SUPPLEMENTARY INFORMATION

Less is more: A simple Methyl-TROSY based pulse scheme offers improved sensitivity in applications to high molecular weight complexes

Nicolas Bolik-Coulon, Alexander I.M. Sever, Robert W. Harkness, James Aramini, Yuki Toyama, D. Flemming Hansen, Lewis E Kay

Supporting Figures

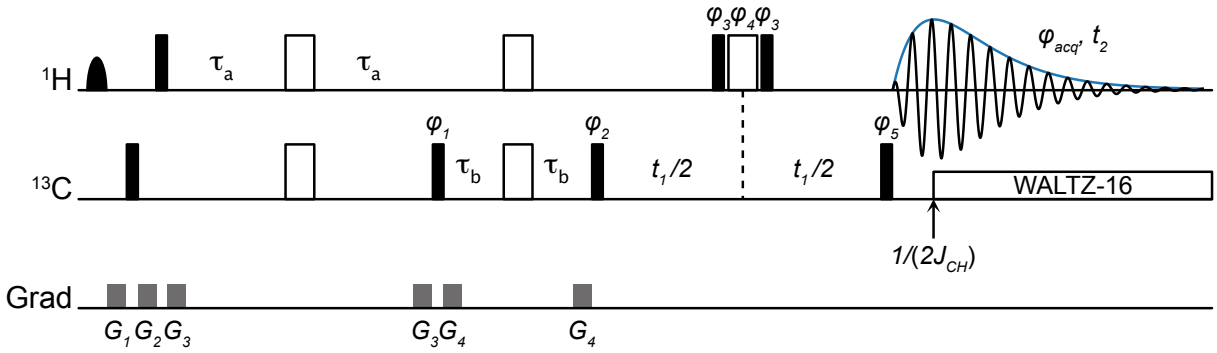


Figure S1. Pulse sequence used to measure purged ddHMQC datasets. Narrow black (wide white) rectangles refer to 90° (180°) pulses. The first proton pulse is a shaped pulse for water selective excitation. The value of τ_a is set to optimize sensitivity, typically less than $1/(4J_{CH})$ and τ_b = $1/(8J_{CH})$ = 1 ms. Pulses are aligned along the x-axis unless otherwise indicated. The phase cycle is: φ_1= x,-x; φ_2= 2(y),2(-y); φ_3= 2(x),2(y),2(-x),2(-y); φ_4= 2(y),2(-x),2(-y),2(x); φ_5= 8(x),8(-x); φ_{acq}= 2(x,2(-x),x),2(-x,2(x),-x). A minimum cycle of 4 is used. Gradient strengths (in % maximum) and durations are: G_1=(10%,1ms), G_2=(20%,1ms), G_3=(30%,0.5ms), G_4=(-15%,0.3ms). Further details are provided in the legend to Figure 1.

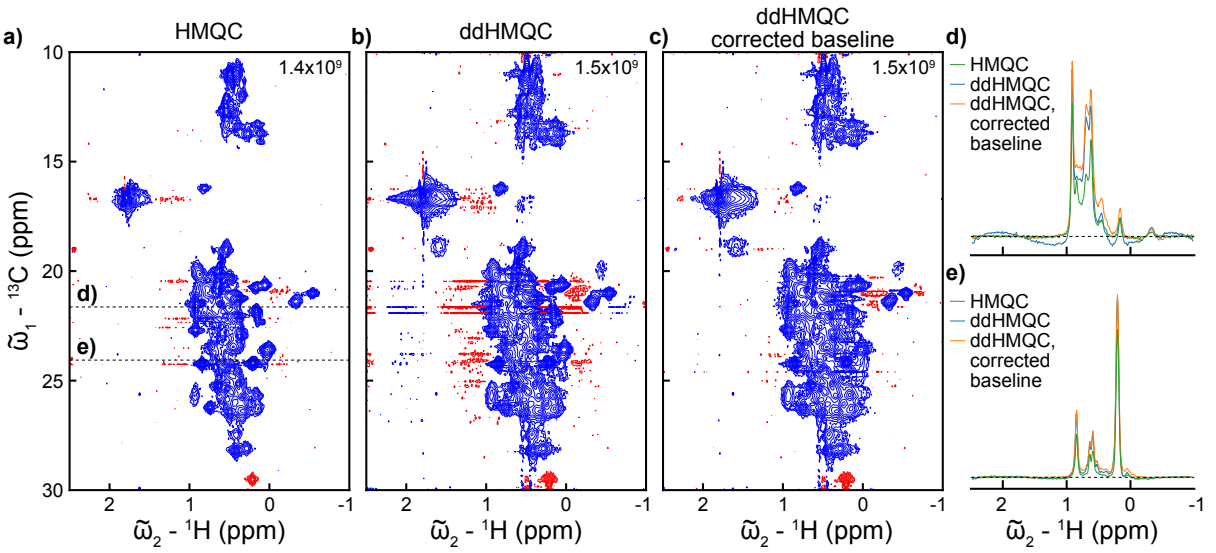


Figure S2. HMQC (a), ddHMQC (b) and ddHMQC with corrected baseline using in-house written software (c; see below) for $\alpha_7\alpha_7$, 1 GHz, 7°C. All three spectra were contoured at the same level, with the noise level indicated on the top right of each spectrum. Horizontal dashed lines in (a) indicate the position of cross-sections shown in (d) and (e). Section (d) is chosen

to illustrate an example of a "bad" baseline in the ddHMQC spectrum (worst of all ^{13}C traces). Spectra are contoured at sufficiently low levels so that some noise is observed. Notably, the quality of the baseline before correction was worse in datasets recorded at the lower temperature, where broad lines were produced (see below).

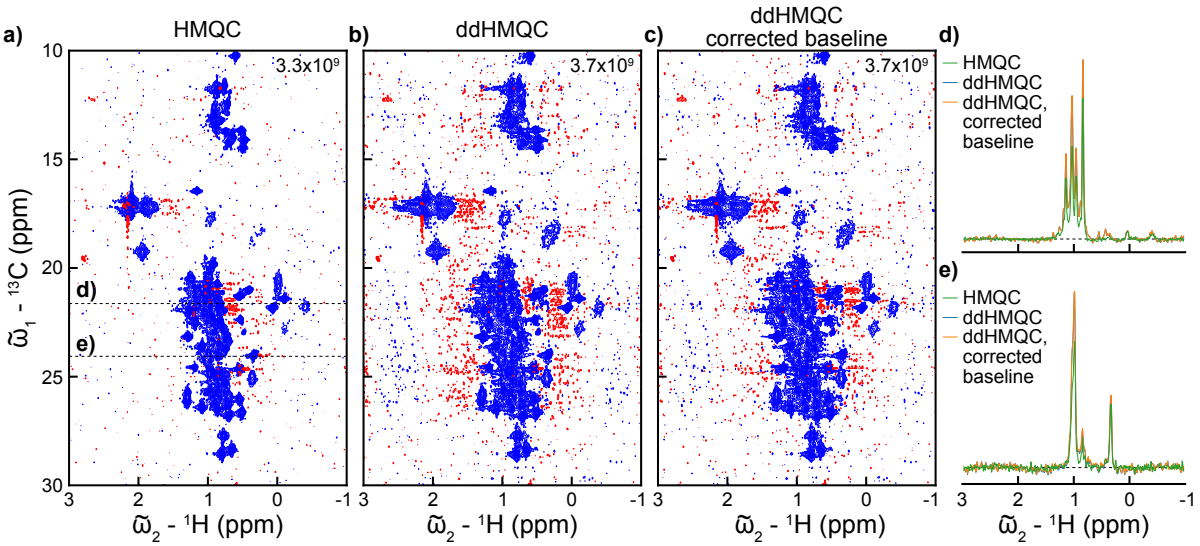


Figure S3. Similar regions as shown in Figure S2 but these spectra were recorded at 1 GHz, 40°C.

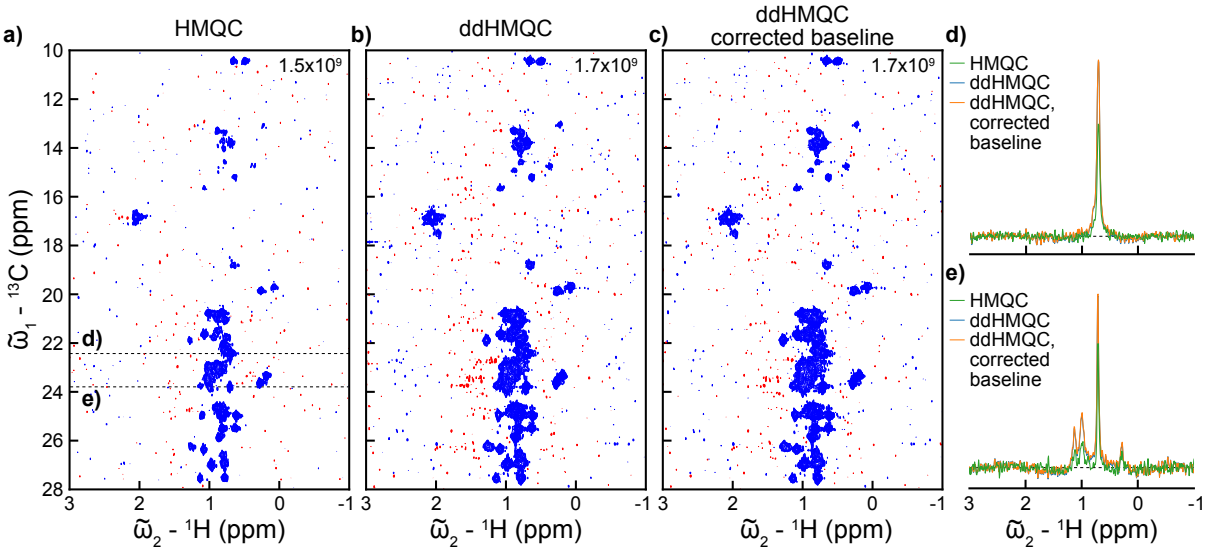


Figure S4. The same comparison of HMQC/ddHMQC experiments as in Figure S2 but for AaLS, 1 GHz, 40°C.

Data processing

Time-domain data are processed using a combination of *nmrPipe* [1] and an in-house written python script. Below we list the scripts that have been used along with the python programs for *J*-decovolution and baseline correction of ddHMQC spectra (available in electronic format from the authors upon request).

(1) The conversion of the Bruker data into *nmrPipe* format is achieved using the fid.com script:

```
#!/bin/csh

bruk2pipe -verb -in ./ser \
  -bad 0.0 -ext -aswap -DMX -decim 1240 -dspfvs 20 -grpdly 68  \
  -xN              2048  -yN                220  \
  -xT              1024  -yT                110  \
  -xMODE            DQD  -yMODE        Complex  \
  -xSW        16129.032  -ySW         5025.068  \
  -xOBS         999.201  -yOBS         251.253  \
  -xCAR           1.000  -yCAR          20.000  \
  -xLAB              1H  -yLAB             13C  \
  -ndim               2  -aq2D         Complex  \
| nmrPipe -fn MULT -c 3.90625e+00 \
  -out ./test.fid -ov
```

It is important to note the -DMX flag, ensuring that the time-domain signal starts with the first point of the FID. The python program Apodization.py (see below) used for *J*-deconvolution requires this format.

(2) The residual water signal is subtracted with the water.com script:

```
#
nmrPipe    -in test.fid                                  \
| nmrPipe  -fn PS -p0 0 -p1 -78721                      \
| nmrPipe  -fn POLY -time                                \
| nmrPipe  -fn PS -p0 0 -p1 78721                       \
   -out test_Apod.fid -verb 2 -ov
```

where the phase p1 is calculated to move the center of the spectrum from 1 ppm to on-resonance with the residual water signal, allowing proper subtraction of water. Note that $p1 = \frac{\Delta}{xSW} 360 X_{PTS}$, where Δ is the amount in Hz to shift the spectrum, xSW is the spectral width in the ^{1}H dimension, and X_{PTS} is the number of complex points in the ^{1}H time domain (after conversion; we recommend loading the converted data into *nmrDraw* to determine the value of X_{PTS}, which will not be equal to xT because of the -DMX flag in the fid.com script. The value of X_{PTS} is 954 in this case).

(3) The correction of peak distortions relies on dividing the time-domain signal by $\sin \pi J_{CH} t$ for $t>0$ where decoupling is not applied (*i.e.*, the first $1/(2J_{CH})$ of the FID). This is performed using the *nmrglue* [2] and *numpy* [3] Python library in the Apodization.py script:

```
import numpy as np
import nmrglue as ng

dic, data_orig = ng.pipe.read('test_Apod.fid')

SW = dic['FDF2SW']              #proton spectral width

npts = data_orig.shape[-1]      #number of points in the proton dimension

times = np.arange(npts)/SW      #times at which data-points are stored

data = data_orig.copy()

J = 125.0               #carbon-proton J-coupling constant in Hz

def Apod(t, J):         #Apodization function
        return np.sin(np.pi * J * t)

for c, t in enumerate(times):
        if c == 0:      #first point cannot be corrected
                pass
        else:
                if t < 1./(2.*J):    #only correct until decoupling is
applied
                        apod = Apod(t, J)
                        data[:, c] /= apod

ng.pipe.write('test_Apod2.fid', dic, data, overwrite=True)
```

Comments are shown in green.

(3) Finally, the corrected time-domain signal is processed using the After_apod.com script:

```
#
nmrPipe    -in test_Apod2.fid                                \
| nmrPipe  -fn ZF -size 954                                  \
| nmrPipe  -fn SP -off 0.35 -end 0.98 -pow 2.0 -c 1.0       \
| nmrPipe  -fn ZF -size 2048                                 \
| nmrPipe  -fn FT                                           \
| nmrPipe  -fn PS -p0 19.6 -p1 0.0 -di                      \
| nmrPipe  -fn EXT -x1 3.0ppm -xn -1ppm  -sw                \
| nmrPipe  -fn POLY -auto -ord 1                            \
| nmrPipe  -fn TP                                           \
| nmrPipe  -fn LP -fb -ord  16                              \
| nmrPipe  -fn SP -off 0.35 -end .98 -pow 2.0 -c 1.0        \
| nmrPipe  -fn ZF -auto                                     \
| nmrPipe  -fn FT                                           \
| nmrPipe  -fn PS -p0 90 -p1 180.0 -di                      \
```

```
| nmrPipe  -fn TP                                                  \
| nmrPipe  -fn POLY -auto -ord 4                                   \
   -out test_apod_finish.ft -verb 2 -ov
```

There are 954 complex t_2 points in the converted (-DMX) data and

```
nmrPipe  -fn ZF -size 954
```

simply ensures that this is the case.

The first point in the proton dimension is equal to 0 (it cannot be corrected using the Apodization.py script), which introduces a DC baseline offset. Thus, it is essential to apply a baseline correction before transposing and processing the carbon dimension to ensure that linear prediction functions properly.

(4) For some applications (typically $\alpha_7\alpha_7$ at 7°C) we improve the final baseline using the FitBaseline.py script:

```
import numpy as np
import nmrglue as ng
from scipy.optimize import curve_fit
import math

dic, data = ng.pipe.read('test_apod_finish.ft')

limMin, limMax = 40, 460      #noise level will be evaluated for points
between 0 and 39, and point 460 to the end; must ensure that there are no
peaks in these regions


NP_H = np.shape(data)[1]      #number of points in the proton dimension
NP_C = np.shape(data)[0]      #number of points in the carbon dimension

def Poly(x, coeff):           #polynomial function that fits the baseline
      s = 0.
      for n in range(len(coeff)):
            s += coeff[n] * x**(len(coeff) - n - 1)

      return s


def Truncate(data, ppm, noise):     #function to retain part of the signal
only within +-2xnoise, where noise is the noise level

      trunc = []
      trunc_ppm = []

      span = 30         #range over which standard deviation is evaluated

      for d in range(0, len(data), span):
```

```
            std = np.std(data[d:d+span])        #standard deviation of the
signal in the considered portion

            if abs(std) <= 2.*abs(noise):
                  trunc.append(data[d:d+span])
                  trunc_ppm.append(ppm[d:d+span])

      trunc = np.concatenate([t for t in trunc])
      trunc_ppm = np.concatenate([f for f in trunc_ppm])

      return trunc, trunc_ppm


for n in range(NP_C):
      Slice = data[n, :].copy()

      N = max(abs(Slice))
      Slice /= N                    #Normalization of the slice

      noise = np.std(np.concatenate([Slice[:limMin], Slice[limMax:]]))
      #noise level for the slice, computed using the above limits

      Slice_c, uc_H_ppm_trunc = Truncate(Slice, uc_H_ppm, noise)


      Coeff = np.polyfit(uc_H_ppm_trunc, Slice_c, 4)       #fit to a
polynomial of degree 4

      data[n, :] -= N * Poly(uc_H_ppm, Coeff)              #subtract the
result of the fit

ng.pipe.write('test_apod_custom.ft', dic, data, overwrite=True)
```

The initial part evaluates the noise level, σ, on a per-cross-section basis, by only considering the portions of the cross-section that have no signals. These portions are determined by the values of the constants *limMin* and *limMax*, chosen such that there are no signals for points lower than *limMin* and higher than *limMax* in each cross-section. Then, the parts of the cross-section with intensities within $\pm 2\sigma$ for a range of `span` points are considered to be baseline and are fitted to a polynomial of degree 4; note that the first `span` points are evaluated [0,`span-1`], followed by the next `span` points [`span`,`2span-1`] and so forth to establish the baseline. Finally, the resulting polynomial is subtracted from the original cross-section.

Comparison of linewidths of two Lorentzian functions of different heights

We write a Lorentzian function

$$\mathcal{L}_1(\omega) = \frac{R_2}{\Delta\omega^2 + R_2^2} \tag{S1}$$

where R_2 is the transverse relaxation rate of the magnetization and $\Delta\omega = \omega - \omega_o$ with ω_o the peak position (rad/s). Consider a second Lorentzian function with the same linewidth at half height but where the peak height is scaled by a factor *k*:

$$\mathcal{L}_2(\omega) = k\frac{R_2}{\Delta\omega^2 + R_2^2} \tag{S2}$$

We can calculate the linewidth of these Lorentzian functions, LW_1 and LW_2 (rad/s) at an arbitrary height, say βT_2, $\beta \leq 1$, where $T_2 = 1/R_2$ is the height of $\mathcal{L}_1(\omega)$ when $\omega = \omega_o$:

$$LW_1 = 2R_2\sqrt{\frac{1-\beta}{\beta}} \tag{S3}$$

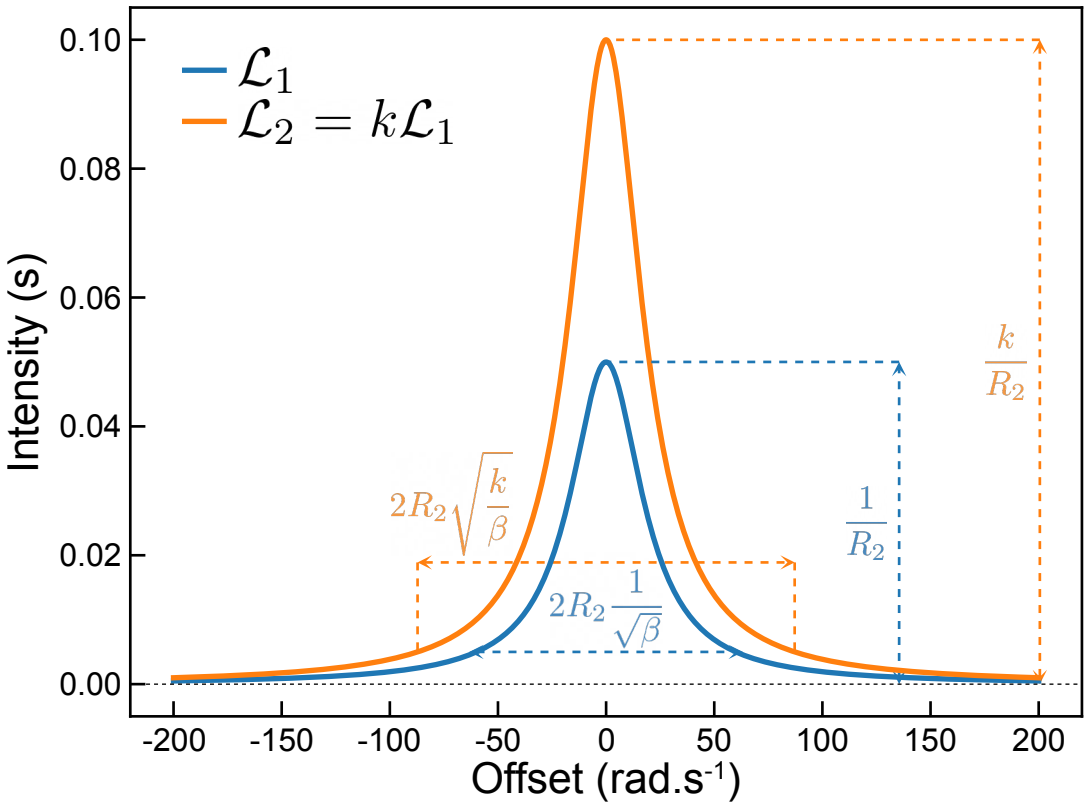
$$LW_2 = 2R_2\sqrt{\frac{k-\beta}{\beta}} \tag{S4}$$

At a level close to the noise, β is very small and the linewidths can be approximated as

$$LW_1 \approx 2R_2\frac{1}{\sqrt{\beta}} \tag{S5}$$

$$LW_2 \approx 2R_2\sqrt{\frac{k}{\beta}} \tag{S6}$$

Thus, at a sufficiently low level, the linewidth of $\mathcal{L}_1(\omega)$ is $\sqrt{k}$ smaller than that of $\mathcal{L}_2(\omega)$ and the peak of higher intensity appears broader. This is illustrated in the scheme below.



Scheme S1. Lineshapes of two Lorentzian functions, with $\mathcal{L}_2(\omega)$=$k\mathcal{L}_1(\omega)$, k=2, and $\omega_o = 0$. The linewidths at β/R_2 with R_2=20 s^{-1} and β=0.1 are shown.

References

[1] F. Delaglio, S. Grzesiek, GeertenW. Vuister, G. Zhu, J. Pfeifer, and A. Bax, "NMRPipe: A multidimensional spectral processing system based on UNIX pipes," *J. Biomol. NMR*, vol. 6, no. 3, pp. 257–293, Nov. 1995, doi: 10.1007/BF00197809.

[2] J. J. Helmus and C. P. Jaroniec, "Nmrglue: an open source Python package for the analysis of multidimensional NMR data," *J. Biomol. NMR*, vol. 55, no. 4, pp. 355–367, Apr. 2013, doi: 10.1007/s10858-013-9718-x.

[3] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, Mar. 2011, doi: 10.1109/MCSE.2011.37.

In what follows we include the pulse sequence code for the ddHMQC, as well as the cpd file for WALTZ-16 delayed decoupling.

Pulse sequence for the ddHMQC:

```
/* hmqc_lek_1G_delaydec_cp

  Used to record 13C 1H HMQC of methyl groups, with options for 15N, 13C
and 2H decoupling during t1
  Use for D2O samples and 15N,13C decoupling turned off

  Prefer to use WALTZ-16 decoupling (starting with delay of 4ms) at 3.3 kHz
B1
   Modify the WALTZ-16 cpd by adding d20 at the start

  Written by LEK

*/

#include <Avance.incl>
#include <Grad.incl>
#include <Delay.incl>

;Define phases
#define zero ph=0.0
#define one ph=90.0
#define two ph=180.0
#define three ph=270.0

;Define Pulses
define pulse pwh
       "pwh=p1"             ; 1H hard pulse at power pl1
define pulse pw_sl1
       "pw_sl1=p14"         ; eburp1 pulse
define pulse pwc
       "pwc=p2"             ; 13C hard pulse at power pl2
define pulse pwd
       "pwd=p4"             ; 2H pulse at power pl4

;Define delays

"in0=inf1/2"
"d11=30m"

define delay taua
       "taua=d5"            ; < 1/(4JCH) 1.8 ms

define delay tau1

define delay hscuba
  "hscuba=30m"
```

```
;define flags
;f1180    ; set zgoptns -Df1180

#ifdef comp9024090_flg
#ifdef f1180
  "d0=(in0 - pwc*4.0/PI - pwh*4.6667 - 16u - 2u - 2u)/2"
#else
  "d0=(0.2u - pwc*4.0/PI - pwh*4.6667 - 16u - 2u - 2u)/2"
#endif
#else
#ifdef f1180
  "d0=(in0 - pwc*4.0/PI - pwh*4.0 - 16u - 2u - 2u)/2"
#else
  "d0=(0.2u - pwc*4.0/PI - pwh*4.0 - 16u - 2u - 2u)/2"
#endif
#endif

#ifdef OneDarray
#else
define loopcounter ni
  "ni=td1/2"
#endif

;Ndec    ; set zgoptns -DNdec
;COdec   ; set zgoptns -DCOdec
;Ddec    ; set zgoptns -DDdec

/* Assign cnsts to check validity of parameter changes */
#ifdef fsat
   "cnst10 = plw10" ; tsatpwr - set to max of 0.00005W
#endif
#ifdef water_flg
   "cnst14=spw14"  ;  power  level  for  eburp1  pulse  preeceding  start  of
sequence
#endif
   "cnst21=plw21" ; dpwr pl21 - set max at 4.5W
#ifdef COdec
   "cnst22=plw22" ; dpwrco pl22/sw22 - set max at 8.0W
#endif

#ifdef Ndec
   "cnst31=plw31" ; dpwr2 pl31 - set max at 5.8W
#endif

#ifdef Ddec
   "cnst4=plw4"   ; dpwr3 pl4 - set max at 10.5W
   "cnst41=plw41" ; dpwr3D pl41 - set max at 1.5W
#endif

#ifdef Ddec
"acqt0 = -2u - 2u - pwd - 4u - 2u"
#endif
```

```
1 ze

; check validity of parameters

  if "d0 < 0.1u "
  {
   2u
   print "warning: initial d0 value is negative - First t1 point will be
incorrect Must LP"
  }

  if "aq > 0.130"
  {
   2u
   print "error: aq is too large <= 130 ms"
   goto HaltAcqu
  }

#ifdef fsat
  if "cnst10 > 0.001"
  {
   2u
   print "error: presat power pl10 is too large"
   goto HaltAcqu
  }
#endif

#ifdef water_flg
  if "cnst14 > 0.015"
  {
   2u
   print "error: power level for eburp1 pulse is too large"
   goto HaltAcqu
  }
#endif

#ifdef COdec
  if " cnst22 > 6.0"
  {
   2u
   print "error: dpwrco pl22 too large"
   goto HaltAcqu
  }
#endif

#ifdef Ndec
  if " cnst31 > 9.0"
  {
   2u
   print "error: dpwr2 pl31 too large"
   goto HaltAcqu
```

```
  }
#endif

  if "pwc > 20u"
  {
   2u
   print "error: pwc too large < 20 us"
   goto HaltAcqu
  }

#ifdef Ddec
  if "cnst4 > 13.5"
  {
   2u
   print "error: dpwr3 pl4 too large"
   goto HaltAcqu
  }

  if "cnst41 > 3.0"
  {
   2u
   print "error: dpwr3D pl41 too large"
   goto HaltAcqu
  }

;   d11 LOCKDEC_ON ; Not required for AvanceIII-HD
  50u LOCKH_ON
  d11 H2_PULSE
  2u pl41:f4      ; set power pl4 for 2H flipback pulses
#endif

2 d11 do:f2

 20u fq=cnst1:f1                     ; jump from methyls to water

#ifdef Ddec    /* D decoupling */
d11 H2_LOCK    ; put lock channel in lock mode
6m LOCKH_OFF   ; turn off lock hold

#ifdef fsat                           /* zgoptn -Dfsat */
  4u pl10:f1                          ; power(tsatpwr) for presaturation
  d1 cw:f1 zero                       ; Hcw(d1)x
  4u do:f1                            ; cw off
  2u pl1:f1                           ; power(tpwr)

#ifdef fscuba                    /* Scuba pulse sequence */
   hscuba                             ; delay(hscuba)
    (pwh zero):f1                     ; H 90x180y90x
    (pwh*2 one):f1
    (pwh zero):f1
   hscuba                             ; delay(hscuba)
#endif                             /* end fscuba */
```

```
#else                                /* if fsat is no */
 2u pl1:f1                           ; power(tpwr)
 d1                                  ; delay(d1)
#endif                               /* end if fsat */

50u  LOCKH_ON
15u H2_PULSE

#else    /* no D decoupling */

#ifdef fsat                              /* zgoptn -Dfsat */
  4u pl10:f1                             ; power(tsatpwr) for presaturation
  d1 cw:f1 zero                          ; Hcw(d1)x
  4u do:f1                               ; cw off
  2u pl1:f1                              ; power(tpwr)

#ifdef fscuba                          /* Scuba pulse sequence */
   hscuba                                ; delay(hscuba)
    (pwh zero):f1                        ; H 90x180y90x
    (pwh*2 one):f1
    (pwh zero):f1
   hscuba                                ; delay(hscuba)
#endif                                 /* end fscuba */

#else                                  /* if fsat is no */
 2u pl1:f1                             ; power(tpwr)
 d1                                    ; delay(d1)
#endif                                 /* end if fsat */
#endif                                 /* end of D_decoupling */

#ifdef Ddec
 20u UNBLKGRAMP          ; dly 20u, unblank gradients
#else
 20u UNBLKGRAD             ; dly 20u, unblank gradients and lock hold
#endif

#ifdef water_flg
; now apply eburp1 pulse

  2u
  (pw_sl1:sp14 zero):f1
  2u
#endif

  2u
  p50:gp0*0.5       ; gradient 0*0.5
  d16

#ifdef buffer_flg
  20u fq=cnst2:f1                      ; jump from methyls to buffer
  2u
```

```
  (pw_sl1:sp14 zero):f1
  2u
#endif

  10u fq=0:f1                          ; jump back to methyls
  2u pl1:f1
  2u pl2:f2
  2u pl31:f3

  (pwc zero):f2   ; C90x - To destroy 13C Boltzman magnetization

  2u
  (p50:gp0)       ; gradient 0
  d16

; This is the real start

"tau1 = d0"

if "tau1 < 0.2u" {
   "tau1 = 0.2u"
}

  (pwh zero):f1                          ; H90x

  2u
  p51:gp1                               ; gradient 1
  d16

 "DELTA = taua - 2.0u - p51 - d16 - pwh*2.0/PI"
  DELTA

  (center(pwh*2 ph26):f1 (pwc*2.0 ph26):f2)

  2u
  p51:gp1                               ; gradient 1
  d16

/*  If zgoptn -DDdec turn on 2H dec */

# ifdef Ddec
 "DELTA = taua - 2.0u - p51 - d16 - 2u - pwd - 2u - 2u - 2u + 2u + 2u + pwd
+ de + 4u + 2u"
  DELTA                                 ; delay 1/4JCH
  2u pl4:f4             ; dly 2u,  set pwr pl4 dpwr3
  (pwd one):f4          ; 2H 90(y)
  2u pl41:f4            ; dly 2u,  set pwr pl41 dpwr3D
  (2u cpd4 zero):f4     ; Turn on 2H decoupling - cpd4, phase x
# else
 "DELTA = taua - 2.0u - p51 - d16 - 2u + de + 4u + 2u" ; if !Ddec then
compensate for all subsequent delays
  DELTA                                 ; delay 1/4JCH
```

```
#endif

  2u pl2:f2
  (pwc ph3):f2                                ; C90ph3

/* If zgoptn -DCOdec turn on off-resonance CO dec */

# ifdef COdec
  2u pl22:f2
  (2u cpds8 zero):f2  ; Turn on CO dec, cpdprg8, sync mode
# else
  2u
  2u
#endif

/*  If zgoptn -DNdec turn on 15N dec */
# ifdef Ndec
  2u pl31:f3   ; set power pl31 for 15N decoupling
  (2u cpds3 zero):f3  ; Turn on 15N decoupling, cpdprg3 - waltz16
#else
  2u
  2u
# endif

  tau1                                  ; t1/2

#ifdef comp9024090_flg
  (pwh ph4):f1                    ; H90x
  2u
  (pwh*2.6667 ph5):f1             ;H 240/90
  2u
  (pwh ph4):f1                    ; H90x
#else
  (pwh ph4):f1                    ; H90x
  2u
  (pwh*2 ph5):f1                  ; H180y
  2u
  (pwh ph4):f1                    ; H90x
#endif

   tau1

# ifdef COdec
  2u do:f2   ; Turn off CO decoupling
  2u pl2:f2  ; set 13C high power
#else
  2u pl2:f2  ; set 13C high power
  2u
# endif

# ifdef Ndec
  2u
```

```
  2u do:f3    ; Turn off 15N decoupling on channel 3
#else
  2u
  2u
# endif

  (pwc ph8):f2                 ; C90 ph8

# ifdef Ddec
  2u do:f4              ; Turn off 2H decoupling on channel 4
  2u pl4:f4             ; dly 2u,  set pwr pl4 dpwr3
  (pwd three):f4        ; 2H 90(-y)
  4u BLKGRAMP
#else
  4u BLKGRAD
#endif

  2u pl21:f2               ; lower power for 13C decoupling

  go=2 ph31 cpds2:f2        ; acquire fid with delayed 13C decoupling
  d11 do:f2 mc #0 to 2
  F1PH(calph(ph3,  +90),calph(ph3,+180)  &  calph(ph31,  +180)  &  caldel(d0,
+in0))

#ifdef Ddec
d11 H2_LOCK
d11 LOCKH_OFF
; d11 LOCKDEC_OFF ; use statement for earlier hardware
#endif

HaltAcqu, 1m
exit

ph3=0 2
ph4=0 0 1 1 2 2 3 3
ph5=1 1 2 2 3 3 0 0
ph6=0
ph7=0 0 2 2
ph8=0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2
ph31=0 2 2 0 0 2 2 0 2 0 0 2 2 0 0 2
ph26=0
ph27=1
ph28=2
ph29=3

;d1 : repetition delay
;d5 : taua ~1/4JCH
;d11 : delay for disk i/o, 30ms
;d16 : gradient recovery delay, 200us
;d20 : set to exactly 1/2JCH when decoupling begins in t2
;pl1 : tpwr - power level for pwh
;pl2 : dhpwr - power level for hard 13C pulse pwc
```

```
;pl21 : dpwr - power level for  13C decoupling cpd2
;pl22 : dpwrcodec - power level for cos modulated seduce
;sp22 : cos modulated seduce decoupling pattern
;spw14 : power level for eburp1 pulse
;spnam14: eburp1 pulse on water
;pl23 : 5 or 6dB higher power than pl21
;pl31 : dpwr2 - power level for 15N cpd3
;pl4 : dpwr3 - power level for 2H flipback pulses
;pl41 : dpwr3D - power level for 2H cpd4
;p1 : pwh
;p14 : eburp1 pulse width, typically 7000u
;p2 : pwc
;p22 : pwco90 (seduce1 dec) pattern length us @ pl22 for CO decoupling
;p31 : pwn at dpwr2 for 15N decoupling during 2*TC(~29ms)
;p4 : pwd at dpwr3 for 2H flipback pulses
;p41 : pwddec at dpwr3D for 2H decoupling
;p63 : set to 1600 us
;ni : td1/2 number of complex points in t1
;cpd2 : 13C decoupling according to program defined by cpdprg2
;cpd3 : 15N decoupling according to program defined by cpdprg3
;cpd4 : 2H decoupling according to program defined by cpdprg4
;cpd8 : 13CO decoupling according to program defined by cpdprg8
;pcpd2:  13C 90 degree pulse at pl21 for cpd2
;pcpd3:  15N 90 degree pulse at pl31 for cpd3
;pcpd4:  2H 90 degree pulse at pl41 for cpd4
;pcpd8:  seduce1 decoupling pattern length for cpd8
;spnam22 : file name for CO decoupling
;spnam28 : file name for higher power bilevel dec
;spnam29 : file name for lower power bilevel dec
;cnst1 : water(Hz) - methyl(Hz)
;cnst2 : buffer(Hz) - methyl(Hz)
;zgoptns                                                                    :
Df1180,DNdec,DCOdec,DDdec,Dfsat,DoneDarray,Dfscuba,Dcomp9024090_flg,Dwater_
flg,Dbuffer_flg
```

Cpd file for decoupling:

```
  d20
1 pcpd*3:180
  pcpd*4:0
  pcpd*2:180
  pcpd*3:0
  pcpd  :180
  pcpd*2:0
  pcpd*4:180
  pcpd*2:0
  pcpd*3:180
2 pcpd*3:0
  pcpd*4:180
  pcpd*2:0
  pcpd*3:180
  pcpd  :0
```

```
  pcpd*2:180
  pcpd*4:0
  pcpd*2:180
  pcpd*3:0
  lo to 2 times 2
3 pcpd*3:180
  pcpd*4:0
  pcpd*2:180
  pcpd*3:0
  pcpd  :180
  pcpd*2:0
  pcpd*4:180
  pcpd*2:0
  pcpd*3:180
  jump to 1
```