ELSEVIER

Magnetic Resonance Materials in Physics, Biology and Medicine 12 (2001) 141–152

MAGMA

Magnetic Resonance Materials in Physics, Biology and Medicine

www.elsevier.com/locate/magma

Java-based graphical user interface for the MRUI quantitation package

A. Naressi [a], C. Couturier [b], J.M. Devos [c], M. Janssen [d], C. Mangeat [a], R. de Beer [b], D. Graveron-Demilly [d,*]

[a] *Facultät für Physik und Geowissenschaften, Universität Leipzig, Linnéstrasse 5, Leipzig, Germany*
[b] *Department of Applied Physics, University of Technology Delft, PO Box 5046, 2600 GA Delft, The Netherlands*
[c] *Signal Processing Laboratory, National Technical University of Athens, Athens, Greece*
[d] *Laboratoire de RMN, CNRS UMR Q5012, Université Claude Bernard LyonI-CPE, 43 Boulevard du 11 Novembre 1918, 69622 Villeurbanne Cédex, France*

Received 3 February 2000; received in revised form 23 October 2000; accepted 26 October 2000

Abstract

This article describes the Java-based version of the magnetic resonance user interface (MRUI) quantitation package. This package allows MR spectroscopists to easily perform time-domain analysis of in vivo MR spectroscopy data. We show that the Java programming language is very well suited for developing highly interactive graphical software applications such as the MRUI software. We have also established that MR quantitation algorithms, programmed in other languages, can easily be embedded into the Java-based MRUI by using the Java native interface (JNI). This new graphical user interface (GUI) has been conceived for the processing of large data sets and uses prior knowledge data-bases to make interactive quantitation algorithms more userfriendly.

Keywords: Java; Graphical user interface (GUI); Magnetic resonance spectroscopy (MRS); Quantitation; MRUI package

1. Introduction

1.1. A new GUI for the MRUI package

The advanced signal processing software package magnetic resonance user interface (MRUI) [1] allows MR spectroscopists to easily perform time-domain analysis of in vivo MR data.

In the first MRUI version, the graphical user interface (GUI) ran in the MATLAB [2] environment [3,4]. In fact, the GUI part was written as a set of MATLAB M-scripts and functions. This requires MATLAB version 4.0 or higher. This MATLAB version is now used world-wide by about 300 groups. In the context of the TMR European project *Advanced Signal Processing for Medical Magnetic Resonance Imaging and Spectroscopy* [5] funding the MRUI development, it was decided to undo the MATLAB software requirement. A new and more user-friendly GUI has been designed, written completely in the Java programming language [6–11]. In this article, we describe the essential components of the GUI which has an extensible architecture, and the (new) functionalities of the MRUI software package.

The Java-based MRUI package offers.

- Black box quantitation algorithms based on singular value decomposition (SVD); the state space methods HSVD [12,13], HLSVD [14] and HTLS [15] and the linear prediction methods LPSVD [16], EPLPSVD [17,18] and LPSVD(CR) [19–21]. These non-interactive black box techniques are efficient for quantitating signals with good signal-to-noise ratios. They are also helpful in parametrizing signals of unknown composition and shape, but they cannot make use of all available prior knowledge.
- Non-linear least-squares (NLLS) quantitation algorithm; AMARES [22]. This is an improved version of VARPRO [13,23]. For more accurate quantitation

* Corresponding author. Tel.: +33-4-72431049; fax: +33-4-72448199.
E-mail address: danielle.graveron@univ-lyon1.fr (D. Graveron-Demilly).

1352-8661 01/$ - see front matter
PII: S1352-8661(01)00111-9

of e.g. in vivo MRS data, prior knowledge has to be imposed on the parameters. This can be easily done using NLLS methods. The Java-based MRUI offers the possibility to store the prior knowledge in a data-base.

- Preprocessing algorithms such as rapid removal of dominant signals using HLSVD [14,24,25], or time-frequency analysis [26], Cadzow enhancement procedure for noise reduction [17,18], ER-Filter [27] for frequency selection and Gabor tools for peak extraction and dynamic phase correction [28].
- Estimation of spectral parameters with their confidence intervals (Cramér-Rao lower bounds) [29–31].
- Conversion routines for data files from most manufacturers.
- Signal simulations.

In future Java-based versions, finite impulse response (FIR) filters [32,33], time-series analysis based on AMARES [22,34], 2-D tools and advanced features such as spin-product operator modeling [35], a Cramér-Rao tool box and algorithms for processing of magnetic resonance spectroscopic imaging (MRSI) data will be included.

1.2. Why Java

Since Java is an *object-oriented* language, writing software applications in Java means that one can benefit from the concepts of object technology [36–40]. Apart from this important general aspect, there are some specific reasons why Java is our programming language of choice.

- Java source code is compiled into Java *bytecode*, which is claimed to be *platform-independent*. Hence Java programs can run on any platform with a Java virtual machine (the Java interpreter and run-time system [6]).
- The standard Java library contains the Swing package [10,11], which is a complete collection of GUI elements.
- Java supports *multi-threading* [41], which means that parts of a computer program can be executed in separate threads. Each thread executes its code independently of the other threads in the program. This offers the opportunity of enhancing the performance of programs, especially in case of interactive graphical applications, and, the optimization of computer resources.
- Java supports the Java native interface (JNI) [42], which enables users to make calls to *native* code.
- With Java one can create so-called *applets*, which are programs that can be embedded in WWW pages. This means that they can be downloaded from any WWW server.
- Because of native code, the MRUI is not an applet. But in the future, embedding presentation functionalities in an applet and computational functionalities from a server could be helpful.

First, we give an overview of the software architecture. Next, we present two examples, the first is a quantitation with AMARES of an in vivo ^{31}P signal of a human brain, the second corresponds to the application of SVD-based methods for quantitating peaks of unknown shape. Finally, we discuss the implementation of the Java-based MRUI in terms of software and hardware.

2. Software architecture

In this section, we give an overview of the software architecture used in our Java-based GUI. We use the so-called unified modeling language (UML) class diagrams. UML is a standard notation for visually describing and interpreting object-oriented software applications [39,40]. An important aspect, is that the GUI has been conceived and designed for the processing of large data sets (time-series, MRSI).

2.1. Directory/package hierarchy

In Fig. 1, the directory/package hierarchy of the MRUI software system is shown. The notation is such that `/PATH/mrui/name` indicates a subdirectory/MRUI/name containing the source codes of the Java package [6] MRUI. Furthermore, `/PATH/mrui/`*name*`/`*methods* stands for subsubdirectories/`mrui`/name/methods, containing *native* source codes (mostly in Fortran 77) of related quantitation or preprocessing methods. The subdirectory/`PATH/mrui/`*graph* contains all the Java source code dealing with the graphical aspects. Finally, `/PATH/mrui/`*images* denotes the subdirectory, `/mrui/`*images* containing system icon files.

From Fig. 1, it can be seen that mrui is the leading Java package. This package contains, among others, the `mrui` class, which is the class running the `main` method [6] of the Java application. The subpackages `mrui.name` take care of various aspects of the MRUI software [3] such as conversion of NMR spectrometer data files, quantitation, preprocessing or simulation of magnetic resonance spectroscopy (MRS) signals, graphical presentation of results and tracking of preprocessing history of MRUI sessions.

In the next subsections the `mrui` package and related `mrui.name` subpackages will be discussed.

2.2. The mrui package

The `mrui` class of the `mrui` package is the class being responsible for launching the MRUI application. To that end, its main method creates an instance (an object) of the `mrui` class. During initialization the `mrui`

constructor creates a `DesktopWin` object and a `History` object (see Fig. 2). Another important task of `mrui` is that it takes care of providing user-selected working modes, such as one-dimensional, two-dimensional or time-series MRS or MRSI. The various MRUI GUI activities, accessible via the `MainWin` main window, are generated by clicking/selecting Swing GUI components of the `MainMenu` or the `MainToolBar` class (see again Fig. 2).

A new interesting feature of the Java-based MRUI GUI is that one can follow the various preprocessing steps, taken in the course of an MRUI session. To that end, instances of the `Preprocessing`, `Data` and `History` classes work closely together. Also, the `OutputToolBar` attribute of `MainWin` (see Fig. 2) is directly involved in this visualization of MRUI preprocessing history. Since the `Data` class implements the `Serializable` interface, complete states of objects of that class can easily be stored on disk by calling the `writeObject` method of an `ObjectOutputStream` [7]. This mechanism offers the MRUI user the opportunity of *undoing* preprocessing steps.

```
/PATH/
  mrui
  mrui/conversion
  mrui/quantitation
  mrui/quantitation/methods
  mrui/preprocessing
  mrui/preprocessing/methods
  mrui/simulation
  mrui/graph
  mrui/history
  mrui/design
  mrui/modelfts
  mrui/modelgabor
  mrui/mrsi
  mrui/images
```

Fig. 1. Directory/package hierarchy of MRUI software system.

2.3. The `mrui.conversion` package

Conversion of the file structure of commercial NMR spectrometer data files into a structure suited for the MRUI package is of paramount importance. The `mrui.conversion` package contains a number of classes to deal with this activity. By checking the file extension of the data files and file internal information, the MRUI software is capable in an automated way of choosing the correct conversion algorithm. At present, conversion algorithms for various commercial NMR spectrometers are available, including Bruker, GE, MR Research (SMIS), Philips, Siemens and Varian. Also, the two internal formats of the MATLAB MRUI (.dat and .mat) and ASCII format are supported.

2.4. The `mrui.quantitation` package

Quantitating (in vivo) MRS signals is one of the core activities when running the MRUI program. The quantitation process in fact means fitting some form of model function to complex-valued data points, sampled in the time-domain [3]. The MRS signals can have either a one-dimensional (1-D) or two-dimensional (2-D) structure, or can be a time-series of 1-D signals. Furthermore, the MRS signals can be acquired either as free induction decays (FID's) or as nuclear spin echoes [3].

Time-domain methods can be classified into two families, *interactive* and *non-interactive* methods. The latter, the black-box methods, require minimal user interaction and make use of the properties of the exponentially damped sinusoids and of singular value decomposition techniques. They are based on linear prediction (LPSVD) [16,19–21] or on state space methods (HTLS, HSVD, HLSVD) [12–15]. Although they are attractive because they are fully automatic, their serious drawback is that only limited prior knowledge about the model function can be incorporated in these algorithms. In contrast, interactive methods require more user interaction but allow inclusion of a considerable degree of prior knowledge. This prior knowledge can be knowledge on spectral parameters of the signals but also knowledge on instrumental effects. These interactive methods, such as VARPRO [23] or AMARES [22,34], are based on NLLS fitting and can accommodate any model functions.

In the next subsubsections, we describe how to impose spectral prior knowledge on MRS signal parameters and how to handle quantitation of MRS time-series. This will be illustrated using AMARES.

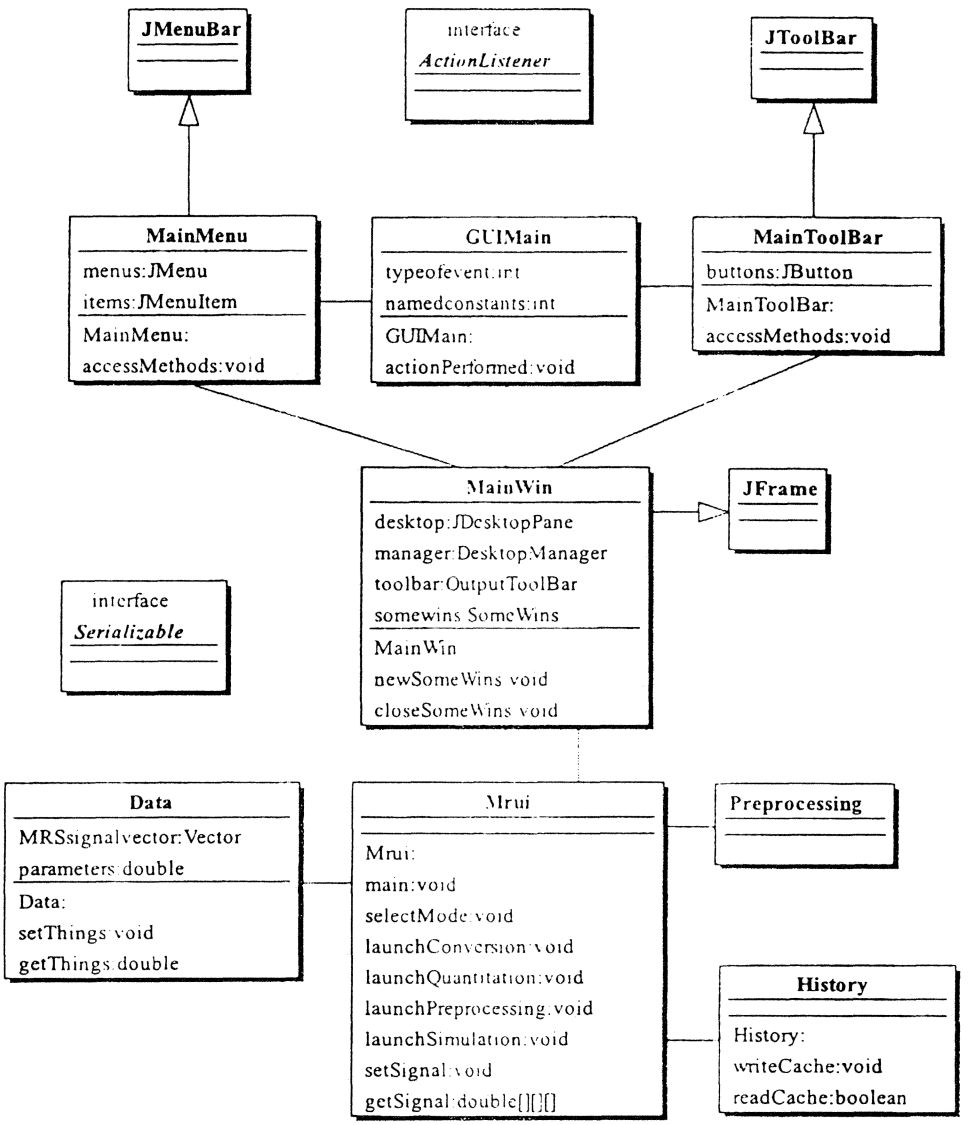

Fig. 2. UML class diagram of the MRUI package.

2.4.1. Handling prior knowledge on MRS signals

Extracting parameters from MRS signals by fitting model functions to the data usually requires a lot of user interaction. This stems from the fact that NL (LS) fitting methods demand starting values for the non-linear parameters involved (see for instance [22]). In addition, providing prior knowledge on the MRS signals requires strong user interaction. This prior knowledge, usually coming from the field of biomedicine/chemistry, can often be expressed as linear relations between model parameters of the same kind. For instance, imposing chemical prior knowledge on multiplet structures (i.e. knowledge of relative frequencies, amplitude ratios) can increase the quantitation precision by an order of magnitude [31]. Its use is highly recommended.

Supposing that an MRS signal of a certain type is being quantitated for the first time, a first step to be performed is to obtain starting values for the spectral parameters via *peak-picking* on the Fourier transform of the signal [3] as can be seen in Fig. 3. Performing peak-picking, actually consists of interactively collecting, for each peak involved, its top/centre position and position at which the linewidth should be measured. The first value provides an estimate of the peak frequency. The second allows to estimate the decay constant, assuming one knows the decay function [3]. These starting values can be saved in a starting value data-base and reused for all experiments of the same type.

After obtaining starting values, the second step of an AMARES-based MRS quantitation, is to provide spectral prior knowledge on the NMR peaks. This step is optional but highly recommended when prior knowledge is available. To that end, a number of classes were

developed, being part of the `mrui.quantitation.Amares.model.priorknowledge` Java package. The central prior knowledge class is called `PriorKnowledge`. Its task is to fill, retrieve and change prior knowledge information. This prior knowledge information concerns the parameters related to an NMR peak, i.e. the frequency, the time-domain damping (decay) factor, the amplitude, the phase and the *kind* of damping function. Directly related to `mrui.quantitation.amares.model.priorK.RelationList`, is the `mrui.quantitation.amares.model.priorK.Relation class` (RelationList is an aggregation of relation). This class is to be used to capture prior knowledge-based constraints on spectral parameters such as ratios and shifts and upper and lower bounds [22].

Providing prior knowledge in the Java-based MRUI is straightforward. The Java code, based on a grammar, allows the users to build/modify the sentences defining the wished constraints (see Fig. 4) for the capture of prior knowledge. Prior knowledge can be stored on disk into a *prior knowledge data-base*. An interesting and powerful feature is that prior knowledge can be saved for the full spectrum (i.e. phosphorus spectrum), a part of the spectrum (i.e. ATP) and/or for some basic structures such as doublet, triplet, etc. It is then possible, to load/reuse/combine the wished prior knowledge by loading the prior knowledge information from the data-base. A still easier way, based on default files, will allow the users dealing with the same types of signals to circumvent these steps and launch AMARES with one click.

Then the AMARES quantitation code [22] is to be called. Since it is written in Fortran, calling the code from the MRUI Java environment should be carried out via the JNI [42]. At present, a direct call to Fortran via JNI is not supported. To circumvent this problem, we work with an interface function written in ANSI C.

2.4.2. Handling quantitation of MRS time-series

For various biomedical applications, it is of interest to follow certain spectral parameters of MRS signals as a function of time. A well known example of such a time-series investigation is the study of ^{31}P-MRS of calf muscle as a function of exercise protocols (see for instance [17,43]).

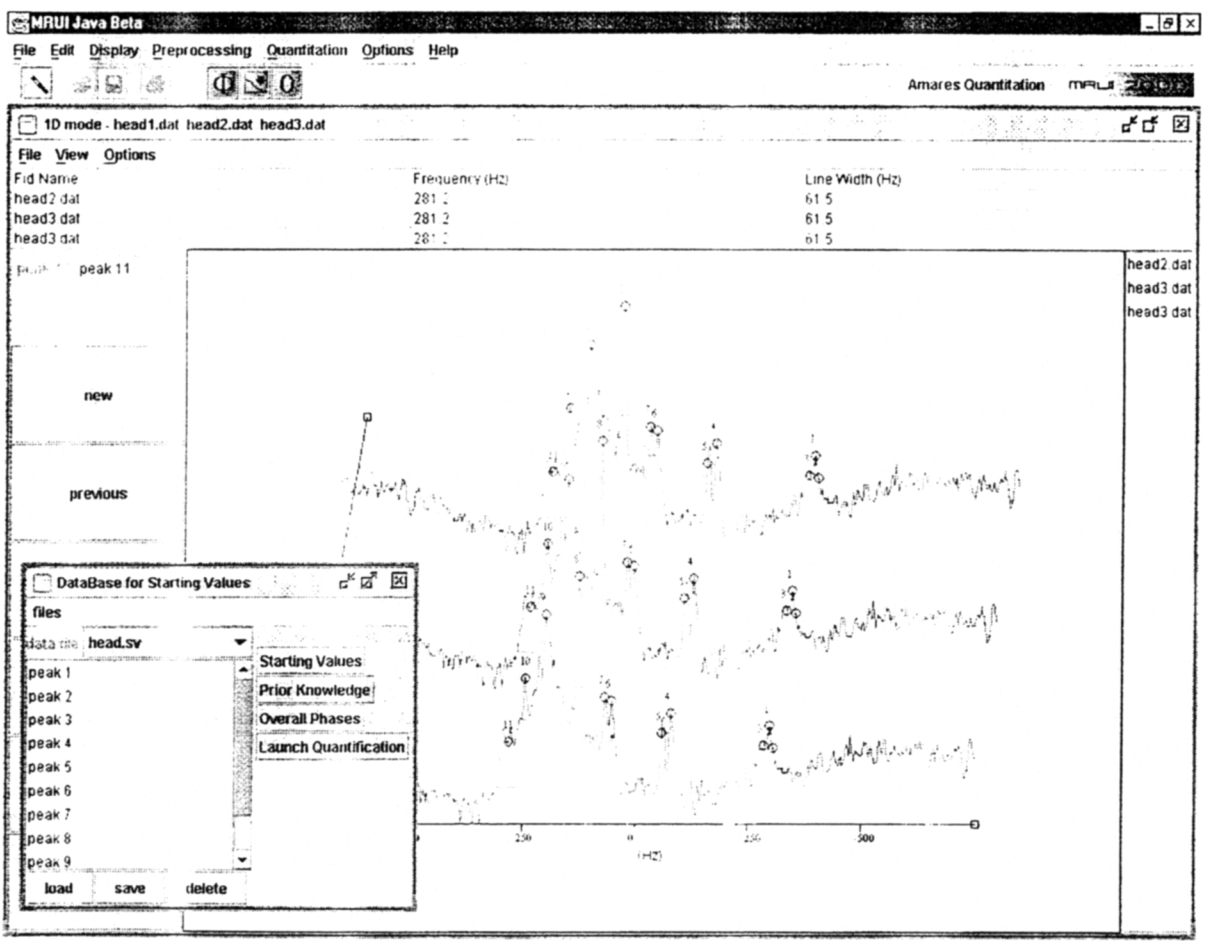

Fig. 3. First step for AMARES quantitation of a series of ^{31}P spectra of a human brain, capture of the starting values on non-linear parameters and saving in the starting value data-base.

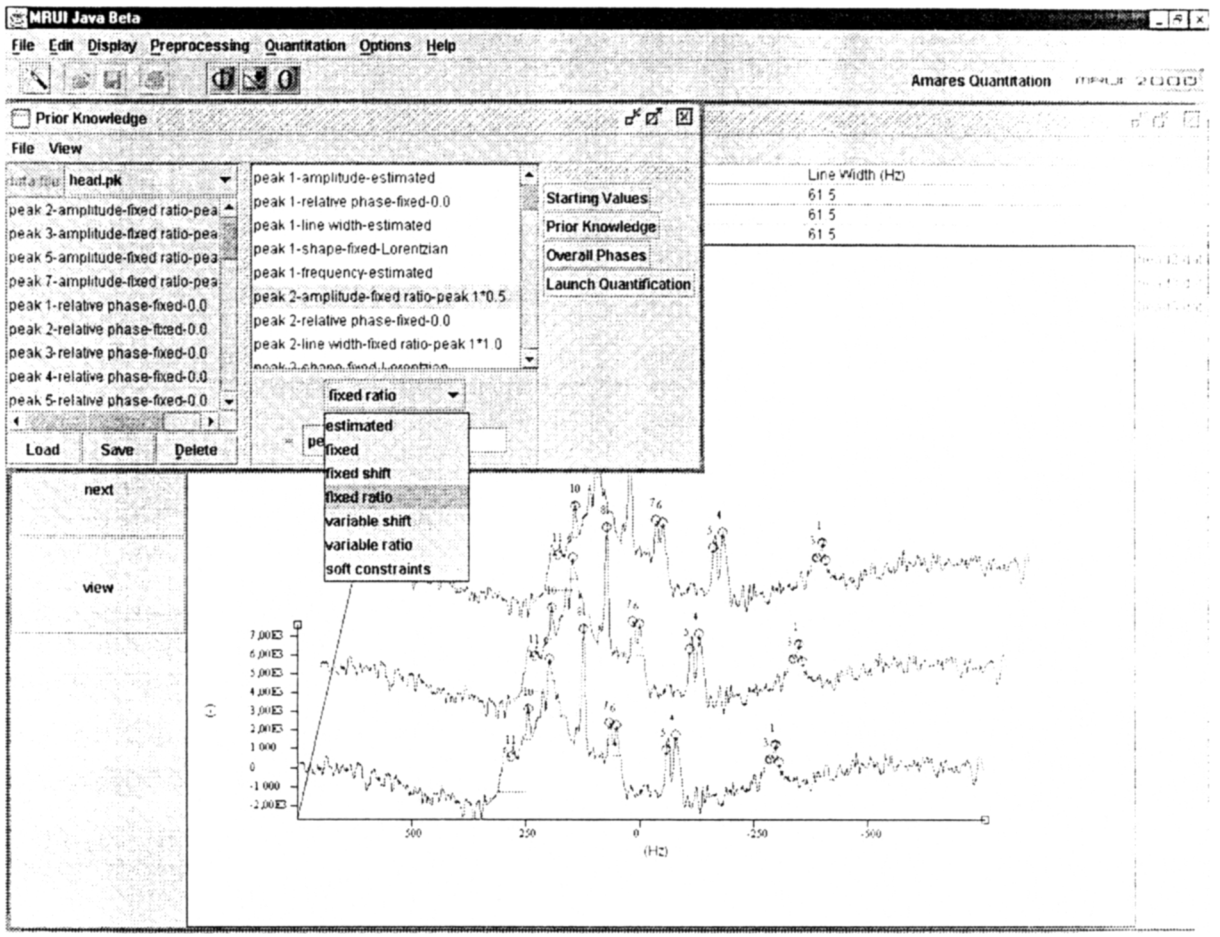

Fig. 4. Second step for AMARES quantitation of a series of ^{31}P spectra of a human brain, capture of the prior knowledge on spectral parameters and saving in the prior knowledge data-base.

The Java-GUI has been conceived for processing such large data-sets. Presently, many signals can be loaded simultaneously, but the quantition is still done sequentially. Very recently Vanhamme et al. [34] have shown that statistically better results are obtained when quantifying *simultaneously* all signals of time-series MRS data sets. The AMARES times-series method, described in [34] (see also [22]), will be embedded in a future version of the MRUI.

2.5. The `mrui.preprocessing` package

Within the MRUI package, preprocessing is defined as performing operations on MRS signals that change the values of the data points [3]. Examples are estimation of noise-free signals by applying the Cadzow enhancement procedure [17] or removing unwanted spectral features by applying SVD-based filtering [14,24,25]. These preprocessing steps are performed before the quantitation step.

The classes, developed for launching the various preprocessing methods, are forming part of the `mrui.preprocessing` Java package. The software offers the opportunity of manipulating CPU-intensive Cadzow preprocessing via the mechanism of adjusting thread priorities [41]. A last method to be mentioned, is the `histo` method. This method plays a role in visualizing the history of preprocessing steps on the MRUI `OutputToolBar` (see again Fig. 2).

A major preprocessing application of SVD-based methods is their use to remove dominant peaks such as the water signal in proton spectroscopy [24,25]. SVD-based methods allow a good fit of an arbitrarily damped sinusoid provided a sufficient number of singular vectors is taken into account. Although the fit thus obtained is mathematical rather than physical, it can still be very useful for removing, e.g. a water peak from a signal. Subtraction is in the time-domain, so broad wings that extend throughout large parts of the spectrum will disappear too, without affecting the amplitudes of the resonances of interest that may reside upon them. In Fig. 5, we show, as an example, the MRUI results obtained by applying Filter-HLSVD for removing the water components from a brain tumor proton signal.

2.6. The mrui.simulation package

For testing new preprocessing and quantitation algorithms, the usual way is to simulate realistic signals with *known* spectral parameters. This can be done either with or without including noise realizations. In the current Java-based MRUI, the simulation functionalities can be called via the classes of the mrui.simulation Java package.

2.7. The mrui.graph package

The graph package aims at providing.

- Signal displaying functionalities, displaying of spectra or signals (real/imaginary/absolute parts), of sets of signals (time-series) and zoom.
- A GUI for capturing values for preprocessing and quantitation methods (peaks characteristics, frequency interval selection, frequency domain shifting, etc).

In an ideal way, this package should be seen as a graphical client, pluggable on a data server.

2.8. Exports

Results are presently exported in the html format which is platform-independent and can be downloaded from any WWW server. Moreover, they can be stored on disk and loaded/reused further.

3. Examples

3.1. Quantitation in presence of overlapping broad background

The problem of quantitating peaks in the presence of overlapping broad background is crucial e.g. for proton spectroscopy when using short spin-echo times and for ^{31}P signals of brain. This point was addressed e.g. in [44] by comparing the performance of quantitation methods.

In Fig. 6, we show as an example the MRUI results obtained by quantitating with AMARES in vivo ^{31}P signals of a human brain. The measurements were

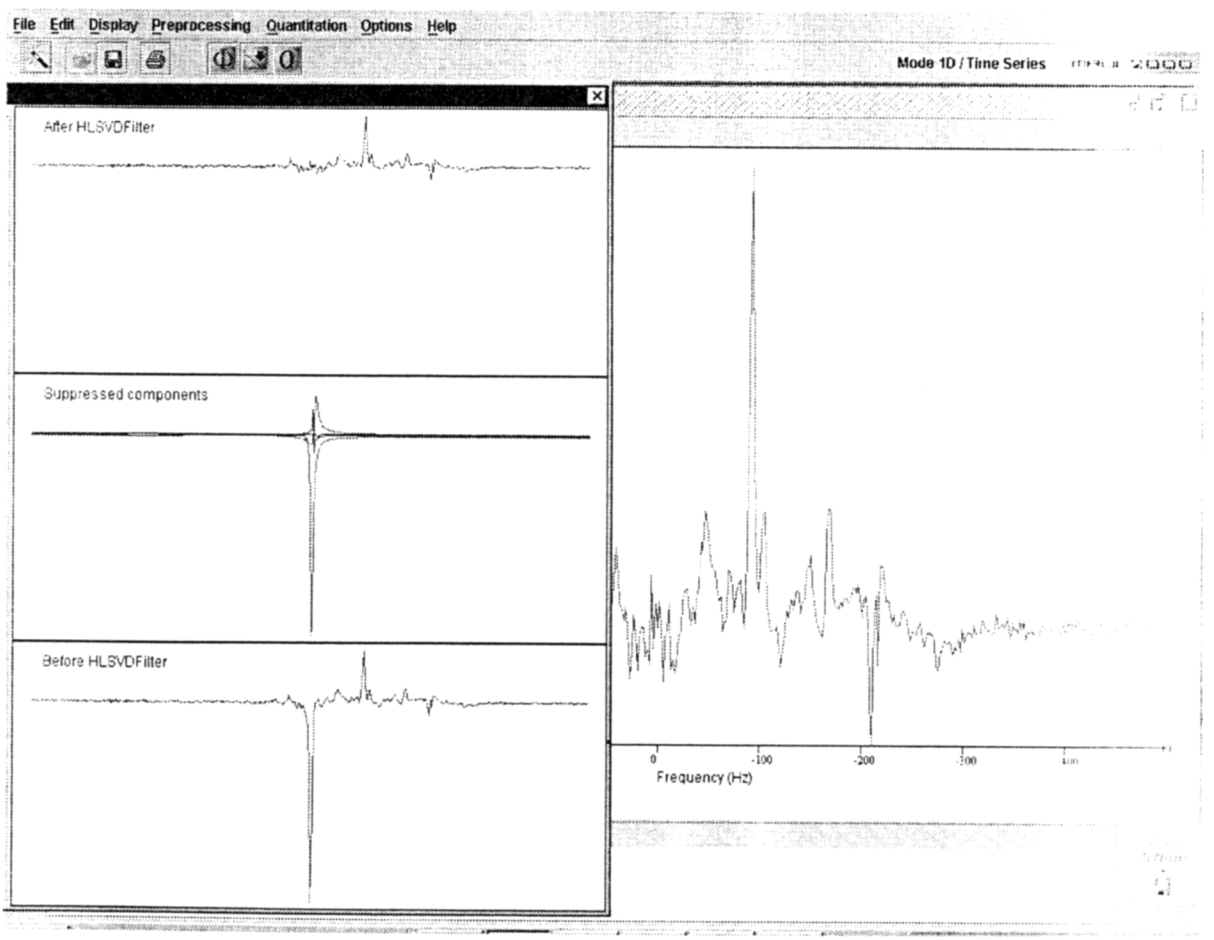

Fig. 5. The HLSVE-Filter window showing a brain tumor proton spectrum, before and after water suppression. This window is superposed on the 1-D mode window displaying the zoom in on the metabolite region.

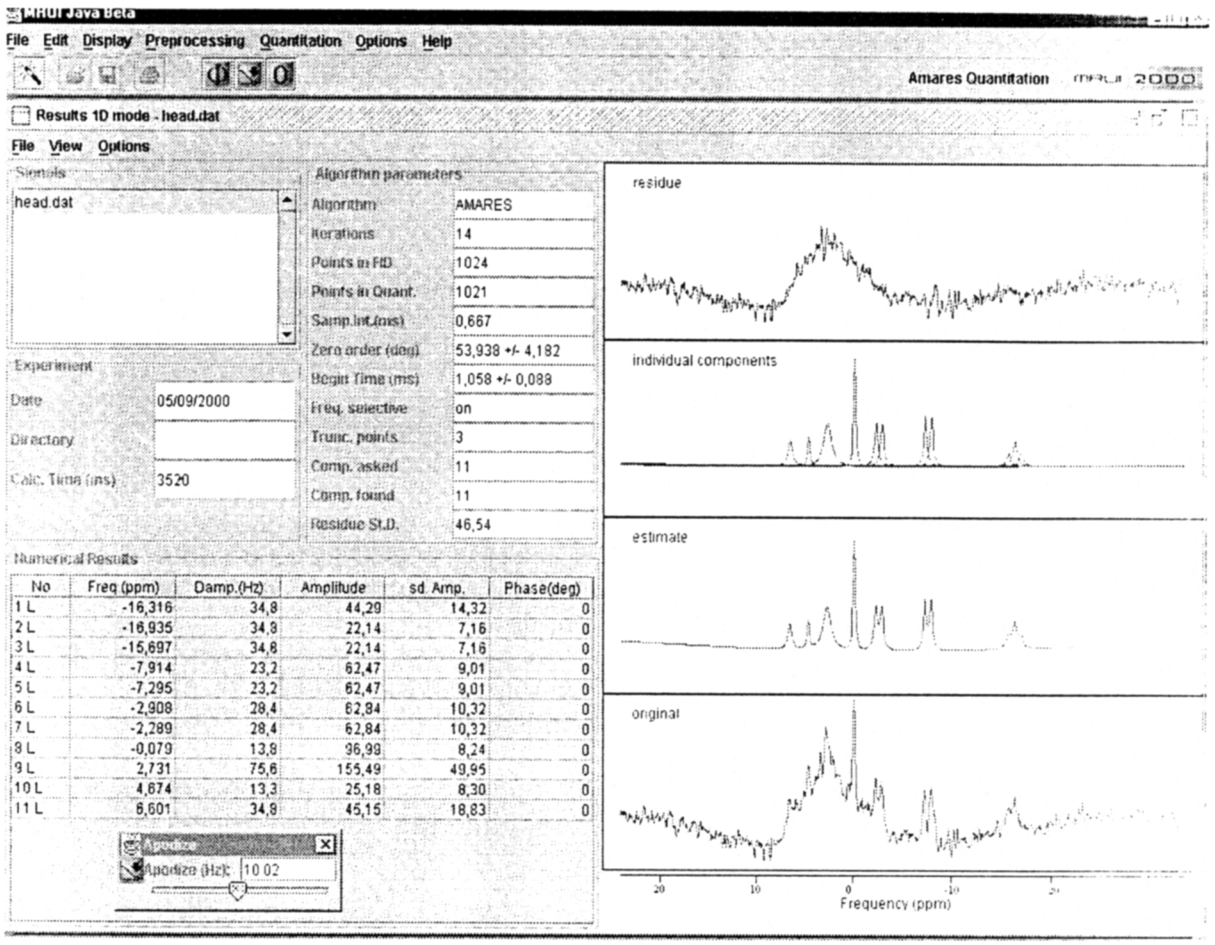

Fig. 6. MRUI results obtained by applying AMARES to a series of ^{31}P spectra of a human brain. From bottom to top, the experimental spectrum obtained after FFT of the acquired signal, the reconstructed spectrum obtained after FFT of the estimated signal, the spectrum of individual estimated peaks and then the spectrum of the residue.

performed at 1.5 Tesla. Four initial points were missing due to the presence of the phase-encoding gradient for the spatial localization. The raw spectrum exhibits a broad hump due to signals from less mobile molecules (phospholipids).

The following prior knowledge was imposed. (1) The ratios between the amplitudes of the peaks were 1:1 for the ATP doublets and 1:2:1 for the ATP triplet, assuming that the weak scalar coupling approximation is valid. (2) The damping factors of the peaks within a multiplet were kept equal. (3) The phases of all peaks were set to 0 relative to the estimated zero-order phase and the dead time of the receiver was estimated. (4) All scalar couplings were put equal to 16 Hz. Only the metabolite peaks were included in the fit and the hump was handled as follows. Since the hump is broad, the corresponding time-domain signal is strongly damped. To reduce its influence, we truncated three initial points and we invoked quarter-sine-wave weighting of the 20 following samples of both the measurement and the model function [45]. Since such weighting strongly reduces initial samples, strongly damped signals are attenuated correspondingly.

3.2. Application of SVD to peaks of unknown shape

It is often of interest to quantitate NMR peaks masked by spectral components of unknown line shapes. In those cases, the unwanted overlapping background features can be modeled in a mathematical way, as sums of exponentially decaying sinusoids. SVD-based methods [44,46] are quite well suited for this, since they do not require any starting values or prior knowledge.

In Fig. 7, we show the MRUI results obtained by quantitating a simulated MRS signal with HLSVD [14]. This complicated signal contains a small NMR peak of interest at 3.13 ppm which is completely masked by a large background peak with an amplitude 40 times larger, centered around 2.35 ppm. This background peak was deliberately given a deviating decay function, leading to a Lorentz-broadened triangle in the fre-

quency domain. HLSVD has approximated this triangle by a sum of Lorentzian peaks. Nevertheless, the estimated parameters of the peak of interest are not too poor, considering the strong overlap problem. For instance, the estimated amplitude is 1.4 whereas the true value is 1. Another proof that the HLSVD black box quantitation was successful, is the fact that the residue between the fitted model function and the simulated MRS signal contains only noise.

The computational time, actually needed for carrying out the HLSVD analysis of the simulated signal, is around one second on a Notebook, with a processor Intel Pentium III 600 MHz and 128 MB RAM, working with Windows 98. This time is comparable with that of the 'stand-alone Fortran 77 HLSVD program, when running on a Pentium II PC with Linux.

4. Implementation

4.1. Hardware and operating systems

We used multimedia PC's for developing and testing our MRUI software. As an example, we worked with a Gateway2000 GP7-500 XL Pentium III computer with 256 MB memory and an 8 GB hard disk. In addition, we used Windows NT 4.0 as the operating system of choice. For testing platform independency of the GUI part, we ran MRUI on Windows 95, Windows 98, Windows 2000, Linux and Unix on SUN (Solaris 2.6) and Silicon Graphics (Irix 6.3) servers.

4.2. Software

The following software tools/packages were used to facilitate the development of the MRUI GUI.

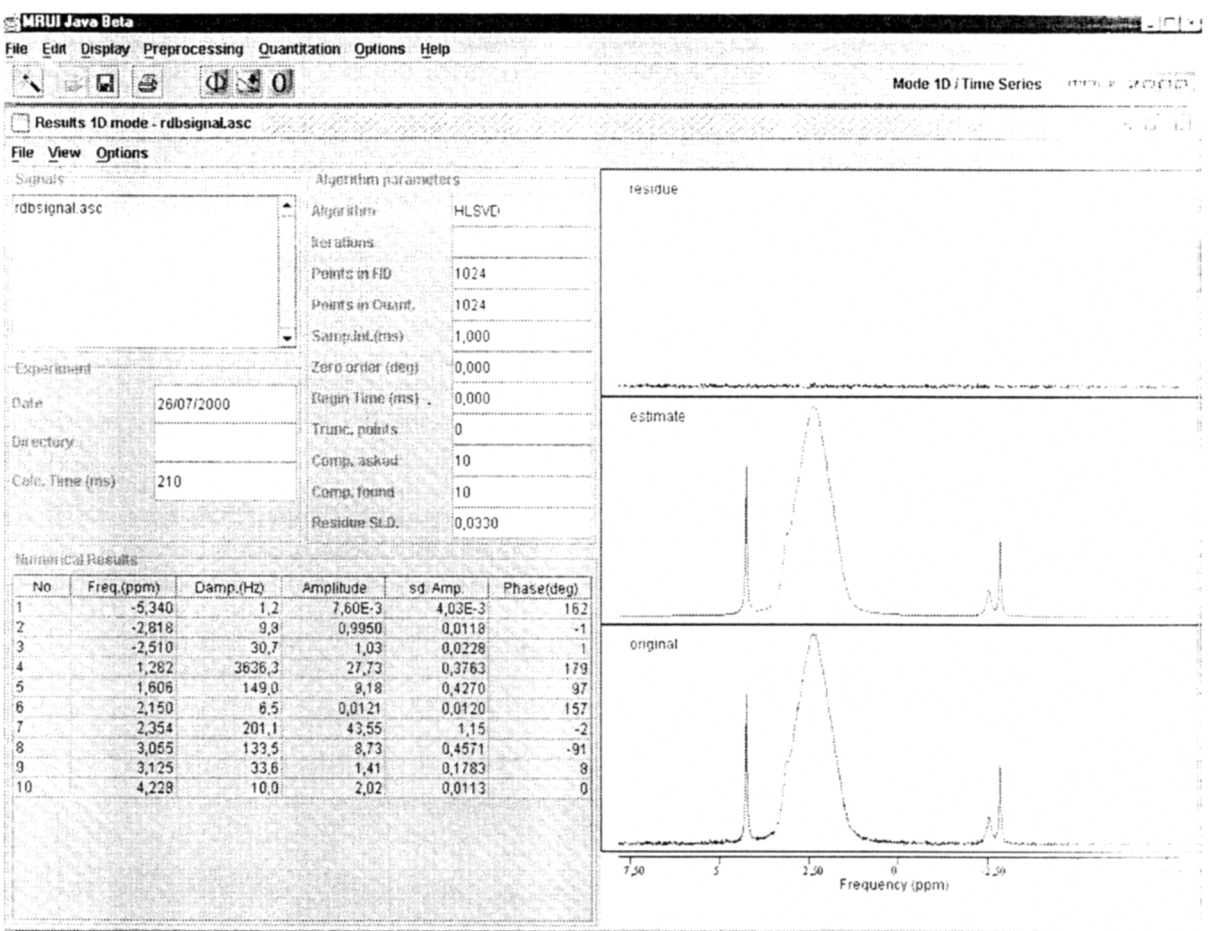

Fig. 7. MRUI result obtained by applying HLSVD to a simulated MRS signal. The large non-Lorentzian peak centered around 2.35 ppm is to be decomposed into a sum of exponentials in the related time-domain. The residue (top) between the filled model (middle) and the signal (bottom) contains only noise.

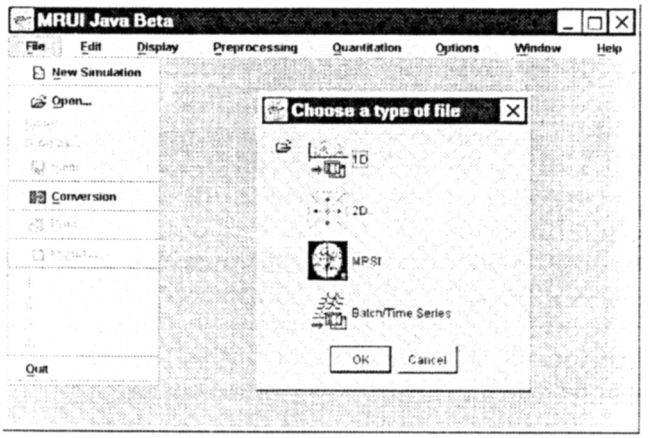

Fig. 8. The MRUI main window, as displayed via Windows 95. Shown are the file main menu and the child window for choosing the MRUI working mode.

- The Java development kit (JDK), this is the standard Java package provided by Sun Microsystems. A problem is that the Java software is still under development. The latest version of Java, we are now working with, is JDK version 1.3 Beta [47].
- Together/J, this is a visual UML modeling tool for the Java programming language. The Java-based applications are designed by drawing the various diagrams of the UML approach [39,40]. In addition, the Together/J program generates automatically the source codes of the corresponding Java classes. At present, we are working with Together/J version 3.0 [48].
- g77, This is the GNU Fortran compiler. We have used this compiler to build the Fortran 77-based libraries. At present, we are working with g77 version 2.91.57 [49].

4.3. Platform dependency

The Java-based MRUI software is only platform-dependent through calls to native code via the JNI [42], as is required for most of the quantitation and preprocessing methods. But, the GUI part of MRUI is in pure Java code and thus is really platform-independent.

In order to illustrate the latter, we have made screen captures of some MRUI GUI windows, when running on various platforms. For instance, in Fig. 8 the MRUI main window is shown, as captured via Windows 95. It also contains the child window for choosing the MRUI working mode. Fig. 9 depicts the same working mode window, this time captured via the Linux operating system. The same Java class bytecode can really be used on both computer platforms, although small differences are visible in the two graphical presentations.

Installation of the Java-based MRUI on the various computer platforms is easy. Once the libraries for the native (Fortran) code have been generated for the platform concerned, most of the work has been done. After that, the whole MRUI file system can be stored into an archive file. For instance, for the Windows 95/98/NT system the current version of MRUI can be stored into a ZIP file of about 4.3 MB. Extraction of the files on the local computer takes about 30 s. This yields a file system with about 6.4 MB of Java-related files and about 4 MB of Fortran-related dynamic linking libraries (DLLs). After extracting the MRUI files, the only other action to be done is installing the Java runtime environment 1.3 (JRE 1.3) from SUN. The Java-based MRUI software comprises now, approximately 62000 lines in Java, 43000 lines in C and 35000 lines in Fortran.

5. Conclusions

The new Java-based MRUI is a non-commercial software developed in the context of the European Union project *Advanced signal processing for Medical Magnetic Resonance Imaging and Spectroscopy* [5]. It is free for academics.

- The package is user-friendly and does not entail hidden costs (license costs, compilers) for the users. Presently, this new MRUI includes most algorithms of the current MATLAB version plus new functionalities, Gabor tools for peak extraction and dynamic phase correction, simultaneous quantitation of large data-sets, prior knowledge stored in a data-base.
- The work on the Java-based MRUI GUI confirms that the Java programming language is indeed well suited for developing highly interactive graphical software applications. Particularly the Swing part of the Java foundations classes [10] [11] appears to be a rich user interface class library.
- Through the JNI approach [42] we can embed code, generated in Fortran, into our software. To that end, an additional interface function written in ANSI C is to be used since the currently applied JNI does not support direct interfacing to Fortran.
- Analyzing UML class diagrams [39,40], produced by the Together/J visual UML modeling tool [48], is

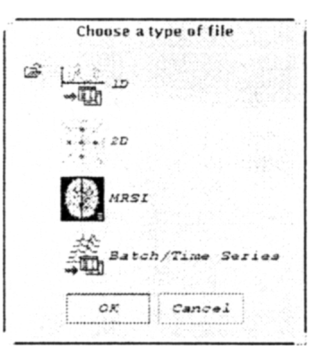

Fig. 9. The MRUI working mode window, as displayed via Linux.

helpful in better understanding and improving the development of systems like the MRUI GUI.
- Platform-independency of the Java class bytecode helps in simplifying the software installations on the various computer platforms.

Acknowledgements

The authors thank Dr S. Cavassila of the University of Lyon, Dr A. Coron of the University of Louvain la Neuve, Dr L. Vanhamme of the University of Leuven, and M. Cabanas of the University of Barcelona, for helpful discussions concerning the GUI. They acknowledge I. Castang, E. Jullo, D. Marquis, A. Zanetti for their contribution in developing the MRUI Java-based GUI. They are also strongly indebted to Dr D. van Ormondt, Coordinator of the TMR project mentioned below, to Professor J.P. Antoine, Dr E. Fotinea, Professor D. Michel, Professor P. Van Hecke, Professor S. Van Huffel and to all partners of this project. This work is supported by the European Union TMR/Networks programme, *Advanced Signal Processing for Medical Magnetic Resonance Imaging and Spectroscopy*, ERB-FMRX-CT97-0160, 1998–2001.

References

[1] The MRUI Home Page. See:http://www.mrui.uab.es/mrui/mruiHomePage.html.
[2] For all details on the MATLAB software, browse through The Mathworks Web site at:http://www.mathworks.com/.
[3] van den Boogaart A. MRUI manual v96.3. A User's Guide to the Magnetic Resonance User Interface Software Package, The European Union Human Capital and Mobility/Networks Programme, Project Number HCM-CHRX-CT94-0432, Advanced Signal Processing for Medical Magnetic Resonance Imaging and Spectroscopy, ISBN: 90-9010509-3, Delft, 1997.
[4] van den Boogaart A, Cavassila S, Vanhamme L, Totz J, Van Hecke P. A complete software package for MR signal processing. In: Eighteenth Annual International Conference of IEEE Engineering Medical and Biological Society, Amsterdam, 1996.
[5] The Home Page of the European Union project TMR/Networks ERB-FMRX-CT97-0160. See:http://azur.univ-lyon1.fr/TMR/tmr.html.
[6] Arnold K, Gosling J. The Java Programming Language. Reading: Addison Wesley, 1996.
[7] Chan P, Lee R, Kramer D. The Java Class Libraries, vol. 1, second ed. Reading: Addison Wesley, 1998.
[8] Chan P, Lee R. The Java Class Libraries, vol. 2, second ed. Reading: Addison Wesley, 1998.
[9] Flanagan D. Java in a Nutshell. Cambridge: O'Reilly, 1997.
[10] Weiner SR, Asbury S. Programming with JFC. New York: Wiley, 1998.
[11] Gutz S. Up to Speed with Swing. User Interfaces with Java Foundation Classes. Greenwich: Manning, 1998.
[12] Barkhuijsen H, de Beer R, van Ormondt D. Improved algorithm for noniterative time-domain model fitting to exponentially damped magnetic resonance signals. J Magn Reson 1987;73:553–7.
[13] de Beer R, van Ormondt D. Analysis of NMR data using time domain fitting. In: Diehl P, Fluck E, Gunther H, Kosfeld R, Seeling J, editors. NMR Basic Principles and Progress. Berlin: Springer, 1992:202–48.
[14] Pijnappel WWF, van den Boogaart A, de Beer R, van Ormondt D. SVD-based quantification of magnetic resonance signals. J Magn Reson 1992;97:122–34.
[15] Van Huffel S, Chen H, Decanniere C, Van Hecke P. Algorithm for time-domain NMR data fitting based on total least squares. J Magn Reson Ser A 1994;110:228–37.
[16] Barkhuijsen H, de Beer R, Bovée WMMJ, van Ormondt D. Retrieval of frequencies, amplitudes, damping factors and phases from time-domain signals using a linear least-squares procedure. J Magn Reson 1985;61:465–81.
[17] Diop A, Briguet A, Graveron-Demilly D. Automatic in vivo NMR data processing based on an enhancement procedure and linear prediction method. Magn Reson Med 1992;27:318–28.
[18] Diop A, Zaim-Wadghiri Y, Briguet A, Graveron-Demilly D. Improvements of quantitation by using the Cadzow enhancement procedure prior to any linear prediction method. J Magn Reson Ser B 1994;105:17–24.
[19] Kolbel W, Schafer H. Improvement and automation of the LPSVD algorithm by continuous regularization of the singular values. J Magn Reson 1992;100:598–603.
[20] Diop A, Kolbel W, Michel D, Briguet A, Graveron-Demilly D. Full automation of in vivo NMR quantification by LPSVD(CR) and EPLPSVD. J Magn Reson Ser B 1994;103:217–21.
[21] Totz J, van den Boogaart A, Van Huffel S, Graveron-Demilly D, Dologlou I, Heidler R, Michel D. The use of continuous regularization in the automated analysis of MRS time domain data. J Magn Reson 1997;124:400–9.
[22] Vanhamme L, van den Boogaart A, Van Huffel S. Improved method for accurate and efficient quantification of MRS data with use of prior knowledge. J Magn Reson 1997;129:35–43.
[23] van der Veen JWC, de Beer R, Luyten PR, van Ormondt D. Accurate quantification of in vivo ^{31}P-NMR signals using the variable projection method and prior knowledge. Magn Reson Med 1988;6:92–8.
[24] van den Boogaart A, van Ormondt D, Pijnappel W, de Beer R, Ala-Korpela M. Removal of the water resonance from ^{1}H-magnetic resonance spectra. In: McWhirter JG, editor. Mathematics in Signal Processing III. Oxford: Clarendon Press, 1994:175–95.
[25] Vanhamme L, Fierro RD, Van Huffel S, de Beer R. Fast removal of residual water in proton spectra. J Magn Reson 1998;132:197–203.
[26] Antoine JP, Coron A, Dereppe JM. Water peak suppression: time-frequency versus time-scale approach. J Magn Reson 2000;144:189–94.
[27] Cavassila S, Fenet B, van den Boogaart A, Rémy C, Briguet A, Graveron-Demilly D. ER-Filter: a preprocessing technique for frequency-selective time-domain analysis. Magn Reson Anal 1997;3:87–92.
[28] Barache D, Antoine JP, Dereppe JM. The continuous wavelet transform, an analysis tool for NMR spectroscopy. J Magn Reson 1997;128:1–11.
[29] Barkhuijsen H, de Beer R, van Ormondt D. Error theory for time-domain signal analysis with linear prediction and singular value decomposition. J Magn Reson 1986;67:371–5.
[30] Cavassila S, Deval S, Huegen C, van Ormondt D, Graveron-Demilly D. The beneficial influence of prior knowledge on the quantitation of in vivo magnetic resonance spectroscopy signals. Invest Radiol 1999;34:242–6.
[31] Cavassila S, Deval S, Huegen C, van Ormondt D, Graveron-Demilly D. Cramér-Rao bound expressions for parametric estimation of overlapping peaks. Influence of prior knowledge. J Magn Reson 2000;143:311–20.

[32] Sundin T, Vanhamme L, Van Hecke P, Dologlou I, Van Huffel S. Accurate quantification of 1H spectra: from finite impulse response filter design for solvent suppression to parameter estimation. J Magn Reson 1999;139:189–204.
[33] Vanhamme L, Sundin T, Van Hecke P, Van Huffel S, Pintelon R. Frequency-selective quantification of biomedical magnetic resonance spectroscopy data. J Magn Reson 2000;143:1–16.
[34] Vanhamme L, Van Huffel S, Van Hecke P, van Ormondt D. Time-domain quantification of series of biomedical magnetic resonance spectroscopy signals. J Magn Reson 1999;140:120–30.
[35] Graveron-Demilly D, Diop A, Briguet A, Fenet B. Product-operator algebra for strongly coupled spin systems. J Magn Reson Ser A 1993;101:233–9.
[36] Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W. Object-Oriented Modeling and Design. Englewood Cliffs: Prentice Hall, 1991.
[37] Booch G. Object-Oriented Analysis and Design. Redwood City: The Benjamin/Cummings Publishing Company, 1994.
[38] Stroustrup B. The C++ Programming Language, third ed. Reading: Addison Wesley, 1997.
[39] Fowler M, Scott K. UML Distilled. Applying the Standard Object Modeling Language. Reading: Addison Wesley, 1997.
[40] Ambler SW. The Unified Modeling Language v1.1 and Beyond: The Techniques of Object-Oriented Modeling. An AmbySoft Inc. White Paper. See:http://www.ambysoft.com/umlAndBeyond.html.
[41] Oaks S, Wong H. Java Threads. Cambridge: O'Reilly, 1997.
[42] Java Native Interface Specification. Sun Microsystems, 1997.
[43] Williams DM, Fencil L, Chenevert TL. Peripheral arterial occlusive disease: P-31 MR spectroscopy of calf muscle. Radiology 1990;175:381–5.
[44] De Beer R, van den Boogaart A, Cady E, Graveron-Demilly D, Knijn A, Langenberger KW, Lindon JC, Ohlhoff A, Serrai H, Wylezinska-Arridge M. Absolute metabolite quantification by in vivo NMR spectroscopy: V. Multicentre quantitative data analysis trial on the overlapping background problem. Magn Reson Imaging 1998;16:1127–37.
[45] Knijn A, de Beer R, van Ormondt D. Frequency-selective quantification in the time domain. J Magn Reson 1992;97:444–50.
[46] De Beer R, van Ormondt D. Background features in magnetic resonance signals: addressed by SVD-based state space modeling. Appl Magn Reson 1994;6:379–90.
[47] The Java Development Kit. See:http://java.sun.com/jdk/.
[48] The Together/J visual UML modeling tool. See:http://www.togethersoft.com/.
[49] The GNU Fortran compiler. See:http://egcs.cygnus.com/.