```
******************************************************************************
******************************************************************************
```

VnmrJ Tips and Tricks

```
******************************************************************************
******************************************************************************
```

Last update: 2013-07-11

```
------------------------------------------------------------------------------
```

NOTE: This collection of articles from Agilent MR News focuses on the aspects of setting up and USING the VnmrJ software. Related issues are discussed in other documents in this directory:

```
    faq/convert_data        Data Conversion from/for 3rd Party Software
    faq/c_pulse_seq         C Pulse Sequence Programming, Practical Hints
    faq/data_export         Exporting VnmrJ / VNMR Data
    faq/gradshim            Hints on Gradient Shimming
    faq/linux_install       Linux Setup and Installation - Tips & Tricks
    faq/linux_tips          Linux Usage - Tips & Tricks
    faq/plot_export         Exporting and Annotating Plot Data
    faq/printer_plotter     Printer / Plotter Issues
    faq/solaris_install     Installing and Setting Up Solaris
    faq/vnmr_arrays         Handling Arrays in VNMR / VnmrJ
    faq/vnmr_integration    Integration in VnmrJ / VNMR
    faq/vnmr_parameters     VnmrJ / VNMR Parameter Handling
    faq/vnmr_referencing    Spectral Referencing in VnmrJ and VNMR
```

```
------------------------------------------------------------------------------
```

List of Titles in the Sequence of Appearance in the Section Below:
==

==

2002-12-16:

A NOTE ON EDITING Tcl-dg AND VnmrJ PARAMETER PANELS:

VNMR Tcl-dg templates and VnmrJ panel layout definitions fall into two classes: pulse sequence specific layouts and a set of "generic" templates. The latter are stored in separate directories ("/vnmr/templates/layout/default" and "/vnmr/user_templates/dg/default"), while the former are stored in sequence-specific entries in the directories "/vnmr/templates/layout" and "/vnmr/user_templates/dg" (Chempack and ProteinPack use additional directories holding generic templates for nD experiments).

Upon editing, you can specify (in the bottom of the editing window) whether to save the Tcl-dg template or VnmrJ panel layout as sequence-specific or as "default". In most cases, users will usually modify sequence layouts, hence the pre-selected target for saving such templates is "sequence specific", i.e., the name of the pulse sequence (rather than "default").

Of course, it is also possible to edit "generic" templates, e.g., if you want to remove certain options (such as specific experiments) from Chempack panels. In this case you should be careful NOT to save the template under the name of the pulse sequence - the result would be erratic behavior (the generic panels would alter with specific pulse sequences). It is easy to forget selecting "default" prior to saving!

[Agilent MR News 2002-12-16]

2003-06-28:

REBUILDING THE LOCATOR DATABASE IN VnmrJ (by Bayard Fetler, Varian):

A user moved some shim files that were registered in VnmrJ's Locator database. Subsequently the database appeared to be corrupted, and the user tried repairing it using "managedb" several times, with various arguments - with the result that the database got corrupted even more.

Users should NOT call "managedb" - you should rather run "dbsetup" - this calls "managedb" with the proper arguments and in proper order - this should make the database functional again by rebuilding it. In the case of the shim files you will also need to select "Update Locator" -> "Update shims" from the "Utilities" menu.

[Agilent MR News 2003-06-28]

2003-07-21:

HINT FOR DESIGNING PARAMETER PANELS IN VnmrJ:

Mis-designed VnmrJ parameter panels can cause VnmrJ to keep the CPU busy to 100% due to an infinite loop. One trap that users may fall into is to SET a parameter in the "Value of item" field of a parameter entry. A "legal" entry for this field would be

```
        $VALUE=nt
```

or

```
        vnmrunits('get','sw'):$VALUE
```

In both cases, only the local variable "$VALUE" is set, and the macro "vnmrunits" does not alter the value of any parameter if the first argument is "'get'", i.e., the value of the VNMR parameters "nt" and "sw" is NOT set in the above fields. "$VALUE=nt nt=nt" (an attempt to cause a related "underscore macro" to be executed) would set the value of "nt", resulting in an infinite loop, and should be avoided. The value of the VNMR parameter must be set ONLY through the field "Vnmr command" which in the above cases would read

```
        nt=$VALUE
```

or

```
        vnmrunits('set','sw',$VALUE)
```

In other words: you MUST AVOID setting a parameter in the "Value of item" field, or calling a macro that alters a parameter value (it's best to avoid using macro calls other than "vnmrunits('get',...)" in this field!).

To explain this in more detail: The field "Vnmr variables" lists VNMR parameters that are used to decide when to update the displayed VnmrJ panel. If the field "Value of Item" SETS the specified VNMR parameter, this triggers a redisplay of the parameter panel, which causes the "Value of Item" string to be executed, which sets the parameter again, etc.

To check whether you have a "loopy VnmrJ panel", you can use one of the following methods:

- open the CPU performance meter, then display the parameter panel which you just designed or modified, and wait for about 10 seconds. If during that time the performance meter stays at 100%, and if it returns to normal levels once you switch to a different panel, you may have a loop in the panel definition.
- Another method is to display a 1D spectrum after opening the relevant panel. A loopy parameter panel often causes such a 1D spectrum to flicker or flash.

Varian supplied parameter panels in VnmrJ avoid this condition and can be used as models for building your own panels.

[Agilent MR News 2003-07-21]

2003-09-20:

VnmrJ SLOWING TO A CRAWL? CLEAN UP THE DATABASE AND BE BACK IN BUSINESS!
(by Maj Hedehus and Glenn Sullivan, Varian)

It has come to our attention that VnmrJ in certain circumstances appears to become increasingly slow, until it is finally at a point where it is unusable. This has been tracked down to a bug in the database management when a large number of files are present (most frequently observed on imaging systems). Even if you are careful about keeping the number of entries in the database to a minimum, this is not always enough to keep it from slowing down.

It turns out that when things are removed from the database, they are just tagged for removal, but not actually removed. Thus, the database manager ends up hogging all the system resources.

As a workaround, we suggest putting the UNIX command

```
        vacuumdb vnmr --analyze
```

(notice the double hyphen) into a cron job and run it on a regular basis, such as every night or once a week, depending on the typical size of your database. To perform this job once every weekday, at 1:30 a.m., type

```
        setenv EDITOR dtpad
        crontab -e
```

(if that fails or if "EDITOR" is not defined, "crontab" may use the default editor, "ed"; exit from "ed" with "q<Return>" and change "dtpad" to "vi" in the commands above), then add the line

```
        30 1 * * 1-5 /vnmr/pgsql/bin/vacuumdb vnmr --analyze >/dev/null
```

For more information on "cron" and "crontab" see "man crontab".

The "vacuumdb" command removes data that had been tagged and cleans up the database, and it reorganizes it back to a nice clean structure. The "analyze" option analyzes the database and saves some information that allows for faster searches.

The next release version of VnmrJ will do this automatically. Thus, after installation of the next version, the above "vacuumdb" command should be removed from the user's "crontab" file: use "crontab -e" again as shown above, and remove the line containing "vacuumdb".

[Agilent MR News 2003-09-20]

2003-10-17:

TIME STAMPS IN VnmrJ:

In Agilent MR News 2003-10-05 we proposed a macro solution for adding a new "timestamp" parameter in VNMR, which would record the time at which an experiment is submitted to the acquisition queue. In VnmrJ, you don't need to change any macro - VnmrJ already comes with several time stamping parameters:

- "time_submitted" records the time when the experiment is submitted to the acquisition queue, i.e., the time when "go" / "ga" / "au" is typed.
- "time_run" records the time when the acquisition starts (this can be much later than "time_submitted" if there are other experiments in the queue).
- "time_complete" records the time when the acquisition finishes.
- "time_saved" records the time when data are saved ("svf"/"svp").
- "time_processed" records the time when "ft"/"wft"/"ft2d"/"wft2d"/"ft3d" are run (when the command finishes for 1D and 2D processing, when the command is called for "ft3d", as that runs in background, asynchronously).
- "time_plotted" records the time when the "page" command is called, i.e., when the plot is submitted to the printer spooling software.

The values of these parameters have the format

```
        yyyymmddTHHMMSS
```

e.g.,

```
        20011102T105551
```

which is one of the combined date/time formats offered by the ISO-8601 standard (the other option would have been "yyyy-mm-ddTHH:MM:SS", such as "2001-11-02T10:55:51" for the example above), see also

```
    http://www.mcs.vuw.ac.nz/technical/software/SGML/doc/iso8601/ISO8601.html
```

The "T" character acts as a separator between the date and the time. One advantage the VnmrJ time stamp parameters offer over the numeric "timestamp" parameter proposed in Agilent MR News 2003-10-05 is that they are strings, i.e.,

```
        time_complete?
```

entered from the command line displays the full information.

The VnmrJ Locator also displays a value "time_created", which is NOT a VnmrJ parameter, but rather a UNIX time stamp, as shown by "ls -l". And of course, there is still the date parameter, which contains the date when the data were acquired. Besides "time_created", the Locator also offers "time_run" and "time_saved" as sorting / selection criteria. Note that the Locator shows the date and the time using the extended ISO-8601 formats: "yyyy-mm-dd HH:MM:SS".

VnmrJ for Secure Environments was mentioned in Agilent MR News 2003-10-05: what this adds to the above is a mechanism that keeps a secure track of when a file has been created (and by whom, etc.), as well as of any modification to the file (remember that "cp" / "cp -r" still alter the UNIX time stamp, and VNMR / VnmrJ offer various means for altering parameter values in saved data).

Thanks to Bayard Fetler (Varian, Palo Alto) for providing valuable input for this article!

[Agilent MR News 2003-10-17]

AUTONAME DATE / TIME FORMATS IN VnmrJ (by Bayard Fetler, Varian):

VnmrJ offers some new date / time formatting options for the "Svfname" and "autoname" parameters, namely "%DATE%", "%TIME%", "%YR%", "%MO%", "%DAY%", "%HR%", "%MIN%", and "%SEC%". The values that these placeholder strings represent are determined from the UNIX time, at the time when "autoname" or the "Svfname" command is called. If you have loaded VnmrJ 1.1C, these new features are of course also available within "VNMR classic", i.e., the traditional VNMR interface that can be launched by typing "vnmr" in a shell window.

[Agilent MR News 2003-10-17]

COMPARING DATES IN VnmrJ:

In Agilent MR News 2003-10-05 we presented a mechanism for logging the experiment submission time (i.e., the time when the experiment is queued with "go", "ga", or "au") in a VNMR parameter, and for then using that information to compare the submission dates of two data sets in a macro.
As the VnmrJ time stamping parameters have a slightly different format, the macro presented in Agilent MR News 2003-10-05 requires some adjustments. Fortunately, these VnmrJ time stamp values are ALMOST ready for numeric comparison. You could expand the construct that we presented for VNMR, e.g.:

```
        $date1=0 $date2=0
        rt($file1)
        exists('time_complete','parameter'):$e
        if $e then
          shell('(echo',time_complete,'| tr "T" ".")'):$date1
        else
          exists('timestamp','parameter'):$e
          if $e then
            $date1=timestamp
          endif
        endif
        rt($file2)
        exists('time_complete','parameter'):$e
        if $e then
          shell('(echo',time_complete,'| tr "T" ".")'):$date2
        else
          exists('timestamp','parameter'):$e
          if $e then
            $date2=timestamp
          endif
        endif
        if $date1>$date2 then
          echo('file %s is more recent than file %s',$file1,$file2)
        elseif $date1<$date2 then
          echo('file %s is older than file %s',$file1,$file2)
        else
          echo('VnmrJ acquisition time stamps not available')
        endif
```

This would now cover data acquired under VnmrJ, with VNMR (and the "go" modification that we proposed in Agilent MR News 2003-10-05), as well as legacy data without parameter time stamps.

[Agilent MR News 2003-10-17]

2004-03-07:

SWITCHING BETWEEN VERSIONS OF VnmrJ (by Frits Vosman, Varian):

There are some instances where users would like to switch between two VNMR or VnmrJ versions. These situations could be:

- you want to switch from the current version to a backup ("vanilla") version of the software - for an example on how to create such a backup copy see Agilent MR News 2003-01-25;
- you want to switch from VnmrJ 1.1C to the beta test version of VnmrJ 1.1D or vice versa.

In this article we primarily want to discuss the steps that are required for switching between two VnmrJ or VNMR installations. For the instructions below we assume that you have an existing VnmrJ 1.1C installation located in "/space/VnmrJ1.1C", and you now (will) have installed the beta test version of VnmrJ 1.1D in "/space/VnmrJ1.1D_beta". If VnmrJ 1.1D beta is the active version, there is a link "/vnmr" pointing to "/space/VnmrJ1.1D_beta". To see how this link is set up type

```
        ls -l /vnmr
```

The instructions below also apply for switching between two versions of VNMR, except that the steps related to the VnmrJ database ("pgsql" / Postgres database / Locator) don't apply to VNMR - this is indicated in the paragraphs below.
Before switching software, exit any running copies of VnmrJ / VNMR. On a spectrometer the first step in switching is to stop the acquisition process ("Expproc" and the related processes that are controlled by "Expproc", or "Acqproc" in the case of older spectrometers):

```
        su acqproc
```

This should report that "Expproc" (or "Acqproc") has been stopped. This step is of course not required on stand-alone workstations.
  For VnmrJ (not for VNMR), you then need to stop the Postgres SQL ("pgsql") database daemon: as root or as VnmrJ master user (usually vnmr1) call

```
        /vnmr/bin/S99pgsql stop
```

You should see an echo

```
        waiting for postgres_daemon to shut down......done
        postgres_daemon successfully shut down
```

Now, adjust the link "/vnmr" to the new target software directory - this must be done as root:

```
        rm /vnmr
        ln -s /space/VnmrJ1.1C /vnmr
```

On spectrometers, the software that resides in the acquisition CPU may be version-dependent; you should therefore do the following steps as root:

```
        /vnmr/bin/setacq
```

which will prompt you to reboot the console (i.e., the acquisition computer). Newer versions of "setacq" automatically start "Expproc", which will be indicated to the user. If this is NOT the case, restart the acquisition process now, with

```
        su acqproc
```

With VnmrJ, you now need to restart the "pgsql" daemon; this should be done as the VnmrJ administrator - other users (except for root) will be asked for the VnmrJ administrator password:

```
        /vnmr/bin/S99pgsql start
```

You should see the feedback

```
        starting postgres_daemon, owner=vnmr1
        postgres_daemon successfully started
```

Note that files that are added will only show up in the database / Locator for the CURRENT version of VnmrJ: If you acquire data under VnmrJ 1.1D beta and then switch back to VnmrJ 1.1C, the Locator in the latter software will not automatically show the files acquired with the beta test version. There are two ways to correct this situation:

- As the VnmrJ administrator run

  ```
        dbupdate once 0
  ```

  This updates the database and may take a couple minutes to complete.
- Any user can select the desired action from the VnmrJ menu "Utilities" -> "Update Locator" in order to update the database, i.e., add the missing files.

A summary recipe for six typical situations is given below. All steps are to be performed as root, except where indicated otherwise; remember that the VNMR or VnmrJ directory paths may differ on your systems. In all cases you must of course first exit VNMR / VnmrJ.

- switching between versions of VnmrJ:
  - to switch from VnmrJ 1.1D beta back to VnmrJ 1.1C on a SPECTROMETER, use

```
        su acqproc
        /vnmr/bin/S99pgsql stop
        rm /vnmr
        ln -s /space/VnmrJ1.1C /vnmr    [ use correct new directory path ]
        /vnmr/bin/setacq                [ reboot the acquisition computer ]
        su acqproc                      [ if not done already by "setacq" ]
        /vnmr/bin/S99pgsql start
        dbupdate once 0                 [ as VnmrJ administrator ]
```

  - to switch from VnmrJ 1.1D beta back to VnmrJ 1.1C on a DATASTATION, use

```
        /vnmr/bin/S99pgsql stop
        rm /vnmr
        ln -s /space/VnmrJ1.1C /vnmr    [ use correct new directory path ]
        /vnmr/bin/S99pgsql start
        dbupdate once 0                 [ as VnmrJ administrator ]
```

- switching between two versions of VNMR:
  - to switch from VNMR 6.1C to a backup version of the same software on a SPECTROMETER, use

```
        su acqproc
        rm /vnmr
        ln -s /space/vnmr6.1C_bk /vnmr  [ use correct new directory path ]
        /vnmr/bin/setacq                [ reboot the acquisition computer ]
        su acqproc                      [ if not done already by "setacq" ]
```

  - to switch from VNMR 6.1C to a backup version of the same software on a DATASTATION, use

```
        rm /vnmr
        ln -s /space/vnmr6.1C_bk /vnmr  [ use correct new directory path ]
```

```
 - switching from VnmrJ back to VNMR and vice versa:
    - to switch from VnmrJ 1.1D beta back to VNMR 6.1C on a SPECTROMETER, use
        su acqproc
        /vnmr/bin/S99pgsql stop
        rm /vnmr
        ln -s /space/vnmr6.1C /vnmr     [ use correct new directory path ]
        /vnmr/bin/setacq                [ reboot the acquisition computer ]
        su acqproc
    - to switch from VNMR 6.1C to a previously installed VnmrJ 1.1C on a
      SPECTROMETER, use
        su acqproc
        rm /vnmr
        ln -s /space/VnmrJ1.1C /vnmr    [ use correct new directory path ]
        /vnmr/bin/setacq                [ reboot the acquisition computer ]
        su acqproc                      [ if not done already by "setacq" ]
        /vnmr/bin/S99pgsql start
        dbupdate once 0                 [ as VnmrJ administrator ]
AMENDMENT: In VnmrJ 1.1D and later, the command
        dbupdate once 0
should be replaced by
        managedb update
(the "dbupdate" utility is obsolete and will be deleted from future versions
of VnmrJ).
                                                [ Agilent MR News 2004-03-07 ]
```

2004-11-02:

DESIGNING PANELS IN VnmrJ 1.1D:

If you are designing panels (layouts) for VnmrJ, you may occasionally run into "missing features" in the panel editor. For example: one of the editor's colleagues recently found that the panel editor apparently lacks control over the "border style" for groups, i.e., one cannot freely select the border style with grouped panel elements within the panel editor, the temptation is there to bypass this limitation by editing the associated XML panel definition file in "/vnmr/templates/layout".
Before reporting this as a "bug" and before starting to edit an XML file you should consider the possibility that this omission is intentional, especially if a feature was available in earlier VnmrJ releases and has now been removed. The above is indeed such an example: in early versions of VnmrJ we hadn't quite settled on a "style guide" for parameter layouts, we were primarily interested in exploring the possibilities. In the early planning stages for VnmrJ 1.1D we were reviewing what the various "panel development groups" (such as the imaging group, or George Gray with his BioPack panels) had achieved so far - and we noticed that (inevitably) there was already some amount of divergence in style and functionality. Since then a "panel panel" was formed which reviewed the existing panels and came up with a "style guide" for VnmrJ parameter panels. However, rather than writing down a large set of rules and guidelines in our manuals (which may or may not be followed by designers of such panels) we decided to enforce such guidelines (design restrictions) through the panel editor (which is much more efficient than a written set of rules that people may or may not read!).
If you encounter cases of design restrictions such as with the panel editor we strongly recommend NOT to try bypassing such design restrictions, e.g., by editing the editing XML files "by hand" (i.e., using a text editor): in order to keep some coherence in the panel layout you should only use features that are available in the panel editor. Users are NOT meant to modify XML files in VnmrJ directly!
On the other hand, we realize that most of the current VnmrJ panel layouts don't yet follow our own recommendations; we do have plans to rework all VnmrJ parameter panels, but at this point we cannot predict when that work will be completed.

[Agilent MR News 2004-11-02]

GROUP BORDERS IN VnmrJ 1.1D PANELS:

The article above mentioned restrictions in the border style with groups in VnmrJ 1.1D panels. It was decided to make the border style (and the overall appearance) of a group dependent on the group type, i.e., the border option for groups in the panel editor has been replaced by the type of groups.

Currently there are three new types of groups: Major, Minor, and Convenience:

```
        Major Group:        Displays border with title (label of group).
        Minor Group:        Displays title (label of group).
        Convenience Group:  Displays no border and no title.
```

To display a group with border, set the type of the group as Major in the panel editor. By default the border type (for groups of type "Major") is set to etched. It is possible (but not necessarily recommended, see the previous article) to change the default values of the border preferences through the file "/vnmr/templates/vnmrj/interface/groupborder", or a user could also set local border preferences by modifying a local copy of this file, i.e., "~/vnmrsys/templates/vnmrj/interface/groupborder".

To give you an idea how the rules look like that were proposed by the "panel panel" (see the article above), we give you here an excerpt on the use of groups in VnmrJ parameter panels:

- There are three types of groups: Major, Minor and Convenience.
- All groups are rectangular.
- All titles, other than page titles, are associated with Major and Minor groups.
- Avoid filling a page with many small groups.
- Major Groups are the top level grouping within a page. Most Major Groups will have titles. They can not be nested, but may contain Minor Groups. The left alignment and width of Major Groups is a multiple of 70 pixels enforced by the panel editor.
- Major Groups have an automatic margin (about 5 pixels) around all edges. (If the group has a title, the top margin is enlarged to include the title.) The grid within the group (and the group's contents) fill only the area inside the margin; an item at 0,0 will be 5 pixels in from the left and top boundaries of an unlabeled group.
- Major Groups are normally butted against each other, both horizontally and vertically. Visual space between them is provided by automatic margins inside the groups.
- Minor Groups can only appear inside of Major Groups and can not be nested or contain Major Groups. The left alignment and width snaps to a 5-pixel grid. A Minor Group may be given a title.
- Convenience Groups are simply used to make editing pages easier and have no visible effect on that page. They can be used anywhere, and have no restrictions on alignment (beyond what individual widgets have). They are independent of hierarchical restrictions placed on other types of groups.
- Major and Minor may also be Mutable. These are groups whose contents change depending on the system state. (If the contents always are the same for a given user, it is one of the normal group types.) To enable this property, set the number of layers greater than one in the Panel Editor. The editor then lets you choose which layer is currently active for editing. Mutable groups will also get a distinctive look.
- Except on pages mainly filled by a table, or by a few large items, most on the items on a page should be within Major Groups. Action buttons that apply to members of more than one Major Group should be outside all Major Groups.

We do NOT mean to open a discussion whether these rules are good, convenient, etc., nor are there plans to post a comprehensive list of all rules in Varian MR News; in any case, you should assume that such rules are fixed (they may not be "cast in stone", but we spent considerable time working them out!).
Thanks to Mamta Rani (Varian, Palo Alto) for forwarding this information!

[Agilent MR News 2004-11-02]

2004-11-08:

VnmrJ PANEL LAYOUT RULES AND GUIDELINES - FOLLOW-UP:

The two articles on VnmrJ panel layout guidelines that we posted in the last issue (Agilent MR News 2004-11-02) seem to have caused some confusion. We therefore would like to try clarifying this a bit:

While Tcl-dg in VNMR 6.1 was (and still is) a powerful utility, it certainly had numerous limitations, both in terms of functionality, as well as in layout flexibility. In VnmrJ, the parameter panels are driven by Java-based software, and we use XML files for the panel layout definitions. These files can be found in "/vnmr/templates/layout", which contains both generic layouts (in a subdirectory "defaults") as well as pulse sequence specific ones (named after the pulse sequence). Every sequence uses a mix of generic and specific panels, whereby we try to limit the number of specific panel layouts to the minimum possible.

While the design of Tcl-dg panes was restricted by the use of a simple grid (typically about 16 rows and 10 columns), in VnmrJ panels items ("widgets") can be placed by dragging & dropping, and using pixel coordinates. There is virtually no limitation in the number of colors that you can use, and you can select from a large number of fonts. Furthermore, widgets in VnmrJ layouts can be grouped, and panels can even be "layered", i.e., you can cause a panel to "change its face" dynamically, as conditions (parameters) change. After having lived with the restrictions in Tcl-dg, in a first phase we wanted to open up the range of possibilities, such that people could explore all the flexibility in the new parameter panels. Over the past years we have gradually built up VnmrJ as a complete software solution (we are of course not finished yet with this), and at the same time, the various groups using the new software have built their sets of parameter panels. Naturally, as initially we did not enforce any strong style guide and rules, this led to considerable divergence in the results, as can be seen by comparing the imaging panels with those that George Gray uses with BioPack, what the solids applications people use, and with what you will see in the upcoming version of Chempack.

While this was good for the exploratory phase of the VnmrJ development, we feel that now the time has come to rationalize some of that layout diversity, hence the formation of the "panel panel" that we mentioned in the previous articles. As discussed, our aim is to come up with a set of guidelines and rules that would unify / "funnel" some of this diversity, such that all panels (at least those which we deliver) in the end will have a common look and feel. There are various aspects to this:

- There are some fundamental rules in graphical interface-building that our panels should follow - e.g.: radio buttons and choice buttons are there to define selections, NOT to perform actions (other than maybe those actions associated with implementing the selected parameter settings); for actions (commands, such as "wft", "dconi", etc.) buttons with the appropriate text labeling ought to be used.
- Another class of rules covers topics such as which fonts, what font and background colors are utilized in the various parts of the parameter panels, or where and how (at what distance from each other) panel widgets are placed in the layout. These topics may appear to be "pure aesthetics" - however, they definitely also have functional aspects: the VnmrJ software includes hundreds of panel layouts (over 250 in standard VnmrJ 1.1D - and BioPack raises that number to about 1100); not only is there a huge number of parameter panels that the user may encounter, but some of these also are quite densely "populated" (while others only feature a handful of widgets). All this can be rather confusing to a user, especially a newcomer to VnmrJ, and we don't really want this to be hard to use! In this situation it is very helpful if at least certain recurring / similar items are placed in common positions in the layout, and if specific colors and shapes help recognize certain types of widgets (and the associated actions).

These are the two areas which were of primary concern in the "panel panel" meetings, and the topic of the rules that the "panel panel" has put together.

As mentioned in the last issue, we could now think of writing a "book of rules" and expect this to be followed by panel designers. However, such books are often not read or consulted, and we don't have any means (nor of course the intent) of punishing people who are designing VnmrJ parameter panels! Instead,

- we decided to try guiding people into common layout design styles by adding (moderate) design limitations / restrictions in the panel editing utility, which will help any panel design that is done with the new editing tools;
- we will rework (NOT redesign from scratch!) the existing panels, such that they will (more closely) follow the new design guidelines. Right now, this task hasn't been started (nor even scheduled) yet - we hope to start this process over the coming year.

Again: we do NOT mean to ENFORCE these design rules, and it is up to you if you want to edit the XML parameter layout definitions in order to exploit all the design freedom that they offer - however, if you want your panels to "mold into" the standard VnmrJ interface philosophy it is HELPFUL if you restrict your panel design to the possibilities offered by the panel editor. We do NOT have the intent of dropping the support for some of the range of options that the layout definition offers, i.e., if you use a text editor to edit the XML panel such that it fits your exact needs, and that produces a functional panel now, you should NOT find yourself in a situation where in a future version of VnmrJ that panel suddenly no longer works!

Having said all this, we realize that very few users so far have started creating VnmrJ panels - the mere range of possibilities may have scared off some users - with VnmrJ 1.1D we certainly would like to encourage users (especially pulse sequence designers) to look into and explore VnmrJ parameter panels - they are powerful and the key element in the VnmrJ user interface!

Finally, a reminder: you can use the panel editor to change both sequence specific and generic panels. If you are editing generic panels, please make sure you save these in the appropriate "default" category (drop-down menu above the "Close" button) - see also Agilent MR News 2002-12-16.

[Agilent MR News 2004-11-08]

2005-01-22:

VnmrJ "INTERNAL" LOGINS AND ACCOUNT SECURITY:

When we designed the "internal user accounts" in VnmrJ 1.1D, we primarily did that with user IDENTIFICATION in mind - these internal "accounts" are not nearly as secure as a UNIX login (e.g., it is not too difficult to get access to the CDE desktop and then open a shell window, etc.), nor are they intended to be. In other words: if you have a problem with one user deliberately (or through negligence / ignorance) damaging another user's data, then you should not rely upon the limited protection of these internal accounts, but you should rather consider giving such users their own UNIX / VnmrJ account.

[Agilent MR News 2005-01-22]

2005-02-05:

USING THE VARIAN NMR USER LIBRARY UNDER LINUX - WARNING:

By the time when VnmrJ LX 1.1D (for Dell PCs running RedHat Linux) was released, the User Library had received VERY MARGINAL TESTING under the new environment. As Krish Krishnamurthy (Eli Lilly, Indianapolis IN) found, the "extract" script that was included on the software media actually had a bug that could disable VnmrJ if an attempt was made to install a contribution in "/vnmr". We STRONGLY RECOMMEND that you download the current version of "extract" (dated 2005-02-01 or later) from the on-line User Library at

http://www.varianinc.com/products/nmr/apps/vnmrusers.html

Please always read the README file that comes with a contribution (if you download by e-mail or using direct FTP), or read the Web page associated with any contribution when downloading via the FTP link through the above Web site and check the compatibility notes. It will take a while until we have re-checked the bulk of the contributions in the new software environments! In any case, it is worthwhile checking the above Web site whether there is an updated version of ANY contribution that you intend to use - the link "Recent Additions and Updates" gives you quick and easy access to updated versions.

Keep in mind that most pulse sequences that have been submitted (with the exception of contributions such as "psglib/BioPack") do NOT contain VnmrJ parameter panel support - they MAY work, but you would have to use them by selecting the "Text Output" panel under the "Process" tab, and typing "dg", "dgs", etc. on the VnmrJ command line.

This of course also (and even more so) applies to VnmrJ_MAC 1.1D running under MacOS X, except that this will not be used to acquire data, i.e., with MacOS X it only affects contributions in areas other than acquisition.

[Agilent MR News 2005-02-05]

2005-10-18:

VnmrJ HINT - ENABLING THE COMMAND LINE FOR OPERATORS:

In VnmrJ, the owner of an account (the user who does a full UNIX login, indicated by the parameter "owner") has access to the command line, while by default, VnmrJ-internal operators (users OTHER THAN the owner of the account, accessed through the "Switch Operators" menu option, and indicated by the parameter "operator") do NOT have access to the VnmrJ command line. This should be adequate for typical open-access sites that use VnmrJ only and which do not have older spectrometers running VNMR: walkup users use VnmrJ through the graphical user interface (GUI) only - they never get into direct contact with the underlying commands and are therefore don't have a need to become familiar with the VnmrJ command and parameter names and their use. At the same time, this feature prevents unexpected results, maybe even inadvertent loss of data from using inappropriate commands, arguments, parameter names and values

by untrained operators.
  However, there are sites where all or some operators are familiar with the VnmrJ / VNMR command line interface (either from predecessor instruments or because they also run older Varian machines that are operated through the "classic VNMR" interface), and in these cases it may be legitimate to let some or all operators have access to the VnmrJ command line.
  Fortunately, that's easy to do: simply edit "/vnmr/maclib/operatorlogin" and look for the following lines (lines 26 - 32 in VnmrJ 2.1A):

```
        if (operator=owner) then
          vnmrjcmd('cmdLine','yes')
          "write('line3', 'cmdLine yes')"
        else
          vnmrjcmd('cmdLine', 'no')
          "write('line3', 'cmdLine no')"
        endif
```

to allow for universal command line access simply shorten that section to

```
        vnmrjcmd('cmdLine','yes')
```

You could also implement a more differentiated access scheme, e.g.:

```
        if (operator=owner) then
          vnmrjcmd('cmdLine','yes')
        elseif (operator='user1') or (operator='user2') or ... then
          vnmrjcmd('cmdLine','yes')
        else
          vnmrjcmd('cmdLine', 'no')
        endif
```

where "user1", "user2", etc. are the names of the "empowered operators".
[Agilent MR News 2005-10-18]

2006-02-08:

PULSE SEQUENCE MODIFICATIONS AND PARAMETER PANELS IN VnmrJ:

  A user made a simple modification to the standard "gHSQC" pulse sequence under VnmrJ 1.1D. the new sequence was given a new name by appending the author's initials to the name, i.e., "gHSQC_XY.c". As the sequence did not require any new parameters, the author wanted to test it based on a setup for the original sequence, simply by setting "seqfil" to the name of the new sequence, and then acquiring and processing comparison spectra. In older VNMR software this was often all you needed to do (and this should still be true for simple 1D sequences in current VnmrJ) - but in the case of VnmrJ all 2D parameter panels were lost as soon as "seqfil" was switched. What happened?
  VnmrJ inherently / automatically uses sequence-specific parameter panels, if such panel layouts are defined. This mechanism is based on the parameter "seqfil". While with the original sequence the panels from the directory "/vnmr/templates/layout/gHSQC" were used, setting "seqfil='gHSQC_XY'" causes the system to look for a directory "/vnmr/templates/layout/gHSQC_XY". As this did not exist in the case above, the parameter panels defaulted to the layout definitions in "/vnmr/templates/layout/default", which defines panels for 1D spectroscopy only, hence the loss of the 2D panels.
  Of course, you are NOT required to go through all the motions of creating the missing panel layouts from scratch. The simplest solution in the above situation would be to create a copy of the "gHSQC" parameter panels with the name of the new sequence, i.e.,

```
        cd ~/vnmrsys/templates/layout
        cp -r /vnmr/templates/layout/gHSQC gHSQC_XY
```

and you are all set. Note that a symbolic link "gHSQC_XY" pointing to the original layout would basically also do the job - but this is NOT A GOOD IDEA, as when you start adjusting parameter panels for your new sequence, you might inadvertently modify the layouts for the original sequence!
  A full duplicate of the original panel layout definitions is actually NOT the minimum requirement, even though starting with a complete, closely related parameter panel layout is usually the recommended procedure. If you DON'T want to take over all the original, sequence-specific parameter panels (for "gHSQC" in the above example), you can still get the 2D panels by switching to the default layout for 2D sequences: use

```
        mkdir -p ~/vnmrsys/templates/layout/gHSQC_XY
```

to create a layout directory for the new sequence, and create a file "DEFAULT" in this new directory, containing a single line

```
        set default default2d
```

(note that unlike all the other layout files, this one is NOT in XML format).

This causes VnmrJ to take the 2D panels from the directory "default2d" (in your "~/vnmrsys/templates/layout" if present, in "/vnmr/templates/layout" otherwise), on top of the panels in "templates/layout/default".
[Agilent MR News 2006-02-08]

2006-12-15:

CREATING A NEW PROTOCOL FOR VnmrJ AUTOMATION:
(by Christine Hofstetter & Bert Heise, Varian)

The topic of creating new protocols for VnmrJ Automation is currently not addressed in detail in the VnmrJ manuals - in the VnmrJ 2.1B manuals, it is missing completely.
Often, users may wish to use default parameter settings for their NMR experiments that differ from the standard settings defined in "/vnmr", which involves creating a modified protocol for VnmrJ. Similarly, users may want to introduce a new protocol for automation, such as a 1D experiment on a nucleus that is currently not covered, e.g., "Silicone". To do this, the Automation account owner (UNIX login), or (better, see below) "vnmr1" can create a new "Protocol" for an experiment that involves a customized parameter set. To make a new protocol,
- Load a basis parameter set (e.g., "Proton");
- Change "tn" / "dn" and other parameters ("nt", "d1", ...) as needed;
- Acquire a spectrum and readjust parameters until you are happy with the result;
- In the "Tools" menu (VnmrJ 2.1B, under "Utilities" in VnmrJ 1.1D), choose "Create Protocols" -> "Make a New Protocol...";
- The parameters "sw", "sw1", "tn", "dn" and "ni" are set by the setup macros "std1d", "homo2d", and "hetero2d". To override these pre-set values with custom settings, changes MUST be placed into the "Customization" entry, in (command line) parameter entry format, e.g.:
  ```
  setsw(tn,15,-3) dn='Si29' setsw1(dn,200,-50):dof
  ```
- Choose a new name for your new protocol ("Protocol Name"). The same name should be used for the "Search Type" (for the Locator).
- In general, 2D protocols should have a "Proton" in the "Required Protocol" field.
- Select "Make Protocol" to save the new protocol. A new entry will automatically be created under the "Local" tab in the experiment selector.

Some additional notes:
- The easiest way to add another nucleus to the 1D experiment selection for walkup users is to start from a standard 13C parameter set. Change "tn" to the nucleus of interest and set other parameters that need to be adjusted, such as "sw", "nt", "bs", "dm", "dmm", etc.; alternatively, the command
  ```
  AuNuc('new_nuc')
  ```
  can be used to create a reasonable parameter set (where "new_nuc" is the nucleus of interest).
- If you ONLY JUST create a local version of a system protocol such as "Ghsqc", the timing in the StudyQ will not reflect the new parameters, e.g.: when choosing "ni=256" instead of the standard setting ("ni=128"), the studyQ will still show 14 instead of 28 minutes; therefore, protocols that should replace system protocols such as "Proton" or "Carbon" MUST be copied to "/vnmr". This can be done with the (undocumented) command "jpublish" which should be issued as user "vnmr1" while the desired parameter set is loaded in the active workspace. If the current user is not "vnmr1", a compressed "tar" file containing all relevant files will be created in the user's "~/vnmrsys" directory which vnmr1 then needs to copy to "/vnmr" and extract with the following procedure:
  ```
  cp Myprotocol_proto.tar.Z /vnmr
  cd /vnmr
  zcat Myprotocol_proto.tar.Z | tar xfBp -
  ```
- To change protocol buttons in the experiment selector for a walkup account, edit their definition in the file "/vnmr/adm/users/protocolListWalkup.xml" by adding or removing entries.
- In the experiment selector, local protocols will automatically pop up under a "Local" protocol tab, in the order in which they were entered. To remove undesired protocols from the "Local" tab in the experiment selector (e.g., because they were transferred to "/vnmr"), remove the relevant entries from "~/vnmrsys/templates/vnmrj/protocols". To ensure that local parameters don't override the new settings in "/vnmr", the associated local parameters ("~/vnmrsys/parlib/Myprotocol.par") should be deleted, too.

```
 - In order for the experiment to show up in the pull-down menu, edit the file
   "/vnmr/templates/vnmrj/interface/MainMenuExpts.xml" or edit a local copy in
   "~/vnmrsys/templates/vnmrj/interface". For a "Boron" 1D experiment, add the
   following lines:
        <mchoice label = "Boron"
          vc = "Boron"
          style = "Menu1"
          show = "exists('s2pul','psglib'):$SHOW"
        />
   Alternatively, make a copy of a similar entry and use it as a template.
 - If you want the experiment to appear in the main experiment selector (not
   just under the "Local" tab), add a new entry to the file defining the
   protocol list, "/vnmr/adm/users/protocolListWalkup.xml": under
        <page name = "Std1D">
   (for a new 1D protocol) add a new line
        <protocol name = "Boron" label = "Boron" />
 - Adding the new nucleus to the probe file is still done the "traditional"
   way: e.g., on the VnmrJ command line type
        addnucleus('B11')
                                                [ Agilent MR News 2006-12-15 ]
```

2007-02-20:

WHAT TO DO WHEN VnmrJ ADMINISTRATION THROUGH "vnmrj adm" FAILS:

The VnmrJ administrative interface, accessed via "vnmrj adm", performs many tasks - such as creating and setting up UNIX accounts and the associated files and directories for VnmrJ users - that (strictly speaking) only "root" should be allowed to do. This could be achieved by requiring the system administrator to log in as root for all such tasks, and in the past this was exactly what we did. However, there are potential problems with this option:

- repetitive use of the root login bears the danger of the root password being disclosed and/or becoming commonly known in a lab, which imposes substantial security risks;
- in some labs the system administrator should be able to create and update VnmrJ user accounts, but may (for security or general policy reasons) NOT be in possession of the root password (there are many IT departments out there which do or would like to enforce such a policy!).

The solution for these issues is "sudo", modern utility (available under both Linux and Solaris) that

- permits executing SINGLE and SPECIFIC commands with root permission (by default, "sudo" will ask for the root password for each execution), but which also
- permits setting up a database of specific commands which specific users are allowed executing with root permission WITHOUT entering the root password with each call; this is what is used for "vnmrj adm".

Needless to say that for SETTING UP the database for the second option you MUST be logged in as root ONCE: VnmrJ is installed as root, and the "sudoers" database is set up as part of that installation.

Recently, the editor has been confronted with a number of cases where the "vnmrj adm" interface suddenly started asking for the root password again (which is an indication that the "sudoers" setup is incorrect) and/or was hanging / getting stuck. In most or all of these cases the issue could be resolved by calling

```
        /vnmr/bin/sudoins
```

once (again) as root, in order to re-generate the VnmrJ "sudoers" database. There is an existing bug report "vnmrj_adm.j1117" that may be related to these incidents, but currently it is unclear whether this is the same issue or a new problem.

[Agilent MR News 2007-02-20]

2007-04-10:

AUTOMATED Wet1D WITH VnmrJ (by Ron Crouch, Varian):

One of the new features introduced with VnmrJ 1.1D software is the concept of dramatically simplified, automated creation of parameters for Wet1D. Subsequent conversion of any solvent-suppression 1D experiment into a 2D also automatically retains all of the needed parameters. The steps to utilize these new tools successfully are quite simple. The purpose of this note is to

outline the tools provided for WET solvent suppression.
  The first step would be to ensure that the parameter set of the starting "Wet1D" ("Wet1D.par/procpar") has the correct entries for "ref_pw90" and "ref_pwr", as these are needed for Pbox to create a proper shaped pulse. This is mostly an issue of secure "fall-back settings": the "wetit" macro (see below) reads these values off the probe file, therefore it is important to ensure that properly calibrated values are also stored in the probe file. Basically, all that is required for obtaining proper values for "ref_pw90" and "ref_pwr" is to determine the "pw90" at a fairly low power level out of the region of amplifier compression. A "tpwr" value such as 52 to 54 should be fine, but depending upon the probe users could choose any power that they desire. The formula to calculate "ref_pw90" and "ref_pwr" from calibrated parameters from the probe file is very simple:

```
        ref_pwr=tpwr
        ref_pw90=pw90*tpwr_cf
```

where "tpwr_cf" is the compression factor at the given "tpwr" value.
  Once proper values for reference power and pulse width values are determined the "Make New Protocol" tool should be used to update (or create local) parameters with these settings. One could also edit the parameters for "Wet1D" in "/vnmr/parlib/Wet1D.par/procpar", but in this case a simple

```
        svp('/vnmr/parlib/'+pslabel)
```

as the user vnmr1 would also be fine. To save the same parameters in the local "~/vnmrsys/parlib" use

```
        svp(userdir+'/parlib/'+pslabel)
```

The mechanism for automated WET pulse calculation in the interface for "experimental" users works as follows:

- Setup the "Wet1D" protocol, and set "wet='n'"
- Acquire a quick spectrum. Set the parameter "wetpeaks" to the number of peaks to suppress.
- Type "wetit". This macro will apply a 40 Hz line broadening and adjust the threshold until only the number of peaks specified in the "wetpeaks" parameter are found. The WET pulse is automatically generated, and all relevant parameters are passed to the current experiment.
- Acquire the 1D spectrum with WET suppression!

If it is desired to adjust the observed spectral window, all that the user needs to do is to type

```
        wetit('recalc')
```

after the "movesw" command, and the WET pulse will be recalculated. Subsequent 2D experiments will inherit the WET parameters.
  In the "walkup" interface, all of these steps are accomplished simply by selecting "Wet1D" in the experiment selector, double-clicking the resulting entry in the StudyQ, and selecting the desired number of peaks for suppression under the "Acquire" menu / "Prescan" page. An important detail for doing this in automation is that it may be necessary to perform this once manually by setting "testacquire='y'" to bring the parameters into the foreground experiment and to update the local parameters, in order to ensure that the WET parameters are available. Doing this avoids errors that could interrupt / halt an automation run. Once the local parameters are OK and saved in a "parlib" directory, subsequent automated WET experiments should proceed properly in that account.
  If for some reason the automation run hangs as would be indicated by the StudyQ entry showing "active" but the acquisition status remaining "idle", recovery is very simple. In a terminal type "su acqproc", wait a few seconds and repeat "su acqproc". After that, the failed active node in the StudyQ can be dragged to the trash. Once the foreground experiment runs without error and the parameters are saved locally, automation runs on new samples should proceed well.
  The information in this note also applies to VnmrJ 2.1B.
[Agilent MR News 2007-04-10]

2007-09-19:

VnmrJ / LOCATOR DATABASE PERFORMANCE VS. MEMORY SIZE:

  The VnmrJ Locator is essentially just the front-end / the user interface for a database that is running in background. For the Locator, we decided to use the "PGSQL" database software. PGSQL is distributed under a BSD-like license and is therefore free software. The name "PGSQL" comes from "PostgreSQL". For detailed information on PGSQL see Wikipedia; briefly: the acronym "SQL" stands for "Structured Query Language" and is an ISO and ANSI standard interface language definition used by most major relational database programs (such as Oracle, MySQL, Microsoft/Sybase, etc.). The name section "Postgres" stands for "Post-Ingres" and refers to a database project "Ingres" that was developed at UC Berkeley up to 1982. Postgres then was a follow-up project that was started in 1985, also at UC Berkeley, and SQL was added in 1994/95. Solaris 10 6/06 and later comes with PostgreSQL included.

Using an SQL database has the advantage that one is more or less independent of the underlying database software (as long as it is SQL-based). PGSQL and similar relational databases are designed to handle huge amounts of data (millions of records), i.e., from a software point-of-view they are certainly powerful enough to handle the needs of even very large NMR / MRI sites.

The interface part of the Locator database was relatively straightforward to implement (though the SQL language is fairly verbose and clumsy) - however, in the area of interdependence of database performance, database maintenance and computer configuration (particularly the RAM size) we had to go through a learning curve. To some degree, this is an issue of testing: while we do have demo / testing systems configured for high throughput / high data volume environments, we hardly have the resources to generate a high throughput situation under realistic conditions - therefore, to some degree we rely upon feedback from the customer base.

- First reports about slow-downs came from imaging sites and high throughput automation labs. We then found that the database did not immediately remove deleted items, but just tagged them for removal. We then proposed a recipe to call
  ```
  vacuumdb vnmr --analyze
  ```
  periodically, in order to keep the database clean of deleted items, see Agilent MR News 2003-09-20. This process has since been built into the VnmrJ software and no longer needs to be implemented by the user.
- More recently we came to some interesting conclusions relating to RAM size vs. database performance. In the early days of VnmrJ we had already seen that for VnmrJ and the underlying Java engine to run satisfactorily, one really should have 1 GByte of RAM rather than just 512 MBytes; increasing the swap space does NOT help. On a PC running Linux, one finds that with VnmrJ running, typically around 600 - 750 MBytes of RAM are "active", i.e., in use (under Linux this can be evaluated using the command "vmstat -a" in a terminal / shell window). In other words: with 1 GByte of RAM, VnmrJ is responsive and working OK, while with only 512 MBytes, paging is observed. HOWEVER, we now found that with a large number of items in the database (we did tests with 1 Million items) and 1 GByte of RAM, "vmstat -a" (Linux) may still indicate 300 MBytes of free RAM, but the database performance may be down by a factor of 10 - and yet we have been unable to track this down using tools such as "vmstat". Our findings so far are that
  - PGSQL appears always to release memory after every search (so it avoids what people call a "memory leak", i.e., a progressive increase in the amount of allocated virtual memory); however,
  - PGSQL appears to try maintaining about 300 MBytes of unused PHYSICAL MEMORY (at least, this is our empirical observation): with a large number of items in the database and only 300 MBytes of free RAM, all transactions appear to involve disk access. If the overall RAM size is increased to 2 GBytes, i.e., the free RAM (as per "vmstat -a" in Linux) is increased substantially beyond 300 MBytes, database slowdowns can be avoided (under the conditions that we tested), as PGSQL now uses RAM caching. Note that increasing the swap space (i.e., the amount of VIRTUAL memory) does NOT help here.

The PC configuration that we currently sell and deliver includes 2 GBytes of RAM, hence we do not expect you to encounter major performance issues with the Locator database - but what does all this mean for older systems with less than 2 GBytes of RAM?

- If you only have a limited number of datasets in the database, 1 GByte of RAM should still be OK.
- If you generate a lot of data, but periodically remove old data from the database, this should avoid the issue; the latest set of VnmrJ patches (see Agilent MR News 2007-09-13) and VnmrJ 2.2C feature a utility for such periodic clean-up of the database (see the documentation for information).
- Obviously, if the requirements of a high throughput lab (such as imaging facilities, or some "heavy duty" automation sites) involve keeping several years worth of data in the Locator database, then such periodic clean-up is not an option. In these cases, if you observe database performance issues, we recommend expanding the RAM to 2 GBytes. Luckily, with current RAM prices, this is not a big expense, but may offer substantial VnmrJ

   performance gains. You may want to use "vmstat -a" (Linux) or "vmstat"
   (Solaris) to evaluate the amount of free RAM.
Note that the Linux "System Monitor" ("Applications" -> "System Tools" ->
"System Monitor", select the "Resource Monitor" tab) is useful to see short
term changes in CPU and VIRTUAL memory consumption (active memory), but does
NOT indicate the amount of unused PHYSICAL RAM.

[Agilent MR News 2007-09-19]

2007-10-01:

VnmrJ TIP: WHAT COMMANDS DO THE GRAPHICAL MENU BUTTONS PERFORM?

Assume you want to write a little macro utility that (among other things, maybe) mimics one or several clicks on buttons in the vertical, graphical menu in VnmrJ. The question is: how do you know what commands are executed "behind" these menu buttons? The buttons only show a graphical icon that is supposed to indicate the underlying action. If you have turned on the VnmrJ tooltips, you can move the mouse pointer over a button to see a text button label - but this again is just a button label (similar or equivalent to the button label text in the "VNMR classic" interface), typically NOT the underlying command(s). For the VnmrJ parameter panels you can use the panel editor to view what VnmrJ commands a button executes ("Vnmr command" field in the button definition), but there is no graphical editor for the graphics menus.
The solution turns out to be fairly easy: the contents of the current graphics menu is held by the three global, arrayed string parameters:

- "micon" holds the names of the files for the graphical icons, such as "dfid.gif" (i.e., a GIF file); the icon files used for menu buttons are stored in "/vnmr/iconlib".
- "mlabel" is an array holding the button labels, e.g., "Display FID"
- finally, the parameter "mstring" holds the commands that are executed when pressing a given menu button - in the above example, it is
  ```
       menu(`fiddisp_1D`)
  ```
  i.e., that button will switch to a new graphical menu and the associated actions. Of course, such actions can also be direct VnmrJ commands such as
  ```
       dconi(`toggle`)
  ```
  or complex actions such as
  ```
       select:$vjms $vjms=$vjms-1 $vjms=trunc($vjms) ds_dim:$vjmdim if \
       $vjms<0.9 then $vjms=1 elseif $vjms>$vjmdim then $vjms=$vjmdim endif \
       ftproc[2]=$vjms ds($vjms)
  ```

These are "parallel arrays", i.e., each of these parameters holds as many string values are there are menu buttons on display. To query the action behind the first (top) menu button simply type
```
        mstring[1]?
```
In VNMR, all menu definitions were stored in a directory "/vnmr/menulib" (or "~/vnmrsys/menulib" for local definitions). The VnmrJ graphical menu definitions are stored in files in "/vnmr/menujlib". If you look at these files you will find that the VnmrJ and the VNMR menu definitions are pretty similar or equivalent. The key differences between VnmrJ and VNMR are that

- the VnmrJ files also define "micon", while VNMR did not have graphical menu icons
- there are many more files in "/vnmr/menulib" that don't have an equivalent in "/vnmr/menujlib"; that's because many tasks from the old style VNMR (classic) menus were transferred to parameter panels in VnmrJ or replaced by other VnmrJ interface components such as top level menus, etc.

You can of course also use "grep" in a shell window to scan the VnmrJ menu definitions. For example, to see where VnmrJ menus call "dconi('toggle')" you could use the commands
```
        cd /vnmr/menujlib
        grep dconi * | grep toggle
```
Of course you can view the menu files directly; their syntax is macro-like (they are macros, actually) and typically fairly straightforward to read.
Thanks to Ryan McKay (NANUC) for (indirectly) suggesting this topic!

[Agilent MR News 2007-10-01]

2007-10-06:

VnmrJ TIPS - EDITING PANELS:

VnmrJ parameter panels are typically easy to edit: go to the page that you want to edit, then select "Edit" -> "Parameter Pages..." from the top menu, which opens the panel editor on that page. Technically, the panel editor is pretty straightforward - though the widget properties may not be trivial to understand for beginners. The panel editor is documented in Chapter 6 of the User Programming manual for VnmrJ 2.2C (that chapter is not present in earlier versions of this manual). Thanks to Maj Hedehus, Bayard Fetler and Everett Schreiber (Varian, Palo Alto) for taking the initiative and writing up that section of the VnmrJ 2.2C documentation!

Also, we have collected a series of articles from past issues of Agilent MR News in a FAQ document "VnmrJ Tips and Tricks" ("faq/vnmrj_tips") that you can find via our "Software Corner" at

http://www.varianinc.com/products/nmr/apps/corner.html

[Agilent MR News 2007-10-06]

VnmrJ TIPS - EDITING PANELS WITH "INVISIBLE" WIDGETS:

There is one rarely used feature in VnmrJ panels which may appear to prevent editing certain aspects of a parameter panels: overlapping groups.

Widgets can be organized in "groups", i.e., several widgets can be placed within a rectangular "group" widget (a "box"), and the widgets in the group are then also controlled by the group's "show condition": if the (parameter) condition is such that a group is not shown, then all widgets in that group are hidden as well. That by itself is not a problem with the editor, as in the editor mode the group is visible even if it is hidden in the "live" panel.

What is tricky to edit is panels with overlapping groups. One such example is the "Study" panel in the VnmrJ 2.2C "walkup" interface: here, the "Tune Check" and the "Gradient Shim Check" groups are placed almost on top of each other. Such an arrangement only makes sense if the "show condition" for the two groups is mutually exclusive, i.e., in the live panel you see either one or the other (in this example, the ProTune check boxes will only be shown if ProTune is present / installed, not otherwise). In this case, one of the panels appears to be on "top", and it's not easy to see or edit the panel "underneath".

In this situation, some users may be tempted to edit the associated panel definition file - in this case the XML file "Study.xml" in the directory "/vnmr/walkup/templates/layout/default" - however, this is NOT the recommended solution: these XML files are NOT meant to be edited "by hand" (i.e., using a text editor), except maybe (rarely) for trivial changes. The proper solution to editing such panels is by

- dragging off the top group while in the editing mode,
- then edit the hidden panel, and
- finally to move back the top panel over the bottom one, to where it was before.
- now save the panel and exit the editor.

Thanks to Bayard Fetler (Varian, Palo Alto) for providing input for this note!

[Agilent MR News 2007-10-06]

2007-11-14:

VnmrJ PARAMETER ENTRY AND GUI DESIGN CONSIDERATIONS:

In the early VNMR software versions the user interface was simple and easy to understand, at least from a basic functionality point-of-view:

- on the command line, command and macro calls were typed, entries were terminated (submitted) by using the [Return] key;
- the menu structure was simple, non-graphical
- the graphical interaction was limited to the graphics area (apart from the use of the mouse for selecting menu buttons).

The key disadvantage of such user interfaces is that users need to memorize a large number of commands and parameter names, and also the user interface offered limited help as to what the next logical steps in the workflow would be.

With Tcl-dg, VNMR 6.1 started a transition to a truly graphical user interface (GUI), which VnmrJ 2.2C now completes to a degree that users can do most or all of their work without ever resorting to the command line. The bulk of the user interaction is through interface buttons ("action widgets"), sliders, menus (scrolling or pull-down), and parameter entry boxes ("entry widgets") in parameter panels - and of course through mouse clicks and dragging in the graphics area ("graphics canvas"). Most of these interface elements are nothing new, as they are commonplace in everybody's desktop software (such as office productivity software, etc.) on PCs and Macs.

Still, VnmrJ - and probably most spectroscopy software in general - is somewhat different, in that it requires frequent entry of numeric or string values into parameter entry widgets (especially for advanced, "experimental" users), and sometimes things don't quite work the way users might expect, causing occasional, subtle hiccups.

The key issue is: how do you enter a value into an entry widget? Generally speaking, there are two basic types of entry widgets:

- widgets that are constantly monitored and processed, such as text entry in a text canvas in word processing software (e.g., with on-the-fly formatting and syntax checking)
- widgets which require some user action to indicate termination of the parameter entry, i.e., where the entry value is processed only when the entry is "terminated" / completed.

All VnmrJ parameter value entry widgets (and somehow also the command line) are of the second type - and this implies that you cannot just type a value and expect the software to "know" when the entry is completed. Instead, entries are completed

- by using the [Return] key
- by using the [Tab] key (in order to move the input focus to the next entry widget to the right of further down in the GUI
- when moving the focus by selecting another parameter entry widget in the same GUI section (by means of a mouse click).

We just found that there are instances where selecting an ACTION WIDGET (such as a button, possibly also other widgets such as radio buttons, or choice widgets) may NOT move the input focus and hence DOESN'T execute the "capture" of a preceding, incomplete (i.e., unterminated) parameter entry, see also bug report "panels.j2208" (we are investigating whether this is a programming bug on our side or rather a "feature" in the underlying software). It is best (and strongly recommended) to make it a habit to terminate value entries with a [Return] or [Tab] key.

A good example for this is the temperature entry widget in the "Setup" -> "Spin / Temp" parameter panel in VnmrJ 2.2C: there is a value entry box, "connected with" a slider. The slider can be used to set the "temp" value without using the keyboard, just through the mouse: you move the slider, and the value in the entry box then serves as a numeric indicator of the currently selected value. The hardware of course does not try to follow this entry directly - the selected value is used either with the next acquisition, with "su" via the command line (or by using the "Setup Hardware" button), or when using the "Regulate Temp" button above the slider. The slider itself has no scale, and some users may dislike setting the temperature using such an "analog" utility. Instead, you can highlight the value in the numeric widget using the mouse (or you can place the insertion mark somewhere in that value and use the arrow keys to move the insertion mark, the backspace key to erase digits), and type a new value. While you type, the slider will not move (i.e., the entry is not terminated yet), only when you hit the [Return] or the [Tab] key or select a different entry widget (to move the focus), the slider will change to the position that corresponds to the new value.

[Agilent MR News 2007-11-14]

2007-11-18:

VnmrJ PARAMETER ENTRY AND GUI DESIGN CONSIDERATIONS - ADDENDUM:

In the last issue (Agilent MR News 2007-11-14) we posted an article that commented on the VnmrJ GUI properties, particularly describing the parameter entry through VnmrJ panels and their input widgets, and the user of the [Tab] and [Return] keys to terminate / confirm entries. As Bayard Fetler (Varian R&D, Palo Alto) pointed out, there are two exceptions in VnmrJ panels that don't follow this description: in the case of the "textfile" and the "editable Text" widgets (both are found in the Locator under "advanced" / "elements" when using "Sort Panels and Components" while using in the panel editor) are different: while the focus is on these widgets, both [Tab] and [Return] characters are added to the text that is being edited through these widgets, rather than being interpreted by the GUI. In this case, you MUST use the mouse pointer to move the focus to a different widget or GUI component. Thanks, Bayard, for this information!

[Agilent MR News 2007-11-18]

2007-11-28:

SWITCHING BETWEEN VERSIONS OF VnmrJ - UPDATE / CORRECTION:

In Agilent MR News 2004-03-07 we posted an article with detailed recipes for switching between two local VNMR / VnmrJ software installations (e.g.: between different versions of VnmrJ, between two versions of VNMR, or between VNMR and VnmrJ and vice versa). Although that recipe was written up before VnmrJ 1.1D was fully released, it is still applicable also for current versions of VnmrJ, with one exception:
Whenever switching TO any version of VnmrJ, our recipe (kindly provided by Frits Vosman, Varian R&D) included a command

```
        dbupdate once 0
```

(to be called as VnmrJ administrator) in order to enforce a Locator database update. As the editor just learned, that command is actually outdated / obsolete and should no longer be used. Most of its tasks - namely the nightly update to keep the Locator database up-to-date - have been integrated into VnmrJ 1.1D and its successor versions, and the "dbupdate" utility (which has never been adapted for Linux systems anyway, see bug report "dbupdate.j2201") will be removed from future versions of VnmrJ.
For switching to VnmrJ 1.1D or any later version (or in order to enforce a database update) you should now use

```
        managedb update
```

in lieu of the above command. We will add an amendment to the article from Agilent MR News 2004-03-07 in our News archive, such that Linux users don't get stuck with "dbupdate".

[Agilent MR News 2007-11-28]

2007-12-10:

PROBLEMS WITH ADMINISTRATIVE TASKS UNDER VnmrJ:

We keep getting occasional inquiries from users and installers who are fighting problems with using the VnmrJ administrative interface ("vnmrj adm") for tasks such as defining or updating VnmrJ users, or defining printers / plotters. In VnmrJ, all these tasks involve the Linux / UNIX "sudo" utility (see also Agilent MR News 2007-02-20), and in most of these cases, it's "sudo" which causes "vnmrj adm" to fail.
While we are still investigating why some of this happens, we have found at least one chain of events that causes such issues. One key action causing subsequent problems is a change in the hostname (see also the article below). The point is that the "sudo" setup involves a text database ("/etc/sudoers") that links users and commands that specific users are allowed to perform with root permission. This database also lists host names, such that in principle the same database file ("/etc/sudoers") can be set up to be usable on all systems within a local network. The point is that when you change the host name, the new name will no longer match any entries in "/etc/sudoers" (as created by "/vnmr/bin/sudoins" when installing VnmrJ), and "sudo" thereafter fails. Consequence: whenever you change the host name, you MUST call

```
        /vnmr/bin/sudoins
```

explicitly, as root. This should also be done when you encounter other, unexpected issues with "vnmrj adm" (such as sudden requests to enter the root password, etc., see also Agilent MR News 2007-02-20), prior to calling for support.

[Agilent MR News 2007-12-10]

2008-02-13:

UNDOCUMENTED OPTIONS FOR FILE NAMING TEMPLATES IN VnmrJ:
(by Daina Avizonis, Varian)

The "autoname" name template for automatic saving of NMR data during an automation run is explained in the Command and Parameter Reference manual. The idea is that users can have their data saved using a naming scheme of their own choice, rather than relying on a fixed naming scheme - i.e., we want to avoid forcing users / administrators to move / rename files for archival purposes, once the automation run is complete. The path that is obtained through the "autoname" mechanism must fulfill two criteria

- file names should be meaningful to the user, permitting associating results with a given sample and - if possible - provide additional information about the sample and possibly the experiments performed, facilitating subsequent retrieval of the data based on intuitive and understandable

criteria / mechanisms;
 - file names must be guaranteed to be unique

The latter point is fulfilled typically by adding a (usually two-digit) revision number into the file name. For the first criterion, "autoname" permits using information from the "sampleinfo" ("enter") file (using template pattern such as "%USER:%"), as well as parameter values, through template elements such as "$pslabel$" - see Agilent MR News 2003-09-08 for a detailed article by Bruce Adams on this topic. In VnmrJ, the "autoname" template is set using the "Save data setup" dialog.

Starting with VnmrJ 1.1C, we added new template elements which permit incorporating date / time information into the "autoname" and "Svfname" templates, see Agilent MR News 2003-10-17. The associated template "tokens" are discussed in the Command and Parameter Reference manual and include

```
        %YR%                   4-digit year    (yyyy)
        %YR2%                  2-digit year    (yy)
        %MO%                   2-digit month   (mm)
        %DAY%                  2-digit day     (dd)
        %HR%                   2-digit hour    (HH)
        %MIN%                  2-digit minute  (MM)
        %SEC%                  2-digit second  (SS)
```

and the "composite" tokens

```
        %DATE%                 yyyymmdd
        %TIME%                 HHMMSS
```

If you would prefer using an ISO representation for date and time, that can be done as well, using the above elements:

```
        %YR%-%MO%-%DAY%        date in ISO format, e.g.: 2008-02-13
        %HR%:%MIN%:%SEC%       time in ISO format, e.g.: 13:21:40
```

We have just noted that two date options are missing in the Command and Parameter Reference manual:

```
        %MOC%          3-character month representation ("Jan", "Feb", etc.)
        %DAC%          3-character day of the week ("Mon", "Tue", etc.)
```

With these, you can construct popular (non-ISO) date representations such as

```
        %DAY%%MOC%%YR%         e.g.: 11Feb2008
        %DAC%_%YR%%MOC%%DAY%   e.g.: Tue_2008Feb12
```

[Agilent MR News 2008-02-13]

2008-03-27:

REDHAT LINUX AND THE "head" AND "tail" COMMANDS:

The article above may appear fairly esoteric and theoretical - but when it comes to concrete changes / evolutions in command definitions, such changes can have very real (and potentially severe) consequences to the user (hence Sun's hesitations to alter their command definitions!). One such "incident" is currently happening with the UNIX / Linux "head" and "tail" commands:

 - in the original SVR4 definition, the number of text lines (N) that are shown by these commands can be specified as "-N", e.g.: "head -3" and "tail -7". The "tail" command can also be used with a "+" character in lieu of the "-", e.g.: "tail +10" prints all lines starting with line 10 (as opposed to counting lines from the end of the file). In the case of "tail" this is the ONLY syntax supported by the Solaris / SVR4 command definition.
 - the X/Open definition of these commands specifies the use of a "-n" command option, followed by a numeric argument. The following calls are the XPG4 equivalent to the SVR4-compliant calls above:
   ```
   head -n 3
   tail -n 7
   tail -n +10
   ```
   This syntax is supported by "/usr/xpg4/bin/tail" and "/usr/xpg4/bin/head" and by the Linux definition of these commands; the SVR4 / Solaris "head" command ("/usr/bin/head") ALSO supports the XPG4 syntax, but the SVR4 "tail" command does NOT.
 - In the Solaris / XPG4 "head" command, the "-n" option only supports positive (unsigned) integers, whereas in Linux
   ```
   head -n -N
   ```
   means to print ALL BUT THE LAST N LINES - this functionality is absent in Solaris (SVR4 or XPG4).

So far, the original SVR4 syntax was also supported by the XPG4 and Linux versions of these commands (also the MacOS X commands support both syntax versions), therefore so far there was no stringent need to switch to the new syntax - HOWEVER, we just found that with RHEL 5.1, the SVR4 "-N" argument syntax is NO LONGER VALID - one MUST use the "-n" option syntax to specify the number of lines, see "man tail". In RHEL 5.1, the SVR4 syntax causes the "-N" argument to be interpreted as an additional file name, with two adverse consequences:

- there is an error message, indicating that the file "-N" was not found:
  ```
  tail: cannot open `-N' for reading: No such file or directory
  ```
  If "tail" is used with an input pipe, this would cause an additional error about ambiguous input specification.
- if a proper file name ("file_name") was specified in connection with "-N", this will be interpreted as multiple files, causing an extra line
  ```
  ==> file_name <==
  ```
  to be inserted at the top of the output.

This change will affect VnmrJ users in various ways:

- If you have macros and/or shell scripts using "head" and/or "tail" calls, you MUST switch to the new syntax. We will probably offer a facility that will make the old syntax still usable within macros (but not in shell scripts that are called from outside of VnmrJ) - but this support will NOT last forever, so better switch to the new syntax NOW!
- If you anticipate having to support RHEL 5.1 systems as well as systems running RHEL 4.0.x or older, you must still switch to the new syntax which is also supported by the older RHEL versions.
- if you don't plan on switching to RHEL 5.1 for the time being, it still is a good idea to use the new syntax, such that your software is future-proof.
- Badly, if you want your macros and shell scripts to be compatible with both Solaris AND all RHEL versions, you can NOT simply switch to the new syntax, but in Solaris you either need to stay with the old syntax, or you need to switch to the executables in "/usr/xpg4/bin" - i.e., you will need to add some "software switches" into such utilities.
- we will of course make new versions of VnmrJ compatible with the RHEL 5.1 "head" / "tail" syntax - but for current releases this not only causes VnmrJ-internal macros and shell scripts to fail where they use "head" or "tail" (this could be fixed by a VnmrJ patch) - it is very likely to cause the VnmrJ INSTALLATION to fail (as a matter of fact, several users have failed installing VnmrJ 2.1 or VnmrJ 2.2 under RHEL 5.1!) - and this cannot easily be fixed with a VnmrJ patch (unless we were to offer a Linux "pre-install patch" for VnmrJ). At this point we have not decided what we are going to do to make existing VnmrJ versions compatible with RHEL 5.1, and for which VnmrJ version(s) we will possibly offer such an installation fix. CONCLUSION: DON'T TRY INSTALLING existing VnmrJ releases on RHEL 5.x!

In the article below, the editor proposes possible solutions for writing macros and shell scripts with "head" and "tail" command calls that work under Solaris as well as all RHEL versions.

[Agilent MR News 2008-03-27]

WRITING UNIVERSAL SHELL SCRIPTS AND MACROS - "head" AND "tail":

As outlined above, you will need to change all macros and shell scripts that use calls to the UNIX / Linux "head" and "tail" commands if you want that software to work under RHEL 5.1. Within RedHat Linux you can simply switch to the new command syntax, but if you want your macros and shell scripts to stay back-compatible with Sun / Solaris systems, the necessary amendments are more complex. Here is a recommendation for shell scripts:

- to find all shell scripts that use "head" or "tail" commands, open a shell terminal and call
  ```
  cd ~/bin
  egrep 'head|tail' *
  ```
  If your "~/bin" also contains binary / compiled executables, you better use
  ```
  cd ~/bin
  egrep 'head|tail' script1 script2 ....
  ```
  i.e., list the shell scripts explicitly and avoid running "grep" or "egrep" on compiled executables, as this could produce binary output, possibly causing havoc in the current shell and forcing you to close / kill the terminal window.
- if you have shell scripts with "head" calls, simply switch these to the new syntax, e.g.: search for "head -" and change all calls
  ```
  head -1 my_file
  ```
  into
  ```
  head -n 1 my_file
  ```
  this also works under Solaris.
- For scripts using "tail", the fix is more complicated. Here's the editor's

```
   proposal (in Bourne or "bash" shell syntax). Near the top of the script use
        #!/bin/sh
        ...
        tail=tail
        if [ -x /usr/xpg4/bin/tail ]; then
          tail=/usr/xpg4/bin/tail
        fi
   If your shell script has a "Solaris / Linux compatibility switch" already,
   you can of course also use that, e.g.,
        tail=tail
        os=`uname -s`
        if [ `echo $os | grep -ci sunos` -ne 0 ]; then
          tail=/usr/xpg4/bin/tail
          ...
        fi
   Thereafter, change all lines calling "tail" from, e.g.,
        tail -3 my_file
   into
        $tail -n 3 my_file
   and all lines with
        tail +3 my_file
   into
        $tail -n +3 my_file
   i.e., switch to the new syntax and as command name use a local variable
   that either translates to "tail" (Linux) or "/usr/xpg4/bin/tail" (Solaris
   only). Note that "/usr/xpg4/bin/tail" is present in Solaris 10 and in all
   predecessor versions (we checked back to Solaris 2.4).
Equivalent constructs need to be added to macros using "head" or "tail" in
MAGICAL "shell" calls:
 - to find all macros that use "head" or "tail" commands, open a UNIX / Linux
   shell terminal and call
        cd ~/vnmrsys/maclib
        egrep 'head|tail' *
 - change all "head" calls to the new syntax, e.g.: change
        shell('head -'+$line, $file):$out
   into
        shell('head -n', $line, $file):$out
   (remember that commas between arguments in "shell" calls will be translated
   to blank spaces, whereas string concatenation using "+" suppresses such
   extra blank characters).
 - as in shell scripts, you will need to insert a switch for "tail" near the
   top of such macros, e.g.:
        $tail='tail'
        exists('/usr/xpg4/bin/tail','file','x'):$e
        if $e then
          $tail='/usr/xpg4/bin/tail'
        endif
   and below that, change all "tail" calls to the new syntax, e.g.:
        shell('tail -'+$line, $file):$out
   into
        shell($tail,'-n', $line, $file):$out
   and in case of calls with "+N" argument change
        shell('tail +'+$line, $file):$out
   into
        shell($tail,'-n +'+$line, $file):$out
   Again, note the subtle difference between concatenated strings and separate
   string arguments!
```

The editor (for for BioPack and his contributions in the "bin" and "maclib" subdirectories) and Krish Krishnamurthy (for Chempack 4.1) have started making the necessary adjustments in the Agilent MR User Library - this will take a while to be complete and tested. You will see indications of the associated updates if you look into "additions" - but at the moment (i.e., prior to the availability of a VnmrJ release that is compatible with RHEL 5.1) this alone is neither relevant nor (by itself) a reason to download and reinstall a contribution.

[Agilent MR News 2008-03-27]

2008-04-17:

COMMAND TRACKING IN VnmrJ:

If you experience a VnmrJ performance slow-down, and the system monitoring utilities (see the article above) indicate that VnmrJ itself (e.g., one of the "Vnmrbg" processes) is causing the problem, that alone isn't very useful information, as VnmrJ is a very complex utility, and it would be extremely helpful to learn what exactly VnmrJ is doing in such slow-down periods.

One option to see what commands are being executed (along with the execution hierarchy) is to use the macro debugging mode, see Agilent MR News 2005-02-27.

The idea here is that you turn macro debugging on, then call the macro or perform the panel action or graphic interaction that causes the slowdown, then switch the debugging mode off again and inspect the debugging record. This type of macro debugging has its limitations, e.g.:

- it does NOT show logical decisions
- it does NOT show parameter assignments, unless parameter changes cause the execution of a macro (e.g., an "underscore macro", see Agilent MR News 1997-09-19, Agilent MR News 1999-12-03)
- it does NOT show math operations on parameters and local variables
- it does NOT indicate / track mouse movements or graphical interactions (e.g., cursor repositioning, etc.), unless such actions cause the execution of a VnmrJ command or macro
- it is limited to actions performed by the active background "processing engine", i.e., macros and commands called via the command line, through parameter panels or through menu selections - it cannot be used to diagnose what is going on in the Java GUI, nor does it give ANY indications about what the Locator is doing.

On the other hand, this DOES allow you to follow which macros and commands are called (by which other macros, etc.), and (at least roughly) where things fall apart (indicated by commands or macros "aborted" rather than "returned").

There IS a utility that permits "watching VnmrJ beyond the scope of macro debugging" - though this is UNLIKELY to help you fixing the system yourself; but this may assist you in gathering valuable information that can be used by our software people and may help them to locate the source of a problem. This assumes that you have exhausted the debugging / tracking options discussed above and in the previous article, and that you have reasons to believe that VnmrJ itself or one of its components is causing the problem (or you are told by one of our support people to gather this information): e.g., while your system is "in crawling state", on the VnmrJ command line type

```
        vnmrjcmd('DEBUG add all')
```

to turn on VnmrJ "GUI debugging". Just let VnmrJ sit there for a few moments. You can use

```
        tail -f ~/vnmrsys/VnmrjMsgLog
```

to see whether debugging information is being added to the log file (use [Ctrl-c] to terminate the above "tail" command). Note that this debugging mode is EXTREMELY VERBOSE (especially if you have a "looping panel" problem or the like!), so don't leave it on very long. After a minute or two, turn the debugging mode off again using

```
        vnmrjcmd('DEBUG remove all')
```

DON'T try doing things with VnmrJ while collecting debugging information, as this will cause lots of additional logging which will clutter up the message log and will make it hard for our programmers to make any sense of that information. We do NOT expect users to be able to read / understand this message log - this is meant to be sent to our support group (primarily when asked to do so), to assist us in locating / debugging specific problems.

Note that the above "vnmrjcmd" calls (and the contents / syntax of the debugging output) are NOT (and are unlikely ever to be) documented: a normal / typical user is unlikely ever to use "vnmrjcmd" directly. Also, the contents of the debugging trace are to be regarded as proprietary / subject to the VnmrJ licensing terms and may therefore NOT be disclosed to third parties.

Thanks to Glenn Sullivan (Varian, Ft.Collins) for providing information on using "vnmrjcmd" for VnmrJ tracking.

[Agilent MR News 2008-04-17]

2008-04-23:

COMMAND TRACKING IN VnmrJ - ADDENDUM:

In the last issue (Agilent MR News 2008-04-17) we mentioned "vnmrjcmd" command calls for enabling and disabling a very verbose tracking mode for VnmrJ, intended for use with specific debugging tasks. Think of this as a last resort for collecting information that might help locating certain Locator database bugs, or bugs in the Java-based GUI, in instances where the information provided by standard output and error log files (e.g., the files "~/vnmrsys/VnmrjMsgLog" for logged VnmrJ messages, "~/vnmrsys/ManagedbMsgLog" for database message logs) and message windows in the VnmrJ GUI is not sufficient.

Note that the two particular "vnmrjcmd" calls we referred to have been implemented in VnmrJ 2.2C (and later) ONLY and will NOT work with earlier VnmrJ releases. And again: this debugging mode is EXTREMELY verbose and should be enabled for short periods (minutes AT MOST) ONLY.

[Agilent MR News 2008-04-23]

2008-05-09:

NAMING RESTRICTIONS IN VnmrJ / MAGICAL:

Even though VnmrJ is essentially written for the Linux / UNIX operating system, it comes with its own set of naming rules that is (in parts, at least) more restrictive than the naming conventions of the operating environment in which it is running (as discussed the article above):

- for files created on any hard disk, the naming restrictions of the operating environment controlling that disk are all applicable (see the article above);
- for the command line, macro / menu and parameter panel syntax, VnmrJ uses its own command interpreter; although that interpreter has similarities with certain UNIX / Linux shells, it uses an entirely different lexical structure, and hence imposes its own set of naming restrictions, especially in the area of macros (which are text files on disk);
- pulse sequence names exist as source files, as well as UNIX / Linux executables in the "seqlib" directory, and are typically associated with macros, parameter files, panel layouts and other files, using file names derived from the sequence name, and hence share the same, "internal" naming restrictions;
- macros are involved in many hidden areas of VnmrJ (some of these macros may never be called directly on the command line), hence the macro naming restrictions also affects other, possibly unexpected areas (one such example is the setup of VnmrJ printers / plotters, see below).

The VnmrJ-specific file naming restrictions are essentially tied to the VnmrJ (and VNMR) command interpreter / language, MAGICAL. Even though it resembles some UNIX shell languages in aspects such as control structures (branching, looping) and the local scope of temporary variables, MAGICAL is different in many ways because it has been optimized for parameter handling, calculations and logical operations in MR experiment setup and processing:

- In most shell languages, variables are typeless (i.e., they may consist of digits only and may have a numeric appearance - but the shell still handles them as strings), but in MAGICAL, there is a strict separation between numeric and string type parameters (both types may be either single values, or they may represent an array of values).
- Most UNIX shells only feature very rudimentary math support (in the Bourne and "bash" shells only integer math is supported, and only via the "eval" command, or through utilities such as "awk" / "nawk" / "gawk"); in MAGICAL, variable assignments, math / in-line calculations, decisions and string manipulations based on a large set of tools and logical operators, etc. are a key part of the language.
- Commands are NOT separated by a semicolon, but are either recognized as stand-alone token, e.g.:
```
     ds
```
  as a token with arguments in parentheses, e.g.:
```
     write('error','this was a mistake!')
```
  as a token with one or several "return arguments", with or without standard arguments in parentheses, e.g.:
```
     nll:$numlines
     integ(2p,5p):$int
     getll(1):$ht,$freq
```
- Commands can either be
  - features built into the interpreter (e.g.: "abort", "aborton", "abortoff", "break", etc., see the User Programming manual, chapter 1),
  - commands built into VnmrJ / VNMR, or
  - macros in one of several "maclib" directories, within which the order of precedence is defined via the "maclibpath" and "appdirs" mechanisms.
- Different from most UNIX shells, there is no facility to define subroutines

(functions / procedures) within macros, but macros may call other macros without practical limitation in nesting depth; also recursive calls are permitted (UNIX / Linux shell scripts can do this as well).
- MAGICAL control structures and commands built into VnmrJ take precedence over macros with the same name (so you can't accidentally override / deactivate a command built into VnmrJ). Note that the MAGICAL command interpreter is case-sensitive.

The above design features, along with the need to provide a clear distinction between string and numeric values, as well as between parameter names and the associated parameter values, lead to additional restrictions in the naming of commands, i.e., newly created macros, as well as names of parameters:
- As outlined in section 1.2 "Programming with MAGICAL" in the VnmrJ User Programming manual (under "Identifyers"), macro names may ONLY contain letters (upper and lower case alphabetic characters, i.e., "[A-Za-z]"), digits ("[0-9]"), and the characters "_" (underscore), "$" and "#", whereby the first character MUST be a alphabetic character or a "_").
- The very same restrictions ALSO apply to parameter names.
- If a macro has the same name as a built-in VnmrJ command, the latter will be executed, NOT the macro, see above.
- It is POSSIBLE, though, to replace / override a built-in command by a macro, using the "hidecommand" utility (see Agilent MR News 2002-03-23): this will alter the name of the built-in command by changing the first character from lower to upper case (and vice versa).
- Remember that MAGICAL is case-sensitive (this is instrumental for the "hidecommand" trick); hence, "COSY", "cosy", "Cosy", and "coSy" are ALL DIFFERENT commands and are only recognized by their exact spelling. Note, however, that from a more global perspective it is probably not a very good idea to use this feature (i.e., have multiple macros with names that only differ in their upper/lower case spelling) extensively, considering that some operating environments (such as MacOS X and MS Windows) are typically NOT operating with case-sensitive file systems.
  VnmrJ actually already includes a fair share of such "case mix-up" instances; on the operating environment platforms that we have used so far (SunOS, Solaris, Linux) this has not been an issue, and even in MacOS X, VnmrJ works OK without switching to case-sensitive mode / file systems - but this is only because VnmrJ for MacOS X is just provided for processing (i.e., not for spectrometer hosts), and macro names that only differ in their upper/lower case spelling are essentially restricted to experiment setup and VnmrJ protocol handling (i.e., acquisition-related tasks).

The VnmrJ naming restrictions primarily concern macros and parameters, but this has subtle implications in (possibly unexpected) other areas:
- VnmrJ pulse sequences are typically associated with macros that have the same name, hence even though the sequence compiler and the Linux / UNIX file handling tolerate the less restrictive UNIX naming, it is highly recommended to apply the macro naming restrictions also to pulse sequence names.
- In Linux / UNIX, logical printer names such as "LaserJet-office" and "color-lab1" should work OK; in VnmrJ, selecting a plotter, or switching between plotters involves calling a macro with the name of the plotter - and therefore the "VnmrJ-internal naming restrictions" for plotters and printers are the SAME AS FOR MACROS. Note that currently, this is NOT explicitly mentioned in the documentation, and the VnmrJ printer / plotter setup utilities ("/vnmr/bin/adddevices", or the User Library contribution "bin/makePrinter") DON'T include comprehensive testing to prevent defining printers / plotters with "illegal" names (see bug report "vnmrj.j2204"). In environments with shared printers that are controlled by an IT department you may need to ask for an alternate (parallel) device definition that uses a "VnmrJ compliant" name. Strictly speaking, this restriction probably applies to plotters only (through the VnmrJ macro "loadcolors"), NOT to printer names.

[Agilent MR News 2008-05-09]

2008-06-18:

APPLICATION DIRECTORIES IN VnmrJ 2.2C VS. "MACLIBPATH":

Historically, the VNMR software was based upon two principal "working directories", the main VNMR software directory (accessed through the symbolic link "/vnmr"), and the user's local "~/vnmrsys" directory, whereby the latter (with subdirectories such as "maclib", "manual", "menulib", "psglib", "seqlib" etc.) takes precedence in the case of subfiles or subdirectories that have the same name as an entry in the corresponding subdirectory in "/vnmr". This allows users to customize many of the VNMR software features, besides adding local / personal features such as pulse sequences, macros, menus and the like.

On top of that of that, in our software versions up to VnmrJ 2.2B, we offered a "path" feature that permitted users to access a set of additional directories (defined through optional global parameters such as "maclibpath", "manualpath", "menulibpath") with a priority between the local files and those in "/vnmr". This offered the opportunity to share features among users WITHOUT placing customized files in "/vnmr" (hereby mixing original and customized files). See the VnmrJ documentation for more information (for a discussion of "shimspath" see also Agilent MR News 2000-12-15).

On top of that, we later added a set of global "systempath" parameters such as "syshelppath", "sysmaclibpath", and "sysmenulibpath" that were used in connection with the imaging and Autotest "appmode" settings, in order to insert yet another set of files / features in-between the optional path directories as just discussed and those in "/vnmr". This permitted switching to the Autotest of imaging modes on a per-user basis, again without altering the contents of the standard software directories.

In the case of macros, VnmrJ offers utilities to determine where the ACTIVE version of a given macro is located:

```
        exists('macro_name','maclib'):$e
```

will not just return 0 (not found) or 1 (found) as for some of the other search types, but an integer indicating the location of the active macro:

```
        1 = active macro located in local "maclib"
        2 = active macro located in directory pointed to by "maclibpath"
        3 = active macro located in directory pointed to by "sysmaclibpath"
        4 = active macro located in "/vnmr/maclib"
```

This is typically used in macros; for interactive use you can simply type

```
        which('macro_name')
```

which will return the path to the active macro.

While this "path approach" is a useful feature, it has its limitations, in that a) path (parameter) definitions had to be made for each of the supported subdirectories, and b) on the other hand, important subdirectories such as parameter panel definitions, but also "psglib" were NOT supported by a path mechanism, preventing pulse sequences from being shared this way - see also Agilent MR News 1996-08-28 and Agilent MR News 1999-04-01.

Starting with VnmrJ 2.2C we are now offering the new, much more powerful and flexible "appdir" approach that is controlled through a dedicated GUI, called via "Edit" -> "Applications...". This offers various advantages and new features compared to the above, preceding approach:

- Instead of using separate entries for software subdirectories such as "help", "maclib, "manual", "menulib", or "shims", the user specifies the PARENT DIRECTORY such as "/vnmr/chempack" or "/vnmr/biopack"; this then covers ALL SUPPORTED SUBDIRECTORIES (as present in that parent directory). NOTE: specifying individual SUBDIRECTORIES (as previously done with the "maclibpath" and related parameters will FAIL: in order to access vnmr1's local "maclib" users must NOT specify "/home/vnmr1/vnmrsys/maclib" in the "Applications" GUI, but just "/home/vnmr1/vnmrsys".
- Compared to the earlier "path method" (discussed above), the "appdir" approach covers additional directories such as "templates/layout" (VnmrJ parameter panel templates) and "templates/vnmrj" (VnmrJ menus, vertical panel definitions, etc.); note, however, that certain restrictions still exist, see below.
- Different from the earlier approach the number of path entries is NOT FIXED: new entries can be inserted at user-defined points in the "appdir" path - but watch out for potential mis-settings:
- The search order for the application directories such as the user's local "vnmrsys" ("USERDIR"), "/vnmr/chempack" (Chempack 4.1), "/vnmr/biopack" (BioPack), "/vnmr", and "/vnmr/autotest" is NOT fixed, but can be set up by the user - but keep in mind that this search order is NOT arbitrary, but may affect the functionality of user generated software, software packages such as BioPack or Chempack 4.1, and/or VnmrJ altogether.
  The default setup is such that the user's directory is searched first, followed by packages such as BioPack or Chempack, followed by "/vnmr" (eventually followed by AutoTest. As long as there is no naming overlap / conflict, the order is indeed irrelevant, but the above standard setup allows packages such as Chempack or BioPack to modify standard VnmrJ features such as parameter panels, macros, menus (and therefore BioPack and Chempack 4.1 MUST be placed ahead of "/vnmr" in the search order), and

```
   the user again has the ability to modify the resulting software feature set
   with local files (leaving "/vnmr" and package installations such as BioPack
   untouched).
   For some situations one could think of an "appdir" setup where "USERDIR" is
   moved BEHIND BioPack, possibly even behind "/vnmr": this would still permit
   developing local macros etc. as long as NEW NAMES are used, but that user
   could NOT easily alter standard VnmrJ (or BioPack etc.) software features
   (macros, menus, parameter panels).
 - Once a new entry has been added to the path, it can be deactivated WITHOUT
   deleting the definition - this makes it very easy to switch from BioPack or
   Chempack 4.1 back to standard VnmrJ software, or vice versa.
 - With the "appdir" method we have expanded the "path" approach to additional
   subdirectories (see above) - but note that the current implementation is
   still restricted to software features that are "purely VnmrJ-internal",
   i.e., it does NOT cover the "bin" subdirectory (accessed through Linux /
   UNIX command path) nor any "expN" directories, "global", "psg", "psglib"
   (as "seqgen", "psggen", "fixpsg" are based on Linux / UNIX path and library
   search definitions that cannot be easily adjusted "on-the-fly" from within
   VnmrJ, especially considering that these utilities can also be accessed /
   called from a shell window, without directly involving VnmrJ), see also
   (non-)bug report / suggestion "seqgen.j2201".
The above "exists" and "which" calls can still be used, but have been adjusted
to support the "appdir" approach:
        exists('macro_name','maclib'):$e,$path
now returns 0 (not found) or the search order index (top down, typically
"USERDIR" = 1) as first return argument, and the active, full macro file path
as second return argument, see "man('exists')" for more information.
  Whenever possible we try to avoid disabling / discontinuing software
features, as this very often breaks user-defined software utilities; however,
in the case of "maclibpath" / "sysmaclibpath" and related features it is
obvious that these are conflicting with the "appdir" approach, and therefore,
starting with VnmrJ 2.2C, all those "path" parameters are indeed now INACTIVE.
You may of course still keep / have these parameters in your global parameter
tree (e.g., for the unlikely case where you were to switch back temporarily to
an earlier version of VnmrJ) - they just won't do anything in VnmrJ 2.2C or
later software versions.
                                                [ Agilent MR News 2008-06-18 ]

MAJOR ENHANCEMENTS IN THE ACCOUNTING SOFTWARE FOR VnmrJ 2.2C:
(by Frits Vosman, Varian)

  We have just published a new patch "2.2CallLNXall402" for Varian NMR Systems
and 400-MR spectrometers running VnmrJ 2.2C. That patch (196 KBytes) can now
be downloaded via our "VnmrJ / VNMR Patches" Web site at
        http://www.varianinc.com/products/nmr/software/patches/
This is a relatively small, stand-alone patch (i.e., its installation is
independent of the presence of any other patches for VnmrJ 2.2C); we will
probably incorporate this into an upcoming, regular patch for VnmrJ 2.2C.
  Patch "2.2CallLNXall402" includes a new, substantially enhanced version of
the VnmrJ accounting software, adding the following features:
 => Fixes bug "vnmr_accounting.j2201": operator tracking is made more robust,
    the printing routines were reworked.
 => Now covers three types of system use tracking:
      - Accounting by acquisition time:
        This the time from when the experiment starts until it is completed,
        including the time for post-acquisition processing. This is tracked
        through a log file "/vnmr/adm/accounting/accounts/goes.txt"
      - Accounting by VnmrJ usage (previously called "login/logout use"):
        This tracks the time for which the owner keeps VnmrJ running, and for
        which an operator stays logged into VnmrJ, i.e., we track operator time
        during walkup mode/use, but now also "owner time", both in walkup and
        in experimental mode. If more than one copy VnmrJ is opened (e.g.,
        through a remote terminal), both are tracked and invoiced. Note that
        this is still the only way to track VnmrJ operator time. This type of
        system use is logged in "/vnmr/adm/tmp/macros.txt".
      - Accounting by Linux login/logout (NEW):
        Here we use standard Linux (operating system) login tracking through
        a log file "/vnmr/log/wtmp". IMPORTANT: the file "/var/log/wtmp" must
        exist for the tracking of Linux login time to be active. As root, use
                 touch /vnmr/log/wtmp
        to create the file if necessary. The accounting software itself does
        not provide a way to create/enable/disable this file. Linux tracks the
        current and the (one) previous month, older information is cleaned up
        automatically. Write your invoices on time: information is deleted at
        midnight on the 1st every month! Linux login time is tracked both in
        walkup and in experimental mode.
        When tracking login/logout time, Linux creates entries for each
        terminal opened (shell window, remote login); the invoice features only
        one charge for the longest entry (i.e., the first login, the last
        logout / exit), overlapping entries are ignored. Linux may also create
        log entries which are incomplete (missing logout) - these are ignored.
        The VnmrJ accounting software creates a file "/vnmr/adm/tmp/last.txt"
        to generate an invoice. Unlike "/var/log/wtmp", this is an ASCII file
        that can be inspected with any text editor.
 => A new option has been added to write the accounting output into a CSV
    (Comma Separated Value) file which can then be imported into an MS Excel
    spreadsheet for further processing. This can be done for an individual
    invoice, or  all invoices can be written to a single CSV file. The
    creation of CSV files is triggered by the existence of a flag file
    "/vnmr/adm/accounting/accounts/CSV_yes" which can be created by vnmr1,
    using the command
        touch /vnmr/adm/accounting/accounts/CSV_yes
    To return to normal printing delete the above file, or rename it to
    "CSV_no".
As any other VnmrJ patch, this patch must be installed when logged in as the
VnmrJ administrator (typically vnmr1).
                                                 [ Agilent MR News 2008-06-18 ]
```

2008-09-17:

REFRESHER ON PROCESSING TIME-SHIFTED FIDS:

Over the past 12 - 15 years, Varian has gone through considerable efforts in trying to provide hard-, firm-, and software permitting to produce FIDs that are NOT time shifted (hence avoiding baseline roll from d.c. offsets in spectra with large first order phase correction) and which at the same time avoid baseline distortions from unwanted amplitude modulations in the first part of the FID (such as from time-dependent audio filter response). The proper recipe for achieving this requires parameter adjustments that depend
- on the instrument architecture (DirectDrive vs. UNITY INOVA), and
- on whether and how DSP is used (on the UNITY INOVA).

However, this is NOT the topic of this article (the editor hopes to provide recipes / refreshers on this in future issues of Agilent MR News).

In this note, we assume that bad data have already been acquired, and that "normal" processing (such as autophasing) fails, and even if you manage to process, you are still fighting a messy baseline - and there is no possibility to reacquire the data with adjusted acquisition parameters for better results. Possible reasons for such results (featuring time shifted FIDs) include
- an operator taking the "quick & dirty approach" to avoiding baseline artifacts from pulse breakthrough (e.g., with low gamma nuclei) or from probe background (either inherent - see Agilent MR News 2002-02-02, or due to probe contamination from a broken sample or from using dirty sample tubes etc., possibly also from solid particles in a liquid sample; the most likely (improper) option taken in these cases would be that "rof2" was set to rather / very large values (possibly the equivalent of many dwell times).
- similarly, an operator trying to avoid acoustic ringing with low frequency nuclei by increasing "rof2" in lieu of trying a pulse sequence that cancels or avoids the effects of acoustic ringing;
- the "rof2" and/or "alfa" parameter being mis-set by accident.

In all these cases the most likely (or inevitable) consequence is that you end up with a FID that is not "synchronized with the NMR coherences". Ideally, a FID should start when all coherences are aligned on the same axis in the X,Y plane. This point is theoretically positioned at the end of an infinitely short RF pulse (or on the "top" of a spin echo); however, finite pulse lengths cause if to move "into" the RF pulse (due to precession during the RF pulse) - and on top of that, for the receiver it is time-shifted due to the effect of the audio filter. In a proper experimental setup, "rof2" / "alfa" are adjusted such that the FID is "timed properly" - here, however, the start of the free induction decay is long over by the time the FID acquisition starts.

A typical consequence of such bad FID timing is that the resulting spectrum cannot be phased easily; autophasing ("aph", often also "aph0") typically fails ("spectrum unsuitable for autophasing"), the baseline may be messy to start with. Luckily, in most cases it is still possible to obtain reasonable (albeit probably not perfect) output from such data. Two basic steps are usually required for this: first, you need to get the phasing right, and after that you will need to "fix the baseline". For the first step you have two options: either

- work with very large first order phase corrections ("lp" << -360), or
- shift the FID using "lsfid" with negative values.

The latter right-shifts the FID by the specified number of complex points, hereby discarding the right-most part of the FID (unless "np" < "fn").

In some situations, the "lp" approach is regarded superior: it is equivalent to a circular shift of the data in the FID, hence no data points are lost (you are not "discarding experiment time"). However, in the above cases you are mostly just discarding noise (unless the acquisition time was too short), plus you will need to reconstruct the first part of the FID (lost due to the excessive delays) anyway, hence a circular data shift would have no advantage.

On the other hand, you may still want to start with phasing the spectrum manually in order to get an idea about the number of points by which you'll need to shift the FID:

- first either use "aph0" or manual phasing to get the largest signal in phase, next,
- the first order phase adjustment will usually take multiple steps: for a first, coarse adjustment select and phase an isolated signal NEARBY; this adjustment may lead to negative "lp" values of several thousand degrees. Then you may want to try adjusting the phase of signal at the opposite end, though for a rough estimate this may not be necessary. That last step may not be easy anyway, as for every 360 degree increment / decrement off zero you get one full sine wave into the baseline (from d.c. offset in the FID), and the more off you are, the more intense these baseline waves will be - sometimes to a point where they dominate the spectrum, and the real signals may barely be visible at all.

Note that by default, "lp" is limited to values between -3600 and +3600; for the second step to work you may need to expand the parameter limits using

```
        setlimit('lp',1e8,-1e8,0)
```

The manual phasing primarily serves to give us an estimate for the value of "lsfid" required to get the timing of the FID approximately "right". If you could re-acquire the FID, you would need to time-shift the FID by "lp/360" dwell times, i.e., you would effectively need to shorten the sum of "rof2" plus "alfa" by

```
        lp/360*(1/sw)
```

in seconds (where "1/sw" is the "dwell time" between sampling points). This is what macros such as "calfa" and "crof2" will help you achieving (assuming a phase-corrected trial spectrum).

Here, we discuss processing, so let's assume you get a value of "lp=-4630". Now, simply type

```
        lsfid=lp/360 lsfid?
```

and you will see that "lsfid" has been set to -13. This assumes "lsfid" was set to 0 beforehand; for a more generic recipe you could use

```
        lsfid=lsfid+lp/360
```

Now we can expand the above construct to reprocess the FID and re-adjust the first-order phase:

```
        lsfid=lp/360 wft lp=lp-lsfid*360
```

i.e., we have now shifted the FID such that "lp" should be within the limits of -180 .. +180 ("lp=50" in this example).

However, processing the time-shifted FID using "lsfid" (AND readjusting "lp" accordingly) will essentially yield the same result as just using a very large first order phase correction; your time-shifted FID starts with noise ("lp" much less than -360) or with "-lsfid" empty data points, both causing serious baseline distortions.

One way of eliminating baseline distortions is through baseline correction, i.e., "bc"; however, in the case of such heavily time-shifted FIDs, "bc" will usually yield very unsatisfactory results, even when tweaking the arguments (see the articles in Agilent MR News 1996-06-25 and Agilent MR News 2008-01-16). The only viable option for such cases is to reconstruct the first part of the FID using linear back-prediction, see the article below.

A last remark: remember that the above recipe is specific to FIDs where the data sampling starts TOO LATE - this can't be compared to processing data from instruments / manufacturers using a DSP that produces FIDs which are NOT time corrected: here, the sampling appears to start TOO EARLY, hence positive values for "lsfid" (i.e., left-shifting) would be needed. In this particular case it is usually preferable NOT to shift the FID, but to use very large "lp" values (positive values: "lp=lsfid*360") for circular FID shifting, in order to AVOID discarding the leftmost part of the FID (see also Agilent MR News 1997-04-08 and Agilent MR News 2002-03-10).

[Agilent MR News 2008-09-17]

REFRESHER ON LINEAR BACK-PREDICTION:

Linear prediction (LP) by now is a well-established technique, often used through macros or point-by-point recipes, without even spending much time thinking about what one is doing in all detail. This isn't necessarily bad - it mainly means that LP is fairly robust and reliable: it just WORKS. This is not just simply "normal" or to be expected, given the fact that it is a mathematical procedure (algorithm) that can (and occasionally DOES, see below) fail - plus, there are various, competing LP algorithms in use (see also Agilent MR News 1998-02-28). The robustness of LP in VnmrJ is simply due to the fact that we had a good implementation to start with (indeed, over many years we haven't changed much, if anything in that software) and that the recipes that we posted many years ago "keep LP on the safe side" for the vast majority of the cases. If you want to read these recipes, you find them in Magnetic Moments V.2, p.11 (1992), see

    http://www.varianinc.com/products/nmr/apps/magmo/V.2/art_i.html

and in Magnetic Moments VII.1, p.30 (1995), see

    http://www.varianinc.com/products/nmr/apps/magmo/VII.1/art_o.html

You can also find these articles (plus two follow-up articles by users) by visiting our "User Pages" at

    http://www.varianinc.com/products/nmr/apps/

then selecting "Publications" -> "Magnetic Moments", and in the alphabetic keyword and subject index select "L".

One should also note that LP is very often used to back-predict a few data points in otherwise well-behaved FIDs, or to expand indirect dimensions in nD spectra (where there are typically only very few signals per trace), or maybe occasionally for "repairing" a bad increment in nD data (see Agilent MR News 2006-10-31). These applications are usually not challenging the algorithm to its limits.

This may be different in the case(s) described in the article above: here, the FID may not always be well-formed, and hence the LP parameters may require some tweaking. In the following section we use that example - a FID that has been right-shifted by 13 data points ("lsfid=-13"). Linear prediction (back prediction in this case) is done after such FID-shifting and is then followed by weighting & subsequent standard Fourier transformation. With "lsfid=-13" we need to reconstruct the first 13 (missing) data points - AT LEAST, because additional points in the FID are likely to be distorted: in back-prediction we use the information from "lpnupts" existing ("source") data points (starting at position "strtlp") to predict "lpext" points (up to position "strtext"), and if the "source points" are distorted, this is not going to help (it can easily cause LP to fail!). The following command line settings can be used in the editor's example:

```
        parlp lpfilt=32 lpnupts=np/4 strtlp=16
        lpext=16 strtext=16 proc='lp' lpopt='b'
        wft
```

In the VnmrJ parameter panel GUI, the equivalent would be

```
        prediction: back
        coefficients: 32
        basis points: [np/4]
        starting at: 16
        predicted points: 16
        starting at: 16
```

Some explanations on this parameter selection:

- Because of distortions in the first "real" points in the FID we predict 16 rather than just 13 points - in fact: with the sample data used by the editor, predicting just 13 points caused the linear prediction to fail, producing completely unusable output.
- It is NOT forbidden to use a data point as "source" for LP which is at the same time recalculated and replaced (by a predicted value) by the very same process - as done with point #16 in the above procedure.
- One rule in LP is that in general, the number of coefficients ("lpfilt") should be roughly comparable to the number of lines in the spectrum;

however, when we back-predict the first 16 data points, we are focusing on spectral (baseline) "features" that aren't narrower than 1/16th of the spectral window, hence "lpfilt=32" should be sufficient. Note that the processing time scales with "lpfilt^3" (see Agilent MR News 1998-02-28), i.e., in nD linear prediction you should be careful with using large values for "lpfilt". The editor found "lpfilt=16" to work as well, but "lpfilt=32" definitely gave better results with the sample data used.
- "lpnupts" (the number of "source" data points) should be AT LEAST twice the value of "lpfilt"; here, we may be dealing with NOISY DATA (such as a low signal/noise C13 FID), therefore it is preferable (or even necessary) to use a larger portion of the FID (half the acquired FID in this case). Note that increasing "lpnupts" does not "cost" as much in processing time as increasing "lpfilt". Note also that "lpnupts" is complex points, whereas the parameter "np" indicates real numbers, i.e., "lpnupts" can't be bigger than "np/2" (actually, "np/2 - strtlp - lsfid", to be accurate).
- Note that (as outlined in the article above) in this case we don't end up with "lp=0", hence intensity distortions in the first data point will still lead to a bow in the baseline. You should make sure "fpmult='n'" or "fpmult=0.5" to avoid problems from that side - even though the rule of setting fpmult=0.5" for linear prediction possibly only applies to cases where "lp" is close to zero.

The editor tried all this on a C13 data set; he was able to produce a spectrum that didn't require ANY further baseline or DC correction.

Thanks to Hsin Wang (CUNY, City College) for suggesting the topic for these two articles, and for providing the sample data set!

[Agilent MR News 2008-09-17]

2008-09-24:

NANOPROBE COMPATIBILITY OF GRADIENT EXPERIMENTS IN VnmrJ:
(by Peter Sandor, Varian)

For optimum performance of pulsed field gradient experiments on a NanoProbe the encoding and decoding gradient durations need to be fine adjusted to ensure that the duration of each gradient pulse corresponds to an integer number of rotations.

In VnmrJ 2.2C (using the "setnano" macro) attempts were made to achieve automatic gradient pulse adjustments for NanoProbe applications. However, this only worked for standard VnmrJ pulse sequences and required following very specific rules in parameter naming, and its concept was not compatible with manual spinning control of the NanoProbe - the only option for consoles not based on the DirectDrive architecture (UNITY INOVA, MERCURY family).

The latest version of Chempack 4.1 which is now available from the Online User Library via

        http://www.varianinc.com/products/nmr/apps/

and as part of the upcoming VnmrJ 2.2D provides a generic solution for the problem - for the time being limited to the framework of the Chempack 4.1 application directory. It can handle both automatic and manual spin control, and is therefore applicable not only to the DirectDrive architecture (400-MR, Varian NMR System), but also to UNITY INOVA, MERCURY-Vx and MERCURYplus.

Once the probe file is set up properly the user hardly needs to do anything to run NanoProbe-compliant experiments:
- for consoles with software spin control (400-MR, Varian NMR System) everything is automatic, while
- for systems with manual spin control, prior to starting the acquisition, the parameter "srate" needs to be set to the actual spinning frequency, either in the command line or in the "Start/Standard" parameter panel.

This software setup relies upon some new probe file entries: there may be up to three NanoProbe related lines / definitions in the probe file:

```
        Probeprobetype          nano
        Probespintype           tach
        Probespinmax            3000
```

corresponding to VnmrJ parameters "probetype", "spintype", and "spinmax". The first one, "probetype", is a new global parameter. The "addprobe" macro now adds this new parameter to the probe file (with a default value of "liquids"), the "updateprobe" macro will add that definition to an existing probe file if it is not present yet. Alternatively you may edit an existing probe file and add the first of these lines exactly as shown above. The other two parameters ("spintype" and "spinmax") do exist since VnmrJ 2.2C and are only relevant for systems with software spin control. Upon switching to a NanoProbe, all three parameters are activated (via the "_probe" macro) and the system is ready to do the extra, NanoProbe related tasks automatically.

This software setup requires a few changes to existing pulse sequences: the actual gradient adjustment takes place within the pulse sequence itself, i.e., is performed at "go" time (through the "Acquire" button or the "cpgo" macro). It is therefore the pulse sequence programmers privilege and responsibility to ensure that the pulse sequence is NanoProbe compliant, see the article below.

[Agilent MR News 2008-09-24]

PULSE SEQUENCE CHANGES FOR NANOPROBE COMPATIBILITY (by Peter Sandor, Varian):

All relevant Chempack 4.1 pulse sequences have been adjusted to be compliant with NanoProbes. If have your own gradient pulse sequence and want to make it NanoProbe compatible, the following steps need to be performed

- Be sure that the probe file has all relevant parameters defined and set, as outlined in the article above.
- In VnmrJ versions prior to VnmrJ 2.2C, install the current version of Chempack 4.1 ("chempack/CP4") from the on-line User Library at
  http://www.varianinc.com/products/nmr/apps/
  In VnmrJ 2.2C, you may need to update the Chempack 4.1 installation (also by downloading and installing "chempack/CP4" from the above address) in order to ensure you are running a version of Chempack dated 2008-09-22 or later.
- In VnmrJ 2.2C or the upcoming VnmrJ 2.2D software, activate the "chempack" applications directory. This is necessary because the relevant macros and parameter panels are only available through Chempack 4.1.
- Include the necessary changes in your pulse sequence code, as described below, and recompile the pulse sequence.

Making an existing gradient pulse sequence compatible with NanoProbes (as discussed in the article above) involves several changes in the pulse sequence code itself:

- Near the top of the pulse sequence, just below the line
  ```
  #include <standard.h>
  ```
  you need to add an extra "include" line for the header file "chempack.h":
  ```
  #include <chempack.h>
  ```
- You need to define which parameter(s) need adjustment by inserting expression "A" (or "A" and "B" together) from below. Note that a homospoil gradient pulse (i.e., a gradient that simply destroys residual transverse magnetization) does NOT need this type of adjustment. Let's assume that the relevant gradient pulse has a duration of "gtE" and an amplitude of "gzlvlE". In this case, expression "A"
  ```
  gtE = syncGradTime("gtE","gzlvlE",1.0);
  ```
  trims the gradient pulse length "gtE" and leaves the amplitude "gzlvlE" unchanged. Each NanoProbe compatible pulse sequence must contain AT LEAST this definition. The third argument is a multiplier that is typically set to 1.0 in sequences with a single gradient pulse duration. If a pulse sequence uses gradient pulses with lengths of both "gtE" and "gtE/2" (as many heteronuclear Chempack sequences), then the multiplier must be set to 0.5 to ensure that the requirements are also met for "gtE/2". A second expression "B"
  ```
  gzlvlE = syncGradLvl("gtE","gzlvlE",1.0);
  ```
  used with a combination with expression "A" above adjusts also the gradient amplitude ("gzlvlE" in this example) such that the gradient area (i.e., the product "gtE*gzlvlE") remains constant.
  The use of both expressions is REQUIRED for all sequences used to measure diffusion rates.
  IMPORTANT: in order to avoid any incompatibilities between current VnmrJ DOSY pulse sequences and the DOSY processing package expressions "A" and "B" must be inserted AFTER the line starting with:
  ```
  putCmd("makedosyparams...
  ```
  (Every Varian-supplied DOSY sequence contains such a line).

Statements "A" and "B" above actually do NOT update the parameter values in any VnmrJ parameter tree, i.e., the gradient pulse duration (and amplitude, if applicable) are adjusted "on-the-fly", the output of "dps" shows the modified values, but after the experiment the VnmrJ parameters will NOT reflect the values actually used. However, this will not have any negative consequences, at least as long as both the gradient pulse duration AND the amplitude are corrected, as the gradient areas in the "real" experiment correspond to the specified values. In principle you COULD use the "putCmd" utility to feed back corrected parameter values into VnmrJ (see Agilent MR News 2005-01-22), but this would be an unnecessary complication here - plus, making this work for arrayed parameters (such as gradient amplitudes in diffusion experiments) is fairly tricky.

[Agilent MR News 2008-09-24]

2009-01-29:

VnmrJ 2.3A RELEASED FOR ALL APPLICATIONS ON Dell OptiPlex 755N / RHEL 5.1:
(by Mark Dixon, Varian)

Last year (see Agilent MR News 2008-07-01) we announced the release of VnmrJ 2.3A for Imaging applications only. Starting this week, all new systems are being shipped with the VnmrJ 2.3A media kit. There are two driving forces behind this update in our software offering:

- It simplifies our offering and brings all current NMR and MRI system platforms onto a single software release, which also helps by simplifying maintenance and administration: the current release for imaging systems with Dell OptiPlex 755N host is VnmrJ 2.3A (see above), while for non-imaging systems with DirectDrive architecture (Varian NMR Systems, 400-MR) so far were restricted to VnmrJ 2.2C, running under RHEL 4.0u3 on Dell Precision 370N, 380N and 390N PC models, and under RHEL 5.1 on the Dell OptiPlex 755N (see Agilent MR News 2008-07-15).
- Since we started shipping OptiPlex 755N PC models with new systems in late June 2008, all non-Imaging application users were shipped VnmrJ 2.2C, up until today. However, we have since discovered an incompatibility with the console communication software, affecting systems running VnmrJ 2.2C under RHEL 5.1. The symptom is a random but rare aborting of an acquisition with an "Acode error". Complex pulse sequences generating lots of Acode (acquisition controller instructions), such as many BioNMR, and Solids NMR sequences, long acquisitions of any type, etc. are statistically more likely to encounter this problem than simple (1D/2D) pulse sequences. Fortunately, the issue had already been resolved in VnmrJ 2.3A as we upgraded the console communication software in that release which became available even before we moved to RHEL 5.1 and the OptiPlex 755N (see again Agilent MR News 2008-07-15).
  Unfortunately, the nature of the issue is such that a patch for VnmrJ 2.2C is just not possible. We will therefore provide a free copy of VnmrJ 2.3A to all customers who bought a Dell OptiPlex 755N (that comes with RHEL 5.1 as standard) with a spectrometer and received VnmrJ 2.2C. Customers that sourced their own OptiPlex 755N PC in order to run a DirectDrive spectrometer with VnmrJ 2.2C should contact their local Varian sales representative to arrange for a media kit to be shipped.

The PC hardware is actually not implicated in the issue above. Systems that were delivered with Dell Precision host PC are not affected, as we made a clean break from RHEL 4.0u3 to RHEL 5.1 on the OptiPlex 755N PC and put a support ceiling of RHEL 4.0u3 on the Linux OS version on the Precision class Dell PC models. Therefore, there is no need for any customer running a Dell Precision 370N, 380N, or 390N to upgrade to VnmrJ 2.3A. Also, those who are running RHEL 5.1 for non-imaging data processing on a stand-alone datastation have no need to upgrade, as the issue is uniquely associated with data acquisition on a spectrometer. Users of VnmrJ 2.2D (see Agilent MR News 2008-12-02) are not affected by this at all, as the issue is specific to the acquisition communication software between DirectDrive spectrometers and a PC host computer running RHEL 5.1.

Although we built many new, exciting features into VnmrJ 2.3A for Imaging applications (see Agilent MR News 2008-07-01), we have released VnmrJ 2.3A for non-Imaging with the exact same feature set as VnmrJ 2.2C (see Agilent MR News 2007-08-31), on purpose. Of course, we are working on new software all the time, but as the need to address the quality / reliability issue referred to in this article that could not be fixed with a patch to VnmrJ 2.2C, we took the decision to roll out VnmrJ 2.3A for non-imaging applications now, but without new features.

A number of bugs have been fixed, and they will be outlined in the release notes for patch "2.3AallLNXvnm102" for VnmrJ 2.3A (required on a Dell OptiPlex 755N running RHEL 5.1). That patch is included in the VnmrJ 2.3A media kit and will (after some more testing) be made available to the current (imaging) users of VnmrJ 2.3A as well, via our public "VnmrJ / VNMR Patches" Web page at

http://www.varianinc.com/products/nmr/software/patches/

Watch out for a separate announcement in Agilent MR News on this. For details about the new media kit for VnmrJ 2.3A and the installation of the software see the note below.

[Agilent MR News 2009-01-29]

INSTALLING VnmrJ 2.3A FOR NON-IMAGING APPLICATIONS:

VnmrJ 2.3A for non-imaging applications is NOT a complete re-release of VnmrJ 2.3A (this would effectively have resulted in a VnmrJ 2.3B, requiring us to redo the entire software DVD, go through a full beta test, and release cycle, etc. - which would have substantially delayed the availability of this software). Instead, users will receive a new VnmrJ 2.3A media kit (p/n 01-921585-02) containing the original VnmrJ 2.3A DVD (p/n 01-921582-00) plus an ancillary DVD (p/n 01-921785-01) that contains all the additions that are required to make VnmrJ 2.3A suitable / usable for non-imaging applications as well. There are three main components on that ancillary DVD:

- a new patch, "2.3AallLNXall102", which does the bulk of the upgrade work for non-imaging applications; the DVD also contains a dedicated patch install script which permits installing the patch directly off the DVD, see below;
- a full set of manuals (imaging and liquids NMR), with an associated install script;
- a complete User Library, again with a dedicated install script.

For the installation, please follow the instructions that are included with the media kit. Here's the outline of the process:

=> first, install VnmrJ 2.3A - the same way as you have installed VnmrJ 2.2C (but of course into a new software directory, NOT on top of VnmrJ 2.2C);

=> then, you should install the new patch, "2.3AallLNXall102". You can do this in several ways (always as the VnmrJ administrator, typically vnmr1):
   - the easiest option is to open a terminal window and typing
```
cd /media/cdrecorder
./install_patch
```
   - the same can be achieved by using the file browser to navigate to the directory "/media/cdrecorder", and double-click on the "install_patch" icon. In the pop-up window, select the "Run in Terminal" button.
   - Of course, you can also use the standard patch installation procedure, by opening a terminal window, navigating to "/media/cdrecorder" and typing
```
patchinstall 2.3AallLNXall102.tar.Z
```
   (you do NOT need to copy the patch to the hard drive).

   In all cases you will be asked to enter the root password. The patch MUST be installed PRIOR TO installing the manuals and/or the User Library.

=> the next step would be to install the manuals (these must be installed on the hard disk - they can't be used directly off the DVD). The manuals are installed either by
   - opening a terminal window and typing
```
cd /media/cdrecorder
./install_manuals
```
   - alternatively, open the file browser for the DVD ("/media/cdrecorder") and double-click on the "install_manuals" icon. Again, in the pop-up window select "Run in Terminal".

=> Last, but not least, the DVD contains a recent version (dated 2008-10-24) of the User Library ("/vnmr/userlib"), more complete and bigger (around 430 MBytes) than ever distributed with VnmrJ: this includes files that would take a long time to download over the Internet, such as
   - a new, Linux-compatible version of "bin/decra" (courtesy of Brian Antalek, Kodak), 93 MBytes
   - a package with sample FIDs for Chempack 4.1 ("chempack/CP4_fids", 63 MBytes)
   - a package with Projection Reconstruction sample data for BioPack ("fidlib/PR_data", 158 MBytes)

   besides of course a full BioPack etc.; needless to say that the User Library has since undergone numerous updates, see the note in Agilent MR News 2009-01-22, or the on-line User Library at
      http://www.varianinc.com/products/nmr/apps/corner.html
   Nevertheless, even if you will not install ANY of the contributions as delivered on the DVD, but instead will download and install the latest versions from the above Web site, you should STILL install the User Library from the DVD, as this will provide the framework for installing contributions and updates downloaded over the Internet or by e-mail. Also here, you have two options for the installation off the DVD:
   - open a terminal window and type

```
                cd /media/cdrecorder
                ./install_userlib
```
- alternatively, open the file browser for the DVD ("/media/cdrecorder") and double-click on the "install_userlib" icon. Again, in the pop-up window select "Run in Terminal".

In the next issue of Agilent MR News we plan on posting information on how to activate Chempack 4.1 in VnmrJ 2.3A. For the time being, (future) Chempack users should make sure they install at least the patch and the User Library as indicated above.

[Agilent MR News 2009-01-29]

2009-02-04:

CHEMPACK 4.1 AND VnmrJ 2.3A:

In Agilent MR News 2008-08-13 and articles referred to therein we discussed how to use Chempack 4.1 within VnmrJ 2.2C, and we mentioned that because the initial software philosophy for the 400-MR did NOT feature a Chempack option, in order to activate Chempack 4.1 it is REQUIRED for these systems first to install a special patch, named "2.2CallLNXallcp4". This must be done ONCE per VnmrJ software installation. Thereafter, Chempack 4.1 should be updated by downloading and installing the current version of Chempack 4.1 from the on-line User Library at
```
        http://www.varianinc.com/products/nmr/apps/corner.html
```
The same applies to other DirectDrive systems (Varian NMR System), in case the "chempack" and/or "userlib" (User Library) options were NOT selected when installing VnmrJ 2.2C.

VnmrJ 2.3A was "born" as an imaging-only software, and it's DVD image was "frozen" (i.e., blocked for any further changes) before we changed our policy about the availability of the User Library and Chempack 4.1 to 400-MR users. Such "freezing" of a software is pre-requisite not just in order to fulfill internal rulings (such as ISO-related procedures), but mainly in order to have a stable software version by the time of the official release: in earlier years we used to be less strict and still allowed for "last minute" bug fixes, such as from bugs reported during the beta test, to be incorporated prior to release - and almost inevitably this led to the introduction of new bugs! A side-effect of VnmrJ 2.3A being released (initially) as imaging-only software is that we did not care updating the built-in User Library from VnmrJ 2.2C, as imaging users so far have made very little use of the User Library.

Now we are "re-releasing" VnmrJ 2.3A based on the same, proven / released DVD as for imaging applications, but with an extra DVD holding the necessary additions, see Agilent MR News 2009-01-29. One consequence of this is that for the 400-MR, the VnmrJ 2.3A installation utility will offer neither "chempack" nor "userlib" as an option. The latter can be installed from the extra VnmrJ 2.3A DVD (see again Agilent MR News 2009-01-29), but this still leaves the Chempack 4.1 directory ("/vnmr/chempack") to be installed before the current version of Chempack 4.1 (and future Chempack updates) can be installed.

This should help understanding the presence of the following Chempack 4.1 installation requirements and options under VnmrJ 2.3A (for VnmrJ 2.2C see Agilent MR News 2008-08-13):

- First, install VnmrJ 2.3A from the "main" software DVD; on the 400-MR select whatever options you need, on the Varian NMR System make sure you also select the "Chempack" and "Userlib" options (the latter is optional).
- Next, make sure you also install the user library from the extra DVD for VnmrJ 2.3A, even if on a Varian NMR System you already selected "Userlib" in the VnmrJ 2.3A installation GUI: the User Library on that extra DVD contains more files and numerous updates compared to the version on the main software DVD which dates back more than 18 months, see also Agilent MR News 2009-01-29.
- On the 400-MR you should then download and install the special Chempack patch for VnmrJ 2.3A, "2.3AallLNXallcp4", that can be downloaded from our "VnmrJ and VNMR Patches" Web site at
  ```
        http://www.varianinc.com/products/nmr/software/patches/
  ```
  On Varian NMR Systems, if you have NOT selected "Chempack" when installing VnmrJ 2.3A, you MUST install the same patch.
  That patch "2.3AallLNXallcp4" can be installed irrespective of whether "/vnmr/chempack" is present already or not, and whether "Chempack" and/or "Userlib" have been loaded from the VnmrJ DVD or not. On the other hand, you only need to install that Chempack patch ONCE for a given VnmrJ installation.

- Once that is done, download the current Chempack 4.1 ("chempack/CP4") and install it following the procedure given in the README file or on the download page at the on-line User Library Web site at
       http://www.varianinc.com/products/nmr/apps/corner.html
  For future updates, just repeat this last step (i.e., re-download and re-install "chempack/CP4" the very same way).

One last point: while this may not be strictly required, it is probably still a good idea ALWAYS to install a Chempack 4.1 update (see the last point above) AFTER having installed a VnmrJ software patch.

[Agilent MR News 2009-02-04]

FILE / PROTOCOL / PRINTER NAMING IN VnmrJ:

Linux (as well as UNIX) is fairly liberal in the allowable file names, at least in theory: not only can file names be VERY long (several lines on a terminal, if you really wanted), but you may also use almost arbitrary characters - however, in reality (i.e., for reasons of practicability) there are numerous restrictions: in Agilent MR News 2008-05-09 we posted an extensive discussion on Linux and VnmrJ file name restrictions. We re-state this here, for the reason that VnmrJ is somewhat more restrictive than Linux, UNIX or other OS types, in that it uses it's own, specialized "command shell", MAGICAL. In the MAGICAL language, the "lexical structure" does NOT allow for command names to start with a numeric character (the first character MUST be alphabetic or "_"). This is unlikely to change, as that could defeat the interpretation of math expressions and the handling of the popular PPM ("p"), "k" or other (also user-defined) units in numeric expressions and entries.

A tricky point is that in using VnmrJ, the user is often asked to enter pieces of information where it is not clear that the very same information is at some point going to be used as / converted to a file name. A prime example for this (even back to the early days of VNMR) is in automation, where the user is asked to enter a comment for each sample, and thereafter the software may extract a string / token from (the start of) that comment to generate a file name that is likely to have more of a meaning to the user than the mere sample tube (tray location) number. The automation software is well protected against the extraction of "illegal" characters from such comment, and we haven't had any issues with this in several years.

However, lately, users have found new "loopholes" where we forgot to add the necessary protection. One example is in the VnmrJ study queue where users can define new protocols. These protocols are primarily saved as a parameter set (hence general file naming restrictions apply) - but (at least within Chempack 4.1) at the same time the software also creates a MACRO with the name of the protocol. While a protocol name "19F" (not an exotic choice in a chemistry environment!) is OK from the point-of-view of generic Linux file naming, such a name CANNOT be used as protocol name because it would result in a macro named "19F" that cannot be called / executed in MAGICAL! Meanwhile, Krish Krishnamurthy has updated Chempack, and the current version of Chempack 4.1 is protected against the creation of illegal protocol names. That latest version can be downloaded from the on-line User Library at
       http://www.varianinc.com/products/nmr/apps/corner.html
On the other hand it is unclear whether the same or related issues exist in VnmrJ, outside of Chempack 4.1. We are posting a bug report "vnmrj.j2206", in order to have this covered in the VnmrJ bug lists, in case a user runs into such issues and is looking for help in the VnmrJ bug reports.

There is one related issue in connection with VnmrJ plotter definitions: when switching between VnmrJ plotter definitions (either by setting the "plotter" parameter directly, by calling "nextplotter", or through any GUI plotter selection utility), the macro "_plotter" is executed, which runs the macro "loadcolors", which in turn activates a plotter-specific color profile (if defined) by executing a macro with the NAME OF THE PLOTTER. Therefore, the VnmrJ macro naming restrictions also apply to plotter names - in other words: plotter names must NOT start with a numeric character.

[Agilent MR News 2009-02-04]

BIOPACK AND THE "appdir" APPROACH IN VnmrJ 2.2C & UP:

In Agilent MR News 2008-06-18 we posted a comparison between the "appdir" (Application Directories) approach in VnmrJ 2.2C, VnmrJ 2.2D and VnmrJ 2.3A, and the predecessor concept, consisting of a set of VnmrJ "path" variables (such as "maclibpath", "manualpath", "menulibpath", etc.) in older VnmrJ and VNMR versions.

When VnmrJ 2.2C became available, also Chempack 4.1 and BioPack were reworked to use the "appdir" approach. What is the effect of this change for the BioPack user?

For one, BioPack never made real use of the "path" variables. Instead, in software versions up to VnmrJ 2.1B, the user was given two basic installation options:

- individual users could (independently) install BioPack in their local VnmrJ directory structure ("~/vnmrsys" and its subdirectories). This implied that if, e.g., on a system four out of seven users wanted to use BioPack (while the others preferred to work with standard, unmodified VnmrJ software), the BioPack users would EACH have to install the ENTIRE BioPack software in their account. This may no longer be a problem of disk space consumption (even though a single BioPack installation now is close to 90 MBytes), but at least it involved considerable amounts of work. For the BioPack users, there was no "easy way back", or at least no easy way to switch between "vanilla" VnmrJ and the "BioPack flavor" (but of course one could always create new accounts - even temporary ones - for that purpose).
- on systems where ALL users were interested in (or willing or agreeable to) working with BioPack, that software could also be installed in "/vnmr" and its subdirectories. This had the advantage of covering all users with a single BioPack installation (saving time, administrative overhead and disk space). The one disadvantage with this solution is that the VnmrJ install directory "/vnmr" got "filled in" with lots of extra files. "File overlap" (i.e., BioPack overwriting standard VnmrJ files) should not be a major issue, as George Gray worked hard on keeping the bulk of BioPack SEPARATE (i.e., with BioPack-specific names) from the standard software - however, SOME overlap still persists (i.e., a BioPack installation would modify certain standard VnmrJ files); what occasionally was a problem was that
  - in case something went wrong, it may occasionally have been necessary to reinstall not only BioPack, but BOTH VnmrJ AND BioPack
  - even though in the BioPack installation all modified files were backed out, there was no easy way to "uninstall" BioPack: there is no tool to take apart (or restore) standard VnmrJ and BioPack files and to revert to standard VnmrJ), i.e., in the case of problems, or if for example Varian service or a specific user wanted to work (e.g., do some system testing) with "vanilla" VnmrJ software, it may have been necessary to install "vanilla" VnmrJ in a separate, temporary directory, and switching between the two VnmrJ installations was somewhat unhandy (see Agilent MR News 2004-03-07 for information on this topic).

The same essentially holds true for Chempack in the same versions of VnmrJ. Fortunately, with the "appdir" approach (VnmrJ 2.2C and up), most or all of the above, "unwieldy" aspects of BioPack (or Chempack 4.1) can be avoided: in most cases, vnmr1 will install BioPack into "/vnmr" (Chempack 4.1 even comes preinstalled this way, provided you select the "Chempack" installation option, see above). In these newer VnmrJ versions, the installation of BioPack will automatically be directed into a SUBdirectory of "/vnmr", "/vnmr/biopack" (or "/vnmr/chempack" in the case of Chempack 4.1), and for vnmr1, the user will be instructed (via banner message) to add and activate "/vnmr/biopack" in the "Applications..." path / utility. The BIG advantage with this solution is that it is simply a matter of deactivating (not even removing) "/vnmr/biopack" in the "Applications..." utility - and you are back to "vanilla" VnmrJ!

There is one minor disadvantage with this solution (a small price to pay for a lot of added flexibility!): installing BioPack into "/vnmr/biopack" DOESN'T automatically activate the software for all users - instead, EVERY user who wants to use BioPack MUST open "Applications..." and insert (and activate) "/vnmr/biopack" in the applications path (but then again, every user can very easily, individually and independently revert to "vanilla" VnmrJ software). We just realized that this last aspect are not covered in the current BioPack README file and documentation - the librarian will add an amendment to the BioPack README file and the download page at

        http://www.varianinc.com/products/nmr/apps/corner.html

in order to have this covered for new BioPack users.

With these options, there is very little - if any - reason NOT to install BioPack or Chempack 4.1 in "/vnmr": having these packages installed in "/vnmr" (and by vnmr1) should be the standard / default choice in VnmrJ 2.2C and later versions.

[Agilent MR News 2009-02-04]

2009-02-12:

BIOPACK AND THE "appdir" APPROACH IN VnmrJ 2.2C & UP:

In the last issue (Agilent MR News 2009-02-04) we posted an article on how BioPack can be installed and used in VnmrJ 2.2C and later releases: the standard method in these VnmrJ versions should be to install BioPack globally, in "/vnmr" (rather than in individual user directories), and then to enter and activate "/vnmr/biopack" in the "Edit" -> "Applications..." GUI. The VnmrJ administrator (vnmr1) gets the necessary instructions upon installation, via "banner" messages. In the article last week we stated that other users then INDIVIDUALLY need to open "Edit" -> "Applications..." and do the same for their account. The same applies to Chempack 4.1 in these VnmrJ versions, except that Chempack in most cases comes preinstalled in "/vnmr/chempack", see Agilent MR News 2009-02-04).

As Dan Iverson (Varian) pointed out, the above solution can be tedious if there are many users on a given system - but there is an alternative option: the VnmrJ administrator (vnmr1, or anyone with write permission into "/vnmr" its subdirectories) can select "Save as global applications directories" and save the current "appdirs" path under a name (to be specified), see the "appdirs" entry in the Command and Parameter Reference manual. That name can then be used in the "vnmrj adm" utility to select / set up the default applications path for all "Walkup" and / or all "Experimental" users (BioPack assumes the "Experimental" VnmrJ user interface). This saves the other users from having to call the "Applications..." GUI - unless they already have set up their own, customized "appdirs" path. Once the defaults are set, users can still override these path settings via "Edit" -> "Applications...".

Similarly, in a Walkup environment where one account has multiple operators, the owner of that account can set the applications path for all operators, and the individual operators do not need to open the "Applications..." pop-up to use these settings - but they CAN override these "appdirs" settings if they want to.

[Agilent MR News 2009-02-12]

PARAMETER CHANGES IN "usergo" MAY CAUSE FID SCAN TO FAIL:

Even on systems with a powerful host computer block size processing during acquisition (e.g., "wbs='wft'", especially with very small block sizes, such as "bs=1") may be fairly disruptive if someone wants to use the system for processing or for setting up new experiments. In order to avoid problems, some users therefore resorted to suppressing "wbs" activities entirely, either by having a macro "usergo" with a command "wbs=''", or maybe even by adding that command to the "go" macro, somewhere above the section where it executes the "Acq" command (though modifying "go" is a bad idea in general, as such changes will be lost with the next software update, maybe even through a VnmrJ patch).

Such a "usergo" solution looks very simple and efficient - HOWEVER, it may create problems by itself, e.g.: "fid_scan" (called through "FID scan" button, used for FID shimming, etc.) REQUIRES specific actions to be performed through "wbs" processing calls (and with "bs=1") - therefore such a "usergo" macro will be detrimental to the use of "FID scan" / FID shimming! Fortunately, by looking at the "fid_scan" macro it is fairly easy to find a solution that avoids this particular clash. The critical part in "fid_scan" is

```
        bs=1
        wbs='fid_display'
```

Based on this, we can write a "usergo" macro that reads

```
        if (wbs <> 'fid_display') then
          wbs=''
        endif
```

i.e., we avoid unconditional resetting of the "wbs" parameter. The call to "fid_display" is specific to "fid_scan". Thanks to Dan Iverson (Varian) for providing this information!

On a related note: the "FID scan" button may fail for reasons other than the above - see the bug reports "fidscan.j2201", "fidscan.j2202", "fidscan.j2203".

[Agilent MR News 2009-02-12]

2009-02-18:

BIOPACK AND THE "appdir" APPROACH IN VnmrJ 2.2C & UP:

In Agilent MR News 2009-02-04 and Agilent MR News 2009-02-12 we posted notes on how BioPack (and Chempack 4.1) are used and fit into the "appdirs" approach within VnmrJ 2.2C and subsequent releases. With this new "applications path" method, BioPack is no longer installed directly in "/vnmr" or in the user's local VnmrJ working directories ("~/vnmrsys" and subdirectories), but rather in new, DEDICATED subdirectories, "/vnmr/biopack" for a global installation inside "/vnmr" (now the recommended standard method) or locally, in "~/vnmrsys/biopack". BioPack can then be activated and deactivated using the "Edit" -> "Applications..." utility.

The changes for installation under VnmrJ 2.2C (and subsequent releases) were implemented about 18 months ago (with the usual, typically minor fixes following in the months thereafter). Triggered by repeated input by Ryan McKay (NANUC) - many thanks, Ryan, for insisting! - the librarian has now revisited the install script, and to his amazement he found that the installation did NOT handle pulse sequences the way they should be: as discussed already in Agilent MR News 2008-06-18, the "appdir" (applications directories / path) approach does NOT cover "bin", "psglib" and "seqlib" (and hence "seqgen"), "psg" (and "psggen" and "fixpsg") and a few other, Linux based software features; the applications path is restricted to "purely internal" aspects of VnmrJ (macros, parameters, templates, etc.).

The installation did NOT completely ignore these rules: the compiled pulse sequences were installed in the general "seqlib" directories ("/vnmr/seqlib" or "~/vnmrsys/seqlib") and did provide the expected global or local access, but the pulse sequence source codes and the extra "psg" files in BioPack were left "hidden" inside the "biopack/psglib" and "biopack/psg" subdirectories, which made pulse sequence alterations and (re-)compilation fairly unhandy for the user: the relevant files first had to be dug out of the "appdir" path and placed in the relevant general directories (in "~/vnmrsys") first.

This has been CORRECTED in the latest version of BioPack and the update package, "psglib/BPupdate" that can be downloaded from the on-line User Library at

        http://www.varianinc.com/products/nmr/apps/corner.html

With these latest updates, pulse sequences (along with any "psg" header files, as and where required) will be left in the standard directories in "/vnmr" (or the user's "~/vnmrsys" in the case of a local installation). This means that for a local installation, the user can simply (change and) recompile any pulse sequence without copying any file(s) - with the added benefit that there will be an unchanged backup version in the user's "~/vnmrsys/biopack/psglib".

In the case of a "global" installation, the backup source codes will be stored in "/vnmr/biopack/psglib", but the active files are all in "/vnmr/psg" ("*.h" files) and "/vnmr/psglib", respectively. This may look like "breaking" the "appdirs" concept - but it merely means that we use that concept within its range of validity. Apart from some amount of clutter in "psglib" through the extra BioPack pulse sequence source codes there is no real disadvantage to this setup: there is no "naming overlap" between BioPack sequences and the standard set of sequences in any current VnmrJ release - and even if there WAS overlap, the original sequences would be backed out by the install software. The big advantage with this setup is that ANY user can locally (re-)compile a BioPack pulse sequence simply by typing "seqgen sequence_name" in a shell terminal: if "seqgen" does not find the specified sequence in the user's "~/vnmrsys/psglib", it will automatically start off by copying the file from "/vnmr/psglib" into the local directory. Also "psg" files are available to all users in "/vnmr/psg", where "seqgen" will find them (even non-BioPack users can recompile BioPack sequences locally - though of course their VnmrJ does not have access to the relevant macros, parameter sets and layout definitions.

These fixes / enhancements have been implemented both in the full BioPack, as well as in the update package, "psglib/BPupdate", though of course, if you now "just" install "BPupdate", only the sequences in the update package will be installed according to the above, new scheme - if you want to take full advantage of the modified installation scheme, you need to reinstall the full BioPack - we are currently NOT enforcing this by switching to a new base version, but we may do so in the not too distant future.

[Agilent MR News 2009-02-18]

2009-02-26:

CUSTOMIZING EXPERIMENTAL CONDITIONS IN CHEMPACK 4.1:

In the "old days" of VNMR or early VnmrJ releases, users would typically set their default experimental conditions by customizing the relevant "stdpar" entry, e.g.:

```
        rtp('/vnmr/stdpar/C13')
        nt=1024 d1=0 np=...
        mkdir(userdir+'/stdpar')
        svp(userdir+'/stdpar/C13')
```
If you are using a recent VnmrJ release and Chempack 4.1 (especially with protocols and the StudyQ in a walkup environment), you will notice that this does NOT have the desired effect, if any at all. The main reason is that with protocols, experiments are NOT set up using the traditional "setup" command (macro), but rather a recently introduced mechanism that SELECTIVELY retrieves SPECIFIC parameter values for the chosen experiment while preserving the (parameter) information that is sample, but NOT experiment-specific. The core of this concept is the macro "rtx" and a new protection bit (bit 14, value 16384, "P_LOCK"), see the manual entries for "rtx" and "setprotect".

In this note, we won't bother you with the details of this concept - the key point we want to address is: what is the best way to customize parameters within the current Chempack 4.1 environment? Here, a macro "cpsetup" - rather than "setup" - acts at the core of protocol setup and retrieves parameters (protocols) from "parlib" (NOT "stdpar"), using "rtx" at some point. Do NOT try customizing that macro - such changes would either be overwritten with the next Chempack update, or (if stored locally) it might disable / defeat future enhancements to that macro. Krish names up to four possible methods for doing customization in protocols, such as changing the "P_LOCK" protection bit on selected, customized parameters in the relevant "parlib" entries, or using the "wrtp" parameter to have customization occur when calling "rtp", or adding a customization segment to macros such as "PROTON_setup" (more generally, the relevant "{PROTOCOL_NAME}_setup" macro - HOWEVER, the method that is preferred above all those is to create a macro "userPROTON" ("user{PROTOCOL_NAME}", more generally) with the relevant customization settings ("nt=1024 d1=0 np=..." in the above example). That macro can be placed in "/vnmr/chempack/maclib" if it applies to all Chempack users, or in "~/vnmrsys/maclib" if applicable to specific accounts only (so that users can have their own, personalized default settings). If you want to see where such user macros are involved, look for "dousermacro" calls in the Chempack 4.1 macros. Thanks to Krish Krishnamurthy (principal Chempack author, working at Eli Lilly & Company), for providing this information! More information is of course also available in the documentation package for Chempack 4.1 (PDF format). To update your Chempack installation (in VnmrJ 1.1D or later versions) to the current level please visit the on-line User Library at

        http://www.varianinc.com/products/nmr/apps/corner.html

to download the current version.

[Agilent MR News 2009-02-26]

2009-03-12:

EXPANDING THE USE OF THE "appdirs" APPROACH:

In Agilent MR News 2008-06-18 we discussed the advantages of the "appdirs" approach (as accessible via "Edit" -> Applications..." in VnmrJ 2.2C and newer releases) over the earlier "maclibpath", "menulibpath", etc. method. The key points about the new approach are

- it does not cover just specific libraries / directories, but an entire subset of the VnmrJ subdirectories, such as "maclib", "templates", etc.;
- the order of priority is freely selectable (also individually, i.e., on a per-user or a per-operator basis);
- path elements such as Chempack (now usually installed in "/vnmr/chempack") or BioPack (now usually installed in "/vnmr/biopack") can be freely enabled and disabled (and switched between), again on a per-user basis (see also Agilent MR News 2009-02-18 and articles referred to therein);
- the installation of software such as BioPack, Chempack, etc. can be done once for all (interested) users - and yet it is not necessary to alter the contents of the standard software directories inside "/vnmr". At the same time, such a global installation does NOT imply that every user on a system can ONLY use VnmrJ with Chempack or BioPack enabled.

Krish Krishnamurthy (Eli Lilly, principal author of Chempack) has taken this concept one step further by using it for purposes that most users may not have considered so far:

Krish is often involved in writing macros / utilities for small or local projects (maybe involving macros, parameter templates, protocols / parameter sets, etc.). The objective of such small projects may be a specific set of experiments / experimental setups, or specialized processing that pertains to a given project only, and therefore such utilities are then needed ONLY for the duration of that project. Macros for such special / small projects are often different from / simpler than generic ones, as they don't need to be generally applicable / usable, hence they may work without arguments and may use hard-coded paths / names, etc., and the names for such macros probably relate to the project and don't need to be short and/or meaningful mnemonics.

In the past, such projects were typically realized in a user's local directories, with the consequence that (unless the administrator created a user SPECIFICALLY for such a project) the files relating to that project mingled with all other files in that user's account. Consequently it may often have become tricky to extract / archive all files relating to a given project (as likely required by GLP rulings), and removing or deactivating such a project involved searching and removing all related files, or logging into a different / "neutral" account.

With VnmrJ 2.2C or later versions, such projects can easily be developed, maintained and used from a dedicated (sub)directory, and such a directory can be placed in a user's home directory ("~/project1"), in a directory that also holds the data for that project (e.g., "/data/vnmr1/project1"), within a user's "vnmrsys" ("~/vnmrsys/project1"), or even inside the local "maclib" ("~/vnmrsys/maclib/project1"), though the last option is NOT really a good idea, see below. In all these cases you may set up the directory structure by first going to the parent directory, e.g.:

```
        cd /data/vnmr1
        mkdir -p project1/data
        cd project1
        mkdir maclib
        mkdir manual
        mkdir parlib
        mkdir -p templates/layout
```

Then you add that directory "project1" (the full, absolute path) to the "appdirs" path, using "Edit" -> "Applications...". Very likely (and different from Chempack or BioPack) you want these utilities to be listed in the TOP position, overriding all other files, even the directories in the user's local "vnmrsys" ("USERDIR"). Now, you can create macros in the new "project1/maclib" directory, save parameters in "project1/parlib", etc., and all project-related files are nicely collected in one single, clean directory structure from where they can easily be archived, even together with the project-related data. When you are done with the project, just disable the "appdirs" entry via "Edit" -> "Applications..."; later re-enabling the project is just as easy.

One major advantage with this approach: many of us (needless to say: the editor included!) carry along this "bag" of a cluttered "maclib" with numerous legacy macros that were once developed for a specific purpose that we vaguely remember - we never dared throwing these away because we may not be sure whether one day we may still need them again, and/or whether there are other macros still calling these tools that may be as old as 10 - 20+ years. Now, you have a way to keep things nicely grouped in smaller "containers"! Finally, we should also mention the following caveats / remarks:

- keep in mind the an "appdirs" entry is always the PARENT directory of common VnmrJ directory entries such as "maclib", "parlib", etc.; if you selected to create the project directory inside your "maclib", you can NOT have macros directly in "~/vnmrsys/maclib/project1", but they must be located in "~/vnmrsys/maclib/project1/maclib"; you COULD theoretically have a "~/vnmrsys/maclib/maclib" and then specify "~/vnmrsys/maclib" as "appdirs" location - but that would be rather confusing (also because you might then end up with a "~/vnmrsys/maclib/parlib", etc.).
- Keep in mind that the "appdirs" approach is limited to software features that are INTERNAL to VnmrJ ONLY, i.e., it does NOT cover UNIX executables (in "bin") and pulse sequence-related files ("psglib", "psg", etc.); one CAN have a "seqlib" inside an application directory, and compiled pulse sequences WILL apparently be found in there - HOWEVER, the editor thinks this is NOT a good idea right now, as
  - you would need to copy the compiled sequence again manually whenever recompiling the sequence
  - after a software upgrade you may have unusable / non-functional legacy executables in such a "seqlib", possibly causing errors that are hard to trace.

  For information on the limitations see also Agilent MR News 2008-06-18 and Agilent MR News 2009-02-18.

Thanks, Krish, for sharing these ideas with us!

[Agilent MR News 2009-03-12]

2009-04-01:

INSTALLING BIOPACK IN "/vnmr":

Up to VnmrJ 2.1B, BioPack offered two basic installation options: either an installation in "/vnmr" (selected automatically if the installer is vnmr1), or an installation in one or several user accounts. In the first case, BioPack was active - enforced - for ALL users of a given system. On a system with non-BioPack users as well as several BioPack users, BioPack was installed in each and every BioPack user's local "~/vnmrsys" directory. This not only involved lots of duplication, it also consumed a lot of disk space for the duplicated files, and the multiple installations took a lot of time. The one advantage of the "multi-install option" was that the non-BioPack accounts such as vnmr1 (and hence Varian service personnel) kept access to standard VnmrJ software, which often is helpful in tracing hard- or software problems when a service engineer or support person is not familiar with bio-NMR. In the case of an installation in "/vnmr" - on systems used for bio-NMR only this was certainly the most efficient option - there was no easy way back to "vanilla" VnmrJ software, unless one wanted to do a temporary (re-)installation of VnmrJ in a new directory (see Agilent MR News 2007-11-28 and articles referred to in that note for more information on how to switch between different VnmrJ installation directories).

As of VnmrJ 2.2C, things have changed dramatically, as discussed in Varian MR News 2009-02-18 and articles referred to therein. In VnmrJ 2.2C and later releases, BioPack makes use of the "appdir" approach, and installation is done in a SUBdirectory of "/vnmr" ("/vnmr/biopack") or (for a local installation) in "~/vnmrsys/biopack" - in other words: the standard software directories are left untouched, with the exception of "psg", "psglib" and "seqlib" (see again Agilent MR News 2009-02-18). This means that you cannot just activate BioPack (or Chempack) using "Edit" -> "Applications..." from the menu, but they can also be DEactivated the very same way, and if BioPack is installed in "/vnmr", ANY VnmrJ user can freely activate or deactivate BioPack for that account - which is why the installation in "/vnmr" (by vnmr1) is now the recommended standard installation option.

Now, some sites have many BioPack users, but - for good & specific reasons - they want to keep the vnmr1 account in "non-BioPack mode". But yet, when you are installing BioPack as vnmr1, you are guided through a number of steps that appear to enforce the activation of BioPack for vnmr1; this may lead users / administrators still to perform multiple local BioPack installs. However, that conclusion misses one point: when installing BioPack as vnmr1, you should perform the installation as instructed - but thereafter, you may as well DEactivate BioPack for vnmr1, again using "Edit" -> "Applications...", while letting all the BioPack users activate the software for themselves. With this, the main change for vnmr1 will be some extra clutter in the "psg", "psglib", and "seqlib" subdirectories in "/vnmr" (see Agilent MR News 2009-02-18), but the extra files should all be BioPack-specific, i.e., they will not alter standard VnmrJ pulse sequences, and related files.

[Agilent MR News 2009-04-01]

2009-04-09:

HOMODECOUPLING ON SYSTEMS WITH DIRECTDRIVE ARCHITECTURE:

We just realized that parts of the information on doing homodecoupling on systems with DirectDrive architecture (Varian NMR System) is missing from our documentation. This probably went unnoticed because so few people are doing homodecoupling experiments these days! In this note we just want to give a few pointers / key information that should permit getting started on this technique.

Traditionally, i.e., on older generation instruments, homodecoupling was performed using the decoupler channel, whereby the frequency was set using the command (macro) "sd" (or "sda" for arrayed homodecoupling experiments). The actual irradiation is then controlled via the "dm" parameter, and the "homo" parameter controlled the receiver and decoupler gatings. In the very old days (some 25 years ago) this method was also used for solvent suppression via presaturation, i.e., for irradiating a solvent line during "d1" (and possibly delays such as "d2" and "mix" in 2D experiments). However, people realized that presaturation experiments were far better when using c.w. irradiation using the observe channel, which has since become one of the standard methods, even though this required adding new features to the relevant pulse sequences.

For actual homodecoupling experiments signals need to be irradiated during the acquisition time, hence (at least on a typical Varian spectrometer setup) leaving the decoupler on continuously is not an option, and receiver gating around the decoupler pulses is necessary to keep decoupling artifacts at an acceptable level (hence the "homogating", controlled by the "homo" parameter). With this option, the decoupler and the associated receiver gating were done asynchronously, controlled by the decoupler board.

Improved results could be obtained on instruments such as the UNITY INOVA, by using specially crafted pulse sequences featuring explicit acquisition and explicit, synchronous gating during the acquisition time; this gives full control over the decoupler and receiver gating duty cycles, and thus permits selecting the appropriate "compromise point" between decoupling efficiency and signal-to-noise loss due to the receiver gating. However, the need for special pulse sequences probably limited the popularity of this technique.

On the Varian NMR System (DirectDrive architecture), these traditional methods (using the decoupler channel) for homodecoupling has not been implemented. Instead, we are now using a new approach:

- homodecoupling is done using the observe rather than the decoupler r.f. channel - "dn", "dof", "dpwr", and "dpwrf" are NOT used;
- homodecoupling is NOT controlled by the "dm" parameter; instead, simply set "homo='y'", which will automatically create all parameters that are or may be used for controlling the homonuclear decoupling;
- as the decoupler channel is not involved, the command "sd" (set decoupler frequency, "dof") CAN'T be used for homodecoupling. Instead, a new command (macro) "shd" must be used ("shd" unfortunately is NOT documented in the Command and Parameter Reference manual);
- homodecoupling is INDEPENDENT of the pulse sequence (provided the sequence does NOT use explicit acquisition, and that the pulse sequence returns the transmitter offset to "tof" before acquisition starts) - and yet it gives the user full control over all relevant experimental parameters, AND it also features pattern-based operation for multi-site irradiation / homodecoupling;
- homodecoupling uses its own set of specific parameters, namely
  - "hdof" (the frequency offset for homodecoupling),
  - "hdpwr" and "hdpwrf" (for power control),

  plus a set of optional parameters such as
  - "dutyc" (RF duty cycle fraction for homodecoupling; default value: 0.1, accepted values are between 0.0 and 0.4)
  - "homorof1", "homorof2", for "homorof3" controlling the gating around the decoupling pulses

  (see the Command and Parameter Reference manual for more detail)
- pattern-based homodecoupling is implemented very similar to traditional heteronuclear decoupling, using
  - "hdseq" (decoupler waveform name)
  - "hdres" (decoupler waveform tip angle resolution)
  - "hdmf" (decoupler waveform modulation frequency, analogous to "dmf")

  If you don't use pattern-based homodecoupling, simply set "hdseq=''". Note that if "hdseq" is set to the name of a pattern, then "hdof" is NOT used - in this case, the offset(s) is (are) defined within the specified pattern.

Key points in getting started are
- start off setting "homo='y'" - this will create all necessary parameters, if not defined yet;
- to disable homodecoupling set "homo='n'" - this will NOT alter or destroy the other parameters as created by "homo='y'" and/or set by the user;
- be careful with the power level: "hdpwr" values of 10 to 20 should be sufficient for this type of experiment (there is no directional coupler involved here!). Note that this assumes a default duty cycle fraction of 0.1; this is also effective if the parameter "dutyc" does not exist.

Thanks to Boban John (R&D) and Dimitris Argyropoulos (Varian U.K.) for information for this article!

[Agilent MR News 2009-04-09]

2009-04-16:

USING "makefid" ON DIRECTDRIVE SYSTEMS:

When Nagarajan Murali (Rutgers University) reported that "makefid" is "not working for floating point FIDs" (see the new bug report "makefid.j2101"), we realized that a recent amendment to the "makefid" utility by mistake didn't make it into the VnmrJ documentation (neither into the "Command and Parameter Reference" manual, nor into "/vnmr/manual/makefid", i.e., "man('makefid')").

The basic command syntax for "makefid" has NOT changed - however, with the advent of the DirectDrive architecture (Varian NMR/MRI system, 400-MR), DSP now is a standard "component" that "comes with" the DirectDigital receiver and can't be turned off - and it yields FIDs in floating point format.
  The original "makefid" definition allowed for the creation of FIDs in 16-bit ("dp='n'") and 32-bit ("dp='y'") integer format (see "man('makefid')") ONLY - it didn't even cope with the task of creating FID files that were acquired with "dsp='i'" (in-line DSP) and therefore were in floating point format even on older system architectures. At least with VnmrJ 2.1B and up, this has now been corrected. As mentioned, the basic command syntax remains the same:

```
        makefid(input_text_file<, element number<, format>>)
```

In "/vnmr/manual/makefid", the paragraph describing arguments #2 and #3 should read as follows:

```
    The element number is any integer larger than 0. If this element already
    exists in your FID file, the program will overwrite the old data. If not
    entered, the default is the first element or FID. The format argument lets
    you select between single precision integer, double precision integer, or
    floating point data in the resulting FID file. Use one of the following
    character strings to specify the format:
        'dp=n'                  single precision (16-bit) data
        'dp=y'                  double precision (32-bit) data
        '16-bit'                single precision (16-bit) data
        '32-bit'                double precision (32-bit) data
        'float'                 floating point   (32-bit) data
```

Note that "writefid" does not have an issue here - the text output file simply reflects the contents of the specified (or the default) FID trace.
  Also, watch out for the information near the bottom of the "makefid" manual description: "makefid" may alter / destroy saved data, as also discussed in an article on the "rt" command in Agilent MR News 1997-11-06 and earlier articles referred to therein.

[Agilent MR News 2009-04-16]

2009-05-20:

HOMODECOUPLING ON SYSTEMS WITH DIRECTDRIVE ARCHITECTURE - FOLLOW-UP:

  In Agilent MR News 2009-04-09 we posted an article with instructions on how to do basic homonuclear decoupling experiments on systems with DirectDrive architecture. The reason for that article was that the software feature had been available for a while already, but parts of the corresponding description are still missing in our manuals. The fact that the lack of documentation on how to do such experiments on our current systems went unnoticed for quite some time is an indicator for the fact that homodecoupling is not performed very frequently these days.
  While simple homodecoupling experiments work OK as described, the editor noted that a macro "shda" for setting up arrayed homodecoupling experiments has not been defined so far - this was logged in bug report / suggestion "shda.j2301" earlier this month. It turned out that the missing "shda" macro is not the only unresolved issue with arrayed homodecoupling: that software feature obviously has not been evaluated and explored yet on our current systems. We now found a software problem with arrayed homodecoupling that cannot be circumvented by a simple workaround. The issue has been fixed now, the correction will become available in the upcoming, major software release, VnmrJ 3.0; with the current software versions for systems with DirectDrive architecture (Varian NMR System), arrayed homodecoupling experiments should be regarded an unimplemented software feature (hence there is no need for users to submit bug reports - we are aware of the issues).
  Referring to our article in Agilent MR News 2009-04-09: the key to using homodecoupling is in the "homo" parameter. Setting "homo='y'" calls a macro "_homo" that creates / initializes the relevant parameters "hdof" (frequency), "hdpwr" and "hdpwrf" (integers), "hdseq" (string), "hdmf", "hdres" and "dutyc" (reals). For this to work, the parameter "homo" MUST have its protection bit #3 (value 8) set - this protection bit causes the execution of an "underscore" macro "_{parameter_name}" to be executed whenever a parameter value is entered directly (e.g., with "homo='y'" or "homo='n'"). The "fixpar" macro (called automatically when retrieving parameter and data sets) ensures that the "homo" parameter exists and calls

```
        setprotect('homo','on',8)
```

to ensure that the protection bit is set as described. With older versions of Chempack 4.1 there could be situations where the above protection bit was not set with the "homo" parameter - this has been corrected in Chempack 4.1 dated 2009-04-13 and later updates.

[Agilent MR News 2009-05-20]

2009-10-01:

LOCKING OUT USERS FROM APPLICATIONS OTHER THAN VnmrJ?

A sysadmin in Germany was upset with users playing games, browsing the Web, etc. rather than using the NMR PC for data acquisition or data processing. He is using a solution similar to the recipe below which was put together by the editor, based on some quick trials on a PC running RHEL 5.4; we do NOT mean to propagate this as a general recipe (nor have we done thorough testing with this, let alone on a spectrometer host) - we merely forward the information because some of you may find this useful and may want to play with such a setup. The idea is as follows:

- on an open access system you only offer ONE SINGLE Linux account for open access users (or for a particular open access session);
- the open access users do NOT know the password for this account, nor to any other account on the system;
- the sysadmin logs into that account at the beginning of an open access period;
- if one of the open access users manages to log out (on purpose of by mistake), or (hopefully not!) makes an attempt to escape from the special account by rebooting the PC, that user will need to see the administrator in order to be logged in again.

So, you can see some potential pitfalls for the sysadmin with this solution! Here's what the admin does to start up the open access account:

- log out as the current user;
- on the login screen, from the "Session" menu select "Failsafe Session";
- log in as open access user; this will not NOT launch a Gnome or KDE session; instead, you will have a terminal window without decoration / handles, etc. (no X Window manager is running);
- in that terminal, type
  ```
  mwm&
  ```
  which starts the basic Motif window manager - you now have a window with handles, and users can theoretically use a very basic / simple background menu and/or type commands in that window (if they see it at all);
- the shell running in that single window has not gone through the standard user login scripts, therefore, the Linux command path is not suited for running VnmrJ, calling "vnmrj" will fail. Therefore, call
  ```
  source ~/.login
  ```
  When asked about the display server name, simply enter [Return] - the X display server is now defined as ":0.0".
- now, type "vnmrj" - and VnmrJ is started, hiding the terminal window. There is no top menu, no bottom utility bar - JUST VnmrJ!
- theoretically, a user may type any command in the terminal window, any game or other program can be launched - but for one, the terminal window is hidden as long as VnmrJ is running and not reduced to an icon - and also, most users will be clueless about the Linux names for typical Gnome or KDE utilities or games.

To log out, you need to

- exit VnmrJ
- terminate the shell in the terminal window, by entering "exit" or [Ctrl-d];
- this will also kill the window manager and log out the user, the standard login screen will be presented.

[Agilent MR News 2009-10-01]

2009-10-08:

AN UNEXPECTED NAMING RESTRICTION WITH VnmrJ ACCOUNTS:

Dan Iverson (Varian, software R&D) just discovered an interesting issue with the "expactive" command: one possible syntax for this command is

```
expactive('current'):$actexp,$who
```

(see the Command and Parameter Reference Manual for details), whereby the first return argument is the number of the active experiment (0 is returned if no experiment is active in acquisition), and the second return argument is the name of the user who owns that experiment. If the system is running in automation mode, the second return argument is set to "'auto'". Unfortunately, it went unnoticed that this creates a conflict in cases where the name of the user holding the active experiment happens to be "auto": some macros will then assume that the system is in automation mode!

The "proper" fix for this situation would be to change the definition of the "expactive" utility - however, this implies a change in the argument syntax, and as the "expactive" command has been around for several VnmrJ releases, this is hardly doable:

- it would now require changing a number of macros that call "expactive";
- these modified macros would become backwards incompatible (i.e., they would no longer work with older VnmrJ versions unless we issued a patch for these versions;
- an unknown number of user-generated macros would break, i.e., would need to be changed as well (and would then have the same compatibility issue with older or unpatched VnmrJ installations).

Essentially, it is too late to correct that command syntax. Instead, we will modify "vnmrj adm" such that it prevents the creation of a VnmrJ user named "auto".

This does of course not correct any installation that ALREADY features a user named "auto": if you indeed have such an account, we STRONGLY recommend renaming it. Actually, rather than renaming, it is better simply to create a NEW account with an acceptable / non-conflicting name by using "vnmrj adm" ("auto1" is OK) and all attributes / settings of the account "auto". Then log in as that new user and import all relevant data into the new account - e.g., in a terminal window use

```
        cd ~auto
        tar cf - mydata archive vnmrsys/data | (cd ~; tar xfBp -)
```

Of course, you could also copy the entire "vnmrsys", even all visible files:

```
        cd ~auto
        tar cf - * | (cd ~; tar xfBp -)
```

It is preferable NOT to copy the "dot" files (by using "." in lieu of "*" in the command above), as several of these files may have the user name encoded. You should then be ready to start using the new account. If after copying all files (second example above) things don't work as expected, you may want to call "makeuser" in a shell window (as that new user, without argument) in order to install a set of standard VnmrJ files / restore standard VnmrJ conditions, or alternatively, update the user through "vnmrj adm".

If you have data on a second disk or another disk partition, you don't need to copy these data, but you can transfer the ownership as root, e.g.:

```
        su
        cd /data
        find . -user auto -exec chown -h auto1 {} \;
```

and of course you may want to rename directories with the name of the old user "auto" to the name of the new automation user.

ONLY when everything works OK you should consider deleting the old "auto" user through "vnmrj adm" (drag the user to the trash can, then empty the trash can); if this leaves behind unwanted remainders of the deleted account, you can call the Linux admin utility "System" -> "Administration" -> "Users and Groups" to ensure that the account definition is deleted at a Linux level - and if the user's home directory persists, that can be deleted (by root) as the very last step.

[Agilent MR News 2009-10-08]

2009-11-04:

CUSTOMIZING VnmrJ:

Customizability has always been one of the strong features in Varian NMR / MRI software, and just as in VNMR 6.1C and older versions, the VnmrJ software can be modified / expanded to accommodate local preferences and operating procedures, and to simplify the user interaction, often down to a few key strokes, or by implementing "push button" operation for frequently used steps. Even though the functionality of VnmrJ has grown substantially over the years, the customizability has of course remained - it has even been extended:

- many utilities that used to be built-in, pre-compiled commands have been converted to macros and are therefore customizable;
- full customizability has been built into the Tcl-dg parameter interface (VNMR 6.1C) and into the VnmrJ parameter panels and other GUI components;
- arbitrary built-in commands can be hidden using "hidecommand" and replaced by macros that may or may not call the hidden command besides performing customized tasks, see Agilent MR News 2002-03-23;

- the customizability has been further facilitated by the addition of the "appdirs" feature in VnmrJ 2.2C & up, which permits easily enabling or disabling entire "customization suites", see Agilent MR News 2009-03-12.

On the other hand, it can't be denied that the overall software complexity has grown considerably over the years, and while typical customizations in the interface used to be a matter of changing one or a few macros, this may no longer be the case, especially if the changes affect VnmrJ panel and/or menu features. Also, macros are used by other macros, by menus and panels, which may complicate the issue (so, checking what other macros call a given macro or command is no longer good enough!). This does NOT imply that customization has become impossible or impracticable: it just means you need to be careful about changing system macros, as this may alter other software features and may even cause other macros, menus, panel buttons, etc. to fail. The primary conclusion from this is NOT that you should stay away from customizing VnmrJ - but before just jumping in and changing macros in "/vnmr/maclib", think about a proper and more careful approach:

- first and foremost: you sure don't want to reinvent the wheel if there is no absolute need to do so! In other words, make sure you check whether the intended capability has already been built into the software - see the note below for an example;
- ALWAYS place customized files (macros, parameter sets, templates, etc.) in a user's local directories, and after adding your changes, thoroughly test the modified files under all aspects of local system use;
- if you have users running different appmodes (e.g., imaging vs. liquids) and/or different "appdirs" settings in VnmrJ 2.2C and later (e.g., with or without BioPack or Chempack 4.1) and you want everybody to use the modified features, have these users test your files as well;
- once you are SURE the new versions suit the intended purpose, you can make these files available to everybody. In versions prior to VnmrJ 2.2C you may place them in the appropriate directories in "/vnmr", after having made backup copies of the original files. In VnmrJ 2.2C and later version it is actually easier and safer to use the "appdirs" approach to include the test user's "~/vnmrsys" or an equivalent, stand-alone directory tree through an "appdirs" entry for all or the appropriate users, see Agilent MR News 2009-03-12.

[Agilent MR News 2009-11-04]

CUSTOMIZING LOCK / SHIM CONDITIONS IN CHEMPACK 4.1 AUTOMATION:

Just for illustration of the points made in the article above: one user of VnmrJ 2.2D and Chempack 4.1 wanted to change the default values of the "alock" and "wshim" parameters that are offered to a user when submitting a sample in automation. Having used many older software versions such as VnmrJ 1.1D and VNMR 6.1C, it seemed natural simply to have a look at "maclib" to find out which macro to alter in order to implement the desired change. That user spent two full days looking for the macro or parameter file responsible and finally gave up, asking for help. The editor checked with Krish (principal author of Chempack 4.1), to make sure he would provide the most efficient answer / solution - and here's the outcome:

- what this user wants to change is a matter of user (or site) preferences, and therefore is described "/vnmr/chempack/CP4_doc/cppreferences.pdf" which is part of the Chempack 4.1 documentation package. The solution:
- Select "CPpreferences" from the "Edit" menu, which opens the "preferences" pop-up. Under the "Queue" tab, there are two checkboxes for the default selection for autolocking and autoshimming.

Krish provided some additional insight into the sequence of events with such user-specific settings.

- When a user starts a "New Study" to submit his/her sample(s), these default values are read.
- The users, of course, have the option to override these defaults for their sample(s) in the "Start" -> "Standard" panel.
- When the sample(s) is/are submitted to automation, these "alock" / "wshim" values (either the default OR the user's overriding selection) will be recorded for that/those sample(s).
- At run time - i.e., when that/those sample(s) are actually run - these sample specific "alock" / "wshim" values (as chosen at submit time) will be used.

Note that in order to offer all this flexibility, such default parameter settings are neither fixed in stored parameter sets (such as "parlib" or "stdpar"), nor anywhere in "maclib" - which explains why that user's macro

search was unsuccessful.
[Agilent MR News 2009-11-04]

2009-11-12:

SHORT-CUTTING VnmrJ ADMINISTRATION / CONFIGURATION?

Just as another example to the recommendations that we posted in Agilent MR News 2009-11-04 about first looking for GUI features doing the desired task in VnmrJ, rather than starting to dig in the wealth of macros to do things "the traditional / old-fashioned way": this not only applies to customization of experiment setup, acquisition, processing and plotting - it also applies to the VnmrJ / spectrometer configuration and administration. Even if you have used Varian NMR software for the past 20+ years and believe you know from past releases how things work internally - you should always aim at using the VnmrJ administration and configuration utilities rather than a "quick, traditional bypass". In other words: use "vnmrj adm" for VnmrJ administration, and "Edit" -> "System settings..." (-> "System config") for configuring the soft- and hardware, unless you are SURE you know what ALL the underlying actions are. For example: in the "old days", calling "traymax=0" on the command line might have been sufficient / adequate to disable the sample changer (and conversely, "traymax=50" or "traymax=100" to re-enable it) - with VnmrJ we STRONGLY recommend using the "System config" utility to do such tasks. A user recently just went through a lengthy and nerve-wrecking "can't get automation to work again after an SMS repair" experience because of exactly this!
[Agilent MR News 2009-11-12]

2009-12-22:

IMPROVED DISPLAY FONTS WHEN USING VnmrJ IN KDE:

When using VnmrJ under Linux, Eugenio Alvarado (University of Michigan, Ann Arbor) prefers KDE over the Gnome GUI. VnmrJ works just fine under KDE, except that Eugenio found the fonts in VnmrJ to look very crude under KDE, and other Java applications show the same problem. It turns out that while in Gnome, font anti-aliasing appears to be active / turned on by default, while in KDE the font anti-aliasing (which smoothes the appearance of fonts that are defined or generated as a bit map) is not active with the default settings. The editor actually found that this "KDE deficiency" (i.e., the lack of font anti-aliasing) is also affecting VnmrJ when displaying remotely (tested with VnmrJ 2.3A), either with standard X.11, or by tunneling through "ssh" (i.e., without setting the "DISPLAY" environment variable in ".login").

Eugenio found an easy solution for this issue: one can edit the wrapper script "/vnmr/bin/vnmrj" by inserting the line
```
        -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true \
```
just prior to the last line reading
```
        vnmr.ui.VNMRFrame >>/dev/null &
```
Testing revealed that for using VnmrJ under the (default) Gnome GUI this alteration makes no visible difference (i.e., there are no adverse effects), while it not only improves the font display for VnmrJ under KDE, but also when displaying VnmrJ remotely through standard X.11 or when using tunneling through "ssh". Depending on the font selection, the effect may be subtle, possibly hardly noticeable (maybe also depending on the graphics hardware / display) - but it is definitely worth trying.

The amount of extra CPU usage due to the anti-aliasing should be negligible, but it is possible that when displaying VnmrJ over remote, slow connections, this may cause a certain amount of extra slow-down (due to the potentially larger color map), see also Agilent MR News 2009-11-12. The editor's tests were done over a fast, local connection - noticeable changes in the performance could not be observed.

Thanks a lot, Eugenio, for this input!
[Agilent MR News 2009-12-22]

2010-02-03:

SAVING FIDS FROM SPECTRAL ADDITION / SUBTRACTION:

VnmrJ can be used to add / subtract 1D FIDs and spectra, using the commands "clradd", "add" / "sub" / "addi" (for FIDs), and "spadd" / "spsub" / "addi" (for spectra); there is a second use to the "clradd" / "add" commands, for building "artificial" arrays from separate 1D FIDs / files, which we discussed in Agilent MR News 1996-08-21 and in Agilent MR News 2002-11-25.

When adding / subtracting FIDs, the result can be accessed in "exp5" which serves as the "add/sub buffer" (make sure you have saved any valuable data in that experiment before calling "clradd"!), and it can also be saved as regular VnmrJ FID file ("*.fid"), using "svf". Note that the saved parameters and the text file (if unaltered) will in this case correspond to those from the FIRST component added into "exp5" through "add" or "sub", and that the resulting FID or its parameters will bear no direct, clear indication that the data are the result of an addition, subtraction, or a linear combination of multiple 1D FIDs in general (it's up to you to add the appropriate comment to the text file, if necessary), nor will the text or the "processed" parameter tree (the acquisition parameters in particular) from any but the first component be preserved / carried forward into the resulting, saved FID file.

Compared to processing FIDs, there are some peculiarities with adding or subtracting spectra in VnmrJ:

- for one, there is additional flexibility in "addi", permitting not just scaling, but also shifting and zooming of one component relative to the other;
- on the other hand, even though the result (a linear combination of 1D component spectra) is equally stored in "exp5", saving the result is not quite so straightforward, as in VnmrJ we usually don't save transformed data: VnmrJ FID files include all relevant processing parameter, such that typically after recalling a data set you can simply retransform the FID in order to obtain the previous spectrum (provided you saved the data AFTER having done the full processing). As there is no command in VnmrJ to save spectra / processed data, there is equally no standard command to IMPORT / READ processed 1D (or nD) spectra.

The "brute force" solution for the above deficiency (the lack of an ability to save processed spectra) would be to use Linux commands to archive the entire "exp5", e.g.:

```
        cd ~/vnmrsys
        cp -rp exp5 exp501
        tar cf - exp501 | bzip2 > exp501.tbz2
```

By copying the experiment first you will not overwrite exp5 when unpacking the data:

```
        cp exp501.tbz2 ~/vnmrsys
        cd ~/vnmrsys
        bzcat exp501.tbz2 | tar xfBp -
```

followed by "jexp(501)" on the VnmrJ command line to access the restored data. Another little VnmrJ quirk: if you use Linux commands to rename experiment directories to names between "exp10" and "exp9999", you are bypassing the VnmrJ "cexp" utility for creating new experiment directories, and hence the equivalent "jexp#" macro ("jexp10" up to "jexp9999", or "jexp501" in the above example) will NOT exist. You therefore need to use "jexp" with a numeric argument instead (note that there are subtle differences between "jexp(#)" and "jexp#": "jexp" does NOT refresh the graphics display - see the "Command and Parameter Reference" manual).

In some environments, however, it would be more appropriate (or it fits the workflow better) to save such results of "spadd" / "spsub" / "addi") as FIDs, even if spectra were added. Here's a solution how this can indeed be achieved in VnmrJ, assuming the result of the addition / subtraction is in "exp5":

```
        jexp5
        cexp(6)                 [ if "exp6" is not defined yet ]
        ft('inverse',6,1)
        jexp6
        svf                     [ or "svf('file_name')" ]
```

In this recipe, we perform an inverse FT into the specified experiment. The expansion factor specified in the last argument remains set to 1 (i.e., we don't expand the frequency domain before doing the inverse transformation). A minor point: unlike stated in the manual, the result of the inverse FT can be stored in ANY DEFINED experiment up to "exp9999", not just in workspaces up to "exp9". Note that the above recipe only works because VnmrJ performs a COMPLEX addition into "~/vnmrsys/exp5/datdir/data", rather than just adding the phased traces into "exp5/datdir/phasefile".

Last, an addendum to the above statement that VnmrJ can't store transformed spectra: User Library contributions permit exporting transformed 1D spectra or single 1D traces from nD spectra into universal ASCII formats:

- "maclib/writespec" writes an ASCII file (Y values only) from the DISPLAYED portion of the PHASED, active 1D spectrum / trace;

- "bin/parhandler" includes utilities for exporting ENTIRE 1D spectra (Y..Y, or X,Y .. X,Y) either in "pure ASCII" (no parameters) or in JCAMP-DX format (the latter includes parameter information).

All this is discussed in detail in articles collected in a FAQ document "Exporting VnmrJ / VNMR Data" ("faq/data_export") in our on-line User Library

        http://www.varianinc.com/products/nmr/apps/corner.html

where you also find the software contributions mentioned above. Note that there is currently no facility for REIMPORTING such spectra into VnmrJ, irrespective of the format of the saved data.

Thanks to Albin Otter (University of Alberta, Edmonton) for suggesting this topic for the newsletter!

[Agilent MR News 2010-02-03]

2010-02-19:

SIMPLE TRICKS FOR AVOIDING PROGRAMMING ERRORS:

Unlike communication between humans (language, writing, etc.), programming languages (i.e., the interface between humans and a computer) are typically anything but error-tolerant: they involve strict, tight syntax rules, and what may look like a "minor error or omission" to a human / programmer will usually cause computer programs to "fall over", i.e., abort, issue error messages, or do something completely unintended - often with fatal consequences (such as data loss, maybe even damage to the software or OS installation, etc.). It is therefore not uncommon that the bulk of the effort in writing a program (pulse sequence or other compiled program, shell script, VnmrJ macro, etc.) is spend on carefully checking / testing and debugging the software. Any help in this tedious and important task can therefore reduce the time required to program a certain task, and can at the same time reduce the number of software bugs and the likelihood for fatal failures, and avoid losses in time and productivity with the end user. Improved compilers and interpreters, debuggers, syntax checkers are helpful, but that's usually not enough. We have in the past given hints on how to minimize the number of programming errors, e.g.,

- strictly adhere to a clean indentation scheme - this has been discussed in Agilent MR News 1999-04-15 for C programs / pulse sequences, and in Varian MR News 2009-07-04 for MAGICAL / VnmrJ macros;
- comment / document your programs, even if this looks unnecessary, redundant or even annoying (see Agilent MR News 2009-07-04 for macros);
- use a "top-down" approach when writing software - this does NOT mean that you start writing the first line and continue on to the last one, but that you start with the major logical constructs and only then start filling in the details - see again Agilent MR News 2009-07-04 for macros;
- besides keeping an eye on the consistency and the proper nesting of logical constructs, always keep watching out for proper pairing and nesting of parentheses, brackets, braces, as well as double, single and back quotes, keep an eye on proper comment termination, etc.;
- use meaningful variable / parameter names;
- in shell scripts, check for (and eliminate) extra trailing blanks, as these may have adverse effects, especially after continuation backslashes with long command calls spread over multiple lines in shell scripts;
- frequently test your macro / program while writing it - this pretty much requires proper top-down programming in first place;
- when dealing with human interaction, DON'T assume correct input / responses to feedback queries, etc., DON'T assume that you are in the correct directory, that required files are there, etc., but have the program check EVERY human input / command line argument and any condition (parameter, file or specific file contents) that is beyond its own, internal control. For more information see
  - Agilent MR News 2004-01-24 for type checking in pulse sequences,
  - Agilent MR News 2003-07-27 and Agilent MR News 2003-07-21 for argument handling in macros,
  - Agilent MR News 2004-07-07 for input handling in macros, and
  - Agilent MR News 2005-11-02 for input handling in shell scripts.

As for MAGICAL macros, there are various possibilities for error checking and debugging, see Agilent MR News 2004-07-07, Agilent MR News 2008-04-17 (and articles referred to therein), and Agilent MR News 2008-04-23.

Overall, the key is to observe ALL of these hints! Fortunately, there was a lot of progress in the programming environment since the days when programmers were sitting in front of dumb, monochrome 80x24 ASCII terminals, working on complex software modules of many hundred, if not thousands of lines, using "vi" or even simpler / more basic text editors:

- Compilers and interpreters have been improved to provide better checking and more accurate feedback on syntax errors (some compilers have special syntax checking / debugging options that can be activated upon demand); in professional programming environments, sophisticated debugging and tracking utilities are available - though these are far beyond the scope of pulse sequence, shell script or macro programming tasks at the level of typical NMR users.
- Editors such as "vi" have special modes that visualize logical structures, e.g., by indicating the corresponding opening parenthesis when typing a closing parenthesis.
- More importantly, newer Linux editors, particularly "vim" ("vi", improved) or the KDE editors "kwrite" and "kate" may use color coding to visualize logical structures and differentiate variable names, strings, comment, etc. in shell scripts, C and other, common programming languages, as well as even system administration files such as "/etc/passwd" and the like.

That last feature is particularly useful and efficient in avoiding errors already while coding new software - once you have started using this feature, you don't want to miss it! Unfortunately, this nice feature does not readily apply to MAGICAL macro writing, as this is a "local language", specific to VNMR and VnmrJ software, and not in use anywhere else. Luckily, these editors are very configurable and can be taught to apply color coding to MAGICAL macros, too, see the article below!

[Agilent MR News 2010-02-19]

NEW IN THE USER LIBRARY - MAGICAL SYNTAX HIGHLIGHTING FOR LINUX EDITORS:

As outlined in the article above, syntax highlighting is an extremely useful tool in programming in general. Linux editors such as "vim" ("vi" improved), or the GUI-based Linux / KDE editors "kwrite" and "kate", are configured to recognize programming languages such as C, as well as various Linux shell languages and typical Linux configuration / administration files. So far, no such help / comfort was available for editing MAGICAL macros for VnmrJ.

However, Eugenio Alvarado (University of Michigan, Ann Arbor) has looked into the relevant configuration files and documentation, and he is now sharing a User Library contribution "misc/MAGICAL-hilite" that fills this gap, and which can be downloaded by sending a message with the line

```
        misc/MAGICAL-hilite
```

to the on-line User Library e-mail responder at

```
        userlib.request@varianinc.com
```

(see Agilent MR News 2010-02-11 for more information on e-mail downloads). A download Web page for this contribution is currently not available, as the updating of our public Web site is still restricted to "slow / manual mode", see Agilent MR News 2009-12-22.

The e-mail responder will send you the contribution along with a README file featuring detailed instructions for installation and use of this new utility. Note that

- the installation differs from that of most other contributions, in that the utility should be extracted into the user's home directory, using
  ```
  cd /vnmr/userlib
  ./extract misc/MAGICAL-hilite ~
  ```
- this will merely extract the package, but NOT activate the utility: in your home directory, you will find two directories "dot-vim" and "kate", along with a copy of "MAGICAL-hilite.README" which contains further instructions.
- one important point is that the configuration & setup for "vim" ("vi") depends on whether you want to use this feature for individual, specific users only, or whether it should be activated system-wide, for all users; the instructions for the latter case (to be performed by root) also depend on the version of "vim" (late versions of RHEL 4.x, as well as RHEL 5.x come with "vim" version 7.0 or higher, early versions of RHEL 4.x use "vim" version 6.x and below). This is the main reason why we have not (yet) set up a fully automatic installation utility: please carefully read and follow the instructions in "MAGICAL-hilite.README" for the additional steps.

The added configuration files fulfill two tasks:

- the file "filetype.vim" (to be added to the existing definition) sets the rules that permit "vim" to recognize a macro as such (Eugenio's current version implies that a macro is located in a directory "maclib" or that it has the typical Varian header line used in all macros that are distributed with the VnmrJ software);
- a second file, "~/.vim/syntax/magical.vim" for a local activation, contains

   the rule set for MAGICAL syntax highlighting.
Especially the second file with the rule set is easy to understand, and easy to modify, restrict or expand the rule set. The editor and the author both welcome any feedback / comments / suggestions on these settings.
  The file "MAGICAL-hilite.README" also contains instructions for adding the same enhancement for the GUI-based Linux / KDE "kwrite" and "kate" editors. The highlighting capabilities in these editors are at least as powerful and flexible as those in "vim" - but unfortunately, the implementation is somewhat trickier:

- the highlighting rules are defined in a much larger, more elaborate XML file "magical.xml";
- so far, Eugenio has not found a way to make these editors recognize MAGICAL macros automatically and with a simple set of rules - instead, such file type recognition requires a small file ".kateconfig" to be dropped in all relevant directories, i.e., in "~/vnmrsys/maclib", "/vnmr/maclib", "/vnmr/chempack/maclib", "/vnmr/biopack/maclib", etc. (i.e., all other active "maclib" directories as activated through "appdirs").

If you are an "old time, hard-core macho programmer", you may at first be reluctant to use syntax highlighting; however, we would like to encourage you to give this at least a try - Eugenio states: "In many, many occasions it has been useful to me, for example to quickly locate comments or syntax errors, like forgotten closing quotation marks, in macros. It makes it easier to identify the different components in the text."
  Again, feedback / comments / ideas for further enhancement in this utility are most welcome! You may provide such feedback simply by replying to the Agilent MR News e-mail - the editor will pass on such input to the author.
  Thanks very much, Eugenio, for this contribution!
[Agilent MR News 2010-02-19]

2011-02-22:

ARRAYED PARAMETERS AND PARAMETER PANELS IN VnmrJ:

  In VnmrJ, parameter arrays are usually set up using the "Parameter Arrays.." utility from the "Acquisition" menu - but that should not preclude entering arrayed parameter values directly in to parameter entry widgets in the panels. An indeed, in most cases you can simply enter parameter arrays as a list of comma-separated values into the appropriate widgets. Once that set of values is entered / accepted, the widget will display "array" to indicate arrayed values: as arrays can be very long, it doesn't make sense even just to try showing arrayed values in the widget. To see the array, you have two options:

- use "Acquisition" -> "Parameter Arrays.." to see the current array(s), or
- switch to the "Text output" pane and type "da" on the command line.

There are some rare cases where things aren't so simple - but in order to discuss this we first need to look at how VnmrJ handles parameter panels and their definitions.
  When you edit a parameter entry widget on a VnmrJ panel, e.g.: select the "Acquisition" -> "Pulse Sequence" panel for a simple "s2pul" 1D experiment and from the "Edit" menu select "Parameter Pages.."), then double-click on the entry widget for the observe pulse width ("pw"). In the central part of the panel editor window you should now see a widget definition reading

```
        Vnmr variables:         pw
        Value of item:          $VALUE=pw
        Enable condition:       on('pw'):$VALUE
        Vnmr command:           pw=$VALUE
        Decimal places:         2
        Disable style:          Grayed out
        ...
```

Some explanations on these definition entries:

- The first field defines which VnmrJ variables will cause that widget to be updated (just "pw" in this case).
- The line "Value of item" defines the value shown on the panel (this value is internally stored in a local variable "$VALUE"). As "pw" is a "pulse" type parameter and therefore defining values in microseconds, the value shown in the widget will be in microseconds, too.
- The next line ("Enable condition") defines how the VnmrJ GUI evaluates whether the parameter ("pw") is active / enabled ("on"): here, the output of the "on('parameter_name')" command is stored in another instance of the local parameter "$VALUE".
- The most important part of the definition is the actual parameter entry:

the line "Vnmr command" defines how values typed into the widget are filled into the parameter - in this case a simple, direct parameter entry, i.e., "pw=$VALUE".
- The following line defines the numerical format of the displayed value; this has no implication for the numerical precision of the actual parameter value (internally, all numeric parameters are handled as double-precision floating point numbers, see Agilent MR News 2008-04-29).
- The last line above ("Disable style") defines that the widget will be grayed out if the parameter is "off" as per "Enable condition" above (you could also make the widget disappear / be hidden when the parameter is disabled).

For those interested in the XML parameter template definition, in this case in the file "/vnmr/templates/layout/s2pul/PulseSequence.xml": the key parts of the above widget definition translate to

```
        vq="pw"
        vc="pw=$VALUE"
        set="$VALUE=pw"
        show="on('pw'):$VALUE"
        digits="2"
```

in the relevant part of the XML file. Note that the definition in the above example is specific to the type of widget used for entering "pw" - other widget types use entirely different fields (and a different number of fields).

A comprehensive discussion of VnmrJ parameter panel definitions is beyond the scope of this article - but let's see what happens if the parameter value is arrayed. If "pw" is arrayed, "$VALUE" in the section "Value of item" will contain the list of values, e.g.: "1,1.5,2,2.5,3,...", which will cause the VnmrJ GUI to display the string "array" in lieu of a numeric value. When you ENTER the array "1,1.5,2,2.5,3,..." in the widget, the expression "pw=$VALUE" (in this example "pw=1,1.5,2,2.5,3,...") will be executed by the VnmrJ command interpreter, the parameters "pw", "array", and "arraydim" will be adjusted by the MAGICAL engine, and the widget in the panel will be refreshed because the parameter "pw" was altered by the entry.

In the next issue we'll have a look at complications that you may encounter when the parameter entry is not just a simple assignment.

[Agilent MR News 2011-02-22]

2011-03-04:

PARAMETER PANELS AND NON-STANDARD PARAMETER UNITS IN VnmrJ:

Back in the years around 1986, when the VNMR (and later VnmrJ) parameter handling scheme was defined, we essentially inherited / transformed the parameter concept used in the previous software architectures, featuring time-related parameters for delays and RF pulses, in seconds & microseconds, respectively, while msec time events (as now frequently used for gradient pulses) were not deemed important enough to deserve their own parameter category. Since then we have not made fundamental changes to the parameter concept in VnmrJ (partly in order not to break existing macros, panels, menus, etc.); however, nowadays, the majority of the pulse sequences used in liquids NMR (let alone of course those used in imaging) use gradient pulses with typical durations in the order of milliseconds, and they may use spinlocking and other pulse sequence elements of similar duration. In the absence of a true millisecond parameter type, how do we best handle this? There are two risky options:
- You can use a "delay" parameter, but in VnmrJ treat it as msec value - and in the pulse sequence you divide the value by 1000.0 before using it, so effectively the parameter values represent millisecond units. This option is relatively safe in that there is little danger that the console will be trying to execute time events of excessive length (possibly with RF or a gradient turned on). Possible hiccups are that the parameter limits for delay parameters may be inadequate for msec values; also, a user who is not aware of that "internal downscaling" may still enter a value in seconds (e.g., 0.02) and the system then performing a time event of a few usec only. This concept may confuse users, as the parameter itself bears no indication of the division by 1000.0 within the pulse sequence. Finally, there is a danger of such parameters being carried over to pulse sequences that do NOT follow the same convention, and then these sequences may indeed try executing inappropriate time events of several (many) seconds duration. The internal RF and gradient load checking software SHOULD protect your hardware, but this could still cause extreme experiment durations, likely

```
   with no result at all.
 - Conversely, there's the option of using "pulse" type parameters for msec
   units, and multiplying the parameter value by 1000.0 in the pulse sequence.
   This is more dangerous than the option above, as a user who is not aware of
   the pulse sequence logics may look up the parameter type, find that it is a
   "pulse" parameter, and then enter tens of thousands of "usec", possibly
   yielding an effective duration of many seconds. Also here, the RF and
   gradient load checking software should protect your hardware - but it's
   better not to rely upon protection software exclusively.
Note that with either of these options, "dps" should indicate the actual event
duration, providing additional protection - but overall, both solutions are
confusing and should probably be avoided. This leaves us with the two obvious,
slightly inconvenient options of
 - entering large numbers into a "pulse" type parameter - with the added
   complication that pulse parameters by default have an upper limit of 8190
   (usec), to circumvent this restriction for larger msec events you could
   either expand the limits or enter the value using "setvalue";
 - entering small fractions of a second into a "delay" type parameter; this
   avoids the complication of having to alter the parameter limit, but typical
   (default) parameter listings or panels may fail to display such values with
   sufficient precision.
This is where VnmrJ panels come in handy. In the last issue (Agilent MR News
2011-02-22) we discussed how parameter entries work in VnmrJ panels: here now
we can properly "emulate" msec parameter entries without the inconveniences
(let alone the risks) of the above solutions:
 - for a regular parameter (delay, pulse), the relevant part of the widget
   definition involves the entries
        Value of item:          $VALUE=d1
        Vnmr command:           d1=$VALUE
   as discussed in Agilent MR News 2011-02-22; the first line defines what
   value the parameter entry widget displays, and the second line sets the
   VnmrJ parameter from a (set of) value(s) entered into the widget. In the
   associated XML file, this translates to the lines
        set="$VALUE=d1"
        vc="d1=$VALUE"
 - now, for a msec type parameter entry we can use a regular "delay" type
   parameter, label the widget with "msec" (or "ms", more properly) as unit
   and do the necessary "translation math" inside the widget definition, i.e.,
   we multiply the delay value by 1000.0 for the value display, and we divide
   entered values by 1000.0 for the parameter entry, e.g.:
        Value of item:          $VALUE=gt1*1e3
        Vnmr command:           gt1=$VALUE/1e3
In VnmrJ, the user should not have to deal with the actual parameter entry
mechanism (i.e., the associated command line entry / syntax), nor even the
name of the parameter involved, so the extra math involved (and the actual
parameter values) are handled internally, i.e., are not shown to the user.
  There is one problem with the above definition: if a user directly enters
an array of values (e.g., "1,2,3,4,5,6") into such a widget, this array is
stored in the internal variable "$VALUE"; the ENTRY part of the above widget
definition then translates to
        gt1=1,2,3,4,5,6/1e3
i.e., only the LAST value of the array is subjected to the appropriate scaling
while all other values would in this case contain seconds rather than msec
values! This has caused problems in the past, as seen in two bug reports for
BioPack "ul_biopack.j2131" and "ul_biopack.j2228" (both are fixed in the
current version of "psglib/BioPack" in the on-line User Library). A proper
implementation that avoids this hiccup is to use
        Vnmr command:           gt1=[$VALUE]/1e3
in the VnmrJ parameter entry widget definition shown above, or, in the XML
file, using the line
        vc="gt1=[$VALUE]/1e3"
With this syntax, ALL elements of the value array will be divided by 1000, as
intended. There is no need to use the "bracket syntax" for the display part
("Value of item", or "set=" in the XML file) of the widget definition, as the
VnmrJ GUI will simply display the string "array" for arrayed values.
  In the next issue we'll discuss the above "bracket syntax" in more detail,
its features and limitations, and its use in macros and/or from the command
line.
                                                  [ Agilent MR News 2011-03-04 ]
```

2011-03-14:

LITTLE KNOWN TOOLS FOR ARRAY HANDLING IN MAGICAL:

In the last issue (Agilent MR News 2011-03-04) we showed how an improper panel definition syntax could cause havoc with the direct entry of parameter arrays in VnmrJ panel widgets - and we gave a recipe and examples for the proper syntax that avoids such potential issues. In this follow-up note we want to show you how you can use the associated MAGICAL syntax element on the VnmrJ command line and in your macros. The editor would like to thank Dan Iverson (Agilent, Santa Clara, CA) for this information.

Traditionally, entering arrayed values could be tedious on the command line; you had to enter the array by listing all values at once, e.g.:

```
        pw=5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24
```

or in batches:

```
        pw=5,6,7,8,9,10,11,12,13,14
        pw[11]=15,16,17,18,19,20,21,22,23,24
```

This latter syntax can also be used to alter individual elements or a set of consecutive elements in an array; note that using an array index on the left-hand side of such an assignment statement assumes that the specified array element or the preceding element already exist - otherwise an error will be issued.

At some point we added the "array" command in order to facilitate entering linear arrays: the above array definition can be specified in one command:

```
        array('pw',20,5,1)
```

However, the "array" command only works for entering ENTIRE, linear arrays at once, you cannot expand or amend existing arrays - and the direct entry of array values with very small or very big numbers can be tedious, e.g.:

```
        gt1=0.0002,0.0004,0.0008,0.0016,0.0032,0.0064,0.00128,...
```

or

```
        gt1=2e-4,4e-4,8e-4,16e-4,32e-4,64e-4,128e-4,...
```

An additional MAGICAL utility for entering array values was introduced with VnmrJ 1.1D in the context of VnmrJ panel layout definitions (see Agilent MR News 2011-03-04): the "square bracket syntax" (NOT available with any VNMR software version, nor with VnmrJ 1.1C or earlier releases) facilitates working with arrays. The "core element" can be illustrated with the following sample syntax

```
        val=[a,b,c,d]/e
```

yielding the same as

```
        val= a/e, b/e, c/e, d/e
```

Here, "val" is the name of a variable (e.g., "d2", or "$value"), "a" .. "d" are numeric values, and "e" is either a numeric value or a numeric parameter. The "[]" can enclose a single value or expression or an array of values or expressions. Any mathematics applied to the "[]" element will be applied individually to each element within the "[]". Some examples.

```
          Entry                           Result
        nt=[1]                          nt=1
        nt=[1,2,3]                      nt=1,2,3
        nt=[1,2,3]*10                   nt=10,20,30
        nt=22*[2*3,r2+6,trunc(r3)]+2    nt=22*2*3+2,22*(r2+6)+2,22*trunc(r3)+2
```

As a side effect of this implementation, you can also use "[]" to specify the precedence in expressions, just like "()". For instance,

```
        nt=[2*[3+4]]
```

yields "nt=14".

Note that there are limitations if the "[]" element is used as part of a mathematical expression, e.g.:

- only a single "[]" element is allowed; an entry such as
  ```
        nt=[1,2]*[3,4]
  ```
  is not allowed - you would get an error message
  ```
        No more than one [--.--]
  ```
- when used in expressions, the "[]" element cannot be mixed with the standard comma (,) arraying element, e.g.:
  ```
        nt=1,[2,3,4]*10
  ```
  is not allowed. You will get the error message
  ```
        Cannot combine , with [--.--]
  ```

These restrictions only occur if mathematical operators are used and the "[]" element itself contains a comma. Simply listing multiple "[]" elements, or combining them with the comma element is OK:

```
          Entry                           Result
        nt=[1,2],3                      nt=1,2,3
```

```
        nt=[1,2],[3,4]                   nt=1,2,3,4
```
What also does NOT work is the following example:
```
        pw=8,9,10,11,12 p1=[pw]*2
```
or
```
        $val1=8,9,10,11,12 $val2=[$val1]*2
```
i.e., you can NOT place an arrayed variable inside the square brackets and expect this to be taken for its arrayed values: in the above case "[pw]" will simply be interpreted as "[pw[1]]" (and "[$val1]" as "[$val1[1]]"), hence "p1" and $val2 in these examples will be set to 16. This appears to contradict the syntax used in VnmrJ panels, as indicated in Agilent MR News 2011-03-04:
```
        Vnmr command:           gt1=[$VALUE]/1e3
```
However, this is not the same environment, as in the context of VnmrJ panels, the parameter "$VALUE" is substituted by its value BEFORE the expression is evaluated, while in macros and on the command line, the square bracket syntax is resolved / interpreted ALONG WITH (or maybe even before) variable values are looked up. What WOULD work is
```
        pw=8,9,10,11,12
        p1=[pw[1],pw[2],pw[3],pw[4],pw[5]]*2
```
but that obviously defeats the idea of having a simplified syntax for array handling. Of course, you can still use the "array" utility to REdefine the complete parameter array rather than manipulating its values - as long as the array is a linear sequence of numeric values. If you would like a similar utility for entering / defining arrays with exponentially spaced values, there is help: the editor has just (re-)posted his "maclib/xarray" contribution which years ago used to exist inside a larger User Library package; for details please visit the on-line User Library via our NMR Software Corner at
        http://www.chem.agilent.com/en-US/Support/Pages/default.aspx
[Agilent MR News 2011-03-14]

2011-03-18:

ABSOLUTE INTENSITY SPECTRAL SCALING IN VnmrJ:

Theoretically, when you accumulate a simple 1D spectrum (single pulse excitation, no saturation, i.e., "d1" + "at" >> T1*), you would expect the size of the NMR signals for your compound to grow linearly with the number of accumulated scans ("ct"), while the noise - due to its random nature - will only grow with the square root of "ct". This is true for the numbers in the FID file - though the direct interpretation of the numeric contents of a FID (using "ddff(1)" in VnmrJ, or by exporting a FID into an ASCII file, e.g., using "writefid('file_name.txt',1)") is non-trivial, especially in the presence of substantial noise components. Of course we can look at transformed data in "ai" (absolute intensity) mode, where the spectrum supposedly is only scaled by multiplying the result of the FT with "vs". Assume we have acquired an array of 1D spectra:
```
        nt=1,4,16,64,256,...
```
Given the above, one would expect a four-fold signal increase with every increment, while the noise should just double (hence the signal-to-noise ratio should double with every increment). However, what we see (e.g., with "dssh") is an array of 1D spectra with (more or less) equal signal height, while the noise decreases by a factor of roughly 2 with every increment (i.e., the s/n grows as expected, but there appears to be an extra, variable downscaling). Even when we look at the FIDs (e.g., using "dfsh"), we see the signal component (start of a regular FID) retain its size, while the noise part (the last part of the FID, assuming "at" >> T2*) goes DOWN by a factor of 2 with each increment in the above array. What is happening?

The explanation for these findings is that VnmrJ scales spectra as well as displayed FIDs by "1/ct", i.e., the intensity values in spectra and displayed FIDs are divided by the number of accumulated scans for every increment, respectively. Note that VnmrJ actually does NOT use the "ct" parameter, as this isn't really usable in the case of an "nt" array - instead, VnmrJ takes this information from the header part of every trace in the FID file. This has been the way in which VnmrJ and VNMR (and even the Varian software prior to the introduction of VNMR, back in the years before 1987) have always worked. This behavior has some key advantages:
- if you watch the spectrum while acquiring (e.g., through "wbs('wft')"), you should see a stable signal amplitude while the noise gradually goes down in size, i.e., you don't need to re-scale the spectrum over the course of the experiment;
- the same holds true if you are watching the FID while acquiring: without

the scaling the initial part of the FID would (at least initially) increase rapidly, possibly hiding a clipped / distorted initial segment from the user;
- in the case of problems (e.g., with the phase cycling, or an experimental stability issue, see below), deviations from a constant signal amplitude in the downscaled spectrum are far easier to recognize than deviations from an expected increase in the signal height without downscaling;
- you can compare, add and subtract spectra independent of their respective "nt" / "ct" values. Needless to say that spectra / FIDs to be compared need to be acquired with the same gain and tuning, especially if spectra from different samples are involved (absolute quantitation is a topic beyond the scope of this article) - and keep in mind that the signal height is also dependent on the line shape, i.e., changes in shim quality, as well as lock instabilities (in the case of "nt" >> 1) will affect the signal height (but should have less of an influence on the absolute integral values).

Note that the raw data in the FID file (as seen via "ddff(#)", or as written out when exporting the FID to an ASCII file using "writefid" are NOT scaled by "1/ct".

The above data (down-)scaling turns out to be the most practical solution overall. If, however (e.g., for teaching / illustration purposes), you would like to see (or demonstrate) how the signals and the noise are growing in the course of an acquisition, or as a function of "nt" / "ct", you could set "vs" to a constant value, multiplied by "ct" (or "nt", if an "nt" array was used).

If you would like to see the signals grow out of a constant noise floor as the accumulation progresses (or as a function of "nt", you could set "vs" to a constant value, multiplied by "sqrt(ct)" or "sqrt(nt[i])". As an example: if you have performed an "nt" array, you could transform all traces, then set the vertical scale ("vs") for any trace and use a macro such as

```
dssh $dim=size('nt') $vs=vs $lastnt=nt[$dim]
$ix=1
repeat
  vs=$vs/sqrt($lastnt/nt[$ix])
  pl($ix)
  sc=sc+ho
  $ix=$ix+1
until $ix>$dim
full vs=$vs dssh
```

With this, the last trace should be scaled to the size shown with "dssh", the preceding traces should show roughly the same noise floor size, with the signals gradually growing out of the noise.

As long as the receiver gain & the probe tuning are not altered, the noise should behave as described above - however here are cases where the signals do NOT grow with "nt", and consequently, with the built-in downscaling, the signal height will NOT stay (approximately) the same across an "nt" array (or with increasing "ct"):
- In experiments with multi-quantum coherence selection through phase cycling the main (single quantum) signal subtraction will typically occur through alternating scans, i.e., with "nt=1" you essentially observe large (single quantum) signals, while with "nt=2" you would expect to see just the much smaller signals from double-quantum transitions - thereafter (i.e., for larger "nt" values), the signal should again follow the above rules.
- in ABSOLUTE VALUE 2D experiments with f1 coherence selection through phase cycling, you can expect to see a 50% signal loss at the point where phase cycling adds or subtracts the N+P signal component to / from the N-P part: if that coherence selection occurs with alternating scans, you should see a 50% signal loss (after downscaling) between "nt=1" and "nt=2". With phase-sensitive, coherence-selected nD experiments the N+P and N-P components are recorded in separate traces ("phase=1" vs. "phase=2"), and the above subtraction is done through processing ("wft2d" / "wft2da"), the signal size should therefore follow the above rules, again unless multiple quantum coherence selection is done through phase cycling.
- "nt" arrays such as "nt=1,2,4,8,16,32,64,128,..." (up to the length of the phase cycling in a given pulse sequence) can be used to verify the phase cycling: if in this array you observe an unexpected deviation from the regular signal/noise increase (no s/n gain, i.e., sudden decrease of the downscaled signal), that probably indicates an error in the phase cycling (again, unless multi-quantum coherence selection is done at this stage of the phase cycle).
- if you are merely looking at the signal height (rather than the integrals), and you observe a significant deviation from the behavior described above

(i.e., an unexpected partial signal loss at some point in a long term accumulation, or in an "nt" array as indicated above, this is likely caused by lineshape variations and could indicate a hardware or other instability (e.g., temperature fluctuations in the lab, or sporadic floor vibrations).
One last point: on older instruments (up to UNITYplus, GEMINI 2000, MERCURY, MERCURY-Vx, MERCURYplus, but also UNITY INOVA with "dsp='n'" or "dsp='r'"), we did integer signal accumulation with a word length of either 16 bits ("dp='n'", maximum numeric value 2^15-1 = 32767) or 32 bits ("dp='y'", maximum numeric value 2^31-1 = 2,147,483,647). Once these maximum values were reached (actually, just before this could happen), our software would downscale the FID and the data stream from the ADC by right-shifting the bits (dividing by 2). This could happen several times in experiments with very large "nt" values. The number of such downscalings is recorded in the "scale" value in the header of every FID trace (see "ddff(n)"). This downscaling is taken into account by our processing software and should not have any effect on the above signal intensity considerations (see Agilent MR News 1998-11-26 for related information).

[Agilent MR News 2011-03-18]

2011-07-18:

A WORD OF CAUTION ABOUT "shell" CALLS IN MACROS:

If you are an experienced shell script programmer who is also programming VnmrJ macros, you may sometimes be tempted to use a "shell shortcut": a shell script solution or a call to existing Linux commands for your programming task may spring to mind, while you would still need to think about a way to do this in MAGICAL. By intuition, you might consider writing a shell script that you can call from within a macro, e.g., using

```
        shell('script_name'):$res1,$res2
```

for more complex tasks, or simply by calling existing Linux commands directly in such a "shell" call. That's basically just fine - HOWEVER, our R&D group recently did some performance monitoring with VnmrJ macros, and they found that a "shell" call can (depending on the OS and the system architecture) be "expensive": on a typical Linux PC, each such "shell" call was found to take AT LEAST 25 ms, in other architectures the time consumption could be as much 50 - 70 milliseconds - and that's just the basic overhead with each such call (i.e., not counting the execution time within the shell call).
In many cases this still is absolutely negligible: you would hardly notice if a macro (with a few "shell" calls) consumes a tenth of a second extra to complete. However, the problem may be much more serious

- inside a macro loop that may run for tens or hundreds of cycles, or
- in (small) utility macros that are called numerous times in a given nested macro execution chain.

As a general recommendation, it is a good idea to AVOID "shell" calls if a) a MAGICAL alternative for a given task exists, and b) such a call is likely to be called frequently or in a loop.
One instance where "shell gurus" are tempted to resort to Linux utilities is in text string manipulations. Note that MAGICAL features a fairly complete set of string manipulation utilities (e.g., "substr", etc.), such that in this area there should be little need for "shell" calls. For details see the VnmrJ User Programming Manual.
Thanks to Dean Sindorf (Agilent R&D) and Bert Heise (Agilent, Yarnton, U.K.) for providing and forwarding that information!

[Agilent MR News 2011-07-18]

2012-05-30:

UPGRADING FROM AN EARLIER VERSION OF VnmrJ - REMINDER:

A fresh installation of VnmrJ (i.e., on a newly delivered PC, or on top of a newly installed operating environment) should be straightforward and easy, in that you should be able to follow the instructions in order to get a functional, "standard" software setup. Software UPGRADES or reinstallations, however, are more complex / demanding, in that you are doing an installation into a new software directory, yet, you want / need to retain / "inherit" any custom settings / adjustments and local (e.g., printer / plotter) definitions etc. from the existing software directory. The mere thought "I may lose my custom settings" may actually keep some users from upgrading their software. Others may try "beating the system", e.g., by

 - making backup copies of files such as "conpar", "devicenames", etc., and
   later "restoring" these files into the newly created software directory, or
 - removing the link "/vnmr" prior to installing the new software, such that
   the installation script ("load.nmr" and related utilities) can't "mess with
   the precious legacy files & settings", or
 - "in order not to loose anything that might be customized" (and many users
   don't keep records of what they change inside "/vnmr"!), they may try
   forcing the installation of the new software into the existing software
   directory.

ALL OF THESE ARE VERY BAD IDEAS! The last item in the list above used to be
the default, i.e., unless a user forced an installation into a new directory,
VNMR (and early versions of VnmrJ) would be installed into a standard path
("/home/vnmr" or "/export/home/vnmr") - but we have learned our lesson:
 - there can be complications from the mere presence of specific legacy files
   in "/vnmr",
 - customized files may be OVERWRITTEN and LOST, and
 - under certain circumstances there may be permission issues when the install
   software tries to unpack software files.

For all these reasons, VnmrJ is now installing into a version-specific path,
hence avoiding collisions when upgrading from an earlier version. Still, you
may want to watch out for a new path in the case of a software REINSTALLATION.

  Besides installing into a new, version-specific directory, our installation
software will also make sure that files and directories such as "conpar",
"devicenames", "devicetable", "shims", "probes" and "gshimdir" are properly
carried forward to the new software directory, so there is no need for you to
take any action on those - in addition, the previous software directory is
preserved, so in the worst case you can still copy over important customized
files where this is not done already (note that in older versions of VnmrJ,
the list of files that were carried over was shorter, possibly incomplete).
We recommend keeping at least one generation of the old software directory
around: disk space is cheap, so the impact of the extra disk space consumption
is minimal.

  So, you definitely don't need to care about carrying forward the above files
and directories. In some cases, such as "devicenames" and "devicetable", doing
so anyway may be harmless - in other cases it is definitely NOT:
 - in the case of "/vnmr/probes/probe.tmplt", a new software version may add
   new components to that file - therefore, this file is removed from the
   "carry-over components", and a fresh - possibly version-specific - file is
   installed;
 - while in the case of "probes/probe.tmplt" just affects spectrometer hosts,
   possibly only operations related to probe calibration, the contents of
   "/vnmr/conpar" are much more critical - a new version of VnmrJ may fail to
   launch if new parameters are missing from that file, e.g.: with an old
   version of "/vnmr/conpar", VnmrJ 3.2 will get stuck in the launch phase due
   to missing "systemglobal" parameters such as "probeiden", "vtflowrange", or
   "TSPQmagnetSN1".

Again: you do NOT need to preserve or manually carry over the "conpar" file
from your earlier VnmrJ software installation: the install software will copy
the old "conpar" into "/vnmr/conpar.prev" and then extract the relevant bits
from that file. This cannot work if the link "/vnmr" is deleted prior to
installation, and it is defeated if you manually copy the old "conpar" onto
the new file.

[Agilent MR News 2012-05-30]

2012-08-03:

VnmrJ PATCH "FAMILIES":

  As a follow-up to the discussion about VnmrJ patches, we are moving to a
Scheme that we hope will provide software corrections to problems that require
immediate action, but will also provide updates in a predictable time frame.
Patches for VnmrJ come in three flavors: "100", "300", and "500" series
patches. In the "VnmrJ 3.2, current patch versions" section of the revised
header section above, we provide a snapshot of the patches that are applicable
to the current VnmrJ release:
 - The "100" series patches are the main patch - there is always only one main
   patch per configuration. The "100" series patches ("101", "102", etc.) are
   cumulative, i.e., each subsequent patch in the series contains all of the
   contents from any preceding patches in the same series. So one can install
   a "103" patch, for example, without first installing the "101" and "102"

patches. We plan to release a new 100 series patch about every four months. A new 100 series patch will obsolete any earlier "100" series patch and "300" series patches.

- The "300" series patches are "hot-fixes" to solve an urgent problem. They typically are "single purpose" patches, and they are NOT cumulative. The "300" series patches will be included in a subsequent "100" series patch. IMPORTANT: care must be taken NOT to load an older "100" series patch after installing a "300" series patch.
- The "500" series patches are also single purpose patches, often to support new PC or OS versions. They are not cumulative. These patches are made when only a subset of users might be interested. For example, if there is a problem with Japanese fonts, we might make a "500" series patch. The "500" series patches will generally be included in the next VnmrJ release. They may or may not make it into a "100" series patch (if they do, we will mark them as "obsolete" in the header section above).

We are also moving to a new patch naming scheme - see the note below.
[Agilent MR News 2012-08-03]

A NEW VnmrJ PATCH NAMING SCHEME:

As mentioned in the article above, we are moving to a new patch naming scheme. This new scheme has already been implemented in the VnmrJ 3.2 release for UNITY INOVA and MERCURYplus / MERCURY-Vx systems. The patch name is used to encode the applicability of a patch to a given VnmrJ installation: it is encoded with the VnmrJ version, OS, console type, and patch number. These attributes are separated by underscores in the patch name. For example, "3.2_LNX_mmi_101.ptc" and "3.2A_LNX_ddr_102.ptc" are potential patch names. The new names are case-insensitive. The new patches are actually "zip" files, but as files with a ".zip" suffix are often blocked by email systems, we chose to use a ".ptc" suffix instead. The "base name" used in the new patch scheme consists of the following segments:

=> The VnmrJ software versions are of the form "VERSION x.y REVISION z"; that information is found in the first line of the file "/vnmr/vnmrrev". The first field of the patch name can match the version ("x.y", e.g., "3.2") OR the version and revision parts ("x.yz", e.g., "3.2A") of the VnmrJ software release. The special key "ANY" will match any VnmrJ version. In the examples above, a patch "3.2_LNX_mmi_101.ptc" can be installed on VnmrJ 3.2 or VnmrJ 3.2A systems, while a patch "3.2A_LNX_ddr_102.ptc" in theory is specific to revision A of VnmrJ 3.2 (VnmrJ 3.2A); note that starting with VnmrJ 3.0 we have effectively stopped using / altering the "revision" part, see Agilent MR News 2010-03-07 for more information.

=> The second field of the patch name indicates the computer operating system. Supported OS values are "LNX", "MAC", and "WIN". The special key "ANY" will match any OS. In the examples above, the patches can be installed on Linux systems only.

=> The third field of the patch name indicates the applicable spectrometer console type. Supported values are
  - "VNMRS" (DD2 and DirectDrive architectures),
  - "MR400" (MR-400 and MR-400 DD2),
  - "Inova" (UNITY INOVA), and
  - "MERCURY" (MERCURYplus, MERCURY-Vx)

This third field can also be set to keywords that represent groups of spectrometer consoles:
  - "MMI": applies to MERCURYplus, MERCURY-Vx, and UNITY INOVA;
  - "DDR": applies to DD2, VNMRS (a.k.a. DirectDrive), MR-400, MR-400 DD2.
  - "ANY" will match any console.

The patch installation software ("patchinstall") extracts the console value from the third line of the file "/vnmr/vnmrrev".

=> The fourth and final field is the patch version. This corresponds to the "100", "300", and "500" series patches described above. It is possible that certain numbers within a series are skipped. For example, in the current set of 300 series patches, there are "302" and "304" patches, but no "301" or "303" patches. This may occur for various reasons - e.g., we might send a "301" patch to a specific user to test to see if it fixes a problem. If it does not fix the problem, we might make a "302" version. The "301" version would then never be released.

This new naming scheme goes along with a new version of the "patchinstall" script (already included with VnmrJ 3.2 for UNITY INOVA and MERCURYplus / MERCURY-Vx systems, as mentioned above). That new script can also handle the old naming scheme, but the new patch naming is of course not usable with

earlier versions of "patchinstall", i.e., it is currently NOT applicable to older VnmrJ releases such as VnmrJ 3.2 for DD2, DirectDrive, 400-MR DD2, and 400-MR, or VnmrJ 3.2 for MacOS and Windows.
  Overall, the new patch naming scheme has a number of advantages over the previous one, such as
 - it offers more clarity (we no longer specify patch content types such as "psg", "bin", "acq", etc.);
 - it is more robust (case-insensitive);
 - patches sent by e-mail are less likely to be rejected (".ptc" extension);
 - covers combinations of target architectures
 - adopts the current VnmrJ naming convention (revision part not required)

On top of this, the new naming scheme is just a by-product of a "patchinstall" that has been thoroughly reworked and now enables uninstalling patches, i.e., if for any reason a patch causes a new issue, one can safely "roll back" the VnmrJ software installation to the pre-patch state (by using the associated, new version of "patchuninstall") - even through multiple, successive patch installs. In addition, the patches now feature a built-in file integrity check, i.e., after downloading one of the new patches, there is no need to evaluate the MD5 hash key in order to check whether the patch file is intact: the patch installation will do this for you.
[Agilent MR News 2012-08-03]

2012-10-19:

POTENTIAL VnmrJ 3.2 PATCH INSTALLATION CONFLICTS:

  As mentioned in the previous note, the new "patchinstall" utility that is included with VnmrJ 3.2 for UNITY INOVA and MERCURYplus/-Vx (see Agilent MR News 2012-08-06) currently retains the ability to install "old style" VnmrJ patches such as "3.2AbinLNXall503". This "feature" was intentional (as a "fall-back" option in the case of issues with the new patching scheme) - however, this accidentally leads to a potentially serious issue with patch "3.2ApsgLNXall102", designed and intended for DirectDrive & DD2 architectures running VnmrJ 3.2: that patch was issued BEFORE we released VnmrJ 3.2 for UNITY INOVA and MERCURYplus/-Vx, and it covered all architectures supported at that time (VNMRS / VMRIS, 400-MR, DD2, 400-MR DD2). However, with the added support for legacy systems, the "all" in the name of that patch suddenly covered a wider and UNINTENDED scope, in that "patchinstall" will now also install this patch on the legacy architectures - and as this patch includes PSG components, it will cause havoc on these systems. We will close this loophole in an upcoming VnmrJ 3.2 patch for UNITY INOVA and MERCURYplus / MERCURY-Vx: this new patch will modify "patchinstall" such that "old style" patches will no longer install.
  The key message from this: make sure you ONLY install patches that are designed for the architecture of your system! The Readme file(s) are clear on this point - so, make sure you DO download and READ the Readme file! Do NOT bypass the Readme information and JUST try "decoding" the patch name in order to determine whether a patch is applicable or not!
[Agilent MR News 2012-10-19]

INSTALLING "NEW STYLE" PATCHES:

  In Agilent MR News 2012-08-03 we presented a new patch naming scheme that is now in place and active as part of VnmrJ 3.2 for UNITY INOVA, MERCURYplus and MERCURY-Vx. The current patches for these systems are
 - patch "3.2_LNX_inova_101" for UNITY INOVA
 - patch "3.2_LNX_mercury_101" for MERCURYplus and MERCURY-Vx

see again Agilent MR News 2012-08-03 for details. As the new patch (naming) scheme involves internal patch integrity checking, we have removed the section on checksum / MD5 hash key validation in the top part of the Readme files for these patches. Accidentally, this also eliminated the generic installation instructions. We therefore specify those here - though we STRONGLY recommend still to study the Readme file before proceeding with the installation:
 - Download the patch file and its Readme file from our FTP site at
       ftp://ftp.agilent.com/mr_patches/
   e.g., the files "3.2_LNX_inova_101.ptc" and "3.2_LNX_inova_101.Readme"
 - Check the Readme file for any actions to be taken PRIOR TO or AFTER the actual patch installation (such actions may include exiting from all active VnmrJ processes, and/or stopping and restarting the acquisition processes, etc.)

```
 - All regular VnmrJ patches MUST be installed as the VnmrJ administrator
   (typically vnmr1).
 - The patch installation itself is done with
        patchinstall patch_name.ptc
   from a Linux shell window. Current examples:
        patchinstall 3.2_LNX_inova_101.ptc
   and
        patchinstall 3.2_LNX_mercury_101.ptc
   The patch file can also be specified with an absolute path, e.g.
        patchinstall /home/vnmr1/3.2_LNX_mercury_101.ptc
But again: you MUST study the Readme file prior to starting the actual patch
installation! The above shell command has been added to the Readme files for
the current "new style" patches.
                                                [ Agilent MR News 2012-10-19 ]

2013-02-19:

INSTALLING VnmrJ 3.2 FOR MS WINDOWS - A REMINDER:

  We continue to receive reports stating that "VnmrJ 3.2 does not work (or
cannot be installed) under MS Windows"; in the majority (maybe all?) of these
cases, this is caused by attempts to bypass our recommendations / instructions
for the installation. The key points here are
 - VnmrJ 3.2 for Windows REQUIRES either Windows XP Professional (32-bit),
   Windows 7 Enterprise, or Windows 7 Ultimate (64-bit); this restriction is
   caused by VnmrJ requiring components (Services for UNIX) that are available
   with the above Windows versions ONLY and CANNOT be retrofitted to simpler
   (lower cost) versions of the OS. See also Agilent MR News 2012-01-20 for
   more and related information.
 - Our documentation states that Anti-Virus software must be DISABLED for the
   VnmrJ 3.2 installation to work.
 - The Windows user account must have adminstrative privileges.
 - The Windows account name must NOT have spaces.
 - We strongly recommend to use an account name with up to 8 lower-case,
   alphanumeric characters ONLY (numeric characters are OK, but have the name
   start with an alphabetic character).
Some users have reported that VnmrJ 3.2 for Windows is very slow: we of course
assume that your PC hardware complies with the Microsoft specifications on the
minimum configuration (in terms of the RAM and disk size, CPU type and speed)
for a given OS version.
                                                [ Agilent MR News 2013-02-19 ]

2013-03-27:

DOWNLOADING VnmrJ PATCHES:

  In Agilent MR News 2012-08-03 we presented a new patch naming scheme, which
currently is applicable to UNITY INOVA, MERCURYplus and MERCURY-Vx systems
running VnmrJ 3.2 (VnmrJ 3.2 MI). These new patches are no longer "tar.Z"
("tar" files compressed with "compress"), but they are compressed archive
files created using "zip". In order to disguise the file type, we are NOT
using the standard ".zip" extension, but instead an "artificial" extension
".ptc". This way, patch files are less likely to be rejected by e-mail
gateways, where IT policies may prohibit incoming e-mail messages with "zip"
archives. The editor can't comment on whether (and to what degree) that is
successful, but some of his colleagues recently ran into a little snag with
".ptc" patch downloads:
  Some Web browsers and the associated software / operating environment (most
prominently MS Internet Explorer) will try determining the type of files that
the user is downloading, and they then try to be "helpful" by unpacking the
file right away (this may include operations such as uncompressing data,
mounting disk images, extracting "tar" and "zip" archives, and the like). For
many applications (downloads of software packages, components or upgrades,
etc.) this may be quite useful and convenient - but in the case of VnmrJ
patches any such unpacking is undesirable, for several reasons
 - the patch may be downloaded under MS Windows or under MacOS X, but is to
   be installed on a Linux PC, and unpacking of the patch archive may make it
   harder, if not sometimes impossible to transfer such data to the target OS
   architecture;
 - once unpacked, the data may be altered by the downloading OS (e.g., when a
```

   user is trying to inspect the file, some text content or the end-of-line
   convention used in that file could be altered);
 - most importantly, the VnmrJ "patchinstall" utility expects an UNALTERED,
   PACKED ARCHIVE - it will actually refuse to install an unpacked patch.
That's not the whole story, though: different from what people might believe, software doesn't just check a file's name extension to determine the file or content type. Traditionally, MS Windows (and its predecessor, MS DOS) used a 3-character name extension (".txt", ".doc", etc.) to assign a data file to an application. Other operating environments (such as MacOS X) would rather look at the beginning of the content (trying to find some "magic numbers") to try figuring out which application to use for opening a file. Early UNIX and Linux systems did not automatically assign data to applications, but the user did so via the command line. In this case, file name extensions were mostly just used to inform the user about the type of a data file. More recently, though, Linux has become more GUI-driven, and the data / application linking (and with this, the use of extensions) has become more important. Also, many applications (e.g., "zip", "gzip", etc.) automatically add or remove file name extensions, as appropriate.

In this context, some of the editor's colleagues experienced an interesting hiccup with VnmrJ 3.2 MI patch downloads: under MS Windows / MS Internet Explorer (IE), they tried downloading a patch such as "3.2_LNX_inova_102.ptc", and the download utility (IE and related software) checked the content of that file (with "unknown" extension), figuring it is a "zip" file - and hence renamed the downloaded file to "3.2_LNX_inova_102.ptc.zip". This led to two problems:
 - "patchinstall" did not recognize this file as a valid VnmrJ 3.2 patch file
   and therefore refused to install;
 - confronted with the associated error message, the user then tried unpacking
   the file using "unzip" - which DID work, but then led to an extracted patch
   directory, which again "patchinstall" would refuse to install (and trying
   to do a "manual" installation of course further aggravates the situation).
Conclusions from this:
 - DON'T try unpacking a patch - this invariably causes problems
 - if clicking on an FTP download link automatically extracts the archive,
   try downloading using "Save linked file as..." or equivalent from the link
   menu (usually press & drag the right mouse button on the link)
 - If the file is downloaded intact, but renamed by adding a ".zip" extension,
   simply RENAME the file, e.g.:

```
        mv 3.2_LNX_inova_102.ptc.zip 3.2_LNX_inova_102.ptc
```

 - for "old style" patches, make sure the name capitalization is correct; if
   necessary, correct the name by renaming, e.g.:

```
        mv 3.2apsglnxall103.tar.z 3.2ApsgLNXall103.tar.Z
```

   The new patch names such as "3.2_LNX_inova_102.ptc" are NOT case-sensitive,
   EXCEPT for the extension, which MUST be ".ptc", all lower case characters.

[Agilent MR News 2013-03-27]

VnmrJ PATCH INSTALLATION - AN ADDENDUM / REMINDER:

In Agilent MR News 2012-12-20 we gave downloading instructions for the current set of VnmrJ patches; right now, these are still valid, but a fundamental change in the way patches are accessed and downloaded is imminent (we will keep our FTP site active for the time being, though).

That note also covered the "503" patch ("3.2AbinLNXall503") for enabling the "SE" (Secure Environments) component in VnmrJ 3.2 for DD2, 400-MR DD2, Varian NMR & MRI Systems, and 400-MR (see Agilent MR News 2009-03-24 for information on the "SE" feature). This patch was previously announced in Agilent MR News 2012-10-19 - it is a "single purpose" patch, with the SOLE function of enabling the "SE" component in VnmrJ 3.2, and it can be installed independent of the presence of any other patches for that software version.

Unfortunately, the initial patch description was not specific about the fact that the patch not only opens the possibility to enable the "SE" feature - it ACTUALLY DOES ENABLE IT. This isn't necessarily what you may want: some users and installers felt "The SE feature may not be used now, but I better install it anyway, so I won't have to remember to install the patch later!" - and then found themselves in having to go through all the (substantial) extra steps and motions (admin, setup, etc.) for activating the "SE" component.

Conclusion: ONLY install the patch "3.2AbinLNXall503" when you REALLY want to use VnmrJ 3.2 in a Secure Environment (e.g., in connection with government submissions according to 21CFRpart11). Thanks to Ghirmai Meresi (Agilent) for pointing this out! In the meantime, we have also amended the patch description to reflect these implications.

[Agilent MR News 2013-03-27]

2013-07-11:

WHICH DIRECTORIES ARE COVERED IN THE LOCATOR DATABASE?

Recently, a user was interested in knowing how one could PREVENT data from being added to the Locator database. The editor checked with R&D (many thanks to Glenn Sullivan for providing the information below!) and learned the following:

First, one should keep in mind that the primary design focus for the Locator database (PGSQL) was/is in making sure ALL relevant data are captured in the database - therefore, there is no built-in mechanism for explicitly EXCLUDING data or directories from being added to the database - though one can still have data that are NOT included in the database - in order to learn how this can be done, we should first discuss how the Locator database gathers data.

The "engine" behind the Locator is a standard database (PGSQL) that is running as a Linux (or UNIX) daemon that is launched as part of the OS boot-up process. PGSQL is the short form of Postgres (the successor of the Ingres project) or PostgreSQL (where "SQL" stands for "Structured Query Language", a standard language used to interact with relational databases); PGSQL is free and open source software - for more details on the history of PGSQL see the Wikipedia under "PostgreSQL". How do we set up, configure and manage the Locator database?

=> The "PGSQL engine" for the Locator is set up and initialized as part of the VnmrJ installation process - no explicit user action is normally required to get the database set up and running.

=> Users rarely ever need to interact with the database daemon directly - the one notable exception is that prior to switching to an alternate VnmrJ directory, one must first stop the daemon, using

```
/vnmr/bin/S99pgsql stop
```

and then re-start the daemon after switching, using

```
/vnmr/bin/S99pgsql start
```

For details see Agilent MR News 2004-03-07. Once it is running, the daemon is silently (as a low priority background process) "crawling" the list of directories that may contain VnmrJ data (see below), constantly keeping the database up-to-date. If you use Linux shell commands (such as "rm -r" or "mv", as opposed to - for example - dragging data to the trash in the Locator GUI) to (re)move data in the database, you should expect a time lag until the database reflects such changes, unless you force an update to the database, e.g., using

```
managedb update
```

=> The database engine relies upon a set of (text) configuration files that determine which directories are scanned ("crawled") for data:

- "/vnmr/adm/users/profiles/user/USER_NAME": this file includes a line starting with "datadir" is filled by the command "dbsetup" (editing that line is not a good idea, as the next time you run "dbsetup", it would be overwritten again). For users other than vnmr1, this line typically lists the directories

  ```
  /home/USER_NAME/vnmrsys/data
  /home/USER_NAME/vnmrsys/parlib
  /home/USER_NAME/vnmrsys/shims
  ```

  and for "vnmr1" the line in addition includes

  ```
  /vnmr/fidlib
  /vnmr/stdpar
  /vnmr/tests
  /vnmr/parlib
  /vnmr/imaging/tests
  /vnmr/shims
  ```

  Editing the script "/vnmr/bin/dbsetup" is not recommended either, as inexperienced users may inadvertently add errors (possibly rendering the Locator, and with it VnmrJ, unusable altogether) - plus, you might need to reimplement such changes after any VnmrJ upgrades or patching.
- "/vnmr/adm/users/profiles/data/USER_NAME": By default, this file just lists the user's "~/vnmrsys/data" (already included in the file above, so this is redundant). This file is created via the "Data Directories" panel in "vnmrj adm" - if in this GUI extra directories were added, these would be listed in this file for the respective user(s) (as far as we can tell, this definition mechanism has rarely ever been used).

Additional directories may be listed / defined locally, on a per-user basis:

- "~/vnmrsys/persistence/ImportedDirs": this file is created if a user calls "Import Files to Locator...", listing the selected directories.
- "~/vnmrsys/persistence/SavedDirList": this file is created (or entries added to it) when a user saves data through "File" -> "Save as...", but NOT when saving data by calling "svf" or "svf('file_name')" on the VnmrJ command line or calling "svf" in a macro.

Again: the Locator database was designed with the intent to ensure that no data are MISSING from the database!

=> Keep in mind that the directories listed above (or included in the above configuration files) are "crawled", i.e., searched recursively; in other words: data at any nesting depth WITHIN one of these directories will (eventually) be found and be listed in the Locator database.

CONCLUSIONS for users who do NOT want specific FIDs to be included in the Locator database:

=> DO NOT save such data within "~/vnmrsys/data" (or "/vnmr", if you are vnmr1).

=> Directories listed in "~/vnmrsys/persistence/ImportedDirs" should NOT be problematic in that context, as you only would call "Import Files to Locator..." when you explicitly WANT to include data in the database. If you called this menu option by mistake, you can either delete the file "ImportedDirs" (if you don't want any of the listed directories included), or you could remove the relevant entries from that file (which only exists once you call the above menu function for the first time). After removing or modifying "~/vnmrsys/persistence/ImportedDirs", update the database using the relevant option from "Tools" -> "Update Locator", or by calling "managedb update" in a Linux shell.

=> The "File" -> "Save as..." function, however, is tricky: it AUTOMATICALLY adds new directories (to be crawled) to the Locator database. If you do NOT want files to be listed to the Locator database, DON'T save them using the menu! In particular, do NOT use "File" -> "Save as..." to save data DIRECTLY in your home directory, as this automatically causes FIDs in your ENTIRE local directory tree to be included in the Locator database! In addition, if "~/vnmrsys/persistence/SavedDirList" exists and contains unwanted directories (i.e., directories that should NOT be covered by the Locator database), you can again either modify or remove that file, then update the database as shown above.

If you wanted to be sophisticated about this last point, you could (as the relevant user(s)) set up a "cron" (or "anacron") job that periodically (say, once per day, or once every week, etc.) removes or modifies the user's "$HOME/vnmrsys/persistence/SavedDirList" and then calls "managedb update".

[Agilent MR News 2013-07-11]

==