in Figure 10 originally comes up with the text cancel. Using the cancel button you can at any moment during your batch series definition bail out and start again. Typing the required information goes in the order from above to below: first the question for the general base name comes up, after you have typed the base name and pressed `[Enter]`, the question for the start number comes up, after you have typed that and pressed `[Enter]`, the questions for the end number comes up, and after you have typed that and pressed `[Enter]`, the lower right button will change from cancel to continue, and two more buttons will appear in the lower left corner of the window, see Figure 10. Note that before you can type in one of the dedicated edit boxes (for base name, start or end number), you must first click with mouse (left mouse button) *inside* the edit box. For further information on loading batches, see section 5.1.1.1.2.
Should you have a consecutive batch, i.e. a data file exists for each serial number between the start and end number, and you wish to include all these files in the batch, then you can ignore these two buttons, and click the continue button. However, if there are files missing for certain serial numbers between the start and end number (if you do not take the appropriate action here, there is always a protection where the existence of each file to be read is checked, and the correct action taken if the file does not exist), or in case you wish to exclude certain files from the batch on purpose, you can click the non-consecutive button, upon which the batch exclusion window will appear (see Figure 11).


Uncheck boxes of file numbers to be excluded from batch

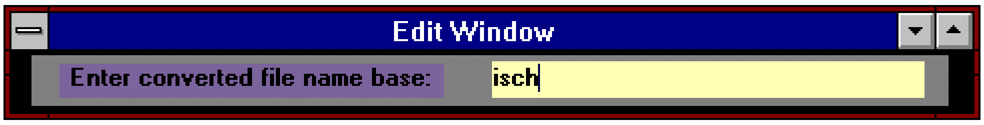
☒ 5 ☒ 6 ☒ 7 ☒ 8 ☒ 9 ☒ 10 ☒ 11 ☒ 12

consecutive [✓ non-consecutive]

Figure 11. Excluding certain file numbers from the batch.

In this window you will see a checkbox for each number in the batch between the start and end number. Initially, all check boxes are checked, which means that by default all files will be included in the batch. By clicking on a checkbox, it will be unchecked, and the corresponding file number excluded from the batch. You can reverse this action by clicking the same checkbox again, upon which it will be checked again, etc. Once you have composed your batch according to your wishes, press the continue button. To make sure all files are included again in one action, press the consecutive button. You could then obtain the checkboxes again by clicking on the non-consecutive button once more, etc. This window can contain a maximum of 100 checkboxes. Should you have a batch larger than that, a supplementary popup menu will appear in the bottom right

corner of the window, which allows you to browse through the fields containing numbers 1..100, 101..200, 201..300, etc. In principle, this allows you to work on batches that have an arbitrarily large number of files.
Once the batch has been defined, you are asked to enter the converted file name base, see Figure 12.


Edit Window

Enter converted file name base: isch

Figure 12. Edit window to type file name base for batch after conversion.

This is all standard for all conversion routines when run in batch conversion. When converting single spectrometer files, the file name interaction is as illustrated in section 4.6, identical for all conversion routines (apart from spectrometer file names and extensions etc.). The specific details of the different spectrometer format conversion tools are described in sections 4.5 and further.

4.4.2 display results

This option determines whether the converted FID and spectrum will be displayed. By default, when batch conversion is off, the display results option is on. When you switch on the batch conversion option, the display results option will be automatically switched off. Should you wish to display the FIDs and spectra of all files in a batch, just switch display results back on. You can of course also disable the display of results for single file conversions (batch conversion off) by switching off display results.

4.5 GE (General Electrics) data

If you have data files acquired on a General Electrics (GE) system, use the appropriate item under this menu to convert them into the MRUI formats. Conversion routines exist for 5x and 4x files, PROBE and non-PROBE files, so-called G files, SAGE/IDL files and Omega files.

4.6 SMIS (Surrey Medical Imaging Systems) data

If you have data files acquired on an SMIS console, use this menu item to convert them into the MRUI formats. As soon as you select the SMIS data menu item, the file manager will come up (if you have Matlab 4.1 or later), from which you can select the SMIS MRD file to be converted (browse through the directory tree if necessary). All files with an `mrd` extension (`*.mrd`) are listed. For conversion of other spectrometer files, the file manager will list all the files with the extension relevant for that type spectrometer. If you have Matlab 4.0, you must type the file name in the get_name window (for an example, see Figure 12); the file name suffices if it resides in your current directory, otherwise type the *full path* of the spectrometer file name. For an example of the interaction with the file manager† in order to indicate the spectrometer file to be converted, see Figure 13.

† During any MRUI session, you'll be using Matlab's file manager (if you have Matlab version 4.2 or higher) rather a lot. The first time in any session it will come up with its default size. Resize it to a convenient size once, and from then on (in that Matlab session) it will have your preferred size (and come up faster, too!).

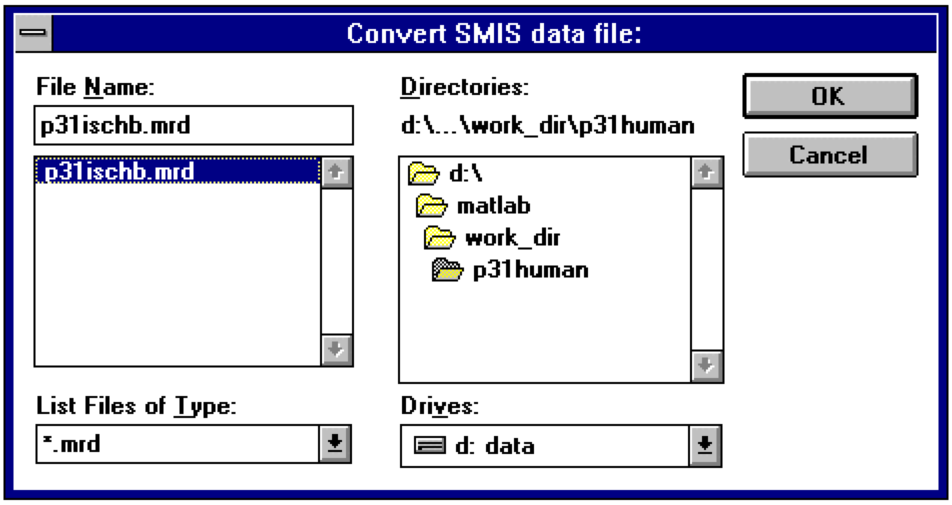


Figure 13. The file manager used to indicate the SMIS spectrometer (MRD) file.

Browse through the directories in the Directories: field, then select the spectrometer file and click on OK (Done on UNIX systems) or press `[Enter]`. You can also double-click on the spectrometer file name. Cancel will abort the file conversion routine. For more information on the file manager, see section 5.1.1.1.1. Next, the following file manager will come up, asking you under which file name to save the converted spectrometer file. See Figure 14.

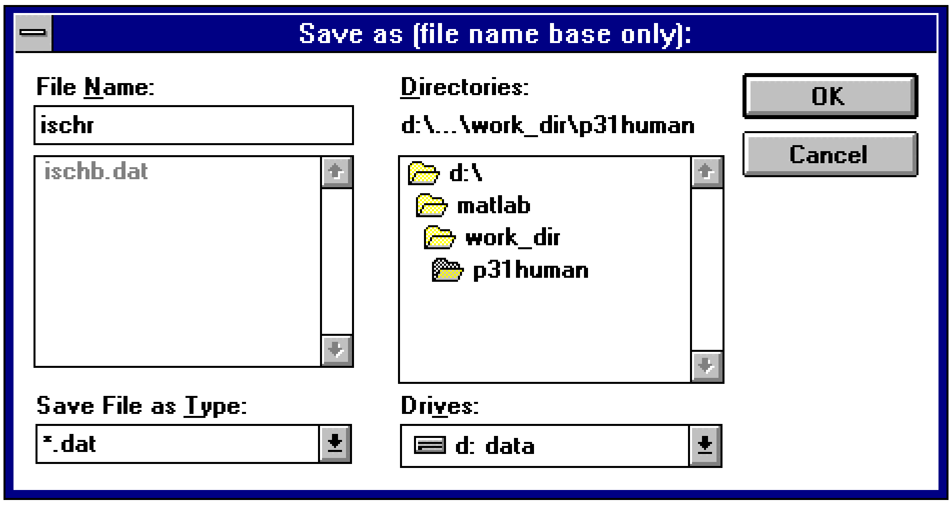
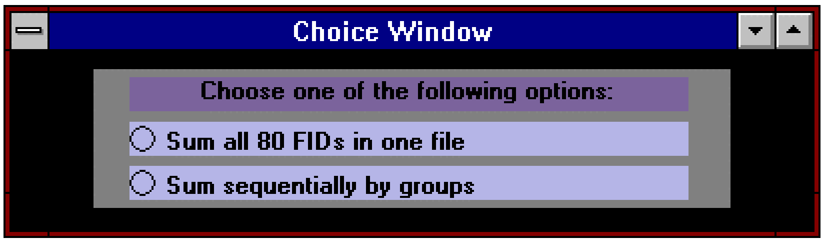


Figure 14. The file manager used to indicate the converted MRUI file name base.

Again, you can browse through the directory tree, and then type the file name base in the File Name field (selection box on UNIX systems), or select an existing `<name>.dat` file if you want to overwrite it. Should the entered or selected file name contain a `.dat` extension (the window title asks you for the file name base only), then MRUI will detect and omit it automatically, so that the conversion program gets the correct file name base to work with. Click on OK (Done on UNIX systems) or press `[Enter]`. Cancel will abort the file conversion routine.
For SMIS files, you will now be asked how the individual blocks of FIDs in the spectrometer file (if more than one were stored in the MRD file) should be stored in the MRUI format files. The following window gives you two choices, see Figure 15.



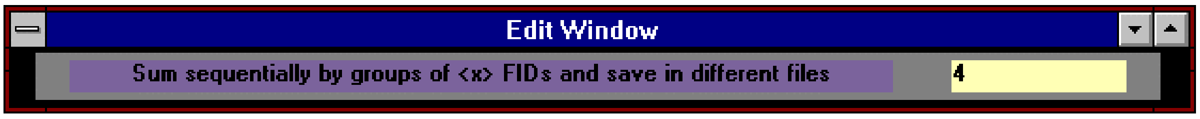
Choose one of the following options:

Sum all 80 FIDs in one file

Sum sequentially by groups

Figure 15. The choice window for storage of blocks of FIDs in SMIS conversion.

If you click on the first radiobutton, the conversion routine will sum all FIDs in the MRD file, and write the resulting signal to `<name>.dat`. You can read from this first radiobutton how many blocks were stored in the MRD file. If you wish to sum sequentially by groups, you are asked how many blocks should be summed together each time (enter 1 to store all FIDs in the MRD file separately), see Figure 16.



Sum sequentially by groups of <x> FIDs and save in different files

4

Figure 16. Edit window to type the number of blocks added together in SMIS conversion.

The SMIS conversion routine will then sum the first four blocks and store the resulting signal in `<name>1.dat`, the sum of blocks five through eight in `<name>2.dat`, etc.
Finally, the conversion routine (this is standard for all conversion routines) displays a new window in which the raw FID(s) and spectrum(a) are plotted. See Figure 17.

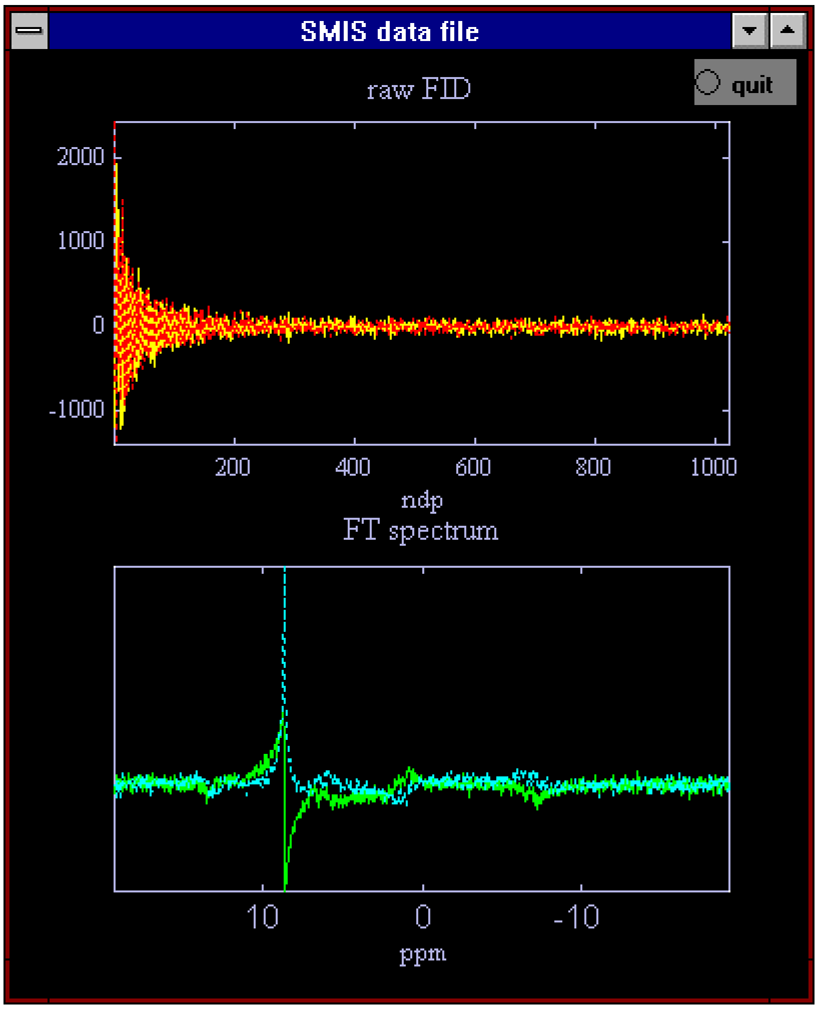


Figure 17. The conversion plot window, displaying the raw FID(s) above (real part in yellow, imaginary part in red) and the raw spectrum(a) below (real part in green, imaginary part in cyan). When multiple FIDs are generated, these will be displayed one by one.

Click on the quit button and the window will disappear. The conversion is finished. Certain spectrometer conversion routines display more than one FID/spectrum plot window. Clicking on a single quit button will suffice to clear all windows.

The experimental parameters (see section 4.3) that can be read from the SMIS spectrometer files are `nucleus` and `sfrq`. The other experimental parameters are defaulted as follows (`ndpfit` and `ndft` are always taken as in the table in section 4.3):

`b0`	1.5
`xun`	'ppm'
`khzref`	0.0
`ppmref`	0.0
`fresign`	1

The scaling parameters (see section 4.2) that can be read from the SMIS spectrometer files are `scans` and `sfrq`. The value of `scans` is taken as the number of FIDs in case they are all added together. In case you sum sequentially in groups of FIDs, the number of FIDs per group is written to the scale file of the corresponding data file.

5. MRUI: Spectral Windows

All spectral analysis functions in MRUI have been organized in three main windows: the spectral window in single spectrum mode, the spectral window in multi spectrum mode and the FID processing window. The FID processing window has been especially developed to display FID processing techniques that require two FIDs (such as summing, subtracting or dividing FIDs). The main spectral window displays the spectrum that is to be processed, and after a fit routine has been carried out, the window will increase in size to display the original spectrum, the Fourier transform spectrum of the reconstructed FID according to the estimated model function parameters, the residual spectrum and the individual peaks (Fourier transforms of the individual complex exponentially damped sinusoids). These windows contain a number of menus, buttons and sliders that are identical within all spectral processing functions, plus a separate menu that is specific to the analysis function activated. The general menus and controls will be described for the three main windows in this chapter. The function-specific menus will be explained in the chapters that describe that particular function. The default sizes and positions of these windows can be customized according to the user's preferences using the customize windows menu from the Matlab menu under the MRUI base window (see section 3.2.1.1).

5.1 The basic spectral window (small)

An example of the basic spectral window is displayed in Figure 18. In this example, the basic spectral window is used for the SVD_1D function from the quantitation menu under the MRUI base window. As can be seen from the figure, one menu is headed SVD; this menu contains the features specific to SVD_1D only (for more details see chapter 10). The other menus recur in all or most spectral processing functions that use this spectral window, and are discussed below. The Results menu only becomes enabled (active) after the signal processing function has finished, and will be discussed in conjunction with the result spectral window in section 5.2.1. Although the Next menu is enabled (active) as soon as a data file is loaded, it will also be discussed in conjunction with the result spectral window in section 5.2.2.

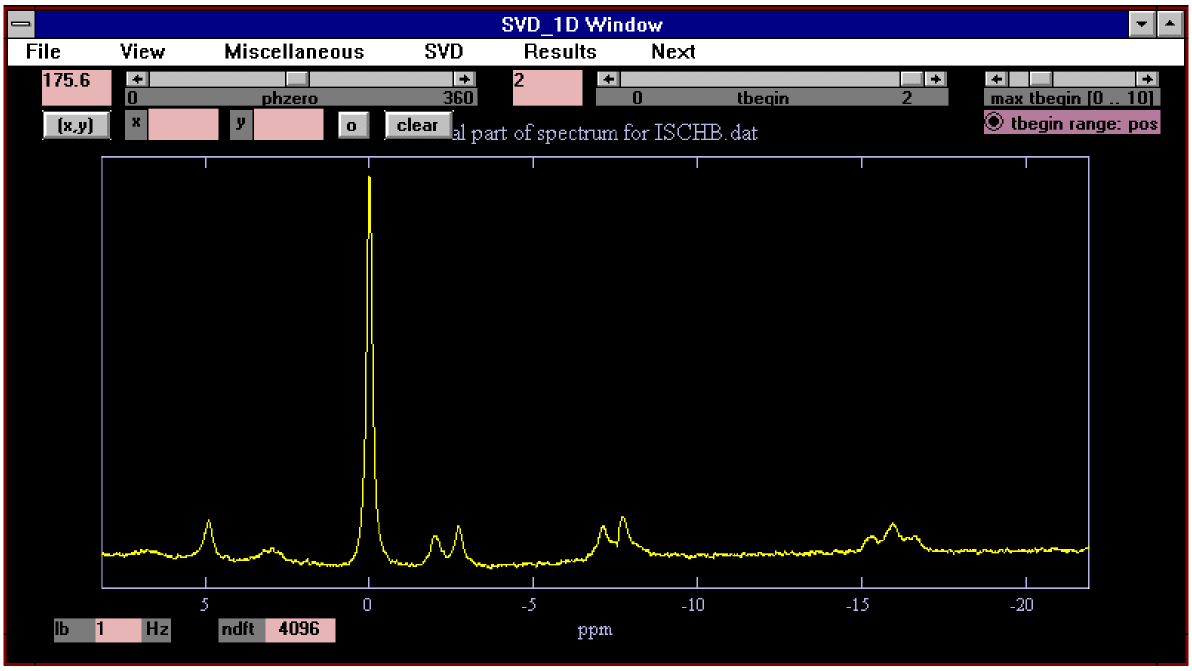


Figure 18. The basic spectral window, in this example as used by the SVD_1D function.

5.1.1 The File menu

The File menu contains some general functions for file management with respect to the spectral display window (loading, saving, exporting, printing). Notice how before you load a data file into the spectral processing function, only the load and quit menu items are active; all other menu items have been disabled until a signal has been loaded into the function. The Results menu (when present) will only become active after the spectral analysis has been carried out, see section 5.2.

5.1.1.1 load

The following three menu items under the load menu are general, although not all functions contain all three of them (see for further details the section belonging to the specific function). The Simulate and Peakpick functions contain extra submenus for the load menu, see their respective chapters 6 and 11.

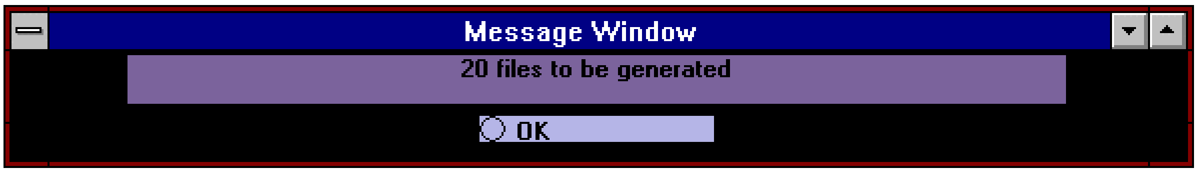
5.1.1.1.1 file

This function allows you to load a file into the spectral function you had selected. The load file menu item will bring up the file manager window which enables you to select a data file to load into the spectral processing function, as described in section 4.6. During any MRUI session, you will be using this file manager rather a lot. The first time in any session it will come up with its default size. Resize it to a convenient size once, and from then on (in that Matlab session) it will have your preferred size (and come up faster, too). All files with a `dat` extension (`*.dat`) are listed. As soon as you press OK after having selected a data file, the file manager window will disappear and the FT spectrum of the time domain signal stored in that data file will be displayed in the spectral window, with the file name as a title above the spectrum (see Figure 18), and most other menu items, buttons and sliders will be enabled. Now you can use these functions as required.

When loading a file using this menu from the result spectral window, i.e. after you had already loaded a previous file and performed your analysis on it, the general (i.e. those that are not signal-specific, such as phasing, line broadening, axis units, etc.) menu items of the other menus will retain their settings as you had applied them in your first analysis. This may greatly enhance easy of use if you wish to carry out certain functions on more than one signal, without running in batch mode. For example, you can load a signal into SVD_1D, change the number of data points, number of components, and algorithm used, and for any subsequent signal you load, the same settings will be applied by default. Should you wish to change settings again, the new settings will be retained in subsequent runs, etc.

5.1.1.1.2 batch

If you wish to process a series of data files in an automatic fashion, use the load batch menu item. This will bring up the batch definition window, see section 4.4.1. After you have defined the batch (giving general base name, start serial number and end serial number of the batch), the first data file from the batch (its file name is composed of the general base name and the start serial number) is loaded, and the FT spectrum of its time domain signal is displayed in the spectral window. The cancel button in the batch definition window by now has changed to a continue button, and the consecutive and non-consecutive radiobuttons have been added to the window. For details on how to use the latter, see section 4.4.1. Clicking on the continue button makes the batch definition window disappear, and most other menu items, buttons and sliders are now enabled. You can use these to set everything in order to have your batch of files run automatically by the spectral processing function. If for the start serial number of the batch neither the `<name>.dat`, *nor* the `<name>.mat` data file is present in the current directory, a message is displayed on the screen (see *Figure 19*) and the load command is canceled.


Message Window

20 files to be generated

Figure 19. If the first file of a batch cannot be found, a message is displayed and the batch load canceled.

When loading a batch of files that was previously batch-processed by some FID processing function (e.g. Filter_HSVD), all processed files in the batch will have got an extension *after* the batch index in the file name. Say you had a batch mybat1.dat, mybat2.dat, ..., mybat8.dat. After using the Filter_HSVD function, the processed files get an _p extension (see section 8.1), so your filtered batch has the following names: mybat1_p.dat, mybat2_p.dat, ..., mybat8_p.dat. After successive processing techniques, this extension might have grown to a few characters (see section 8.1), but all after the batch index in the file name. So what is now the general base name of the batch, and how can MRUI separate that from the batch index? Well, for that purpose the underscore sign _ was introduced in the file name, and therefore you were advised not to introduce underscore signs in your file names yourself (see section 4.1.3). MRUI has a smart 'name-disentangling' routine that automatically takes the appropriate file names in a batch. So if for your original batch in the above example you would have entered for the general file name base mybat with start number 1 and end number 8, you would now, after Filter_HSVD processing, enter for the general file name base mybat_p, with start number 1 and end number 8. Hence for a batch the 'processing extension' comes directly at the end of the original general file name base when defining the batch as in Figure 10. This works completely analogously for batches resulting from all other FID processing techniques.
When loading an irregularly numbered batch of files, you can pretend your nose bleeds, as long as you give correct start- and end numbers of the batch. For instance, you may have done an experiment to determine metabolite T_2's, so you have measured your ^{1}H MRS data with varying T_E's, and your data files are named accordingly. Say you measured at echo times of 30, 50, 68 and 136 ms, and you have four files named myte30.dat, myte50.dat, myte68.dat and myte136.dat. You can process these files in batch by defining a batch with general base name myte, start number of batch 30, and end number of batch 136. It is *not* necessary to uncheck all intermediate indices using the non-consecutive menu (explained in section 4.4.1), because there are no data files with such numbers. When the batch starts running at the start number 30, it will process that file, then look for 31, 32, etc., but since there are no data files found with a corresponding file name, it will just skip over all these intermediate indices until it gets to 50, where it does find a corresponding data file, and will process it. The batch continues analogously until it reaches the end number 136, by which time it will have checked the existence of 136-30+1=107 data files, but it will have actually processed on four of them: the ones you provided.
As the batch definition window does *not* allow you to browse through your directory structure, it is imperative that you select the directory containing your batch first, using the select dir function from the Matlab menu under the MRUI base window, see section 3.2.2.3.
If you loaded a batch of files, once you activate the final menu item that starts the processing function, it will be carried out automatically for all files present in the batch. For this reason it is important that you organize the display settings etc. (see section 5.1.3.5) before you start the batch process. When processing a batch of files, data files with serial numbers that were excluded using the non-consecutive function, will not be processed during the automatic batch. For all serial numbers that were included in the batch, if the corresponding <name>.dat *and* <name>.mat data files do not exist, the batch process skips over that serial number and in most cases displays a message in the Matlab command window. If only the <name>.dat or the <name>.mat data file exists, the other is automatically regenerated.

5.1.1.1.3 result

This function is disabled (inactive) for all functions except Filter_HSVD, SVD_1D and VARPRO. Only these spectral fitting routines yield parameter estimation results that are stored in ASCII result files, rather than automatically saving all pertaining signals (such as e.g. reconstructed model function, individual components and residual) into new data files. The load result menu item will bring up the file manager window, which now lists all files with an extension related to the result files for the selected processing function. In Filter_HSVD the filtered data files are listed (*_p.dat), in SVD_1D the result files *.hrs (see section 10.11) are listed, and in VARPRO the result files *.vrs (see section 13.7) are listed. As soon as you press OK after having selected a result file, the file manager window will disappear and the result window as it was produced when you last did the analysis on the corresponding data file (see e.g. Figure 22), inclusive of the file and fit information table and plotted FT spectra for original signal, reconstruction and residual, are displayed in the spectral window, which immediately takes the larger size as it does after running the fit procedure (see section 5.2). Most menu items, including the Results menu (see section 5.2.1), buttons and sliders will be enabled. Now you can use these functions as required. This is a useful function to recall previous analysis results, complete with all flexibility to change the screen contents, display of individual components or not, possibility to plot results with different phase or line broadening values, etc., without the need for re-doing the actual analyses.

5.1.1.2 print

A number of print functions is available, most of them relying on the Matlab `print` command. For more information on the Matlab print command, type in the Matlab command window:
```
>> help print
```
In some cases the gathering of graphical information from the window to be printed may take a few seconds. During this period the mouse cursor will have the shape of an hour-glass, and you will not be able to use any of the menus or buttons in the window. Only after the mouse cursor has changed back to its usual arrow shape should you start using the menus again.

5.1.1.2.1 to printer

This function is the direct use of the Matlab `print` command without options. It has the same effect as typing
```
>> print
```
in the Matlab command window: the graphics contents (those related to the `axes` variables) of the current window are printed on your default printer, using the settings in the Matlab configuration file `printopt.m`. See the Matlab manual or the on-line help for the `print` command, or type in the Matlab command window:
```
>> help printopt
```

5.1.1.2.2 to file

This menu item allows you to print the graphics contents of the current window to a file. You could transfer such a file to a remote host and have it printed out there, send the graphics printout file via e-mail to a colleague, load the graphics file into a software package for making slides or into a word processing document, etc. etc. Printing to file will bring up the file manager window, which now lists all files with an extension related to the particular graphics format you selected (indicated in parentheses in the following description of formats). Type the full file name for the print file (including extension) in the available field (see also section 4.6, Figure 14), or select an existing file if you want to overwrite it.
A variety of graphics formats is available to store the printout of the spectral window. The default format is the format pertaining to your default printer as configured in the `printopt.m` file, see section 5.1.1.2.1. The extension for these files is depending on the graphics format. The PostScript format (`*.ps`) and Encapsulated PostScript format (`*.eps`) can be used for remote printing, sending graphics by e-mail, or for inclusion in TEX or LATEX documents. GIF format files (`*.gif`) retain the screen colors in the graphics file, and can be inserted into word processing and slide making software packages, or sent by e-mail. It was this graphics format that we used for the figures in the WWW home pages. PCX format files (`*.pcx`) also retain the screen colors, and can be inserted into some more old-fashioned slide making software packages. It should be noted that both for the GIF format files and for the PCX format files, a large white border around the spectral display is saved into the graphics print file. This is a property of the Matlab `print` command that unfortunately cannot be avoided. If your software package has no way to correct for it, you may need some sort of "capture" software to select only the relevant plot area. Also, the printed spectrum and axes have undergone a 90^o rotation in the graphics print file. Finally, the HPGL format files (`*.hpg`) can be used to remote print on a Hewlett Packard plotter.

5.1.1.2.3 to clipboard

This function is only enabled on PCs. If you wish to include the graphics as displayed in the spectral windows into some MS-Windows software application, to clipboard is definitely the best way to do it. It is basically a "copy" function, just like you would use the Edit - Copy menu or `[Ctrl]-C` keys in another MS-Windows application. You can check out the copied graphics contents in the Clipboard or ClipBook Viewer program, or just "paste" it into your target application, using the Edit - paste menu or `[Ctrl]-V` keys.

5.1.1.2.4 numerical results

This menu item will only become active after the spectral analysis has been carried out, see also section 5.2. It prints out the most informative result file for the selected spectral analysis function. For SVD_1D this is the

<name>.hns (see section 10.11) file, and for the VARPRO function this is the <name>.vns (see section 13.7) file. For other spectral processing functions this menu item is disabled. The print command used is the default operating system print command, as taken from the Matlab configuration file printopt.m, see also section 5.1.1.2.1.

5.1.1.3 save ascii plot

With save ascii plot you can save spectra or FIDs to an ASCII file, in order to load them into your software program of choice, such as a spreadsheet program or graphics program. This function brings up the file manager window, which now lists all files with an asp extension (*.asp). Type the full file name for the ASCII plot file (including extension) in the available field (see also section 4.6, Figure 14), or select an existing file if you want to overwrite it. You may use a different file extension from the suggested one. For example, to make the plot file easily recognizable for the gnuplot software, use a .gnu extension.
The x-axis and the data are stored as columns in the ASCII file. If you save ascii plot spectra, the first column contains the x-axis (frequency or chemical shift) according to the axis units displayed in the spectral window (see section 5.1.3.1). If you save ascii plot FIDs, the first column contains the x-axis in data point indices. The region of the spectra or FIDs saved to file is identical to the zoomed in region displayed in the spectral window (see section 5.1.2.1). Non-displayed points are not saved to the ASCII plot file. Further, only the real or the imaginary part of the spectra or FIDs is saved to file, according to which part of the complex data is displayed in the spectral window (see section 5.1.3.2). If the real part of the spectrum is displayed, save ascii plot will save the real part of the spectra or FIDs to the ASCII plot file and vice versa.
In the basic spectral window, when only one spectrum is displayed, save ascii plot will save only that one spectrum or FID, i.e. the ASCII plot file will contain two columns. However, after the spectral analysis has been carried out, and the spectral window has increased in size to display multiple spectra, these functions are automatically updated so that they save all spectra or FIDs to the ASCII plot file. The same rules apply to the numbers saved for the x-axis, the region saved, and whether the real or imaginary parts are saved, as described above for a single signal. The individual components will also be saved as separate columns to the ASCII plot files if they are displayed in the spectral window. The order of the spectra or FIDs as columns in the ASCII plot file is as follows: x-axis, original, reconstructed, residual, and finally all individual components if displayed in the spectral window. The order of the individual components in the ASCII plot file is identical to their order in the parameter estimation table displayed in the left half of the result spectral window.
Further, if the result spectral window is used to display a singular value plot after a Filter_HSVD or SVD_1D analysis (see section 5.2.1.1), the save ascii plot function will automatically change function. The FIDs menu item gets disabled (inactive), and the spectra menu item changes to sinvals. Selecting this function will create an ASCII plot file with the singular values indices in the first column, and the singular values in the second column, ordered according to decreasing magnitude. Restoring the spectral display will also restore the save ascii plot functions.

5.1.1.4 quit

This menu item quits the selected function, clears the spectral window and all corresponding Matlab variables in the workspace.

As you may have seen, the windows that MRUI presents to you are created through Matlab, and dependent on your particular windows environment. Matlab takes care of all this. If you are familiar with the windows on your workstation or PC, you will undoubtedly know that it contains a "close" button of some kind, and a small submenu that also gives you the possibility to close the window. Thus, in any given application, be it the MRUI VARPRO result window or a simple operating system shell tool window, you can always close the window using its particular X window close button. You may have got used to this in other applications or shelltools, but ***DO NOT USE THESE CLOSE BUTTONS/MENUS IN MRUI WINDOWS***. Always use the quit menu item under the File menu, and *not* use the X window close function. This is because the MRUI window, and current function (such as Filter_HSVD, Inputvp, VARPRO, etc.), use many variables that regulate the behavior of the display and signal processing method, and which are stored in memory. If you use the menu item quit to close window, a special MRUI script is called that properly clears these variables, or gives them default values. It also resets the figure window settings. Thus you can carry on with the next function without problems. Should you close the MRUI window using its X window button, the only thing that happens is that the 'X window process' gets killed, and the window indeed disappears. However, all the underlying MRUI variables retain their values belonging to the previously used function. This may cause severe conflicts when starting the next function, and may lead to MRUI crashing.

5.1.2 The View menu

The View menu governs part of the spectral display settings. Other spectral display settings can be changed using the phasing sliders (section 5.1.4), the line broadening (lb, section 5.1.6) and number of FT data points (ndft, section 5.1.7) edit boxes, and from the Miscellaneous menu (section 5.1.3). At this point it is important to realize that a number of these spectral display parameters is saved to a small ASCII file, so that re-loading of the same data file at a later time will automatically restore the spectral display as it was last time you displayed the signal. This auxiliary ASCII file has a default name of `default.gra`, indicating that it stores the graphics parameters. For each data file `<name>.dat/mat`, a copy of `default.gra` is made under the name `<name>.gra`, so that you always retain the desired spectral display for each individual data file. If you want to store such a graphics parameter file for general use in other directories, or load one from another directory, use the default.gra file menu item from the Database menu under the MRUI base window, see section 7.3. The graphic display parameters saved to the `default.gra` and `<name>.gra` files are the zero order phase from the phzero slider (section 5.1.4.1), the begin time from the tbegin slider (section 5.1.4.2), the line broadening from the lb edit box (section 5.1.6), the left and right zoom limits of the x-axis from the zoom settings (section 5.1.2.1) and the x-axis units from the axis units menu item (section 5.1.3.1). The number of data points for the FT spectrum (ndft edit box; section 5.1.7) is stored in the `default.xpt` ASCII file (see section 4.3). Each time you change any of these parameters, the corresponding ASCII file automatically gets updated.

5.1.2.1 zoom in

With the zoom in function you can display a narrower spectral region. The mouse cursor changes into a cross-hair, and an instruction message text bar appears in the lower right corner of the spectral window, saying "1. select left limit of zoom region". Place the cross-hair on the left limit of your intended zoom region and click the left mouse button. A blue vertical line will be displayed on this position The instruction message will change to "2. select right limit of zoom region". Place the cross-hair on the right limit of your intended zoom region and click the left mouse button. A blue vertical line will be displayed on this position, and then the selected spectral region will be displayed in the spectral window. When you get more familiar with the spectral interface, you may wish to suppress the appearance of the blue vertical lines and the instruction message when you zoom, as they may make the zooming process a little slower. Set your preferences using the customize preferences menu from the Matlab menu under the MRUI base window (see section 3.2.1.3).

Apart from this 'one-dimensional' zoom function (only with respect to the frequency axis; for vertical 'zooming' use the scale up menu item, see section 5.1.2.4), another free zoom function is provided for users who have Matlab version 4.2 or later. You can place the mouse cursor on any position within the axis borders, press the left mouse button, and while keeping the left mouse button depressed, drag an area contained in a dotted-line square box. Releasing the left mouse button redraws the spectrum over the selected area only. Pressing the right button now zooms out in steps until the original display has been reached. Quickly double-clicking the left mouse button restores the original display in one step. You can also simply click the left mouse button on any position inside the axis borders, this will perform a very detailed zoom in on that particular region. Zooming out goes as described for zooming an area. However, these (Matlab) zoom settings can not be retained when the spectral display is refreshed. Hence, for settings that should remain throughout the various plots, it is better to keep using the zoom functions from the View menu.

5.1.2.2 zoom out

This menu item displays the spectrum over its entire spectral width.

5.1.2.3 zoom from file

This menu item allows you to zoom a spectrum according to the zoom settings you performed on another spectrum. It brings up the file manager window, which now lists all files with a `gra` extension (`*.gra`). You can

browse through the directory structure if necessary, and then select the graphics parameter file that you wish to use. Upon pressing OK, the spectrum in your spectral window will be zoomed in according to the zoom settings for the other data file, and its own `<name>.gra` file will be updated.
Most zoom functions (including the Matlab free area zoom) behave identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis. Only the zoom from file function is disabled (inactive) in the singular value plot display after Filter_HSVD or SVD_1D analysis.

5.1.2.4 scale up

With the scale up function you can display a narrower region of the y-axis. The mouse cursor changes into a cross-hair, and an instruction message text bar appears in the lower right corner of the spectral window, saying "1. select low limit of scale region". Place the cross-hair on the lower limit of your intended vertical zoom region and click the left mouse button. A blue horizontal line will be displayed on this position The instruction message will change to "2. select high limit of scale region". Place the cross-hair on the upper limit of your intended vertical zoom region and click the left mouse button. A blue horizontal line will be displayed on this position, and then the selected vertical region will be displayed in the spectral window. When you get more familiar with the spectral interface, you may wish to suppress the appearance of the blue horizontal lines and the instruction message when you scale, as they may make the scaling process a little slower. Set your preferences using the customize preferences menu from the Matlab menu under the MRUI base window (see section 3.2.1.3).
When you use the scale up function in the result spectral window with multiple spectra on display, it is important that you select both the lower and upper scale limit within the same spectrum, otherwise a warning message appears in the lower right corner of the spectral window, and the scaling will not be performed. After you have selected a vertical scale region in one spectrum, all other spectra on display will be scaled analogously.

5.1.2.5 scale down

This menu item displays the spectrum over its vertical span, including a five percent margin below the minimum intensity and above the maximum intensity. The exact result of the scale down function is strongly dependent on the automatic scaling setting from the Miscellaneous menu (see section 5.1.3.4). If automatic scaling is on, then the scale down function will display the spectrum over its vertical span *in the displayed spectral region only*. If automatic scaling is off, then the scale down function will display the spectrum over its entire vertical span based on *the entire spectral width*. This may cause large visual differences if the zoomed in spectral region displays only low-intensity peaks, with a high-intensity peak outside the displayed spectral region, see also section 5.1.3.4. The scale down function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis.

5.1.2.6 auto scale

This menu item displays the spectrum over its vertical span *in the displayed spectral region only*, including a five percent margin below the minimum intensity and above the maximum intensity. This means that the peaks visible in the displayed spectral region will be fully scaled to the maximum available y-axis height. The auto scale function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis.

5.1.2.7 x-axis labels

With this menu item you can switch the x-axis labels (i.e. the values of the frequencies or chemical shifts) on or off. The x-axis unit string below the x-axis labels will toggle likewise. The x-axis labels function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis.

5.1.2.8 *y-axis labels*

With this menu item you can switch the y-axis labels (i.e. the spectral intensity values) on or off. The y-axis labels function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis.

5.1.2.9 *annotate*

With the annotate menu item you can add text to your graphics display. Once you select annotate, the annotation window will appear, see Figure 20. Place the mouse cursor inside the edit box, click the left mouse button, and type your annotation. After you press `[Enter]` (or on some machines, after you move the mouse cursor to outside the edit box), a message appears, telling you to click with the mouse in the correct position (see Figure 20). Move your mouse cursor over the spectral display and you will notice that its shape has changed to a crosshair. Point it at the position where you want your text to start, and then click the left mouse button. Your annotation will then be added to the spectral display at the position you indicated.

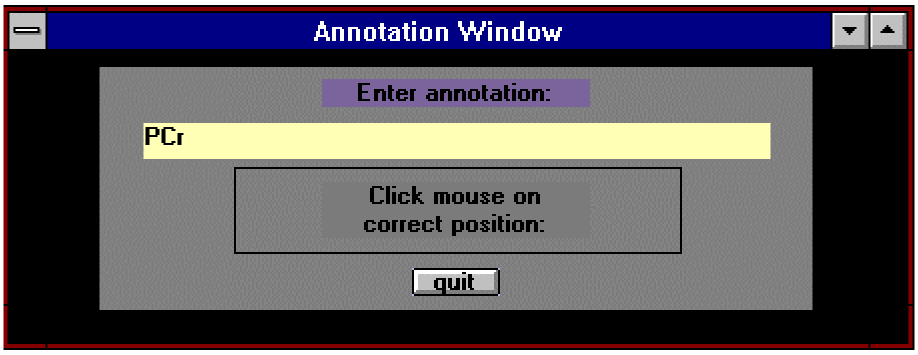


Figure 20. The annotation window can be used to add text to your spectral display.

The annotation window will remain visible, and you can type your next annotation. You may have to delete the previously typed characters for this, or you can select characters to be replaced, just as in a word processor program. You can type a line of words, using spaces and punctuation signs, but you cannot annotate with multi-line text. In this case you will have to do multiple annotations and try and place them correctly on the spectral display. Once you are finished with your annotations, click on the quit button in the annotation window (see Figure 20). All print functions will include the annotations in the spectral display. See Figure 21 for an annotated spectral display.

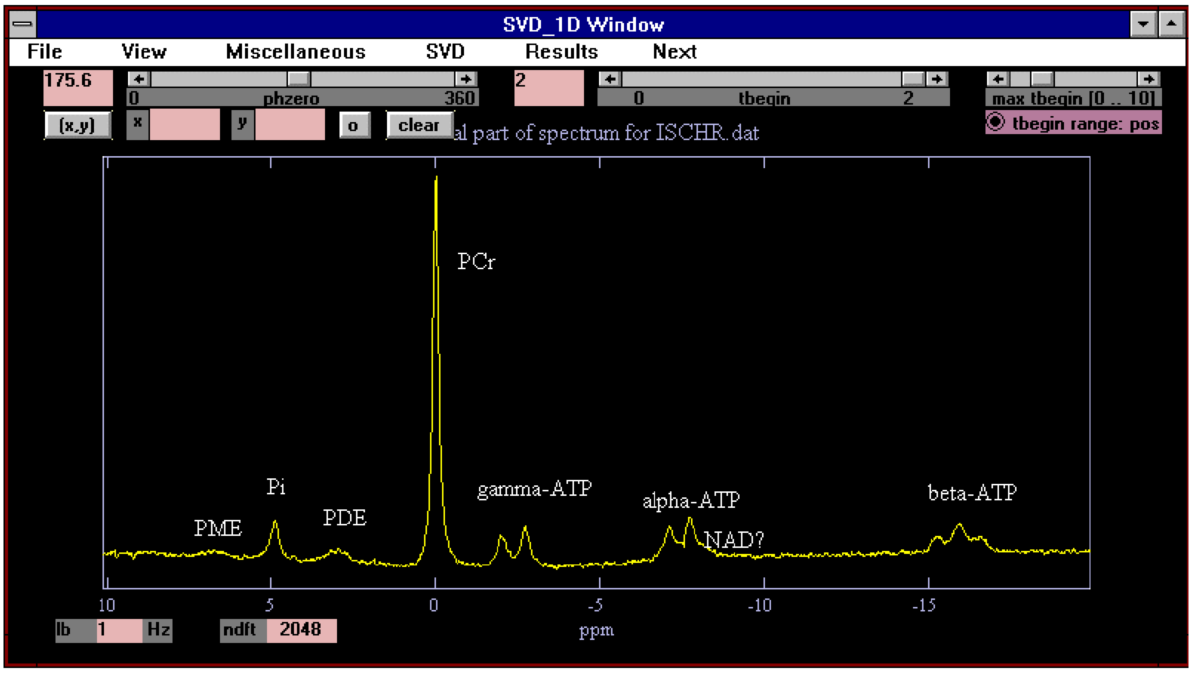


Figure 21. The spectral window displaying a zoomed in region of a ^{31}P spectrum with peak annotations.

Be aware that once you change the spectral contents (e.g. by changing the line broadening, the number of points used in the FT spectrum or the phasing), that the annotation text is lost once a new spectrum is drawn. For this reason, you should best wait with your annotations until all other graphical display settings are as desired, and you only wish to add the text and then use a print function. If you placed an annotation that you wish to undo, there is no other option than to refresh the spectral contents. You can do this for instance by using the line broadening edit box but typing the same lb value (see section 5.1.6), or typing the current zero order phase value again in the phzero edit box and pressing `[Enter]` (see section 5.1.4.1). Unfortunately, this means that the other annotation are also undone and that you have to start over. You can also use the reset function to undo annotations, but this may change other graphical display values as well (see section 5.1.2.10). The annotate function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis.

5.1.2.10 reset

The reset function resets graphical display parameters to their original values and redraws the spectrum. However, on this occasion, the `default.gra` and `<name>.gra` files are *not* updated. The reset function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis.

5.1.3 The Miscellaneous menu

This menu contains the remaining functions that determine the response of the spectral display.

5.1.3.1 axis units

With this menu item you can change the axis units between kilohertz (kHz), hertz (Hz), parts per million (ppm) and the point index for the FT spectrum (points). When displaying the frequency axis as chemical shifts in ppm, the ppm reference is taken from the settings of the SETUP: experimental function (see section 7.1.5). Note that these settings of the axis units determine the value in which the estimated frequencies are displayed in the table in the result spectral window (see sections 5.2, 10.7 and 13.4), and in the `<name>.hns` file (for SVD_1D, see section 10.11) and the `<name>.vns` file (for VARPRO, see section 13.7). Changing the axis units after the result spectral window has been generated no longer affects the frequency values in the table and the result files. The axis units function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window. In the singular value plot display after Filter_HSVD or SVD_1D analysis, the axis units menu is disabled (inactive). Finally, it should be noted that there are some known problems with the implementation of the points function. This may cause unexpected behavior of your functions when updating spectral views in which the axis units are expressed in FT spectral points. It is on 'the repair-list'.

5.1.3.2 real / imaginary

With this function you can toggle between display of the real or imaginary parts of the spectra. This setting does not get written to the `default.gra` file (see section 5.1.2), but its default value can be set using the customize preferences function (see section 3.2.1.3) from the Matlab menu under the MRUI base window. The package default is to display the real parts of the spectra. Note that the setting of this function determines which part of the complex spectra or FIDs get written to the ASCII plot file (see section 5.1.1.3). The real/imaginary function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window. In the singular value plot display after Filter_HSVD or SVD_1D analysis, the real/imaginary menu is disabled (inactive).

5.1.3.3 normal / reverse

With this function you can toggle between display of the spectra according to the convention of physics, i.e. from negative frequencies to positive frequencies (select normal), or according to the convention of NMR spectroscopists, i.e. from positive frequencies/chemical shifts to negative frequencies/chemical shifts (select reverse). This setting does not get written to the `default.gra` file (see section 5.1.2), but its default value can be set using the customize preferences function (see section 3.2.1.3) from the Matlab menu under the MRUI base window. The package default is the display in reverse mode (NMR standard). Note that the setting of this function determines in which direction the spectra get written to the ASCII plot file (see section 5.1.1.3). This setting has no effect on how the FIDs are written to the ASCII plot file. The normal/reverse function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window. In the singular value plot display after Filter_HSVD or SVD_1D analysis, the normal/reverse menu is disabled (inactive).

5.1.3.4 automatic scaling

This is the toggle on/off function for the auto scale property of the zoom functions described in sections 5.1.2.1-5.1.2.3 and it works as described for the auto scale menu item in section 5.1.2.6. If the automatic scaling is set to off, then using the zoom in and zoom out functions has no effect on the vertical scaling. This means that if you have one intense peak in your spectrum, many other peaks may be very small. Zooming in over their region would still display them as very small with a lot of nothing above them. However, if you set the automatic scaling option to on, the spectral view will always be automatically scaled up or down, depending on the maximum intensity in the spectral region you zoom in on (or to maximum intensity when you zoom out). Notice that there is always a 5% empty margin below and above the spectra so that the spectral graph does not touch the axes. This option only effects the zoom in, zoom out and zoom from file functions, and not Matlab's free area zoom function (see section 5.1.2.1). This option may be especially beneficial when switching from real to imaginary display, or when zero order phasing. Notice that the actual spectrum remains untouched, and that the y-axis labels are indicative of your position relative to the original vertical position of the real part of the spectrum. automatic scaling is on by default. The auto scale menu item from the View menu (see section 5.1.2.6) can carry out this 'window-filling-scaling' at any chosen moment. The automatic scaling function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis.

5.1.3.5 batch functions

The batch functions menu generally contains two submenus, which characterize the spectral display during the running of an automatic batch. When having selected either the Peakpick (see chapter 11) or VARPRO (see chapter 13) function, an extra menu item will be added to this menu, in order to determine part of the algorithmic behavior of the spectral analysis function during the running of an automatic batch. These extra functions are discussed together with their corresponding spectral analysis techniques. See section 11.2 for Peakpick's extra functions concerning the use of the phzero and tbegin sliders and section 13.1 for VARPRO's extra functions concerning the use of the parameter estimation results of signal number `i` as starting values for the fit of signal `i+1`, and the use of Peakpick and Inputvp files found from a previous run.

5.1.3.5.1 display all

With this function you can toggle between display of the result spectral window (see section 5.2) during the automatic batch. If display all is set to on, the result spectral window will be displayed for each file that is processed in the batch. This may make your batch processing time a little slower, but it may be more informative if you wish to check the running of your batch regularly. If display all is set to off (*and* print all set to off, see the next section), then the spectrum of the first file of the batch will remain on display in the basic spectral window (see e.g. Figure 18) until the last file of the batch has been processed. Then the result spectral window will be displayed for that last file of the batch. With display all (and print all) set to off, you can make an icon of the spectral window, and it will only open up after the entire batch has been processed. After a batch has been processed, the display all menu item is still active, and changing it will affect the display of the next batch you load.

5.1.3.5.2 print all

With this function you can toggle between printing of the result spectral window (see section 5.2) during the automatic batch. If print all is set to on, the result spectral window will be printed (in exactly the same way as if you would select print to printer for an individual display, see section 5.1.1.2.1) for each file that is processed in the batch. As follows from the description of the printing process in section 5.1.1.2, the result spectral window has to be on display before it can be printed. So whether display all has been set on or off, the result spectral window will be displayed for each file that is processed in the batch. After a batch has been processed, the print all menu item is still active, and changing it will affect the printing and display of the next batch you load.

Please think of the trees, and of the rain forests and their precious wild animals and plants. Use this function only if you feel you have to, and if you checked before that the batch is going to be successful, so that the printouts will serve as some important work of reference.

5.1.3.6 peak numbers

The peak numbers menu is only enabled (active) after the analysis has finished, and the result spectral window displayed (section 5.2). You can use it to switch the display of the peak numbers in the multiple spectrum display on or off. You can change the default setting for the peak numbers menu using the customize preferences menu (see also section 3.2.1.3). The package default is to display peak numbers (i.e. on).

5.1.4 Phasing

To phase the spectra that are displayed in the spectral window, you can use the two sliders above the spectrum: the phzero slider for the zero order phase and the tbegin slider for the first order phase. It is important to realize that since the spectral analysis methods operate in the time domain (i.e. on the FIDs), phasing the spectrum only affects its display, and *not* the raw data that are being analyzed. However, phasing the spectrum can still be very important for your recognition of resonances, supplying starting values (see chapter 11), and visual interpretation of the data and results. Also, the actual values of the zero order phase and begin time, as read from their sliders, function as starting values in case they are to be estimated by VARPRO (see section 12.4.6).
On the other hand, if you wish to affect your raw FID data points with the zero order phase (multiply the FID with a complex phase factor) and/or begin time, you can use a similar spectral display with identical sliders within the Phase function of the FID Maths menu, see section 8.10.

The phasing sliders behave identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window. In the singular value plot display after Filter_HSVD or SVD_1D analysis, the phasing sliders and edit boxes disappear altogether.

5.1.4.1 zero order phase

The zero order phase ('overall phase', equal for all peaks) of the spectrum can be changed either by pulling the phzero slider to the left or the right (between 0 and 360 degrees only; all other values have their equivalent within this range). See section 3.2.1.2 on how to use MRUI sliders. Notice how with each step of the slider, the spectral display is updated with the new phase value. In most cases it is best to drag the slider along to its approximate position, and then do some fine tuning with smaller steps. If you wish to impose a certain numerical value immediately, you can type that zero order phase value into the edit box on the left of the phzero slider. Manually editing the zero order phase value in the phzero edit box immediately updates the spectral display and the position of the phzero slider, just as moving the phzero slider updates the numerical value in the phzero edit box. The value of the zero order phase, as you have set it using the phzero slider or edit box, is stored in the `default.gra` and `<name>.gra` files, see section 5.1.2.

5.1.4.2 begin time / first order phase

The tbegin slider essentially effectuates a first order phasing of the spectrum, since changing the delay time of the FID (time difference between the true time origin `t=0` and the first data point of the FID) causes a frequency dependent (in a linear fashion) phase modulation of the entire spectrum[2]. This 'first order phase correction' then always has its *pivot* at 0 Hz. For this reason it is important that you first use the zero order phase slider or edit box to properly phase the region around 0 Hz before you start 'first order phasing' using the begin time slider. The tbegin slider and its corresponding edit box work exactly like the phzero slider and its edit box, as described in the previous session. As for the range of the tbegin slider, it ranges from a negative minimum to a positive maximum of the same value. The package default for that value is 4, but this default can be changed using the customize preferences function from the Matlab menu under the MRUI base window, see Figure 5 in section 3.2.1.3. During a session, you can change this maximum absolute value for the tbegin slider from 0 to 10 using the small slider in the upper right corner (max tbegin). Any other values can be typed in the tbegin edit box. A small maximum value of your tbegin slider allows accurate fine tuning of the begin time variable (first order phase of the spectrum) using the slider. By default, the tbegin slider ranges from negative to positive values, as negative begin times can occur due to digital filters in the electronic circuit of the spectrometer acquisition modules[2]. However, you can toggle between a tbegin slider that ranges from negative to positive and a range from zero to a positive maximum, using the radio button tbegin range on the right. If you only use positive values for the begin time, this again allows more accurate positioning of the slider because the range of values is limited. The value of the begin time, as you have set it using the tbegin slider or edit box, is stored in the `default.gra` and `<name>.gra` files, see section 5.1.2.

Other phasing features (e.g. linearly dependent on the frequency but with an arbitrary pivot, or higher order phases), meant to allow greater control over individual peak phases, are not implemented in the MRUI software, as they have no relevance for time domain signal analysis, where the zero order phase and begin time ('first order phase') have a physical meaning in the model function (but you might want to read section 8.6.2.2 anyway). Individual peak phases are always estimated by the SVD_1D techniques (because in these techniques all model function parameters are free variables, see chapter 10), and can be estimated by VARPRO by fixing the zero order phase and begin time to their current values, and then leaving phases for a peak or a group of peaks unconstrained (see section 12.5.1.2). This does not require starting values and therefore no spectral interaction.

5.1.5 (x,y) co-ordinates

Below the phzero slider, you can see three pushbuttons ((x,y), o and clear) and two edit boxes (x and y). These allow you to accurately determine x and y co-ordinates in the spectral display, and therefore also chemical shifts of various spectral features of which you cannot readily determine their chemical shifts from the axis labels. They also allow you to draw circles on pre-determined positions, which could be combined with annotation (see section 5.1.2.9) before a printout of the spectral display. To determine the x and y co-ordinates of an arbitrary point in the spectral display, click on the (x,y) button and, with the mouse cursor changed into a cross-hair, click with the left mouse button on the desired position in the spectrum. The x and y co-ordinates will then be displayed in the x and y edit boxes, respectively. The y co-ordinate is arbitrary, but the x co-ordinate indicates the chemical shift in the units as displayed on the x-axis (see section 5.1.3.1). Whenever a pair of x and y co-ordinates is visible in their respective edit boxes, you can click on the o button, and it will plot a cyan circle on the corresponding position. You can thus have cyan circles displayed on pre-determined chemical shifts, simply by typing the chemical shift value (in units of the displayed x-axis) in the x edit box and then clicking on the o button. To make sure the circles are drawn at the appropriate height, use the (x,y) button first, click on the desired height and then only change the value in the x edit box. By pressing the clear button, all currently displayed cyan circles will be cleared from the spectral display. The (x,y) co-ordinates function behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window, or in the singular value plot display after Filter_HSVD or SVD_1D analysis. In the singular value plot display after Filter_HSVD or SVD_1D analysis, the (x,y) co-ordinates function may be particularly useful in identifying the exact singular value index at which the transition from signal related singular to the noise plateau occurs.

5.1.6 line broadening

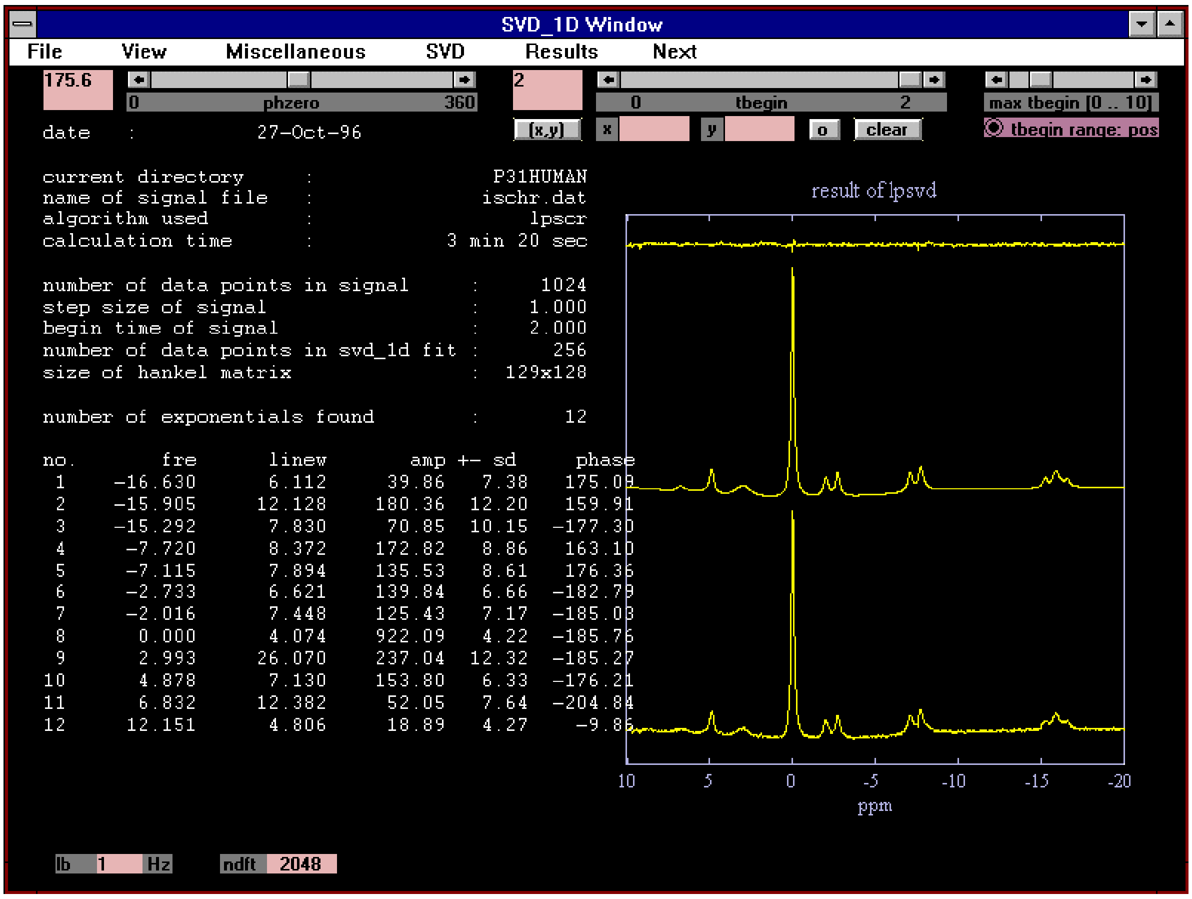
To line broaden the spectra that are displayed in the spectral window, you can use the lb edit box in the lower left corner of the spectral window. This value is 0 by default, but you can type your preferred value (expressed in Hz) here. The line broadening is realized by multiplying the FID with a Lorentzian apodization function before Fourier transform. Notice that typing a negative value here effectively yields a 'line narrowing' of your spectra. To undo the line broadening, type `0` in the lb edit box. It is important to realize that since the spectral analysis methods operate in the time domain (i.e. on the FIDs), line broadening the spectrum only affects its display, and *not* the raw data that are being analyzed. However, line broadening the spectrum can still be very important for your recognition of resonances, supplying starting values (see chapter 11), and visual interpretation of the data and results.
On the other hand, if you wish to affect your raw FID data points with the line broadening (or 'narrowing') factor (e.g. if you want to apply a so-called 'matched filter' to your FID before parameter estimation), you can use he Apodize function of the FID Maths menu, see section 8.11. With this function you can also choose between Lorentzian and Gaussian apodization.
The value of the line broadening, as you have set it using the lb edit box, is stored in the `default.gra` and `<name>.gra` files, see section 5.1.2. The lb edit box behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window. In the singular value plot display after Filter_HSVD or SVD_1D analysis, the lb edit box disappears altogether.

5.1.7 number of points in FT spectrum

To change the number of points in the FT spectra that are displayed in the spectral window, you can use the ndft edit box in the lower left corner of the spectral window. The default value for this number depends on the number of data points in the signal and the conversion routine used (see section 4.3). You can type your preferred number here. Typing a number smaller than the number of data points in the FID will cause the displayed spectrum to be the Fourier transform of the first part of the signal only. Typing a number larger than the number of data points in the FID will effectively involve zero filling the time domain signal before Fourier transform. If the default number of spectral points for your application involves zero filling, you may suffer from severe spectral wiggles if your FID is not fully decayed. In such a case, it is best to take ndft equal to the number of data points in the FID (i.e. complete Fourier transform without zero filling). It is important to realize that since the spectral analysis methods operate in the time domain (i.e. on the FIDs), changing the number of points in the spectrum only affects its display, and *not* the raw data that are being analyzed. However, zero filling (yielding interpolation between adjacent FT points in the spectrum) can still be very important for your recognition of resonances, supplying starting values (see chapter 11), and visual interpretation of the data and results. The number of points for the FT spectrum, as you have set it using the ndft edit box, is stored in the `default.xpt` file, see section 4.3. The ndft edit box behaves identically whether in single spectrum display in the basic spectral window, or in multiple spectra display in the result spectral window. In the singular value plot display after Filter_HSVD or SVD_1D analysis, the ndft edit box disappears altogether.

5.2 The result spectral window (large)

After the spectral analysis has been carried out, the spectral window changes size in order to display the analysis results using a multi-spectra display, involving the original spectrum, the FT spectrum of the time domain model function with the estimated parameters filled in, and the residual, which is the difference between those two. A list with data file and spectral analysis information, plus estimated parameters, is also displayed. An example of the result spectral window is displayed in Figure 22. In this example, the result spectral window displays the results of the SVD_1D fit on the *in vivo* ^{31}P MRS signal from human calf described in section 3.3. For more technical details on the SVD_1D analysis, see chapter 10. The Results menu is now enabled (active), and so is the print numerical results menu item (see section 5.1.1.2.4).



```
date          :          27-Oct-96

current directory      :                P31HUMAN
name of signal file    :               ischr.dat
algorithm used         :                   lpscr
calculation time       :          3 min 20 sec

number of data points in signal      :      1024
step size of signal                  :     1.000
begin time of signal                 :     2.000
number of data points in svd_1d fit  :       256
size of hankel matrix                :   129x128

number of exponentials found         :        12

no.       fre       linew        amp +- sd      phase
 1    -16.630      6.112      39.86   7.38    175.08
 2    -15.905     12.128     180.36  12.20    159.91
 3    -15.292      7.830      70.85  10.15   -177.30
 4     -7.720      8.372     172.82   8.86    163.10
 5     -7.115      7.894     135.53   8.61    176.36
 6     -2.733      6.621     139.84   6.66   -182.79
 7     -2.016      7.448     125.43   7.17   -185.03
 8      0.000      4.074     922.09   4.22   -185.76
 9      2.993     26.070     237.04  12.32   -185.27
10      4.878      7.130     153.80   6.33   -176.21
11      6.832     12.382      52.05   7.64   -204.84
12     12.151      4.806      18.89   4.27     -9.86
```

Figure 22. The result spectral window, in this example after an SVD_1D fit.

The large result spectral window is only displayed for fitting results of the functions Filter_HSVD, SVD_1D and VARPRO. The result window will then be displayed when you are processing a single file, or when in batch you have display all set to on, print all set to on, or if you have come to the last file of the batch (see section 5.1.3.5).

The file and fit information in the table on the left is taken from the numerical results file (`<name>.hns` for Filter_HSVD and SVD_1D; `<name>.vns` for VARPRO). The exact contents of this table are also saved to a separate ASCII result file, called `<name>.lrf` for Filter_HSVD, `<name>.lrs` for SVD_1D, and `<name>.lrv` for VARPRO, respectively. You can use these files for inclusion into word processing documents, or print them separately.
The calculation time listed in this table is the Matlab CPU time between launching the signal processing method and the reconstruction of the time domain model function with the estimated parameters, and may therefore be considerably longer than the actual calculation time needed by the Fortran signal processing algorithm.
The estimated model function parameters are summarized in the table. For each peak are listed the peak number, its frequency or chemical shift, linewidth, amplitude plus standard deviation and phase. The full fit results,

including standard deviations for all parameters, are listed in the numerical result files (`<name>.hns` for Filter_HSVD and SVD_1D, and `<name>.vns` for VARPRO). These parameters need some explanation.

peak number (no.)
The peak numbers in the table help you identify to which spectral resonances the estimated parameters belong. The fitted peaks are numbered accordingly in the reconstruction spectrum in the right half of the result spectral window (see e.g. Figure 52, section 13.4). If the reconstruction spectrum is not displayed, the peak numbers are displayed over the individual components, and if those are not displayed either, the peak numbers are displayed in the bottom spectrum. See also section 5.1.3.6. Notice how for SVD_1D the peak numbers are in numerical order, whereas for VARPRO they run in the order you selected the peaks using Peakpick. The peak numbers in the table may be important when indicating which peaks are to be used for calculating pH (see sections 10.10 and 13.6) or absolute concentrations (see sections 10.7 and 13.4).

frequency / chemical shift (fre)
The chemical shifts in this column are listed in the units of the displayed axis, see section 5.1.3.1. The frequency parameters in the exponentially decaying sinusoid model function[3], as fitted by the time domain analysis techniques, are the positions of the spectral resonances on the frequency axis. The frequencies in this column are adapted to the current display of the x-axis, but the original estimated frequencies in kHz can still be found in the result files `<name>.hrs` for Filter_HSVD and SVD_1D, and `<name>.vrs` for VARPRO.

linewidth (linew)
The linewidths in this column are always listed in Hz. The linewidths of the spectral resonances are the damping factor parameters in kHz (divided by π) in the exponentially decaying sinusoid model function[4], as fitted by the time domain analysis techniques. These damping factors in kHz are the reciprocals of the effective transverse relaxation times T_2^* in ms. You can check in this column whether all resonances have been appropriately fitted; e.g. unrealistically large linewidth values may indicate fitting of broad underlying components instead of the desired narrow spectral resonances. The original estimated damping factors in kHz can still be found in the result files `<name>.hrs` for Filter_HSVD and SVD_1D, and `<name>.vrs` for VARPRO. In some cases, especially when overestimating the model order in the SVD_1D (or Filter_HSVD) analysis of FIDs with few data points, that are not fully decayed, it is possible that the 'black-box' SVD methods (see chapter 10) find an optimal solution in which one or a few of the components have a positive damping factor. Combined with a small amplitude, such an exponentially increasing component may well be fitted over the limited number of data points. This is the theoretical explanation for the occurrence of negative linewidths in the result table. If you then display the spectra with a number of spectral points larger than the number of FID data points (see section 5.1.7), you may seriously suffer from severe wiggles in your spectral display, and you may have to decrease the number of spectral points to interpret the results.

amplitude (amp)
Some consternation has arisen over the actual meaning of the amplitude values in this column. Are they the peak areas? Are they the amplitudes of the peaks, i.e. the peak heights? Well, they are the estimated *amplitudes of the exponentially damped sinusoids in the time domain signal.* This is the intensity of the individual fitted component at the true time origin `t=0` (i.e. the value of the begin time has been included in its calculation). This value (say `a`) is directly proportional to the total integrated area under the real part of the Lorentzian or Gaussian peak in the frequency domain spectrum (`A=a/2`)[3,4]. To obtain the peak height of the real part of a Lorentzian peak (assuming anyone would want that in the first place) you must divide the amplitude value by the damping factor (or divide by `linew*π`). This column also lists an estimate of the standard deviation for the estimated amplitude (estimated standard deviations for all parameters are listed in the numerical result files), which is calculated using the Cramér Rao theory[3,5]. It is important to realize that amplitudes (and therefore also peak areas) of a single data set can only be expressed in arbitrary units. For this reason the absolute values only become relevant when comparing between signals. By default, the amplitudes listed in this table are the amplitudes of the exponentially damped sinusoids in the *scaled* time domain signal. They are also listed as such in the result files `<name>.hrs` for Filter_HSVD and SVD_1D, and `<name>.vrs` for VARPRO. For details on the scaling of the FIDs, see section 4.2. This means that before you can compare the values between signals, you must first correct for the scaling factor, which was stored in the `<name>.scl` file. However, MRUI has already done this for you, and lists the scale-corrected *absolute amplitudes*, i.e. the true intensities of the exponentially damped sinusoids in the original spectrometer file, together with the corrected standard deviation estimates, in the result files `<name>.hns` for Filter_HSVD and SVD_1D, and `<name>.vns` for VARPRO. You can have these scale-corrected absolute amplitude values displayed in your table by default, instead of the scaled amplitudes,

but remember that the scaling is carried out partly to avoid excessively large intensity values distorting the table and spectral display in the result spectral window. Should you wish to have the true intensities displayed in your result spectral window tables by default, then use the amplitudes in result table menu item from the SVD_1D and/or VARPRO menu in the customize fitting parameters window, accessible from the Matlab menu under the MRUI base window, see section 3.2.1.4.

phase (phase)
The phases in this column are listed in degrees. The phase parameters in the exponentially decaying sinusoid model function[3,4], as fitted by the time domain analysis techniques, are the phases of the spectral resonances in the spectrum.

zero order phase and/or begin time
After VARPRO analysis of a data file, this table may also list the estimated zero order phase and/or estimated begin time. See sections 5.1.4 and 13.4.

5.2.1 The Results menu

The Results menu allows you to modify the result spectral window display according to your preferences.

5.2.1.1 display

In the VARPRO function, this menu item has only one submenu, labeled spectra, which is checked by default. Selecting this menu item then only results in the spectral display being refreshed. After Filter_HSVD or SVD_1D analysis however, you can use this menu to toggle between display of the resulting spectra as in Figure 22, or display of the singular value plot[3], see Figure 23.

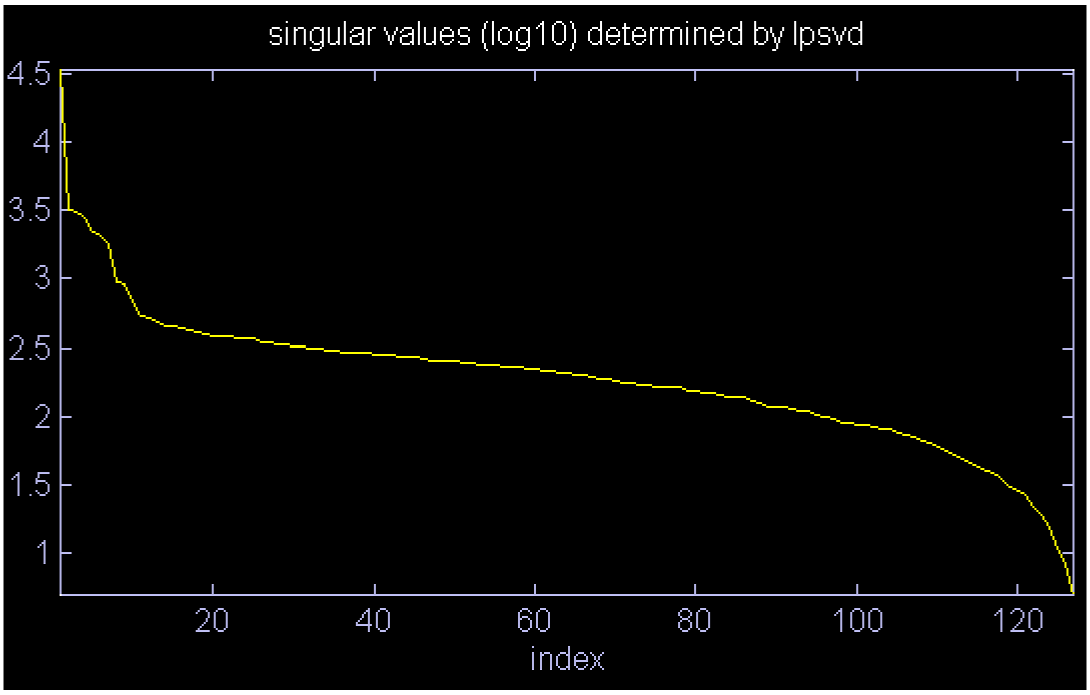


Figure 23. A singular value plot after SVD_1D analysis as it can be displayed in the result spectral display.

You have the option to display the singular value plot on a linear axes plot via sinvals, or on a semi-log plot (the 10-log of the singular values plotted against singular value index, as in Figure 23) via log(sinvals). When the singular value plot is displayed, the phzero and tbegin sliders, plus the lb and ndft edit boxes disappear, and some functions under the Miscellaneous menu are disabled, as these functions have no relevance for a singular value plot. To return from the singular value plot to normal spectra display, select spectra from the display menu item. The functions that had previously disappeared or been disabled will now return or be enabled again.

5.2.1.2 plotting order

The spectra in the result spectral window are always displayed in the order original - reconstruction - individuals - residual, but with this menu item you can choose whether the original spectrum is displayed at the bottom and the residual at the top (via original bottom), or whether the original spectrum is displayed on the top and the residual at the bottom (via original top). This is simply to suit your personal preference, e.g. for making a printout. It does *not* affect the order in which the spectra are written to the ASCII plot file when selecting save ascii plot - spectra from the File menu (see section 5.1.1.3). As package default, the original spectrum is displayed at the bottom. However, you can change the default setting using the Spectral results menu in the customize preferences window (see section 3.2.1.3). You can create separate default settings for the Filter_HSVD, SVD_1D and VARPRO functions.

5.2.1.3 which spectra

This menu item has four submenus: original spectrum, reconstructed spectrum, residual spectrum and individual components. The first three of them are checked by default, which means that by default, the original spectrum, the FT spectrum of the reconstructed time domain model function, and the residual between them are displayed in the result spectral window. After Filter_HSVD, the label reconstructed spectrum is changed to HSVD fit spectrum, and the label residual spectrum is changed to filtered spectrum, respectively. In that case only the original spectrum and the filtered spectrum are on display by default (see also chapter 9). You can change the default setting using the Spectral results menu in the customize preferences window (see section 3.2.1.3). You can create separate default settings for the Filter_HSVD, SVD_1D and VARPRO functions. By selecting a submenu item, you toggle its display between on and off. With each corresponding modification of the spectral display, the spectra will be rescaled such that the entire axes are used to allow maximum resolution (but read section 5.2.1.4). As a minimum you can have one particular spectrum on display; you can not switch all spectra off, the last spectrum will remain on display and a warning message will appear in a message text box in the lower right corner of the spectral window. By selecting individual components, all individual fitted resonances will displayed in one spectrum, i.e. all overlapping (but with zero baseline). If you fitted many components, this may be slow the first time you switch on the display of individual components, because they are only calculated the first time this function is selected (to save calculation time in case you do not want them displayed anyway). Notice that the peak numbering (in VARPRO) is attached to the reconstructed spectrum, so switching off its display may make it more difficult to relate estimated parameters in the table on the left to resonances in the remaining spectra on display.
To have the individual components saved to the ASCII plot file when selecting save ascii plot from the File menu (see section 5.1.1.3), they have to be in the spectral display.

5.2.1.4 scale to

This function determines which of the various spectra is fully scaled (within the displayed frequency region). By default, the scale setting is to overall, which means that all spectral points of all spectra in the displayed frequency region are visible. Only after Filter_HSVD is the default scale setting to filtered spectrum, so that the effect of the filtering on the resonances of interest is best visible. However, you can change the default setting using the Spectral results menu in the customize preferences window (see section 3.2.1.3). You can create separate default settings for the Filter_HSVD, SVD_1D and VARPRO functions. Setting scale to another spectrum which is smaller than others (e.g. in case you select scale to residual) causes the larger spectra to be clipped. See Figure 24 in which the scale was set to residual.

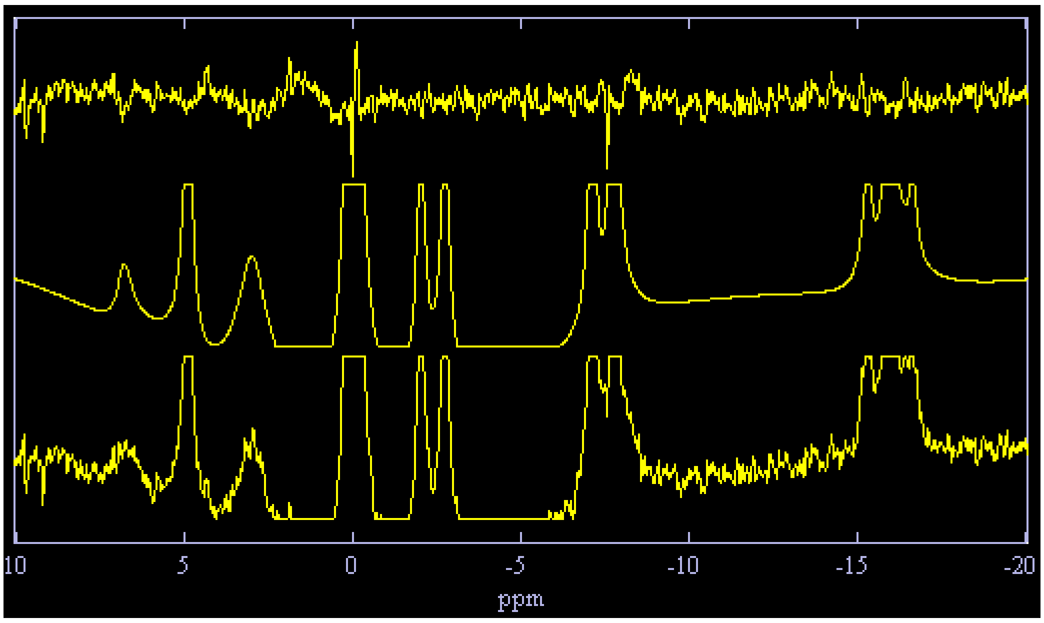


Figure 24. The spectral display has been scaled to the residual. The larger spectra are clipped.

You can only use scale to individual components if the latter are displayed. If you do, the minimum and maximum are taken over *all* individual components. For the other three spectra goes that you can set the scale to them, without them being on display.

5.2.1.5 peak numbers

This menu item allows you to toggle the display of the peak numbers in the multiple spectra display on or off. You can change the default setting using the customize preferences window - see section 3.2.1.3), the package default is on. In the SVD_1D function ,the peak numbers will disappear automatically when you display the singular value plot (see section 5.2.1.1), and will re-appear when you return to the normal display of the spectra.

5.2.1.6 spectrum labels

The spectrum labels are text bars on the left hand side of the stacked result spectra, indicating what those spectra are, such as original, reconstruction, individuals or residual (slightly different after Filter_HSVD) You can switch these on or off with this menu item. In Figure 22 you see that they are not displayed, because on a small 14" PC monitor they will start to overlap with the file and fit information table on the left (in fact, you can already see that the file and fit information table has started to overlap with the spectral plot on the right). On larger screens (or at higher pixel resolution) this effect does not occur, but in the current case it was useful to be able to switch the spectrum labels off. Even if the spectrum labels menu is set to on and the labels displayed (this is the default setting; change the default setting using the customize preferences window - see section 3.2.1.3), they will not appear on paper if you use the print menu, because the spectrum labels are so-called Matlab "uicontrols", and these do not get printed (see your Matlab user guide). The spectrum labels will disappear automatically when you display the singular value plot (see section 5.2.1.1) or when you switch the result info list off (see next section), and will re-appear when you return to the normal display of the spectra.

5.2.1.7 result info list

With this function you can toggle between the default display of the file and fit information table on the left and the spectral display on the right, and a display in which the stacked spectra occupy the entire result spectral window, see . This may be useful for presentation purposes.

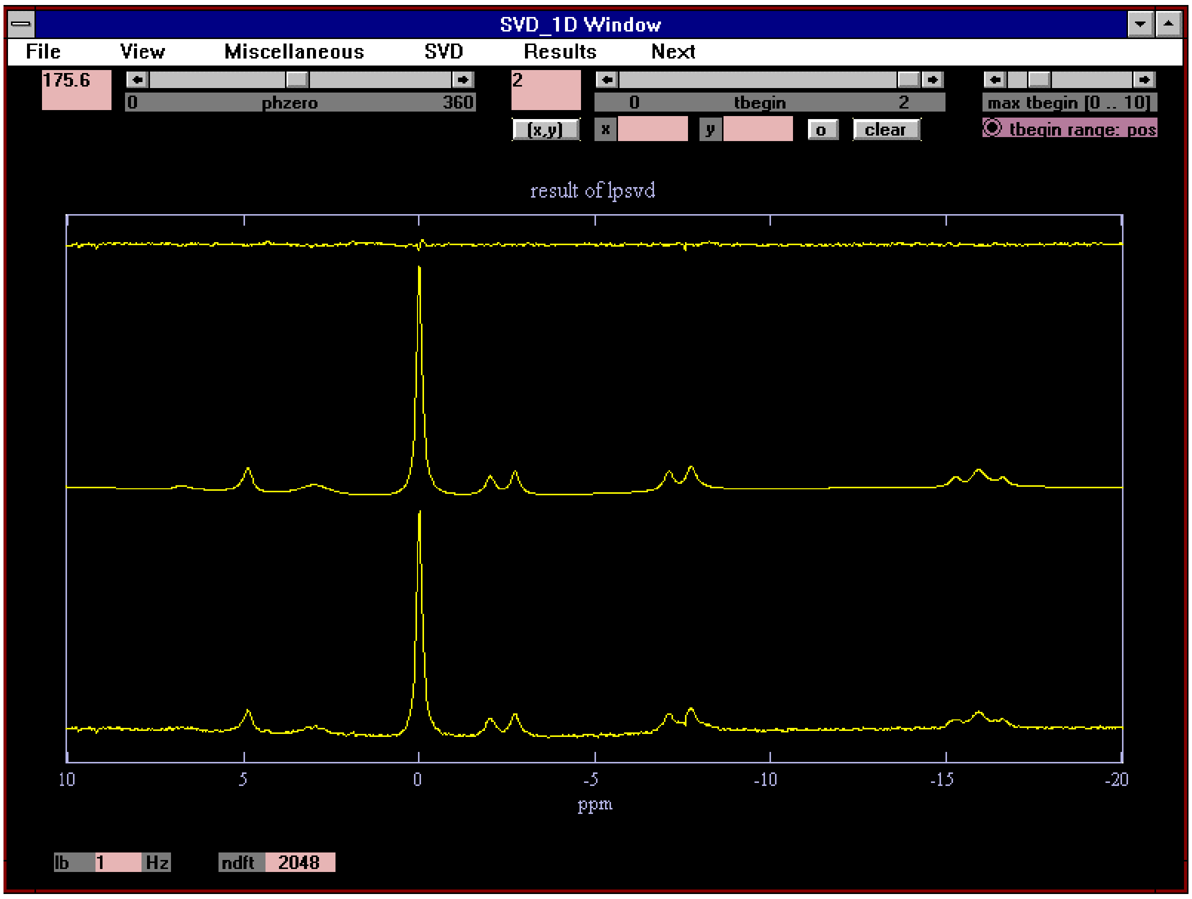


Figure 25. The result spectral window, with the result info list switched off, and the spectral display occupying the entire window.

For unknown reasons, spectral functions such as phasing, line broadening, number of FT points, or other view and display menu items, may have an ill effect when the spectra are displayed in this window-filling fashion. This is subject to further investigation. For the moment, it is strongly advised to carry out all your spectral display function in the normal display mode with the result info list set to on, and only in the end, just before you print or capture/snapshot, switch the result info list off.

5.2.1.8 *save signals*

In the SVD_1D and VARPRO functions, this function has two submenu items: reconstruction and residual. In the Filter_HSVD function it only has the submenu item reconstruction, because then the residual signal is saved to file anyway (since that is the filtered signal, `<name>_p.dat/mat`, see section 9.8). Selecting a submenu item brings up the Matlab file manager window, and you can save either the time domain signal reconstructed from the model function with the estimated parameters filled in (reconstruction), or the time domain signal containing the difference between the original signal and the reconstructed signal (residual) to a new set of MRUI data files `<name>.dat/mat`. A corresponding `<name>.gra` and `<name>.scl` file will also be produced, these are copied from the original data file.

5.2.2 The Next menu

The Next menu allows you to easily switch from one function to the next, without having to re-load the data file. From any function, you can take the current data file (whether this is the one you just loaded when you are still in the basic spectral window, the file you loaded on which you have just finished your parameter estimation, or a new file that has just resulted from some FID processing function) into one of the functions SETUP (chapter 7), ER Filter (section 8.3), Filter_HSVD (chapter 9), SVD_1D (chapter 10), Peakpick (chapter 11), Inputvp (chapter 12) or VARPRO (chapter 13).

If you loaded a batch, the Next menu is disabled (inactive), because it is then ambiguous which file to load into the next function, and the entire batch settings are not readily transferred between functions.

5.3 The FID processing window

An example of the FID processing window is displayed in Figure 26. In this example, the FID processing window was used for the Subtract function of the FID Maths menu, from the processing menu under the MRUI base window. Two FIDs are involved in the processing technique. As can be seen from the figure, one menu is headed Subtract; this menu contains the features specific to Subtract only (for more details see section 8.8). The other menus recur in all or most processing functions that use this FID processing window, and are discussed below. Notice that just like for the basic spectral window (see section 5.1), the menus only become enabled (active) after a file or a batch of files has been loaded.

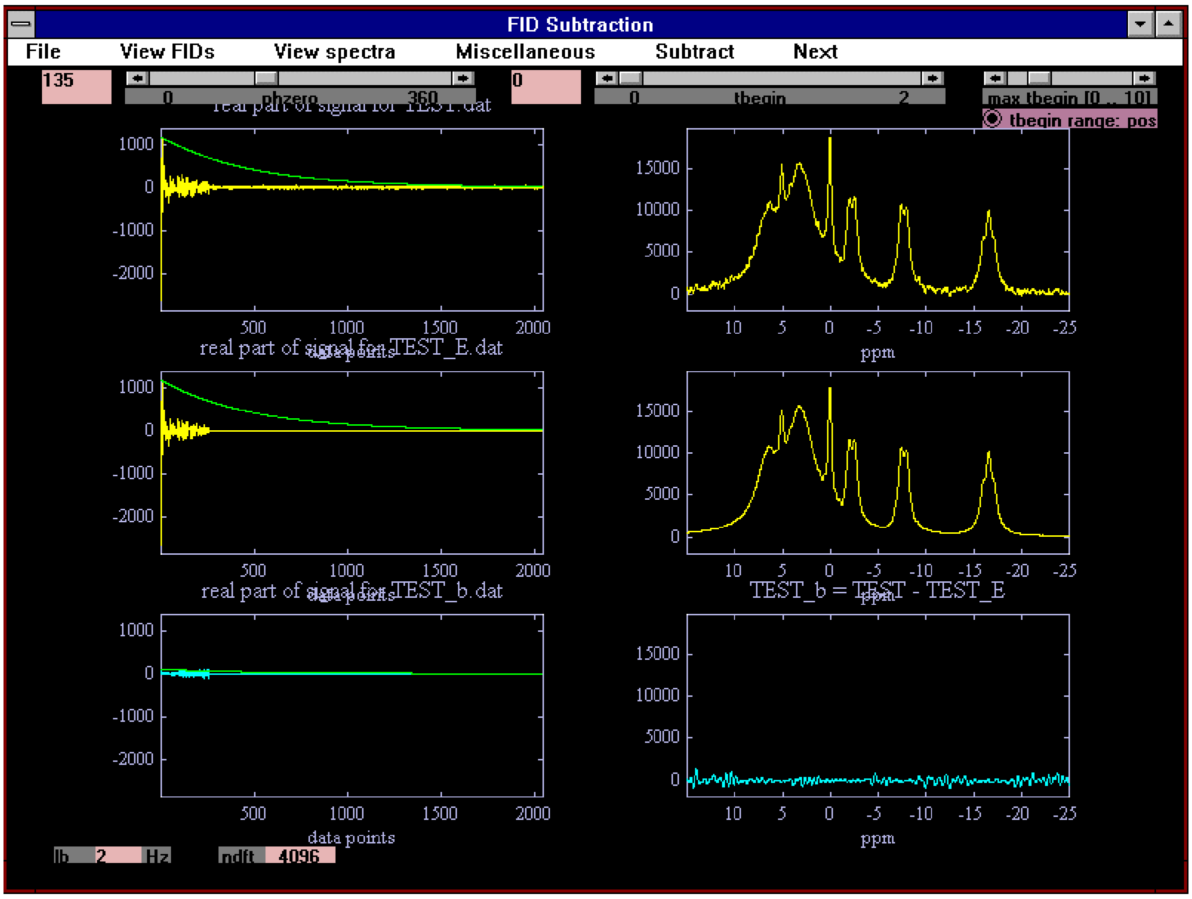


Figure 26. The FID processing window when subtracting two FIDs from each other.

Figure 26 illustrates nicely how for the FID processing window, the FIDs are displayed on the left, and the corresponding spectra on the right. This is because the processing techniques are mostly targeted at the raw FIDs, and the effect on the latter should be visualized. The original signal is displayed at the top row (in yellow), on the second row the second signal (in yellow; if another signal is involved in the particular processing technique), and the result of the processing on the bottom row (in cyan). If the FID processing technique involves only one signal, the result of the processing is displayed on the second row (in cyan), with the bottom row empty. The FIDs and spectra on the various rows are always scaled towards the largest one, so that intensities can be compared between them. The only exception is the second signal in the Divide function, which is only included for its phase term (the intensity of that signal can have arbitrarily large values), and therefore scaled individually.

5.3.1 The File menu

The File menu works exactly as described in section 5.1.1, except that there is no load result menu item (not relevant, results of processing techniques can be re-loaded simply by loading the resulting signal in the basic spectral window for any function) and that there is no save ascii plot menu item (again, that can be obtained after loading the signals in the basic spectral window for any function).

5.3.2 The View FIDs menu

The zoom in, zoom out, scale up, scale down and reset functions behave exactly as explained in sections 5.1.2.1, 5.1.2.2, 5.1.2.4, 5.1.2.5 and 5.1.2.10, respectively, except that they only influence the FID display in the left hand column. These menu items do not change the contents of the `default.gra` and `<name>.gra` file (see section 5.1.2).

5.3.3 The View spectra menu

The zoom in, zoom out, scale up, scale down, auto scale and reset functions behave exactly as explained in sections 5.1.2.1, 5.1.2.2, 5.1.2.4, 5.1.2.5, 5.1.2.6 and 5.1.2.10, respectively, except that they only influence the spectra display in the right hand column. These menu items change the contents of the `default.gra` and `<name>.gra` file as explained in the aforementioned sections. The x- and y-axis labels are always displayed in the FID processing window.

5.3.4 The Miscellaneous menu

The axis units and normal/reverse menu items work exactly as described in sections 5.1.3.1 and 5.1.3.3, respectively, but they only affect the spectra in the right hand column. There is no real/imaginary menu item; the FID processing window displays the real parts of the FIDs and spectra only. The automatic scaling and batch results menu items work exactly as described in sections 5.1.3.4 and 5.1.3.5, respectively.

5.3.5 The Next menu

The Next menu works exactly as described in section 5.2.2.

5.3.6 Phasing

The phzero and tbegin sliders and edit boxes behave identically to the ones in the spectral window, see sections 5.1.4.1 and 5.1.4.2, respectively, but they only affect the spectra in the right hand column. The `default.gra` and `<name>.gra` file are updated as explained in the aforementioned sections.

5.3.7 line broadening

The line broadening lb edit box works exactly as described in section 5.1.6, with the addition that in the FID processing window, the actual line broadening apodization function is drawn in the FID displays, see e.g. Figure 26. The Lorentzian apodization function is drawn in green, and although its maximum intensity is of course always 1, it is drawn with the first point at the maximum intensity of the FID to allow easier visual interpretation. The additional plotting of the apodization function in the FID display allows you to check how much of the FID is suppressed.

5.3.8 number of points in FT spectrum

The ndft edit box works exactly as described in section 5.1.7.

6. The Simulation Tool

To synthesize your own signals, use the Simulate function under the processing menu in the MRUI base window. It brings up the basic spectral window (see section 5.1), titled FID Simulation Window. Initially, all menus and sliders will be disabled (inactive), except for the load and quit menu items under the File menu. The starting point of simulating your own signals is an ASCII file containing some information about the signal-to-be, and a list of all its parameters (frequencies, damping factors, amplitudes and phases). The default simulation parameter file is called `simul.par`. You will find a template in the `$HOME/matlab/work_dir/TEST` directory. An example of a `simul.par` file is illustrated in Figure 27 below.

```
name of signal file:              sim31p.dat
number of data points in signal:        1024
step size of signal (ms):              0.500
begin time of signal (ms):             0.000
number of peaks:                          11
gaussian fraction in lineshape:            0
noise standard deviation:                  0
monte carlo simulation:                    0

    FRE        DAM          AMP        PHASE
    (Hz)       (Hz)        (a.u.)       (deg)

  330.000   -200.000   0.500000e+03        0.0
  290.000    -25.000   0.600000e+02        0.0
  240.000   -285.714   0.140000e+04        0.0
  160.000    -25.000   0.150000e+03        0.0
  108.000    -50.000   0.150000e+03        0.0
   92.000    -50.000   0.150000e+03        0.0
  -32.000    -50.000   0.150000e+03        0.0
  -48.000    -50.000   0.150000e+03        0.0
 -254.000    -50.000   0.750000e+02        0.0
 -270.000    -50.000   0.150000e+03        0.0
 -286.000    -50.000   0.750000e+02        0.0
```

Figure 27. The simul.par file containing details and model function parameter to synthesize a simulation signal.

You can use your preferred text editor to manually edit a `simul.par` file and to enter your desired number of peaks, and their parameters. Remember that this is where you start. You can always interactively change the synthetic signal once the parameter file has been loaded. When you edit the `simul.par` file, make sure that you do not change the format of the file, as file is expected to have this exact structure when it is loaded. Notice that in the simul.par file, the model function parameters are expressed for the simulated time domain signal: frequencies in Hz and damping factors in Hz and as negative values. You must take this into account when preparing your signal. Note that you can distort your simulation signal with a zero order phase by giving all phases in the table the same offset.

Instead of starting with from scratch with a `simul.par` file, you can also use results of other functions to create a simulation signal. In the next section we will explain how you can use the parameters in a Peakpick file, in an SVD_1D result file or in a VARPRO result file to build a simulated FID.

6.1 The File load menu

As you can see, the File load menu in the Simulate spectral window is rather different from the general File load menu as described in section 5.1.1.1. For Simulate, you do not load a data file, but an ASCII file containing parameters with which to synthesize a new signal. These parameters can be read from the default file `simul.par`, or from result files of other functions: select <name>.sim to load the results of a previous simulation (see section 6.2.10), select <name>.scr to load the results of an SVD_1D fit (see section 10.11), select <name>.pkp to load the results of a Peakpick session (see section 11.8), and select <name>.gol to load the

results of a VARPRO fit (see section 13.7). In that case the parameters from such a previous function are read in and used to constitute a noise-free time domain signal. When loading a `<name>.***` result file for simulation parameters, Simulate searches for a corresponding `<name>.scl` file with the original data file, and if it is found, the amplitudes read from the `<name>.***` result file are scaled back according to their original intensities in the measured FID. This ensures that the signal you synthesize has intensities comparable to the original signals (e.g. necessary if you intend to sum or subtract in the future). If you choose to load one of these result files, you will be prompted by the file manager window to select one of the listed files (`*.sim`, `*.scr`, `*.pkp` or `*.gol`, respectively). You can browse through the directory structure if you need to. See sections 4.6 and 5.1.1.1.1 for more information on using the file manager window.
If you wish to load the simulation parameters from the `simul.par` file, then that `simul.par` file is expected to be found in your current directory; the file manager is not activated for this menu item. Hence it is important to change to the directory containing your `simul.par` file first, just like you would need to do when loading a batch of files in any other function, as explained in section 5.1.1.1.2. You can select the directory containing your `simul.par` file using the select dir function from the Matlab menu under the MRUI base window, see section 3.2.2.3.

Once you have loaded a selected file, or in this example the above `simul.par` file, the Simulate program will create a noiseless time domain model function (lineshape according to Gaussian fraction in the file), using the number of data points, begin time, sample time step and number of peaks as listed in the file, and insert the values of the parameters. If you edited a value of the standard deviation of the noise level (in the same units as the amplitude parameters) in the `simul.par` file, Simulate will generate a white noise realization with zero mean and the listed standard deviation, of the same length as the noiseless simulation signal, and add that noise realization to that simulation signal. Next it will display the Fourier transform spectrum in the basic spectral window (see Figure 28), and enable all menus and sliders. If you had a standard deviation for the noise level, the noise slider will be immediately updated, see also section 6.2.8. The peaks are numbered in the order they were read from the parameter file.

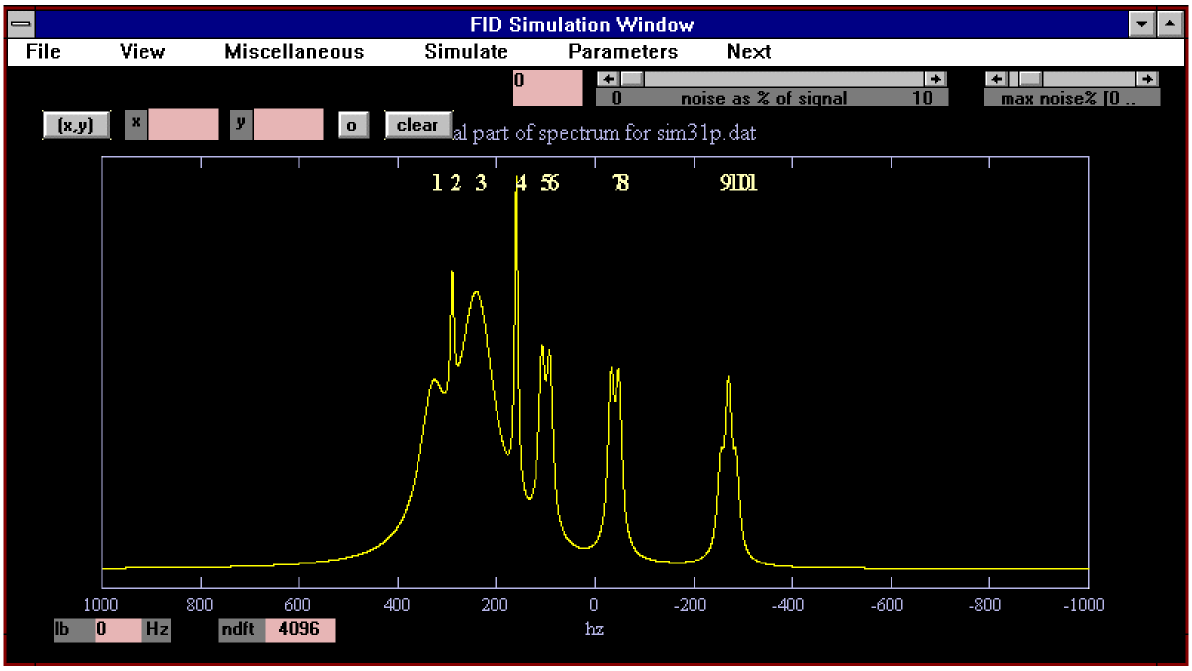


Figure 28. The spectral window for the Simulation tool displaying the FT spectrum of a noiseless simulation signal synthesized using the parameters from the simul.par file in Figure 27.

Now you can modify your synthetic signal according to your requirements. From the noiseless spectrum you can see that the simulation parameters from the `simul.par` file in Figure 27 were meant to mimic an *in vivo* ^{31}P MRS signal from human brain. You can also use the View and Miscellaneous menus, and the lb and ndft edit boxes, as described in section 5.1, but remember that these only affect the visual spectral display, and not the synthesized signal.

6.2 The Simulate menu

6.2.1 add peak

With this function you can add an extra peak to the simulated signal. If you select add peak, you are prompted to enter for that new peak its frequency in Hz, damping factor in Hz (negative value!), amplitude in arbitrary units and phase in degrees. You enter your desired values in the displayed Edit Window, see for an example Figure 16. In the current example peak 12 was added with a frequency of 650 Hz, damping factor of -90 Hz, amplitude of 100 a.u. and phase of 0 degrees (mimicking a reference signal). The result of all modifications mentioned for this example is displayed in Figure 29. The displayed spectrum will be updated.

6.2.2 remove peak

With this function you can remove any of the peaks from the simulated signal. If you select remove peak, you will be prompted by an Edit Window in which you can type the number of the peak that should disappear. The displayed spectrum will be updated, and the numbers of the remaining peaks will be updated accordingly.

6.2.3 add signal

With this function you can add an entire signal to your simulated signal. If you select add signal, the file manager window appears, listing the `*.mat` files. You can browse through the directories to find the data file you wish to add to your simulated signal. See sections 4.6 and 5.1.1.1.1 for more details on using the file manager window. This function may be particularly useful if you wish to create simulation signals with truly experimental distortions by adding a data file containing a measured signal containing only baseline humps (e.g. obtained using a metabolite nulling experiment[6]). You can also use it to add signals that resulted from previous Simulate sessions. However, be aware that the sample time step and begin time must match between the simulated signal and the signal you wish to add. If the numbers of data points do not match, the smaller of the two is used, and a warning message displayed. You should also take notice of scaling factors (see section 4.2) involved with the signal you are adding, as the loaded signal is first re-scaled to its original intensities, using the parameters read from the `<name>.scl` file, before it gets added to your synthesized FID. The displayed spectrum will be updated.

6.2.4 lineshape

With the lineshape menu item, you can toggle between Lorentzian or Gaussian lines in the simulated model function. If you set the Gaussian fraction in the `simul.par` file to 1, the lines will be Gaussian, but otherwise the default is to have a Lorentzian model lineshape. The displayed spectrum will be updated.

6.2.5 no. of data points

With this menu item you can change the number of data points in the simulated signal by typing it in an Edit Window. When you change `ndp`, the value in the submenu item will be updated. The number of data points you enter is checked against the absolute maximum number of data points allowed in the MRUI software. This maximum value protects the Fortran read and write routines in the signal processing and conversion programs from overflowing the data arrays, which would incur uncontrolled loss of data points and lead to aberrant results. The default maximum in the MRUI package is set at 16384, but you can change this value in the maxndp submenu item of the SETUP menu in the customize fitting parameters window. However, you should do this *after* you have increased the maximum array sizes in the corresponding Fortran routines, see Appendix E. The default value is for `ndp` taken from the parameter file. In this example the number of data points was reduced to 512 (see result in Figure 29). The displayed spectrum will be updated.

6.2.6 time step

With this menu item you can change the sample time step of the simulated signal by typing it in an Edit Window. The default value is taken from the parameter file. The time step value is expressed in milliseconds, and is the reciprocal of the spectral width in kHz. The displayed spectrum will be updated.

6.2.7 begin time

With this menu item you can change the begin time (delay time, time difference between the true time origin `t=0` and the first data point) of the simulated signal by typing it in an Edit Window. The begin time value is expressed in milliseconds. In this example the begin time was changed to 1 ms (see result in Figure 29). The displayed spectrum will be updated, and with increasing begin time you will notice the convolution of the displayed spectrum with a sinc function which is the Fourier transform of the step function consisting of value 0 for the missing point in the delay time period and of value 1 for the available data points.

6.2.8 noise level

With the noise level menu item you can set the standard deviation (in the same arbitrary units as the amplitude values) of the noise realization that will be added to your noiseless simulated time domain signal, by typing it in an Edit Window. The noise realization is also complex (via two different real-valued noise realizations, one added to the real part of the simulated signal, and the other to its imaginary part), with zero mean. The displayed spectrum will be updated, so that you can judge the affect of the added noise, and if necessary change its value again.
Another way to change the noise level, is by using the noise as % of signal slider or its edit box, in the upper right corner of the spectral window. They are on the positions where in other functions the tbegin slider and its edit box reside (see section 5.1.4.2), and you operate both identically (see section also section 3.2.1.2 for details on using sliders). As opposed to the noise level menu item, this slider does not control the value of the noise standard deviation directly as an absolute value, but as a percentage of the maximum signal intensity of the noiseless time domain signal. So you could for instance, using this slider, add a noise realization with a standard deviation of 10% of the noiseless simulation signal. The auxiliary max noise % slider on the right can be used to change the maximum range of the noise as % of signal slider, just like the max tbegin slider does for the tbegin slider (see section 5.1.4.2). The default maximum percentage on the noise as % of signal slider is 10, but you can set this default according to your own preference using the customize preferences menu from the Matlab menu under the MRUI base window. See also Figure 5 in section 3.2.1.3.
Notice that when you change the standard deviation either via the noise level menu item, or via the noise as % of signal slider or edit box, both the menu item and the slider and its edit box are updated. The default value in the noise level menu item is that read from the `simul.par` file (or from the other parameter files if a relevant value can be found). The default value will immediately be converted to a percentage value for the slider and its edit box as well.
In the current example (result in Figure 29), the noise level menu item was used to add a noise realization with a standard deviation of 10 units.

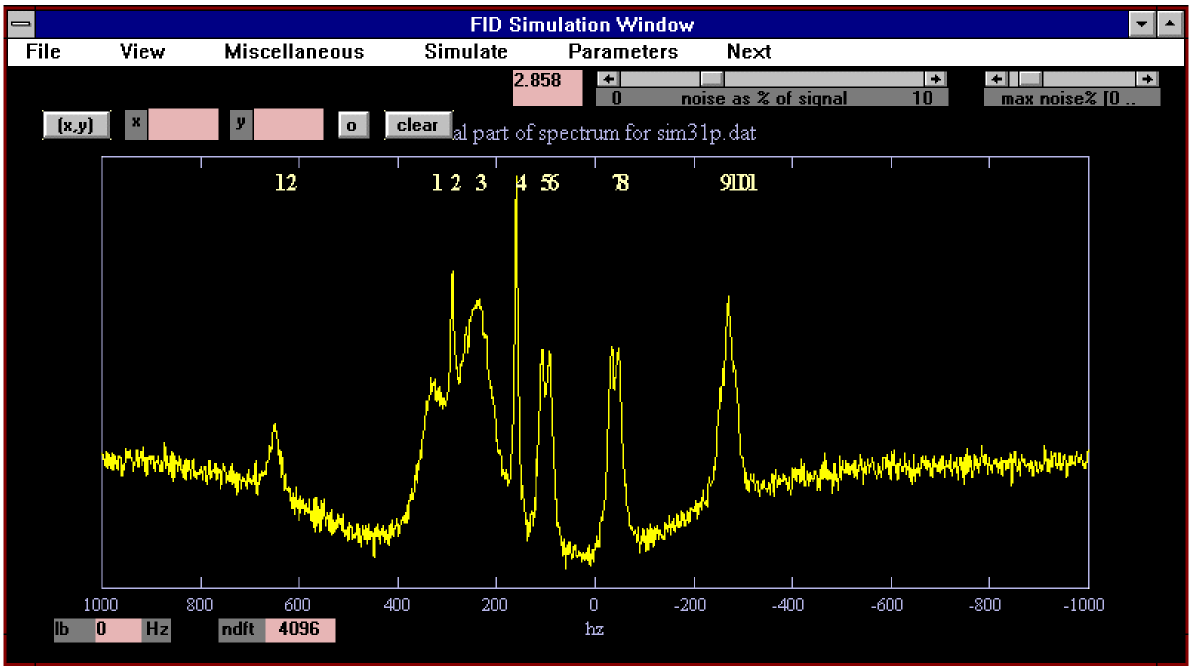


Figure 29. The Simulation Window after modifying the noiseless simulated FID of Figure 28, by adding an extra peak (12), changing the number of data points and the begin time, and adding noise with a standard deviation of 10 intensity units.

6.2.9 Monte Carlo batch

With this function you can create a Monte Carlo batch. This is a large series of signals consisting all of the same noiseless simulation signal, but each with a different noise realization (though of identical mean and standard deviation) added to it. This is a good way to test the consistency of a new method for example[3]. To exclude the possibility of an accidental 'hit" or 'miss', and to enable statistical statements to be made about the performance of the method, a large number N_{MC} of noise realizations should be generated, all with identical mean and standard deviation. When adding each to a noiseless model function, N_{MC} noisy simulation signals are obtained, each containing the same true noiseless signal, but perturbed by a different noise realization. Next, all N_{MC} noisy simulation signals are processed identically with the parameter estimation method, so that N_{MC} sets of parameter estimates are produced. Each parameter is then averaged over its N_{MC} estimates, and the averaged values can be compared to the true parameter values used to create the noiseless model function. For an accurate parameter estimation method there should be no *bias*: for every parameter, the difference between its true value and the mean over its N_{MC} estimates should be smaller than the population standard deviation over the N_{MC} estimates. For maximum likelihood estimators (which the VARPRO method in principle is), theoretical lower limits, the so-called Cramér Rao lower bounds, for the standard deviations of the parameter estimates in the model function, depending on the values of the parameters and the standard deviations of the noise, can be calculated[3]. It can thus be tested in a Monte Carlo study whether the population standard deviations over the N_{MC} parameter estimates reach the theoretical lower limits. This would prove that for the signals used, the method under investigation gives most accurate parameter estimates.
You can toggle the Monte Carlo batch menu item between on and off, and if you switch it on, you can set the number of noise realizations (i.e. the total number of simulation signals produced; N_{MC}), by typing it in an Edit Window. When the menu item is set to on, it will also display the value of N_{MC}. The default value for N_{MC} and the corresponding menu item setting are taken from the `simul.par` file. When loading another type of parameter file (see section 6.1), the menu item will be switched off, and N_{MC}=0, i.e. only one simulation signal will be produced when saving.
If you set a Monte Carlo batch with this function, a numbered batch series of data files will be produced when saving (see the next section), such that it can immediately be loaded as a batch in any of the other MRUI functions.

6.2.10 save FID

This function has a different response depending on the setting for Monte Carlo batch (see previous section). If the latter is switched off, only a single simulation signal needs to be saved to file. In this case the file manager window is displayed, prompting you to type the name of the data file under which the simulation signal needs to be stored. You can browse through the directory tree, and then type the *full* file name (i.e. including the `.dat` extension) in the File Name field (selection box on UNIX systems), or select an existing `<name>.dat` file if you want to overwrite it. Click on OK (Done on UNIX systems) or press `[Enter]`. If you click on Cancel, you will *not* abort the FID saving function, but the signal will be saved under the file name found in the ASCII parameter file (e.g. `simul.par`, see Figure 27).
If you had prepared the settings for the generation of a Monte Carlo batch, an Edit Window will come up instead of the file manager window, asking you for the general file name base (as in Figure 12). A numbered batch of files will then be stored on disk, each file name containing the general file name base, followed by a number ranging from 1 to N_{MC}, and the usual `.dat` and `.mat` extensions. This batch can directly be loaded into any of the other MRUI functions that allow batch processing, as explained in sections 4.4.1 and 5.1.1.1.2.
After having indicated the name(s) under which the signal(s) is/are to be saved, a file manager window will come up (again), asking you for a name of an ASCII file, having the format of the `simul.par` file (see Figure 27), to store all simulation parameters in. The suggested extension is `.sim`, so that you can easily load it later on using the File load menu in the Simulation tool (see section 6.1). Cancel finishes the save FID function without creating the simulation parameter file.
As for all data files upon conversion (see section 4.2), the simulation signal is normalized when it is saved to the data file, and a corresponding `<name>.scl` file is produced.

6.3 The Parameters menu

Next to the Simulate menu, you will find the Parameters menu, which contains as many submenus as there are peaks in your simulated signal. Then each submenu for each peak contains four submenu functions: fre, dam, amp and pha. With these, you can change for the corresponding signal component its values for the frequency in Hz, the time domain damping factor (negative value) in Hz, the amplitude in arbitrary units and the phase in degrees. You do this by typing the new value in an Edit Window. The current values of the parameters are always listed in the corresponding menu items. The noiseless signal and hence the displayed spectrum will immediately be updated when you change any of the component parameters. You can change as many or as few individual parameters as you wish.

6.4 Files produced

When having saved the simulated signal(s) and you are finished using the Simulate function, the following files will have been produced (barring any exceptions in which you used e.g. the Cancel button in the file manager window):

single signal (N_{MC}=0)	Monte Carlo batch
`<name>.dat`	`<name>1.dat`, `<name>2.dat`, `...`, `<name><`N_{MC}`>.dat`
`<name>.mat`	`<name>1.mat`, `<name>2.mat`, `...`, `<name><`N_{MC}`>.mat`
`<name>.scl`	`<name>1.scl`, `<name>2.scl`, `...`, `<name><`N_{MC}`>.scl`
	`<name>.sim`

`default.xpt` (temporary ASCII auxiliary file, see chapter 7)

7. Database - SETUP: experimental

In section 4.3 you have read how at the time of spectrometer file conversion, the most important experimental parameters are ideally directly obtained from the spectrometer file, in order to allow a spectral display in the MRUI interface according to the spectral display on your spectrometer software. However, not in all cases can all necessary experimental parameters (see the table in section 4.3) be read from the spectrometer file. This may be because the conversion routine for this type of spectrometer files is not fully optimized, or because that type of spectrometer file does not contain all the required parameters. However, in all cases, an ASCII auxiliary file, called `default.xpt`, is written to the directory in which the converted file(s) is/are stored. For parameters that could not be read from the spectrometer file, default values are written to this file (see the table in section 4.3). Remember that the default values taken by MRUI can be changed according to your preferences using the SETUP menu from the Customize fitting parameters function (see section 3.2.1.4). To check whether the experimental parameter settings are correct for your converted spectrometer file, and to set any missing (default) values to your preferred values, use the SETUP: experimental function from the Database menu under the MRUI base window. This function also allows you to set graphical parameters (phasing, line broadening, zooming, etc.), which are then written to the `default.gra` and `<name>.gra` files, so that everything is properly set before you start analyzing your data (see section 5.1.2).
It is important to realize that you have to change these settings *only once* in any directory. The SETUP: experimental function stores the experimental parameters in the ASCII file `default.xpt` (see also section 4.3), with no analogue that belongs to specific data files. Hence the changes that you make to `default.xpt` once, are preserved for all data files in that directory. When you do a new spectrometer file conversion in that same directory, only the experimental parameters that could be read from the spectrometer file are overwritten in the `default.xpt` file. The parameters that could not be read from the spectrometer file remain unchanged in the `default.xpt` file. Thus it is good policy to use the SETUP: experimental function in each directory immediately after your first conversion (see also section 3.3), after which you can carry out all your subsequent data analysis in that directory without have to re-do the SETUP.

Make sure you store data files of the same experiment type in one directory. I.e. data files of the same nucleus, using the same transmitter frequency, and with the same ppm reference, etc. If you don't, you will end up having to re-do the SETUP over and over again when analyzing different data files in that directory. The idea of the SETUP: experimental function and the categorized data storage approach is that you get data files of one particular type of experiment together in one directory, so that data analysis after using SETUP once requires minimum effort.

The SETUP: experimental function is organized in the basic spectral window, with all functions and menus as explained in section 5.1. Figure 30 shows the basic spectral window for the SETUP function, after having loaded the example data file (see section 3.3) `ischr.dat`. Notice that the basic spectral window for SETUP displays a few extra information features: the ppm offset in terms of kHz (top left), the number of data points in the data file (bottom) and the spectral width in kHz (bottom). You see that with the default values for a number of parameters (see also section 4.6), the spectral display is not completely ideal, and it needs some modification in order to please the spectroscopist.

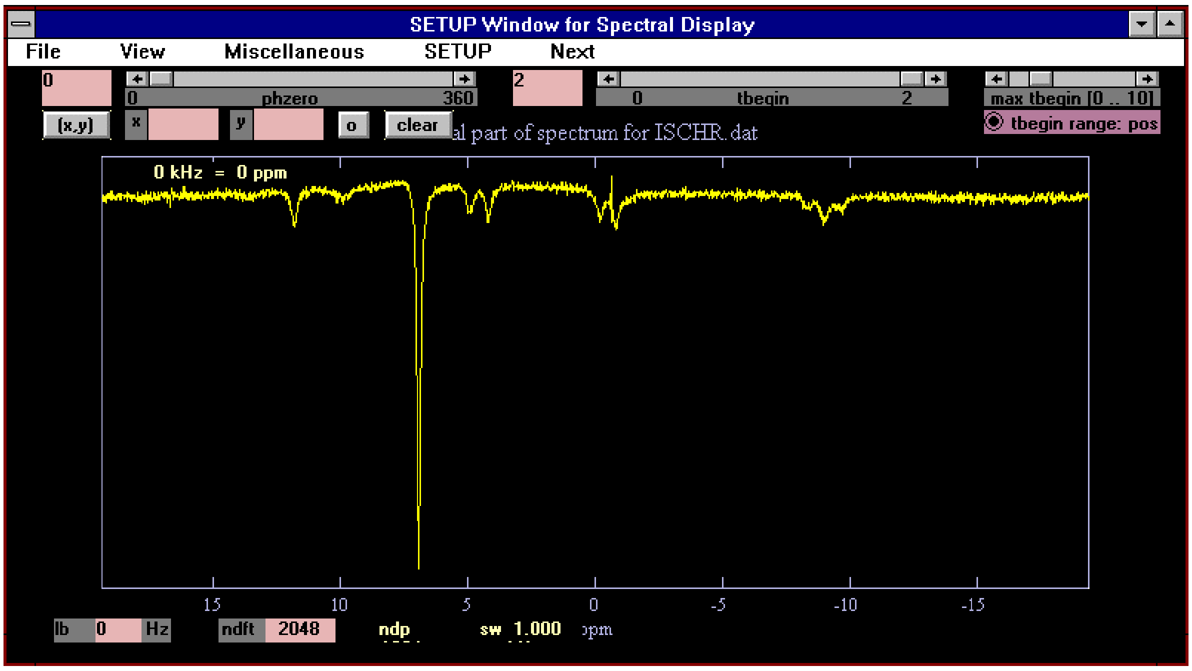


Figure 30. The basic spectral window for the SETUP: experimental function. With some experimental parameters defaulted, the spectral display needs some modification in order to conform to the spectrometer software display.

First of all, the spectrum is zero order phased (as explained in section 5.1.4.1). First order phasing is in this case not necessary, because the SMIS conversion routine is able to read the value for the spectrometer dead time from the spectrometer file, see section 4.6. A line broadening of 1 Hz is applied (see section 5.1.6), so that it starts to look like a familiar *in vivo* ^{31}P MR spectrum from muscle. The final results are displayed in Figure 31.

7.1 The SETUP menu

The SETUP menu contains submenus for the experimental parameters that can be changed using this function, plus one submenu to write the changes to file.

7.1.1 nucleus

With this function you can indicate the measured nucleus. It has no direct effect on the parameter estimation as such, but it determines whether certain options need to be enabled or disabled. For instance, the calculate pH menu (see sections 10.10,13.6) is only enabled if the nucleus is set to P-31. You can select between H-1 (proton), P-31 (phosphorus), F-19 (fluorine) or C-13 (carbon), or you can select other, and then click on the empty submenu, after which you can type a character notation for your nucleus of interest. In this example the nucleus submenu item was already correctly set to P-31, because the SMIS conversion routine is able to read the type of nucleus from the spectrometer file (see section 4.6). The package default would have been H-1 (see the table in section 4.3), but you can set it according to your preferences using the SETUP menu in the Customize fitting parameters function (see section 3.2.1.4). Changing the setting for nucleus does *not* affect the value of B0 (section 7.1.2) or transmitter freq (section 7.1.3).

7.1.2 B0

This submenu determines the strength of the static magnetic field in Tesla. This is currently however a redundant parameter, and does not affect any of the other parameters or the spectral display. It has been added for completeness and may receive some purpose in the future. You can select between 1.5 T, 2 T, 2.35 T, 4.7 T or 7.1 T, or you can select other, and then click on the empty submenu, after which you can type your specific magnetic field strength. In this example the B0 submenu item was already correctly set to 1.5 T, because the

SMIS conversion routine defaults to that value (see section 4.6). The package default would also have been 1.5 T (see the table in section 4.3), but you can set it according to your preferences using the SETUP menu in the Customize fitting parameters function (see section 3.2.1.4). Changing the value of B0 does *not* affect the setting for nucleus (section 7.1.1) or the value of transmitter freq (section 7.1.3).

7.1.3 transmitter freq

With this menu item you can modify the value for the transmitter frequency in MHz, by typing a new value in the Edit Window after you have selected the submenu containing the current value. The transmitter frequency *is* an important experimental parameter, as it directly determines the values of the chemical shifts on the axis. The spectral display will be immediately updated. In this example the transmitter freq was already correctly set to 25.79 MHz, because the SMIS conversion routine is able to read its value from the spectrometer file (see section 4.6). The package default would have been 63.86 MHz (see the table in section 4.3), but you can set it according to your preferences using the SETUP menu in the Customize fitting parameters function (see section 3.2.1.4). Changing the value of transmitter freq does *not* affect the setting for nucleus (section 7.1.1) or the value of B0 (section 7.1.2).

7.1.4 axis direction

This menu item has been added as a sort of emergency way out. In some cases, after converting a spectrometer file, the spectrum may be displayed the wrong way round. Not in terms of the normal/reverse functions (see section 5.1.3.3), but with peaks really residing at the wrong frequencies or chemical shifts. For instance, with ^{1}H MRS at 1.5 T, NAA resonates at about -170 Hz or 2.05 ppm. Imagine your surprise if your basic spectral window would display NAA at 170 Hz, or 7.35 ppm! This may in fact occur, as on some spectrometers the reference direction of resonance frequencies is inverted. For spectrometers for which this is known, the conversion routines automatically correct for this. However, it may still happen that you are presented with a spectrum that is placed on the frequency/chemical shift axis in the wrong direction. In these cases you can use the axis direction menu item and select the wrong! submenu. This will mirror the spectrum around 0 Hz, relative to the axis, and then display the new combination in the direction governed by the normal/reverse setting (see section 5.1.3.3). Every subsequent selection of the wrong! submenu will once more mirror the spectrum around 0 Hz, relative the axis. The OK submenu is in fact redundant, as its meaning is that everything is displayed as it should be, with no modification required. Nothing happens when you select it, but it was added to indicate the meaning of the two.

7.1.5 ppm reference

With this function you can set the ppm reference (or 'offset') for the chemical shift axis, in terms of true frequencies in kHz. You select a true frequency (position on the x-axis) in kHz, either by clicking with the left mouse button in the spectrum (when you select click peak), or by typing the value in an Edit Window (when you select type value). Next you enter the chemical shift in ppm for that position in an Edit Window. For this example of the *in vivo* ^{31}P MR spectrum from muscle, it would be most appropriate to set the phosphocreatine (PCr, the large peak in Figure 30) peak at 0 ppm. Select ppm reference and click peak, and a message will appear in the lower right corner of the spectral window, telling you to "click on center/top of peak for chemical shift reference", and the mouse cursor changes to a cross-hair. Position the cross-hair where you would want your ppm reference to be, and press the left mouse button. An Edit Window will pop up in which you can type the value for the ppm reference (in this case 0, but for ^{1}H MRS you might want to type 4.7 ppm at the water frequency, etc.). The spectral display is immediately updated and the spectrum (with axis in ppm) will now display the metabolite peaks at their familiar chemical shifts. The result is displayed in Figure 31.

7.1.6 write to file

This menu item has to submenus: default and database. When you select the default submenu, all modifications are written to the `default.xpt` file. This function is more or less redundant, because every time you change any of the experimental parameters through one of the menu items described above, the `default.xpt` file is automatically updated. You can select database to store all experimental parameters in an ASCII file of the same format as `default.xpt`, but you can give it your own name, and store it in any directory using Matlab's file manager window (see e.g. section 4.6). You can do this for example to create a dedicated `default.xpt` file for ^{31}P MRS data, call it e.g. `p31set.xpt`, and store it in a general directory (e.g. the `work_dir` directory). You can then later copy it to a new data directory to circumvent the SETUP: experimental function altogether. See also section 7.3, describing the database functions available for the auxiliary ASCII files.

Finally, the spectrum is zoomed in between 10 and -20 ppm. Figure 31 displays the basic spectral window for the SETUP: experimental function after the modifications described above.

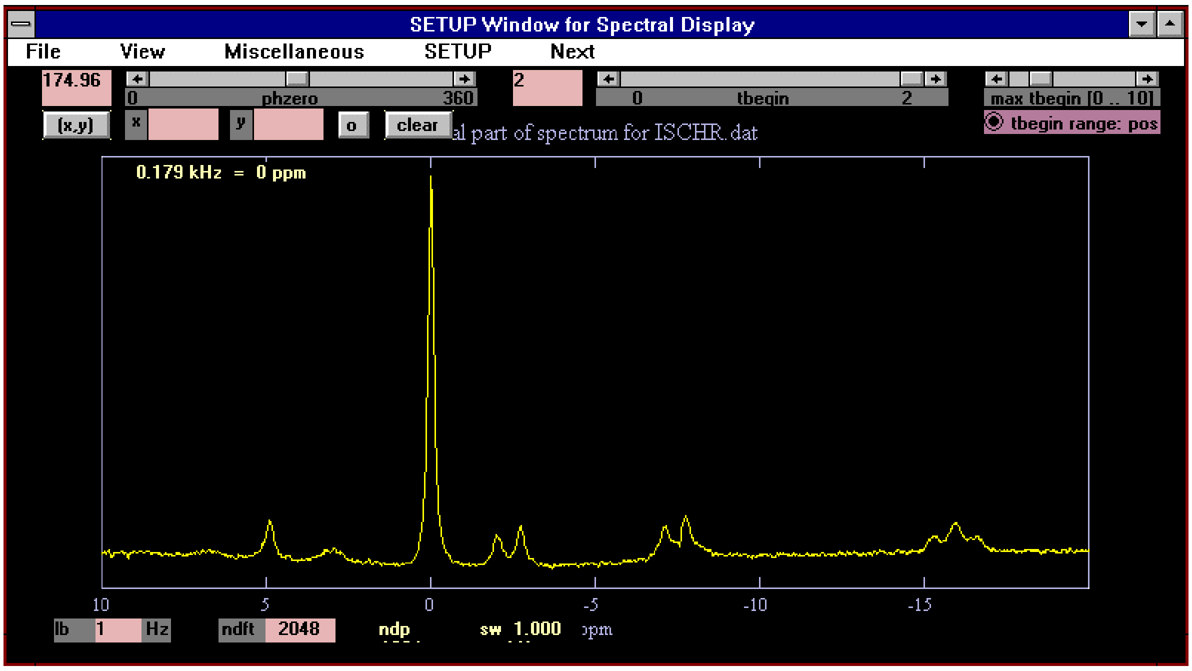


Figure 31. The basic spectral window for the SETUP: experimental function after the experimental parameters have been properly adjusted.

7.2 Files produced

Once you are finished using the SETUP: experimental function, the following files have been produced or updated:

```
default.xpt
default.gra
<name>.gra
```

If you selected write to file - database, then there will be a copy of the `default.xpt` file under the name you gave, and in the directory you indicated.

7.3 Database: saving and loading parameter files

The five remaining submenus below the Database menu serve to store your auxiliary ASCII files for re-use. If you end up using some of the related functions over and over again in much the same manner, you can use these Database functions to store the result of one such function under a general name and in a general directory. Later, when you start new analyses on similar data files, perhaps in different data directories, you can load these parameter files through the Database menu, and store them in your current data directory. This avoids having to re-do all the processing steps automatically saved in the parameter files last time. This is equally valid for the settings of the View and Miscellaneous menu in the spectral window (stored in the `default.gra` file, see section 5.1.2), the SETUP: experimental parameters (stored in the `default.xpt` file, see section 7.1.6), the starting values from the Peakpick function (stored in `peakpick.par`, see section 11.8) and the imposed prior knowledge from the Inputvp function (stored in `inputvp.ans` and `link.ans`, see section 12.6) and the Inputnv function (stored in `inputnv.ans`, see chapter 14). The five submenus peakpick file, inputvp file, inputnv file, default.xpt file and default.gra file all have a save and a load submenu. They all work analogously as described below, with minor differences as indicated.

7.3.1 save parameter file

The file manager window comes up, and you can browse through the directory structure for an appropriately general location (perhaps taking into consideration that other users on your system should also be able to access it!) to store your parameter file for re-use, and you can enter the name of the parameter file. For more details on the use of the file manager window for saving files, see section 4.6. The default file extensions are as follows:

Database submenu	default extension for saved parameter file
peakpick file	`*.pkp`
inputvp file	`*.inp`
inputnv file	`*.ipk`
default.xpt file	`*.xpt`
default.gra file	`*.gra`

You can type different extensions for the file you wish to save, but remember that when loading a parameter file through the Database menu (see next section), only the files with these default extension will be listed in the file manager window.
Suppose you choose to store your current default parameter file in the directory `my_dbs`, and you give it a name with base `p31set`, with the default extension suggested by the file manager window. In this case the following files would be copied and created:

Database submenu	file copied from current directory	to
peakpick file	`peakpick.par`	`my_dbs/p31set.pkp`
	`default.gra`	`my_dbs/p31set.gra`
inputvp file	`inputvp.ans`	`my_dbs/p31set.inp`
	`link.ans`	`my_dbs/p31set.lnk`
inputnv file	`inputnv.ans`	`my_dbs/p31set.ipk`
default.xpt file	`default.xpt`	`my_dbs/p31set.xpt`
default.gra file	`default.gra`	`my_dbs/p31set.gra`

Notice that when you save a Peakpick file, the `default.gra` file is also copied to ensure consistent display.

7.3.2 load parameter file

You can use this function to create a default parameter file in your current data directory, thus bypassing running a complete MRUI function. The default parameter files, default extensions, and files copied, are exactly the same as explained in the previous section, except of course that the copying goes the other way round. The Database load function is not necessarily restricted to files stored in general locations using the Database save function. Of course you can point your file manager window at files that have been automatically produced by the MRUI software in the course of a previous analysis. You can simply indicate a `<name>.gra` file (see section 5.1.2) in any data directory, and it will be copied to your current data directory under the name of `default.gra`. The same goes for loading a `<name>.xpt` file. When loading a Peakpick file, an Inputvp file or an Inputnv file however, a second file manager window will pop up, asking you for which data file in the current directory the loaded parameter file is to be used. Indicate your current data file, and the parameter files will also be stored in your current data directory as `<name>.pkp`, `<name>.inp` and `<name>.lnk`, or `<name>.ipk`, respectively.

8. The FID Maths library

In many publications on time domain analysis, it is stressed that one of the major advantages of fitting in the time domain, as compared to fitting in the frequency domain, is that the raw, measured data are immediately used in the fit, without having to carry out the Fourier transform and all its related preprocessing functions (windowing, apodization, phasing, baseline correction, etc.). In this manual it has been emphasized that although the fitting is being done in the time domain, the Fourier transform MR spectrum is still an invaluable tool to the spectroscopist, in order to properly interpret data and fitting results. For that reason, all the preprocessing functions related to the Fourier transform MR spectrum, have been added to the spectral window (see chapter 5), but it was made clear that these spectral processing features only affected the displayed MR spectrum, and *not* the raw FID data. This approach was implemented to gain maximum benefit from the combination of highly accurate time domain fitting of the raw data that had not suffered from operator-dependent preprocessing functions on the one hand, and the MR spectrum, massaged to the visual desires of the spectroscopist for easy interpretation, on the other.
However, not in all cases would a spectroscopist merely wish to analyze the raw FID (s)he measured, or a time series of them. It might be necessary to sum or subtract measured FIDs first, before doing the analysis. And, even for time domain analysis, there are a number of preprocessing functions that could be beneficial to the subsequent data analysis. An example is lineshape correction using reference line deconvolution. Or suppose you have an old or badly tuned spectrometer system that yields FIDs with a constant DC offset. The latter would not harm the spectrum, except for a narrow spike at 0 Hz, but it would harm the time domain analysis. For such preprocessing functions one would need MRUI functions that would actually directly affect the raw time domain points of the FIDs in the data files. Of course a copy of the original data file would need to remain, and a new data file would need to be created for the processed time domain signal. Such functions are provided in the FID Maths library, which you can select from the processing menu in the MRUI base window. One other useful preprocessing function, Filter_HSVD, which removes the residual water signal or any other unwanted features (solvent resonances, lipid resonances, etc.) from your signal, was considered important enough to earn a complete function and chapter for itself, see chapter 9.
When you select FID Maths from the processing menu, you are presented with the FID Maths library window (see), that lets you select a certain preprocessing function. You can also click on the Cancel button to make the FID Maths library window disappear without any further action.

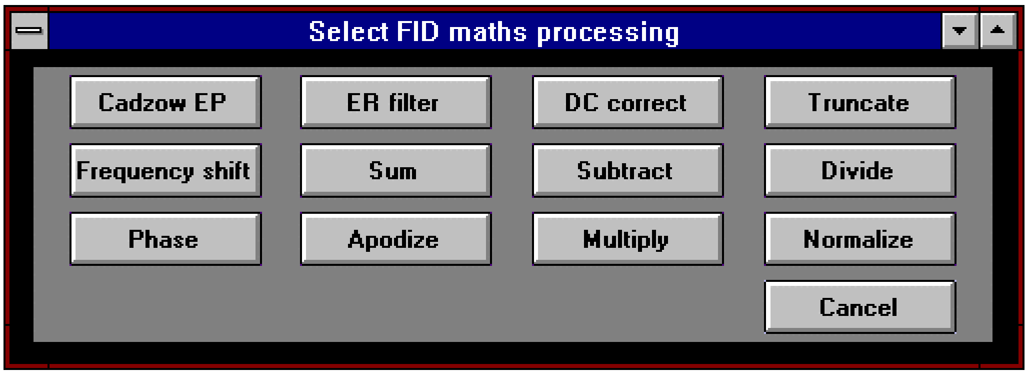


Figure 32. The FID Maths library window lets you select which preprocessing function you want to perform on your time domain FID.

Having selected your required preprocessing function, a window will pop up in which you can load your data file(s) and carry out the preprocessing function, view the results etc. In most cases this window will be the FID processing window (see for general features section 5.3), but for some preprocessing functions it turned out to be easier to use the basic spectral window (see for general features section 5.1). The various FID Maths processing functions are described in this chapter, but first we will describe the nomenclature of the data files for the processed FIDs.

8.1 Processed FID data files: names

It was just said that when processing the actual FID, a copy of the original data file should remain, and the processed FID should be saved to a new file. To indicate the relation of the processed signal to the original signal, the new data file name bares the same file name base, but with a processing-specific extension, before the normal `.dat` and `.mat` extension. Normally, this processing-specific name extension is separated from the original file name base by an underscore "_". For this reason you were advised not to create file names with underscores yourself (section 4.1.3), as MRUI might mistake them for processed FID files, and start looking for the original file which of course it then would not find. However, once you preprocess a data file that is itself the result of a preprocessing function (regardless which function that was), the file name will already contain the underscore followed by the previous processing-specific character, and in this case no new underscore will be added, but simply the new processing-specific character. The following table lists the processing-specific character extensions:

processing function	processing-specific extension		see
Filter_HSVD	`_p`	`p`	chapter 9
Cadzow EP	`_e`	`e`	section 8.2
ER filter	`_x`	`x`	section 8.3
DC correct	`_o`	`o`	section 8.4
Truncate	`_t`	`t`	section 8.5
Frequency shift	`_v`	`v`	section 8.6
Sum	`_s`	`s`	section 8.7
Subtract	`_b`	`b`	section 8.8
Divide	`_d`	`d`	section 8.9
Phase	`_f`	`f`	section 8.10
Apodize	`_a`	`a`	section 8.11
Multiply	`_m`	`m`	section 8.12
Normalize	`_n`	`n`	section 8.12

Thus, if you have an original data file called `<name>.dat/mat`, and you perform DC correct on it, the new data file will be called `<name>_o.dat/mat`. If you would next use Filter_HSVD to remove the residual water signal from that signal, the new data file would be called `<name>_op.dat/mat`. A subsequent region extraction with ER Filter would result in a new data file called `<name>_opx.dat/mat`, etc.

Notice that when you process a batch of files, the processing-specific extension gets added at the *end* of the file name base, i.e. only *after* the serial number of the file. This does however *not* harm your subsequent loading the processed batch into any other MRUI function, see section 5.1.1.1.2.

For all new data files produced by any of the preprocessing functions, a corresponding scale file `<name>.scl` will be produced, whether or not the original data file had a corresponding scale file. If the original data file had a scale file, its scaling factors will be taken into account when writing the scale file for the processed data file, so that you can carry out all your subsequent analyses without having to look after the scaling factors involved. If the original did not have a scale file, the processed signal will simply be normalized to the norm (1000 by default) and the appropriate scaling factors written to the new `<name>.scl` file, see also section 4.2.

8.2 Cadzow EP

The theory of this preprocessing function is best described in references [7,8,9]. The time domain data points are stored in a Hankel data matrix (see also section 10.1), and a number of singular value decompositions (SVDs) is iteratively carried out on that Hankel data matrix, each time truncating the SVD to the number of specified singular values, and restoring the Hankel symmetry of the matrix by averaging on the anti-diagonals[7]. Effectively, this separates the signal components from the noise components, and after a few iterations a signal is obtained of which the noise components have been removed. The resulting signal fits the noisy signal in the least squares sense if algorithm parameters are well chosen. Suggestions for these parameter values are outlined below. However, one has to be careful applying this method, as it is based on a number of assumptions, and may introduce systematic errors on the estimated parameters, especially the amplitudes[10]. On the other hand, if you intend to use SVD-based methods for parameter estimation (see also chapter 10) and the signal-to-noise ratio (SNR) of your data is low, the preprocessing step with the EP method may make results much more consistent and robust[9]. To conclude this part, we recommend that for accurate analysis of low SNR *in vivo* MRS data, you use the VARPRO method (see chapter 13) with prior knowledge (chapter 12) on the raw data. Should you insist on using an SVD-based method such as HLSVD or LPSVD for parameter estimation of low SNR *in vivo* signals, then it is recommended to precede the parameter estimation by the Cadzow Enhancement Procedure, but only after having consulted references [7,8,9].

The practicalities of using the Cadzow EP function in MRUI are described using the same example data set as before, the file ischr.dat/mat. The Cadzow EP preprocessing function uses the FID processing window as explained in section 5.3. Because only one signal is involved in the Cadzow EP process, the first two rows of the FID processing window are used, and the bottom row remains empty. The function-specific menu is titled EP SVD and allows you to set the algorithm parameters for the iterative (according to Cadzow, see for details and references e.g. ref. 7) SVDs as appropriate for your particular application. These are briefly described below, with references to the corresponding section in chapter 10, which describes SVD-based parameter estimation. The result of the Cadzow EP preprocessing method on the raw signal of ischr.dat, using the algorithm parameters as described below, is displayed in Figure 33.

8.2.1 ndp

This is the total number of data points that should be included in the Cadzow iterative SVDs. When storing the time domain data points in the Hankel data matrix, ndp points are stored. You can change it by typing a new number into an Edit Window. This number should be taken such that the FID has completely decayed into the noise level, but not too far beyond that point (or you would be involving ‘noise-only’ points in the matrix). As a rule of thumb, select ndp such that $\mathtt{ndp} * \mathtt{step} \cong 3 * T_2$, where step is the sampling time step (or dwell time, see e.g. sections 4.1.1,6.2.6), and T_2 the longest relaxation time present in the signal. To ensure reasonable computation times, ndp should not be set too large. When you load a signal, a default value for ndp is set, and displayed in the menu item. This default value is normally taken as the minimum of the default value set in the customize fitting parameters menu for EP (package default is 256, see also section 3.2.1.4), the actual number of signal points in the data file, and the maximum number of data points allowed for the method. If you change the value for ndp, it will again be tested against those maxima. Once you have run Cadzow EP on a signal, the next time you load the signal the same parameters will automatically be set as defaults (see also section 8.2.6). For the signal of ischr.dat, it could be seen in the FID plot (see Figure 55 in section 16.1) that 256 data points is indeed a good number to include the signal contribution and exclude noise-only points. Hence ndp could be left at its default value.

8.2.2 matrix

This is the size of the Hankel data matrix in which the time domain data points are stored, and which will be subjected to the iterative SVDs. You can change it by typing a new number of columns into an Edit Window. For a matrix with Hankel symmetry, the number of rows (nrows) is immediately known when both the number of columns (ncols) and the number of data points are known, as ncols + nrows = ndp + 1. See also section 10.1. The number of columns and the number of rows should at least be larger than the number of spectral components (model order of the signal; truncation point[7] in SVD; nssv parameter, see next section). The matrix has to be tall rather than long, i.e. more rows than columns, with a (nearly) square matrix as the extreme. However, to enhance computational times and resolving power, the data matrix for Cadzow EP SVD should best be taken tall (nrows >> ncols). When you load a signal, a default matrix size is set, and displayed in the menu item. This default size is normally taken from the default number of columns set in the customize fitting parameters menu for EP (package default is 57 for matrix size 200x57 at 256 data points, see also section 3.2.1.4), but corrected corresponding to the value of ndp if necessary. If you change the size of the matrix, it will again be tested against the appropriate minima and maxima. Once you have run Cadzow EP on a signal, the next time you load the signal the same parameters will automatically be set as defaults (see also section 8.2.6). For the signal of ischr.dat, the matrix size could be left at its default value: 200x57.

8.2.3 nssv

This parameter stands for ‘number of signal-related singular values’ (see also section 10.1). You can change it by typing a new number into an Edit Window. Its value determines where the singular value decomposition is truncated (subsequent entries in the singular value matrix set to zero) before back-multiplication to the data matrix of which the Hankel symmetry needs to be restored by averaging over the anti-diagonals[7]. This is also called the *model order*. Just as for SVD-based parameter estimation (remember that Cadzow EP is a signal enhancement method you use to preprocess your raw data in order to improve your SVD-based parameter estimation), this value should be set according to the number of spectral components. If your spectra suffer from overlap, underlying broad components or large noise features, it is safest to slightly overestimate the model order here, in order to cater for unforeseen signal contributions. When you load a signal, a default value for `nssv` is set, and displayed in the menu item. This default value is normally taken as the minimum of the default value set in the customize fitting parameters menu for EP (package default is 11, see also section 3.2.1.4), the number of data points divided by 4, and the maximum number of singular values allowed for the method. If you change the value for `nssv`, it will again be tested against those maxima. Once you have run Cadzow EP on a signal, the next time you load the signal the same parameters will automatically be set as defaults (see also section 8.2.6). For the signal of `ischr.dat`, it could be seen in the spectral window of the SETUP: experimental function (see Figure 31 in section 7.1) that 11 signal-related components (7 for ATP, 4 for PCr, PDE, P_i and PME) is indeed a good number, and `nssv` could be left at its default value.

8.2.4 nit

This is the number of Cadzow SVD iterations performed on the data matrix before producing the approximated signal. You can change it by typing a new number into an Edit Window. One iteration consists of SVD, truncation at `nssv` singular values, back-multiplication of the data matrix, and restoring Hankel symmetry by averaging on the anti-diagonals. Theoretically, these iterations should be performed until convergence, i.e. until the resulting elements on the anti-diagonals of the back-multiplied data matrix are within some error criterion already equal to each other, making averaging unnecessary to obtain Hankel symmetry. Usually, convergence is obtained after 10 to 15 Cadzow iterations[3,7,9]. Of course, performing 10 SVDs would mean a serious assault on your computation time, and it is now believed to be even beneficial in terms of parameter bias (when carrying out the subsequent SVD-based parameter estimation[3,9]) to perform only a relatively small number of Cadzow iterations, e.g. 3 or 5. When you load a signal, a default value for `nit` is set, and displayed in the menu item. This default value is normally taken as the minimum of the default value set in the customize fitting parameters menu for EP (package default is 5, see also section 3.2.1.4), and the maximum number of iterations allowed for the method. If you change the value for `nit`, it will again be tested against its maximum. Once you have run Cadzow EP on a signal, the next time you load the signal the same parameters will automatically be set as defaults (see also section 8.2.6). To demonstrate the use of the Cadzow EP method for the signal of `ischr.dat`, the number of Cadzow iterations was set to 3.

8.2.5 go!

After all the algorithm parameters have been set according to your wishes, select the go! menu item to start the process. If you had loaded a batch, the whole series of files will now be processed using the same algorithm parameters. The FID processing window will display a message ("running cadzow ep"; with serial index for batch processing) in the text bar in the lower right corner of the window, and after it has finished it will display the Cadzow EP results both for the time domain signal and the spectrum. See Figure 33. The name of the new data file containing the processed signal (in this case `ischr_e.dat`) is displayed over the resulting spectrum. Notice that regardless the length of the original signal, the new signal has only `ndp` data points. If this number is smaller than the original number of signal points, then the remaining length of the resulting signal is plotted as zeros (Matlab convention). These zeros are *not* saved to the `<name>_e.dat/mat` file. The resulting spectrum after Cadzow EP is always plotted without line broadening, because it is no longer perturbed by measurement noise.

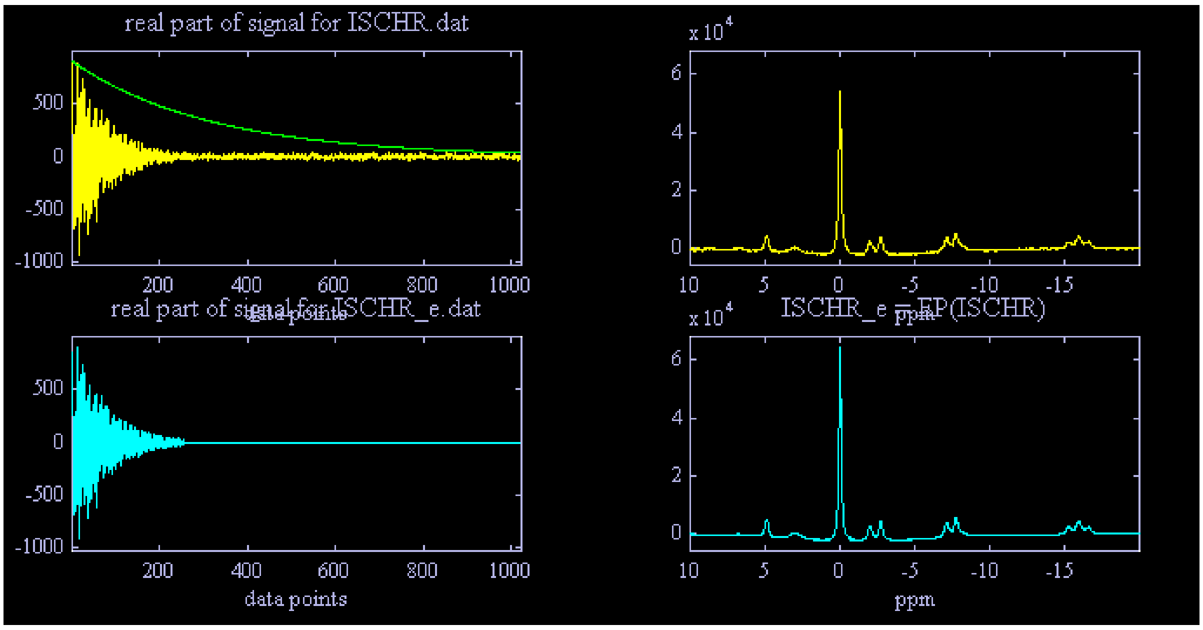


Figure 33. Part of the FID processing window after Cadzow EP preprocessing of ischr.dat. The resulting signal contains only ndp signal points. The resulting spectrum is always plotted without line broadening.

8.2.6 Files produced

Once you are finished using the Cadzow EP function, the following files will have been produced or updated:

```
<name>_e.dat
<name>_e.mat
<name>_e.scl
default.pep
<name>_e.pep
```

The file `default.pep` is an auxiliary ASCII file containing the algorithm parameters set in the menu items just before running the Cadzow EP method. The file `<name>_e.pep` is a file-specific copy of it. When you next load a data file `<name>.dat` into the Cadzow EP function, MRUI first checks if there is a corresponding `<name>_e.pep` file in that data directory, then if there is a `default.pep` file in that data directory. If one is found, the default values for the algorithm parameters are set accordingly and displayed in the menu items. If no `*.pep` file is found in the data directory, the default values are taken from the settings of the customize fitting parameters menu (see section 3.2.1.4).

8.3 ER Filter

The theory of this preprocessing function is briefly described in reference [11]. A full manuscript has been prepared and submitted[†]. This method is based on a filtering process by simple region extraction in the frequency domain spectrum. A region is selected in the Fourier transform MR spectrum, cut out, its central frequency shifted, and then inverse Fourier transformed to yield a time domain signal containing only the signal contributions with central frequencies in the selected spectral region, with a significantly decreased number of time domain data points (equal to the number of spectral points in the selected region). This gives the enhanced resolving power at a greatly reduced problem size for subsequent SVD-based parameter estimation. This method may be very beneficial if your spectrum is oversampled, or if it has a huge information content, with many large and overlapping peaks, and you are interested only in a small (isolated) subset of it. Often, when trying to quantify narrow doublets, one has to include an increasingly large number of data points in the time domain fitting rou-

[†] S. Cavassila, B. Fenet, A. van den Boogaart, C. Remy, A. Briguet, D. Graveron-Demilly, ER-Filter: a preprocessing technique to improve the performance of SVD-based quantitation methods. *J. Magn. Reson. Anal.* (1997)

tine, in order to properly resolve them. Increasing the number of data points in an SVD-based fit involves increasing the size of the data matrix (see section 10.1), which in turn may make computation times prohibitively long. After the ER Filter method, the signal content of interest is contained in a time domain signal of very short length, and can be rapidly estimated by any SVD-based method[11]. However, as for the Cadzow EP method, one has to reserve a certain amount of care in using this method for low SNR *in vivo* MRS data. If the resonances of interest can be consistently quantified from the raw data, using the VARPRO method with its frequency selective feature (see section 12.4.3), perhaps using a somewhat larger number of data points (for the VARPRO method, computation time increases linearly with signal size, for the SVD methods this involves a quadratic or even cubic factor), then that is definitely the preferred method. The ER Filter method does not yet implement a spectral window function; the selected region is simply cut out and directly inverse Fourier transformed. This may cause a sudden increase of signal intensity at the end of the resulting FID, instead of a decay towards zero (see Figure 36). When subsequently quantifying this signal with a time domain method, one has to be sure to exclude those final data points from the analysis. Also, the simple 'cut-and-transform' filter is not a so-called 'zero-phase filter'. This means that slight phase modulations may be introduced. Especially when selecting the spectral region close around the bottom of the peak, or when repeatedly carrying our the ER Filter technique on a signal, one can see phase distortions appear in the resulting spectra.

The practicalities of using the ER Filter function in MRUI are illustrated using the same example data set as before, the file `ischr.dat/mat`. The ER Filter preprocessing function uses the basic spectral window as explained in section 5.1, as it already contained the necessary features to implement the region extraction. The function-specific menu is titled ER_Filter. Its menu items are briefly described below. The result of the ER Filter preprocessing method can be viewed as the selected spectral region or the resulting FID, both in the basic spectral window. See Figure 35 and Figure 36.

8.3.1 enter region limits

This menu has two submenus, click with mouse and type values in. The selected setting simply defines how the spectral region of interest will be determined at the moment you activate ER Filter using the select region menu item. The default setting can be changed using the customize fitting parameters menu (see section 3.2.1.4). The package default is to define the region of interest using the mouse.

8.3.2 select region

Selecting this menu item starts the ER Filter process, by allowing you to select the spectral region of interest that is to be filtered out. It will carry out the whole process immediately. How you select the spectral region of interest depends on the settings of the enter region limits menu item (see previous section). Usually, the latter will be set to click with mouse, which means that with select region, you are supposed to click once to the left of the region, and then to the right of the region (or the other way, the minimum and maximum are automatically determined from the two entered values). A message text bar will appear in the bottom right corner, telling you what is expected from you, and blue vertical lines will be displayed where you have selected your region limits. You can use the customize preferences menu to disable the blue lines and/or message text in order to work faster, see section 3.2.1.3. See Figure 34. If the enter region limits menu item was set to type values in, you will be presented with an Edit Window twice, asking you for the low and high frequency limit of the region of interest, which you must type in the units as displayed on the frequencies (e.g. in ppm's in this example). In this case the blue vertical lines will also be displayed, but the message text bar will not appear. If you had loaded a batch, the whole series of files will be subjected to the ER Filter preprocessing technique using the same selected spectral region of interest.

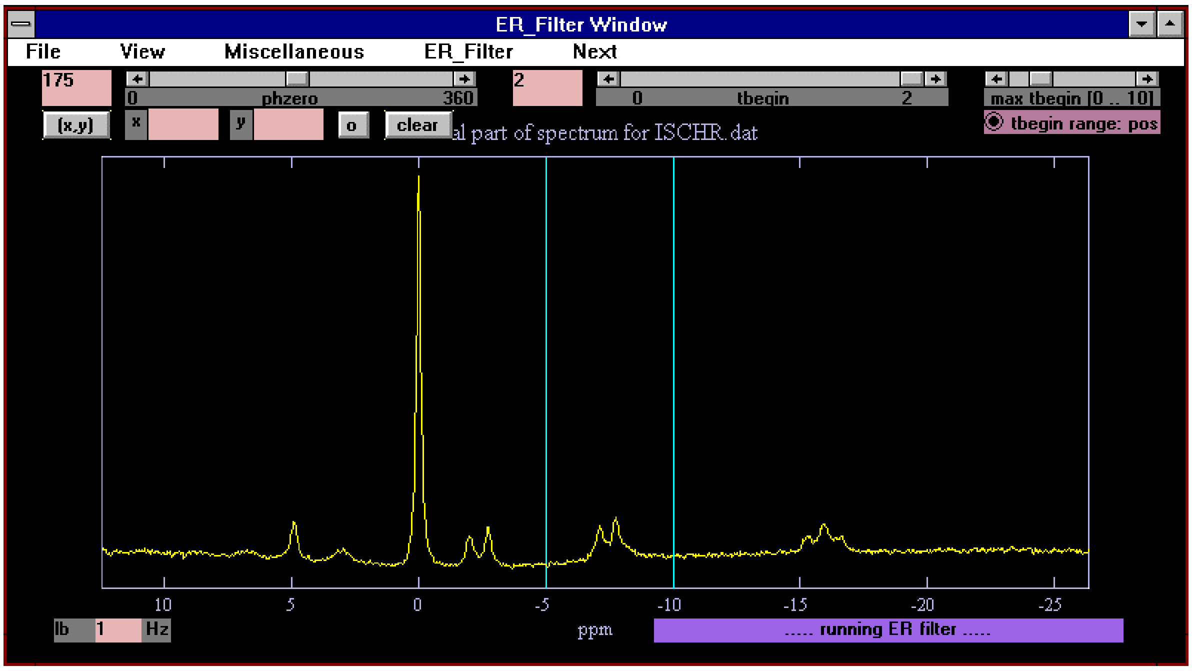


Figure 34. The basic spectral window after indicating the spectral region of interest to be 'cut out' by the ER Filter pre-processing technique.

8.3.3 auto ER filter

This menu item allows you to fully automate the ER Filter technique, by setting the appropriate fixed value for the spectral region to be filtered in the first submenu item (if you select this menu item you are presented with an Edit Window twice, as explained in the previous section), and then pressing the go! submenu item. Of course, if you would have to set these values each time before pressing go! under auto ER filter, it would not be very automatic, so you can set the default spectral region using the customize fitting parameters menu, see also section 3.2.1.4. The package default is between 25 and -25 Hz (quite a useless setting by the way, it is the same package default setting as for the Filter_HSVD method, see section 9.5; however in the case of ER Filter, requirements will be so diverse that no sensible package default could be selected). If you use the ER Filter technique regularly on similar data sets, you can set your default spectral region using the customize fitting parameters menu, and from that moment on you can ER Filter your signals simply by loading them into the function and selecting go! under auto ER filter.

After the region of interest has been selected, or the go! menu selected under auto ER filter, the function will cut out the selected region from the spectrum, shift its central frequency, and inverse Fourier transform to obtain the '*extracted*' FID. The latter has the same number of data points as there were spectral points in the selected spectral region. When ER Filter has finished, it will display the filtered spectral region fully zoomed out, see Figure 35.

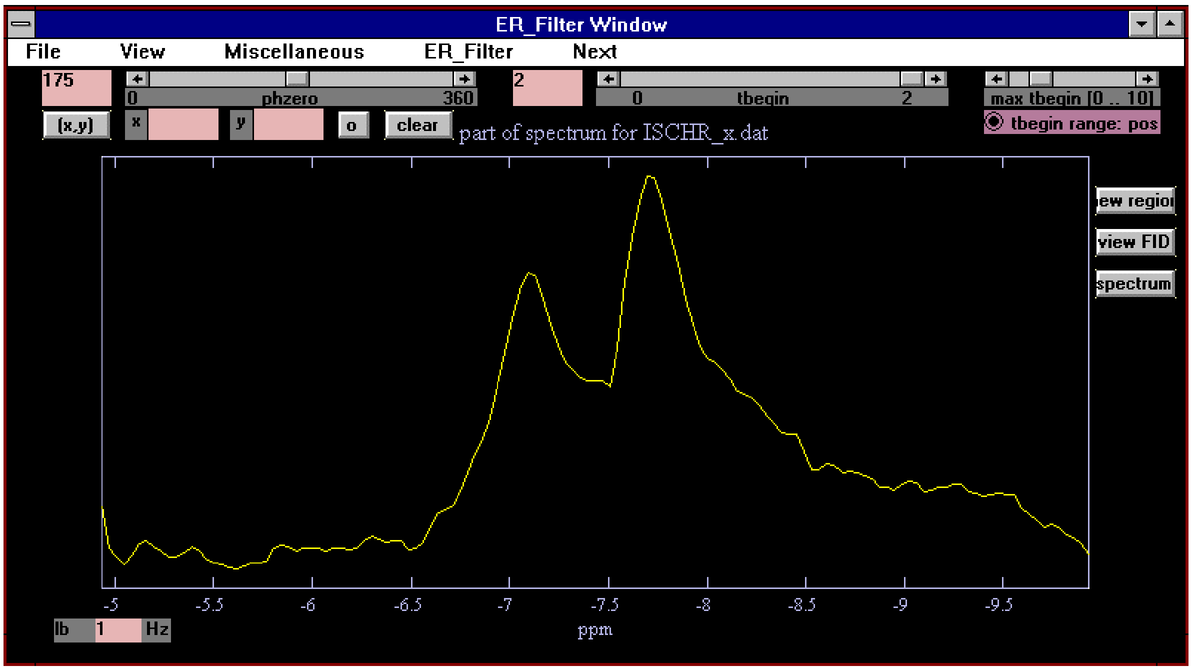
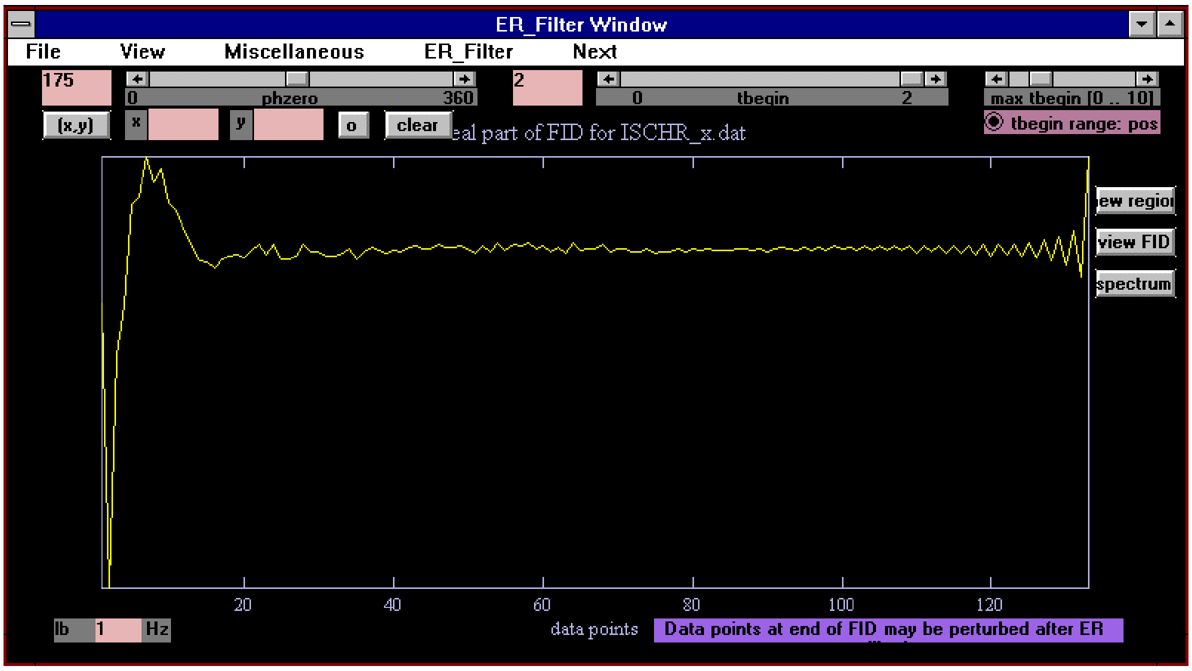


Figure 35. The ER_Filter window displays the filtered region of the spectrum fully zoomed out, after it has been cut out.

Note that the basic spectral window now contains three new pushbuttons at the far right of the window: new region, view FID and spectrum. The button view FID is important to interpret the resulting extracted FID, as it is often the case that, due to lack of windowing after spectral cutting, before inverse Fourier transform, the intensities of the last data points in the resulting FID show a sudden increase (see also the first paragraph of the ER Filter section). Displaying the resulting FID in the basic spectral window will then also display a message text bar in the bottom right corner of the window, saying that the data points at the end of the resulting FID may be perturbed. See Figure 36. Check out a ‘safe’ number of FID data points here to use for the subsequent parameter estimation method (e.g. 100 in this example). Pressing the spectrum button restores the result spectral display.



Data points at end of FID may be perturbed after ER

Figure 36. After pressing the FID button, the resulting FID is displayed. It has as many data points as there were spectral points in the selected spectral region. Notice how the data points at the end of the resulting FID show a sudden increase in intensity.

The new region button allows you to start over if you are not happy with the result. It will display the original spectrum again, and immediately prompts you to redo the spectral region selection.

Even if you do not press the new region button, the ER Filter menu item select region remains active. This means that within the same window, without having to reload files, you can perform subsequent ER Filter processes on the same signal. The data files will then of course be called `<name>_x.dat`, `<name>_xx.dat`, etc. However do be careful with subsequent ER Filter processes on the same signal, as phase modulations of the filter may lead to phase distortions.

8.3.4 Files produced

Once you are finished using the ER Filter function, the following files will have been produced or updated:

```
<name>_x.dat
<name>_x.mat
<name>_x.scl
```

8.4 DC correct

This preprocessing method involves a simple correction for any DC offset (constant intensity offset) of your measured FID(s). This offset can be calculated from the last data points of your FID, provided the MR signal has died out by that time. In the latter case, the last data points contain noise only, or perhaps very small oscillating signal contributions, and it can be assumed that the average (mean) intensity over those data points should be zero. Any difference from zero of the average over those data points can then be regarded as a constant intensity offset for your measured signal (a constant DC offset of the FID results in a narrow spike at 0 Hz in the spectrum), and is subtract to yield a DC offset correct FID, which can subsequently be used in the parameter estimation process instead of the DC offset perturbed FID.

To clearly show these effects, the practicalities of using the DC correct function in MRUI are described using a simulated signal (see chapter 6 on how to create these) containing two normal signal components and a DC offset. The DC offset was generated in the Simulate tool by adding a new component with a frequency of 0 kHz and a damping factor of 0 kHz. The amplitude of that component is then the value of the constant DC offset (in this example simulation signal this value was 10.0). The simulated signal was saved in the file `fidoff.dat/mat`. The DC correct preprocessing function uses the FID processing window as explained in section 5.3. Because only one signal is involved in the DC correct process, the first two rows of the FID processing window are used, and the bottom row remains empty. The function-specific menu is titled DC Offset Correction, and is briefly described below. The result of the DC correct preprocessing method on the raw signal of `fidoff.dat` is displayed in Figure 37.

8.4.1 ndcoff

This menu item contains the only available setting for this preprocessing method: the number of data points to be used from the end of the FID in order to estimate the constant intensity offset as the mean of those data points. After loading the raw signal into the function, the FID and spectrum are already displayed, see also Figure 37. Check the raw FID in the top left plot for the point where the signal components have died out, and the remaining signal contains noise only. When you load a signal, a default value for ndcoff is set, and displayed in the menu item. This default value is normally taken as the minimum of the default value set in the customize fitting parameters menu for DC correct (package default is 100, see also section 3.2.1.4) and the actual number of data points. If you change the value for `ndcoff`, it will again be tested against this maximum. Upon selecting the ndcoff menu item, you can type your new value into an Edit Window. The menu item display will be updated with this value. For this signal it is possible to set `ndcoff` for instance at 500.

8.4.2 go!

After you have set `ndcoff`, select the go! menu item to start the process. If you had loaded a batch, the whole series of files will now be processed using the same `ndcoff` parameter. The FID processing window will display the results both for the time domain signal and the spectrum. See Figure 37. The name of the new data file containing the processed signal (in this case `fidoff_o.dat`) is displayed over the resulting spectrum, together with the value of the DC offset it estimated, which can be seen to be good (the true DC offset of the noiseless simulation signal was 10). The DC spike has also disappeared from the spectrum.

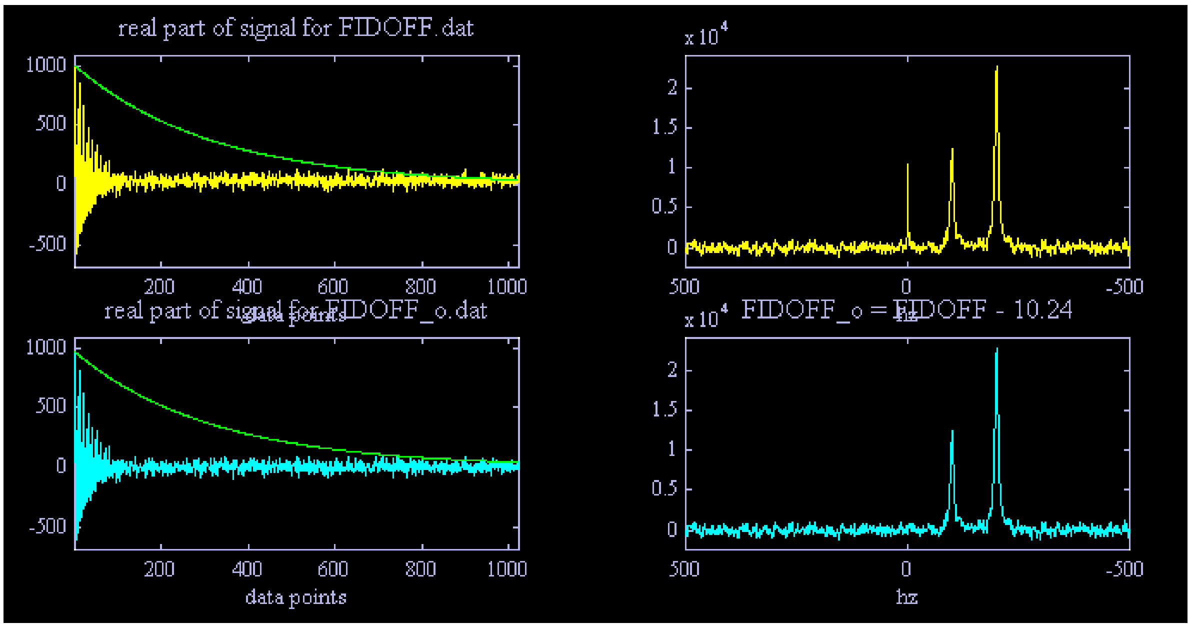


Figure 37. Part of the FID processing window displaying the result of the DC offset correction method both in the time and the frequency domain. After the constant (DC) intensity offset has been subtracted from the FID, the DC spike also disappears from the spectrum.

8.4.3 Files produced

Once you are finished using the DC correct function, the following files will have been produced or updated:

```
<name>_o.dat
<name>_o.mat
<name>_o.scl
```

8.5 Truncate

This preprocessing method involves removing data points from the measured FID, either at the beginning or at the end. Removing points from the end of a measured FID allows you to throw away points that may for some reasons have been perturbed by a ‘spike’ at the end of the FID acquisition, or you may truncate the final part of the FIDs that have resulted from the ER Filter method (see section 8.3). However, you should realize that for the parameter estimation techniques this is unnecessary, as you can set the number of data points included in the fit such, that data points at the end of the fit are excluded. The same goes for removing points from the beginning of the fit. With the VARPRO method, you have the liberty of excluding initial data points from the fit, in order to eliminate the influence of receiver distortions on initial FID points, or of rapidly decaying signal components, corresponding to broad underlying baseline humps in the spectrum (see section 12.4.2). However, the SVD-based methods do not have such a built-in feature, and you may use this preprocessing method to remove any suspicious initial data points, and carry out the SVD-based parameter estimation on the truncated FID. Note that truncating initial data points involves a change of the begin time for the resulting signal (the begin time increases by `begoff*step`, where `begoff` is the number of points truncated from the beginning of the FID, and `step` the sampling time). This may cause a rolling baseline in the resulting spectrum, due to the sinc convolution of the original spectrum with the Fourier transform of the step function that the point removal effectively is, see also Figure 38. This rolling spectral baseline is a direct consequence of the Fourier transform of an incomplete signal, and will *not* influence the time domain signal analysis by SVD_1D or VARPRO.

The practicalities of using the Truncate function in MRUI are described using the same example data set as before, the file `ischr.dat/mat`. The Truncate preprocessing function uses the FID processing window as explained in section 5.3. Because only one signal is involved in the Truncate process, the first two rows of the FID processing window are used, and the bottom row remains empty. The function-specific menu is titled Truncation, and is briefly described below. The result of the Truncate preprocessing method on the raw signal of `ischr.dat` is displayed in Figure 38.

8.5.1 off begin & off end

These two menu items contain the two available settings for this preprocessing method: the number of data points to be removed from the beginning of the FID (off begin) and the number of data points to be removed from the end of the FID (off end). After loading the raw signal into the function, the FID and spectrum are already displayed, see also Figure 38. Check the raw FID in the top left plot for suspiciously large initial data points, or spikes near the end. When you load a signal, default values for off begin and off end are set, and displayed in the menu items. These default values are normally taken as the minimum of the default value set in the customize fitting parameters menu for Truncate (package default is 0 for both, see also section 3.2.1.4) and the actual number of data points minus 1. If you change either value, it will again be tested against this maximum. Upon selecting the off begin or off end menu item, you can type your new value into an Edit Window. The menu item display will be updated with this value. To illustrate this example, the first 3 data points were truncated (for instance to eliminate receiver distortions), and the last 500 points, as these can be removed without harming subsequent SVD-based parameter estimation as the signal has died out by then.

8.5.2 go!

After you have set `the` number(s) of data points to be truncated, select the go! menu item to start the process. If you had loaded a batch, the whole series of files will now be processed using the same number(s) of data points. The FID processing window will display the results both for the time domain signal and the spectrum. See Figure 38. The name of the new data file containing the processed signal (in this case `ischr_t.dat`) is displayed over the resulting spectrum.
Notice that the new signal has only `ndp-begoff-endoff` data points. The excluded data points from the resulting FID are plotted as zeros (Matlab convention). These zeros are *not* saved to the `ischr_t.dat/mat` file.

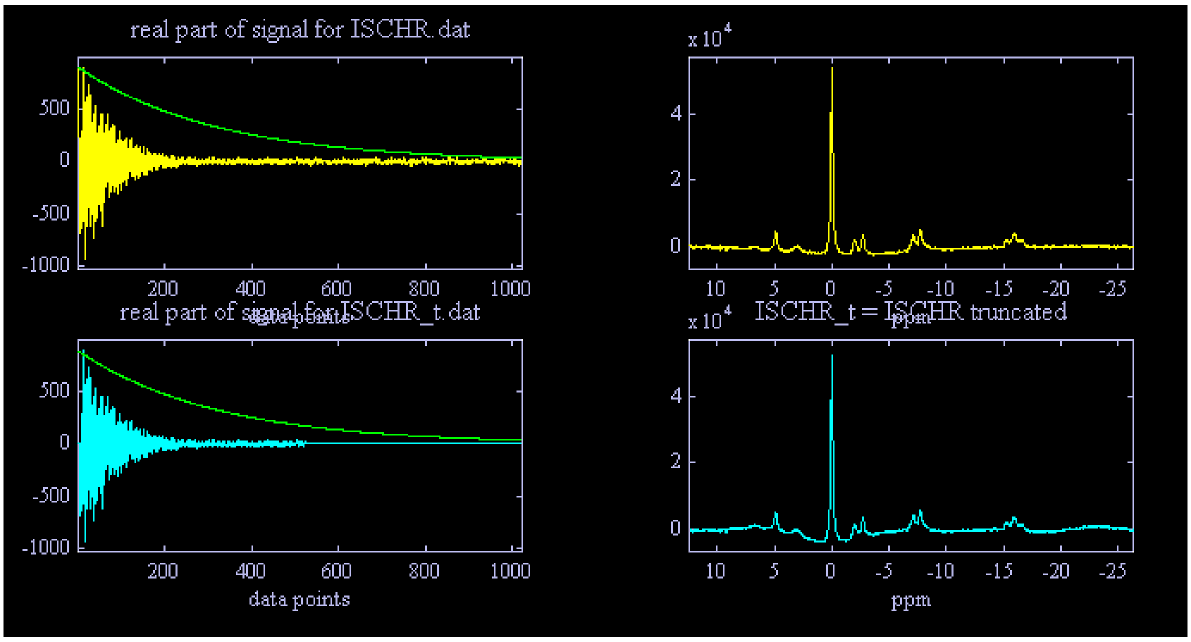


Figure 38. Part of the FID processing window showing the effects of truncation both on the FID and the Fourier transform spectrum. The zeros in the resulting FID are plotted due to Matlab conventions and are not saved to the resulting data file.

8.5.3 Files produced

Once you are finished using the Truncate function, the following files will have been produced or updated:

```
<name>_t.dat
<name>_t.mat
<name>_t.scl
```

8.6 Frequency shift

Should you have reason to believe that the actual true frequencies in kHz of the original signal are to be adapted, then you can use this Frequency shift preprocessing function, which applies the frequency shift also to the time domain FID and stores it as a new signal. You might for example have two FIDs, acquired at different days. If you know that the spectrometer's frequency drift in that period was 2 Hz, you can 'line up' the data sets again, by applying a frequency shift of -2 Hz to the second signal, using this function. This is principally different from the difference in chemical shift offset as you can bring about using the ppm reference menu item in the SETUP: experimental function, see section 7.1.5. The latter only affects the spectral display, not the true frequencies in the signal. The frequency shift applied to the time domain signal may be especially important if you wish to re-use sets of starting values or prior knowledge between files. You can either enter a constant frequency offset by which all frequencies in the spectrum should be shifted, or click on a peak and then enter the frequency it should have. You can select whether the kHz frequency reference for ppm units (as described in section 7.1.5) should be shifted as well.

The Frequency shift preprocessing function uses the basic spectral window as explained in section 5.1, as it already contained the necessary features to implement the frequency selection. The function-specific menu is titled Freq_shift. Its menu items are briefly described below. Because this function hardly causes visual changes to the displayed spectrum, its results are not displayed in a figure.

8.6.1 update ref. frequency (SETUP)

In section 7.1.5 it was explained how you could match the chemical shift axis in ppm's to your expectations by setting the ppm offset for a certain frequency in the spectrum. That action links the variable `ppmref` (the ppm value you enter) to the variable khzref (the true frequency in kHz that is calibrated to have that chemical shift offset). These two variables are stored in the default.xpt file, see also section 4.3. The menu item update ref. frequency (SETUP) determines whether, once you have applied your absolute frequency shift in (k)Hz, this frequency reference variable khzref should be shifted likewise. Setting this option to yes would ensure that as long as a peak changes its true frequency through your action, it would retain its original chemical shift. If you want the chemical shifts in ppm to 'move with' the shifted frequencies, you should set this option to no. The default setting can be changed using the customize fitting parameters menu (see section 3.2.1.4). The package default is yes.

8.6.2 determine frequency shift

Selecting this menu item starts the Frequency shift process, by allowing you to indicate the absolute frequency shift. It will carry out the whole process immediately. You can select between two different ways of applying the frequency shift.

8.6.2.1 constant offset value

This submenu item would be the most common way to apply a spectrometer frequency drift to a signal of a series, measured at different time points. It presents an Edit Window, in which you can type the frequency shift in Hz. It will apply this frequency shift to the time domain data, and update the spectral display. Whether the chemical shift axis in ppm's will change depends on the setting of the update ref. frequency (SETUP) menu.

8.6.2.2 click peak as reference

Selecting this menu item allows you to select a peak in the spectrum, and indicate the true frequency that should be valid for that position. This works much like the ppm reference menu item in the SETUP: experimental function: you click on a peak, and then type its 'true' frequency in Hz in the Edit Window that pops up. In fact, using this function, you could actually place the 0 Hz position wherever you like. That could be useful to make sure your water resonance is dead on 0 Hz, but it also gives you the chance just to see what would happen with the zero order phasing if it had its *pivot* on a different peak! In section 5.1.4.2 it was explained that due to the characteristics of the begin time, the pivot had to be at 0 Hz, but if you can change the 0 Hz point with this function, you are actually provided with the means of checking out the phasing with the first order phase pivot positioned at different peaks. The corresponding modifications in the spectrum will be immediately updated in the spectral display.

8.6.3 Files produced

Once you are finished using the Frequency shift function, the following files will have been produced or updated:

```
<name>_v.dat
<name>_v.mat
<name>_v.scl
default.xpt
```

The file default.xpt is only updated if you had the update ref. frequency (SETUP) menu item set to yes (changes the variable khzref which is stored in default.xpt).

8.7 Sum

With this preprocessing function you can sum two FIDs together, you can sum one FID to each FID of a batch of files, you can sum two batches pair-wise together, or you can sum all FIDs in a batch together to create one file containing the sum of the FIDs in the batch. Whenever two FIDs are added together, the function checks whether the number of data points, sampling time and begin time are identical for the two FIDs. If there is a difference for the number of data points, the summed FID will get the smaller number of data points of the two. If the sampling time and/or begin time values are different between the two, Sum will use the value from the originally loaded signal for the summed FID.

The practicalities of using the Sum function in MRUI are illustrated using the same example data set as before, the file `ischr.dat/mat`. It will be summed to another FID of the same experiment, which was stored in the file `ischb.dat/mat`. The Sum preprocessing function uses the FID processing window as explained in section 5.3. Because usually two signals are involved in the Sum process, the first two rows of the FID processing window are used for the FIDs being added together, and the bottom row displays their sum. All FIDs and spectra will be scaled according to the largest signal. The function-specific menu is titled Sum, and is briefly described below. The result of the Sum preprocessing method on the raw signals of `ischr.dat` and `ischb.dat` is displayed in Figure 39.

8.7.1 load

Depending on whether you initially loaded a single file or a batch, this menu will have one or two active submenus. The file submenu is always active, batch only if you initially loaded a batch.

8.7.1.1 file

If you initially loaded a single file, this is your only option. Selecting this submenu presents you with the file manager window, and you can select (see also section 5.1.1.1.1) the file containing the FID you want to sum to the initially loaded one. This signal will then be displayed on the second row, and the sum of the two finally on the bottom row. This was the case in the current example, where the file `ischb.dat` from the same directory (not necessary) was selected and added to `ischr.dat`. The result is displayed in Figure 39. The summed FID is saved as `ischr_s.dat/mat`, as is indicated over the resulting spectrum.

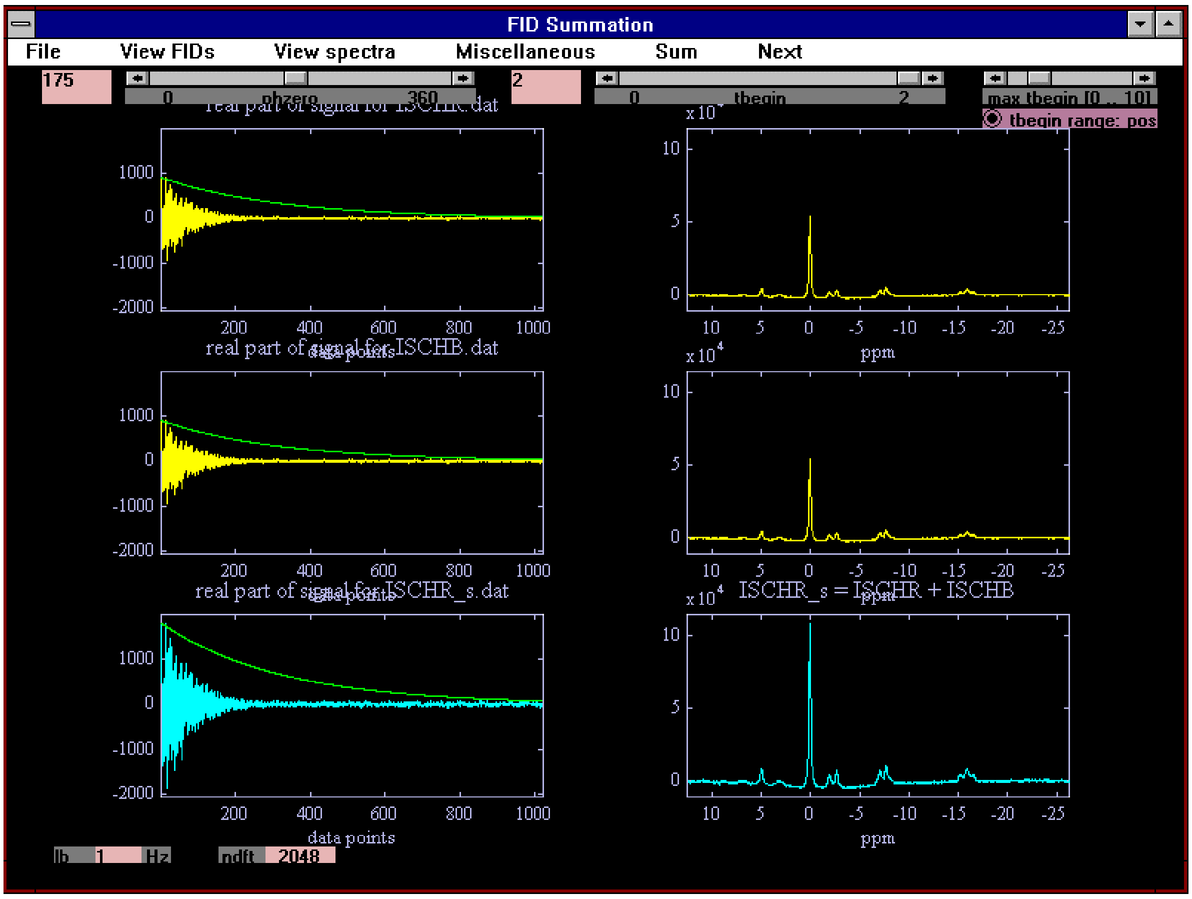


Figure 39. The FID processing window displaying the result of the two FIDs from ischr.dat and ischb.dat being summed together.

If you had initially loaded a batch, selecting file now also presents you with the file manager window, and the FID in the single data file you select will be summed to *each* signal in the loaded batch. This is done fully automatically. The resulting signals are stored as their original basenames in the batch, including their serial number, followed by the _s or s extension.

8.7.1.2 *batch*

This submenu item is only enabled (active) if you initially loaded a batch. In this case you can load a second batch, and the files of the two batches will be added together pair-wise. In this case the functions only sums FIDs together between the batches if they have matching serial numbers. The resulting signals are stored as their original basenames in the batch, including their serial number, followed by the _s or s extension.

8.7.2 sum the batch

This submenu item is only enabled (active) if you initially loaded a batch. By selecting this menu item, you sum all FIDs in the loaded batch together. The resulting summed FID will be stored in a file whose name is the original general base name of the loaded batch (i.e. without the serial numbers), followed by the _s or s extension. The display in the FID processing window is slightly different in this case. As with all the batches, initially only the first signal of the batch is displayed (on the first row). After all FIDs in the batch have been added together, the resultant, summed FID is displayed both in the time domain and the frequency domain on the second row.

8.7.3 Files produced

Once you are finished using the Sum function, the following files will have been produced or updated:

```
<name>_s.dat
<name>_s.mat
<name>_s.scl
```

8.8 Subtract

With this preprocessing function you can subtract one FID from another, you can subtract one FID from each FID of a batch of files, or you can subtract one batch from another, pair-wise. Whenever a FID is subtracted from another, the function checks whether the number of data points, sampling time and begin time are identical for the two FIDs. If there is a difference for the number of data points, the resulting FID will get the smaller number of data points of the two. If the sampling time and/or begin time values are different between the two, Subtract will use the value from the originally loaded signal for the resulting FID.

The Subtract preprocessing function uses the FID processing window as explained in section 5.3. Because two signals are involved in the Subtract process, the first two rows of the FID processing window are used for the original FID and the FID to be subtracted from it, and the bottom row displays their difference. All FIDs and spectra will be scaled according to the largest signal. The function-specific menu is titled Subtract, and is briefly described below. The result of a Subtract preprocessing is displayed in Figure 26 of section 5.3. In this case the original file `test.dat/mat` was taken (a simulation signal mimicking an *in vivo* ^{31}P MRS signal from brain) and first subjected to the Cadzow EP preprocessing method (not shown). This resulted in a noise-suppressed FID stored in `test_e.dat/mat`. The latter signal was subtracted from the original signal in order to establish if the difference signal, the contributions removed by the Cadzow EP method, contained no signal components. Since the original signal had 2048 data points, but the Cadzow EP method had been performed with 256 data points only, the two signals had different lengths (missing data points are plotted as zeros; Matlab convention), and Subtract took the smaller of the two, 256, as the number of data points for the difference signal. The zeros plotted in the bottom FID are not saved to the result file `test_b.dat/mat`.

8.8.1 load

Depending on whether you initially loaded a single file or a batch, this menu will have one or two active submenus. The file submenu is always active, batch only if you initially loaded a batch.

8.8.1.1 file

If you initially loaded a single file, this is your only option. Selecting this submenu presents you with the file manager window, and you can select (see also section 5.1.1.1.1) the file containing the FID you want to subtract from the initially loaded one. This signal will then be displayed on the second row, and the difference of the two finally on the bottom row. See also Figure 26 in section 5.3. The difference FID is saved as `test_b.dat/mat`, as is indicated over the resulting spectrum.

If you had initially loaded a batch, selecting file now also presents you with the file manager window, and the FID in the single data file you select will be subtracted from *each* signal in the loaded batch. This is done fully automatically. The resulting signals are stored as their original basenames in the batch, including their serial number, followed by the _b or b extension.

8.8.1.2 batch

This submenu item is only enabled (active) if you initially loaded a batch. In this case you can load a second batch, and the files of the second batch will be pair-wise subtracted from the files of the original batch. In this case the function only performs the subtraction between files of the two batches if they have matching serial numbers. The resulting signals are stored as their original basenames in the batch, including their serial number, followed by the _b or b extension.

8.8.2 Files produced

Once you are finished using the Subtract function, the following files will have been produced or updated:

```
<name>_b.dat
<name>_b.mat
<name>_b.scl
```

8.9 Divide

With this preprocessing function you can divide one FID by another, you can divide each FID of a batch of files by one particular FID, or you can divide one batch by another, pair-wise. Whenever a FID is divided by another, the function checks whether the number of data points, sampling time and begin time are identical for the two FIDs. If there is a difference for the number of data points, the function has to abort. If the sampling time and/or begin time values are different between the two, Divide will use the value from the originally loaded signal for the resulting FID. The FID division is not as straightforward as the summation or subtraction. In the time domain, the FIDs are divided point-wise, i.e. each point of FID_1 is divided by the corresponding point of FID_2. This must be done taking into account that both two points in the division are complex entities. To get meaningful results, the FID by which you divide should be normalized to unity. Further, because FID_2 is oscillating and decaying, there must be a protection against the denominator of the division becoming zero or very small, which would give extreme and unrealistic spikes in the resulting signal. Because FID_2 is normalized to itself, FID_1 is actually divided by the phase term of FID_2. Using this method, you can carry out signal correction for eddy currents and magnetic field inhomogeneities for *in vivo* ^{1}H MR signals, by dividing the measured water-suppressed metabolite signal by the phase term of the unsuppressed water signal. The latter contains the frequency offset, and any modulations brought about by eddy currents and inhomogeneities. Provided the unsuppressed water signal was measured from the same voxel under identical experimental conditions, these distortions are divided out of the water-suppressed metabolite signal. This is especially clearly described in Uwe Klose's publication[12]. This method is also known as QUALITY[13]. Certain spectrometer conversion routines (for GE PROBE files and Picker data) use this algorithm automatically to produce the water-suppressed metabolite FID, the unsuppressed water FID, and immediately a third signal: the lineshape-corrected metabolite FID. See sections 4.5 and ***. If your the conversion routine you use does not have this feature, but you have measured water-suppressed metabolite FIDs and unsuppressed water FIDs from the same voxel, you can use this preprocessing function to carry out the 'eddy current correction' manually. Of course you can also use this function to divide out phase terms of totally arbitrary signals.

The Divide preprocessing function uses the FID processing window as explained in section 5.3. Because two signals are involved in the Divide process, the first two rows of the FID processing window are used for the original FID and the FID it is to be divided by, and the bottom row displays the resulting signal. The FIDs and spectra in the top and bottom row will be scaled according to the largest signal, but the FID and spectrum in the second row are scaled individually, because their magnitudes do not play a role in the division process (of the second FID, only the complex phase factor is used in the division). The function-specific menu is titled Divide, and is briefly described below. The result of the Divide preprocessing method is displayed in Figure 40. In this case the converted GE PROBE files (see section 4.5) `abm.dat` (water-suppressed metabolite signal) and `abw.dat` (unsuppressed water signal) were manually divided to yield the lineshape corrected water-suppressed metabolite signal, which was now stored as `abm_d.dat`, but contains of course the same signal as `abz.dat` from the GE PROBE conversion routine (compare Figure 40 with *** and *** in section 4.5).

8.9.1 load

Depending on whether you initially loaded a single file or a batch, this menu will have one or two active submenus. The file submenu is always active, batch only if you initially loaded a batch.

8.9.1.1 file

If you initially loaded a single file, this is your only option. Selecting this submenu presents you with the file manager window, and you can select (see also section 5.1.1.1.1) the file containing the FID you want the initially loaded one to be divided by. This signal will then be displayed on the second row, and the division result of the two finally on the bottom row. See Figure 40. The divided FID is saved as `abm_d.dat/mat`, as is indicated over the resulting spectrum.

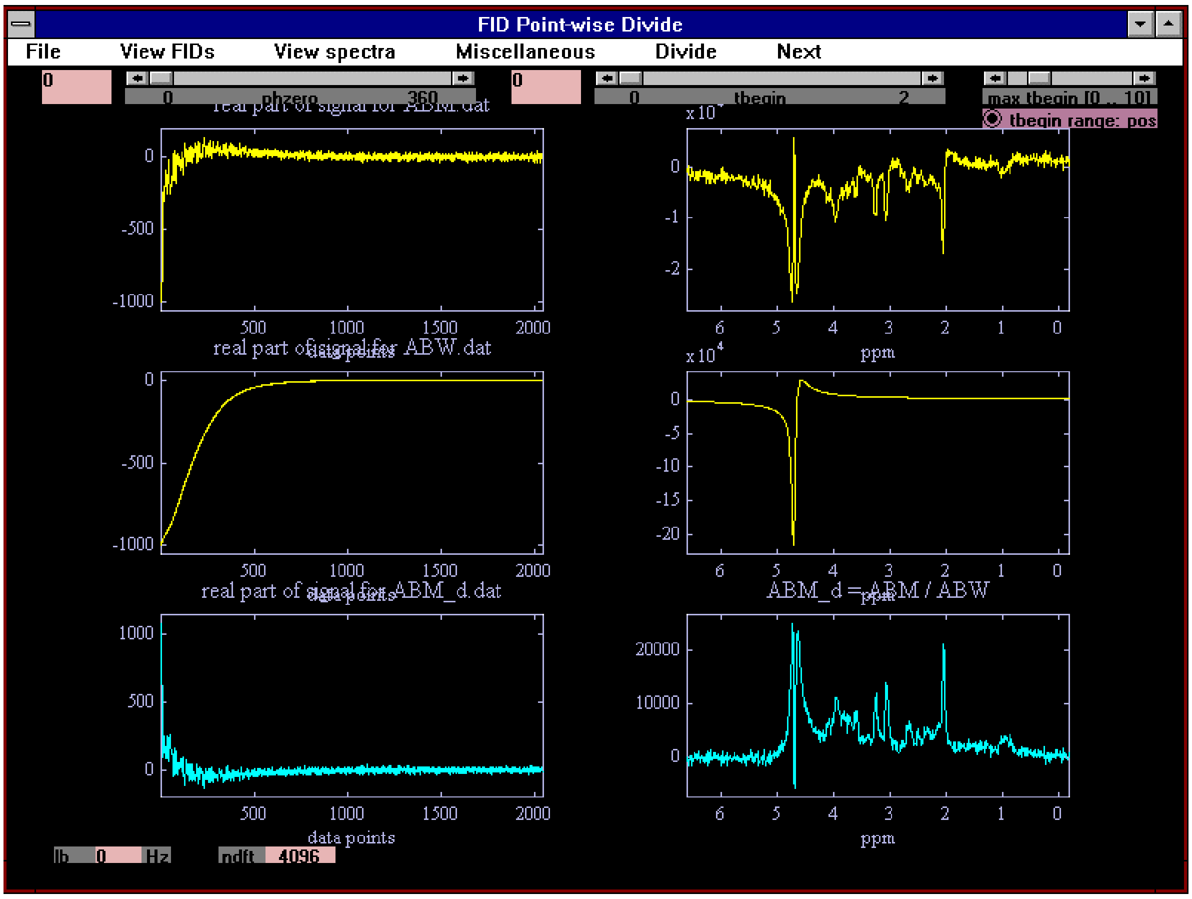


Figure 40. The FID processing window displays the results of the Divide function: the signal of abm.dat has been divided by the complex phase factor of the (unsuppressed water) signal in abw.dat. This eliminates frequency and phase modulations; notice that the resulting spectrum is automatically phased.

If you had initially loaded a batch, selecting file now also presents you with the file manager window, and *each* signal in the loaded batch will be divided by the phase term of the FID in the single data file you select. This allows fully automatic ‘eddy current correction’ of a time series of data files. The resulting signals are stored as their original basenames in the batch, including their serial number, followed by the _d or d extension.

8.9.1.2 batch

This submenu item is only enabled (active) if you initially loaded a batch. In this case you can load a second batch, and the files of the original batch will be pair-wise divided by the files of the second batch. In this case the functions only performs the division between files of the two batches if they have matching serial numbers. The resulting signals are stored as their original basenames in the batch, including their serial number, followed by the _d or d extension.

8.9.2 Files produced

Once you are finished using the Divide function, the following files will have been produced or updated:

`<name>_d.dat`

```
<name>_d.mat
<name>_d.scl
```

8.10 Phase

This preprocessing function is nothing different from the normal basic spectral window with its phzero and tbegin sliders. It allows you to phase the displayed spectrum using the sliders, but in this particular case, the phase modifications are actually stored in the time domain FID signal. The zero order phase is taken from the phzero slider, and the FID is multiplied by a complex exponential term containing that zero order phase. If you perform 'first order phasing' by changing the begin time using the tbegin slider, the `begin` variable in the data file (see section 4.1.1) is modified according to the value of the slider. This means that you are actually '*phasing the FID*'. This may be very important if you wish to sum or subtract two FIDs from different experiments, that might have e.g. a different zero order phase (for example, this is the case in a metabolite nulling experiment[6]). The actual phasing works exactly as described in section 5.1.4.

The Phase preprocessing function uses the basic spectral window as explained in section 5.1, as it already contained the necessary features to implement the phase modulations. The function-specific menu is titled FID phase. Its only menu item is briefly described below. This preprocessing function only involves phasing the spectrum (after which the resulting phase factors take their effect on the FID), so no special figure is dedicated to it.

8.10.1 save phased FID

Once you have phased the spectrum to your satisfaction, select this menu item and the function will apply the phase modulations to the time domain FID, and store the resulting signal in the `<name>_f.dat/mat` file.

8.10.2 Files produced

Once you are finished using the Phase function, the following files will have been produced or updated:

```
<name>.gra
<name>_f.dat
<name>_f.mat
<name>_f.scl
<name>_f.gra
default.gra
```

8.11 Apodize

You have read in section 5.1.6 how you could apply line broadening to the displayed spectrum. However, it was already mentioned in that section, that on some occasions you might also want the apodization (multiplication of the FID with a decaying function) to affect your FID data points. This would correspond to applying a so-called *matched filter* to your signal before carrying out the parameter estimation. The use of matched filters for time domain analysis in MRS is currently being investigated by two MRUI user groups. Theoretically, a matched filter would mean that the decay factor of the apodization function is equal to the effective decay factor of the envelope of your signal (hence the term 'matched'). This causes a filtering of your FID in terms of signal and noise: the apodization function has values close to 1 in the beginning, where the FID contains mostly MRS signal, and values close to 0 at the end, where the FID contains mostly noise. Effectively, the apodization preprocessing technique is a noise suppression method, from which subsequent parameter estimation methods might benefit. Beware however that the multiplication with a Lorentzian of certain width brings about an overestimation of the true damping factors (linewidths) of the same factor. The damping factor (linewidth) bias when apodizing with a Gaussian is more complex (unless the signal itself exhibits Gaussian lineshape, in which case the situation is reversed). Apodization is in spectral jargon also termed line broadening, and that is exactly what happens: noise features are reduced, and spectral resonances become broader, causing closely spaced peaks to start overlapping more and more at larger line broadening factors, yielding an overall 'smoothing' of the spectrum. With the Apodize preprocessing function you can apodize your FID(s) and check the result on the displayed spectrum. You can choose between Lorentzian and Gaussian apodization. The Lorentzian apodization function shows an exponential decrease in intensity, whereas the Gaussian apodization function has mostly high intensity values in the beginning, then a steep decrease, following by a flat, nearly 0 portion at the end of the signal (see Figure 41). That function might be appropriate to leave your signal content in the initial part of the FID unaffected, whereas noise contributions at the end of the FID are more strongly suppressed. The point where the apodization has decreased to `1/e` of its initial value is determined by the apodization value in Hz that you provide.

The practicalities of using the Apodize function in MRUI are described using the same example data set as before, the file `ischr.dat/mat`. The Apodize preprocessing function uses the FID processing window as explained in section 5.3. Because only one signal is involved in the Apodize process, the first two rows of the FID processing window are used, and the bottom row remains empty. Obviously, the loaded signal on the first row is displayed without 'cosmetic' line broadening, and the lb edit box (see section 5.1.6) is unavailable. The function-specific menu is titled Apodization, and is briefly described below. The result of the Apodize preprocessing method, applying Gaussian apodization, on the raw signal of `ischr.dat` is displayed in Figure 41.

8.11.1 Lorentzian

This menu item presents you with an Edit Window in which you can type the Lorentzian filter value in Hz. You can enter negative values if you want. When you press `[Enter]` after your value, the Lorentzian apodization will be immediately applied to the signal, and the results displayed in the FID processing window.

8.11.2 Gaussian

This menu item presents you with an Edit Window in which you can type the Gaussian filter value in Hz. You can enter negative values if you want. When you press `[Enter]` after your value, the Gaussian apodization will be immediately applied to the signal, and the results displayed in the FID processing window, see Figure 41. The signals and spectra on the two rows are scaled according to the largest signal. The actual apodization function is displayed in green in the time domain plot for the original signal (although the maximum intensity of the apodization function is of course always 1, it is drawn with the first point at the maximum intensity of the FID to allow easier visual interpretation). The name of the new data file containing the processed signal (in this case `ischr_a.dat`) is displayed over the resulting spectrum. If you had loaded a batch, the whole series of files will now be processed using the same number(s) of data points.

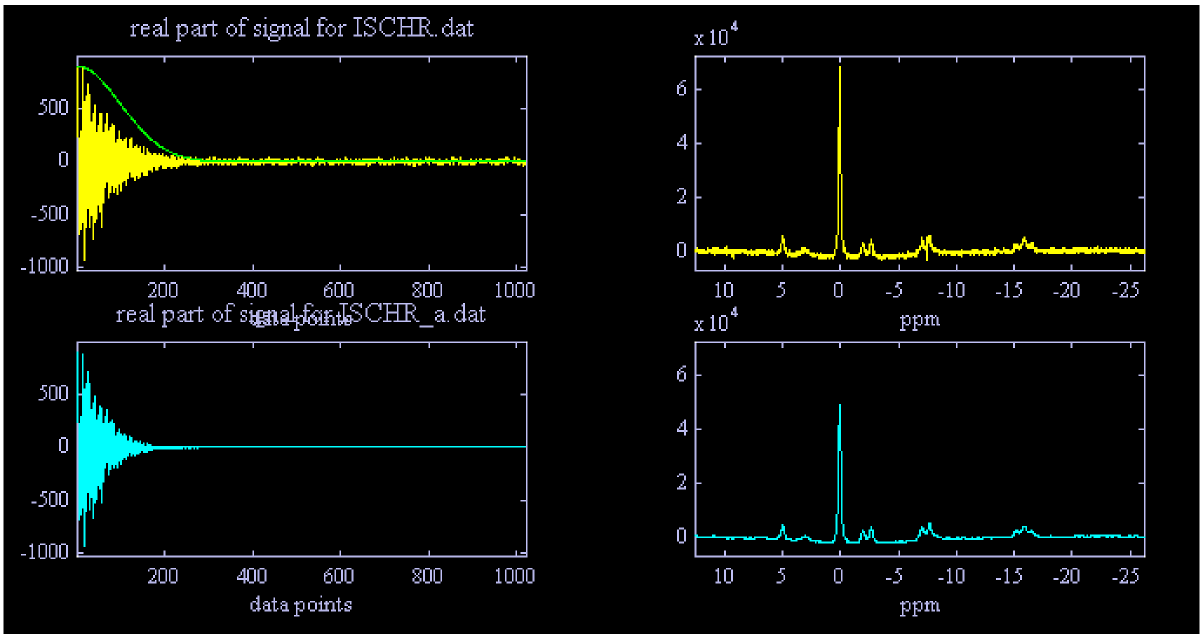


Figure 41. Part of the FID processing window displaying the result of a Gaussian apodization of 4 Hz on the raw FID of ischr.dat.

After having applied the apodization, both menu items are disabled (inactive), so that you can not apply successive apodizations to the same signal in one go, or re-do the apodization with a different value as you could for the 'cosmetic' line broadening. To re-do the apodization on the same or resulting signal, a new file has to be loaded first.

8.11.3 Files produced

Once you are finished using the Apodize function, the following files will have been produced or updated:

```
<name>_a.dat
<name>_a.mat
<name>_a.scl
```

8.12 Multiply & Normalize

These two preprocessing functions share the same FID processing window (see section 5.3) and function-specific (titled Multiply) menu. They allow you to multiply your FID with an arbitrary constant number, or normalize it to an arbitrary maximum value.

The practicalities of using the Multiply and Normalize functions in MRUI are described using the same example data set as before, the file `ischr.dat/mat`. Because only one signal is involved in the Multiply and Normalize processes, the first two rows of the FID processing window are used, and the bottom row remains empty. The result of the Multiply preprocessing method, applied to the raw signal of `ischr.dat` is displayed in Figure 42. The Normalize function works completely analogously, and is effectively a multiplication as well, with the multiplication factor determined from the maximum intensity of the signal and your new normalization value.

8.12.1 by constant

This menu item presents you with an Edit Window in which you can type the multiplication factor. You can enter negative values if you want, but beware that this involves an implicit 180^o phase shift to the signal and spectrum. When you press `[Enter]` after your value, the signal will be immediately multiplied, and the results displayed in the FID processing window. The signals and spectra on the two rows are scaled according to the largest signal. For this example, the signal of the file `ischr.dat` was multiplied by 2. The result is displayed in Figure 42. The name of the new data file containing the processed signal (in this case `ischr_m.dat`) is displayed over the resulting spectrum. If you had loaded a batch, the whole series of files will now be processed using the same multiplication factor.

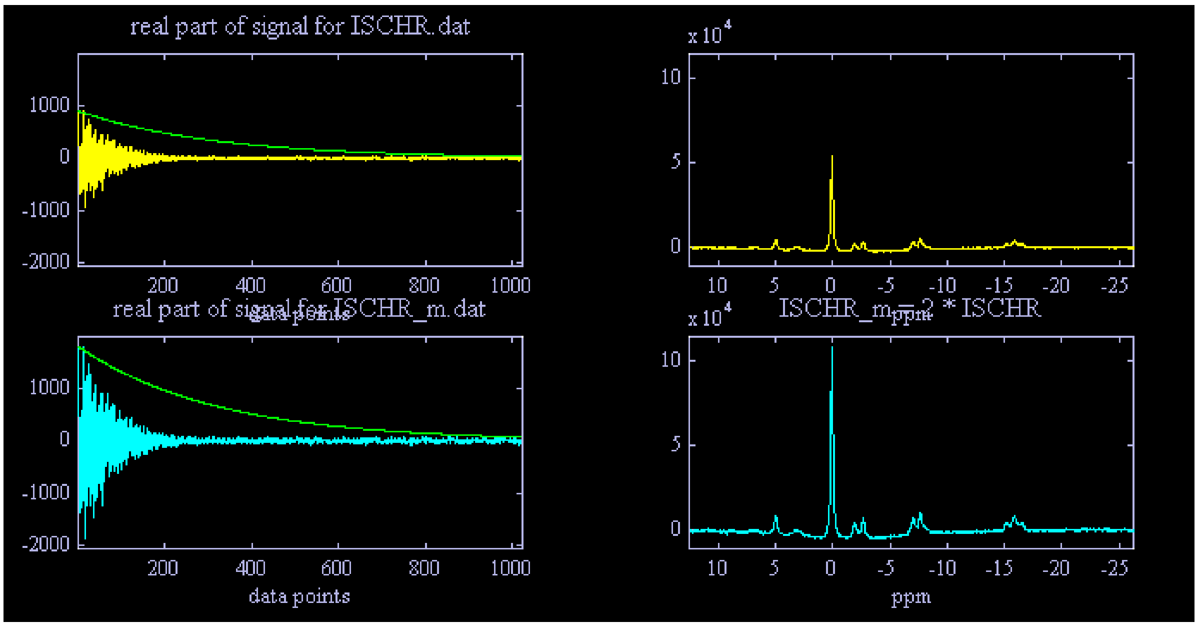


Figure 42. Part of the FID processing window displaying the result of the Multiply/Normalize function: the signal of the file ischr.dat has been multiplied by 2.

8.12.2 normalize to

This menu item presents you with an Edit Window in which you can type the new normalization value. The function will determine the maximum intensity of the current FID (maximum real value of the combined real and imaginary parts of the data points), and set its value to the normalization value you entered. All other data points in the FID will be multiplied accordingly. You can enter negative values if you want, but beware that this involves an implicit 180^o phase shift to the signal and spectrum. When you press `[Enter]` after your value, the signal will be immediately re-normalized, and the results displayed in the FID processing window. The signals and spectra on the two rows are scaled identically, because the signal is only normalized to a different value. This involves a different `scale` and `norm` value being written to the `<name>_n.scl` file (see section 4.2), but the original signal intensities remain unaffected. If you had loaded a batch, the whole series of files will now be processed using the same normalization factor.

After having applied the multiplication or normalization, both menu items are disabled (inactive), so that you can not apply successive multiplications or normalizations to the same signal in one go, or re-do the multiplication or normalization with a different value. To re-do the multiplication or normalization on the same or resulting signal, a new file has to be loaded first.

8.12.3 Files produced

Once you are finished using the Multiply or Normalize function, the following files will have been produced or updated:

Multiply function	Normalize function
`<name>_m.dat`	`<name>_n.dat`
`<name>_m.mat`	`<name>_n.mat`
`<name>_m.scl`	`<name>_n.scl`

9. Filter_HSVD

In many MRS applications, it is desirable to remove large, unwanted resonances before performing accurate parameter estimation on the smaller, neighboring resonances. The most encountered application in this respect is the residual (after water suppression in the pulse sequence) water resonance in ^{1}H MR spectra[4,14,15,16]. Other useful applications would be the removal of a solvent resonances in *in vitro* spectroscopy, large lipid resonances in *in vivo* ^{1}H spectroscopy, the extracellular component of the ^{23}Na resonance[17], or other large resonances that impede the accurate quantitation of complex, low intensity metabolite signals[18]. This can be done using the Filter_HSVD function. This function uses an SVD-based fitting algorithm to reconstruct the time domain signal components corresponding to the unwanted resonances, and subtracting the reconstructed signal from the original signal. The resulting time domain signal no longer suffers from the contributions of the unwanted resonances, and the peaks no longer appear in the spectrum. Because this subtraction is carried out in the time domain, the *entire* signal contribution is removed, which includes the 'tails' of the resonances that might spread throughout the spectrum, giving rise to baseline distortion under the resonances of interest. Because of this, the resonances of interest come down to their 'natural' baseline, without being affected themselves. The characteristics of the Filter_HSVD method will be illustrated using the lineshape corrected *in vivo* ^{1}H MRS signal `abz.dat` from the GE spectrometer file of section 4.5. The full theory of this method is laid out in references[4,15,19]. A brief summary, just enough to understand the basics of the method well enough for proper use, follows now .

As you will read in section 10.1, the SVD-based fitting methods use the Lorentzian lineshape in the model function (exponentially damped sinusoids). However, the residual water resonance (or other resonances such as lipids and solvents) in the spectrum can be far from Lorentzian, and hence cannot be described by one exponentially damped sinusoid in the time domain signal. However, it is known[20] that any arbitrary lineshape can be fitted by the H(L)SVD program, as long as one assigns a sufficient number of Lorentzians (exponentially damped sinusoids) to the non-Lorentzian resonance. In practice, partly due to the effects of noise, it is found that the number of singular values necessary to represent the non-Lorentzian lineshape is rather limited[15]. The following protocol is proposed to remove the water resonance from a ^{1}H spectrum and subsequently subject the resulting signal to an accurate parameter estimation method:

I. With the HLSVD[17] method, a very fast fit of the time domain FID is made, using only a small number of singular values (other SVD-based methods can in principle be used, but they do not have the advantage that they can stop as soon as the required number of largest components has been found, which is the great advantage of this filtering method). This number can be based on a singular value plot (see e.g. Figure 23) by the -relatively slow- HSVD[21] method or on the judgment of the spectroscopist. This is not crucial though, as excess singular values only result in slightly longer calculation time. For *in vivo* ^{1}H MRS experiments, both at short and long echo times, a number of eight to twelve has been found to suffice.

II. The *water region* is to be defined, i.e. the region containing the central frequencies of the water resonance. Conventionally, this is around 4.7 ppm.

III. From the list of estimated parameters per sinusoid (one sinusoid for every non-zero singular value, i.e. the number used in I), the sinusoids with a frequency corresponding to the water region are used to reconstruct the time domain water signal.

IV. This reconstructed water signal is then subtracted from the original time domain FID. In the MRUI package, this resulting FID is stored in a new data file, having the same base name as the original file, but with a _p or p extension, see section 8.1.

V. The resulting FID is now free from the dominating water signal, and can be subjected to parameter estimation programs that can incorporate prior knowledge of the metabolites of interest, such as VARPRO (chapter 13).

The more singular values that are used in step I, the more completely the water resonance will be removed, but the longer the calculation will be. It is found[15] that a relatively small number like three or four will sufficiently clear the large resonances, even though this may leave some spikes (heights similar to metabolite peak heights, but linewidths approaching zero) in the water region, but these no longer have a negative effect on the parameter estimation. If you plot this region with a slightly larger line broadening or without zero filling, these spikes will largely disappear. An important advantage of this protocol is that the entire water resonance is removed, including the long 'tails', without affecting the metabolites that reside upon them, because their central frequencies lie outside the water region. This is an advantage of a time domain analysis method over a frequency domain method when used for this purpose. Correcting for the 'baseline' under these peaks with e.g. a polynomial could not be done without affecting the peaks of interest. A further advantage of the HLSVD algorithm for water removal over other post-processing methods is its flexibility. Where other methods remove all signal contributions around 0 Hz, or between two limits of an interval, the HLSVD protocol actually allows the option of leaving certain components (such as small metabolite peaks) in the region which was defined as the water region in step II. You can do this by imposing multiple regions, one leading up to your resonance of interest, and a second on the other side of it.

It is important that you do not supply too small a number of singular values, because otherwise the entire signal will be described by too few signal components, which may lead to large approximations. One exponential may then be used to describe e.g. both a part of the water resonance, and a closely neighboring peak. When subtracted, the intensity of the neighboring peak will then of course be affected.

The proposed protocol can be applied to other unwanted resonances analogously, but one has to bear in mind that filtering of large unwanted resonances is fast with HLSVD because the largest singular values are found first, and then the program can stop. Should you wish to remove resonances which are of the same order or smaller than your metabolite resonances of interest (which would be unnecessary, especially given the frequency selective features of the VARPRO method, see section 12.4.3), you need to include a larger number of singular values in order to find them, which means considerably longer computation times.

The Filter_HSVD function uses both the basic and result spectral windows, see for its general menus and functions sections 5.1 and 5.2. The function-specific menu is titled Filter. Its menu items are described below, in terms of the algorithm parameters for the protocol described above. Initially, after selecting Filter_HSVD from the processing popup menu of the MRUI base window (see section 3.1), the basic spectral window will be displayed empty, and after loading the file to be filtered, its spectrum will be displayed in it, and the menu items activated (see section 5.1). Then the protocol parameters can be set according to your specific application, after which the filter region can be indicated, and the Filter_HSVD method will do its work. Finally, the spectral window will change size (see section 5.2) to display the Filter_HSVD results and activate the Results menu (section 5.2.1).

9.1 enter region limits

This menu has two submenus, click with mouse and type values in. The selected setting simply defines how the *water region* of step II in the protocol above (or in a broader sense the spectral region containing the central frequencies of the unwanted resonances) will be determined at the moment you activate Filter_HSVD using one of the region(s) menu items (see section 9.4). The default setting can be changed using the customize fitting parameters menu (see section 3.2.1.4). The package default is to define the filter region using the mouse.

9.2 ndp

This is the total number of data points that should be included in the SVD-based fitting process of step I. When storing the time domain data points in the Hankel data matrix for SVD (see section 10.1), `ndp` points are stored. You can change it by typing a new number into an Edit Window. When you load a signal, a default value for `ndp` is set, and displayed in the menu item. This default value is normally taken as the minimum of the actual number of signal points in the data file and the maximum number of data points allowed for the method.. In section 8.2.2, it was explained how the `ndp` data points were stored in a data matrix with Hankel structure. The Filter_HSVD function uses a rectangular matrix as standard, with `ndp/4` rows and `3*ndp/4` columns.
If you wish to subject the resulting FID to VARPRO parameter estimation, it is best that you include all data points (or the maximum number allowed by the VARPRO method) in the Filter_HSVD function. However, if calculation times are of prime importance, you can reduce the number of data points, have the SVD-based method estimate the water components from the beginning of the signal (where these components are largest anyway), and extrapolate until the full length of the original FID, such that after subtraction, the filtered FID has just as many data points as the original FID, regardless the number you set for `ndp`. In this case, be careful that if for some reason SVD has fitted a component with a positive damping factor (this may be possible in short data length fits, see also section 5.2 under *linew*), the extrapolated reconstruction will 'blow up', and the residual signal (i.e. your filtered signal) will suffer likewise. For the current example in which we remove the residual

water resonance from abz.dat/mat, all 2048 data points in the original signal will be used in the HLSVD fitting process (hence with a matrix size of 512x1537).

9.3 nssv

This parameter stands for 'number of signal-related singular values' (see also section 10.1). You can change it by typing a new number into an Edit Window. Its value determines where the singular value decomposition is truncated (subsequent entries in the singular value matrix set to zero). This is also called the *model order*. Remember that the SVD-based fit is carried out in the time domain on the *entire* signal, not only the filter region, so the number of components you enter here must cater for the total number of resonances in the spectrum. However, if your resonance(s) to be filtered are of much larger intensity than the metabolite resonances, you may reduce this number so that it comprises the major resonances only. Beware that highly non-Lorentzian lines require more than one exponentially damped sinusoid, i.e. more than one signal related singular value. If your spectra suffer from overlap, underlying broad components or large noise features, it is safest to slightly overestimate the model order here, in order to cater for unforeseen signal contributions. When you load a signal, a default value for nssv is set, and displayed in the menu item. This default value is taken directly from the value set in the customize fitting parameters menu for auto filter under the Filter_HSVD menu (package default is 10, see also section 3.2.1.4). For the current example, an *in vivo* ^{1}H MR spectrum from human brain at short echo time (see section 4.5), displaying a relatively large number of high intensity resonances, broad components and a highly non-Lorentzian residual water resonance (see Figure 43), nssv was set at 12, which had been found to be consistently sufficient for this type of signals in earlier studies[19].

Because HLSVD is fastest when the components to be removed constitute the largest components of the signal (i.e. largest singular values after which the algorithm can be stopped), Filter_HSVD will be slower once the intensities of the residual water resonance are of the same order of magnitude as the metabolite resonances, as is the case in this example. If the residual water resonance is very prominent in the metabolite spectrum, this might at first glance not be very positive, but in fact it makes its removal by HLSVD all the easier and faster. For this reason, you might want to consider in your experimental setup to make the water suppression pulses not too strong, which at the same time would avoid any distortive influence on closely neighboring metabolite resonances, such as the methylene resonance of Cr/PCr or the lactate quartet near 4.1 ppm. For quantitative purposes, a water suppression scheme that would bring down the water resonance far enough to get good dynamic range behavior, but not too far so that it leaves a prominent residual water resonance, would be optimal.

9.4 region(s)

Selecting one of these menu items starts the Filter_HSVD process, by allowing you to select the spectral region(s) containing the central frequencies of the components you want removed. You can select multiple regions, as explained above. The menu provides items to carry out the Filter_HSVD function with up to 5 regions. Should you for some reason wish to remove resonances from even more non-adjacent spectral regions, select the menu item enter # regions, after which you can type your desired number in an Edit Window, and start off. In this example only one region will be indicated: that around the central frequencies of the residual water resonance. Because this particular residual water resonance is so non-Lorentzian (the different relaxation behavior from brain tissue water and water in CSF caused the negative 'trough' in the center), the region should be taken sufficiently wide, while taking care not to include chemical shifts that might contain central frequencies of resonances of interest. See Figure 43. After having selected the relevant region(s), Filter_HSVD will carry out the whole process immediately. How you select the spectral region(s) depends on the settings of the enter region limits menu item (see section 9.1). Usually, the latter will be set to click with mouse, which means that you are supposed to click once to the left of each region, and then to the right of that region (or the other way, the minimum and maximum are automatically determined from the two entered values). See Figure 43 A message text bar will appear in the bottom right corner, telling you what is expected from you, and blue vertical lines will be displayed where you have selected your region limits. You can use the customize preferences menu to disable the blue lines and/or message text in order to work faster, see section 3.2.1.3.. If the enter region limits menu item was set to type values in, you will be presented with an Edit Window twice for each region, asking you for the low and high frequency limit of that region, which you must type in the units as displayed on the frequencies (e.g. in ppm's in this example). In this case the blue vertical lines will also be displayed, but the message text bar will not appear. If you had loaded a batch, the whole series of files will be subjected to the Filter_HSVD method using the same selected spectral region(s).

If you have a number of files in the same directory that all require filtering, and you cannot process them in batch, it is wise to first do all the Filter_HSVDs on all the files, and then start doing the parameter estimation on all the result files. This is because for Filter_HSVD you will usually zoom in on a different region of interest in the spectrum than you would on the filtered spectrum for e.g. VARPRO fitting. This way you would have to re-zoom only once for the whole series, instead of once for each data file.

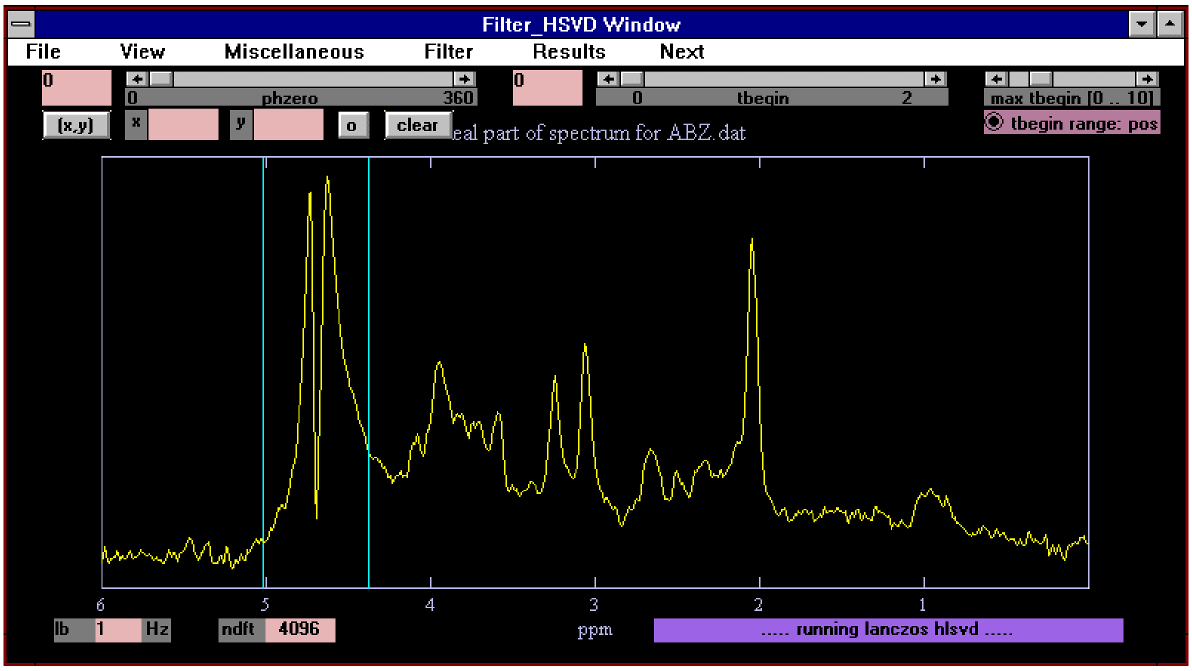


Figure 43. The Filter_HSVD basic spectral window, after indicating the spectral region(s) (1 in this case) containing the central frequencies (read time domain characteristics of the method in the accompanying text) of the resonance(s) that you wish to have removed from the signal (and consequently its FT spectrum). The entire signal contribution, corresponding to the central frequencies in the selected region(s) will be subtracted, which means that in the FT spectrum the entire component will disappear, inclusive of its dispersive 'tails' that might run over a much wider chemical shift range, possibly underlying metabolite resonances of interest (such as the 3.96 ppm methylene resonance of Cr/PCr in this figure.

9.5 auto filter

This menu item allows you to fully automate the Filter_HSVD technique, provided you use only one region for filtering, by setting the appropriate fixed value for the number of signal related singular values (see section 9.3) in the first submenu item (nssv), and the default spectral region to be filtered in the second (region; if you select this menu item you are presented with an Edit Window twice, as explained in the previous section), and then pressing the go! submenu item. Of course, if you would have to set these values each time before pressing go! under auto filter, it would not be very automatic, so you can set the default number for nssv and the spectral region using the customize fitting parameters menu, see also section 3.2.1.4. The package default is 10 signal related singular values, and a spectral region between 25 and -25 Hz. The default values from the customize fitting parameters menu are automatically filled in in these submenu items under the auto filter menu of the Filter_HSVD function. If you use the Filter_HSVD technique regularly on similar data sets, for example to remove the residual water resonance around 0 Hz, you can set your default spectral region using the customize fitting parameters menu accordingly, and a number of components that is relevant to your type of data, and from that moment on you can Filter_HSVD your signals simply by loading them into the function and selecting go! under auto filter.

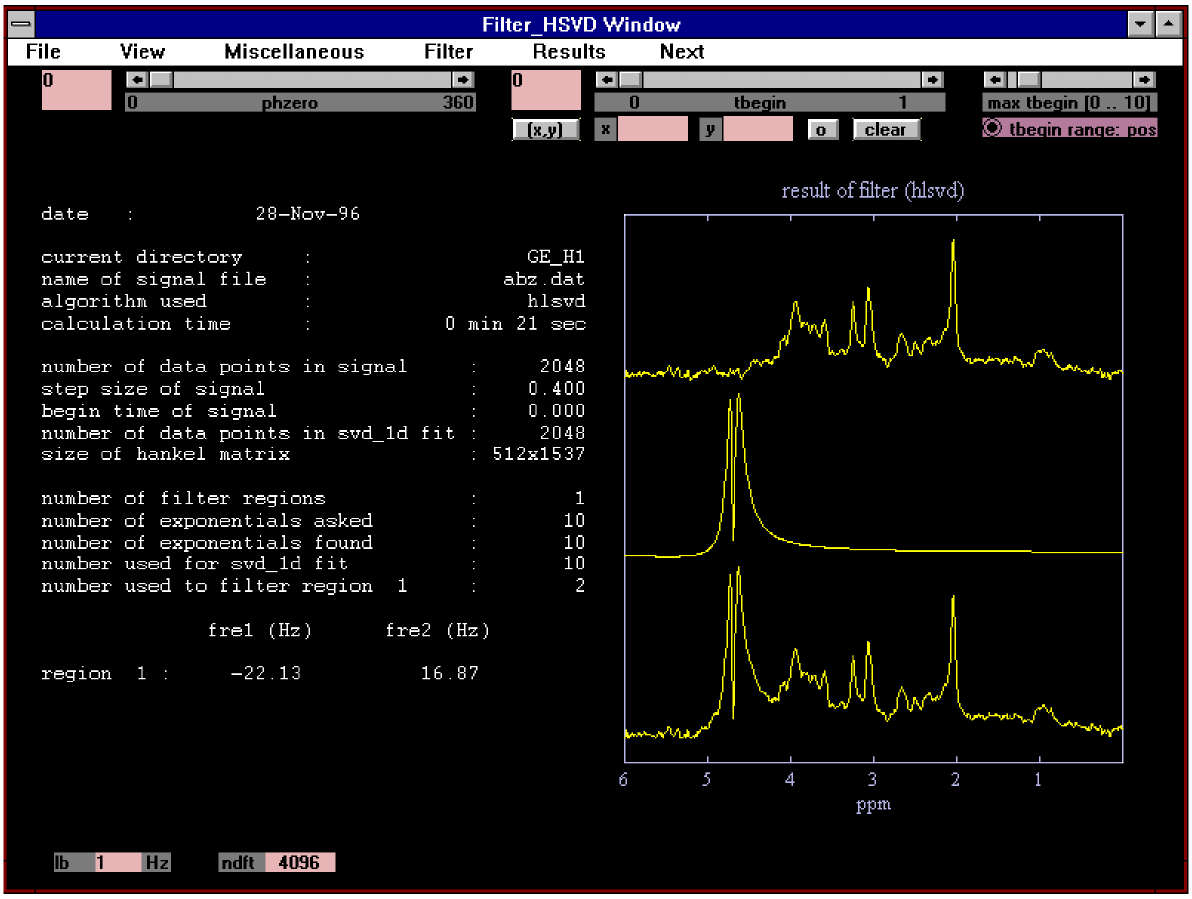
9.6 algorithm

With this menu item you can set the State Space / SVD algorithm that will be run for the SVD-based fitting process of step I in the protocol above. The most efficient SVD-based method for water filtering is definitely the HLSVD[15,17] method (lanczos hlsvd), because it can stop after a predetermined number of singular values (nssv) have been found. Especially for water removal, where the signal components to be removed are largest (the largest singular values are found first), this can yield large computation time gains. However, you are free to use the conventional normal equations approach for State Space, with either the Nhous HSVD[21] method (nhous hsvd), or a HSVD algorithm programmed in Matlab[†]. When trying to filter out small resonances (even though you know there is no reason to do so, see the frequency selective feature of the VARPRO method in section 12.4.3), you might wish to use either one of the latter two methods for more robust SVD fitting. You can change the default setting for this submenu using the customize fitting parameters menu for algorithm under the Filter_HSVD menu (package default is lanczos hlsvd, see also section 3.2.1.4). For the current example there was no reason to change this.

9.7 Results

After the SVD-based parameter estimation has been carried out on the signal, the components that have an estimated central frequency within (one of) the selected spectral region(s) will be used to reconstruct a time domain model function, which represents the signal contributions of the unwanted resonances. This reconstructed signal will then be subtracted from the original signal (steps III and IV in the above protocol). The filtered signal is the difference between the original signal and the noiseless signal reconstructed from the estimated unwanted resonances. The spectral window changes size in order to display the analysis results both as a numerical table and a multi-spectra display, see also section 5.2. The result spectral window for the current example is displayed in Figure 44.

[†] Kindly provided by ir. Frank Wajer, of the University of Technology Delft (NL), Applied Physics laboratory, Magnetic Resonance Imaging group.



```
date   :           28-Nov-96

current directory      :                      GE_H1
name of signal file    :                    abz.dat
algorithm used         :                      hlsvd
calculation time       :               0 min 21 sec

number of data points in signal       :        2048
step size of signal                   :       0.400
begin time of signal                  :       0.000
number of data points in svd_1d fit   :        2048
size of hankel matrix                 :    512x1537

number of filter regions              :           1
number of exponentials asked          :          10
number of exponentials found          :          10
number used for svd_1d fit            :          10
number used to filter region  1       :           2

              fre1 (Hz)      fre2 (Hz)

region  1 :     -22.13          16.87
```

Figure 44. When Filter_HSVD has finished, it displays the results in the result spectral window, using the numerical information list on the left, and the multi-spectra display on the right. By default, only the original and residual (filtered) spectrum are displayed. For this example the HLSVD reconstruction of the unwanted residual water resonance was also displayed to indicate how non-Lorentzian composite lines can be mathematically described by a sum of Lorentzians to allow complete subtraction. The table lists the lower and upper frequencies in Hz of the selected spectral region(s).

The table on the left displays some information about the data file, and the Filter_HSVD process. It lists the number of singular values asked, found, and used for the actual fit. For Nhous HSVD and Matlab HSVD, these three numbers are always identical, but due to its iterative algorithm, Lanczos HLSVD may find more or less singular values than you asked for[17]. In such a case, the minimum of the number asked and the number found is used for the fit. For each selected region, the table lists how many components (number of singular values is number of fitted components) were used to represent, and then remove, the unwanted resonances in that region. Finally, the table lists for each selected region the upper and lower frequency limit in Hz. If you usually select a region manually before each Filter_HSVD, you may check the values of the upper and lower limits you have indicated each time, and use these to set a reasonable default for the auto filter feature via the customize fitting parameters menu (see section 9.5). The current region for this example was selected between +16 and -22 Hz, which means a default water removal region from 25 to -25 Hz would have been just fine. Notice how only 2 components were used to filter the non-Lorentzian residual water resonance completely. If you would look at the individual components of the reconstructed water resonance, you would see that two Lorentzians with almost opposite phases mathematically described the water resonance, even though the individual components themselves no longer have a physical meaning[15]. As for the spectral display of the results, the package default displays the original spectrum at the bottom, and the filtered (residual after HLSVD fit) spectrum at the top, with all spectra scaled to the filtered spectrum. HSVD fit spectrum and individual components are not displayed by default. You can change the spectral display using the Results menu (see section 5.2.1), or change the default settings using the customize preferences menu (see section 3.2.1.3).

9.8 Files produced

Once you are finished using the Filter_HSVD function, the following files will have been produced or updated:

file produced	contents
`<name>.siv`	singular values found by HSVD on `<name>.dat`
`<name>.tsv`	information on number of singular values found at each update of Lanczos iterations (only for lanczos hlsvd)
`<name>.scr`	simple parameter estimation output file from H(L)SVD method (formerly known as `hsvscr.par` file)
`<name>.hrs`	parameter estimation output file with better layout and Cramér Rao standard deviations with parameters (formerly known as `outputh.par`, obtained after running `errorh`)
`<name>.hns`	presentation file of numerical parameter estimation results, containing output of `<name>.hrs` file, but with estimated frequencies in units as displayed in the spectral display on screen (e.g. ppm), damping factors translated to linewidths at half height in Hz, and with amplitude estimates and CR values scaled back to original intensities
`<name>.lrf`	filter analysis report, directly taken from the list in the left half of the result spectral window
`<name>.rew`	Matlab format file containing the individual time domain signal components constituting the filtered resonances. This file may be helpful if you later wish to display these individual components by loading a result into Filter_HSVD. However, these files may become quite large, depending on the situation. Whether they are saved to disk or not, can be set using save lorentzians to file item under the Filter_HSVD menu in the customize fitting parameters window.
`<name>_p.dat`	Rdb format data file containing filtered signal
`<name>_p.mat`	Matlab format data file containing filtered signal
`<name>_p.scl`	scale file for the `<name>_p.dat/mat` data file

10. SVD_1D parameter estimation

As indicated in section 3.3, you can carry out your MRS quantitation almost fully automatically using ‘black-box’ parameter estimation methods based on SVD, if the signals are well-conditioned (high SNR, low to moderate overlap, etc.).

10.1 SVD-based parameter estimation methods

A number of such SVD-based methods has been included in the `SVD_1D` function of the MRUI software. These methods solve the least squares (LS) problem of fitting the time domain model function using matrix algebra, i.e. non-iteratively, by making full use of the mathematical properties of the exponential decay model (this is only valid for the Lorentzian decay). See references[3,20]. To this end, the `ndp` measured data points x_0,...x_{ndp-1} are stored in a data matrix **X** of size `mxl` as follows:

$$\mathbf{X} = \begin{pmatrix} x_0 & x_1 & x_2 & . & . & . & x_{l-1} \\ x_1 & x_2 & . & . & & & . \\ x_2 & . & . & & & & . \\ . & . & & & & & . \\ . & . & & & & & . \\ . & . & & & & & . \\ x_{m-1} & . & . & . & . & . & x_{ndp-1} \end{pmatrix}$$

The data matrix has so-called *Hankel* structure: all elements on an anti-diagonal are identical; `m+l=ndp-1`. In all methods, the Hankel data matrix is subjected to singular value decomposition (SVD) in order to estimate the signal poles: the exponential term in the model function containing the damping factors and frequencies. After the damping factors and frequencies have been estimated using the particular form of matrix algebra characteristic to the selected method, their values are filled in into the model function, and the amplitudes and phases can be estimated in one linear LS round.

The SVD-based methods available in the MRUI function `SVD_1D` and their respective characteristics and references are:

method	characteristics	reference(s)
LPSVD	Original LPSVD method, improved with a statistical method for root selection and a Simplex routine for the polynomial rooting process.	3,7,20,22
LPSVD(CR)	LPSVD method as outlined above, but implementing a continuous regularization scheme that yields a lower failure rate and allows automatic determination of the model order (number of signal components).	8,23
Nhous HSVD	Original State-space HSVD algorithm, calculating a full SVD of the data matrix via a robust triangularization scheme and the normal equations approach.	3,20,21
Matlab HSVD	Matlab implementation of the State-space method.	†
HTLS	Improved and more accurate State-space algorithm, implementing total least squares (TLS) instead of a single SVD in order to estimate the signal poles.	10,24
Lanczos HLSVD	Improved and accelerated version of the HSVD method, which distinguishes itself by its ability to calculated only a desired number of singular values (using a Lanczos recursion scheme) instead of having to	3,15,17,20

† Written by ir. Frank Wajer, of the University of Technology Delft (NL), Applied Physics laboratory, Magnetic Resonance Imaging group.

perform the SVD of the entire data matrix. Combined with its intelligent exploitation of the Hankel symmetry of the data matrix, it makes for the fastest SVD-based method if only the signal components are to be estimated (if you would want to estimate noise features as well, the Nhous HSVD or HTLS method is recommended)

The practicalities of using the SVD_1D function in MRUI are described using the same example data set as before, the file `ischr.dat/mat`. The SVD_1D function uses both the basic and result spectral windows, see for its general menus and functions sections 5.1 and 5.2. The function-specific menu is titled SVD. Its menu items are described below, in terms of the algorithm parameters for the methods described above. Initially, after selecting SVD_1D from the quantitation popup menu of the MRUI base window (see section 3.1), the basic spectral window will be displayed empty, and after loading the file to be analyzed, its spectrum will be displayed in it, and the menu items activated (see section 5.1). Then the protocol parameters can be set according to your specific application, after which you can press go!, and the selected method in the SVD_1D function will do its work. Finally, the spectral window will change size (see section 5.2) to display the SVD_1D fitting results and activate the Results menu (section 5.2.1).

10.2 ndp

This is the total number of data points that should be included in the SVD-based fitting process. These `ndp` points are stored Hankel data matrix for SVD (see above). You can change it by typing a new number into an Edit Window. This number should be taken such that the FID has completely decayed into the noise level, but not too far beyond that point (or you would be involving ‘noise-only’ points in the matrix). As a rule of thumb, select `ndp` such that `ndp * step` $\cong$ `3 * T`$_2$, where `step` is the sampling time step (or dwell time, see e.g. sections 4.1.1,6.2.6), and `T`$_2$ the longest relaxation time present in the signal. To ensure reasonable computation times, `ndp` should not be set too large. When you load a signal, a default value for `ndp` is set, and displayed in the menu item. This default value is normally taken as the minimum of the default value set in the customize fitting parameters menu for the selected SVD algorithm (package defaults are 1024 for the rapid Lanczos HLSVD, with smaller numbers for the slower methods: 512 for the HSVD and HTLS methods, and 256 for the LP methods, see also section 3.2.1.4), the actual number of signal points in the data file, and the maximum number of data points allowed for the method. If you change the value for `ndp`, it will again be tested against those maxima. For the example signal of `ischr.dat`, it could be seen in the FID plot (see Figure 55 in section 16.1) that 256 data points may be just sufficient to include the signal contribution and exclude noise-only points, but `ndp` was set to be 512 in order to make sure that all signal contributions were involved, and to ensure proper resolving power of the SVD methods. With the choice of the Lanczos HLSVD method (see section 10.9), this number does not pose any problems as to calculation times.

10.3 matrix

This is the size of the Hankel data matrix in which the time domain data points are stored, and which will be subjected to the SVD. You can change it by typing a new number of columns into an Edit Window. The number of columns and the number of rows should at least be larger than the number of spectral components (model order of the signal; truncation point[3,20] in SVD; `nssv` parameter, see next section). For the H(L)SVD and HTLS methods, the matrix should be flat (`l > m`), with most robust performance for a square matrix, and shorter computation times for flatter matrices. For the LPSVD and LPSVD(CR), the matrix has to be tall rather than long (`l < m`), with a recommendation for a tall matrix. When you load a signal, a default matrix size is set, and displayed in the menu item. This default size is normally taken from the default number of columns set in the customize fitting parameters menu for the selected SVD algorithm (package defaults are a square matrix for Lanczos HLSVD, a flat matrix with `l:m=3:1` for the HSVD and HTLS methods, and a tall matrix for the LPSVD and LPSVD(CR) methods, see also section 3.2.1.4), but corrected corresponding to the value of `ndp` if necessary. If you change the size of the matrix, it will again be tested against the appropriate minima and maxima. For the signal of `ischr.dat`, to be fitted by the Lanczos HLSVD method, the matrix was left square: 256x257.

10.4 nssv

This parameter stands for 'number of signal-related singular values'. You can change it by typing a new number into an Edit Window. Its value determines where the singular value decomposition is truncated (subsequent entries in the singular value matrix set to zero) before estimation of the signal poles takes place[3,20]. This is also called the *model order*. This value should be set according to the number of spectral components. If your spectra suffer from overlap, underlying broad components or large noise features, it is safest to slightly overestimate the model order here, in order to cater for unforeseen signal contributions. When you load a signal, a default value for `nssv` is set, and displayed in the menu item. This default value is normally taken as the minimum of the default value set in the customize fitting parameters menu for the selected SVD algorithm (package default is 12 for the H(L)SVD and HTLS methods, and 11 for the LP methods, see also section 3.2.1.4), the number of data points divided by 4, and the maximum number of singular values allowed for the method. If you change the value for `nssv`, it will again be tested against those maxima. For the signal of `ischr.dat`, it could be seen in the spectral window of the SETUP: experimental function (see Figure 31 in section 7.1) that 11 signal-related components (7 for ATP, 4 for PCr, PDE, P_i and PME) is a good number, however it turned out that at `ndp=512`, 15 components had to be included in order to resolve the βATP triplet. The 'extra' singular values may account for either NAD, noise or baseline features, or PCr lineshape distortion. In this example, `nssv` was hence set to 15. The `nssv` menu item is inactive (disabled) when the selected algorithm is LPSVD(CR), because that method automatically determines the number of signal-related singular values[8] on the basis of the noise level (see also section 10.6).

10.5 nit

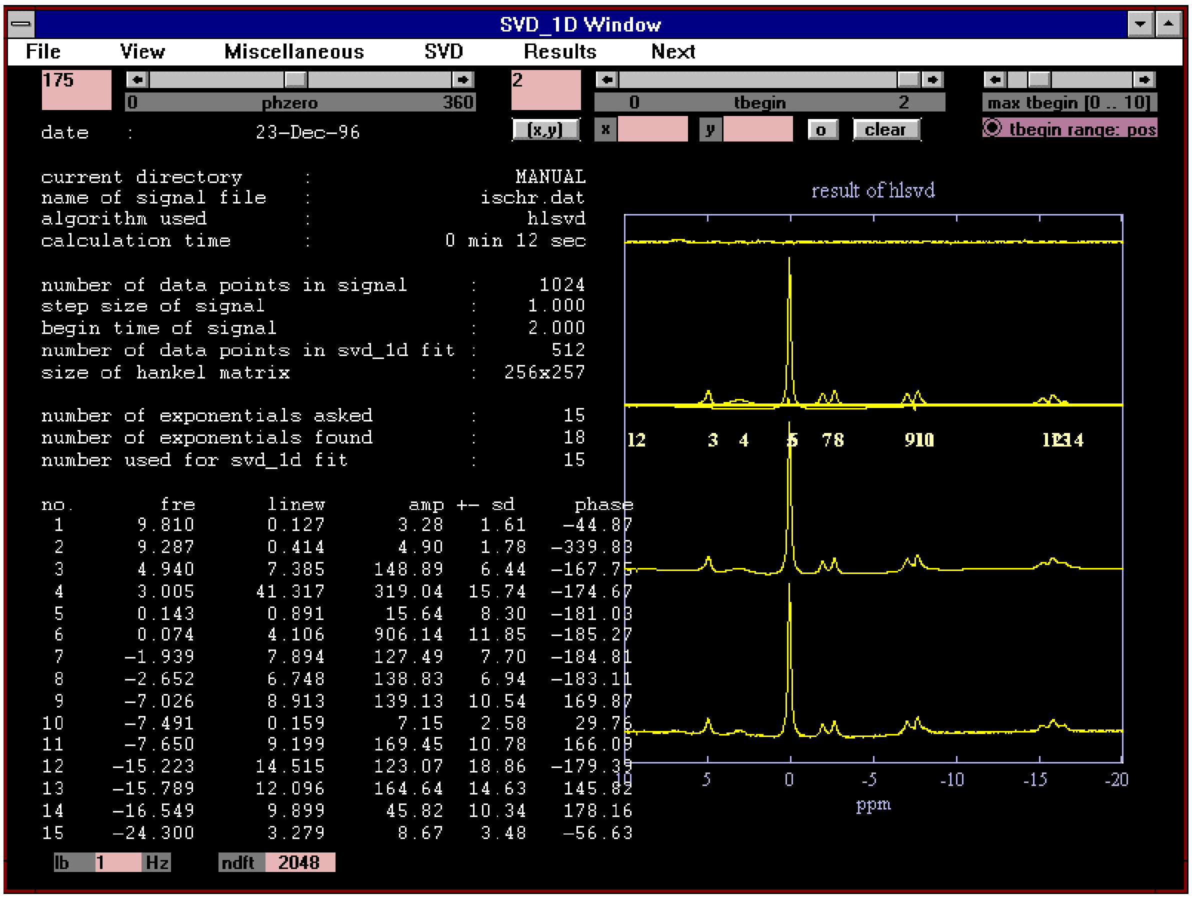
This menu item is only enabled (activated) when the Lanczos HLSVD algorithm is selected, as the latter uses the iterative Lanczos recursion scheme for determination of the singular values[17,25]. `nit` is a starting value for the number of Lanczos recursions[17,25] , performed to determine the requested `nssv` signal related singular values in the Lanczos HLSVD method. At the end of `nit` Lanczos iterations, it is checked whether the required number of singular values has been found, and if not, an additional `nit/10` Lanczos iterations are performed, a new check carried out, etc. You can change it by typing a new number into an Edit Window. When you load a signal, a default value for `nit` is set, and displayed in the menu item. This default value is normally taken as 10 times the requested number of singular values `nssv`[25], but taking into account the default value set in the customize fitting parameters menu for Lanczos HLSVD (package default is 120, which is 10 times the package default for `nssv`, see also section 3.2.1.4), and the maximum number of iterations allowed for the method. You can change it by typing a new number into an Edit Window. If you change the value for `nit`, it will again be tested against its maximum. If you change the value of `nssv`, the value for `nit` will automatically be updated according to `10*nssv` (taking into account its maximum).

10.6 noise level

This noise level submenu items are only enabled (activated) when the LPSVD(CR) algorithm is selected, as the latter needs a standard against which to determine whether a singular value belongs to a signal component or a noise feature[8,23]. When you load a signal, a default submenu item is set. This default value is taken from the default option set in the customize fitting parameters menu for LPSVD(CR) noise level under the SVD_1D menu (package default is from last data points: 50, see also section 3.2.1.4). With this (recommended) default setting, the method determines the standard deviation of the noise from the last data points of the FID (how many final data points are used in this calculation can be set by selecting the from last data points submenu item or in the customize fitting parameters menu). These really are the last data points of the full FID stored in the data file, not the last points from the length of `ndp` points you selected to include in the LPSVD(CR) fit. However, should the end of your FID still contain signal contributions (FID not fully decayed), you can enter a fixed value for the standard deviation of the noise.

10.7 go!

After all the algorithm parameters have been set according to your wishes, select the go! menu item to start the SVD-based fitting and parameter estimation. If you had loaded a batch, the whole series of files will now be processed using the same algorithm parameters. The basic spectral window will display a message ("running <selected algorithm>"; with serial index for batch processing) in the text bar in the lower right corner of the window, and after it has finished it will display the SVD_1D results both as a numerical table and a multi-spectra display, using the result spectral window (section 5.2). See Figure 45.



```
date          :             23-Dec-96

current directory          :              MANUAL
name of signal file        :            ischr.dat
algorithm used             :                hlsvd
calculation time           :       0 min 12 sec

number of data points in signal         :     1024
step size of signal                     :    1.000
begin time of signal                    :    2.000
number of data points in svd_1d fit     :      512
size of hankel matrix                   :  256x257

number of exponentials asked            :       15
number of exponentials found            :       18
number used for svd_1d fit              :       15

no.        fre       linew        amp +- sd       phase
 1       9.810      0.127      3.28    1.61    -44.87
 2       9.287      0.414      4.90    1.78   -339.83
 3       4.940      7.385    148.89    6.44   -167.75
 4       3.005     41.317    319.04   15.74   -174.67
 5       0.143      0.891     15.64    8.30   -181.03
 6       0.074      4.106    906.14   11.85   -185.27
 7      -1.939      7.894    127.49    7.70   -184.81
 8      -2.652      6.748    138.83    6.94   -183.11
 9      -7.026      8.913    139.13   10.54    169.87
10      -7.491      0.159      7.15    2.58     29.76
11      -7.650      9.199    169.45   10.78    166.09
12     -15.223     14.515    123.07   18.86   -179.39
13     -15.789     12.096    164.64   14.63    145.82
14     -16.549      9.899     45.82   10.34    178.16
15     -24.300      3.279      8.67    3.48    -56.63
```

Figure 45. SVD_1D result spectral window after Lanczos HLSVD parameter estimation on ischr.dat. Although only 11 metabolite resonances are apparent, 15 singular values (signal-related components) needed to be included in the SVD_1D fit in order to resolve the βATP triplet. Even so, the PME resonance has not been fitted, whereas spurious features have been included in the fit (see the individual components in the spectral display on the right).

The table on the left displays some information about the data file, and the SVD_1D process. It lists the number of singular values asked, found, and used for the actual fit. For Nhous HSVD, Matlab HSVD, HTLS and LPSVD, these three numbers are always identical, but due to its iterative algorithm, Lanczos HLSVD may find more or less singular values than you asked for[17]. In such a case, the minimum of the number asked and the number found is used for the fit. The number LPSVD(CR) lists as singular values found, is in fact an estimation result specific to the method[23]. The display of the numerical table and spectra has been explained in section 5.2. The results displayed here after Lanczos HLSVD parameter estimation on the signal of ischr.dat can be seen as rather general for the SVD_1D methods. In this case, the results of the other SVD-based methods would have been only marginally different, except of course for the calculation time, which is significantly shorter for the Lanczos HLSVD methods (the other SVD-based methods took minutes where Lanczos HLSVD needed only 12 seconds, see Figure 45). Only for slowly decaying FIDs, or closely overlapping multiplets, where large numbers of data points are required to resolve all peaks, will the Lanczos HLSVD yield increased capabilities, as the other methods grind down when ndp is set too large.

As for the current example, the power and the weaknesses of the SVD_1D function (whichever algorithm selected) for parameter estimation become apparent. It may be regarded as an advantage that simply by indicating the number of data points and number of signal-related components (both parameters can be set by default in the customize fitting parameters menu in order to fully automate the SVD_1D function), one obtains a complete parametric analysis of the signal, with resonance frequencies, linewidths, amplitudes (directly proportional to peak areas) and phases for each peak in the spectrum. However, there are a number of serious weaknesses of this type of method in relation to the physical relevance of the estimated parameters when the signal-to-noise ratio or resolution of certain peaks becomes poor. In this example, it can be seen that the resonance of PME (phosphomonoesters, near 6.6 ppm) has a very low intensity, while being rather broad, which renders its SNR very low. Even when providing the SVD algorithm with 15 signal-related singular values, the PME peak was not fitted (see Figure 45). This is the disadvantage of the 'black-box' method in that it does not allow control over exactly which resonances are fitted. Also, the individual phases of some of the fitted resonances, especially in the ATP multiplets, can be seen to differ rather significantly, which is caused by the mathematical properties of the algorithm trying to achieve an optimum fit with minimum root mean square (RMS), without the possibility to apply constraints. Of course such diverging phases within an ATP multiplet are not physically relevant for the applied measurement method. Similarly, the ratios of the amplitudes (or peak areas) of the ATP multiplets, especially the βATP triplet, do not conform to our expectations. Finally, some spurious narrow peaks have been fitted, of which it is difficult to understand why they have been fitted in preference to other, visually more prominent, signal contributions. It is this lack of control over the parametric fit that renders the SVD_1D methods rather vulnerable under *in vivo* MRS measurement conditions. For this reason the use of the VARPRO method (see chapter 13), which allows control over which peaks are fitted (by supplying appropriate starting values using Peakpick, see chapter 11), and over physical relations that are valid for certain groups of peaks (by supplying prior knowledge using Inputvp, see chapter 12).

If the SVD_1D analysis would have given estimated parameters so large, or with such large standard deviations (e.g. when the fitting was unsuccessful), that the numerical values could not fit into the prepared spaces in the result files, character strings, such as `********` or NaN or `Inf`, would be written to the result files. MRUI detects these characters, and appropriate measures are taken. If standard deviations are so big that they do not fit, the fit was unsuccessful, and the standard deviations are set to `Inf` (infinity) in the numerical result table and in the `<name>.hns` file. As for the parameters, their estimated values are *always* recovered (via the `<name>.scr` file, see section 10.11).

SVD_1D checks after the analysis whether all damping factors are smaller than zero. If this is the case (physical solution) the reconstructed model function is extrapolated to the number of data points present in the original signal (if of course larger than the value for ndp). This avoids heavy sinc wiggles throughout the displayed reconstructed spectrum when fitting slowly damped signals with a small number of data points.

The SVD_1D function can also calculate a value for the signal-to-noise ratio (SNR) of the time domain signal in dB, calculated from the estimated amplitudes and damping factors, and the standard deviation of the noise, via the ^{10}log of (energy in signal over energy in noise). This may be used to compare signals according to time domain fitting difficulty. The SNR values in dB are stored in the `<name>.hns` file (see section 10.11). Be aware that this calculation may make your overall SVD_1D analysis times considerably longer. You can define whether or not SVD_1D should calculate the SNR in the customize fitting parameters menu for calculate SNR (dB) under the SVD_1D menu (package default is no, see also section 3.2.1.4).

SVD_1D provides the same facilities for absolute quantitation and calculation of pH (see also section 10.10) as VARPRO, see sections 13.4 and 13.6, respectively. This may require the following actions from you after the result spectral window has been displayed:

- If the nucleus set in `default.xpt` is proton (h, see section 7.1.1), and no logical corresponding water file can be found, you will be suggested to indicate an appropriate water file for the internal concentration reference. This is done via a file manager window. See section 13.4. You also need to indicate which peaks correspond to the metabolite you wish to quantify.
- If the nucleus set in `default.xpt` is phosphorus (p, see section 7.1.1), and you have selected one or more of the pH calculation options (either through the calculate pH menu of the SVD_1D function or via the customize fitting parameters menu), you will be suggested to indicate which peaks in the reconstructed spectrum represent the relevant metabolite resonances. See section 13.6.

10.8 water

If a loaded (batch of) signal(s) contains (an) unsuppressed water signal(s), of which the amplitude needs to be determined, selecting the water menu automatically sets the SVD parameters (all data points used, square matrix, nssv=1) correctly and runs the SVD_1D analysis. Apart from the usual result files, an extra ASCII file, <name>.wat, will be produced, which simply contains the amplitude and linewidth of the unsuppressed water signal. These can later be used for absolute quantitation (see section 13.4) or to provide an overall linewidth starting value for VARPRO (see section 11.4.2). However, it is stressed that the latter functions contain an automatic procedure to carry out the SVD fit on an unsuppressed water signal if no <name>.wat is found. The water menu item in the SVD_1D function is provided simply to allow you some more visual control over the fitting of the unsuppressed water resonances.

10.9 algorithm

This menu item lets you select which SVD-based algorithm is used for the SVD_1D fit. See the table in section 10.1. The default algorithm can be set using the algorithm menu item under the SVD_1D menu in the customize fitting parameters window (package default is lanczos hlsvd). Because the LAPACK subroutines were found to give run-time problems on PCs, the htls option is disabled (inactive) when running MRUI on PC.

10.10 calculate pH

This menu item is only enabled (active) when the nucleus set in default.xpt is phosphorus (p, see section 7.1.1). It allows you to calculate intra- or extracellular pH based on the chemical shift differences between P_i and αATP, P_i and PCr, or APP (extracellular marker) and αATP. The use of this menu is explained in section 13.6.

10.11 Files produced

Once you are finished using the SVD_1D function, the following files will have been produced or updated:

file produced	contents
<name>.siv	singular values found by SVD on <name>.dat
<name>.tsv	information on number of singular values found at each update of Lanczos iterations (only for Lanczos HLSVD)
<name>.scr	simple parameter estimation output file from SVD-based method (formerly known as hsvscr.par file)
<name>.hrs	parameter estimation output file with better layout and Cramér Rao standard deviations (listed in table as *two times* CR!) with parameters (formerly known as outputh.par, obtained after running errorh)
<name>.hns	presentation file of numerical parameter estimation results, containing output of <name>.hrs file, but with estimated frequencies in units as displayed in the spectral display on screen (e.g. ppm), damping factors translated to linewidths at half height in Hz, and with Cramér Rao standard deviations listed as one sd, i.e. parameter ± CRsd also contains an extra table with amplitude estimates and CR values scaled back to original intensities if carried out, this file also contains absolute metabolite concentrations, pH estimations and/or SNR in dB
<name>.lrs	SVD_1D analysis report, directly taken from the list in the left half of the result spectral window
<name>.wat	auxiliary ASCII file containing estimated amplitude and linewidth of water signal, either if an unsuppressed water line was estimated via the water menu item, or if absolute concentration was carried out and no <name>.wat file found at the time

Running SVD_1D in batch produces extra files which list the estimated parameters for all peaks in all files of the batch series as follows:

file produced	tabulates	corresponding file	tabulates
`<name>.amp`	estimated amplitudes (in arbitrary units)	`<name>.ams`	Cramér Rao standard deviations on estimated amplitudes
`<name>.aam`	absolute amplitudes (as true intensities in the FIDs, i.e. corrected for scaling)	`<name>.aas`	Cramér Rao standard deviations on absolute intensities
`<name>.fre`	estimated frequencies (in units as displayed on screen, e.g. Hz or ppm)	`<name>.frs`	Cramér Rao standard deviations on frequencies
`<name>.lw`	estimated linewidths (in Hz)	`<name>.lws`	Cramér Rao standard deviations on linewidths
`<name>.pha`	estimated phases (in degrees)	`<name>.pas`	Cramér Rao standard deviations on phases
`<name>.ph`	estimated pH (if calculated)	`<name>.phs`	Cramér Rao standard deviations on pHs
`<name>.cnc`	absolute metabolite concentrations (if calculated)	`<name>.cns`	Cramér Rao standard deviations on absolute concentrations
`<name>.wal`	estimated amplitude and linewidth for unsuppressed water signal (single Lorentzian)		

The first column contains the serial numbers of each batch-processed file, and the following columns contain the respective parameter for all peaks. You can directly load these ASCII column files into a spreadsheet or graphics program. The serial batch results can be plotted using Plot batch results (see section 16.3). The files containing the Cramér Rao estimates for the standard deviations on the estimated parameters can be used in your spreadsheet software in order to create graphs with error bars.

11. Starting values for VARPRO/AMARES: Peakpick

As explained in section 10.7, it is for most *in vivo* MRS applications desirable to make use of a nonlinear least squares method that allows you to control certain aspects of the fitting and parameter estimation process, such as which components are to be fitted (and which are not), and relations between certain model parameters on the basis of physically relevant, biomolecular or experimental, prior knowledge[26]. The VARPRO/AMARES method provides this function, but the nonlinear least squares VARPRO/AMARES function needs to be preceded first by Peakpick (to supply starting values) and Inputvp (to supply prior knowledge). In principle, when minimizing the difference between the sum of squares of the model function and the measured time domain data, starting values need to be supplied for all parameters, in order to provide a starting condition for the iterative least squares search to the global (or deepest local in the vicinity) minimum. However, for the variable projection method, Golub and Pereyra[27] proved that the nonlinear and linear parts of the model function could be separated, and the entire fitting process split into one iterative nonlinear least squares in which the nonlinear parameters (frequencies and damping factors) were fitted, and a subsequent one-round linear least squares in which the nonlinear parameters were fixed according to the values just estimated, and the linear parameters (amplitudes and phases) fitted. This way, starting values need to be provided *only* for the nonlinear parameters, because only they are determined iteratively. Both implemented VARPRO algorithms (Bolstad and MINPACK, see section 13.2) use this variable projection method. The new AMARES algorithm (NL2SOL, see section 13.2) minimizes the entire model function, but automatically obtains starting values for amplitudes and phases in a pre-fit linear least squares fit of the model function to the data, with the starting values you supplied for frequencies and damping factors filled in for the nonlinear parameters[28].

Hence, before you can perform VARPRO/AMARES parameter estimation on your data, you have to supply starting values for the frequencies and damping factors of the resonances you wish to include in the VARPRO/AMARES fit (see also section 3.3). This immediately gives you to the possibility to select the peaks that need to be fitted: peaks for which no starting values are given, will not be fitted (unless starting values from a small selected peak 'wander off' to a large neighboring, non-selected peak). The displayed FT spectrum is an ideal means to provide starting values for the frequencies and damping factors, as the tops of the peaks should be very near to the central resonance frequency, and the linewidth at half height is inversely proportional to the damping factor[3,4]. Hence, by pointing the mouse cursor at the top of a resonance you can supply a starting value for the frequency of that component, and by indicating the linewidth of that peak you enable the Peakpick function to calculate a starting value for the damping factor.

The practicalities of using the Peakpick function in MRUI are described using the same example data set as before, the file `ischr.dat/mat`. The Peakpick function uses only the basic spectral window, see for its general menus and functions sections 5.1. As mentioned before, the Peakpick function has extensions to both the File load menu (section 5.1.1.1) and the Miscellaneous batch functions menu (section 5.1.3.5), which will be discussed here. The function-specific menu is titled Peakpick, its menu items are described below. Initially, after selecting Peakpick from the quantitation popup menu of the MRUI base window (see section 3.1), the basic spectral window will be displayed empty, and after loading the file to be analyzed, its spectrum will be displayed in it, and the menu items activated (see section 5.1). You can then set about supplying the starting values.

11.1 The File load menu

11.1.1 file / batch

You load a single file into the Peakpick function when you wish to analyze a single data file, but when you wish to analyze a batch series of data files, you have the choice between loading a single file or a batch. Of course, when preparing to analyze a batch of files, you can load the batch into Peakpick and carry out the peakpicking for starting values on each file of the batch. However, that would annihilate most of the advantage of having an automatic batch procedure available in the software. Usually, all files in a batch show similar spectral resonances, and in most cases it is possible to prepare one set of starting values which will serve for the VARPRO/AMARES analysis of each file in the batch. You can then simply load any data file of the batch into

the Peakpick function, provide the starting values, and then run VARPRO/AMARES on the batch. This is because when you produce a set of starting values, they are stored in the general ASCII file `peakpick.par`, which remains in your current directory. A dedicated copy of it is made for the data file you loaded to perform the Peakpick process: `<name>.pkp` (see also section 11.8). When VARPRO/AMARES carries out the analysis on the batch of files, it will for each data file check whether a dedicated starting value file `<name>.pkp` exists. However, if none is found, it uses the general starting value file `peakpick.par`. This way, preparing a starting value set for an arbitrary file of your batch series allows it to be used for all other files of the batch automatically, as long as these files are still in the same directory as the peakpick.par file, and as long as care is taken when the `peakpick.par` file is overwritten after you have carried out another Peakpick process in the same directory (see also section 11.8).
Should you be worried that this approach is too static, e.g. when you have resonances that shift frequency during the time series in which your batch was measured, you may use the special batch function available for VARPRO/AMARES (see section 13.1), which allows the parameter estimation results of signal `i` of the batch to be used as starting values for the fit on signal `i+1` of the batch (under controlled conditions). This way, your starting values 'track' changes throughout your batch after having supplied an initial set of starting values on the first file of the series.
Finally, if you have even more challenging series of files, where extra resonances appear (e.g. a splitting of the inorganic phosphate peak in ^{31}P MRS of exercising muscle) or resonances disappear (such as drug uptake measurements in ^{19}F MRS), even this 'tracking' approach may cause severe problems when starting values are given to VARPRO/AMARES for a peak that no longer exists. In this case, you should take the following strategy. Consider the example of a ^{19}F MRS drug uptake experiment, yielding a series of 36 data files (base name `tum5fu`). Initially, directly after injecting the drug, only 1 resonance is present. Then, in the forth spectrum, a by-product resonance appears, and in the twelfth spectrum a second one. From spectrum 25, the drug resonance has completely disappeared, leaving only the two peaks of by-products in the spectrum.

- Display the batch of spectra using the Plot batch spectra function (see section 16.2), and run the highlighted spectrum through the batch, making a note of the batch index for each spectrum where the number of resonances changes. In this example, that is 1, 4, 12 and 25.
- Prepare sets of starting values for those data files containing a signal with a modified number of resonances. In this example, run Peakpick on these files as follows:
  - `tum5fu1.dat`, peak-picking the one drug resonance (1 peak)
  - `tum5fu4.dat`, peak-picking the drug resonance and its first by-product resonance (2 peaks)
  - `tum5fu12.dat`, peak-picking the drug resonance and both by-product resonances (3 peaks)
  - `tum5ful25.dat`, peak-picking only the by-product resonances (2 peaks)
- Run Inputvp on these files accordingly (see chapter 12).

Now when you run the VARPRO/AMARES batch, it will start at the first file, and find `tum5fu1.pkp` for starting values (which it will copy to `peakpick.par`). Next, for index 2 and 3, it will find no dedicated `<name>.pkp`, and it will use the `peakpick.par` provided by the first signal. Then, for index 4, it will find `tum5fu4.pkp`, which it will use and copy to `peakpick.par` (the previous one will be overwritten, but of course still exists as `tum5fu1.pkp`). Next, for index 5 to 11, it will again find no dedicated `<name>.pkp`, hence it will re-use the `peakpick.par` provided by the Peakpick on signal 4, which contains starting values for two peaks. From index 12 to 24 `tum5fu112.pkp`, containing starting values for three peaks, will be used, and from index 25 on, `tum5ful25.pkp`, containing starting values for the two by-product resonances. This way, the whole batch, containing signals with varying numbers of resonances, can be analyzed fully automatically, having provided starting values only for the small number of batch indices where the number of peaks changed. It should be noted however, that in such a situation, the produced batch results files (see section 13.7) can no longer be properly used in the Plot batch results function (see section 16.3), as the peak number columns are no longer properly aligned. However, you can load these files into your spreadsheet software, and restore the correct column alignment manually (because you know at which rows the numbers of peaks changed, and which peaks appeared and disappeared), after which you can plot the graphs in your spreadsheet software.

11.1.2 peakpick.par / <name>.pkp / <name>.scr / <name>.gol

Having selected Peakpick from the quantitation popup menu of the MRUI base window adds a number of specific submenus to the File load menu (section 5.1.1.1). Initially, these will be disabled (inactive), but after you have loaded a single data file or a batch of data files, these four submenus will be enabled (activated). You can then select one of them to pre-load a set of starting values, upon which you can make minor modifications and save the updated set of starting values. Hence, the first time you analyze data from a particular type of experiment, you will have to supply starting values from scratch. However, once you have pre-existing files with parameter values for the resonances in your signals, you can use these to load into the Peakpick function, so that a basis of starting values is already present. After having loaded such an initial set of starting values, the menu items remove peak (section 11.6) and add peaks until [done] (section 11.7) under the Peakpick menu are enabled (activated), so that you can reduce or modify an existing set and add to it. The start peakpick menu (see section 11.5) will be disabled (inactive), otherwise you would overwrite the starting values just loaded. Hence, after having pre-loaded an initial set of starting values, you can remove peaks from it, or add new ones until you press the done radiobutton (see section 11.7), i.e. you cannot preselect the number of peaks that are to be added. If you wish to discard the pre-loaded set again, you will have to reload the data file via the File load menu.

This option may be very useful if you carried out a VARPRO/AMARES analysis on a data set, but upon checking the fitting results, you came to the conclusion that for one or a few peaks you wish to have better starting values, or that you wish to do the same analysis but with an extra spectral resonance included. It also allows you to create a template starting value set for each specific type of application (i.e. for each different organ, pathology, etc.), save it via the Database menu (see section 7.3.1), and then reload it into your Peakpick function for a dedicated set of starting values on a new data file, requiring perhaps only minimal modifications.

If you simply wish to create a new Peakpick file that is identical to the previous one, but with one or more peaks removed, act as follows. Load the previous `<name>.pkp` or `peakpick.par` file, remove the peak(s) you wish to exclude from the new VARPRO/AMARES fit (see section 11.6 for the remove peak menu item), then use add peaks until [done] (see section 11.7), and click on the done button immediately.

You can choose to pre-load as starting values the values found for the frequencies and damping factors in the current `peakpick.par` in the current directory, or you can select a previous file-specific set of starting values `<name>.pkp` (section 11.8), an SVD_1D result file `<name>.scr` (section 10.11), or a VARPRO/AMARES result file `<name>.gol` (section 13.7). When selecting one of the latter three, you will be presented with the file manager window, which you can use to browse through your directory structure, and select the appropriate ASCII file containing the frequency and damping factor values you wish to use as an initial set of starting values before completing the Peakpick process. When you have loaded such an initial set of starting values, the peaks for which starting values were found in the ASCII file will be numbered in the displayed spectrum. Just to show this effect, the file `ischr.dat` was loaded, after which the SVD_1D fitting results, obtained in the previous chapter (see section 10.7) were pre-loaded by selecting `ischr.scr` in the current directory. See Figure 46.

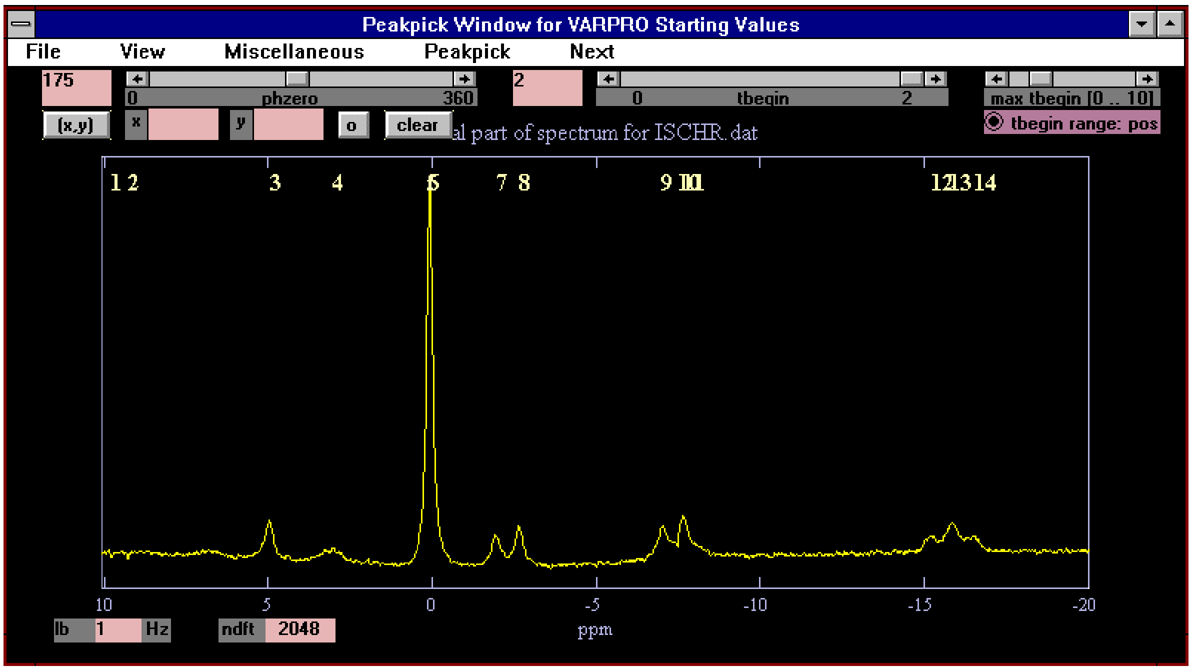


Figure 46. The Peakpick window after having loaded ischr.dat, upon which the SVD_1D fitting results (as stored in ischr.scr) were pre-loaded as an initial set of starting values. The frequencies found in the ASCII file are numbered in the displayed spectrum.

11.2 The Miscellaneous batch functions menu

When running a batch in Peakpick, you could previously only use the phasing sliders and edit boxes on the first spectrum, where you had to set everything correctly, and then when the batch started, each occurring spectrum only allowed direct clicking on peaks for starting values (except for the zoom feature added in MRUI_96.2, see section 11.5). It was not possible to rephase a spectrum e.g. halfway through a batch. This would be useful for instance when processing a (non-field-corrected) set of MRSI voxels in a batch series. You can now use the batch functions menu item from the Miscellaneous menu in Peakpick, *before* you start the batch Peakpick, where you can indicate which of the four phasing features you want to enable for each spectrum as it comes along in the batch. Be aware that the more features you enable, the slower the program will respond to your mouse clicks, as the possibility of doing other things during a batch involves pausing the program and checking those sliders and edit boxes continuously. Then when you start the batch Peakpick, you *must* rephase each spectrum, using the activated phasing features, first, and then you have to click on the done button for the real peak-picking to start on that spectrum. This process will be repeated for all spectra in the batch. This means that even if you do not wish to rephase a particular spectrum, you have to access the activated phasing features, if only to retype or re'slide' it to the same value it already had.

11.3 automatic by HLSVD

If the signal quality is high, you can have the starting values determined fully automatically via a Lanczos HLSVD (see section 10.1) fit on the signal. Selecting the automatic by HLSVD menu item will toggle its setting between on (checked) and off (unchecked). Then, when you select the number of peaks (see section 11.5), Peakpick will run Lanczos HLSVD with that number of peaks and the total number of data points. Should HLSVD crash, a message appears to that extent, and Peakpick resorts to awaiting your manual input. Should HLSVD crash for a certain file in a Peakpick batch, a message to that extent appears in the Matlab command window, and no starting values are produced for that file.
Note that HLSVD lists the frequencies from right to left (when displaying in reverse mode that is; HLSVD always lists frequencies from negative to positive values). This may differ from your personal approach in manual peak-picking, and hence calls for prudence when loading these values into the Inputvp window for prior knowledge.

Be aware that for low SNR data, the <n> peaks that HLSVD fits, are not necessarily the peaks for which you wish to have starting values. Also, the frequencies and damping factors estimated by HLSVD may suffer from the low SNR, and hence manual peak-picking might be much more reliable in these cases. Remember (section 10.7) that for an automatic black-box method such as HLSVD, there is no way to control which peaks are fitted or within what regions the fitted parameters may reside. Therefore we recommend the manual peakpick for *in vivo* data, and leave the automatic SVD determination as an interesting option for high SNR data sets.

11.4 linewidths from reference

Instead of manually providing a frequency and damping factor (via the resonance linewidth) for each peak, you can provide frequency starting values for each peak, with one overall damping factor (linewidth) as a starting value for all damping factors of all peaks. Especially for *in vivo* ^{1}H MRS, with the effective linewidth (damping factor according to T_2^*) largely dominated by inhomogeneities and magnetic susceptibility effects (the 'shim'), this can be a very valid approach if individual linewidths do not vary too much. Moreover, this is only a starting value and no constraint.

If, for the VARPRO/AMARES analysis, the starting values for the linewidths are not going to be very accurate, this is usually not a problem, but it is good to know that VARPRO performs better with starting linewidths that are just too narrow, rather than just too broad. AMARES is less sensitive than VARPRO to an over- or under-estimation of the actual linewidths.

11.4.1 peak in spectrum

In this case, the starting value for all damping factors will be taken from the linewidth of an arbitrary reference peak in the spectrum. This reference peak does not necessarily need to be included in your regular Peakpick process, and simply serves as a graphical way to provide an overall starting value for all linewidths. Select one for which it is easy to indicate the linewidth (i.e. preferably isolated), and which is narrow rather than broad compared to the other resonances of interest. After you have selected this submenu item (which then gets checked), you should start the normal Peakpick procedure (see section 11.5), but you will now only have to indicate the top/center of each peak you wish to include. When finished, Peakpick will prompt you to click on both the top/center and the half height (the position from which you decide it should read the linewidth) of the reference peak, which will now be indicated by red crosses (+).

11.4.2 unsuppressed water

The overall starting value for all linewidths can also be taken from an unsuppressed water signal if available. This effectively takes the shim value as a starting value for the linewidths. After you have selected this submenu item (which then gets checked), you should start the normal Peakpick procedure (see section 11.5), but you will now only have to indicate the top/center of each peak you wish to include. When finished, Peakpick will check if a `<name>.wat` file (see sections 10.8,10.11) already exists in the current directory, and if so, it reads its value for the unsuppressed water linewidth and uses it as a linewidth starting value for the current set. If no `<name>.wat` file is found, Peakpick searches for a corresponding data file containing unsuppressed water (such as `<name>w.dat`, see section 4.5) in the current directory, and if found, carries out a Lanczos HLSVD fit on it, as described in section 10.8, in order to produce the `<name>.wat` file and read the unsuppressed water linewidth. If no `<name>.wat` and no corresponding unsuppressed water file can be found, Peakpick will prompt you to type the starting value for the linewidths in an Edit Window.

11.4.3 type value

In this case, after you have carried out the Peakpick procedure (see section 11.5), indicating only the top/center of each peak you wish to include, Peakpick will prompt you to type the starting value for the linewidths in an Edit Window.

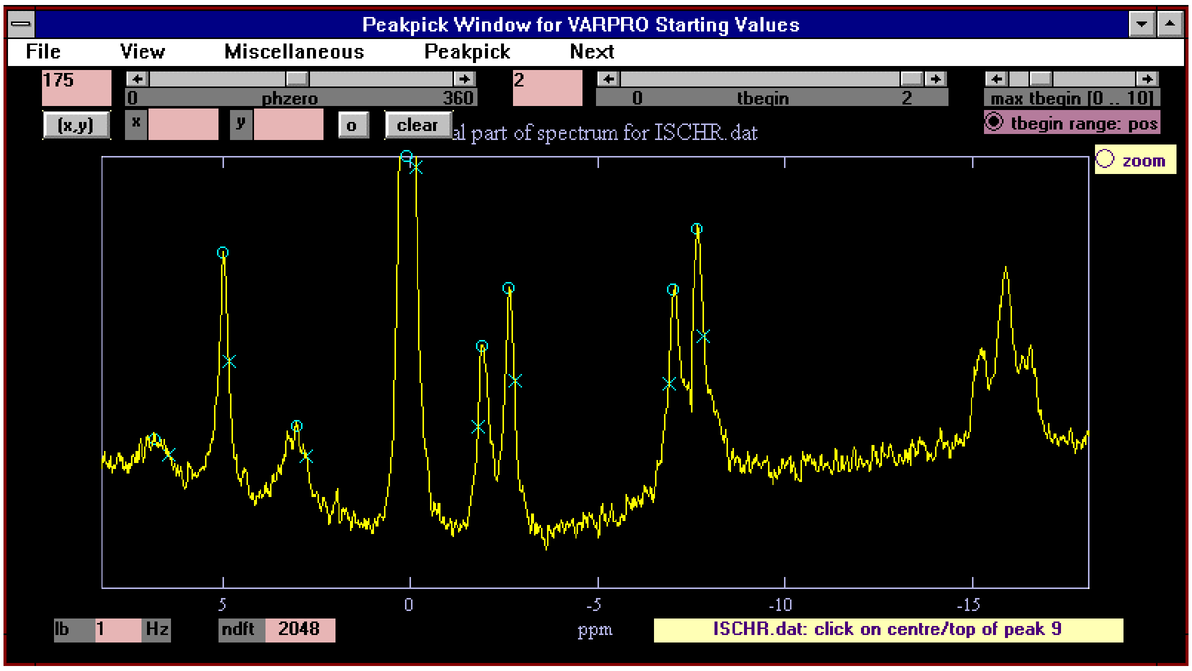
11.5 start peakpick

The actual Peakpick procedure, as explained above, consists of indicating, for each peak you wish to include in the VARPRO/AMARES fit, its top/center and position at which the linewidth should be measured (tentatively called the ‘half height’). From indicating the top of the peak, MRUI can immediately read the frequency and use it as a starting value to be stored in the peakpick.par file. From the ‘half height’, MRUI will determine the frequency difference between the pixel selected for the top/center and that selected for the linewidth position, and that frequency difference is a measure for the half-width. Because this difference is calculated by its absolute value, it does not matter whether you position your click for the ‘half height’ to the left or the right of the center of the peak (just use what is easiest, see also Figure 47). Multiplying it by 2 gives the full linewidth, and division by π and getting the units correct yields a starting value for the damping factor. Finally, the current settings of the phzero and tbegin sliders (see section 5.1.4) are also written to the `peakpick.par` file, where they will serve as starting values for the zero order phase and begin time, should you decide to have them estimated by VARPRO/AMARES (see section 12.4.6). In that case it is hence a good idea to at least phase the spectrum as correctly as possible. The start peakpick menu item allows you to select the number of peaks you wish to include in the VARPRO/AMARES fit from the appropriate submenu. The maximum number of peaks is determined in the VARPRO/AMARES source code (see Appendix E) and should be set accordingly in the customize fitting parameters menu (see section 3.2.1.4) in order to have these submenus display the only possible options. You can also start peak-picking without defining the number of peaks you wish to pick. Select the menu item peakpick until [done]. Next, Peakpick will immediately prompt you to start your manual peak-picking for peak 1. A message text bar will appear in the lower right corner of the basic spectral window, telling you e.g. “ischr.dat: click on center/top of peak 1”. You can use the customize preferences menu to disable the message text in order to work faster (see section 3.2.1.3). After you have clicked on the center/top of that peak, a cyan circle (o) will be drawn there, see Figure 47. The next action depends on whether you have used the linewidths from reference menu as described in the previous section. If you did, you are prompted to click on the center/top of the second peak, etc., until you have reached the pre-selected number of peaks. If you did not prefer to use an overall linewidth value as starting value for all peaks, you will now be prompted to “ischr.dat: click on half height of peak 1”. The message text bars are color-coded for easier recognition: yellow background with purple text to click on the top, and purple background with yellow text to click on the half height. This should not necessarily be the exact half height between the top of the peak and the x-axis, but the position at which you decide MRUI should measure the linewidth for that peak. At the position you indicated, a cyan cross (x) will be drawn, see Figure 47. You will be asked to click on center/top and half height of peaks in turn, until you have reached the pre-selected number of peaks. If you selected to peakpick until [done], a radiobutton done will appear on the far right of the window. Click on it when you have finished your peak-picking. If it turns out that you have clicked on done after an odd number of clicks, Peakpick will discard the last click. (not if you set the option linewidths from reference, in which case you only need to click on the tops of the peaks you select, and you can have an odd number of peaks of course).

If the spectra are not terribly complex, it is usually not necessary to immaculately click on the exact centers and true linewidths of all peaks. Peakpick after all deals with starting values, the mere initial conditions from which to start the iterative nonlinear least squares search in VARPRO/AMARES. Of course you would do best to get them as accurate as possible, but you should not waste lots of time trying to get the mouse cursor dead on the top of the peak. There is another danger in that, because for overlapping peaks, the top of a resonance in the FT spectrum might in fact be positioned slightly off the central frequency for that resonance, and noise features can make it look as if the top of the peak is a few Hz away from where the actual top would have been in a noiseless spectrum. However, as an extreme you should indicate the frequency starting value at least within the linewidth of that peak, and you should take care that the position of the linewidth does not come too close to the position you clicked for the frequency, as that difference is important in order to calculate the decay constant, which is the reciprocal of the linewidth. If the frequency difference between your clicks for frequency and linewidth is too small or even zero, the decay constant will tend to infinity, which will cause problems in the subsequent VARPRO/AMARES analysis.

Note that, because a starting value needs to be given for the *central frequency*, it is not necessary to actually click at the top of a peak. If you have a symmetric peak, you might just as well click at half height, just in the middle. This can be very useful if you have many small peaks and one or a few large ones, just as in the current example. You can scale up the entire spectrum to the level of the smaller peaks, clipping the large peak of PCr, as long as you click sensibly for the clipped peak, see also Figure 47. This greatly facilitates accurate clicking for the smaller peaks.

For the current example, 11 peaks are to be included for the VARPRO/AMARES fit of ischr.dat (PME, P_i, PDE, PCr, γATP_1, γATP_2, αATP_1, αATP_2, βATP_1, βATP_2, βATP_3; a possible contribution of NAD is discussed in section 13.4), and Peakpick was carried out completely, i.e. both for frequencies and for all linewidths. The situation just after having indicated the αATP doublet is displayed in Figure 47.



ISCHR.dat: click on centre/top of peak 9

Figure 47. The Peakpick spectral window, scaled up to the smaller resonances of interest, with circles (o) on the positions of the mouse clicks for the center/tops of the peaks, and crosses (x) at the positions from which the linewidths are to be calculated.

You can re-zoom to different parts of the spectrum at any time while peak-picking. Even though the message bar says that it expects a center/top or half height click for peak number <n>, click on the zoom radiobutton on the far right. The spectrum will be automatically zoomed out to its full spectral width, and you can zoom in to a new spectral region as you like (see section 5.1.2.1). Then Peakpick will ask you again for the expected click, and now the zoom radiobutton is disabled (inactive) until you have actually entered a new peak-pick value (i.e. you cannot zoom again immediately). This is also possible when peak-picking a linewidth reference peak (see section 11.4). At the end Peakpick will automatically zoom the spectrum to its initial view width.

11.6 remove peak

With this function you can remove any of the peaks from the pre-loaded set of starting values (section 11.1.2), otherwise this menu item will disabled (inactive). If you select remove peak, you will be prompted by an Edit Window in which you can type the number of the peak that should disappear. The displayed spectrum will be updated, and the numbers of the remaining peaks will be updated accordingly.

11.7 add peaks until [done]

This menu item is only enabled (active) if you pre-loaded a set of starting values as described in section 11.1.2. It completely replaces the action of the peakpick until [done] submenu item under start peakpick (see section 11.5), but retains the pre-loaded set (of which peaks might have been removed, see previous section) and adds the newly selected peaks to it. This can also be done using an overall damping factor as a starting value for all peaks (see section 11.4) and also allows intermediate re-zooming (see section 11.5), but does not permit automatic determination of starting values via HLSVD (section 11.3).

11.8 Files produced

Once you are finished using the Peakpick function, the following files will have been produced or updated:

```
peakpick.par
<name>.pkp
```

If the starting values for all damping factors were taken from the linewidth of the unsuppressed water signal (see section 11.4.2), and no `<name>.wat` was found, but a corresponding unsuppressed water file present, a new `<name>.wat` will have been produced.

12. Prior knowledge for VARPRO: Inputvp

As was stressed in sections 3.3,10.7 and the beginning of chapter 11, the most important reason for running VARPRO on *in vivo* MRS data sets, is the possibility to impose prior knowledge on the fitting, hence controlling the physical relationships between the various parameters. The importance of imposing prior knowledge has been discussed in the literature[3,4,18,19,20,26,29]. For this manual chapter we start off with explaining the available prior knowledge that can be imposed as a set of linear parameter relations on the VARPRO fit for the current example file `ischr.dat/mat`, containing an *in vivo* ^{31}P MRS signal from human muscle at 1.5 T.

The practicalities of using the Inputvp function in MRUI are described using the same example data set as before, the file `ischr.dat/mat`. The Inputvp function uses the basic spectral window, see for its general menus and functions sections 5.1, and a special prior knowledge window. The function-specific menus in the basic spectral window are titled Inputvp and Group Peaks, the menu items are described below. Initially, after selecting Inputvp VARPRO from the quantitation popup menu of the MRUI base window (see section 3.1), the basic spectral window will be displayed empty, and after loading the file to be analyzed, its spectrum will be displayed in it, and the menu items activated (see section 5.1). You can then set about grouping peaks together and imposing the prior knowledge.

12.1 Prior knowledge and parameter relations

For the adenosine triphosphate multiplets (see e.g. Figure 47), the following biochemical prior knowledge is available. The linear relations (constraints) between the various parameters, according to which the available prior knowledge is implemented in the VARPRO algorithm is also displayed in the table. The selected peaks are numbered from left to right, see also Figure 48.

parameter	prior knowledge	linear relations / constraints
amplitude/ peak area	At high fields, the ATP amplitudes (peak areas) relate as 1:1 for γATP, 1:1 for αATP and 1:2:1 for βATP. The total integrated intensity for the different ATP multiplets should also be identical, as there is just one atom of each per molecule: γATP:αATP:βATP = 1:1:1. Should you wish to implement this 'overall' ATP amplitude constraint, you must take care to supply the βATP ratio as 0.5:1:0.5 in Inputvp.	$a_5=a_6=a_7=a_8=2a_9=a_{10}=2a_{11}=a_{ATP}$
	At the current field strength of 1.5T, similar ratios have been deduced[3]: 1:1.09 for γATP, 1:1.18 for αATP and 1.29:2.25:1 for βATP. The overall amplitudes will not be linked in this example.	$a_6=1.09a_5$ $a_8=1.18a_7$ $a_9=1.29a_{11}$ $a_{10}=2.25a_{11}$
linewidth/ damping factor	Because the relaxation rate T_2 is more or less identical for the whole ATP molecule, all separate resonances should have the same damping factors (decay constants) and line widths.	$\alpha_5=\alpha_6=\alpha_7=\alpha_8=\alpha_9=\alpha_{10}=\alpha_{11}=\alpha_{ATP}$
frequency	Due to the spin-spin interaction, the frequency splittings between the individual resonances within the multiplets will be Δν=J Hz, with J the spin-spin coupling constant for ATP.	$\nu_6-\nu_5=\nu_8-\nu_7=\nu_{10}-\nu_9=\nu_{11}-\nu_{10}=J$
	At a field strength of 1.5T, and for in vivo ^{31}P MR of human muscle, J=16 Hz may be used[3] for all ATP multiplets in the current example.	$\nu_6-\nu_5=-0.016$ kHz $\nu_8-\nu_7=-0.016$ kHz $\nu_{10}-\nu_9=\nu_{11}-\nu_{10}=-0.016$ kHz
phase	The true individual phases of all peaks in the current spectrum should be identical (and equal to the overall zero order phase of the spectrum), as there are no strongly coupled multiplets.	$\phi_5=\phi_6=\phi_7=\phi_8=\phi_9=\phi_{10}=\phi_{11}=\phi_0$

phzero	The overall zero order phase ϕ_0 can be estimated by the VARPRO algorithm, or it can be fixed to a value, e.g. from manually phasing the spectrum (175^o in this example, see e.g. Figure 47). For the VARPRO analysis in the current example, the zero order phase was estimated, in order to reduce the operator dependence of the results.	ϕ_0 is a free parameter
tbegin	The begin time t_0 can be estimated by the VARPRO algorithm, or it can be fixed to a value, e.g. known from experimental parameters (2 ms in this example, see e.g. section 4.6 and Figure 47). For the VARPRO analysis in the current example, the begin time was fixed to the value read from the original spectrometer file.	$t_0=2$ ms

For the remaining peaks (PME, Pi, PDE and PCr), no specific biomolecular prior knowledge is available, and only the phase prior knowledge is relevant: $\phi_1=\phi_2=\phi_3=\phi_4=\phi_0$.

It can be seen that imposing the biomolecular and experimental prior knowledge as a set of constraints significantly reduces the number of free parameters in the model function to be fitted. This reduction leads to shorter calculation times, better convergence behavior and improved parameter estimation results which are more reliable and more reproducible. The implementation of prior knowledge also enables consistent fitting of small metabolite resonances in the presence of large broad background signals[3,4,19] or in low SNR data[18,19,26].

12.2 The File load menu

The load menu item under the File menu only has one submenu: file. You cannot run Inputvp on batches, but only on single files. Read section 11.1.1 on how to prepare sets of starting values for subsequent batch VARPRO analysis. Just as Peakpick produces a general file `peakpick.par` that can be used if no file-specific set of starting values is found, a general prior knowledge scratch file `inputvp.ans` and `link.ans` is produced by Inputvp, in addition to the file-specific versions `<name>.inp` and `<name>.lnk`, respectively (see also section 12.6). This means that when running VARPRO on any particular data file, whether in single file mode or in batch, MRUI checks whether a file-specific set of starting values and prior knowledge exists, and if either or both do(es) not exist, it will use the general `peakpick.par` and/or `inputvp.ans` and `link.ans` instead. This means that if you have many similar data files in the same directory, even if they do not belong to a batch, you can keep running VARPRO without having to re-run Inputvp as long as the peak numbering matches between the starting value set and the prior knowledge set. Whenever VARPRO needs to rely on a general `inputvp.ans` and `link.ans`, it first checks whether at least the number of peaks in the prior knowledge set is identical to the number of peaks in the starting value set, otherwise the VARPRO algorithm would crash. If the numbers differ, a message is displayed, and VARPRO aborts.

When loading a file into Inputvp, MRUI searches for a previous `<name>.inp/lnk` file with this data file. If one is found (i.e. if you did Inputvp on this file before), all previous settings are loaded back into the Inputvp interface, i.e. all groups are selected as they were before, parameter relations are filled in like before etc., *if* the number of peaks in the old `inputvp.ans` file corresponds to the number of peaks in the current `peakpick.par` file. This allows you to make only minor changes and exit.

12.3 The Group Peaks menu

When you load the signal to be analyzed in the Inputvp function, its FT spectrum will be displayed as usual, but the resonances will be numbered in the spectrum, according to the order in which you selected them in the Peakpick function (see chapter 11), see also Figure 48. A second row of numbers indicates in which group the resonance with that peak number is grouped together with other peaks. Initially, all peaks are grouped together into group number 1; there is no default prior knowledge assumed. Once you have loaded the file, the Group Peaks menu will get its menu items, as many as their are peak starting value sets read in from the

`<name>.pkp` file (see section 11.8), say `npeak` resonances were selected in the Peakpick function. Each menu item lists the peak number, and has itself again `npeak` submenu items, from which you can select the group this peak should belong to. There can be as many groups as there are peaks, i.e. each peak constitutes its own group, but normally you will group peaks together in order to impose the parameter constraints as described in the table in the previous section. You can also group together the peaks for which you have no biomolecular prior knowledge, if only to reduce the amount of user interaction, using a 'no-constraint' group or a group with merely a phase constraint (e.g. all peaks in it have a phase equal to the zero order phase). In the current example it is evident that the ATP multiplets are going to be grouped together, and the remaining peaks are left in their default group 1, in order to give that group a general phase constraint only. The grouping of peaks for the current example can be summarized as follows:

peak number	metabolite resonance	group number
1	PME	1
2	P_i	1
3	PDE	1
4	PCr	1
5	γATP_1	2
6	γATP_2	2
7	αATP_1	3
8	αATP_2	3
9	βATP_1	4
10	βATP_2	4
11	βATP_3	4

You can group for example the second αATP peak into group 3 by selecting under the Group Peaks menu the menu item peak 7, and then selecting its submenu item group 3. The group numbers over the peaks in the displayed spectrum will be immediately updated. See Figure 48 for the spectral display after grouping all peaks according to the previous table.

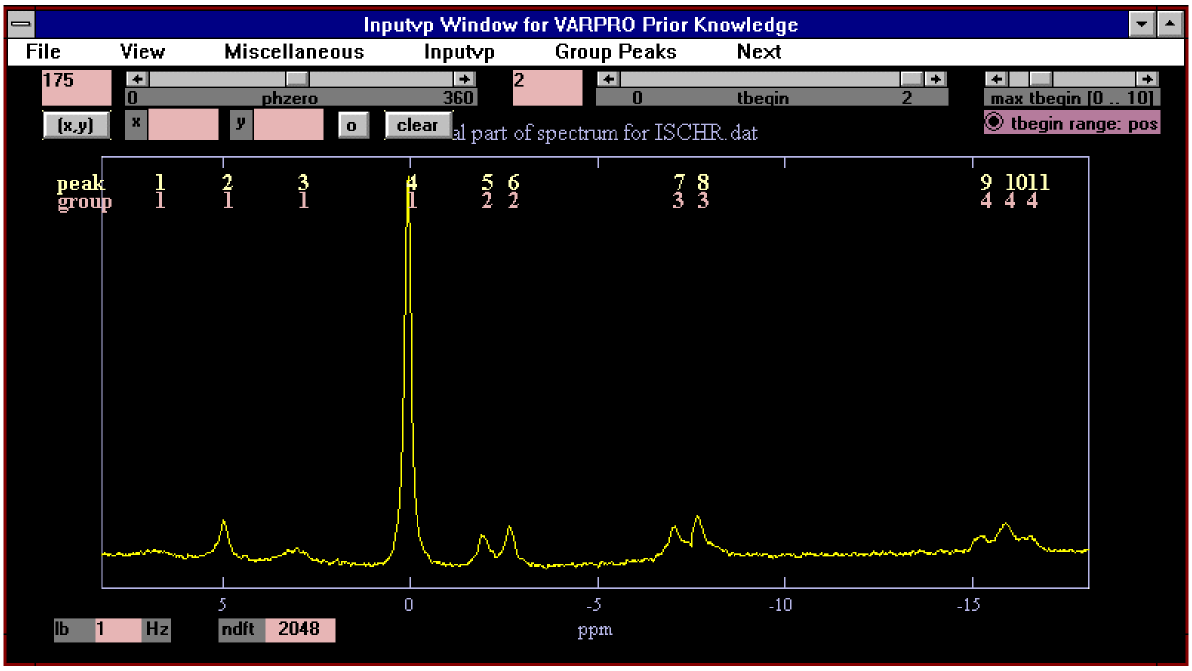


Figure 48. Inputvp displaying the FT spectrum, with peaks numbered according to their Peakpick selection (top row) and according to which prior knowledge group they have they have been placed in.

Once you have grouped the peaks in preparation of the parameter relations, you can start supplying the parameter constraints in the prior knowledge window, activated via the prior knowledge menu item under the Inputvp menu. However, there are a number of other options to be set for the VARPRO algorithm, also under the Inputvp menu. As long as you have not selected the prior knowledge menu yet, you can change all your settings under the Inputvp menu and even the grouping of the peaks. We recommend therefore to proceed in the following order: Group Peaks menu, Inputvp menus, prior knowledge. The phasing of the spectrum will still update the value of the zero order phase and/or begin time in the prior knowledge (if their values are fixed, see section 12.4.6) even after you have activated the prior knowledge menu.

12.4 The Inputvp menus

12.4.1 cov / corr table

With this menu item you can indicate whether VARPRO should, at the end of its run, display the covariance / correlation matrix in the Matlab command window. The covariance / correlation table is a matrix that displays all covariances and correlations between the estimated parameters. This can be of statistical value in cases where resonances severely overlap, for example. However, it is an option that has been put into the software by the developers for experimental reasons, in order to investigate the full potential of the statistical parameter estimation algorithm, and to investigate properties of different model function lineshapes. For example, if two parameters are highly correlated, this means that there are many valid solutions to their joint fit in the model function, which might prompt you to supply slightly different prior knowledge, new starting values, or perhaps exclude a peak or introduce another. Selecting the cov / corr table menu item toggles its setting between ‘no’ and ‘yes’, which is indicated by an ‘n’ or a ‘y’ in the menu item. You can change its default setting using the Inputvp menu in the customize fitting parameters menu (see also section 3.2.1.4). The package default is ‘no’. In the current example of `ischr.dat`, no covariance / correlation table was desired.

12.4.2 truncate initial points

The reasons behind removing data points at the end or beginning of the FID have been explained in section 8.5. Of course, physically removing data points at the end of an FID is not really necessary, as both in the SVD_1D methods and in VARPRO, the number of data points included in the model function fit, `ndp`, can be set such that an arbitrary number of points at the end of the FID is excluded from the fit See also sections 10.2 and 13.3. However, as for excluding initial data points from the fit, the SVD_1D methods do not have an internal feature that can take care of this, so initial data points to be excluded have to be physically removed from the signal first, using the truncate function in the FID Maths library, see section 8.5. VARPRO on the other hand, has an internal weighting vector, that is multiplied with the data vector *and* the model function vector before the non-linear least squares fit takes place, see also section 12.4.3. If the entire FID is to be fitted, and the frequency selective feature is not used, the entire weighting vector simply consists of ones on all positions, so that both the data vector and model function vector are unaltered. However, `begoff` initial data points can be easily excluded from the VARPRO fit, simply by setting the first `begoff` elements in the weighting vector to zero. It is this feature which is controlled by the truncate initial points menu item. If you select it, you can type the number of initial data points to be excluded from the fit in an Edit Window, after which the text in the menu item will be updated accordingly. You can change the default setting using the Inputvp menu in the customize fitting parameters menu (see also section 3.2.1.4). The package default is 0, i.e. no truncation. In the current example of `ischr.dat`, we truncate the first data point only, just in case it might contain receiver distortions.

There is no general suggestion for the number of data points you would have to truncate in order to minimize the effects of broad underlying humps. You would perhaps need to find out by trial and error on a typical signal of your data set, but numbers between 1 and 15 are most often used. Check the beginning of your FID in the plot FID function (see section 16.1) for suspicious first data points, these might contain receiver distortions or pulse breakthrough.

As mentioned in section 8.5, large and less mobile molecules give rise to broad resonances throughout the spectrum, thereby hindering the accurate quantitation of the resonances of interest. According to the reciprocal relation between the relaxation time and linewidth of a resonance, these broad resonances arise from sinusoids with short relaxation times and very large damping factors: their time domain signal will decay rapidly to zero, within the first few data points. See Figure *49*. The solid lines are the exponentially damped sinusoid and its corresponding Lorentzian in the FT spectrum, but they both suffer from the resonance of a less mobile compound (dotted line). In the FT spectrum, this causes a broad underlying baseline hump, which in practice is very hard to approximate or separate from metabolite resonances of interest, see the total resonance in *Figure 49* (dashed line). With time domain analysis methods, it is possible to process only that part of the FID where the quickly decaying baseline resonances have already died out and no longer contribute.

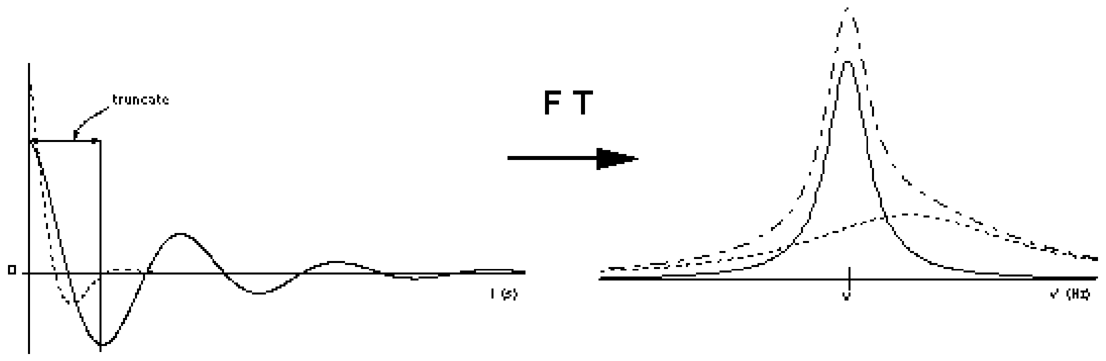


Figure 49. The effect of a rapidly decaying signal (dotted line) on a sinusoid / Lorentzian of interest (solid line). The total resonance in the spectrum (sum of the two) is indicated with a dashed line. Time domain analysis allows the elimination of these effects on the model function fit by excluding the initial data points from the FID and model function accordingly. In the frequency domain these effects can only be approximately corrected, with negative consequences for the accurate determination of the true peak area of the Lorentzian of interest.

As explained in section 8.5, the effective begin time will change when initial data points are excluded, but this is corrected for in the estimation of the amplitudes at time zero and the phases.

If you use the frequency selective feature of section 12.4.3, you must realize that the first 20 data points are multiplied by the first quarter of a squared sine wave. This means that the first data point is multiplied by zero, and hence effectively truncated, and the following few points are also multiplied by very small numbers. This may already account for all the truncation you had in mind! If you use both truncation *and* the frequency selective feature, the sine square weighting starts from the first point included in the fit (i.e. *after* the `begoff` excluded initial data points), so the two effective truncations 'add up'.

12.4.3 frequency selective

The weighting of the data points for frequency selective fitting is explained in reference [30]. In short, performing frequency selective quantification in the time domain means fitting only a limited number of components (i.e. by selecting only a subset of all peaks when providing starting values in Peakpick, see section 11.5), and ignoring other, possibly large, contributions to the FID. Inherently, this will cause systematic errors in the parameters of the fitted components. However, Arno Knijn and coworkers have found that they can drastically reduce these systematic errors, to below the level of the statistical errors, by weighting the first 20 data points of the FID by the first quarter of a squared sine function, i.e. a smoothly rising function from 0 to 1. The weighting vector was already explained in the previous section. Frequency selective VARPRO fitting may even be a good idea if you intend to fit all prominent resonances in the signal, because it might significantly reduce the effects of broad baseline components, see also section 12.4.2. Since both the signal vector and the model function are weighted identically, there is no negative effect on the parameter estimation, except of course that you exclude part of the beginning of the FID in the fit, which effectively has the highest signal-to-noise ratio. Should you have data sets with hardly or no baseline humps and you include all peaks in the VARPRO fit, or if you are analyzing all peaks of a simulated signal, then you should switch frequency selective off, in order to analyze the raw unprocessed FID. This is because the weighting of data points introduces a slight 'coloring of the noise', which affects the exact Maximum Likelihood properties of the algorithm[31].
Selecting the frequency selective menu item toggles its setting between 'no' and 'yes', which is indicated by an 'n' or a 'y' in the menu item. You can change its default setting using the Inputvp menu in the customize fitting parameters menu (see also section 3.2.1.4). The package default is 'yes'. In the current example of `ischr.dat`, frequency selective was applied.

12.4.4 lineshape

With this menu item you can select whether the lineshape of all individual components to be fitted should be Lorentzian or Gaussian. The result table (see Figure 52, section 13.4) and result files (see section 13.7) will have an entry for `gaussian fraction in lineshape`, but this fraction can be only 0 or 1, i.e. you can fit only with pure Lorentzians or pure Gaussians. You can change the default setting using the Inputvp menu in the customize fitting parameters menu (see also section 3.2.1.4). The package default is Lorentzian. In the current example of `ischr.`dat, the Lorentzian model was applied.

12.4.5 noise level

The VARPRO algorithm needs an estimate of the noise standard deviation, in order to compare against the root mean square of the fit in the result file, and in order to calculate the Cramér Rao lower bounds as estimates for the standard deviations of the estimated parameters[31], which are also written to the result files (see e.g. section 13.7). You can select from three different ways of determining the standard deviation of the noise. You can change the default setting using the Inputvp menu in the customize fitting parameters menu (see also section 3.2.1.4). The package default is to calculate the noise from the last 100 data points of the FID. This default option was used for the current example of `ischr.dat`.

12.4.5.1 from last data points

With this (recommended) default setting, the method determines the standard deviation of the noise from the last data points of the FID (how many final data points are used in this calculation can be set by selecting the from last data points submenu item or in the customize fitting parameters menu). These really are the last data points of the full FID stored in the data file, not the last points from the length of ndp points you selected to include in the VARPRO fit. Of course, this assumes that these final data points no longer contain any signal contributions, i.e. the FID is fully decayed.

12.4.5.2 from residual after fit

Should the end of your FID still contain signal contributions (FID not fully decayed), you can have the noise standard deviation estimated from the residual after the VARPRO fit, on the assumption that the residual contains noise only. Hence, this is only possible if your data do not suffer from baseline humps, and if you fitted all signal components present. In this case, the values for the noise standard deviation and the root mean square of the VARPRO fit in the result files (see section 13.7) will be identical.

12.4.5.3 fixed value

If neither of the two above options seem applicable, you can always enter a fixed value of the noise standard deviation, should you happen to know it. Selecting this submenu item prompts you with an Edit Window, in which you can type the value of the noise standard deviation, which then immediately updates the text in the fixed value submenu item. Remember that the signal which VARPRO is analyzing is the *scaled* signal (see section 4.2), so the noise standard deviation you type here must be according to the scaled signal. If you measured the noise standard deviation from your original signal (before MRUI conversion took place), you must take this into account, and use the `scale` and `norm` factors to calculate the scaled value of the noise standard deviation you must supply here.
One situation where you would know the standard deviation of the noise exactly, is when you run VARPRO on a (set of) simulation signal(s), see also chapter 6. In this case it is even preferable to use the true value of the noise standard deviation, in order to calculate the correct values for the Cramér Rao lower bounds on the standard deviations of the estimated parameters, enabling their comparison to the population standard deviations in a Monte Carlo study, for instance (see section 6.2.9).
If you have FIDs which are not fully relaxed at the end, *and* your spectra contain many peaks you do not wish to fit (which end up in the residual so you cannot use the option from residual after fit), and you have no way of knowing the exact value of the noise standard deviation, you might use the following way out:

Acquire a short length of signal from your spectrometer with the same experimental settings, but without the sample in the coil. This will generate a signal which only contains the noise, which you could use for the calculation of the noise standard deviation, in one of two ways:

1. Convert the spectrometer file containing the complex noise signal, so that you get a Matlab file, e.g. `noise.mat`. You can load this signal into Matlab, calculate the standard deviation for the real and imaginary parts of the noise separately, using Matlab's `std` function (type **`help std`** in the Matlab command window for more information), and take the average of those as the fixed value for the noise level you enter here in Inputvp:

```
>> load noise
>> resd  = std(real(signal));
>> imsd  = std(imag(signal));
>> noisd = (resd + imsd) / 2
```

Be careful that in this case you must first use the `scale` and `norm` factors from the file `noise.scl` to calculate the original standard deviation of the noise (because upon MRUI conversion the noise signal was normalized to 1000, see section 4.2), and then the `scale` and `norm` factors from the file `<name>.scl` of your signal of interest, in order to calculate the noise standard deviation according to the scaled signal in `<name>.dat/mat`.

2. Write a script (Matlab, Fortran, C, whatever) that appends your noise signal at the end of your signal of interest in the data file. Make sure you update the number of data points in the first element of the file heading buffer (see section 4.1.1). In this case you must also be extremely cautious with scaling factors.
In Inputvp, you can now select the option to calculate the noise level from the last 100 data points or so, depending on how long your 'noise appendum' is.
In VARPRO, be sure to set the number of data points for your signal analysis, `ndp` (see section 13.3), such that VARPRO is not fitting your 'noise appendum'.

12.4.6 zero order phase & begin time

Both the overall zero order phase of the spectrum, and the begin time (which represents itself as a frequency-dependent first order phase throughout the spectrum with 'pivot' at 0 Hz), can be fixed as numerical values in the model function, or they can be treated as free model function parameters, in which case they can be estimated by VARPRO.
You can fix one or both of these parameters by selecting the constrained submenu item. In this case, the current value of the zero order phase and/or begin time, taken from the graphical display via the phzero and tbegin sliders, respectively, are used as the values to be filled in in the model function, and the text of the constrained submenu item is immediately updated accordingly. In the current example of `ischr.dat`, it is valid to fix the value of the begin time, since it was read from the SMIS data file as an experimental parameter (see section 4.6), and the spectrum appears to be phased correctly by it. On the other hand, the zero order phase of the spectrum also seems to be quite appropriate, but because it was obtained by manually phasing the spectrum in the basic spectral window, we do not wish to fix the zero order phase in the model function to this value, in order to avoid introduction of operator bias. However, if you are very confident in your experience as a spectroscopist, and you dare say that the zero order phase and begin time you have obtained from manual phasing of the spectrum (or perhaps you have obtained both parameters from the machine setup) are correct, then you can fix either or both parameters in the model function to the values found by your manual phasing. This means that you have decreased the number of degrees of freedom in the parameter estimation process by applying prior knowledge, and this will improve the estimation of the peak parameters (given, of course, that your zero order phase and begin time really are correct). In the current example we will select for the zero order phase the submenu item estimated by VARPRO. If the zero order phase and/or begin time are to be estimated by VARPRO, their values as present in the `<name>.pkp` file (at that time also read from the phzero and tbegin sliders) will be used as starting values. Because the starting values are in the `<name>.pkp` file, they can no longer be changed here in Inputvp. However, if the zero order phase and/or begin time are constrained, you can still use the phzero and tbegin sliders and edit boxes with the spectrum, to fine tune their values. These values are automatically updated in the prior knowledge table and are indicated in their respective submenu items.
You can change the default settings for the determination of the zero order phase and begin time parameters using the Inputvp menu in the customize fitting parameters menu (see also section 3.2.1.4). The package default is to have both estimated by VARPRO.

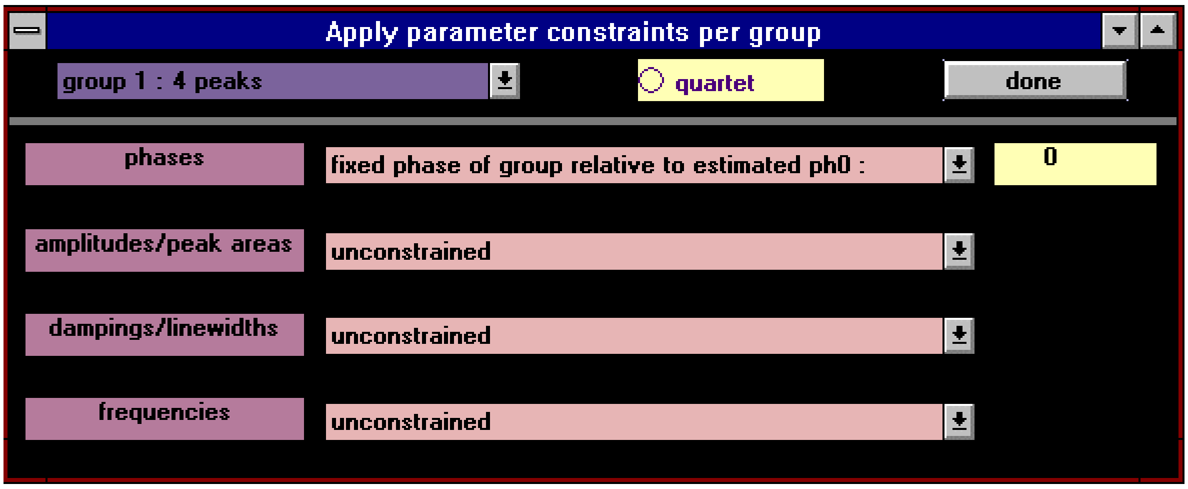
Once you have decided you wish to leave the zero order phase and/or begin time free, to be estimated by VARPRO, you *must* constrain the phases of each group of peaks in the prior knowledge window, relative to the zero order phase that will be estimated. That is, you can not have both a free zero order phase and free individual phases (theoretically impossible, this would yield an undetermined problem). See also section 12.5.1.2.

It is theoretically impossible to estimate the zero order phase and/or begin time when fitting only one peak. Inputvp has been protected against this, fixing the zero order phase and begin time constraints automatically if you have starting values for one peak only, and it will display an info message when you try to change these constraints (which you cannot).

12.5 The prior knowledge menus

12.5.1 prior knowledge

To apply the parameter constraints in linear relations for the grouped peaks, as explained in section 12.1, you select the prior knowledge menu item after which the prior knowledge will be displayed, starting with group 1, see Figure 50.


Apply parameter constraints per group

group 1 : 4 peaks | quartet | done

phases	fixed phase of group relative to estimated ph0 :	0
amplitudes/peak areas	unconstrained	
dampings/linewidths	unconstrained	
frequencies	unconstrained	

Figure 50. The prior knowledge window, allowing you to impose parameter constraints for the groups of peaks you have selected.

This window contains the following features which you can use to apply the linear relations for all parameters for all groups.

12.5.1.1 group popup menu

The group popup menu lists the groups as determined by your use of the Group Peaks menu (see section 12.3), both with the group number, and the number of peaks contained in each group. With this popup menu you can select for which group you wish to supply the parameter constraints. This does not necessarily have to be in numerical order, you can go forth and back between groups, and change your settings again, as long as you have not pressed the done pushbutton (see section 12.5.1.7). Whenever you select a new group, the parameter constraints as valid for that group at that moment are filled in in the window.

12.5.1.2 constraints for phases

For the phase constraint of a group, you can normally choose between two options from the popup menu:

unconstrained
fixed phase of group relative to estimated/fixed ph0

The text on the latter option of course depends on whether the zero order phase was fixed or left free to be estimated by VARPRO (section 12.4.6), but its function is identical: it expects a numerical value as a constraint in the edit box on the right, which it will insert in the model function for the individual phases of all peaks of the current group, regardless of whether the zero order phase is fixed to a numerical value, or left as a free parameter. If you set the phase of each group to 0 relative to the estimated zero order phase, this means that all peaks will get the phase that VARPRO estimates for the zero order phase. If you fixed the zero order phase, you can choose whether to leave all individual phases within a group of peaks free, or whether they are linked by a relative phase difference to the fixed zero order phase. If you set this constraint (relative to the fixed zero order phase) to 0, this means that all peaks in that group will have a phase of the value that you set for the zero order phase, either in Peakpick or in Inputvp. If you have peaks with different phases, e.g. the lactate doublet in ^{1}H MRS at an echo time of 136 ms (lactate doublet inverted), you should put these peaks in a separate group, and then give a phase constraint for that group of 180 relative to the zero order phase, whether the latter be estimated or fixed.
The unconstrained option indicates that the phases of all peaks in the current group are free parameters, and can get any value, depending on the fit procedure (but if the amplitudes are constrained according to some ratio, the phases of the peaks will turn out to be equal). As explained in section 12.4.6, this means that the option to leave all individual phases unconstrained, can not be used when the zero order phase and/or begin time are left free to be estimated. For this reason, the unconstrained option will only be found in this popup menu if the zero order phase was fixed to a value. If the unconstrained option is selected, there is no edit box on the right. Once you select to constrain the phases of the group, MRUI will apply a relative value of 0 by default. If the zero order phase and/or the begin time are to be estimated, the phases of the group have to be constrained, and then also a relative value of 0 will be filled in by default, see e.g. Figure 50. You can change the relative value for the phase constraint simply by placing the cursor in the edit box on the right and either selecting or backspacing over the current value, and typing your new desired value.

12.5.1.3 *constraints for amplitudes / peak areas*

For the amplitude constraint of a group, you can choose between two options from the popup menu:

unconstrained
constrain, enter ratios:

If you select the unconstrained option, the amplitudes of all peaks in the current group are free parameters, and can get any (positive) value, depending on the fit procedure. In this case, there is no text box on the right. If you have linear relations between the amplitudes of a group, such as for the ATP multiplets in the current example, you can select the option constrain, enter ratios, after you will be prompted by an Edit Window as many times as there are peaks in the current group. This way you can enter your peak ratios, e.g. for βATP we entered 1.29 in the first Edit Window, 2.25 in the second, and 1 in the third. MRUI will display the ratio in the text box on the right, see e.g. Figure 51.

Once you have constrained the amplitudes for a group, a large radiobutton will appear below, saying "amplitude constraints of this group will not be linked to other group(s)". However, if you have other groups with amplitude constraints, and you wish to link the ratios between groups, you can select this radiobutton, after which the text will say "amplitude constraints of this group will be linked to other group(s)". You must take this group linking into account when supplying the group amplitude ratios. E.g. for an ideal ATP spectrum, with amplitude ratios of 1:1 for γATP and αATP, and 1:2:1 for βATP, you must either enter these as 2:2/2:2/1:2:1 or as 1:1/1:1/0.5:1:0.5 if you wish to impose the constraint that the overall intensities of the three ATP multiplets are identical. Be aware that you can only have *one* overall group linking per parameter. I.e. the group linking for amplitude constraints can be switched on or off, and such for all groups. In this Inputvp implementation, there is no way to have multiple, different, parameter linkings between different groups.

12.5.1.4 *constraints for dampings / linewidths*

For the damping factor (linewidth) constraint of a group, you can choose between four options from the popup menu:

unconstrained
constrain, all equal
constrain, enter ratios:
fix, enter value:

If you select the unconstrained option, the linewidths of all peaks in the current group are free parameters, and can get any (positive) value, depending on the fit procedure. In this case, there is no text box on the right. If you have linear relations between the linewidths of a group, such as for the ATP multiplets in the current example, you can select the option constrain, enter ratios, after you will be prompted by an Edit Window as many times as there are peaks in the current group, just like for the amplitudes in the previous section. However, in most cases the damping factor prior knowledge will be that the linewidths of the peaks in the group are identical, whether based on true relaxation time knowledge, or assumptions on inhomogeneities and magnetic susceptibilities. In this case you can reduce the interaction of typing the ratios and simply select the constrain, all equal option. MRUI will display the ratio in the text box on the right, see e.g. Figure 51. Finally, you can fix the values of the linewidths of the current group in Hz. Although the Inputvp prior knowledge function is meant to impose relations on the parameters in the time domain model function (in this case the damping factors in kHz), you may type in the Edit Window the value of the actual peak linewidth in Hz as you know it from the FT spectrum analysis. MRUI will display the fixed value in the text box on the right.

Once you have constrained the linewidths for a group (not when you fixed the value, because then there is nothing left to link), a large radiobutton will appear below, saying "linewidth constraints of this group will not be linked to other group(s)". However, if you have other groups with linewidth constraints, and you wish to link the ratios between groups, you can select this radiobutton, after which the text will say "linewidth constraints of this group will be linked to other group(s)". You must take this group linking into account when supplying the group linewidth ratios, as explained for the amplitude constraints in the previous section. Be aware that you can only have *one* overall group linking per parameter. I.e. the group linking for linewidth constraints can be switched on or off, and such for all groups. In this Inputvp implementation, there is no way to have multiple, different, parameter linkings between different groups.

12.5.1.5 constraints for frequencies

For the frequency constraint of a group, you can choose between four options from the popup menu:

unconstrained
frequencies linked, splitting variable
frequencies linked, splitting fixed
frequencies fixed, splitting fixed

If you select the unconstrained option, the frequencies of all peaks in the current group are free parameters, and can get any value, depending on the fit procedure. In this case, there is no text box on the right. If you have linear relations between the frequencies of a group, such as for the ATP multiplets in the current example, you can select the option frequencies linked, splitting variable, MRUI will display 'link; link' in the text box on the right. Simply linking the frequencies by a variable splitting does not reduce the number of free parameters for this group, and only takes effect if you set the same constraint for other groups, and then link the variable splitting between the groups (see end of this section). For the ATP example, you could use the frequencies linked, splitting variable option for all three ATP multiplets, and then switch on the group linking for each of them, so that VARPRO would estimate the frequency splitting (i.e. the J coupling constant) for ATP, which would then be identical for γATP, αATP and βATP.
If you know the J coupling for your current application, you can use the option frequencies linked, splitting fixed. You will be prompted by an Edit Window in which you can type the frequency splitting for that group (the J coupling constant of the multiplet) in Hz. For the current example of `ischr.dat`, we selected this option for the three ATP multiplets, and entered a splitting of 16 Hz for each. Because you have actually numerically constrained the frequency parameters in this case, the group linking radiobutton will not appear. The text box on the right will display the linking of the frequencies and the value of the fixed frequency splitting in Hz, see e.g. Figure 51.
Finally, you can fix the frequency *and* the splitting of the current group. Fixing the splitting works exactly as described above. You also fix the frequency of the group by typing its value in an Edit Window, but in this case the units of the frequency you enter depend on the spectral display. If the frequency axis in your spectral window is displayed in ppm, you must also type the fixed frequency in ppm, etc. With the fixed frequency the *central* frequency of the multiplet is meant. For the βATP multiplet this is the frequency of the peak in the middle (peak 10), for an ATP doublet it is the frequency halfway between the two peaks, etc. In older MRUI versions, and in the Fortran `inputvp` program, you needed to supply the frequency of the left-most peak (upfield) to be fixed, and `inputvp` would then determine the other frequencies in the multiplet from adding multiples of the splitting. This was regarded as undesired in that most biochemists and spectroscopists would of course prefer to enter the true central frequency of the multiplet. Currently, in the MRUI Inputvp function, you can actually type

the true central frequency of a multiplet as prior knowledge. From the typed value for the fixed splitting, Inputvp will calculate a correction factor so that the fixed frequency of the left-most peak is still used in the Fortran `inputvp` program, without the MRUI user having to worry about it any more. MRUI will display the fixed values (now both in units of Hz) in the text box on the right. If you fix the frequency of a group, you *must* also fix the splitting. If your group contains only one peak and you fix the frequency, you will not be asked to enter the fixed splitting (which is a redundant parameter in that case and should be set at 0).

In the multiplet approach of this Inputvp & VARPRO implementation, the splitting for a multiplet is the distance between two individual peaks in the multiplet, which is assumed to be identical for all peaks in that multiplet. For instance for the βATP triplet with a splitting of 16 Hz, this means that the frequency difference between peak 9 and 10 is set to 16 Hz, and the frequency difference between peak 10 and 11 also 16 Hz.

Once you have constrained the linewidths for a group (not when you fixed the value, because then there is nothing left to link), a large radiobutton will appear below, saying "linewidth constraints of this group will not be linked to other group(s)". However, if you have other groups with linewidth constraints, and you wish to link the ratios between groups, you can select this radiobutton, after which the text will say "linewidth constraints of this group will be linked to other group(s)". You must take this group linking into account when supplying the group linewidth ratios, as explained for the amplitude constraints in the previous section. Be aware that you can only have *one* overall group linking per parameter. I.e. the group linking for linewidth constraints can be switched on or off, and such for all groups. In this Inputvp implementation, there is no way to have multiple, different, parameter linkings between different groups.

12.5.1.6 multiplet radiobutton

For each group that contains two, three or four peaks, the multiplet radiobutton will appear. For a group of two peaks it is labeled doublet, for a group of three peaks triplet (see e.g. Figure 51), and for a group of four peak it is labeled quartet (see e.g. Figure 50). For a different number of peaks in the group, the button is invisible. Pushing the button sets the constraints for all parameters of the group according to the theoretical relations for a regular multiplet:

phases	fixed phase relative to the zero order phase	0	
amplitudes	constrain, enter ratios:	doublet	1 : 1
		triplet	0.5 : 1 : 0.5
		quartet	0.25 : 0.75 : 0.75 : 0.25
linewidths	constrained to be all equal		
frequencies	frequencies linked, splitting variable		

Switching the multiplet button 'on' will *not* override previously set constraints in the group; it only applies the theoretical relations where previously the settings were unconstrained. Even if you do not have such theoretical multiplets in your spectrum, you could use the radiobutton to quickly select the basic options for your multiplets, after which you can make minor modifications to the constraints where necessary.

12.5.1.7 done pushbutton

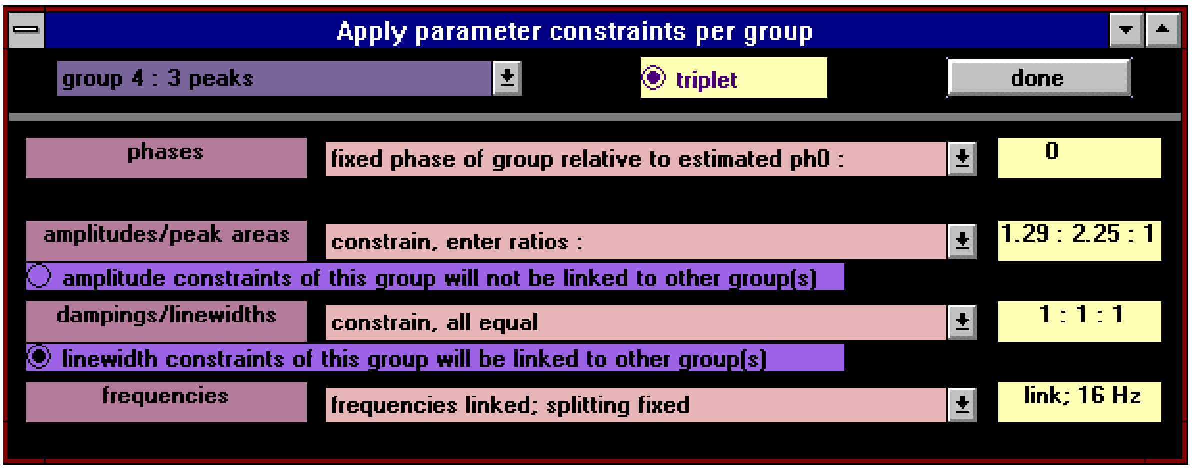
When you are finished applying the parameter constraints for all groups, you should press the done pushbutton, which writes all the necessary information to the `inputvp.ans` and `link.ans` (see section 12.6) files, and closes the prior knowledge window.

For the current example of `ischr.dat`, the following prior knowledge was imposed as parameter constraints for the four groups:

group	peaks	parameters	constraints	group link
1	1,2,3,4	phases	fixed relative to estimated ph0; 0	
		amplitudes / peak areas	unconstrained	
		dampings / linewidths	unconstrained	
		frequencies	unconstrained	
2	5,6	phases	fixed relative to estimated ph0; 0	
		amplitudes / peak areas	constrained, enter ratios; 1 : 1.09	no
		dampings / linewidths	constrained, all equal	yes

		frequencies	frequencies linked, splitting fixed; 16 Hz	
3	7,8	phases	fixed relative to estimated ph0; 0	
		amplitudes / peak areas	constrained, enter ratios; 1 : 1.18	no
		dampings / linewidths	constrained, all equal	yes
		frequencies	frequencies linked, splitting fixed; 16 Hz	
4	9,10,11	phases	fixed relative to estimated ph0; 0	
		amplitudes / peak areas	constrained, enter ratios; 1.29 : 2.25 : 1	no
		dampings / linewidths	constrained, all equal	yes
		frequencies	frequencies linked, splitting fixed; 16 Hz	

The prior knowledge window, with the parameter constraints filled in for group 4 (the βATP triplet) is displayed in Figure 51.


Apply parameter constraints per group

group 4 : 3 peaks

[✓ triplet]

done

phases: fixed phase of group relative to estimated ph0 : 0

amplitudes/peak areas: constrain, enter ratios : 1.29 : 2.25 : 1

amplitude constraints of this group will not be linked to other group(s)

dampings/linewidths: constrain, all equal : 1 : 1 : 1

[✓ linewidth constraints of this group will be linked to other group(s)]

frequencies: frequencies linked; splitting fixed : link; 16 Hz

Figure 51. The prior knowledge window with the parameter constraints for the βATP triplet group.

12.5.2 done

Using the done menu item, you can finish Inputvp without having to go into the prior knowledge window, i.e. in case you only wanted to make minor changes to the Inputvp parameters (section 12.4) after reloading a data file on which you have done a complete prior knowledge session before, or if you do not wish to apply any further prior knowledge in terms of parameter constraints.

12.6 Files produced

Once you are finished using the Inputvp function, the following files will have been produced or updated:

```
inputvp.ans
<name>.inp
link.ans
<name>.lnk
```

The `inputvp.ans` and its file-specific copy `<name>.inp` contain the answers to be given to the Fortran program `inputvp`, such that MRUI can build the input file for VARPRO (`inputvp.par`) via:

```
% inputvp < inputvp.ans
```

The linking between groups is also saved separately into `link.ans` and its file-specific copy `<name>.lnk`, in order to restore the complete previous prior knowledge settings when reloading a data file into Inputvp (see section 12.2)

13. VARPRO parameter estimation

Once you have created a set of starting values for the nonlinear parameters (frequencies and damping factors / linewidths) in a `peakpick.par` and/or `<name>.pkp` file via the Peakpick function (chapter 11), and supplied the necessary and optional prior knowledge settings in an `inputvp.ans` and/or `<name>.inp` file via the Inputvp function (chapter 12), you can let loose the variable projection[3,4,20,27,29,30] nonlinear least squares fitting algorithm without much further ado in the VARPRO function.

The practicalities of using the VARPRO function in MRUI are described using the same example data set as before, the file `ischr.dat/mat`. The VARPRO function uses both the basic and result spectral windows, see for its general menus and functions sections 5.1 and 5.2. As mentioned before, the VARPRO function has extensions to the Miscellaneous batch functions menu (section 5.1.3.5), which will be discussed here. The function-specific menu is titled VARPRO / AMARES. Its menu items are described below. Initially, after selecting VARPRO / AMARES from the quantitation popup menu of the MRUI base window (see section 3.1), the basic spectral window will be displayed empty, and after loading the file to be analyzed, its spectrum will be displayed in it, and the menu items activated (see section 5.1). Then the protocol parameters can be set according to your specific application, after which you can press go!, and the selected method in the VARPRO function will do its work. Finally, the spectral window will change size (see section 5.2) to display the VARPRO fitting results and activate the Results menu (section 5.2.1).

When you load the signal to be analyzed in the VARPRO function, its FT spectrum will be displayed as usual, but the resonances will be numbered in the spectrum, according to the order in which you selected them in the Peakpick function (see chapter 11). This is simply for recognition (especially useful if you run VARPRO on a data file for which you have not run a separate Peakpick and/or Inputvp session, relying on an existing `peakpick.par` and/or `inputvp.ans` file) and serves no further purpose at this point.
When loading a file into VARPRO, MRUI searches for a previous `<name>.pkp` and `<name>.inp` file with this data file. If no file-specific copy of either one of those is not found, it will search for the general `peakpick.par` and/or `inputvp.ans` file. If either the starting value file and/or the prior knowledge file is not found in the current directory, it will display a warning message and abort the function. If both a starting value file and prior knowledge file are found, it will check whether the number of peaks in both files corresponds. If not, MRUI will also display a warning message and abort the function. If the starting value file and prior knowledge file are correct, you can continue with the VARPRO function.

13.1 The Miscellaneous batch functions menu

13.1.1 results run i => starting values run i+1

When running a batch in VARPRO, you can now set the option that after each `i`th run, the results of file no. `i` (read from the `<name><i>.gol` file; see also section 13.7) are used as starting values for file no. `i+1` (written into the `<name><i+1>.pkp` file). This allows you to 'track' the starting values of the resonances in a batch series of files where resonances may shift frequency during an experiment (see also section 11.1.1), which might result in a significantly shorter calculation time and less failures for the batch. You activate this option using the submenu item results run i => starting values run i+1 from the batch functions submenu under the Miscellaneous menu. Conversion of results to starting values is always done subject to three checks:

- The number of iterations must be positive (negative number of iterations indicates VARPRO error, see also section 13.4).
- The number of peaks found for file `i` must be identical to the number of peaks in the starting value file for file `i+1`, if the latter exists.
- All damping factors must be negative (positive linewidths).

A forth test is optional, which is the number of iterations. This can be checked against a maximum, set by yourself (thinking along the lines of "if more than an expected `ierr` iterations were necessary, then the solution might not be ideal, so do not use those results for starting values of next run”). You activate this option using the menu item max no. iterations under the batch functions submenu. The default settings for this option can be set using the batch functions menu item under the VARPRO menu in the customize fitting parameters window (package defaults are results run i => starting values run i+1 not selected, and the max. no. iterations activated --just in case you would select the first option-- with a maximum number of iterations of 35). See also section 13.5 for the single-file-version of this approach.

13.1.2 use existing (Peakpick and) Input files when found

When running a batch in VARPRO, it will for each data file in the series check whether a file-specific Peakpick (`<name>.pkp`; except if you use the extra function described in the previous section) and Inputvp (`<name>.inp`) file exists, see also section 11.1.1 and the introduction of this chapter. If it does not find one, it will use the general file (`peakpick.par` and `inputvp.ans`, respectively), and after a successful fit on that data file, copies the general file to the file-specific Peakpick and/or Inputvp files for that particular data file. This means that once you re-do a VARPRO analysis on a batch, it will find a file-specific `<name>.pkp` and `<name>.inp` for each data file in the batch.

This may actually be very undesired. Suppose you ran a batch on a set of signals, having carried out the Peakpick and Inputvp functions only on the first data file of the series, selecting 11 resonances. Upon considering the results of the batch analysis, you come to the conclusion that you wish to change some of the prior knowledge, or even that you should have fitted an extra resonance for more accurate quantitation (e.g. the NAD that is overlapping the αNTP resonance). You re-do the Peakpick and Inputvp for your first data file, which creates a new `<name>1.pkp` and `<name>1.inp`, as well as a new `peakpick.par` and `inputvp.ans`. Then you run the batch VARPRO again. However, as soon as VARPRO finished the first file (now using 12 peaks), it will find for the second data file find the `<name>2.pkp` and `<name>2.inp` which remained from the previous batch analysis, and so on for all the other files of the batch. Hence it will never use the new `peakpick.par` and `inputvp.ans`, and all files of the batch, except the one on which you carried out the Peakpick and Inputvp function, will be fitted with 11 resonances and the old prior knowledge. Apart from the obvious problem with this, it will also make that the number of columns in the batch result files (see section 13.7) is different for row 1 and the subsequent rows, which means that the Plot batch results function (section 16.3) will crash (although you can still use these files in your spreadsheet software).

The straightforward solution to this problem would be to remove all the `*.pkp`, `*.inp` and `*.lnk` files for that batch (except perhaps for the first file if you want to re-use your previous settings) in that data directory before you use the Peakpick and Inputvp again. However, the use existing (Peakpick and) Inputvp files when found menu item takes a more sophisticated approach. The part Peakpick and is in parentheses, because if you selected the results run i => starting values run i+1 menu, the existing Peakpick file is not used anyway (rather the VARPRO result file `<name>.gol` from the previous file in the batch, see previous section), and MRUI changes the menu item to use existing Inputvp files when found, and changes it back to use existing Peakpick and Inputvp files when found if the results run i => starting values run i+1 menu is unselected. The default settings for this option can be set using the batch functions menu item under the VARPRO menu in the customize fitting parameters window (package defaults are use existing Peakpick and/or Inputvp files when found selected). If the menu item is *not* selected, the VARPRO function will *not* check for the existence of the `<name><b>.pkp` and `<name><b>.inp` files during the batch, but only use the `peakpick.par` and `inputvp.ans` files, which have been created by your use of the Peakpick and Inputvp functions for the data file of your choice to indicate the new starting values and prior knowledge, perhaps even for a different number of peaks.

13.2 algorithm

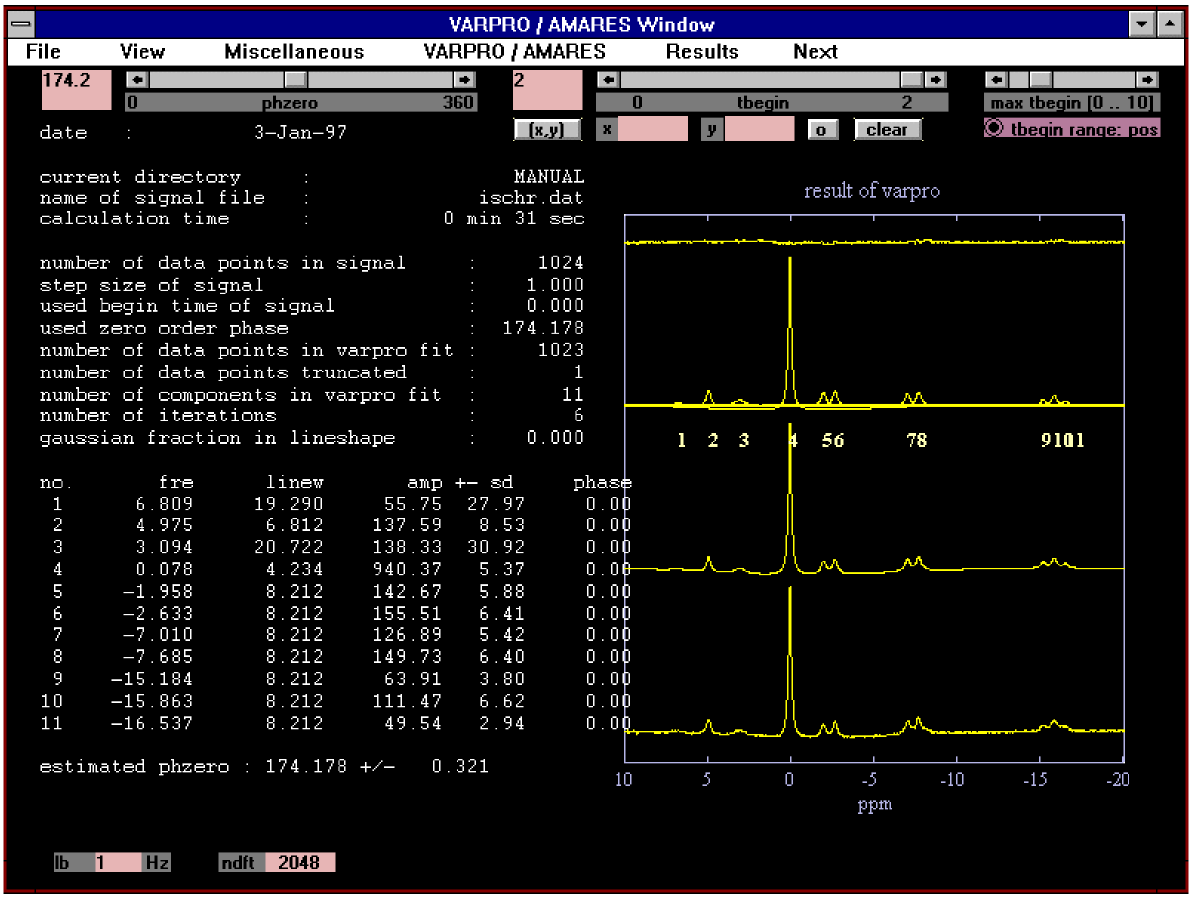
This menu item lets you select which variable projection algorithm is used for the nonlinear least squares VARPRO fit. The bol_vp (varpro) algorithm is the original Fortran implementation of Golub and Pereyra’s algorithm by Bolstad, as built into the VARPRO method dedicated to MRS application by van der Veen *et al.*[29] The minpack (varpro) algorithm is a faster and more robust nonlinear least squares minimization algorithm from the Minpack library of Netlib[32]. In the theoretical case where all frequencies and damping factors would be fixed, the entire minimization procedure would be reduced to a linear least squares algorithm. In such a case the minpack algorithm would no longer be the optimal algorithm, and a built-in detector immediately switches to a more accurate and robust *linear* least squares algorithm. The default algorithm can be set using the algorithm menu item under the VARPRO menu in the customize fitting parameters window (package default is minpack). For the current example file the MINPACK algorithm was used. The nl2sol (amares) algorithm is the new improved nonlinear least squares algorithm corresponding to the AMARES method (see chapters 14 and 15).

13.3 ndp

This is the total number of data points that should be included in the VARPRO fitting process. You can change it by typing a new number into an Edit Window. For the SVD-based methods (see section 10.2), it was necessary to closely consider the number of data points where the FID has decayed, otherwise noise-only points would be included in the matrix algebra fit, which would reduce its performance, both in terms of accuracy and calculation time. For the VARPRO method, including extra points beyond the end of the FID only results in slightly longer calculation times (linear relation between calculation time and number of data points included). Because the nonlinear least squares minimization is a maximum likelihood (ML) estimator[31] in the case of white Gaussian noise, including extra points beyond the end of the signal contributions does not impede the accuracy of the fit, and in fact leads to better validity of the ML assumptions and the derivation of the Cramér Rao lower bounds for the standard deviations of the estimated parameters. When you load a signal, a default value for `ndp` is set, and displayed in the menu item. This default value is normally taken as the minimum of the default value set in the customize fitting parameters menu for the VARPRO algorithm (package default is 1024, see also section 3.2.1.4), the actual number of signal points in the data file, and the maximum number of data points allowed for the method. If you change the value for `ndp`, it will again be tested against those maxima. For the example signal of `ischr.dat`, it could be seen in the FID plot (see Figure 55 in section 16.1) that 256 data points may be just sufficient to include the signal contribution and exclude noise-only points, but all 1024 signal points were included in the VARPRO fit for reason explained above.

13.4 go!

After the algorithm protocol has been set according to your wishes, select the go! menu item to start the VARPRO nonlinear least squares fitting and parameter estimation. The iteration progress of the algorithm can be watched in the Matlab command window. If you run VARPRO on a PC, this will be displayed in a separate DOS window. If you had loaded a batch, the whole series of files will now be processed using the same algorithm and number of data points. As for the starting values and prior knowledge used throughout the batch, see sections 11.1.1 and 13.1. The basic spectral window will display a message (“running VARPRO”; with serial index for batch processing) in the text bar in the lower right corner of the window, and after it has finished it will display the VARPRO results both as a numerical table and a multi-spectra display, using the result spectral window (section 5.2). See Figure 52. The table on the left displays some information about the data file, and the VARPRO process. The display of the numerical table and spectra has been explained in section 5.2.



```
date      :              3-Jan-97

current directory        :                MANUAL
name of signal file      :             ischr.dat
calculation time         :         0 min 31 sec

number of data points in signal      :    1024
step size of signal                  :   1.000
used begin time of signal            :   0.000
used zero order phase                : 174.178
number of data points in varpro fit :    1023
number of data points truncated      :       1
number of components in varpro fit   :      11
number of iterations                 :       6
gaussian fraction in lineshape       :   0.000

no.      fre     linew      amp +- sd      phase
 1      6.809   19.290    55.75  27.97     0.00
 2      4.975    6.812   137.59   8.53     0.00
 3      3.094   20.722   138.33  30.92     0.00
 4      0.078    4.234   940.37   5.37     0.00
 5     -1.958    8.212   142.67   5.88     0.00
 6     -2.633    8.212   155.51   6.41     0.00
 7     -7.010    8.212   126.89   5.42     0.00
 8     -7.685    8.212   149.73   6.40     0.00
 9    -15.184    8.212    63.91   3.80     0.00
10    -15.863    8.212   111.47   6.62     0.00
11    -16.537    8.212    49.54   2.94     0.00

estimated phzero :  174.178 +/-    0.321
```

Figure 52. VARPRO result window after parameter estimation on ischr.dat. All 11 resonances were fitted as indicated in the Peakpick function, using biochemical and experimental prior knowledge (via Inputvp) to control the physical relevance of the solution.

With this example, the power of the VARPRO function (whichever algorithm selected) for parameter estimation of *in vivo* MRS data sets, as compared to the performance of the SVD_1D methods (see section 10.7) becomes apparent. It is clear that the VARPRO solution is much more physically relevant than that of the SVD_1D method in Figure 45. The PME resonance has now been fitted, despite it being broad and of low intensity, and the individual phases of all fitted resonances are identical and equal to the overall zero order phase of the spectrum, which itself was estimated by the VARPRO algorithm, thus reducing operator dependence. Similarly, the ratios of the amplitudes (or peak areas) of the ATP multiplets now conform to our expectations. In this particular case, there might be a small contribution of NAD to the right of the αATP doublet, but only the αATP resonances have been fitted, and the possible NAD contribution ended up in the residual. The NAD resonance could have been fitted as a peak by providing an extra resonance with starting values at the supposed position of NAD in the Peakpick function (followed by updating the Inputvp files for this signal). Finally, no spurious peaks have been fitted. The increased control over the parametric fit renders the VARPRO method very reliable even under poorer *in vivo* MRS measurement conditions. The compromise of having to supply starting values and prior knowledge (i.e. more operator interaction) is hopefully made less obstructive by the extensive automation features of the MRUI Peakpick, Inputvp and VARPRO functions, combined with the possibility of setting all the default options and values yourself via the customize menus (section 3.2.1).

The spectra in the VARPRO result display are phased such that the reconstructed spectrum (i.e. with the zero order phase and the begin time as resulting from the VARPRO analysis) is phased. The other spectra are phased accordingly, and the phzero and tbegin sliders updated. If the VARPRO fitting was carried out with free phases for all peaks (i.e. zero order phase constrained but individual phases free), the spectra are phased to the zero order phase of the display from before the VARPRO (as read from the phzero slider value in the `<name>.gra` file; this could e.g. be from when you set the zero order phase as a fixed value in the Inputvp function). With respect to the zero order phase and begin time, it was mentioned in section 12.4.6 that it is theoretically impossible to estimate the zero order phase and/or begin time when fitting only one peak. When

VARPRO has for some reason been allowed to do such a run, MRUI will salvage the results as best it can, and display an info message about what happened.

If a VARPRO run crashes in single file mode, a warning message will be displayed in the special message window, and the VARPRO function aborted. If a particular run crashes in batch mode, a message containing the corresponding serial number of that file will be displayed in the Matlab command window, so that at the end of an automatic batch you can check from the Matlab command window for which files the VARPRO fit was unsuccessful. You could then re-do these files in single file mode, perhaps after supplying new starting values or prior knowledge for them.
For the bol_vp algorithm, the `ierr` error messages (an error message indicated by a negative final number of iterations) have the following meaning:

`ierr`	error description in terms of Fortran algorithm
-1	Iterations terminated because maximum number of iterations reached. Currently set at 50.
-2	Iterations terminated due to ill-conditioning (Marquardt parameter too large). See also `ierr=-8`.
-4	Input error in parameter N, L, NL, LPP2 or NMAX.
-5	`INC()` matrix improperly specified, or P disagrees with LPP2.
-6	A weight was negative.
-7	A "constant" column was computed more than once.
-8	Catastrophic failure - a column of matrix A has become zero.
-9	`INC()` matrix contains elements not equal to `1` or `0`.

If the VARPRO analysis would have given estimated parameters so large, or with such large standard deviations (e.g. when the fitting was unsuccessful), that the numerical values could not fit into the prepared spaces in the result files, character strings, such as `********` or NaN or `Inf`, would be written to the result files. MRUI detects these characters, and appropriate measures are taken. If standard deviations are so big that they do not fit, the fit was unsuccessful, and the standard deviations are set to `Inf` (infinity) in the numerical result table and in the `<name>.vns` file. As for the parameters, their estimated values are *always* recovered (via the `<name>.gol` file, see section 13.7).

VARPRO checks after the analysis whether all damping factors are smaller than zero. If this is the case (physical solution) the reconstructed model function is extrapolated to the number of data points present in the original signal (if of course larger than the value for `ndp`). This avoids heavy sinc wiggles throughout the displayed reconstructed spectrum when fitting slowly damped signals with a small number of data points.

The VARPRO function can also calculate a value for the signal-to-noise ratio (SNR) of the time domain signal in dB, calculated from the estimated amplitudes and damping factors, and the standard deviation of the noise, via the ^{10}log of (energy in signal over energy in noise). This may be used to compare signals according to time domain fitting difficulty. The SNR values in dB are stored in the `<name>.vns` file (see section 13.7). Be aware that this calculation may make your overall VARPRO analysis times considerably longer. You can define whether or not VARPRO should calculate the SNR in the customize fitting parameters menu for calculate SNR (dB) under the VARPRO menu (package default is no, see also section 3.2.1.4).

The VARPRO function provides facilities for absolute quantitation and calculation of pH (see also section 13.6). This may require the following actions from you after the result spectral window has been displayed, and a list of estimated parameters with peak numbers has been displayed in the Matlab command window:

- If the nucleus set in `default.xpt` is proton (h, see section 7.1.1), and no logical corresponding water file can be found, you will be suggested to indicate an appropriate water file for the internal concentration reference. This is done via a file manager window. You select which data file contains the unsuppressed water signal corresponding to the current metabolite signal (e.g. a `<name>w.dat`, see section 4.5). MRUI then automatically runs a Lanczos HLSVD on it as described in section 10.8, producing the `<name>.wat` file from which the unsuppressed water signal amplitude can be read. You also need to indicate which peaks in the reconstructed spectrum (the latter are numbered, see Figure 52) correspond to the metabolites you wish to quantify. Absolute metabolite concentrations are then calculated according to a set of default values currently implemented in the MRUI software. This static approach (for ^{1}H MRS only as yet) will soon be replaced by a fully flexible customize absolute quantitation menu (see section 3.2.1.5). The current defaults are (T_1 and T_2 corrections not yet included either; the assumed brain water concentration is 83.4 M[33]):

metabolite	label	resonance	number of protons	approx. ppm
water	`H2O`	H_2O	2	4.7
myo-inositol	`mI`	$-CH_2O$	2	3.56
total choline	`tCho`	$-CH_3$	9	3.22
(phospho-)creatine	`Cr/PCr`	$-CH_3$	3	3.03
N-acetyl aspartate	`NA`	$-CH_3$	3	2.02
lactate	`Lac`	$-CH_3$	3/2	1.33
other	type in		type in	

- If the nucleus set in `default.xpt` is phosphorus (`p`, see section 7.1.1), and you have selected one or more of the pH calculation options (either through the calculate pH menu of the VARPRO function or via the customize fitting parameters menu), you will be prompted to indicate which peaks in the reconstructed spectrum represent the relevant metabolite resonances. See section 13.6.

If either of these two functions have been used, the results will be appended to the `<name>.vns` file (see also section 13.7). Be aware that if you are prompted to indicate peak numbers either for absolute quantitation or for pH calculation, that none of the menus or graphical features will work. Only when you have properly finished the user interaction in the Matlab command window will you be able to use the Results menu, sliders, etc.

13.5 recycle results

This menu item is enabled (active) only when a single file has been loaded, but then both before and after the VARPRO run. Its function is to create a new set of starting values for the loaded file from the results of a previous VARPRO run on that same signal. This may be useful if your VARPRO run has given you results of insufficient accuracy, for which you want to modify some of the prior knowledge, and re-run the VARPRO analysis. However, those previous results might have been already better than the original starting point, so taking those previous results as new starting values will also improve on the performance. This is similar to the ‘tracking’ approach for batch processing as explained in section 13.1, but then with respect to the same data file. If you select the recycle results menu item *before* you press go!, MRUI will search for the `<name>.gol` file (see also section 13.7), and then convert its contents into a new `<name>.pkp` and `peakpick.par` file. It will display a message “Results recycled, ready to run VARPRO (go!)” in the message text bar in the lower right corner, and when you next press go!, you will start the VARPRO analysis using the results from a previous analysis as new starting values. If you select the recycle results menu item immediately *after* the VARPRO analysis has finished, i.e. in the results spectral window, VARPRO is re-launched with the results just obtained as new starting values. Effectively, you simply make VARPRO continue where it has just stopped, and therefore this function is only relevant if your original VARPRO fit did not fully converge, but reached the maximum number of iterations and stopped.

13.6 calculate pH

This menu item is only enabled (active) when the nucleus set in `default.xpt` is phosphorus (`p`, see section 7.1.1), but then both before and after the VARPRO run. It allows you to calculate intra- or extracellular pH based on the chemical shift differences between P_i and αATP, P_i and PCr, or APP (extracellular marker) and αATP. MRUI calculates the internal (if the chemical shift of P_i is measured) or external pH (if the chemical shift of APP is measured) based on the calibration equation as presented by Prichard *et al.*[34], currently still using a set of default constants[†]. This static approach will soon be replaced by a fully flexible customize absolute quantitation menu (see section 3.2.1.5). The current defaults are:

metabolite	shift to P_i			shift to APP		
	pK_a	c_1	c_2	pK_a	c_1	c_2
PCr	6.66	3.079	5.57			
αNTP	6.66	10.649	13.14	6.91	31.11	34.30

[†] Kindly provided by Dr. Cheryl McCoy, St. George’s Hospital Medical School London (UK), Division of Biochemistry, CRC Biomedical Magnetic Resonance Research Group.

Should you wish to change these default values in the mean time, edit the MRUI script for pH calculation, which lists these parameters; it is `$MRUIDIR/mrui/utils/phcalc.m`.
If an estimated chemical shift is out of bound with respect to the calibration equation (i.e. one term in the logarithmic term would become smaller than zero), a message will be displayed and the corresponding pH calculation aborted. The pH calculation using αNTP as a chemical shift reference needs the central frequency of αNTP; after fitting a doublet for αNTP, MRUI will ask for both peak numbers of the αNTP doublet, and use the average frequency of the two for the pH calculation.
If you select one or more calculate pH menu items *before* you press go!, or if one or more were set by default using the customize fitting parameters menu, MRUI will prompt you immediately after the VARPRO analysis to indicate which peak numbers in the reconstructed spectrum correspond to the relevant metabolites; see the end of section 13.4. MRUI then calculates the desired pH(s) after having calculated the necessary chemical shift(s) as the difference(s) between the estimated central frequencies for the corresponding resonances, and includes the results in the `<name>.vns` file. If you select a calculate pH menu item *after* the VARPRO analysis has finished, i.e. in the results spectral window, you are immediately prompted to indicate which peak numbers in the reconstructed spectrum correspond to the relevant metabolites, after which MRUI will calculate that pH and append the result at the end of the `<name>.vns` file. For the current example of `ischr.dat`, the submenu item Pi-PCr was selected after the VARPRO analysis had finished, which resulted in the following interaction in the Matlab command window:

```
»
 peak number for Pi:   2

 peak number for PCr:  4
```

after which the following line was appended at the end of the `ischr.vns` file:

```
pH measured using Pi  and PCr       :  7.092  +/-   0.001
```

The pH estimation seems to be so accurate because the standard deviation for the pH is calculated from the standard deviations (Cramér Rao lower bounds) of the estimated frequencies involved. The latter are so low because generally frequencies can be estimated very accurately using a time domain fitting method. This high accuracy of the calculated pH value might be deceiving, because biological variability will usually lead to a population standard deviation of around 0.1 pH unit.

13.7 Files produced

Once you are finished using the VARPRO function, the following files will have been produced or updated:

file produced	contents
`<name>.tsv`	information on number of singular values found at each update of Lanczos iterations (only for Lanczos HLSVD)
`<name>.gol`	simple parameter estimation output file from VARPRO method (formerly known as `golscr.par` file)
`<name>.vrs`	parameter estimation output file with better layout and Cramér Rao standard deviations (listed in table as *two times* CR!) with parameters (formerly known as `outputvp.par`)
`<name>.vns`	presentation file of numerical parameter estimation results, containing output of `<name>.vrs` file, but with estimated frequencies in units as displayed in the spectral display on screen (e.g. ppm), damping factors translated to linewidths at half height in Hz, and with Cramér Rao standard deviations listed as one sd, i.e. parameter ± CRsd also contains an extra table with amplitude estimates and CR values scaled back to original intensities if carried out, this file also contains absolute metabolite concentrations, pH estimations and/or SNR in dB
`<name>.lrv`	VARPRO analysis report, directly taken from the list in the left half of the result spectral window
`<name>.wat`	auxiliary ASCII file containing estimated amplitude and linewidth of water signal, in case absolute concentration was carried out and no `<name>.wat` file found at the time

`<name>.pkp`	only if you used the recycle results menu
`peakpick.par`	only if you used the recycle results menu

Running VARPRO in batch produces extra files which list the estimated parameters for all peaks in all files of the batch series as follows:

file produced	tabulates	corresponding file	tabulates
`<name>.amp`	estimated amplitudes (in arbitrary units)	`<name>.ams`	Cramér Rao standard deviations on estimated amplitudes
`<name>.aam`	absolute amplitudes (as true intensities in the FIDs, i.e. corrected for scaling)	`<name>.aas`	Cramér Rao standard deviations on absolute intensities
`<name>.fre`	estimated frequencies (in units as displayed on screen, e.g. Hz or ppm)	`<name>.frs`	Cramér Rao standard deviations on frequencies
`<name>.lw`	estimated linewidths (in Hz)	`<name>.lws`	Cramér Rao standard deviations on linewidths
`<name>.pha`	estimated phases (in degrees)	`<name>.pas`	Cramér Rao standard deviations on phases
`<name>.ph0`	estimated zero order phase (in degrees; if estimated)	`<name>.p0s`	Cramér Rao standard deviations on zero order phase
`<name>.t0`	estimated begin time (in ms)	`<name>.t0s`	Cramér Rao standard deviations on begin time
`<name>.ph`	estimated pH (if calculated)	`<name>.phs`	Cramér Rao standard deviations on pHs
`<name>.cnc`	absolute metabolite concentrations (if calculated)	`<name>.cns`	Cramér Rao standard deviations on absolute concentrations
`<name>.wal`	estimated amplitude and linewidth for unsuppressed water signal (single Lorentzian)		

The first column contains the serial numbers of each batch-processed file, and the following columns contain the respective parameter for all peaks. You can directly load these ASCII column files into a spreadsheet or graphics program. The serial batch results can be plotted using Plot batch results (see section 16.3). The files containing the Cramér Rao estimates for the standard deviations on the estimated parameters can be used in your spreadsheet software in order to create graphs with error bars.

14. Prior knowledge for AMARES: Inputnv

This chapter was written by:
Leentje Vanhamme
Katholieke Universiteit Leuven
Department of Electrical Engineering
ESAT / SISTA *tel* *+ 32 - 16 – 32 17 97*
Kardinaal Mercierlaan 94 *fax* *+ 32 – 16 – 32 19 86*
B – 3001 Leuven – Heverlee *e-mail* *Leentje.Vanhamme@esat.kuleuven.ac.be*

AMARES stands for Advanced Method for Accurate, Robust and Efficient Spectral fitting. It was developed to be the successor of VARPRO. It is the result of a 2 year study as part of the Ph.D. project of L. Vanhamme carried out under supervision of A. van den Boogaart and S. Van Huffel. AMARES, like VARPRO, is based on solving a nonlinear least squares problem. AMARES however minimizes the squared difference between the data and the model function instead of a more complicated functional. It is shown in Ref. 28 that as a result, AMARES is a more robust method than VARPRO. To actually solve the minimization problem, `dn2gb`, a version of NL2SOL, is used. The `dn2gb` algorithm was obtained from the PORT library of Netlib[35]. AMARES also uses a singlet approach for imposition of prior knowledge instead of the multiplet approach of VARPRO, which greatly extends the possible combinations of prior knowledge that can be invoked. Note that everything that was possible with VARPRO is still possible with AMARES. This will be illustrated using the current example file `ischr.dat/mat`, containing an *in vivo* ^{31}P MRS signal from human muscle at 1.5 T. Examples of the extended possibilities of imposing prior knowledge will also be given.

AMARES uses an entirely new prior knowledge function: Inputnv. The Inputnv function uses the basic spectral window (see section 5.1for its general menus and functions) and a special prior knowledge window. The function-specific menu in the basic spectral window is titled Inputnv and its menu items are described below. Initially, after selecting Inputnv from the quantitation popup menu of the MRUI base window (see section 3.1), the basic spectral window will be displayed empty, and after loading the file to be analyzed, its spectrum will be displayed in it, and the menu items activated (see section 5.1). You can then start imposing the prior knowledge.

14.1 Prior knowledge and parameter relations

A lot of prior knowledge can already be implemented using VARPRO and this has been crucial in the analysis of MRS data in the past. However, recently more prior knowledge has become available which VARPRO is not able to incorporate in its data fitting procedure. In this section some examples are given to illustrate the improved possibilities of imposing prior knowledge using AMARES.
Suppose we have 4 peaks of which we know the phase (relative to the zero order phase). If these phases are all different, then, using VARPRO / Inputvp, we have to put each peak in a different group. It then becomes impossible to link the frequencies between these peaks since this can only be done between peaks belonging to the same group. If, for instance, the frequency splittings between the peaks are known in addition to the individual phases, this frequency prior knowledge cannot be imposed in VARPRO. Also, due to the fact that VARPRO allows only one overall linking for the dampings (or amplitudes), it is not possible to impose that e.g. the dampings of peak 1 and 4 are equal as well as the dampings of peak 2 and 3 (without 1, 2, 3 and 4 being all identical).Another example is formed by the six glycogen resonances in ^{13}C MRS, for which the relative frequency splittings are known but different. This can therefore not be implemented as a regular multiplet in VARPRO.
In AMARES this type of prior knowledge can easily be incorporated. These are only a few examples of the increased possibilities of AMARES. Basically, using AMARES, individual parameters of a peak can be linked to an arbitrary reference peak and a peak can have different reference peaks for different parameters. An identical treatment of all parameters is possible. Each of the parameters can be unconstrained or kept fixed. A fixed shift or ratio with respect to any fixed or unconstrained parameter of the same type can be imposed. It is also possible to impose a *variable* shift or ratio with respect to any unconstrained or fixed parameter of the same type. These variable shifts or ratios can then be linked between different groups of peaks.
In addition, AMARES makes it possible for each peak individually to choose the lineshape to be Lorentzian or Gaussian. You can also impose lower and upper bounds for the estimated parameters, so-called “soft constraints”. If you do not impose specific soft constraints, AMARES will automatically select a relevant default

interval (like e.g. the spectral width for the frequencies). In addition, it is possible to fit complete echo signals, which means that you can use all the information present in an echo signal instead of ignoring its left part.
All this will be explained in more detail in the following sections.

14.2 The File load menu

The only difference with the corresponding section of chapter 12 is that Inputnv produces the files `inputnv.ans` and `<name>.ipk` (instead of `inputvp.ans` and `<name>.inp` for VARPRO) . See section 12.2 for further details on this menu.

14.3 The Inputnv menus

14.3.1 truncate initial points, frequency selective, noise level

All these features are the same as explained in the corresponding sections 12.4.2, 12.4.3 en 12.4.5 of chapter 12.

14.3.2 zero order phase & begin time

For the general explanation on these parameters see section 12.4.6. In addition, for AMARES, the upper and lower bounds of the zero order phase are always set to -360^0 and 360^0.
If you choose to estimate the begin time you can enter lower and upper bounds on `tbegin` using the soft constraint boxes of the phase of the *first* peak. A text box above the phase text box indicates this.

14.3.3 lineshape

With this menu item you can select whether the lineshape of all individual components to be fitted should be Lorentzian or Gaussian. If you select the option mixed you will be presented with the option to choose between a Lorentzian or Gaussian lineshape for each peak individually in the prior knowledge window. The package default is Lorentzian.

14.3.4 echo signal

14.3.4.1 imposing prior knowledge

If your data are FIDs you do not need this feature. However, if you analyze echo signals and you want to use all the available data, this feature is very important. Essentially, an echo is modeled as two FIDs back to back. The left and the right part of the echo are considered to have the same amplitudes, frequencies and phases but different dampings. The dampings of the left and right part can however be linked using AMARES as we explain below.
When you select the echo menu item the following menu appears:

no
number of points in left half: 0
same damping prior knowledge left and right
same damping values left and right

The three last options are not activated. If you change no to yes, you have the possibility of selecting one of the other three features. If you click on the second menu item you will be presented with an edit box in which you have to type the number of points in the left half of your echo signal. The top of the echo is considered to belong to the right part of the echo. Of course the top of the echo does not need to coincide with an actual data point so it is important that you take that into account. Take, for instance, a signal consisting of 256 data points and suppose the top of the echo lies between the 50th and the 51st data point. The number of points in the left part is then 50. If the top like in this case does not coincide with an actual data point you must leave the value of the begin time unconstrained. AMARES will then return a negative begin time, smaller than the time step between two samples. This negative begin time indicates that the 51st data point is not the actual time origin but that its position is somewhere between data point 50 and 51. It is important that you check the actual value AMARES returns for the begin time in this case. This value should be negative and smaller than the time step between two samples. If this is not the case your original assumption about the number of data points in the left part of the echo is wrong and you *should start all over with a different value for the number of peaks in the left part of the echo.*

The dampings of the left and right part of the echo do not need to be the same in an echo signal. But normally you will have the same kind of constraints on the dampings for the left and the right part. Therefore in MRUI we presented you with the option same damping prior knowledge left and right.

That means that if e.g. the dampings of peak 1 and 2 are equal in the right part of the echo you also suppose the corresponding dampings to be equal in the left part of the echo. If you choose the option same damping values left and right you indicate that the dampings *values* of the left and right part of the echo are equal. Make sure that a check mark appears to the left of the option you selected, otherwise you did not select anything.

If you choose neither of the last two options, the dampings of the left part of the echo are unconstrained and will all be fitted independently of each other and of the dampings of the right part of the echo.

14.3.4.2 displaying and peakpicking echo signals: a work-around

Since the fitting of echo signals is a brand new feature in MRUI it is not yet possible to visualize and peakpick echo signals directly. This will be added in a future MRUI release. But do not worry, there is a simple work-around. Of course some extra manipulations are needed to work with a full echo signal, but this disadvantage does not outweigh the advantages of being able to work with full echo signals...

Suppose your echo signal is saved as `echo.dat`. Starting from that file you have to generate a file containing only the right part of the original echo signal. This can easily be done using theTruncate function from the FID Maths library as explained in section 8.5. This function creates an `echo_t.dat` file, which can be displayed normally. Next you go on with the Peakpick function to provide starting values for the frequencies and the dampings of the right part of the echo. If the dampings of the left part are left unconstrained, AMARES takes the dampings of the right part of the echo as starting values for them. Since the dampings of the left and right part of the echo do not differ too much in general, this causes no problems. After you finished peakpicking you continue with the Inputnv AMARES function where you enter the prior knowledge for the signal as if it were the real echo signal `echo.dat`. So you have to select yes from the echo menu item, after that you have to enter the number data points in the left part of the echo signal and you have to indicate how you want to link the dampings of the left part of the echo to those of the right part. You can then go on imposing other kinds of prior knowledge on the parameters. After finishing the input of the prior knowledge, you should type the following commands in your operating system shell (make sure that you type these commands in the directory where your data files are)

```
 % cp echo_t.pkp echo.pkp
 % cp echo_t.ipk echo.ipk
 % cp echo_t.gra echo.gra
```

You then select VARPRO / AMARES from the MRUI base window and load the `echo.dat` file using the load file option from the load menu. A figure will be displayed which looks a bit strange but that does not matter. Choose the nl2sol (amares) algorithm from the VARPRO / AMARES menu. If you do not want to change the total number of data points that should be included in the AMARES fitting process, you can select the go! menu item.

After having run AMARES, MRUI displays the results both as a numerical table and a multi-spectra display, using the result spectral window. However, no information about the dampings of the left part of the echo is displayed. To see the estimated parameters, you have to take a look inside the files `<name>.gol` and/or `<name>.vrs`, see also section 13.7.

14.4 The prior knowledge menus

14.4.1 prior knowledge

To apply the available prior knowledge as linear constraints between parameters you select the prior knowledge menu item, after which the prior knowledge will be displayed for each peak individually, starting with peak 1.

14.4.1.1 constraints for phases

For the phase constraint of a peak, you can normally choose between six options from the popup menu:

unconstrained
impose shift
impose ratio
impose variable shift
impose variable ratio
fix value

If you asked to estimate the zero order phase and/or begin time only the fix value option can be used, because otherwise you would get a mathematical problem with too many degrees of freedom. For this reason all the six options will only be found in the popup menu when the zero order phase and the begin time are fixed to a value.
The unconstrained option indicates that the phase of the peak is a free parameter and can get any value, depending on the fit procedure. In the latter case you also have the option to impose soft constraints on the phase of the peak. This means that if you know the upper and lower bounds for the phase of the peak, you can enter these values in the two boxes that appear on the right if you select yes from the soft constraint button. If you do not impose soft constraints on the phase explicitly, AMARES will automatically select the values -360^0 and 360^0, which are mathematically relevant. If you set the soft constraint radiobutton to yes, by default -360 and 360 will appear for the lower and upper bound respectively. You can then change these values by placing the cursor in the edit box on the right and either selecting or backspacing over the current value, and typing your new desired value.
The impose shift option allows you to impose a fixed phase shift with respect to any other peak with a phase that is either left unconstrained or fixed to a value. If you select this option an edit box on the right will appear where you have to enter a numerical value corresponding to the phase shift (in degrees) you want to impose. The default value filled in is zero. You can change the relative value for the phase constraint if you want to. To the right of this box is a pull down menu from which you have to choose the reference peak. The peaks marked with an “X” in the popup menu cannot be used as a reference peak. It is e.g. clear that it is not possible to link a peak to itself. Take e.g. an example with three peaks. Suppose you know the relative phase difference between the peaks. You can choose either one as a reference peak. Suppose we take peak 1 as a reference. Then, for peak 2, we can express the relative phase shift with respect to peak 1. For peak 3 we cannot take peak 2 as a reference (although we also know the phase shift of peak 3 with respect to peak 2), we have to take peak 1 as a reference. Note that this does not impose any kind of restrictions on the phase shifts that can be imposed, it is just a matter of keeping track of the imposed constraints in an easy way.
The impose ratio option functions in exactly the same way as the impose shift option. The only difference is the fact that this option allows you to impose fixed ratios between phases instead of fixed shifts.
The impose variable shift option is a bit more complicated to explain. So take as an arbitrary example a signal consisting of three peaks. Imposing a variable shift leads to the introduction of a new variable Δ.We can then e.g. express the following numerical relations between the phases of the peaks as ϕ_1, $\phi_2=\phi_1+ x\Delta_1$, $\phi_3=\phi_1+ y\Delta_1$; where x and y are numerical values supplied by the user. If x is e.g. 1 and y is 2, we express that the phase difference between peak 1 and peak 2 is the same as the one between peak 2 and peak 3. Using the prior knowledge window we can express this as follows. For peak 1 we choose the unconstrained option. For peak 2 we select the impose variable shift option from the pull down menu. Three boxes will appear to the right. In the first one we enter 1 (the x value). Then we have to select a reference peak. In this example the reference peak is peak 1. The third box indicates the ordinal number of Δ. In this case we enter 1. For the third peak we also select the impose variable shift option and we fill in 2 in the first box. As a reference peak we select again peak 1 and we enter 1 as number of Δ. Indeed it is possible to introduce more than one variable Δ. Suppose that in the same example you have another set of three peaks with also equal relative phase differences, but different from Δ_1. Then you can express this using a second variable Δ_2, yielding ϕ_4, $\phi_5=\phi_4+ \Delta_2$, $\phi_6=\phi_5+ 2\Delta_2$. In this case you have to enter 2 as ordinal number of Δ. It is important that in the case you have several Δ’s that you number them with subsequent integer numbers starting with 1 (1,2,3,…), otherwise you might get undesired behavior.
The impose variable ratio is analogous to the impose variable shift option, the only difference is that instead of variable shifts you can impose variable ratios.

The fix value option allows you to enter a numerical value as a constraint in the edit box on the right, which it will insert in the model function for the individual phase of the peak. By default MRUI displays 0. If you asked to estimate the zero order phase and/or the begin time, you will only be presented with the option fix value. If you then leave the phase of each peak equal to 0 relative to the estimated zero order phase, all peaks will get the phase that AMARES estimates for the zero order phase. If you fixed the zero order phase and you enter a value for the phase of a certain peak in the edit box, the total phase of the peak will be the zero order phase + the value you entered.
When using Inputnv on PC, do not use values smaller than -10^{97}, values between -10^{-97} and 10^{-97}, or values larger than 10^{97}. This is because exponential numbers with more than three digits in the exponential cannot be written to a file in Microsoft Fortran.

14.4.1.2 constraints for amplitudes / peak areas

When you want to impose constraints on the amplitudes you will always be presented with the following six options from the popup menu:

unconstrained
impose shift
impose ratio
impose variable shift
impose variable ratio
fix value

Since AMARES treats all the parameters identically there is a large similarity between imposing prior knowledge for the different parameters. Therefore, in order not to repeat the explanations we already gave for the phases, we highlight here the special points you have to take into account when imposing constraints for amplitudes.
If you select the unconstrained option, the amplitude is a free parameter and you can get any *positive* value, depending on the fit procedure. Indeed, physically the amplitude of a peak cannot become negative, negative amplitudes correspond to a 180^0 phase shift. If you do not impose explicit lower and upper bounds on the amplitude AMARES will automatically constrain the amplitude to be greater or equal to zero. If you want you can impose other lower and upper bounds by entering them in the edit boxes, which appear if you set the soft constraint radiobutton to yes. This soft constraints button automatically appears when you select the unconstrained option. By default 0 and a large positive value will be displayed in the edit boxes as lower and upper values. If you want to change this lower and upper bounds it is important here to note that you have to enter these values according to the *original intensities.* MRUI takes care of passing the right values to the Fortran programs. The same is true when you enter a fixed shift or if you fix the value of the amplitude.
When using Inputnv on PC, do not use values smaller than -10^{97}, values between -10^{-97} and 10^{-97}, or values larger than 10^{97}. This is because exponential numbers with more than three digits in the exponential cannot be written to a file in Microsoft Fortran.

14.4.1.3 constraints for dampings / linewidths

When you want to impose constraints on the dampings/linewidths you will always be presented with the following six options from the popup menu:

unconstrained
impose shift
impose ratio
impose variable shift
impose variable ratio
fix value

Since AMARES treats all the parameters identically there is a large similarity between imposing prior knowledge for the different parameters. Therefore, in order not to repeat the explanations we already did for the phases, we highlight here the special points you have to take into account when imposing constraints for dampings.
If you select the unconstrained option, the damping is a free parameter and you can get any *positive* value, depending on the fit procedure. Indeed, each peak is modeled as a decaying signal and due to the nature of the

model function used in AMARES, this means that the value you get out of the fitting procedure must be positive. If you do not impose explicit lower and upper bounds on the dampings, AMARES will automatically constrain the dampings to be greater than or equal to zero. If you want you can impose other lower and upper bounds by entering them in the edit boxes, which appear if you select yes from the soft constraint radiobutton. This button automatically appears when you select the unconstrained option. By default 0 and a large positive value will be displayed in the edit boxes as lower and upper values. If you want to change them it is important to note here that you have to enter these values according to the linewidths *in Hz*. The same is true when you enter a fixed shift or if you fix the value of the damping.
When using Inputnv on PC, do not use values smaller than -10^{97}, values between -10^{-97} and 10^{-97}, or values larger than 10^{97}. This is because exponential numbers with more than three digits in the exponential cannot be written to a file in Microsoft Fortran.

14.4.1.4 constraints for frequencies

When you want to impose constraints on the frequencies you will always be presented with the following six options from the popup menu:

unconstrained
impose shift
impose ratio
impose variable shift
impose variable ratio
fix value

Since AMARES treats all the parameters identically there is a large similarity between imposing prior knowledge for the different parameters. Therefore, in order not to repeat the explanations we already did for the phases, we highlight here the special points you have to take into account when imposing constraints for frequencies.
If you select the unconstrained option, the frequency is a free parameter and you can get any value within the spectral width, depending on the fit procedure. If you do not impose explicit lower and upper bounds on the dampings, AMARES will automatically constrain the frequencies to lie within the spectral width. If you want you can impose other lower and upper bounds by entering them in the edit boxes, which appear if you select yes from the soft constraint radiobutton. This button automatically appears when you select the unconstrained option. Values corresponding to the spectral width will be displayed in the edit boxes as lower and upper values. If you want to change these bounds it is important to note here that you have to enter these values according to the frequency unit as indicated by the x-axis unit displayed in the spectral plot. The same is true when you enter a fixed shift or if you fix the value of the frequency. *We advise you to temporarily set the x-axis unit to Hz and then enter the constraints in Hz.* In that way you will avoid making mistakes.
When using Inputnv on PC, do not use values smaller than -10^{97}, values between -10^{-97} and 10^{-97}, or values larger than 10^{97}. This is because exponential numbers with more than three digits in the exponential cannot be written to a file in Microsoft Fortran.

14.4.1.5 done pushbutton

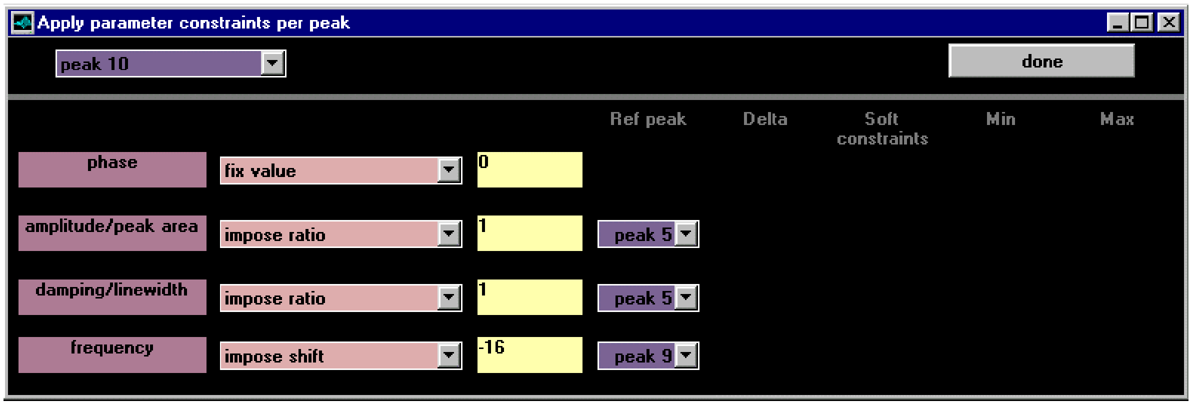
When you are finished applying the parameter constraints for all peaks, you should press the done pushbutton, which writes all the necessary information to the `inputnv.ans` and `link.ans` (see section 12.6) files, and closes the prior knowledge window.

To impose the prior knowledge for the `ischr.dat` file described in section 12.1, you have to provide the following information to the AMARES prior knowledge function:

peaks	parameters	constraints	reference peak
1,2,3,4,5	phase	fix value;0	
	amplitude / peak area	unconstrained	
	damping / linewidth	unconstrained	
	frequency	unconstrained	
6	phase	fix value;0	
	amplitude / peak area	impose ratio;1.09	5

	damping / linewidth	impose ratio;1	5
	frequency	impose shift;-16	5
7	phase	fix value;0	
	amplitude / peak area	unconstrained	
	damping / linewidth	impose ratio;1	5
	frequency	unconstrained	
8	phase	fix value;0	
	amplitude / peak area	impose ratio;1.18	7
	damping / linewidth	impose ratio;1	5
	frequency	impose shift;-16	7
9	phase	fix value;0	
	amplitude / peak area	impose ratio;1.29	11
	damping / linewidth	impose ratio;1	5
	frequency	unconstrained	
10	phase	fix value;0	
	amplitude / peak area	impose ratio;2.25	11
	damping / linewidth	impose ratio;1	5
	frequency	impose shift;-16	9
11	phase	fix value;0	
	amplitude / peak area	unconstrained	
	damping / linewidth	impose ratio;1	5
	frequency	impose shift;-32	9

The prior knowledge window with the parameter constraints filled in for peak 10 is displayed in Figure 53.



phase	fix value	0	
amplitude/peak area	impose ratio	1	peak 5
damping/linewidth	impose ratio	1	peak 5
frequency	impose shift	-16	peak 9

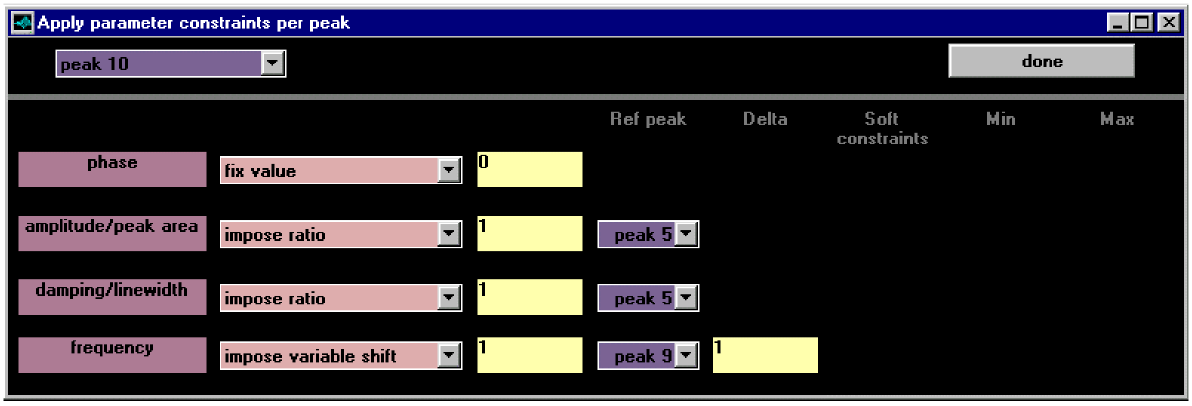
Figure 53. The prior knowledge window with the parameter constraints for peak 10. The amplitude/peak area and damping /linewidth parameters are taken equal to those of peak 5. For the frequency a fixed shift of -16 Hz is imposed with respect to peak 9.

When you would like to estimate the frequency splitting of the multiplet instead of imposing a fixed value of 16 Hz, you can do it in the following way:

peaks	parameters	constraints	reference peak	delta
1,2,3,4,5	phase	fix value;0		
	amplitude / peak area	unconstrained		
	damping / linewidth	unconstrained		
	frequency	unconstrained		
6	phase	fix value;0		
	amplitude / peak area	impose ratio;1.09	5	
	damping / linewidth	impose ratio;1	5	
	frequency	impose variable shift	5	1
7	phase	fix value;0		
	amplitude / peak area	unconstrained		
	damping / linewidth	impose ratio;1	5	

	frequency	unconstrained		
8	phase	fix value;0		
	amplitude / peak area	impose ratio;1.18	7	
	damping / linewidth	impose ratio;1	5	
	frequency	impose variable shift;1	7	1
9	phase	fix value;0		
	amplitude / peak area	impose ratio;1.29	11	
	damping / linewidth	impose ratio;1	5	
	frequency	unconstrained		
10	phase	fix value;0		
	amplitude / peak area	impose ratio;2.25	11	
	damping / linewidth	impose ratio;1	5	
	frequency	impose variable shift;1	9	1
11	phase	fix value;0		
	amplitude / peak area	unconstrained		
	damping / linewidth	impose ratio;1	5	
	frequency	impose variable shift;2	9	1

The prior knowledge window with the parameter constraints filled in for peak 10 is displayed in Figure 54.


Apply parameter constraints per peak

peak 10 | done

			Ref peak	Delta	Soft constraints	Min	Max
phase	fix value	0					
amplitude/peak area	impose ratio	1	peak 5				
damping/linewidth	impose ratio	1	peak 5				
frequency	impose variable shift	1	peak 9	1			

Figure 54. The prior knowledge window with the parameter constraints for peak 10. The amplitude/peak area and damping /linewidth parameters are taken equal to those of peak 5. A variable frequency shift is imposed with respect to peak 9.

14.4.2 done

Using the done menu item, you can finish Inputnv without having to go into the prior knowledge window. This may be useful in case you only wanted to make minor changes to the Inputnv parameters after reloading a data file on which you have done a complete prior knowledge session before, or if you do not wish to apply any further prior knowledge in terms of parameter constraints.

14.5 Files produced

Once you have finished using the Inputnv function, the following files will have been produced or updated:

```
inputnv.ans
<name>.ipk
```

The `inputnv.ans` and its file-specific copy `<name>.ipk` contain the anwers to be given to the Fortran program `inputnv`, such that MRUI can build the input file for AMARES (`input.par`) via:

```
% inputnv < inputnv.ans
```

15. AMARES parameter estimation

This chapter was written by:
Leentje Vanhamme
Katholieke Universiteit Leuven
Department of Electrical Engineering
ESAT / SISTA *tel* *+ 32 - 16 – 32 17 97*
Kardinaal Mercierlaan 94 *fax* *+ 32 – 16 – 32 19 86*
B – 3001 Leuven – Heverlee *e-mail* *Leentje.Vanhamme@esat.kuleuven.ac.be*

Once you have created a set of starting values for the nonlinear parameters (frequencies and damping factors/linewidths) in a `peakpick.par` and/or `<name>.pkp` via the Peakpick function (chapter 11), and supplied the necessary and optional prior knowledge settings in `inputnv.ans` and/or `<name>.inp` file via the Inputnv function (chapter 14), you can start the MRS fitting algorithm AMARES by selecting nl2sol (amares) as algorithm in the VARPRO/AMARES algorithm submenu. AMARES is based on solving a nonlinear least squares problem using NL2SOL[35]. AMARES reads starting values for the fitting procedure from the `<name>.pkp` or `peakpick.par` file, and the prior knowledge from the `<name>.ipk` or `inputnv.ans` via the Inputnv function. AMARES uses the same spectral window and same menu functions as VARPRO, and can therefore be used completely analogously to the VARPRO function described in chapter 13. We refer to chapter 13 for an explanation on the various menu functions. When reading, keep in mind that the `<name>.ipk` and `inputnv.ans` files of AMARES are the equivalents of the `<name>.inp` and `inputvp.ans`, respectively, files for VARPRO.
In this chapter we only discuss the different return messages of AMARES and the slight differences in output files.

15.1 AMARES return messages

After running AMARES an info message is displayed. If the message number is between three and six this indicates that AMARES converged, that is, that the program, following its own internal convergence criteria, claims to have found a local minimum. It is up to the user to decide if the answer is relevant for his/her application. On the other hand, a negative number is an indication of an error in the `input.par` file or non-convergence of AMARES (again following its own internal convergence criteria).
In summary, the most important info messages displayed by AMARES in the Matlab command window have the following meaning:

`info`	info description
3	X-convergence. The scaled relative difference between the current parameter vector X and a locally optimal parameter x^* is very likely small.
4	relative function-convergence. The relative difference between the current function value and its locally optimal value is very likely small.
5	both X and relative function-convergence, the condition for `info=3` and `info=4` hold.
6	absolute function convergence. The current function value (half the sum of squares) is very small.
-1	prior knowledge specified in `input.par` file is not valid
-7	singular convergence. The Hessian near the current X appears to be singular or nearly so.
-8	false convergence. The iterates appear to be converging to a noncritical point.
-9	function evaluation limit reached without complete convergence.
-10	iteration limit reached without complete convergence.

So if the `info` value is `3`, `4`, `5` or `6` there is normally nothing to worry about. If `info=-1`, AMARES also displays extra information in the Matlab command window, indicating the type of error that occurred. Normally using MRUI, this type of error should not occur since MRUI takes care of writing the `input.par` file. If you get an error of this type when using MRUI, this indicates that there is a bug in the MRUI Matlab code which generates the file `inputnv.ans` or in the Fortran program `inputnv` that generates the actual `input.par` file. When `info=-7`, the Hessian becomes singular, which means that the model is overdetermined, i.e. it contains too many parameters, at least near the current parameter estimate. It is possible that a different starting

value would lead AMARES to convergence in a strong way (`info=3`, `4`, `5` or `6`). However it is much more likely that your model has too many free parameters, so you have to change the prior knowledge you impose or change the number of spectral components you are fitting. This would typically occur if you tried to fit e.g. the zero order phase and the individual phases of all the peaks at the same time. In any case, you do not have to worry about the latter when using MRUI, since MRUI prevents you from doing this.
For a very detailed explanation on all these parameters we refer you to the report available at `http://netlib.bell-labs.com/cm/cs/cstr/153.ps.gz`.

15.2 Files produced

The files produced by the AMARES function are basically the same as those produced by the VARPRO function as described in section [13.7]. Only the files `<name>.gol, <name>.vrs` have some entries in addition to those produced by the VARPRO function as we will explain below.
If no echo signal is fitted we will get the following additional entries in both files:

```
echo signal fitted                        n
number of used points of backward part    0
  gval
0.0000
0.0000
```

The ‘`n`’ indicates that no echo signal is fitted, of course as a consequence the number of used points in the left (backward) part of the echo is zero. `gval` indicates the lineshape used for each peak. There are as many entries of `gval` as there are peaks, i.e. in this example we only have two peaks. When `gval` equals `0` (`0.0000` to be precise...) the lineshape used in the model function is a Lorentzian, if `gval` equals `1` the lineshape used is a Gaussian.
If an echo signal is fitted you will get the following entries in both files:

```
echo signal fitted                        y
number of used points of backward part   50
  gval     fdam    f2sd      bdam    b2sd
0.0000  -0.1505  0.0149   -0.1505  0.0149
0.0000  -0.1505  0.0149   -0.1505  0.0149
```

The ‘`y`’ indicates that an echo signal is fitted and ‘`50`’ indicates that there were 50 data points belonging to the left (backward) part of the echo. `gval` again indicates the lineshape used for each peak. The dampings of the right part of the echo are found in the column indicated by `fdam`, `f2sd` denotes two times the standard deviation on these values. The dampings of the left part of the echo are found in the column indicated by `bdam`, `b2sd` indicates two times the standard deviation on these values. Again there are as many rows in these columns as there are peaks. In `<name>.gol` the values of the dampings are expressed in ms, in `<name>.vrs` the values of the dampings are expressed in kHz.

16. The Plot facility

The MRUI software contains a few plot functions simply for visual interpretation of data and results that cannot be otherwise displayed using one of the previously described functions. These functions can be found under the Plot menu in the MRUI base window.

16.1 plot FID

With the plot FID function you can display the complex time signal as stored in the `<name>.dat/mat` file. Normally, in all other MRUI functions (see previous chapters), you automatically get to see the FT spectrum of it. You can save the time domain FID(s) to an ASCII file (see section 5.1.1.3), but then you would first have to load that ASCII file into some other software program to view the FID. Only in the FID processing window can you see the FIDs (see section 5.3), but you would not normally go into all that merely to check the time domain data points of the FID. In that case, use the plot FID menu item under the Plot menu. The special plot window will be displayed empty, and can be used as follows.

16.1.1 The File menu

The load menu (no submenu items!) works just like the load file function described in section 5.1.1.1.1, except that the file manager window now lists the `*.mat` files instead of the `*.dat` files (see for difference section 4.1). After selecting the data file, its time domain signal will be displayed in the window; the real part of the signal as a yellow solid line, and the imaginary part of the signal as a red dash-dot line. In Figure 55 the FID of `ischr.mat` is displayed.

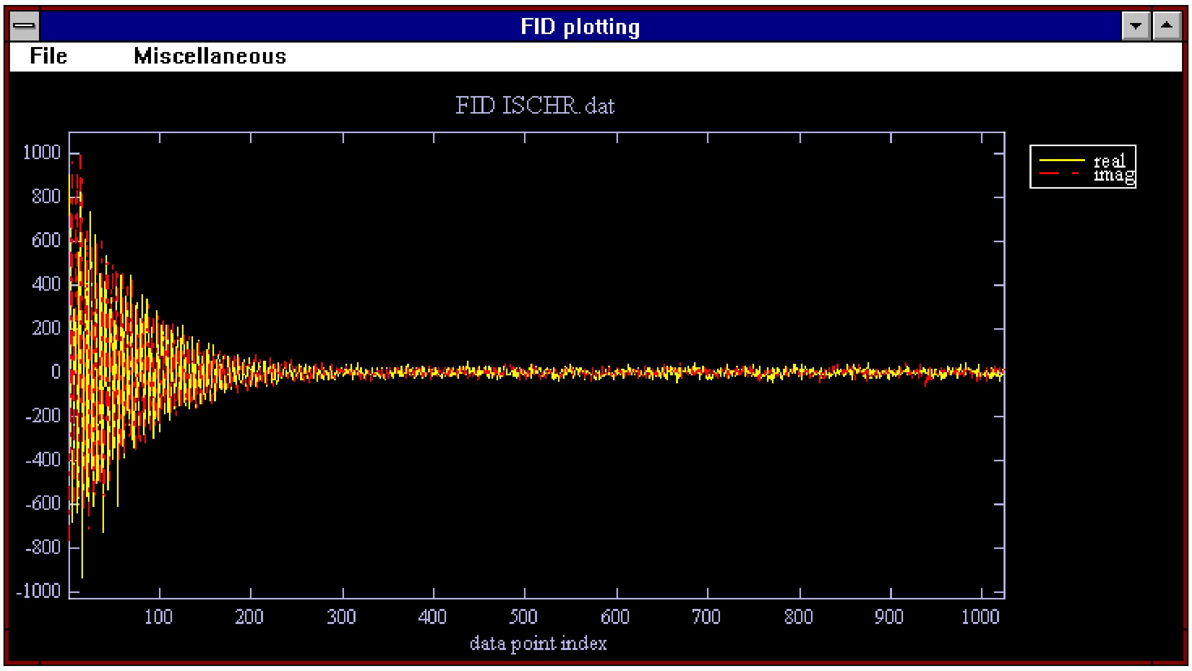


Figure 55. The plot FID window displaying the time domain signal of ischr.mat (solid line is real part, dash-dot line is imaginary part).

Once the FID has been plotted, you can zoom in and out and scale up and down, all using Matlab's free area zoom function (see section 5.1.2.1).

The print and quit menus work exactly as described in sections 5.1.1.2 and 5.1.1.4, respectively.

16.1.2 The Miscellaneous menu

16.1.2.1 axis units

With this menu item you can select whether the time axis should be displayed in units of points (index) or time. If you select points, as in Figure 55, the x-axis will display the data point index running from `1` to `ndp`. If you select time, the data point labels will display their true value in milliseconds, i.e. `begin+n*step`, with `n` running from `0` to `ndp-1` (see for more information on these parameters section 4.1.1). The default is to display the x-axis by the data point index.

16.1.2.2 real/imaginary

This menu item allows you to switch on/off the display of the real or imaginary part of the time domain signal. Selecting either the real or imaginary submenu item toggles its settings: if displayed the submenu item will be checked, otherwise unchecked. The plotted FID will be immediately updated. The default is to have both parts of the complex signal displayed; the real part always as a yellow solid line and the imaginary part always as a red dash-dot line.

16.2 batch spectra

This function plots an entire batch in one display. All spectra of the batch are plotted on top of each other as dotted lines. The first spectrum in the batch is then overlaid as a highlighted solid line. Using the vertical slider on the right, you can change the serial number of the signal that is displayed as a highlighted solid line. This way you can ‘walk’ through the batch, watching for peaks decreasing or increasing in intensity, or shifting frequency. This may be useful to ‘get to know’ your batch before you attempt automatic batch processing; see also section 11.1.1. When selecting the batch spectra menu item, the plot window will be displayed empty, and can be used as follows.

All spectra are displayed as their real part, phased according to the phasing values found in `<name><b>.gra` or `default.gra`. It also uses default zoom, line broadening and axis units settings read from those files. It may therefore be useful to make sure the first spectrum of the batch is properly displayed, e.g. using the SETUP: experimental function (see chapter 7 and section 5.1.4).

16.2.1 The File menu

The load menu (no submenu items!) works just like the load batch function described in section 5.1.1.1.2, again you have to change directory to the directory containing the batch first. After defining the batch, the FT spectrum of the first signal in the batch will be displayed as a dotted line (package default is dark blue, see section 16.2.3), and the show batch pushbutton appears. Figure 56 displays this situation for a batch of ^{31}P MR spectra collected from perfused livers[†].

[†] Data kindly provided by Nanna Gillis, Koen Bruynseels and Dr. Flor Vanstapel, Catholic University of Leuven (B), Biomedical NMR Unit.

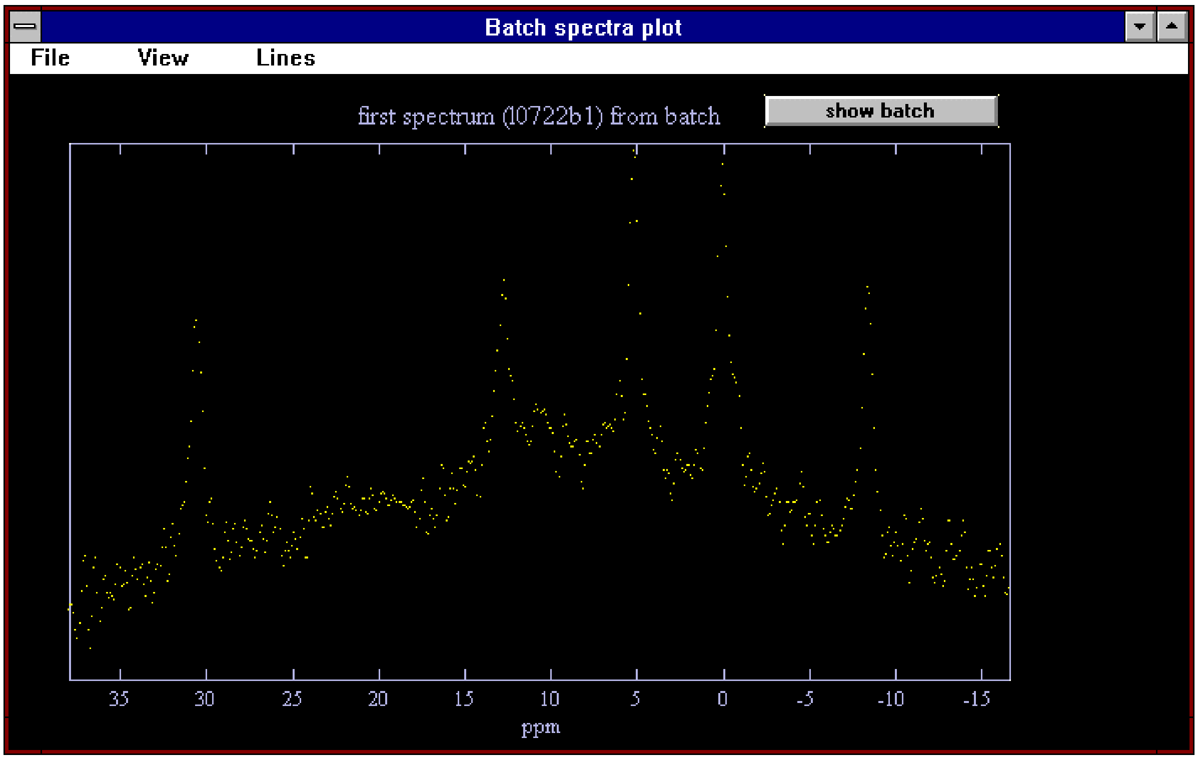


Figure 56. The plot window for the batch spectra function, displaying the first spectrum of the batch after having defined the series, plus the show batch pushbutton, which launches the display of all spectra on top of each other.

When you press the show batch pushbutton, all spectra of the batch will be displayed on top of each other, all as dark blue dotted lines, with the first spectrum of the batch overlaid as a highlighted (package default is cyan, see section 16.2.3) solid line. The show batch pushbutton then disappears and a vertical slider appears on the right, with two text boxes at its right hand side; one at the bottom displaying the start number of the batch, and one at the top displaying the end number of the batch. Next to the sliding part of the slider is an edit box displaying the current serial number in the batch of which file the spectrum is displayed as a highlighted solid line. Initially this will be start number of the batch. However, you can move the slider up and down to display different spectra in the batch. When you move the slider, the edit box on the right will be automatically updated to indicate the serial number of the highlighted spectrum. Use the arrow-ends of the slider to step through the batch one by one. You can also edit the number in the edit box on the right to immediately highlight the spectrum you are particularly interested in. One problem with this slider is that it is completely insensitive to resizing the plot batch spectra window. This is a shortcoming of Matlab: it was either this quirk, or the slider had to be horizontal. So make sure you install the right default size for this window using the customize windows menu (see section 3.2.1.1) first! Figure 57 displays the window with a spectrum more or less halfway through the batch highlighted.

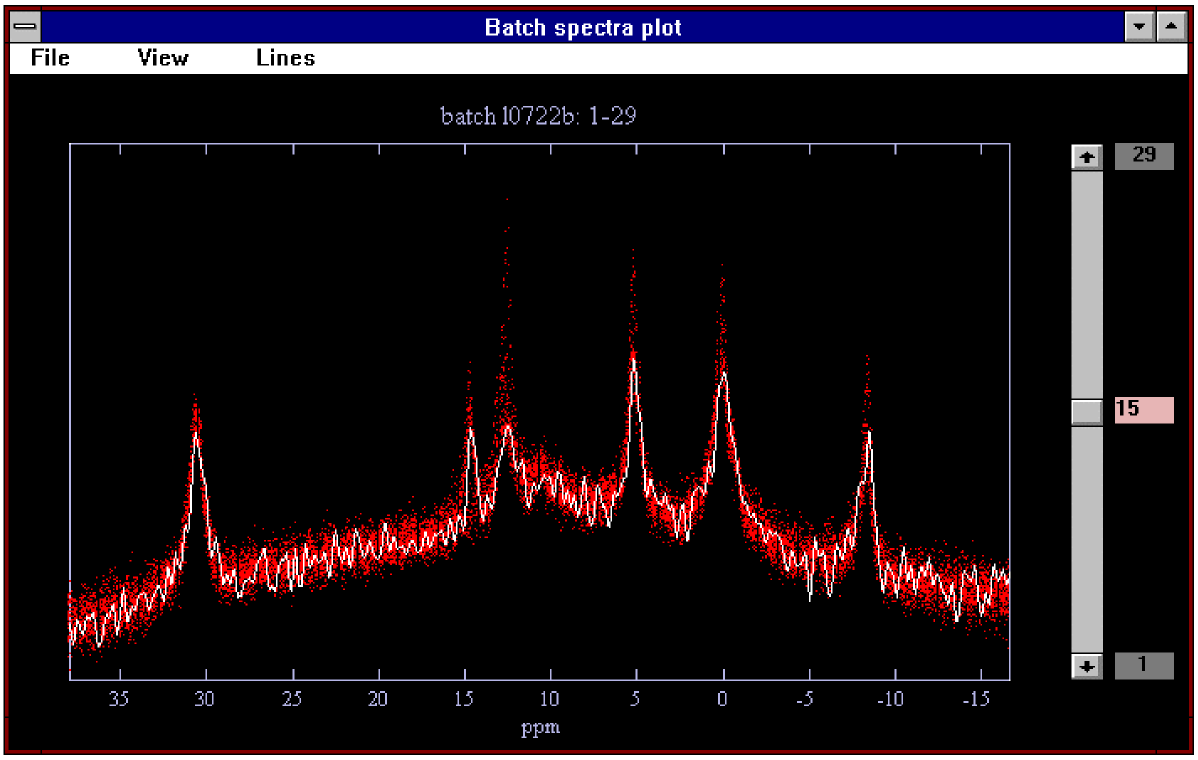


Figure 57. The plot window showing the entire batch of the perfused liver experiment. All spectra are plotted on top of each other (as blue dotted lines), which spectrum number 15 (see slider and edit box on the right) is highlighted. Notice how some peaks are clearly lower in the highlighted spectrum than in other spectra of the batch, whereas the peak at 15 ppm (sugar phosphates) was initially non-visible, and is steadily rising throughout the perfusion experiment.

The print and quit menus work exactly as described in sections 5.1.1.2 and 5.1.1.4, respectively.

16.2.2 The View menu

The View menu can be used to zoom in and out and scale up and down. The zoom in, zoom out, scale up and scale down functions works exactly as described in sections 5.1.2.1, 5.1.2.2, 5.1.2.4 and 5.1.2.5, respectively.

16.2.3 The Lines menu

With this menu you can change the color and the marker of the background spectra, and the color of the foreground spectrum. You can change their default settings using the Plot batch spectra menu in the customize preferences window (see also section 3.2.1.3).

16.3 batch results

It was explained in sections 10.11 and 13.7 that after automatic parameter estimation on a batch of files, separate ASCII files would be produced, each containing all estimates for all peaks in all signal of the batch for a certain parameter. One file for all amplitudes, one for all frequencies, etc. These parameters are stored in the ASCII files a columns; the first column listing the batch index, and then a column for each peak included in the fit, in the order as numbered in the spectral window or result table, listing the specific parameter for that peak as estimated for all signals in the batch. These ASCII spreadsheet files can be easily imported into your desired spreadsheet software, be it for applying statistics, plotting time graphs of specific parameters, etc. However, if you want a quick glance at the course of a certain type of parameter during your experiment of which you acquired the batch, you can load the ASCII spreadsheet files into the batch results function. When selecting the batch results menu item, the plot window will be displayed empty, and can be used as follows.

16.3.1 The File menu

The load menu contains a submenu item for each type of parameter for which an ASCII spreadsheet file might have been produced. Loading an ASCII spreadsheet file works just like the load file function described in section 5.1.1.1.1, except that the file manager window lists the spreadsheet files according to their extensions (see also section 13.7):

parameters in ASCII file to be loaded	ASCII batch result files listed
amplitudes	`*.amp`
frequencies	`*.fre`
linewidths	`*.lw`
phases	`*.pha`
zero order phase	`*.ph0`
begin time	`*.t0`
pH	`*.ph`
absolute concentrations	`*.cnc`
absolute intensities	`*.aam`
water amp/lw	`*.wal`

The file to be loaded should not start with a non-numerical character (otherwise the load command will fail, see also section 4.1.3), and all columns should contain the same number of entries, i.e. the number of peaks must have remained constant during the batch for unambiguous plotting of batch results.

For the current example we used the VARPRO batch result file for the absolute intensities `l0722b.aam` (i.e. the estimated amplitudes scaled back to the original intensities in the FID, using the `l0722b<b>.scl` files) of the batch displayed in Figure 57 in section 16.2. After having loaded the result file, graphs will be displayed for the estimated parameter each component, with a legend at the right of the figure. See Figure 58.

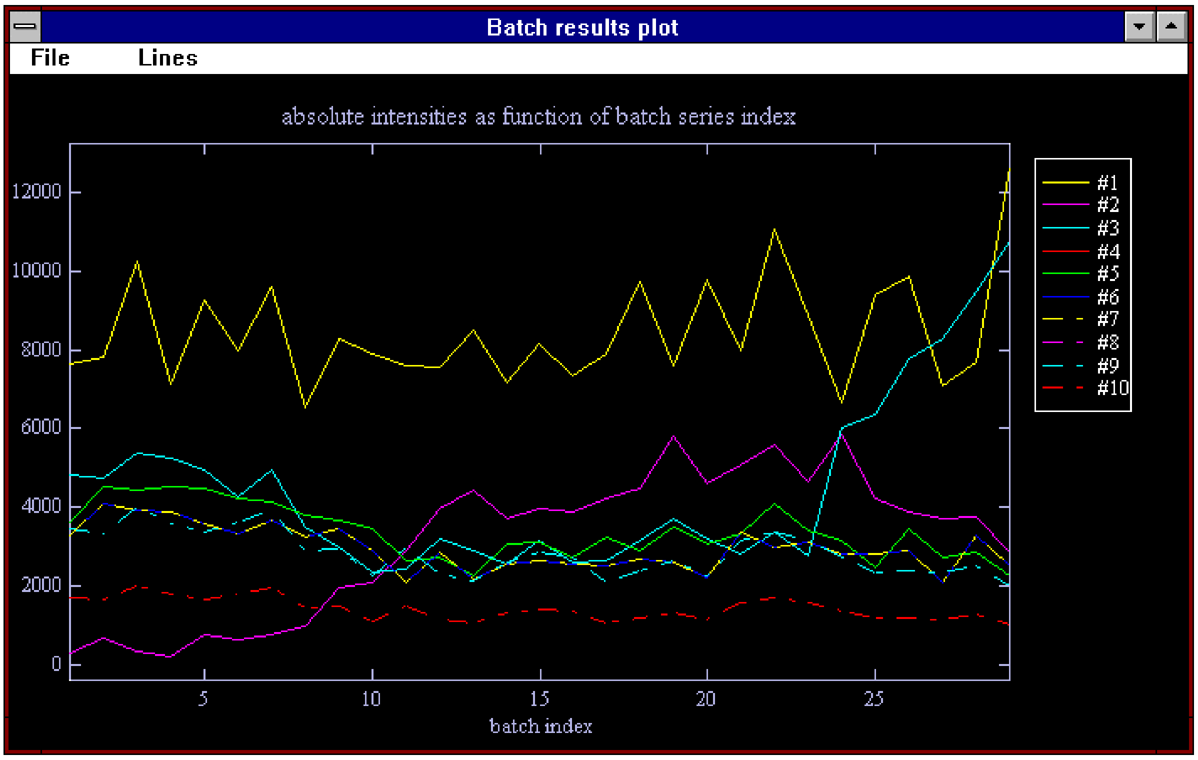


Figure 58. The plot window displaying the estimated original intensities of the 10 fitted components for the batch of the perfused liver experiment displayed in Figure 57. As indicated in the caption of that figure, the amplitude of PME (peak 2) can be seen to rise after its original amplitude near 0, with a slight decrease at the end. The P_i amplitude can be seen to decrease after spectrum 7, followed by a sharp increase after spectrum no. 23.

16.3.2 The Lines menu

With this menu you can change the overall appearance of the graphs. You cannot change their colors and markers using this menu, but only indicate whether the graphs should be displayed using solid lines or markers. This is the overall settings for all lines in the plot. By default, the individual colors and markers of each line in the plot (there are as many lines as there are peaks) are circulated from the basic Matlab set. The legend on the right will be updated according to your changing the graph type. You can change the default settings using the Plot batch results menu in the customize preferences window (see also section 3.2.1.3). The package defaults is to display the graphs as lines only.

16.3.3 Changing the properties of individual lines

You can double-click on a line to change its properties (visible on/off, color, marker). As soon as you double-click a line, the "Change line properties" window will appear, see
Figure 59 after double-clicking the line for peak 3.

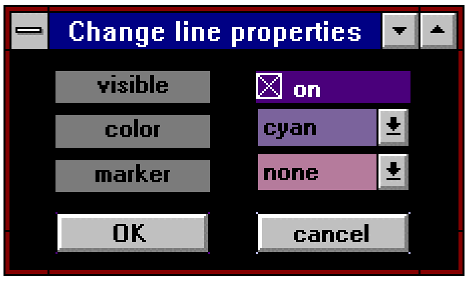


Figure 59. The Change line properties window which you can use to change the properties for the individual lines in the Plot batch results function.

The visible checkbox can be checked and unchecked, to set the visible property on or off. Be careful that once you switch the visible property of a line to off, the line really does become invisible, which means that you will have a tough time clicking it again (i.e. finding it) if you want to change the properties of the line once more. The color property of the line can be changed by selecting a new color from the popup menu, and the marker property of the line can be changed by selecting a marker (or 'none') from the popup menu. Selecting a marker will display the graph as markers only. Once you change any of the individual line properties, the display will immediately be updated (*not* the legend). When you are happy with the changes, press the OK button. After pressing the OK button, the display will be updated again, and at that time the legend will be updated too. Pressing the cancel button will undo the changes and return to the previous display before double-clicking the line.

16.4 contour plot

This function has not been completed yet and cannot be used. The menu item is disabled (inactive).

16.5 mesh plot

This function has not been completed yet and cannot be used. The menu item is disabled (inactive).

16.6 image

This function has not been completed yet and cannot be used. The menu item is disabled (inactive).

Appendix A
Financial support

From April 1st, 1995 to March 31st, 1997, the development of the MRUI software package, and its official distribution, was part of, and funded by, the **EC Human Capital & Mobility / Networks** program, project number **HCM-CHRX-CT94-0432**, *Advanced Signal Processing for Medical Magnetic Resonance Imaging and Spectroscopy*. This project, co-ordinated by Dr. Dirk van Ormondt of Delft University of Technology, was awarded to:

Dr. D. van Ormondt
Applied Physics Department
Delft University of Technology
P.O. Box 5046
2600 GA Delft
THE NETHERLANDS

Dr. D. Graveron-Demilly
Lab de RMN, CNRS UPRESA Q5012
Université Claude Bernard Lyon I, Bât. 308
43, Boulevard du 11 Novembre 1918
69622 Villeurbanne Cédex
FRANCE

Prof. S. Van Huffel
Dept. Electrical Engineering, ESAT / SISTA
Katholieke Universiteit Leuven
Kardinaal Mercierlaan 94
3001 Heverlee
BELGIUM

Prof. P. Van Hecke
Biomedical NMR Unit
Katholieke Universiteit Leuven
Gasthuisberg
3000 Leuven
BELGIUM

Prof. Dr. D. Michel
Fakultät für Physik und Geowissenschaften
Universität Leipzig
Linnestrasse 5
04103 Leipzig
GERMANY

Prof. B.G. Mertzios
Automatic Control Systems Laboratory
Dept. Electrical and Computer Engineering
Democritus University of Thrace
67100 Xanthi
GREECE

Prof. J-P. Antoine
Unité de Physique Théorique et Physique
Dept. Physique - Faculté des Sciences
Université Catholique de Louvain
1348 Louvain-La-Neuve
BELGIUM

Prof. G. Carayannis
Computer Science Division
Dept. Electrical Engineering
National Technical University of Athens
9 Iroon Polytechniou St.
15773 Zographou
GREECE

Prof. J.M. Dereppe
Chimie Physique Moleculaire et Cristallographie
Dept. Chimie - Faculté des Sciences
Bât. Lavoisier, Place Louis Pasteur 1, Bte 3a
Université Catholique de Louvain
1348 Louvain-La-Neuve
BELGIUM

For more information, please visit the HCM *Advanced Signal Processing for Medical Magnetic Resonance Imaging and Spectroscopy* WWW home pages at `http://azur.univ-lyon1.fr/HCM/hcm.html`.

From November 1st, 1993, to March 31st, 1995, the development of the MRUI software was part of the author's Ph.D. project at St. George's Hospital Medical School, under the supervision of Prof. John Griffiths. The distribution of the software was co-ordinated in close collaboration with Dr. Ron de Beer of Delft University of Technology. The author's Ph.D. project "*The Use of Signal Processing Algorithms to Obtain Biochemically Relevant Parameters from Magnetic Resonance Data Sets*" was part of the biomedical MR research of Prof. Griffiths' Biomedical MR Research Group, fully funded by the Cancer Research Campaign, UK, grant number SP1971/0402.

Prof. J.R. Griffiths
CRC Biomedical MR Research Group
Division of Biochemistry
St. George’s Hospital Medical School
Cranmer Terrace
London SW17 0RE
UNITED KINGDOM

Dr. R. de Beer
Applied Physics Department
Delft University of Technology
P.O. Box 5046
2600 GA Delft
THE NETHERLANDS

During the same period, the development of the MRUI software further benefited from a fruitful collaboration between Prof. Griffiths’ group and that of Dr. Carles Arús of the Autonomous University of Barcelona, funded by the British Council and the Spanish Ministry of Education & Science, joint project HB 94-258 *Pattern Recognition and Neural Network Processing of in vivo and in vitro NMR Spectra of Human Brain Tumors for Diagnostic Applications.* This project was awarded to:

Prof. J.R. Griffiths
CRC Biomedical MR Research Group
Division of Biochemistry
St. George’s Hospital Medical School
Cranmer Terrace
London SW17 0RE
UNITED KINGDOM

Dr. Carles Arús
Dept. Bioquímica I de Biologia Molecular
Unitat de Bioquímica de Ciències
Universitat Autònoma de Barcelona
Edifici C
08193 Bellaterra (Barcelona)
SPAIN

Appendix B
Bug fixes in MRUI_96.3

The following bugs were fixed in MRUI version 96.3.

Contributions by :	Mika Ala-Korpela	University of Kuopio	Finland
	Koen Bruynseels	Catholic University Leuven	Belgium
	Miquel Cabañas	Universitat Autònoma Barcelona	Spain
	Blaise Frederick	McLean Hospital Belmont	USA
	Nanna Gillis	Catholic University Leuven	Belgium
	Yvonne Heidkamp	University Hospital Nijmegen	The Netherlands
	Thomas Hoess	University of Tübingen	Germany
	Reino Laatikainen	University of Kuopio	Finland
	Ross Maxwell	Skejby University Hospital	Denmark
	Dominick McIntyre	St. George's Hospital London	England
	Jim Murdoch	Picker International Ohio	USA
	Richard Nolde	Yale MR Center	USA
	Takashi Ogino	Institute of Neuroscience Tokyo	Japan
	Alessandro Ponti	University of Milan	Italy
	Manoj Saranathan	University of Washington	USA
	Jeff Stanley	University of Pittsburgh	USA
	Leentje Vanhamme	Catholic University Leuven	Belgium
	Paul Van Hecke	Catholic University Leuven	Belgium

By the way, sometimes when you get an error message in the Matlab command window, you can track down which scripts the function went until it crashed. When you report an error, make a note of these script or function names, and also the order in which they were called. Before you report the error, you can get some more information on the scripts that were involved, by using the Matlab `help` command on the name of the MRUI script, each of which contains an informative header about its function, input and output parameters, last update, etc. This header information can be displayed in the Matlab command window by typing:

```
>> help <scriptname>
```

where `<scriptname>` is the name of the Matlab or MRUI script or function, with or without its `.m` extension.

MRUI base window

1. In Plot batch spectra, spectra in the batch could be plotted with the wrong phase if the begin time in the `.mat` data file differed from the begin time as set by the tbegin slider.
2. Loading a peakpick file through the Database menu would crash if the cancel button were used in the file manager window.

Customization

1. Customizing the prior knowledge window no longer worked.
2. Clicking on the empty bar in customize windows would cause an error message.
3. Changing the customize settings for pH calculation under the VARPRO menu had no effect in further VARPRO runs.
4. Customize preferences seemed defect from the fifth line downwards.

Spectral windows general

1. Autoscale no longer worked.
2. Using Next after a batch caused a crash, which is normal because the Next menus should have been disabled after a batch of files. They now are.
3. When loading a batch series, the use of the cancel button in the batch definition window could sometimes generate an error message.
4. Saving spectra in an ascii plot file would crash.

5. When running SVD_1D or VARPRO on a spectrum that had not had any zoom in (i.e. displayed over entire spectral width), an error could occur before MRUI would plot the result window with reconstructed spectrum etc.
6. Changing the number of points in the FT (`ndft`, edit box in lower left corner) in the result window, with the spectral labels off, would bring back the spectral labels on the screen.
7. Displaying only one type of spectrum in the results window (e.g. only the residual) resulted in an error message.
8. When switching x-axis labels off, the x-axis unit (e.g. ppm) would still be displayed.
9. If you had the spectrum labels switched on, they would not re-appear after switching the result info list off and back on.
10. Under certain circumstances, the following error could lead to a crash:
    ```
    >> ??? Error using ==> set
    Invalid object handle.
    Error in ==> /home/boogaart/MRUI_96.3/mrui/graphics/new_axis.m
    On line 93  ==>     eval(['set(',uic,'a,''checked'',''off'')'])
    ```
11. Apart from that one in 10, other "Invalid object handle" crashes could occur for objects (text or buttons etc. in a window) that were cleared when closing a window, and apparently not properly re-initiated by Matlab upon opening a new window. Such objects are no longer cleared when you close a window, so that they keep on getting overwritten, but Matlab won't complain about not knowing what to do with them.

Plot

1. Using the Plot batch spectra function would crash if no `<name>.gra` or `default.gra` file could be found.

Tools

1. For non-Matlab users: `reconvp` did not use the correct tbegin value for first order phasing of the reconstructed and residual spectrum if the begin time had been fixed to, or estimated at, a different value from the one in the data file. The `tbegin` value in the data file was used for `reconvp` phasing. Now the tbegin value from the `golscr.par` file is used for phasing.

Simulate

1. The percentage noise slider could display a negative value in the edit box on the left, depending on the phases of the signal components.
2. Some problems to do with scaling factors and intensities of simulated signals were fixed.
3. Loading a new simulation file into the Simulate function on PC would not update the simulation parameters and simulated signal.
4. Loading a new simulation file into the Simulate function would not update the Parameters menu.
5. Simulate could crash if you changed the number of data points, if the new number exceeded the current number of FT spectral points.
6. Saving a file from Simulate would give some unwanted side effects if the entered file name would be different from the file name listed in the `simul.par` file.

Conversion

1. When multiple windows with spectral plots would be displayed by a conversion routine (such as for GE data files were a metabolite, water and divided signal would be displayed in three different windows, or when displaying all conversions in a batch), the quit button of only the last displayed window would work. The other windows would remain visible, and could only be closed using the Xwindow feature of your Desktop software. This bug has been fixed, and any quit button on any of the displayed windows will now quit all conversion display windows at the same time.
2. Pressing cancel in the batch series definition window for batch conversion would still bring up the window in which to define the name of the file to be saved, and the batch window would not be cleared.
3. In GE data file conversion, when saving reference corrected metabolite FIDs (the '`z`' files, after point-wise division of the water-suppressed metabolite '`m`' signal by the unsuppressed water '`w`' signal), the scale factors in the `<name>.scl` file would be wrong (versions 96.1 and later), leading to ridiculously large numbers when attempting absolute quantitation.
4. Conversion of GE G files failed with new system-independent `strd` program.
5. Batch conversion of Bruker X32 (fid & acqus) files assumed serial numbers to be integrated in the full path names at the end, e.g.: `??/L960118/1/fid<b>`, where `<b>` is the serial number in the batch. This

was not very realistic, as the directory that contains the `fid` and `acqus` file has the name of the serial number. So now batch conversion of Bruker X32 (fid & acqus) files assumes the path name to be: `??/L960118/<b>/fid`.
6. For Bruker X32 files in `*.fid` format, the transmitter frequency was read in MHz and not converted to Hz (factor 10^6 wrong).
7. Philips conversion failed to set the proper ppm unit for other nuclei than protons.
8. Philips conversion failed to convert old Philips data files, in wich the parameter file `nr_time_series_spectra = 0`.
9. SMIS conversion crashed on a PC due to different `dec2hex` and `uiputfile` function output in PC-Matlab.
10. Varian NMR conversion failed for very long pathnames or in some cases failed altogether.
11. Conversion of old format Felix files in batch did not work.
12. MRUI mat_1d to dat_1d conversion failed on PC.
13. MRUI mat_1d to dat_1d conversion for PC contained a bug which inserted four zeroes at the start of the imaginary data points in the `.dat` file if the number of data points was an integer multiple of 4. This would lead to weird phases and useless fitting results.
14. MRUI mat_1d to dat_1d and vice versa did not work properly on a DEC RISC. Work-around provided.
15. MRUI dat_1d to asc_1d on PC produced a mat_1d file instead of an asc_1d file.
16. MRUI mat_1d to dat_1d on PC did not work in batch (only the first file of the batch converted).

Process

1. In the FID processing window, the result spectrum was never line broadened.
2. Some programs (such as divide) suffered from the routine reading the `*.scl` files not having a default norm in case no `*.scl` file was found; the empty norm variable would cause empty data vectors.
3. Running ER_Filter would display the wrong names over the displayed result spectra (both for individual files and for batches), though the produced files did have the correct names.
4. When using ER_Filter on a signal for which the phzero was different from 0, the resulting spectrum `<name>_x` got phased twice in the result display (although the actual signal in the `<name>_x` file was OK).
5. ER Filter did not fully take into consideration the phase and begin time of a signal, and the values for phase and begin time set by the sliders in the spectral window.
6. Normal/reverse menu item did not work in FID Processing.
7. The FID processing window was not sufficiently cleared (previous result spectra remained) when loading a new file into Cadzow EP, DC Offset correction or Multiply.

SVD_1D

1. After display of a singular values plot, the save ascii plot - spectra function would produce a singular value plot in the ascii plot file instead of a spectrum plot, or it would crash.
2. In the result spectral display, switching the result info list off when the singular value plot was displayed, would cause a crash.

Peakpick

1. Trying to change the axis units in Peakpick would cause an error message from `new_axis.m`, although the new axis units were applied, but the menu item wouldn't be checked.
2. Running Peakpick in batch would cause a crash after the first spectrum, a problem with `delete(pkph)`.
3. The ndp function in Peakpick, which allowed setting of the number of data points for the VARPRO fit, was not always successful in that when calling the VARPRO function this new number was not recognized. For this reason the ndp function has been taken out of the Peakpick routine, as it is also more relevant to change it just before running VARPRO.
4. When peakpicking in batches, you would have to press the OK button in the popup message each time a file in the series was not found. Now a message is displayed in the Matlab command window instead, to allow smoother continuation of the batch.

Inputvp

1. When changing the constraints for phzero and/or tbegin in an existing Inputvp set, the response of the group phase constraints in the subsequent Inputvp window was not ideal, and in some cases erroneous constraint values could be written to the prior knowledge (e.g. always phase 0 relative to phzero, regardless of what you had just typed).

2. In addition to the previous bug, under comparable circumstances, the relative phase constraints could be listed OK, i.e. with the previous constraint values in the edit boxes, but the value of 0 written to the `inputvp.ans` file.
3. Having a loaded a signal with `npk` peaks in Inputvp, you could see, after loading in the same Inputvp function a new file for which a Peakpick file existed with less peaks, that the redundant extra entries in the Group Peaks menu were still visible.

<u>VARPRO</u>

1. The pH menu no longer worked, gave an error message.
2. pH calculation in batches using the αNTP frequency would only work for the first spectrum.
3. VARPRO analysis did not take into account any default fitting values set by customize fitting parameters.
4. pH calculation for APP could give unreasonable `"out of range"` errors because the `c1` and `c2` constants could be tested in the wrong order.

Appendix C
New features in MRUI_96.3

The following new features were added in MRUI version 96.3.

Contributions by :	Mika Ala-Korpela	University of Kuopio	Finland
	Koen Bruynseels	Catholic University Leuven	Belgium
	Miquel Cabañas	Universitat Autònoma Barcelona	Spain
	Blaise Frederick	McLean Hospital Belmont	USA
	Nanna Gillis	Catholic University Leuven	Belgium
	Yvonne Heidkamp	University Hospital Nijmegen	The Netherlands
	Thomas Hoess	University of Tübingen	Germany
	Reino Laatikainen	University of Kuopio	Finland
	Ross Maxwell	Skejby University Hospital	Denmark
	Dominick McIntyre	St. George's Hospital London	England
	Jim Murdoch	Picker International Ohio	USA
	Richard Nolde	Yale MR Center	USA
	Takashi Ogino	Institute of Neuroscience Tokyo	Japan
	Alessandro Ponti	University of Milan	Italy
	Manoj Saranathan	University of Washington	USA
	Jeff Stanley	University of Pittsburgh	USA
	Leentje Vanhamme	Catholic University Leuven	Belgium
	Paul Van Hecke	Catholic University Leuven	Belgium

Installation

1. Automatic script that prepares user environment for running MRUI (environment variables set in `.cshrc` or `.profile`; creation of `$HOME/matlab/work_dir` with necessary files).
2. The MRUI installation on PC has been successfully completed, both for Windows 3.11 and Windows 95. Instructions can be found in `mrui_pc.txt`. The binary executable programs for the Fortran methods can be distributed so that a Fortran and C compiler are not required for the PC. As a compromise towards running under Microsoft, HLSVD and HSVD cannot use the more robust LAPACK routines for the linear least squares step, and we had to put the original routines back in. For the same reason, HTLS won't run under DOS due to LAPACK problems, and (EP)LPSVD(CR) will run, but the conversion to and from the Felix format (necessary for the LP methods) does not work as it requires the linking of Fortran code with C code, in which we have not succeeded using Microsoft compilers.
3. The Matlab-version (`MRUI_96.3`) and the non-Matlab version (`MRUI_FC`) of MRUI have been combined again. There is now one single package that should serve all MRUI users, those with Matlab, those without, those on UNIX systems and those on PCs.

MRUI base window

1. Quit MRUI and Quit Matlab have an OK/cancel interface.
2. Through customize fitting parameters you now have the possibility to select whether the estimated amplitudes (SVD and VARPRO fit results), as they appear in the table on the result screen, are listed as scaled amplitudes according to the FID normalization, or unscaled according to the original FID intensities. The latter might be preferable, but often the original FID intensities are so large that it will deface the entire window. The amplitude values according to the original FID intensities can always be found in the `<name>.hns` or `<name>.vns` file.
3. The Database menu now also contains a function for the inputnv file resulting from the Inputnv AMARES function.

Customize

1. Customize preferences now includes the default settings for the spectral result display. I.e. you can indicate whether you would always want the individual components to be displayed, etc. You can also change the default to which spectrum the display should be scaled, or the plotting order (bottom to top or top to bottom). These options you can set separately for Filter_HSVD, SVD_1D and VARPRO. You can also set

whether by default the spectrum labels (original, reconstructed, individual components, residual) should be displayed or not. Especially PC users may wish to disable these, as they tend to run through the table for small screen monitors. This setting is in the customize preferences menu at the position where previously the plotting order was controlled.
2. Customize fitting parameters has been protected for the absolute maximum number of data points, currently 16384, which can be handled by the Fortran programs. See also new feature 5 under Conversion.

Spectral windows general

1. PC users have an extra print option, print to clipboard. The window graphics contents are sent to clipboard in the Windows Meta Format (WMF), after which you can paste them into any Windows application.
2. The save ascii plot and save gnuplot functions have been merged under the label save ascii plot. The plot files have the same format. Should you want to make the plot file easily recognizable for your gnuplot software, just save the ascii plot file with a `.gnu` extension.
3. In the result window (after VARPRO analysis), the peak numbers were previously only displayed if the reconstructed spectrum was displayed. Now, if the reconstructed spectrum is not displayed, they will be displayed over the individual components if they are displayed, or else over the original spectrum, or if only the residual is displayed, over the residual spectrum.
4. The peak numbers menu item moved from the Miscellaneous menu to the Results menu.

Conversion

1. Picker spectrometer data file conversion.
2. GE Omega data file conversion (Manoj Saranathan) and conversion for 5.x stack (time series measurements) files.
3. Conversion of GE 5x files can now be performed on any machine, thanks to the `strd` emulation program, written by Blaise Frederick.
4. The scale files (`*.scl`) have been greatly simplified to aid clearer absolute quantitation. This upgrade requires conversion of old files when they are to be re-used in MRUI_96.3. You can do this in one go by using the `mruio2n` command. Change directory to where the old files are that you want to re-use, and simply type `mruio2n` in the an operating system shell, or `!mruio2n` in the Matlab command window. This lists some information on the use of the program. `mruio2n all` updates all files in the current directory, `mruio2n <name>` all files that belong to the data file with general file name base `<name>`. `mruio2n help` gives more detailed help. You can also call the `mruio2n` command from the Conversion menu under the MRUI base window; it is the OLD -> NEW menu item under the MRUI menu item. The `mruio2n` command will update *all* types of file automatically, whether binary data files, peakpick files, SVD_1D/VARPRO result files, or scale files. It will automatically decide whether a file is up to date or whether it needs to be converted. A detailed report is written to a file called `mruio2n.history` in your home directory. The `mruio2n` command only works on UNIX machines (but PC users have no old version to update from...).
5. All conversion routines now produce `<name>.dat/mat` files with a maximum of 16384 complex data points. Files with more data points would overflow the Fortran read routines, leading to aberrant results. You can change the maximum in customize fitting parameters (SETUP menu maxndp). However, you should only do this after having increased the maximum array size in the corresponding Fortran routines and recompiled.

SETUP

1. The number of data points for the fit and for the Fourier transform spectrum have been taken out of the SETUP menu as they are rather irrelevant here. It is still present in the customize fitting parameters function for SETUP, so that you can arrange a default value that suits you best.

FID Maths/Processing

1. FID truncation. The truncated FID is stored in `<name>_t.dat/mat`.
2. FID phasing. In a similar spectral display as e.g. Peakpick, you can phase the spectrum using the phzero and tbegin sliders. When the spectrum is properly phased, use the save phased FID menu item to multiply the FID by the exponential zero order phase factor, and change its delay time to the tbegin read from the slider. The phased FID is stored in a file `<name>_f.dat/mat`. This may be useful when you wish to sum or subtract two FIDs that have different phases.
3. FID normalization (uses same functions as multiplication). You can normalize any FID to a given maximum value. The normalized FID is stored in `<name>_n.dat/mat`.

4. Frequency shifts. You can either enter a constant frequency offset by which all frequencies in the spectrum should be shifted, or click on a peak and then enter the frequency it should have. This is all in intrinsic frequencies in Hz (i.e. different from the ppm referencing as in SETUP: experimental). The time domain data points are corrected for the frequency shift as well, and stored in a file `<name>_v.dat/mat`. You can select whether the kHz frequency reference for ppm units should be shifted as well (default is yes; can be permanently changed via customize fitting parameters).
5. FID Apodization works just like line broadening, but for this processing function the FID is also multiplied, and the apodized FID stored in `<name>_a.dat/mat`. You can choose between Lorentzian and Gaussian apodization, and you can also enter negative apodization constants ('line narrowing').
6. ER_Filter no longer takes the x-axis limits from the `*.gra` file, but displays the entire spectral width before you decide on which region to 'cut out'. This is a more relevant approach.
7. All functions now include the production of a corresponding scale file (`<name>.scl`), for the purpose of absolute quantitation.
8. FID division is now fully integrated in the FID processing window. The divide function divides a signal by the complex phase factor (so the signal divide by its own magnitude) of another signal. You can use this to divide water suppressed metabolite FIDs by the unsuppressed water FID to correct for eddy currents and magnetic field inhomogeneities (Uwe Klose's method, also known as QUALITY) if this correction is not automatically carried out during file storage (Siemens) or conversion in MRUI (GE, Picker). You can run divide in a batch, either dividing a series of files by the same FID, or dividing each file in a series by a different file (i.e. if you have two batches of the same size, e.g. 12 metabolite FIDs and 12 water FIDs from different measurements).
9. FID subtraction is now fully integrated in the FID processing window. The subtract function subtracts a signal from another signal. You can run subtract in a batch, either subtracting one particular signal from a series of files, or subtracting a different file from each file in a series (i.e. if you have two batches of the same size).
10. FID summation is now fully integrated in the FID processing window. The sum function sums a signal to another signal. You can run sum in a batch, either summing one particular signal to a series of files, or summing a different file to each file in a series (i.e. if you have two batches of the same size). Alternatively, you can sum all files in the loaded batch together.

Simulate

1. Simulate is protected against choosing a number of data points for the signal larger than 16384. Files with more data points would overflow the Fortran read routines, leading to aberrant results. You can change the maximum in customize fitting parameters (SETUP menu maxndp). However, you should only do this after having increased the maximum array size in the corresponding Fortran routines and recompiled.

Filter_HSVD

1. The Filter_HSVD result signal (the reconstructed time domain signal) can be saved to file as ordinary data file, for use in later analysis if desired. Select the save signals menu item under the Results menu.
2. The default number of data points for the Filter_HSVD method has been limited to the maximum number of data points for the SVD algorithm used, already in the MRUI menu.

SVD_1D

1. The result file `<name>.hns` now also contains a value for the signal-to-noise ratio (SNR) of the time domain signal in dB, calculated from the estimated amplitudes and damping factors, and the standard deviation of the noise, via the 10-log of (energy in signal over energy in noise).
2. Through customize fitting parameters you now have the possibility to select whether the estimated amplitudes (SVD_1D fit results), as they appear in the table on the result screen, are listed as scaled amplitudes according to the FID normalization, or unscaled according to the original FID intensities. The latter might be preferable, but often the original FID intensities are so large that it will deface the entire window. The amplitude values according to the original FID intensities can always be found in the `<name>.hns` file.
3. The SVD_1D result window also displays peak numbers, just like VARPRO.
4. The SVD_1D results signals (the reconstructed and residual time domain signal) can be saved to file as ordinary data file, for use in later analysis if desired. Select the save signals menu item under the Results menu.
5. In the pH calculation, you can now indicate whether αNTP was fitted as a doublet, and MRUI will take the average of the two frequencies for the αNTP chemical shift in the pH calculation.

6. The batch result files produced after running a batch, each containing a particular parameter for all peaks in all signals of the batch, are now accompanied by a file of identical format, listing the Cramér Rao estimates for the standard deviations of the estimated parameter(s). The file extensions are those of the parameter file, with an `s` as third character.

Peakpick

1. When running a batch in Peakpick, you could previously only use the phasing sliders and edit boxes on the first spectrum, where you had to set everything correctly, and then when the batch started, each occurring spectrum only allowed direct clicking on peaks for starting values (except for the zoom feature added in MRUI_96.2). It was not possible to re-phase a spectrum e.g. halfway through a batch. This would be useful for instance when processing a (non-field-corrected) set of MRSI voxels in a batch series. You can now use the batch functions menu item from the Miscellaneous menu in Peakick, *before* you start the batch Peakpick, where you can indicate which of the four phasing features you want to enable for each spectrum as it comes along in the batch (the more features you enable, the slower the program will respond to your mouse clicks, as the possibility of doing other things during a batch involves pausing the program and checking those sliders and edit boxes continuously). Then when you start the batch Peakpick, you can re-phase each spectrum first, and then you have to click on the done button for the real peak-picking to start on that spectrum. This process will be repeated for all spectra in the batch.

Inputnv

1. A completely new Inputnv function has been added, to supply prior knowledge for the AMARES algorithm (see new feature 1 in VARPRO). It works more or less like the Inputvp function for VARPRO, except that the parameter relations are implemented on a peak-to-peak basis, rather than for whole groups at the same time.

VARPRO

1. Thanks to long and hard work by Leentje Vanhamme, as part of her Ph.D. project, we are proud to present you with a completely new VARPRO-type nonlinear least squares algorithm, which we dubbed AMARES (Advanced Method for Accurate, Robust and Efficient Spectral fitting; pronounce it to get the gist of the acronym), and which is listed in the MRUI VARPRO function as NL2SOL (which is the core optimization algorithm implemented). It has the following advantages over the existing algorithms, but because it involves an entirely new prior knowledge interface, we left the existing methods in the interface for those of you who are happy with the current programs, and who do not need the new possibilities.
2. The result file `<name>.vns` now also contains a value for the signal-to-noise ratio (SNR) of the time domain signal in dB, calculated from the estimated amplitudes and damping factors, and the standard deviation of the noise, via the 10-log of (energy in signal over energy in noise).
3. When running a batch in VARPRO, you can now set the option that after each `<i>`-th run, the results of file no. `<i>` (read from the `<name><i>.gol` file) are used as starting values for file no. `<I+1>` (written into the `<name><i+1>.pkp` file). You activate this option using the menu item results run i => starting values run I+1 from the batch functions sub-menu under the Miscellaneous menu. Conversion of results to starting values is always done using three checks:
    - the number of iterations must be positive (negative number of iterations indicates VARPRO error)
    - the number of peaks found for file `<i>` must be identical to the number of peaks in the starting value file for file `<i+1>`, if the latter exists
    - all damping factors must be negative (positive linewidths)
4. A forth test is optional, which is the number of iterations. This can be checked against a maximum set by yourself (i.e. if more than an expected `<ierr>` iterations were necessary, then the solution might not be ideal, so don't use results for starting values of next run). You activate this option using the menu item max no. iterations under the batch functions menu. The number listed in this item, used for the check, can be set (semi-permanently) using the customize fitting parameters menu.
5. If a pH value would be undefined due to `"chemical shift out of range"` in a batch, a message would appear and wait until OK was clicked, halting the batch. Now, in batches, a message saying a similar thing, with batch file number, appears in the Matlab command window and the batch keeps on running.
6. Through customize fitting parameters you now have the possibility to select whether the estimated amplitudes (VARPRO fit results), as they appear in the table on the result screen, are listed as scaled amplitudes according to the FID normalization, or unscaled according to the original FID intensities. The latter might be preferable, but often the original FID intensities are so large that it will deface the entire window.

The amplitude values according to the original FID intensities can always be found in the `<name>.vns` file.

7. For users of the `reconvp` program (for GNU plots or after running VARPRO without Matlab), it can now also incorporate your Gaussian decay into the reconstruction, plus a line broadening factor you enter as a command line argument.
8. When running a batch in VARPRO, you can now set the option whether existing Peakpick (`<name>.pkp`) and Inputvp (`<name>.inp`) should be used when found or not. If you want to re-do an analysis of an entire batch after having re-done the Peakpick and Inputvp on the first file only, setting a new number of peaks, then you'll have to switch this option off. You can change the default setting in the customize fitting parameters menu for VARPRO.
9. The VARPRO results signals (the reconstructed and residual time domain signal) can be saved to file as ordinary data file, for use in later analysis if desired. Select the save signals menu item under the Results menu.
10. In the pH calculation, you can now indicate whether αNTP was fitted as a doublet, and MRUI will take the average of the two frequencies for the αNTP chemical shift in the pH calculation.
11. The batch result files produced after running a batch, each containing a particular parameter for all peaks in all signals of the batch, are now accompanied by a file of identical format, listing the Cramér Rao estimates for the standard deviations of the estimated parameter(s). The file extensions are those of the parameter file, with an `s` as third character.

Appendix D
MRUI and compatibility

Officially, the MRUI software is supposed to be platform-independent, which is especially true because it is largely based on the Matlab software, which itself behaves identically on different platforms (when you buy Matlab you have to say what kind of computer you have, and they'll provide you with the Matlab binaries for that type of machine, which has a machine-specific bottom end and a more or less platform-independent front end). Most of the differences, especially between UNIX machines and PC's have been accounted for in the MRUI software, be it in conditional machine tests in the MRUI scripts, a machine-dependent command database in the `$MRUIDIR/mrui/utils/compat.m` script (see section 2.1.3), or machine-dependent settings in the Makefiles for the Fortran and C programs (sections 2.3.2 and 2.4.2).
Obviously, differences will remain, the most prominent one concerning the looks of windows and menus between UNIX Xclient software and Microsoft's Windows software. That one should not harm the use of MRUI though. I would like to point out a few machine-specific quirks though, which could not be fully solved.

- PC users with Windows 95 should edit the `compat.m` script so that the COPY command reads:

  ```
  copy /y
  ```

- PC users may experience some resistance of the Matlab edit boxes to take your typed string: upon pressing `[Enter]` nothing may happen. It seems as if the pressing of `[Enter]` must really 'click' with Matlab's attention. So just try a few times, and you'll see that it will respond normally at one point.

- PC user may sometimes get unexplainable problems when MRUI tries to run a DOS command, e.g. when using exit without save in one of the customize functions:

  ```
  » ???  Problem running DOS command.
  Error in ==> d:\mrui_96.3\mrui\base_win\opt_ext.m
  On line 35  ==>
  eval(['! ',ucopy,' ',test_wd usep,'mrui.bak ',test_wd usep,'mrui.ini']);
  ??? Error using ==> !
  Error while evaluating callback string.
  ```

  This is caused by a file sharing violation in MS-DOS, which means that two processes try to access the same file simultaneously, which is not allowed. In this case for example, the file `mrui.ini` is replaced by `mrui.bak`, whereas apparently MRUI is still accessing `mrui.ini`, for example trying to read from or write to it. However, in the MRUI scripts, by the time this DOS copy command is issued, the `mrui.ini` file has already long been closed using the Matlab `fclose` command, so there should be no problem. There may be some inconsistency by Matlab's use of the `fclose` command under MS-DOS, or its interaction with the MS-DOS operating system. This problem has been brought to the attention of The Mathworks, Inc. Their first reaction was that Matlab's event queue does not get flushed continuously, which means that Matlab might carry on with the next line of code without having completely finished the previous one. Usually this is very clever (gains lots of calculation time!), but in this case it might actually try to overwrite the file before having properly closed it. A good way to flush the event queue (i.e. make sure Matlab finished the `fclose` command before issuing the DOS `copy` command) is to use the Matlab `pause` command. The fact that the use of the `pause` command did not help the situation indicates that the problem is more intricate and probably at the level of Matlab's communication with the DOS operating system. We are still waiting for a fix, but the problem is that this bug cannot be consistently reproduced. Sometimes it happens, other times it doesn't. Hopefully a Matlab upgrade (when is that Matlab 5.0 coming anyway?) will do wonders for buggy things like these.

- PC users who have partitioned their hard disk, with the Matlab software on disk C: and their data, or maybe also their `matlab\work_dir` directory on another disk (e.g. D:), start on disk C: when Matlab is fired up. MRUI will not run, because the `HOME` variable was set to `\` and `\matlab\work_dir\mrui.ini` will not be found on disk C: (it's on D:). Hence you must first change to disk D: before you can start up MRUI. In Matlab, changing disk is analogous to changing directory, and you can do both in one go:

  ```
  » cd d:\matlab\work_dir
  ```

- Users of Sun SPARC 20 workstations (really, that is the only type of machine on which this curiosity has been noticed) may find that the done button in the Peakpick function (see section 11.5) does not always register correctly. If that is the case, it will continue to ask you for the next peak, and the click on the done button has been taken as a frequency starting value, which is of course well outside the region of interest. In this case you should either edit the `peakpick.par` file manually (and don't forget to create an identical copy under the name `<name>.pkp`!) or re-do the Peakpick. Users of Sun SPARC 20 workstations unfortunately have to be recommended not to use the peakpick until [done] menu item in the Peakpick function.

- Some concern has been raised by users about the following messages:

  ```
  Note: IEEE floating-point exception flags raised:
      Inexact;  Underflow;
  See the Numerical Computation Guide, ieee_flags(3M)
  ```

  Or something like that. Especially Sun users can suffer from them. Especially when running batches, the whole Matlab command window can be filled by them, and they seem to pop up arbitrarily, without bad errors in fits or whatever. In fact, they can even occur when you simply convert a data file, and tests have proved that you can convert e.g. ASCII to binary, getting such an error, then convert the binary back to ASCII, getting an error again, and the two ASCII files will be *exactly* identical. Also, we have run SVD or VARPRO on simulation signals, and with the occurrence of the above ‘error’, the estimated parameters were still absolutely identical to the simulated parameter values.
  There is absolutely no need to worry. For certain compiler configurations, these messages can pop up at virtually every call to any Fortran program, whether there is any calculating in it or not. Let’s see what the Sun Manual has to say about it:

  “*The other two exceptions (underflow, and inexact) are seen very often, and they usually do not indicate program error. In fact, many more floating-point operations than not cause the inexact exception*”

  As a matter of fact, Bart de Schutter of the Katholieke Universiteit Leuven (ESAT/SISTA) has suggested a ‘fix’, which would prevent the warning from being displayed at all. As it will only work on Sun, I have not added the lines to the code of the MRUI programs distributed to the users. Sun users can get rid of the warning by adding the following line at the end of each Fortran source code file:

  ```
        i=ieee_flags("clear","exception","all")
  ```

  You may have to declare the variable `i` as an integer, and perhaps `ieee_flags` as an external function, and you may need to link to a specific IEEE library of your compiler.
  The purpose of this message is to take away any worries or suspicion from users who encounter these messages.

- Hewlett Packard users would get a Matlab ‘bus error’ (about the worst error you can possibly get) whenever they would use the File quit menu item. This works fine on all other machines, so it had something to with how the machine-dependent Matlab binary program for Hewlett Packard machines goes about clearing massive amounts of memory at the same time. When contacting The Mathworks, Inc. about this, they indeed admitted this bug in the Matlab binary program for HP in version Matlab version 4.2, and promised improvements for the next Matlab upgrade. However, what would an MRUI user on HP do if (s)he couldn’t even quit functions? The work-around I programmed was that instead of completely clearing the window concerned, using Matlab’s `delete()` function, I simply made the window invisible (of course *only* if MRUI would detect it was being used on an HP, all other platforms are *not* affected by this joke). This works fine too, but of course after a while your computer’s memory will get full, and you’ll have restart MRUI at regular intervals.

- DEC RISC users will get a Matlab ‘bus error’ (about the worst error you can possibly get) whenever they use the Next menus. The fact that no other type of machine yields this reproducible bug indicates that it has something to do with the machine-dependent Matlab binary program for DEC RISC (of course possibly in combination with the MRUI code, but it can’t be just me because it works fine on all other systems). See the previous HP oddity, which was of similar origins, and the reply of The Mathworks, Inc. Unfortunately, there is no work-around for this one, so DEC RISC users will have to live without the Next menus, at least until the next Matlab upgrade.

Appendix E
MRUI and its use of memory

This appendix explains how you can edit the software so that you can change the maximum number of peaks and the maximum number of data points. Some users (e.g. those with *in vitro* data) may wish to increase both variables at the cost of extra memory and longer calculation times. But the average user can read below how to reduce both variables and end up with programs that require much less memory, and run much faster!

Let me start with an apology. In MRUI_95.2, I increased the maximum number of peaks for VARPRO from 20 to 100, and for H(L)SVD from 50 to 100. I had understood that my HP workstation had 32 MB of RAM memory, and since the new settings did not cause me any trouble, I decided to make the changes in the official version. Then I started getting complaints from users with 32 (or less) MB RAM workstations about “out of memory” messages, and neither VARPRO nor (Filter_)HSVD would work. It turned out I had been wrongly informed, and that my HP workstation did in fact have 64 MB of RAM memory! Ron de Beer (Delft University of Technology) provided the following calculation to show why VARPRO with 100 peaks would cause memory problems:

“The largest array in the VARPRO code is the 2-dimensional array A, which for 2048 complex data points and 100 peaks can reach a size of around 32 MB. This is a working array, that has to contain the Jacobian matrix, amongst others. The large size of this working array can be understood considering that 2048 * 2 * 100 * 4 * 8 is already 13 MB (2048 data points; 2 because of complex data points; 100 peaks 4 parameters per peak; 8 because double precision is real*8). We now suffer from the Fortran77 requirement that the largest possible array be declared at initialization. Perhaps in the future with a Fortran90 version of VARPRO we can overcome this problem using run-time memory allocation.”

I checked the declaration of array A in VARPRO, and its dimensions are `i2nmax*ilpp2m`, where `i2nmax` is the number of data points (reals, i.e. 2*complex, i.e. 4096 for 2048 complex data points), and `ilpp2m` is `(ilm+i1inc)*icomp+2`, where `ilm` is 2 and `i1inc` is 8. 4 parameters per peak is not really implemented in the software; there are also reservations for estimation of the zero order phase and the begin time, and for Gaussian components. Hence the memory requirement for array A is: 2048 * 2 * 100 * 10 * 8 is exactly 32 MB. This caused the problems.

Currently the only solution to “out of memory” crashes is to reduce the number of peaks or the number of data points in the Fortran code -- or buy more RAM memory. For a workstation with 32 MB of RAM, you can use 2048 data points and 50 peaks without problem. If all your applications require a smaller number of resonances and/or data points, you should set the maximum numbers in the code as small as possible, because this will make the arrays smaller, hence reduce memory requirements and therefore significantly speed up the software! For MRUI_96.*, I have set the defaults back to 25 for VARPRO and 50 for H(L)SVD. The maximum number of data points remains at 2048, but you should realize that if your FIDs usually have died out before 1024, that setting the maximum in the code to 1024 greatly reduces the size and calculation time of the software. I will now explain how you can change the maximum number of peaks and data points in the Fortran code, and make corresponding changes in the MRUI scripts. Realize that after making changes to the Fortran code, you will have to recompile using the Makefiles (and distribute the executables to the target directories) for the changes to take effect. Changes made to the MRUI scripts will take effect once MRUI has been restarted.

At the top of each Fortran file, there is a `parameter(....)` statement. It contains constant variable settings, such as the maximum number of peaks and the maximum number of data points. I will refer to the names of the variables that have to be changed in these `parameter(....)` statements (these are the only positions where you will have to make your changes in the Fortran code. It is easiest to load the Fortran files in your favorite editor, search for the string `parameter` (skip the occurrences in text or comments) until you find the `parameter(....)` statement between the declaration of the variables, and simply edit the numerical value of the correct variable. See below.

Changes in the Fortran algorithms: VARPRO Fortran code

The variable for the maximum number of peaks is `icomp`, and for the maximum number of complex data points it is `inmax`.

Files in which you must change `icomp`:

```
            MRUI_9x.x/varpro/drivervp.f
            MRUI_9x.x/varpro/ada.f
            MRUI_9x.x/varpro/adad.f
            MRUI_9x.x/varpro/fcn.f
            MRUI_9x.x/varpro/inputvp.f
```

Files in which you must change `inmax`:

```
            MRUI_9x.x/varpro/driverp.f
            MRUI_9x.x/varpro/inputvp.f
```

Changes in the Fortran algorithms: AMARES Fortran code

The variable for the maximum number of peaks is `icomp`, and for the maximum number of complex data points it is `inmax`.

Files in which you must change `icomp`:

```
            MRUI_96.3/amares/drivernv.f
            MRUI_96.3/amares/update.f
            MRUI_96.3/amares/jaco.f
            MRUI_96.3/amares/func.f
            MRUI_96.3/amares/finderror.f
            MRUI_96.3/amares/linpar.f
            MRUI_96.3/amares/inpcheck.f
            MRUI_96.3/amares/inputnv.f
```

Files in which you must change `inmax`:

```
            MRUI_96.3/amares/drivernv.f
            MRUI_96.3/amares/update.f
            MRUI_96.3/amares/jaco.f
            MRUI_96.3/amares/func.f
            MRUI_96.3/amares/finderror.f
            MRUI_96.3/amares/linpar.f
            MRUI_96.3/amares/inpcheck.f
```

Changes in the Fortran algorithms: H(L)SVD Fortran code

The variable for the maximum number of peaks is `id3`, and for the maximum number of complex data points it is `id2`, which you set by setting `id1` to twice the number of complex data points. In HLSVD, there is an extra variable called `ma`, which determines the maximum number of Lanczos iterations. This parameter is rather elusive, but you should realize that when using HLSVD, the singular values are determined in an iterative process. HLSVD needs a starting value for the number of iterations it has to carry out in order to find the number of singular values you asked. This variable is `nit`. If you do not enter a value, MRUI will use an assumed default of 10 times the number of singular values (i.e. the number of components, or peaks, you're after). We have found from experience that this is a reasonable value if you fit only signal components (when trying to include many smaller components that are of the order of the noise, the number of iterations required will increase faster). After HLSVD has completed its `nit` iterations, it checks whether the number of singular values it has found is identical to, or larger than, the number you asked for. If this is the case it stops and calculates your parameters, and if it is not the case, it does an extra (numberofsinvals/10) iterations, checks again, etc. Note that re-checking costs calculation time! Each HLSVD run produces a file `<name>.tsv`. Check this file to become familiar with HLSVD behavior on your data. If you see only one line in `<name>.tsv`, saying that it has found <xx> singular values after <xxx> iterations, this means that it has found all your singular values in the number of iterations you (or MRUI) provided. Each extra line that occurs in this file means a restart of Lanczos iterations, and therefore some loss of calculation time due to rechecking the singular values. If you see this happen, set `nit` in MRUI to a bigger value so that you do not get restarts and HLSVD is more effective. Of course you do not want HLSVD to do too many iterations either, because that would also be a waste of time. If you try and find the right compromise for your applications, your analyses will be much faster! Because with Fortran77 the maximum array sizes have to be declared, we had to put in a maximum number of Lanczos iterations as well. Its variable is `ma`. The same consideration is valid here that the smaller you can keep this, the better it is. If your number of peaks is usually around 20, and you find that between 200 and 400 iterations always suffices, reduce `ma` from 1000 to e.g. 400 (keep a safe margin if you can). Normally, if for some reason the maximum is reached during a run, HLSVD will complete the analysis for the number of peaks it could determine so far.

Files in which you must change `id3`:

```
            MRUI_9x.x/hsvd/hlmain.f
            MRUI_9x.x/hsvd/hlsvd.f
            MRUI_9x.x/hsvd/hmain.f
            MRUI_9x.x/hsvd/hsvd.f
```

Files in which you must change `id1`:

```
            MRUI_9x.x/hsvd/hlmain.f
            MRUI_9x.x/hsvd/hlsvd.f
            MRUI_9x.x/hsvd/lanczos.f
            MRUI_9x.x/hsvd/hmain.f
            MRUI_9x.x/hsvd/hsvd.f
```

Files in which you must change `ma`:

```
            MRUI_9x.x/hsvd/hlmain.f
            MRUI_9x.x/hsvd/lanczos.f
```

If you make changes in the Fortran codes for maximum number of peaks and maximum number of data points, don’t forget to make the corresponding modifications in the MRUI Customize Fitting parameters (see section 3.2.1.4).

Tracing why and how HLSVD slobbered up my memory

By Miquel Cabañas, Universitat Autònoma de Barcelona.

This is a sad story with a happy end. In brief, I got MRUI 96.1 and followed all the steps to compile and install all the files as I am used to do. Surprisingly this time, I could not get the binaries to work (`hlmain`, `hmain`, etc.), neither from within MRUI nor from the command line:

```
> hlmain x1m.dat 1024 769 10 100
Killed
```

as you can see, there were not too many clues to know what was going on.
After some research, I found that this was due to the fact that I was running short of memory (too many applications eating bytes at the same time). As soon as I increased the available swap space (larger swap file) the problem was fixed. Now, MRUI_96.1 works fine, in fact, it works faster than former releases, and i would say that it also converges in less iterations!

If you have time, read the details below. I wrote this for Aad and he asked me to post it just in case someone had the same problems.
I downloaded MRUI_96.1 and compiled the binaries as usual. I always give the compiler the opportunity to optimize the code, not that I have found that the optimized executables run faster, actually, I have not made any comparison, I am just trusting the people that wrote the compilers (Sun `f77` and FSF `gcc`). In case I have problems with the compiled code this is one of the first things I get rid off.
Well, this time everything ran flawlessly... if it had not been for the fact that MRUI kept crashing every time I tried to filter the residual water peak (HLSVD). Then, I tried to run the program from the command line, and it came out with a mean ‘`Killed`’ comment:

```
> hlmain x1m.dat 1024 769 10 100
Killed
```

At this point I tried to debug [1] the code or trace it [2] during execution... All this to no success: the program insisted on killing itself before giving a chance to the other programs to catch what was going on :-(
So, I gave up, quit Matlab and decided I would compile the programs again, this time without the optimization option. And so I did, I recompiled the programs, and I tested them from the command line and from within MRUI and found them to work fine. Since it was late and I had an appointment, I sent a mail to Aad mentioning this issue and left the lab.
I should have known that it could not be that easy... This morning I checked that the MRUI was still up and running with the same file I had been working yesterday, and it worked fine. Then I tried another file, and it did not...
It was starting to make no sense at all. I had been filtering the water peak from one file without problems, but I could not do it with a different file! So I started again to try things: it did not work from the command line either

if MRUI was running (if I had been smart enough this should have pointed me to what the problem was), but I could run it when Matlab was not running.
And then I did it, I tried to run it from the command line, Matlab was running but I had not called MRUI yet (not that I did it on purpose)...

```
> hlmain x1m.dat 1024 769 10 100
  ld.so.1: hlmain: fatal: /opt/SUNWspro/lib/libF77.so.2:
     can't map segment: errno=11
     Killed
```

Finally! A message!
Of course, there was only one possible interpretation: not enough memory available... something I checked with `vmstat(1M)` [3] and `top`. By enlarging the swap space I was able to run all files without problem (by the way, it was in excess just for a bunch of bytes)
In conclusion, if you experience problems when running MRUI and they can not be ascribed to a bug in the code:
- run the program from the command line with MRUI loaded (running)
- monitor the virtual memory usage in a separate window
- run the MRUI programs from the command line

If you see any shortage of memory associated with a program crash you know what to do: ask your system administrator to increase the amount of memory, either on-board (expensive) or in the swap file (no cost).

[1] To debug the code, a symbolic table must be created at compilation time. This is accomplished by appending the `-g` option to the `COPTS` and `FOPTS` lines in the Makefiles.

[2] In Solaris 2.x (and I think in SunOS too) the command `truss(1)` is used to trace the execution of a program (memory allocation, system calls and so on). I have not found an equivalent command under IRIX, and I do not know about other O.S.

[3] `vmstat(1)` reports certain statistics about the system: virtual memory, processes status, CPU activity, and others. I know it can be found under Solaris 2.x and probably other O.S. have an equivalent command.

Printing on non-postscript printers and memory

Miquel Cabañas also hands us a suggestion to do with memory optimization. It turned out that in some cases Matlab performance (and hence MRUI) was degraded by very slow memory handling due to a (nearly) full swap space. Miquel found that the `/tmp` directory would get full in case you print from Matlab without having a real Postscript printer. If you do not have a Postscript printer, Matlab uses an emulator (`ghostscript`) so that the essentially Postscript graphs can be printed on your non-PS printer. However, this involves some extra information to go with the print job, and this is stored in two files, `*.rsp` and `*.jet`, in the `/tmp` directory. These files usually start with an uppercase `T`. The file `T*.rsp` contains the `ghostscript` commands and the file `T*.jet` contains the formatted printout. This is all done automatically through Matlab's `$MATLAB/toolbox/matlab/graphics/print.m` function, but unfortunately, these two files are never deleted once the print job has been completed. Thus, `/tmp` can get full very quickly if you print regularly. It might be, that this is not the case on other workstations, or that this flaw has been fixed in later Matlab versions, or that your system manager has set up scripts so that `/tmp` is cleared regularly. However, if you do suffer from poor memory performance, it might be worth having a look in your `/tmp` directory, and make sure it is not full of `*.rsp` and `*.jet` files. You could insert a line in one of your startup files like:

```
# rm /tmp/*.rsp /tmp/*.jet
```

To clear it automatically at login, if you have noticed that you do create these files.

Memory Management in Matlab

Finally, since we are on the subject of memory, you might want to read the following Matlab Technical Note:

```
Topics Common to All Versions of MATLAB:

o  MATLAB has three functions to improve the way in which memory
   is handled: CLEAR, PACK and QUIT.
```

CLEAR	- removes variables from memory
PACK	- saves existing variables to disk, then reloads them contiguously
QUIT	- exits MATLAB and returns all allocated memory to the system

For more information on the above commands, please refer to the MATLAB Reference Guide.

What Does "Out of Memory" Mean?

Typically when the "Out of Memory" message appears, it is because MATLAB asked the operating system for a segment of memory larger than that which is currently available. Use of the above mentioned techniques: pre-allocation, CLEAR, PACK, SAVE, LOAD, and QUIT, help optimize the available memory. If the "Out of Memory" message still appears consistently:

o Increase the size of the swap file.

o Make sure that there are no external constraints on the memory accessible to MATLAB (on UNIX systems use the LIMIT command to check).

o Add more memory to the system.

o Reduce the size of the data you are using.

Appendix F
Upgrading from an older MRUI version

Once you have established your specific settings in the Makefiles, save them under a backup name so that they will not be overwritten by the general version that will appear in the next setup. During an upgrade you can then just copy your previous Makefile back to their positions and do compilations simple by typing `make`. If the contents of a Makefile change you can easily establish this by typing:

```
# diff Makefile My_Tuned_Makefile
```

Where `My_Tuned_Makefile` stands for the name of your specific Makefile. If there is nothing new in the Makefile that I supply, you will only see the differences in settings that you have made. When I have added or removed files or changed names in the Makefile, you will see this as well, and you can make your changes accordingly.

Making result files from previous versions conform to the new format in newer versions

There have been very minor changes in the text of a few scratch files. This may cause unexpected results when loading old result files into the newer versions (from version 96.1 onwards) of MRUI. Users who have written their own scripts to read these scratch files, may need to modify their scripts slightly and can obtain from me a list of the exact changes. The scale files (`*.scl`) have been greatly simplified to aid clearer absolute quantitation. This upgrade requires conversion of old files when they are to be re-used in MRUI_96.3. You can do this in one go by using the `mruio2n` command. Change directory to where the old files are that you want to re-use, and simply type `mruio2n` in the an operating system shell, or`!mruio2n` in the Matlab command window. This lists some information on the use of the program. `mruio2n all` updates all files in the current directory, `mruio2n <name>` all files that belong to the data file with general file name base <name>. `mruio2n help` gives more detailed help. You can also call the `mruio2n` command from the Conversion menu under the MRUI base window; it is the OLD -> NEW menu item under the MRUI menu item. The `mruio2n` command will update all types of file automatically, whether binary data files, Peakpick files, SVD_1D/VARPRO result files, or scale files. It will automatically decide whether a file is up to date or whether it needs to be converted. A detailed report is written to a file called `mruio2n.history` in your home directory. The `mruio2n` command only works on UNIX machines (but PC users have no old version to update from...).

Finally, when you upgrade from version MRUI_96.1 or older, do not forget to run the `opt_upg` script in the Matlab command window before starting the new MRUI. See sections 2.3.4 and 2.4.4.

Appendix G
Compiling with public domain Fortran / C compilers

The following is a detailed report on how to compile the Fortran and C programs of MRUI using the public domain GNU `g77` and `gcc` compilers. This is all due to the hard work put in by Irini Lekka-Banos of the Uppsala University Hospital in Sweden (`Irini.Lekka_Banos@radiol.uu.se`). In fact, she even describes how to install the gcc and g77 compilers on your computer.

MRUI_96.x installation using `gcc` and `g77` on SunOS 4.1.3

This document gives a brief description on:

a) How to install `gcc-2.7.2` and `g77-0.5.18` on a Sun SARC Workstation running SunOS 4.1.3.
b) How to install MRUI_96.x using `gcc` and `g77`.

Prerequisites

You have to download from a GNU site the following files:

`gcc-2.7.2.tar.gz`	(GNU C compiler)
`g77-0.5.18.tar.gz`	(GNU Fortran compiler, to be merged with `gcc`)

You will also need to download from the same place the following tools, if they do not already exist on your system:

`gzip`	(`gunzip` utility to unpack the above)
`patch`	(patch program to patch some `g77` files)
`make`	

You must also have a C compiler (`cc`), Matlab and about 120 MB of free space.

Installation procedure

It is assumed that `gcc-2.7.2.tar.gz` and `g77-0.5.18.tar.gz` reside in `/usr/local` and that `gcc` and `g77` will be installed under `/usr/local`. It is also assumed that Matlab is installed under `/usr/local`. `gcc` can be installed and run independently but `g77` requires `gcc` in order to be made.

1. Unpack the `gcc` and `g77` distributions:
```
# cd /usr/local
# gunzip gcc-2.7.2.tar.gz
# tar xvf gcc-2.7.2.tar
# gunzip g77-0.5.18.tar.gz
# tar xvf g77-0.5.18.tar
```

2. Make symbolic links for convenience and move `g77` under `gcc`.
```
# ln -s gcc-2.7.2 gcc
# ln -s g77-0.5.18 g77
# mv -i g77/* gcc
```

3. Install all the necessary patches.
```
# patch -p1 -V t -d gcc-2.7.2 < gcc-2.7.2/f/gbe/2.7.2.diff
# cd gcc
```

4. If your system does not have an `f77` command then:
```
# touch f2c-install-ok
# touch f77-install-ok
```

5. Configure your system and set the target directory. In this case, it is `/usr/local`. Then `gcc` and `g77` will be installed in `/usr/local/bin` and `libfc2.a` in `/usr/local/lib`. If you wish to have another target directory you have to define it here.

```
# ./configure --prefix=/usr/local
```

6. Do the GNU Fortran patch just for the SUNOS system. In the file `gcc/f/proj.h` edit the line reading

```
#define FFEPROJ_STRTOUL 1 ...
```

   by replacing `1` with `0`.

7. At this point, the `INSTALL` file recommends to make `gcc` and `g77` in one step just by typing:

```
# make bootstrap
```

   However, it is mentioned that you may get a few error messages, like I did. So, I ran the make in stages and I also linked the `libgcc.a` library in order to resolve the `___main undefined` that I was getting (for more details on this refer to `gcc.info-8` in the `gcc` distribution). Instead of the `make bootstrap` I ran:

```
# make LANGUAGES=c
# make stage1
# make CC="stage1/xgcc -Bstage1/" CFLAGS="-g -O" LDFLAGS="-lgcc"
# make stage2
# make CC="stage2/xgcc -Bstage2/" CFLAGS="-g -O" LDFLAGS="-lgcc"
```

   Ignore the following warnings:

```
"statement not reached" in insn-emit.c
"unknown escape sequence"
```

8. Remove the stage1 compiler to free up some space and then install and verify that `gcc` and `g77` are installed.

```
# rm -rf stage1
# make -k install
# gcc -v
# g77 -v
```

   In `/usr/local/bin` you will find both `f77` and `g77`. They are exactly the same. If you need more details or information on the above procedure, please refer to the `INSTALL` and `gcc.info` files under the `gcc` and `g77` distributions. At this point `gcc` and `g77` are installed, your system is therefore ready for the MRUI_96.x installation.

9. You must have an `mrui962.tar.gz` file on your system. Move it under the matlab directory and unpack it.

```
# mv mrui962.tar.gz /usr/local/matlab
# cd /usr/local/matlab
# gunzip mrui962.tar.gz
# tar xvf mrui962.tar
```

10. The MRUI_96.x compilation and installation is fairly easy. I simply followed the instructions as described under `MRUI_96.x/info/readme.1st`. I set the following variables in the Makefiles:

```
STD_LIB = -lf2c
IEEE_LIB = -lm_p
```

   `TARG_DIR` and `LIB_DIR` were set to the destination directories of my choice. `gcc` and `f77`(`g77`) are already set as the default compilers in the Makefile.

I then ran `make all` and `make install` for all the programs, as described in `readme.1st`. I got two problems:

i) A set of `undefine`s under `a2b`:

```
read_1d_mat__
write_1d_mat__
read_2d_mat__
write_2d_mat__
```

   To resolve these I had to edit the function names in `matlab.c` from

```
read_1d_mat_
write_1d_mat_
read_2d_mat_
```

```
        write_2d_mat_
```
to:
```
        read_1d_mat__
        write_1d_mat__
        read_2d_mat__
        write_2d_mat__
```

ii) I also had to edit in `lpsvd/lp_scr.f`:
```
        external function iargc
        external function getarg
```
to:
```
        external iargc
        external getargc.
```

Index

Colophon

This manual was edited using Microsoft Word for Windows version 6.0c, on a MESH 486DX2/66 PC with 32 MB RAM memory, running MS-DOS 6.0 and MS-Windows 3.1. Original pages were printed on a Brother HL-660 laser printer. All figures in this manual are captured bitmap files (Windows Meta File format, WMF), from running MRUI_96.3 on the above mentioned PC under Matlab 4.2c.1. The shareware program Paint Shop Pro version 3.0 was used to capture the MRUI windows, reduce their color depth (from 16 million colors to 256), and save them as Windows Meta Files (WMF), after which they could be inserted in the Word document.

Bibliography

[1] T. Binzoni, V. Quaresima, E. Hiltbrand, L. Gürke, P. Cerretelli, M. Ferrari, Influence of repeated ischaemia / reperfusion cycles (ischaemic preconditioning) on human calf energy metabolism by simultaneous near infrared spectroscopy and ^{31}P-NMR measurements. *XXIV Int. Soc. Oxygen Transport to Tissue*, S4.4, Dundee (1996)

[2] V. Vondra, A. van den Boogaart, D. Graveron-Demilly, D. van Ormondt, Automatic estimation of the time origin and overall phase of an FID signal. *J. Magn. Reson. A* **119,** 271-274 (1996)

[3] R. de Beer, "Quantitative *In Vivo* NMR", lecture notes c59, U. Delft, 1994; accessible at `http://dutnsic.tn.tudelft.nl:8080/c59_to_html/c59.html`

[4] A. van den Boogaart, M. Ala-Korpela, J. Jokisaari, J.R. Griffiths, Time and frequency domain analysis of NMR data compared: an application to 1-D ^{1}H spectra of lipoproteins. *Magn. Reson. Med.* **31,** 347-358 (1994)

[5] H. Barkhuijsen, R. de Beer, D. van Ormondt, Error theory for time-domain signal analysis with linear prediction and singular value decomposition. *J. Magn. Reson.* **67,** 371-375 (1986)

[6] K.L. Behar, D.L. Rothman, D.D. Spencer, O.A.C. Petroff, Analysis of macromolecule resonances in ^{1}H NMR spectra of human brain. *Magn. Reson. Med.* **32,** 294-302 (1994)

[7] A. Diop, A. Briguet, D. Graveron-Demilly, Automatic *in vivo* NMR data processing based on an Enhancement Procedure and Linear Prediction method. *Magn. Reson. Med.* **27,** 318-328 (1992)

[8] A. Diop, W. Kölbel, D. Michel, A. Briguet, D. Graveron-Demilly, Full automation of quantitation of *in vivo* NMR by LPSVD(CR) and EPLPSVD. *J. Magn. Reson. B* **103,** 217-221 (1994)

[9] A. Diop, Y. Zaim-Wadghiri, A. Briguet, D. Graveron-Demilly, Improvements of quantitation by using the Cadzow Enhancement Procedure prior to any Linear-Prediction methods. *J. Magn. Reson. B* **105,** 17-24 (1994)

[10] H. Chen, S. Van Huffel, C. Decanniere, P. Van Hecke, A signal-enhancement algorithm for the quantification of NMR data in the time domain. *J. Magn. Reson. A* **109,** 46-55 (1994)

[11] S. Cavassila, B. Fenet, D. Graveron-Demilly, Improvements of the spectral resolution of linear prediction methods. *Proc. III SMR,* 1951 (1996)

[12] U. Klose, *In vivo* proton spectroscopy in presence of eddy currents. *Magn. Reson. Med.* **14,** 26-30 (1990)

[13] A.A. de Graaf, W.M.M.J. Bovée, QUALITY: quantification improvement by converting to the Lorentzian type. *Magn. Reson. Med.* **13,** 343-357 (1990)

[14] R. de Beer, A. van den Boogaart, D. van Ormondt, W.W.F. Pijnappel, J.A. den Hollander, A.J.H. Mariën, P.R. Luyten, Application of time-domain fitting in the quantification of *in vivo* ^{1}H spectroscopic imaging data sets. *NMR Biomed.* **5,** 171-178 (1992)

[15] A. van den Boogaart, D. van Ormondt, W.W.F. Pijnappel, R. de Beer, M. Ala-Korpela, *in* "Mathematics in Signal Processing III", (J.G. McWhirter, Ed.), p. 175-195, Clarendon Press, Oxford, 1994

[16] D.E. Saunders, F.A. Howe, A. van den Boogaart, M.A. McLean, J.R. Griffiths, M.M. Brown, Continuing ischaemic damage following acute middle cerebral artery infection in man demonstrated by short echo proton spectroscopy. *Stroke* **26,** 1007-1013 (1995)

[17] W.W.F. Pijnappel, A. van den Boogaart, R. de Beer, D. van Ormondt, SVD-based quantification of magnetic resonance signals. *J. Magn. Reson.* **97,** 122-134 (1992)

[18] M. Stubbs, A. van den Boogaart, C.L. Bashford, P.M.C. Miranda, L.M. Rodrigues, F.A. Howe, J.R. Griffiths, ^{31}P magnetic resonance spectroscopy studies of nucleated and non-nucleated erythrocytes; time domain analysis (VARPRO) incorporating prior knowledge can give information on the binding of ADP. *Biochim. Biophys. Acta* **1291,** 143-148 (1996)

[19] A. van den Boogaart, "The use of signal processing algorithms to obtain biochemically relevant parameters from magnetic resonance data sets". Ph.D. thesis, U. London, 1995

[20] R. de Beer, D. van Ormondt, Analysis of NMR data using time domain fitting procedures. *in* "NMR basic principles and progress", (P.Diehl, E. Fluck, H. Günther, R. Kosfeld, J. Seelig, M. Rudin, Eds.), p. 201-248, Vol. NMR 26, Springer-Verlag, Berlin, 1992

[21] H. Barkhuijsen, R. de Beer, D. van Ormondt, Improved algorithm for noniterative time-domain model fitting to exponentially damped magnetic resonance signals. *J. Magn. Reson.* **73,** 553-557 (1987)

[22] H. Barkhuijsen, R. de Beer, W.M.M.J. Bovée, D. van Ormondt, Retrieval of Frequencies, Amplitudes, Damping Factors, and Phases from Time-Domain Signals Using a Linear Least-Squares Procedure. *J. Magn. Reson.* **61,** 465-481 (1985)

[23] W. Kölbel, H. Schäfer, Improvement and Automation of the LPSVD Algorithm by Continuous Regularization of the Singular Values. *J. Magn. Reson.* **100,** 598-603 (1992)

[24] S. Van Huffel, H. Chen, C. Decanniere, P. Van Hecke, Algorithm for Time-Domain NMR Data Fitting Based on Total Least Squares. *J. Magn. Reson. A* **110,** 228-237 (1994)

[25] W.W.F. Pijnappel, "Quantification of 1-D and 2-D Magnetic Resonance Spectroscopic Data", Ph.D. Thesis, U. Delft, 1991

[26] A. van den Boogaart, F.A. Howe, L.M. Rodrigues, M. Stubbs, J.R. Griffiths, In Vivo ^{31}P MRS: Absolute Concentrations, Signal-to-Noise and Prior Knowledge. *NMR Biomed.* **8,** 87-93 (1995)

[27] G.H. Golub, R. Pereyra, The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate. *SIAM J. Numer. Anal.* **10,** 413-432 (1973)

[28] L. Vanhamme, A. van den Boogaart, S. Van Huffel, Improved Method for Accurate and Efficient Quantification of MRS Data with Use of Prior Knowledge. *submitted to J. Magn. Reson.* (1996); also Internal Report 9702, U. Leuven (1997)

[29] J.W.C. van der Veen, R. de Beer, P.R. Luyten, D. van Ormondt, Accurate Quantification of *in Vivo* ^{31}P NMR Signals Using the Variable Projection Method and Prior Knowledge. *Magn. Reson. Med.* **6,** 92-98 (1988)

[30] A. Knijn, R. de Beer, D. van Ormondt, Frequency-Selective Quantification in the Time Domain. *J. Magn. Reson.* **97,** 444-450 (1992)

[31] A. van den Bos, Parameter Estimation. *in* "Handbook of Measurement Science" (P.H. Sydenham, Ed.), pp. 331-377, Vol. 1, John Wiley & Sons Ltd., London, 1982

[32] accessible at `http://www.netlib.org/`

[33] D.E. Saunders, F.A. Howe, A. van den Boogaart, M.A. McLean, J.R. Griffiths, M.M. Brown, Continuing Ischaemic Damage Following Acute Middle Cerebral Artery Infarction in Man Demonstrated by Short Echo Proton Spectroscopy. *Stroke* **26,** 1007-1013 (1995)

[34] J.W. Prichard, J.R. Alger, K.L. Behar, O.A.C. Petroff, R.G. Shulman, Cerebral Metabolic Studies *in Vivo* by ^{31}P NMR. *Proc. Natl. Acad. Sci. USA* **80,** 2748-2751 (1983)

[35] J.E.Dennis, D.M.Gay, R.E.Welsch, Algorithm 573. NL2SOL-An Adaptive Nonlinear Least-Squares algorithm. *ACM TOMS* **7,** 369-383 (1981)