Journal of Biomolecular NMR, 8 (1996) 445–452
ESCOM

J-Bio NMR 387

Gifa V. 4: A complete package for NMR data set processing

Jean-Luc Pons, Thérèse E. Malliavin and Marc A. Delsuc*

Centre de Biochimie Structurale, INSERM U414, CNRS UMR 9955, Université Montpellier 1, Faculté de Pharmacie, 15 Avenue Charles Flahault, F-34060 Montpellier, France

Received 2 April 1996
Accepted 9 August 1996

Keywords: Linear prediction; FFT; Maximum entropy; Signal processing; MaxEnt; Software; Command language; Graphic user interface

Summary

The Gifa program is designed for processing, displaying and analysing 1D, 2D and 3D NMR data sets. It has been constructed in a modular fashion, based on three independent modules: a set of commands that perform all the basic processing operations such as apodisation functions, a complete set of Fourier transforms, phasing and baseline correction, peak-picking and line fitting, linear prediction and maximum entropy processing; a set of command language primitives that permit the execution of complex macro commands; and a set of graphic commands that permit to build a complete graphic user interface, allowing the user to interact easily with the program. We have tried to create a versatile program that can be easily extended according to the user's requirements and that is adapted to a novice as well as an experienced user. The program runs on any UNIX computer, with or without graphic display, in interactive or batch mode.

Introduction

The development of NMR spectroscopy in the past few years has been marked by a growth in the size of the data sets and the outburst of many new processing methods. The increase of computer system power and the actual capability of data storage have contributed to this evolution. Consequently, the modern spectroscopist requires NMR processing programs that have to meet some new features; programs have to adapt to different data set sizes and have to be extensible in order to follow the evolution of NMR spectroscopy methods. Moreover, they have to contain fast processing protocols and must be easy to use with the help of a comfortable graphic interface. All these aspects have been respected during the conception of the Gifa software. The previous versions of Gifa were designed essentially for classical as well as maximum entropy processing of 1D, 2D and 3D NMR data sets (Delsuc, 1989). The new version presented here includes some new processing tools but is mostly characterised by a complete rebuilding of the user interface, including the command language as well as the graphic interface. According to the requirements designed before, the Gifa V. 4 software permits the processing of data sets unlimited in size. A modular graphic interface and a powerful command language for additional developments have been made available.

Methods

The Gifa software has been written in the C and Fortran programming languages, using the X11R5 and Motif 1.2 graphic libraries. These choices make Gifa a portable software. Gifa runs on different UNIX computer systems such as Hewlett-Packard, Silicon Graphics, IBM, Linux-PC or Sun, with or without graphic display, in interactive or batch mode. During start-up, Gifa tests some hardware features (like the presence of an X-window connection) in order to determine the running style.

The Gifa program is set up as an interactive program, which gets commands from the user in an interactive manner and executes the corresponding actions. Some commands may prompt the user for additional parameters.

Basic data processing

The basic data processing commands are designed so as to correspond to elementary processing tasks, which

*To whom correspondence should be addressed.

can be easily chained. These commands apply to data held in the memory and provisions are made in the software to directly handle 1D, 2D and 3D data sets. Gifa includes all the usual basic processing functions for high-resolution NMR data processing. These functions concern Free Induction Decay (FID) preprocessing, spectral analysis of the FID and, finally, spectrum postprocessing and analysis. 1D, 2D and 3D data sets are handled in an equivalent manner by the software, and all the procedures presented in this section can be applied to 1D, 2D as well as 3D data sets.

Any apodisation function can be applied to the FID. The classical ones (sine, squared sine, exponential, Gaussian) are built-in commands of the program, and it is possible to design and apply any other apodisation function by using the Gifa command language.

A complete set of Fourier transforms (FT) (Delsuc et al., 1991; Delsuc, 1996) based on the fast Fourier transform algorithm is available. Direct and inverse FT are available for complex as well as real data sets. Direct and inverse Hilbert transforms (Zolnai et al., 1990) are also implemented. All kinds of 1D, 2D and 3D data sets (complex, real and hypercomplex) can thus be processed in a straightforward manner.

Phasing is possible on hypercomplex and complex spectra, but also on real spectra with the use of the Hilbert transform. Automatic phasing is implemented in Gifa by the APSL method (Heuer, 1991).

We should also mention several baseline correction modules. One is based on a spline fitting of the data, another is built from a statistical module for the separation of the baseline from a set of lines (Rouh et al., 1993). Any basic processing command can be applied independently on each spectral axis.

Spectral enhancement and analysis

A maximum entropy module (MaxEnt) (Delsuc, 1989; Delsuc et al., 1991; Robin et al., 1991) is present in the Gifa program. Several convergence algorithms are included: the original algorithm by Gull and Daniel (1978), with the possibility to use the correction from Wu (1984), a regular conjugated gradient, and an optimised fixed-point algorithm (Delsuc, 1989). Gifa actually stands for the acronym of 'Generalised Iterative Fixed-point Algorithm', the name of this optimised fixed-point algorithm. This module implements 1D and 2D deconvolution, multichannel deconvolution (Sibisi, 1983), and 1D and 2D J-deconvolution (Delsuc and Levy, 1988).

A comprehensive Linear Prediction (LP) (Malliavin et al., 1991) module, including the Burg (Ni and Scheraga, 1986), LP-SVD (Barkhuijsen et al., 1985) and forward and backward methods (Delsuc et al., 1987) is also available. Due to the very modular structure of this module, and the command approach of the program, Gifa permits the use of LP in many different situations, such as missing points reconstruction, baseline correction, spectral analysis, the Cadzow procedure (Cadzow, 1988; Brissac et al., 1995), water line suppression (Marion et al., 1989) and analysis of NOESY buildup curves (Malliavin et al., 1992; Reisdorf et al., 1992).

With the combined use of the extraction tool and the inverse Fourier transform, it is possible to apply the time-consuming techniques such as MaxEnt or LP on a small region of the spectra, thus minimising the computation time and the memory burden.

A simple peak-picker and integrator (Stoven et al., 1989) are available for 1D, 2D and 3D experiments. They are completed with a comprehensive line-fitter, which permits an independent fit of each peak parameter for Gaussian or Lorentzian lines. The line-fitter is built onto a Levenberg–Marquardt minimiser (Press et al., 1988).

All the information gathered by the peak-picker, the integrator or the line-fitter is stored into an internal peak table. Several commands are available for handling, displaying, modifying and storing this peak table.

Data management

Within Gifa, the working area is divided into three independent buffers devoted respectively to 1D, 2D and 3D processing. These buffers are statically allocated, and can handle only one data set at a time. Commands issued by the user relate to the current 'working' buffer, which can be selected and changed at any time by the user. The three buffers are contiguous in memory, thus permitting to join them when a larger data set must be handled. For instance, if no 3D processing has to be done, the 3D buffer can be joined to the 2D buffer to allow processing of a larger 2D data set. In the same way, parts of these buffers are allocated when operations needing large temporary buffers have to be performed.

Data handling has been designed to optimise the speed of processing. Many processing commands, like apodisation, FT, phasing or baseline correction, are thus realised in-place onto the working buffer. Indeed, this approach allows to minimise the memory needs and to optimise the use of the processor cache. However, some commands (e.g. projection commands, plane or row extraction) may deposit the result of their operation in another buffer than the working one.

Several other data buffers are also available to the user to store information or data independently of the main buffers: (i) there is an additional, multi-purpose buffer, which can hold a 1D, 2D or (small) 3D data set, and which can only be stored, retrieved and added. This multi-purpose buffer serves as an off-place location, which may be very useful in certain processing steps; (ii) the command language (see below) also provides the user with variables and arrays in which any kind of data type can be stored; and finally (iii) a support to the UNIX data-

```
; this macro takes two arguments :
;       - the index of a peak in the internal peak-table
;       - a list of 2D experiment names,
;               as a list of blank separated names, eg: "exp1 exp2 exp3"
; it generates a file called "result" which contains the exact coordinates
; of the peak and its integral value in each experiment

; first, the user interface part
if ($DIM != 2) error "Works  only in 2D"          ; check current state

message "Enter peak index"              ; message to the user
                                        ; user gets prompted with message line, only if
                                        ; no argument is available on the calling line

set index = $_                          ; $_ corresponds to the next argument
if ($index > $NBPK2D) \                 ; $NBPK2D is the number of peaks
   error "Index outside peak-table limits"          ; in the 2D peak-table

message "Enter list of experiment (within quotes)"
set file_list = $_

set out_file = "result"                 ; then sets-up the output file
open $out_file

; gets coordinates of selected peaks, $PK2D_FXF[] contains the peak coordinates,
; itop(index,dimension,axis) is the index_to_ppm convertion function
set ppmf1 = (itop($PK2D_F1F[$index],2,1))
set ppmf2 = (itop($PK2D_F2F[$index],2,2))
fprint $out_file ("Peak #"; $index; "at"; $ppmf1; $ppmf2)

; loops over file names
while ($file_list s! " ")               ; while there are experiments to process
  set file = (head($file_list))         ; head(string) gets the first word of 'string'
  set file_list = (tail($file_list))    ; tail(string) gets everything but the first one
  read $file                            ; read in data-set
  integ %%                              ; realizes the integration, using the current
                                        ; peak table and the default parameters
  fprint $out_file \
      ("experiment :"; $file; "intensity :";$PK2D_A[$index])
                                ; PK2D_A[] is the computed volume of the peak
endwhile

close $out_file                 ; close output file
exit                            ; and returns
```

Fig. 1. An example of a macro file, performing the integration of a given peak for a series of 2D experiments. This macro shows several programming constructions available in the control language, such as user variables, passing of arguments, access to internal parameter values, creation and access to an output file, string manipulation (head and tail commands) and control structures. Comments in the first part of the macro are used for automatic help generation. Note that the \ character marks a line continuation.

base format (DBM) is provided and will be described below in greater detail.

When the static memory buffers appear to be too small to handle a given data set, it is possible to work on a file rather than in memory. This is performed through a special cache memory system that is implemented into the Gifa program. The cache memory works by subdividing the data set into blocks or submatrices, which tile the data set. When accessing a part of the data set, only those blocks that actually contain information are loaded from disk into the cache memory. The cache memory manages to keep these blocks in memory in an optimal fashion, by keeping track of their usage. The least recently used blocks are discarded first. Enough memory is allocated in the cache memory for the blocks to insure at least room for the largest row for a 2D data set, and the largest plane in the case of a 3D data set. The blocks located in the cache memory are accessed through a hash table system. This set-up permits a very fast access to the data already stored in the cache memory, and an optimal access to the file, whether sequential access to the file is used or not. Some special commands permit the user to access regions of the data set, and to load rows or planes from the data set. A set of macros has been written in order to process data sets through the cache memory system, but keeping the regular in-memory syntax.

a

```
; This macro creates a form box which permits to modify the
; parameters of the contour-mode display window.
; The user can    - open / close the contour-mode display window
;                 - choose to display positive / negative / both sides
;                 - change the number and spacing of contour levels.

;Preset parameters:
if ($CDISP2D==1) then      ;$cdisp2d describes graphic states.
    set cdef = on
else
    set cdef = off
endif
if ($SIGN==1) then         ;Describes which levels are displayed
    set pndef = positive   ; (positive, negative, both)
elsif ($SIGN == 0) then
    set pndef = both
else
    set pndef = negative
endif
;Build the form
formbox "Display control" \
    "dispcont_doit $cont $posneg $lev $log" \          ;Callback
    "Contour Display (cdisp2d)" \
        enum "on,off" cont $cdef \                     ;Define fields
    "Display mode (sign)" \
        enum "positive,negative,both" posneg $pndef \
    "Number of levels (level)" int lev $LEVEL \
    "Level Algo : (loga)" real log $LOGA \
    "(x-1 : equidistant; x>1 : by power of x)" message \
    *                                                  ; * is end of list
exit
```

Fig. 2. Example of a set of macros that creates a form box with different editable fields, controlling the state of the contour-mode display window. (a) Text of the macro *dispcont*, which builds the control form box. After scanning the environment (described by the values of $CDISP2D and $SIGN) in order to set the default values for the variables, the form is built. The first argument of the command form box is the name of the window itself; the second argument is the callback associated with the action button, which here consists in calling the associated macro *dispcont_doit*; subsequent arguments define the different entries of the box. The fields are here of three kinds: pop-up menu for the selection in a list, typed field, and message field; other kinds of field are available (e.g., action button, text, or separator). (b) The associated *dispcont_doit* macro, which gets and tests arguments before setting the internal Gifa variables and refreshing the display. (c) The resulting form is a static graphic box with an action button 'Apply', with which the macro command *dispcont_doit* is associated.

The command language

A complete command language is embedded in the Gifa user interface in order to create macros, which can be used by the user exactly as native Gifa commands. All commands, as well as other macros, can be called during a macro execution.

There is support for all the structures needed for a full-fledged computer language. User variables are available and can be global or local. Variables are untyped; they can store indiscriminately any type of data: integers, reals, character strings or composite information. The data typing is forced by the kind of operation that is applied. Users can also build associative arrays (much in the manner that the UNIX program awk does). Associative arrays are very convenient for handling any kind of structured information, and can for instance be used to store and handle assignment information. Evaluation of mathematical, logical, and string expressions is possible. In addition to the set of operators and functions needed by a scientific programming language, there is an extended set of functions permitting to access internal parameters of the program (e.g., the size of the current data set, or the state of a graphic window), as well as to compute quantities 'natural' to NMR (e.g., unit conversion between ppm, Hz and spectral index). Several control structures are available for macro programming, such as loops

b

```
; macro dispcont_doit
; called by the form built with the dispcont macro

set cont = $_              ; Get arguments. $_ is for "next argument"
set posneg = $_
set lev = $_
set log = $_
;Process:
if ($cont s= "off") then
     cdisp2d 0
else
     if ($lev != $LEVEL) then
       if ($lev < 1 | $lev > 64) error "Wrong number of levels"
       level $lev
     endif
     if ($log != $LOGA) then
       if ($log < 1) error "Loga should be >1"
       loga $log
     endif
     cdisp2d 1
endif
if ($posneg s= "positive") then
     sign 1
elsif ($posneg s= "negative") then
     sign -1
else
     sign 0
endif
exit
```

c

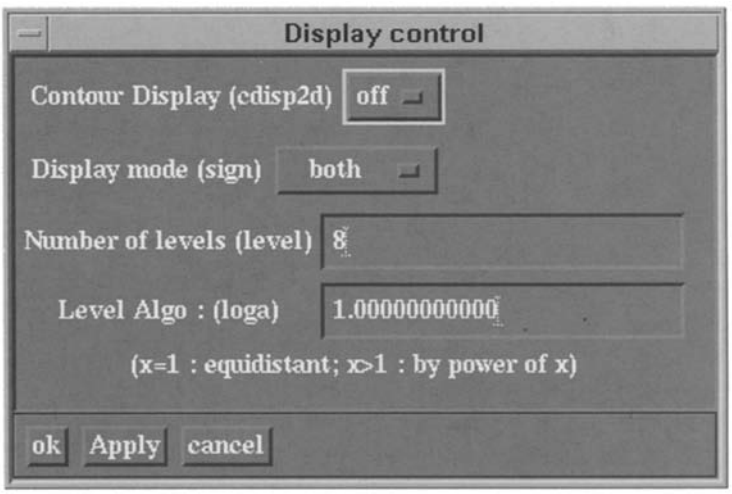

Fig. 2. (continued).

and tests. In order to give Gifa the capability of handling databases, a module has been implemented that permits to access DBM files. DBM is a standard format for flat databases, available in the UNIX OS. Access to DBM files can thus be achieved from any other programming environment. In Gifa, database entries are handled by binding them to pseudo-associative arrays, so there is no additional syntax needed. As an illustrative example, Fig. 1 presents a typical macro file using the main features of the command language.

TABLE 1
DIFFERENT PROCESSING TIMES OBSERVED USING THE Gifa PROGRAM

Processing protocol	Time (s)			
	SUN	IBM	SGI	HP
2D	17.5	12.6	10.5	7.5
3D	1064	329	193	172

The following machines were used: SUN: Sparc 10-30 with the SunOS 4.1.3 operating system; IBM: RS6000-560 with the AIX V3 operating system; SGI: Silicon Graphics R4000 at 175 MHz, with the IRIX 5.3 operating system; HP: Hewlett-Packard 735 at 99 MHz, with the HP-UX 9.01 operating system. Two processing protocols were used for these benchmarks. 2D is an in-memory processing of a 512×2048 2D data set, consisting of an exponential apodisation in t_2, followed by real Fourier transform, phase correction, reduction to real type and five-point spline baseline correction, and a sine-bell apodisation in t_1, followed by one level of zero-filling, real Fourier transform and phase correction. 3D is a $256 \times 64 \times 512$ 3D NMR data processing with sine-bell apodisation, zero-filling, complex Fourier transform, phasing and reduction to real type in all three dimensions, a linear baseline correction being applied on the F3 axis.

A graphic user interface

The Graphic User Interface (GUI) in an NMR processing program may be considered to be composed of two independent parts. The first part concerns the display of the NMR data set (FID and spectra), while the second is devoted to interaction with the user for commands and controls.

In Gifa, the content of the working buffer (1D, 2D or 3D) is displayed to the user by several graphic windows. Following the modular conception of Gifa, each graphical window presents only one object and all windows are optional. All windows are interactive: they can be moved, resized, closed and iconized. The user chooses to visualise 1D, 2D and/or 3D spectra with the help of specific commands. 2D data sets can be displayed either in bit-map mode or as a contour plot. 3D data sets can be viewed as series of 2D planes as well as real 3D contours. It is also possible to interact directly with the data displayed on the graphic. The user can scale data, zoom in on spectra, or shift to different parts of spectra with the help of the mouse or with a set of specific commands. The current zoomed window is always shown on a small and complete spectrum representation, displayed on the standard interface. The spectral coordinates in different units (index, ppm, Hz or s) may be easily displayed. It is also possible to add text or graphic annotations to any of the Gifa graphic windows. In addition to the interactive windows, any window can be duplicated at any time and its content 'frozen' in order to keep the display of the data at different stages of the processing. Finally, it is possible to superimpose different spectra on the same display.

The part of the GUI that permits the user to interact with the software has been designed to be modular and flexible. There is no predefined GUI, but rather, a possibility has been built into the command language to define elements of the interface. There are three primitive commands for creating menus, dialogue boxes and forms. Menus are composed of a series of buttons, each one associated with a command (or a series of commands) of the Gifa language. The dialogue boxes associate editable graphic fields with user variables, permitting extensive user interaction. Any change in the content of any editable field is readily echoed to the associated user variable. Provision is made for creating specific fields: a typed field (string, real, integer), associated with syntax checking; selection in a list, associated with pop-up menus; and file entry, associated with a standard file selection interface. Forms are static versions of dialogue boxes. They also present the user with a series of editable fields, but there is an additional action button associated with a Gifa command line, in which the values of the field can be used as parameters. Forms permit to build any kind of control boxes, which can even be resident on the screen. Figure 2 shows an example of such a form, designed for the control of the graphic display. Finally, there is also the possibility to create alert boxes and error boxes. It should be noted that this GUI is completely optional in Gifa, and that the program can be fully utilised in a non-graphic environment, and actually even in batch mode.

Additionally, any display can be plotted in HPGL or PostScript format. Commands permit to control the size and the rotation of the plot on the page. Commands are also available to add text, titles, annotations and axis labelling to the plots.

File input / output

The standard Gifa file format is divided into two different parts: the header and the data parts. The header holds all the parameters of the data in text format. It can be displayed, modified and expanded. The data area is divided into data blocks. It is possible to access any region of the data set with a minimal number of disk accesses.

One of the problems the modern NMR spectroscopist has to deal with is the transfer of NMR data sets between different computer programs. In this respect, several file formats are directly available for reading and some utilities exist to reformat files. Moreover, the data format specific to the Gifa software has been made public-domain, as has the source code of the library giving access to these data files.

Results and Discussion

Following the growth of multidimensional NMR spectroscopy, several NMR data processing programs have recently been developed, such as FELIX (Biosym Technologies Inc., San Diego, CA, U.S.A.), NMRView (Johnson and Blevins, 1994), PROSA (Güntert et al., 1992) and NMRPipe (Delaglio et al., 1995). Each one of these soft-

ware systems is widely used and appreciated for several well-developed and highly efficient specific tasks. We wish to show in this section that many Gifa features are very competitive with the other programs.

The Gifa data processing package is quite developed and includes all the basic processing tools as well as sophisticated spectral enhancement and analysis modules. Furthermore, the data access is optimised for all kinds of processed data. Indeed, the Gifa software has been designed in order to perform in two complementary modes. In contrast with most NMR programs that process data on disk files, the standard way of working on an NMR data set with Gifa is to hold all the information in memory. This method permits very fast processing of small everyday experiments. Nevertheless, the development of 2D and 3D experiments has increased the size of the available NMR information. On-file processing has become a requisite and the cache memory system is devised for this purpose. Its design permits one to minimise disk accesses, resulting in speed increase for processing of large files. Generally, whatever the data set size, Gifa is optimised and uses the faster algorithms available and the appropriate compilation options. A comparison of the processing speed on different computers is shown in Table 1.

Another advantage of the Gifa software is the versatile and programmable graphic user interface. As described above, all the graphic windows are optional, and Gifa is built in a modular way. This modularity permits the user to tailor the graphic environment and macro library to fit specific needs. The graphic interface and data processing part of Gifa can easily be expanded. The graphic primitives associated with the command language allow one to build a rich and versatile user interface or to extend it with new functionalities (additional buttons in order to run new commands, message boxes for the dialogue with the user, configuration forms). The start-up file can be modified according to user requirements.

In order to facilitate the first contact with the software, a default graphic environment has been written and is invoked during the start-up. For the novice user, this default graphic environment can be used in order to do most of the usual processing. It includes the basic command set for 1D, 2D and 3D processing, as well as support for displaying, plotting, peak-picking, integration, peak-fitting and assignment. Figure 3 shows the default graphic environment with 1D and 2D spectra, and the menus and zoom control boxes.

A high-level command language is rarely available in the other NMR processing packages. The Gifa command language is a structured programming language (with function calls), which gives access to shell commands and to high-level data structure organisation such as DBM

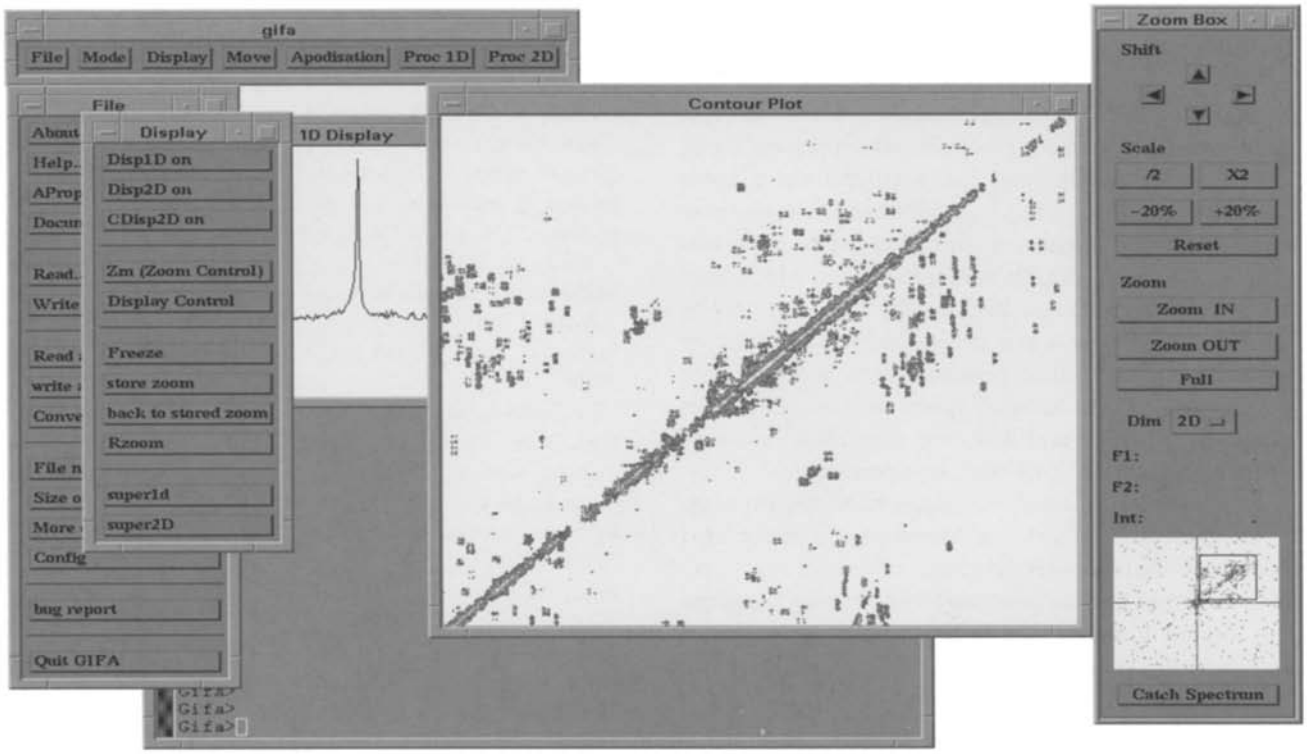

Fig. 3. The default graphic environment as invoked during start-up of Gifa. All the basic commands can be run from this graphic environment. The control box called 'Zoom Box' permits interaction with the graphic window. The data are currently displayed in a contour plot 2D window and in a 1D window. The small vignet seen in the 'Zoom Box' is a view of the complete data set held into the 2D buffer, the square in it indicates the current zoom window.

databases or associative arrays. It is possible to write commands and to extend the software by adding new data-processing functionalities. For instance, many tools such as the peak-picker and line-fitter graphic interface, the Cadzow procedure and the automatic phasing have been written in the Gifa command language. Another striking example of this extensibility is given by the assignment module, which uses all the functionalities of the macro language: the assignment information is stored hierarchically in DBM databases, and the user interacts with the assignment data through standard dialogue and form boxes written in the command language.

As a result of the modular conception of the program, there are different ways of interacting with the software. The user may choose between a graphic command interface (buttons, forms and dialogue boxes), a command line (prompt level) or a language interface (macros). Finally, according to the internal organisation, it is very well possible to run the program in batch mode when a long processing time is expected.

Some on-line help functionalities are available. Gifa includes explicit error messages, a complete on-line help function for every command and a comprehensive interactive manual, which explains and illustrates the features of the Gifa commands. A history file (log file) is available in which all the input generated by the user and all the output generated by the program are journalised. Additionally, a complete documentation has been written.

Conclusions

The Gifa NMR processing program provides the user with a comprehensive set of optimised processing tools. Furthermore, the command language and the graphic primitives provide Gifa with capabilities close to those found in an application generator. Because of these features, Gifa is a very flexible software system, which does not impose any work method on the user, but can be adapted to all purposes and can be successfully used by the novice as well as the experienced user. Moreover, the software can be easily extended to new processing methods not yet implemented and it is even possible to use it in domains external to NMR data set processing.

Because of the software techniques used, the program does not depend on any special hardware and can be used on most UNIX computer systems.

The program Gifa is now used by about 60 registered users, located in many different countries. It is possible to download the Gifa program through anonymous ftp (ftp://www.cbs.univ-montp1.fr). The program is free for academic laboratories, after signing a licence.

Acknowledgements

We would like to acknowledge all the people who have been involved in the development of Gifa since its very first version, most notably: M. Robin, A. Rouh, G. Salnikov, V. Stoven and C. van Heijenoort. We also thank F. Bontemps, M.A. Cremonini and F. Penin for giving access to their computers. We acknowledge the French M.E.N.E.S.R.I.P. for grant ACC-SV13, project 'infobiosud'.

References

Barkhuijsen, H., De Beer, R., Bovée, W.M.M.J. and Van Ormondt, D. (1985) *J. Magn. Reson.*, **61**, 465–481.

Brissac, C., Malliavin, T.E. and Delsuc, M.A. (1995) *J. Biomol. NMR*, **6**, 361–365.

Cadzow, J.A. (1988) *IEEE Trans. Acous. Speech Signal Proc.*, **36**, 49–62.

Delaglio, F., Grzesiek, S., Vuister, G., Zhu, G., Pfeifer, J. and Bax, A. (1995) *J. Biomol. NMR*, **5**, 277–295.

Delsuc, M.A., Ni, F. and Levy, G.C. (1987) *J. Magn. Reson.*, **73**, 548–552.

Delsuc, M.A. and Levy, G.C. (1988) *J. Magn. Reson.*, **76**, 306–315.

Delsuc, M.A. (1989) In *Maximum Entropy and Bayesian Methods, Cambridge 1988* (Ed., Skilling, J.), Kluwer, Dordrecht, The Netherlands, pp. 285–290.

Delsuc, M.A., Robin, M., Van Heijenoort, C., Reisdorf, C.B., Guittet, E. and Lallemand, J.Y. (1991) In *Computational Aspects of the Study of Biological Macromolecules by NMR Spectroscopy*, Vol. 225 (Eds. Hoch, J.C., Poulsen, F.M. and Redfield, C.), Plenum Press, New York, NY, U.S.A., pp. 163–174.

Delsuc, M.A. (1996) In *Signal Treatment and Signal Analysis in NMR*, Vol. 19 (Ed., Rutledge, D.N.), Elsevier, Amsterdam, The Netherlands, pp. 374–406.

Gull, S.F. and Daniell, G.J. (1978) *Nature*, **272**, 686–690.

Güntert, P., Dötsch, V., Wider, G. and Wüthrich, K. (1992) *J. Biomol. NMR*, **2**, 619–629.

Heuer, A. (1991) *J. Magn. Reson.*, **91**, 241–253.

Johnson, B. and Blevins, R. (1994) *J. Biomol. NMR*, **4**, 603–614.

Malliavin, T.E., Delsuc, M.A. and Lallemand, J.Y. (1991) *J. Magn. Reson.*, **94**, 630–634.

Malliavin, T.E., Delsuc, M.A. and Lallemand, J.Y. (1992) *J. Biomol. NMR*, **2**, 349–360.

Marion, D., Ikura, M. and Bax, A. (1989) *J. Magn. Reson.*, **84**, 425–430.

Ni, F. and Scheraga, H.A. (1986) *J. Magn. Reson.*, **70**, 506–511.

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (1988) *Numerical Recipes in C, the Art of Scientific Computing*, Cambridge University Press, New York, NY, U.S.A., pp. 523–528.

Reisdorf, C., Malliavin, T.E. and Delsuc, M.A. (1992) *Biochimie*, **74**, 809–813.

Robin, M., Delsuc, M.A., Guittet, E. and Lallemand, J.Y. (1991) *J. Magn. Reson.*, **92**, 645–650.

Rouh, A., Delsuc, M.A., Bertrand, G. and Lallemand, J.Y. (1993) *J. Magn. Reson.*, **A102**, 357–359.

Sibisi, S. (1983) *Nature*, **301**, 134–136.

Stoven, V., Mikou, A., Piveteau, D., Guittet, E. and Lallemand, J.-Y. (1989) *J. Magn. Reson.*, **82**, 163–168.

Wu, N.I. (1984) *Astron Astrophys.*, **139**, 555–557.

Zolnai, Z., Macura, S. and Markley, J. (1990) *J. Magn. Reson.*, **89**, 94–101.