Quantitative Analysis in VNMR

updated: 2008 Feb 27 (cg fry)

Some aspects of performing quantitative analysis within VNMR can be maddeningly complex. It made sense to some crazy Varian programmer once upon a time, but is *very* non-intuitive. And Varian's documentation is far from enlightening. [If you know of good documentation on this subject, please email me as to it's location.] **The following should give you a start on the "odder" aspects of performing quantitative analysis in VNMR.**

Sections I-III discuss quantitative work with one-dimensional data. Section IV describes some basics for performing volume integrals with VNMR.

I. Peak Intensities

A. Often setting the mode to absolute intensity, command **ai** (can interrogate using **aig?**), and setting **vs** to a set value is sufficient.

B. But since Varian's documentation is less than clear, here's more detail:

1. **nl** will provide the **vs**-scaled height of a peak. It appears that **getll** is identical to **nl**.
2. **dll** will provide a listing of all peaks in the display region larger than **th**. **dll** provides the **vs**-scaled peak heights. **dll** does not produce a file of **llamp** and **llfrq** (as might be expected from reading the description of **dll**) but rather sets values to these *parameters*.
3. **llamp** is an array containing the *absolute peak intensities* (independent of **vs**; *absolute* here does not mean "always positive", but rather in the Varian sense of being "not normalized", analogous to **ai** versus **nm**). These are the values you likely want.

   llamp is one of those crazy Varian parameters that not only contains values, but can also be set to *on* or *off*, using **llamp='y'** or **llamp='n'** respectively. Interrogate the second value in the array using **llamp='y'** (which is not completely necessary; the value in parens is the correct value when **llamp='n'**) and **llamp[2]?** **da** will not work. **llfrq** is similarly odd, providing the frequency from the right edge of the spectrum, not the frequency from the 0ppm reference, as is displayed by **dll**.
4. **fp** will produce a listing of the frequencies and peak height through an array, placing the output into a file **fp.out** in the exp (i.e., in ~/vnmrsys/exp# of the exp you are currently in). Supposing you're in exp2, going to a Unix terminal and typing the following will give the listing:

   ```
   cd ~/vnmrsys/exp2↵
   more fp.out↵
   ```

   It is important to understand the limitations on **fp**, however: it assume the peak is at an identical position through the array. If the peak moves due to some change in the sample, **fp** will produce an incorrect listing.

5. The height value (let's say of the 3rd peak) given by **dll** or **nl** is:

$$\boxed{\text{height}[3]_{\text{dll}} = \text{llamp}[3]*\text{vs} / \text{ct}} \tag{1}$$

   where **llamp[3]** is the absolute value height, and **ct** is the number of transients acquired.

6. If you are getting zero 0 values, check **th**. To get **th** to go negative, type:

   setlimit('th',1e6,-1e6,.1)↵

7. Note that baseline corrections will affect the height, and that there is no way to automatically apply the correction through an arrayed set of data. So use **dc** or **bc(5)** for each array set as you work through. Using **dll** now is seen to not be a great idea for best quantitation. Use a macro to loop through the array, performing a proper baseline correction on each spectrum; see section III below.

II. Integrals

Integrals work similarly to peak heights, but are more important for careful quantitative analysis; peak heights are typically useful only for semi-quantitative work. [Peak heights are sensitive to the Fourier number **fn**, changes in magnet homogeneity, and other effects that might change the linewidth of the peaks. Integrals are much less sensitive to these effects.]

1. A macro for listing integrations is **dli**. This macro lists all integral regions in the display region, but note that the **region #** column uses all the integral regions in the spectrum.
2. It is important for quantitative use of integrations in VNMR that **insref** be set to a value, e.g., **insref='y' insref=1e-6**. Setting **insref=0** will remove it from eq 2 below, but the value suggested provides a good scaling for most integrations, leading to **liamp** values that are simple to read. But for all spectra that are compared, **insref** and **ins** must be set to the same value (or corrected using eq 2).
3. As long as **intmod='partial'**, the **dli** output should be pretty obvious. The integral that it produces is far from obvious, however:

$$\boxed{\text{Integral}[i]_{\text{dli}} = \frac{\text{liamp}[2i]\times\text{ins}}{\text{insref}\times\text{fn}\times\text{ct}}} \tag{2}$$

   Yes, I agree…. Now you *know* why I wrote this section!

   a) The *i*th integral from **dli**, **Integral[i]$_{\textbf{dli}}$** , is computed as stated above.

   b) **liamp** is another of these crazy parameters that is arrayed (but cannot be displayed using **da**), and can be turned *on* (**liamp='y'**) and *off* (**liamp='n'**), similar to **llamp** above for peak heights. The *j*th element of **liamp** can be interrogated using **liamp[j]?** This value is the *absolute integral value,* and is the value you want.

   c) BUT! **liamp** fills with values for every region, including those showing as integrals and also those in-between! So the total number of **liamp** values is twice the number of integrals. bizarre. Restating in detail, even those sections between integrals (when **intmod='partial'**) are given integral values in **liamp**. So the 2nd integral region from the **dli** command is the 4th value → **liamp[4]** , thus the *ith* integral is the *2ith* liamp value.

d) **ins** is the integral scaling factor. Apparently this parameter has some limitations, so Varian introduced **insref** which appears to have a very similar function as **ins**, except that **insref** goes into the denominator in eq 2.

e) **fn** is the *fourier number*, the number of points the fid is transformed into. The integral value is scaled by this, as integrals should represent an *area* = height × width; the width is directly proportional to **fn**. The actual integral is computed however by summing the heights of each digital point in the region; dividing by **fn** provides the proper correction to integral *area*.

f) OK. So we can get the absolute value integral either from **liamp**, or by performing a calculation from rearrangement of eq (2) above:

$$\boxed{\text{ai-integral}[j] = \text{Integral}[j]_{\text{dli}} \times \frac{\text{insref} \times \text{fn}}{\text{ins}} \times \text{ct}} \qquad (3)$$

If **ins**, **insref**, **ct** and **fn** are all kept constant, we can simply use the integrals listed by **dli** . Right? Well, no, not really. See next section.

III. Baseline Corrections

It is critical for good quantitative work that baselines be accurately flattened prior to using integrals (the most typical use), or peak heights (not recommended, unless you *know* that the linewidths will stay constant), or for peak deconvolution (required anytime there is significant peak overlap that has to be separated).

dc – Performs a simple 0 and 1st order (vertical offset and slope, respectively) baseline correction, using just the display region; can be canceled with **cdc**. The edges of the display region are used (so make sure these are regions of good baseline) for determining the straight line, which is subtracted from the data.

bc – Without options, performs a spline fit; ***not recommended*** for quantitative work.

bc(5) Applies a 5th order polynomial fit to the data. **bc(7)** similarly applies a 7th order polynomial fit; higher order fits than 7th are not recommended. All regions (even those outside the display region) not under an integral are used as baseline in the fitting procedure, so setting proper integration regions prior to applying a **bc(5)** is mandatory.

Proper application of one of the functions described above will result in a good baseline. Integrals regions can now be reset to give quantitative data.

The userlib macro **mz** can be used to copy integral regions from one experiment to another (analogous to **mf** and **md**).

Unfortunately, the baseline correction is not applied through an arrayed data set (it only corrects the currently display spectrum); kinetics data are usually acquired as arrays. In such cases, a macro is often needed to reduce the time involved in applying integrations across a large data set. The last page of this section shows an example macro with a variety of functions (see /gaim/vnmr/maclib/integarray.UW):

IV. Volume Integrals in VNMR

Volume integrals rely on the same principles as integrations in one dimension. Good phasing and flat baselines are crucial to getting the best values. VNMR provides reasonable tools, but a number of other software packages exist that may be better. NMRFAM provides good baseline flattening routines—at the least for Felix, if not for NMRpipe. Only a cursory introduction to the use of VNMR's **ll2d** command is provided here.

Similar to the discussion above for 1d integrals, the 2d volumes are computed in VNMR as follows:

$$\text{ai-volume}[i]_{\text{info}} = \text{Reported\_volume}[i] \times \frac{\text{ins2ref} \times \text{fn} \times \text{fn1}}{\text{ins2}} \times \text{ct} \qquad (4)$$

There does not appear to be an analog to *liamp*, so the volume has to be computed from the **ll2d('info',#)** or **ll2d('writetext','*filename*')** commands. The following have been found to be good values:

ins2 = 1e9

ins2ref = 1

It is recommended that you set integral regions on a high resolution acquisition of the 1st increment prior to acquiring the 2d data set. If this is done, you can apply baseline corrections using **bc('f2',7)** , as one example.

Performing **cfpmult** is important for removing dc offsets, especially in homonuclear 2d data.

Performing **calfa** can be crucial to removing baseline roll.

ll2d is the primary routine for performing volume integrals in VNMR. The following are the most useful commands and options for this routine:

ll2d('reset')	removes all regions
ll2d('volume')	automatically applies regions to all observed peaks in display region; decrease **vs2d** until you see only peaks (no noise)
ll2d('combine')	will combine peaks within a cursor box into one box; similarly, you can list a series of existing regions that will be combined, as, e.g., **ll2d('combine',1,3,6,7,8,9)**.
ll2d('mark')	this is the best method of applying new volume regions; set a box using the cursors about the peak(s) first
ll2d('unmark',#)	deletes the volume region #
ll2d('writetext','*filename*')	writes a text file with proper information
pconpos pll2d page	will give a simple plot with the boxed regions included
vnmrprint *filename*	typed inside a unix terminal window will printout the writetext file

```
"integarray - Katya Delak (Sahai group) and cgfry – written a long time ago"
"  In this particular example, the peaks are shifting and changing in linewidth."
"  The macro accommodates both issues by using nll to find the peak/line"
"    and dres to estimate the width of the peak."
"  The integral is set 1.5x the width of the peak, and is centered on the peak from nll."
"  Note that th must be set prior to running the macro; thadj did not work."
"printon"
$comparr=celem                      "celem is then number of arrayed fids/spectra"
$n=1
$first=pad[$n]

"set up an output file with list of integrals and times"
input ('Enter the filename for output data: '):$filename
input ('What is the time lapse before the first acquisition is initiated (in seconds)? '):$time
"comment out next line, and uncomment printon to test"
write('file',$filename,'%6s %10s %14s %5s', 'index',' time','integral','lifrq' )

repeat
   select($n)                           "selects nth spectrum from the array"
   dc
   "thadj"                              "adjusts threshhold based on S/N, etc."
   cz                                   "clears prev. integral regions"
   intmod='partial'                     "sets up partial integral mode"
   nll('pos'):$lines                    "identifies # of lines/peaks in spectrum to $lines"
   $line=1                              "initializes loop at first line/peak"
   $time=$time+(pad+nt*(at+d1))         "calculate time for this spectrum"
   repeat
      getll($line):$ht,$freq$line         "gets list of lines. defines ht, freq for indexed line"
      dres($freq$line):$width$line,$res   "identifies width & res. of indexed line"
      $width$line=1.5*$width$line
      z($freq$line-$width$line,$freq$line+$width$line)  "identifies integ. regions based"
                                        " on freq&linewidth"
      $line=$line+1                     "iterates to next line"
      until $line>$lines
      ins=100
      dlni
      display('lifrq')
      $i=1
      repeat
         $int$i= liamp[$i]*ins/insref/fn
         write('file',$filename,'%6.0f %10.1f %14.8f %1s%1.0f%1s',$n,$time,$int$i,'[',$i,']')
         $i=$i+1
      until $i>size('lifrq')
      $n=$n+1
   until $n>$comparr
printoff($filename)
```