Acquiring Kinetics Data in VNMR

updated: 2011 Dec 1

Improvements in ^{1}H Kinetics Experiments (new as of 22 Nov 2011)

Investigations carried out by Paul White and cgfry in fall 2011 demonstrate clearly the adverse effects too short a repetition delay (d1+at) can have on kinetics data:

a) Peak intensities and integrations are significantly affected, because of incomplete relaxation recovery. Measurement of T_1 values may enable some correction of the results.

b) In a standard experiment (e.g., s2pul) that does not include homospoil or gradient crushers, the peak intensities become increasingly dependent on chemical shift as the repetition delay is shortened. Somewhere between 1s ≤ d1+at ≤ 3s, the data become unusable for quantitative studies; reasonably accurate post-acquisition corrections to the data cannot be made.

The following set of recommendations should be added to experimental setup protocols for ^{1}H kinetics experiments; they minimize errors arising from small repetition delay times, and enable reasonable methods to be used to correct the data. They will always improve data quality, and should therefore *always* be used for quantitative work.

1. Use gradient crushing as enabled by **sspul='y'** in the new sequence **s2sspul** for all ^{1}H kinetics experiments. This change is effected on our Varian high-field spectrometers (but not yet on Hermes) by setting up the kinetics run as done previously (and described in the next section), and then typing into vnmr:

   s2sspul↵

   This macro changes the experiment (from **s2pul** to **s2sspul**) which adds gradient crushing before the **d1** period. All other parameters are left unchanged.

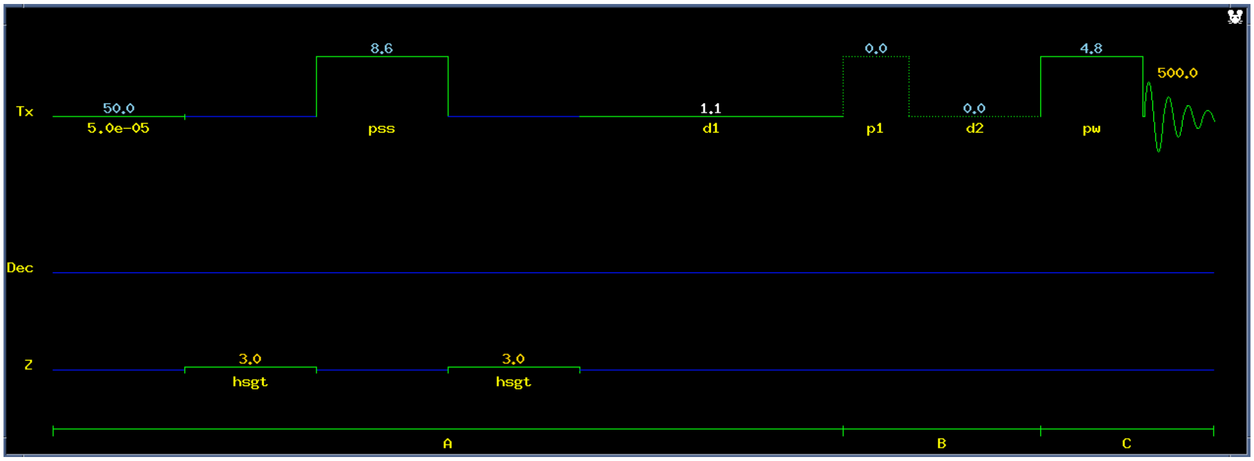

Figure 1. The s2sspul pulse sequence with G_Z–90°[pss]–G_Z gradient crusher sequence preceding the d1 period. The rest of the sequence is standard s2pul sequence.

2. The gradient crushing period removes any **at** (acquisition time) dependence from the repetition delay: with **s2sspul**, the longitudinal magnetization utilized in the next experiment

is recovered only during **d1**. Recommendations for repetition delays in **s2sspul** experiments thus equate to recommendations for setting the **d1** value.

3. When **d1 ≥ 3×T_1** , use a 90° pulse: **pw=pw90**.

   When **d1 < 3×T_1** , use a 30° pulse. Estimations of systematic errors and corrections to the results can be made with knowledge/measurement of the proton T_1 values. Note that T_1 values can change during a reaction, so measurements of T_1 both before and after the reaction may be required. cgfry can assist with how make such estimations and corrections.

Experimental Setup Protocols

Arrays in VNMR turn acquiring kinetics data into a fairly simple task (Section I). More complex situations can also be accommodated, involving relatively complex acquisition schemes with accurate time-stamps being generated with the spectra.

I. Typical (Simplest) Setup

A. Set up a normal ^1H (or ^{13}C, or ^{31}P, etc.) experiment, and be confident with your temperature calibrations. Acquire a spectrum with good sensitivity and good line shape.

B. Make sure **d1**+**at** ≥ 3×T_1; this is not a strict requirement, since it is the *difference* in T_1 values between the two (or more) interesting protons (or ^{13}C, etc.) that matters; if the T_1 values are identical, all values of **d1**+**at** will give quantitative results (albeit with different signal-to-noise). But some awareness of the T_1 values is important to insure that the data is quantitative. Using 3×T_1 may introduce up to 5% systematic errors; 5×T_1 reduces these errors to ≤ 2%. **at** need be only long enough to provide *sufficient* resolution.

C. To assist with fast kinetics, shim on a non-reacting sample having identical solution height—or keep the solution height ≥ 3× coil length; 3×16mm = 48mm for the 500 MHz broadband and hcx probes—and good quality tubes (at least Wilmad 507, 528's being better). These conditions will give the best shims when the reacting sample is introduced, with careful work allowing immediate acquisition of reasonable quality spectra.

D. For faster kinetics (half-life < 10 min), keep **nt** as small as possible; for slower kinetics, keep **nt** equal to a multiple of 8. The time per experiment, t_i, will (typically) equal:

$$t_i = \mathbf{nt} \times (\mathbf{d1} + \mathbf{at})$$

E. If **nt=1** is used for fast kinetics, beware of the center glitch, especially on the UNITY-500. Change **tof** (using **movetof**) so no interesting peaks are at the center of the spectrum.

F. Array **pad**, the *p*re-*a*cquisition *d*elay parameter, using something like:

   array('pad',50,30,0)

   which will give 50 values all with **pad**=30 secs. The timing in a proton experiment with

   nt=8, **d1**=0, **at**=2,

   and **pad** as set above will be as follows, where td_i = time at mid-point of each experiment:

   1st pad: 30 s

1st exp:	$t_1 = 8\times(0+2) = 16$ s	$td_1 = 30 + 16/2 = 38$ s
2nd pad:	30 s	
2nd exp:	$t_2 = 16$ s	$td_2 = 30 + 16 + 30 + 16/2 = 84$ s
3rd pad:	30 s	
3rd exp:	$t_3 = 16$ s	$td_3 = 3\times30 + 2\times16 + 16/2 = 130$ s
…		
ith exp:		$td_i = i \times \mathbf{pad} + (i - 1/2) \times \mathbf{nt} \times (\mathbf{d1}+\mathbf{at})$

→ The 1st pad value can be changed, as would usually be desired, to zero by entering:

```
pad[1]=0
```

G. For the example above, ***time-stamps*** for each experiment can be written to a file during the acquisition: one simply needs to set **bs=4** (or **bs = ½ × nt**). The **bs** events cause a time-stamp to be written into the ***datafolder*/log** file at the mid-point of each experiment. For a kinetics run of this type, the time of each experiment is simple to calculate. But the log file will provide a check on the accuracy of the experiment timing.

Unfortunately, this trick will not work for **nt=1**.

Some care must be used with setting ***bs*** *to small values, especially when using the UNITY-500:* **bs** should never be set such that < 2 secs occurs between **bs** events or the end-of-experiments. For **nt=2** and **d1+at=0.5** secs, as one example, setting **bs=1** will freeze the SCSI connection on the UNITY, requiring a hard-reboot and loss of data.

II. Setting pad Manually

It can be useful to set **pad** differently from the simple command given above. One simple change, often desired, is to avoid the initial 30 s delay. To do this, set **pad** as follows:

```
pad=0,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,
        30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30
```

The full string must be entered as a single line. The easiest way to accomplish commands of this length is to write a macro containing the command. To do this, type in a terminal (Unix) window:

```
cd↵
cd  vnmrsys/maclib↵
textedit  filename↵
```

and enter the command in a single line (which will wrap on the screen; do not hit RETURN inside the line). Any valid Unix filename can be used for ***filename***, but use some care to not use a name already defined by vnmr. Do not, for example, use the names **array** or **svf** or **rtp**. After creating the macro, type it into the vnmr command line to set the array.

III. Setting pad and nt Manually

A. *Non-linear* **pad** *arrays:*

A more efficient dataset can be obtained by setting **pad** such that more spectra are acquired early on, and fewer later in the run. In the example above, no data is acquired after 40 mins into the reaction. To acquire data out to multiple hours would require a very large array if **pad** values are kept constant at 30s; many more spectra will be acquired than would really be required. One solution would be to lengthen the **pad** values later in the array as follows:

pad=0,1,1,1,1,1,2,2,2,3,3,4,5,6,7,8,10,12,14,16,18,20,23,26,29,32,35,40,45,50,55,60,65,70,80,90,100,120,140,180,220,300,400,600,800,1400,2000,3000,5000,10000

which will acquire data out to ~ 7 h. These 50 spectra may much better determine the kinetics than the previous example. The **pad** array above, even so, was made up ad-hoc, not being well-thought out. To best optimize the kinetics run, some modeling of the kinetics is recommended, to estimate the optimum set of pad values.

B. *Arraying* **nt** *diagonally with a non-linear* **pad** *array:*

In the example above, issues are easy to identify at both extremes of the **pad** array:

(i) For the early **pad** values, if **nt=8** is used, the time-resolution is 16 secs; reducing **nt**, with **nt=1**, will improve that to 2 secs (but, of course, only if the sensitivity allows).

(ii) For the later **pad** values, keeping **nt=8** (or smaller) may limit the data quality. Much larger **nt** could be performed with no negative effects in time-resolution, since the reaction is proceeding more slowly.

It therefore makes sense to increase both ***nt*** *and* ***pad*** *throughout the kinetics run.* An example macro that sets **nt** and **pad**, and makes them track with each other (i.e., diagonalizes them) is given in /vnmr/maclib/**padnt.UW**. In this example, the kinetics are modeled by $k = 0.003\ s^{-1}$ at the working temperature. **d1**+**at**=3s are estimated to be a sufficiently long repetition delay. 25 spectra will be acquired. If the sensitivity is allows **nt**=1, we can now accurately plot out the reaction time-coarse, and estimate the times at which equal amounts of reaction will occur. For 25 spectra, we want data after every 4% of reaction which occur at:

Example 1: $k = 0.003\ s^{-1}$ (**at**=2, **d1**=1) .

optimal times = 0,14,28,43,58,74,91,110,129,149,170,193,218,245,274,305,341,380,424,476,536,611,707,842,1073

If **d1+at=3**, and **nt=1** are OK, we can adjust the parameters to match as in **padnt.UW**:

pad=0,8,8,9,9,10,11,10,7,8,9,11,13,15,11,7,12,15,20,28,24,27,48,63,135
nt=1,2,2,2,2,2,2,4,4,4,4,4,4,4,8,8,8,8,8,8,16,16,16,32,32
array=’(pad,nt)’

where the computed mid-spectrum times for each experiment are identical to the *optimal times*, except for the first spectrum with a mid-point at 1.5s. The last statement in the macro is required, which forces **pad** and **nt** values to increment together; this statement *diagonalizes* the square matrix.

Example 2: *k* = 0.0003 s^{-1} (**at**=2, **d1**=3) .

optimal times **=** 0,136,277,426,581,743,914,1095,1285,1487,1702,1932,2179,2446, 2736,3054,3405,3798,4243,4757,5364,6108,7067,8419,10729

Here, the optimal times are a simple order-of-magnitude increase from Example 1, but the computed parameters are not. Data quality is improved by keeping **nt** a multiple of 8:

pad=0,56,62,68,75,83,91,100,71,42,55,70,87,107,130,157,192,152,125,194,288,264, 319,391,1031
nt=8,16,16,16,16,16,16,16,32,32,32,32,32,32,32,32,32,64,64,64,64,128,128,256,256
array='(pad,nt)'

The *optimal times*, **pad**, and **nt** arrays are computed using an Excel spreadsheet; these two examples are given in /vnmr/maclib/**padnt.xls.zip**.

C. Acquiring non-linear arrayed kinetics data sets with time-stamp data:

Once the padnt array is set, the data must still be acquired. Obtaining the data with accurate time-stamps is now very useful. To do this, first shim and set up the 1D parameters as discussed above (setting **pad** and **nt** by any method). Then enter:

kintime↵

at the vnmr command prompt.

i) **kintime** queries whether an ASCII file kintimes.txt from a previous run is backed up. kintimes.txt will be written to your current data location (**pwd** at the vnmr command line), and typically should be moved into the dataset folder after data acquisition is complete. For example, suppose I acquire a kinetics dataset and save it using **svf('kinexample')** . This will write the data to the folder

/b6hf/cic/fry/kinexample.fid

If I used **kintime** and **au** to acquire the dataset, a file /b6hf/cic/fry/kintimes.txt will have been written during the kinetics run. To properly save this data, it needs to be moved into the dataset folder using the following command in a Unix terminal window:

```
mv  kintimes.txt  kinexample.fid↵
```

ii) **kintime** sets the parameter: **wnt='kintimewr'**

Now you are ready to insert the reacting sample. Do so and enter:

au↵

to start the acquisition.

au is a version of **go** that executes the parameter string **wnt** at the end of each arrayed experiment. **kintimewr** writes four values for each experiment: (i) the experiment #, (ii) the date, (iii) the time-stamp for the i^{th} experiment, and (iv) the length of the i^{th} experiment.

The time-stamp is the accurate ***END-TIME*** of the *i*th experiment, not the middle. Taking this time and subtracting ½ the length of the *i*th experiment — $\mathbf{nt_i} \times \mathbf{(d1+at)/2}$ — will give the correct mid-point time for each experiment.