Electronic Supplementary Information

Ultra-selective 1D clean in-phase correlation spectroscopy

Daniel A. Taylor,[a] Peter Kiraly,[b] Paul Bowyer,[c] Mathias Nilsson,[a] Laura Castañar,[a,d] Gareth A. Morris[a] and Ralph W. Adams*[a]

[a] Department of Chemistry, University of Manchester, Oxford Road, Manchester, M13 9PL

[b] JEOL UK Ltd., Bankside, Long Hanborough, OX29 8LJ

[c] JEOL UK Ltd., Silver Court, Welwyn Garden City, AL7 1LT

[d] Department of Organic Chemistry, Faculty of Chemical Science, University Complutense of Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain.

*Email: ralph.adams@manchester.ac.uk

Contents

S1. Experimental details

The lasalocid sample concentration was 10 mM in chloroform-*d*. The cyclosporin sample concentration was 100 mM in benzene-d_6. Lasalocid A sodium salt, and chloroform-*d* were supplied by Sigma-Aldrich; cyclosporin was supplied by Bio Basic; and benzene-d_6 was supplied by Apollo Scientific. All chemicals were used as supplied without any further purification.

NMR experiments were carried out using a 600 MHz JEOL JNM-ECZR (Delta 6.2) spectrometer with an HFX Royal probe equipped with Z-gradients, and a Bruker DRX 300 (TopSpin 1.3pl10) spectrometer with a dual $^{13}C/^{1}H$ probe equipped with Z-gradients. Raw experimental data, pulse programs and setup macros are available at DOI: 10.48420/21909600 and from our website (https://nmr.chemistry.manchester.ac.uk). The GEMSTONE 1D CLIP-COSY pulse program was derived from the GEMSTONE and CLIP-COSY pulse programs.[1,2] Alternatively other COSY pulse programs may be concatenated with GEMSTONE (see section S2 below).

The following parameters were used in the GEMSTONE 1D CLIP-COSY experiments on lasalocid. 20480 complex points were collected using a spectrum width of 13.5 kHz which corresponds to 1.51 s acquisition time. 4 dummy scans were discarded and 256 scans were accumulated with 0.5 s relaxation delay, corresponding to an experiment time of 10 minutes approximately. In the GEMSTONE part of the sequence, adiabatic swept frequency 180° pulses were 10 % smoothed CHIRP waveforms with duration $\tau_p = 80$ ms, sweep width of 2.5 kHz, and $B_1^{max} = 233.9$ Hz corresponding to adiabaticity $Q = 11$, applied in the presence of rectangular shaped magnetic field gradients of constant amplitude ± 0.22 G cm^{-1}; the band-selective refocusing 180° pulse was an rSNOB waveform with an effective bandwidth of 100 Hz and duration of 18.98 ms, flanked by rectangular gradient pulses (1 ms, 10.0 G cm^{-1}).

The CLIP-COSY mixing time (τ_m) was set to 120 ms, which is the optimal transfer for an 8.3 Hz *J*-coupling in an AX spin system. Zero-quantum suppression was employed using 10 % smoothed CHIRP 180° pulses with durations of 50 and 30 ms, sweep-width of 27.05 kHz, and B_1^{max} of 656 and 847 Hz corresponding to adiabaticity $Q = 5$. All waveforms were generated automatically using the user adjustable experiment parameters as implemented in standard JEOL experiments. Automation scripts for walk-up mode are available from the authors upon request. NMR spectra were processed using 0.5 Hz Gaussian apodization, zero filled to 65536 points prior to Fourier transformation and phase corrected automatically in JASON version 2.2.

The following parameters were used in the GEMSTONE 1D CLIP-COSY experiments on cyclosporin. 8192 complex points were collected over a spectral width of 6 kHz which corresponds to an acquisition time of 1.363 s. 4 dummy scans were discarded and 1024 scans were accumulated with a 3 s relaxation delay, corresponding to an experiment time of approximately 1 hour and 20 minutes. In the GEMSTONE part of the sequence, adiabatic swept frequency 180° pulses were WURST-80 waveforms with duration $\tau_p = 120$ ms, sweep width of 7.5 kHz, and $B_1^{max} = 302.0$ Hz corresponding to adiabaticity $Q = 11$, applied in the presence of rectangular shaped magnetic field gradients of constant amplitude $\pm$ 1.15 G cm^{-1}; the band-selective refocusing 180° pulse was an rSNOB waveform with an effective bandwidth of 200 Hz and duration of 11.66 ms, and was flanked by gradient pulses (1 ms, trapezoid shaped with a maximum amplitude of 7.95 G cm^{-1} and an integral factor of 0.9) to enforce the desired coherence transfer pathway (CTP) without additional phase cycling. The CLIP-COSY mixing time (τ_m) was set to 40 ms, which is the optimal transfer for a 25 Hz *J*-coupling in an AX spin system. Zero-quantum suppression was employed using adiabatic swept frequency 180° pulses which were WURST-80 waveforms with durations of 10 and 30 ms, sweep widths of 20 kHz, and $B_1^{max} = 1871.2$ and 1080.3 Hz corresponding to adiabaticity of $Q = 11$, applied simultaneously with rectangular shaped magnetic field gradients of constant amplitude 1.6 and

2.7 G cm^{-1}, respectively. All waveforms were generated using the Bruker "*stdisp*" shape tool, but can be generated automatically using WaveMaker in later versions of TopSpin. NMR spectra were processed using 0.8 Hz Gaussian apodization, zero filled to 65536 points prior to Fourier transformation and phase corrected automatically in TopSpin 4.0.8.

S2. GEMSTONE COSY variants

GEMSTONE can replace conventional selective excitation approaches in any 1D NMR experiment to cleanly excite a multiplet, even when it is severely overlapped. It uses magnetic field gradients to enforce the desired magnetisation transfer pathway; no additional experiment time due to phase cycling is required.

CLIP-COSY[2] is the most practical method for absorption mode 1D COSY, as the signals of both excited and correlated spins are in-phase. Other COSY sequences typically provide phase-twist (*i.e.*, mixtures of absorption and dispersion mode) or anti-phase absorption mode lineshapes, which can be difficult to analyse and interpret. Nevertheless, GEMSTONE can be used with any COSY mixing sequence. A schematic pulse sequence for GEMSTONE concatenated with the standard COSY element, as implemented on Bruker NMR spectrometers, is shown in **Figure S1**. GEMSTONE COSY and GEMSTONE CLIP-COSY spectra are provided for lasalocid (**Figure S2**) and cyclosporin (**Figure S3**) using the pulse sequences depicted in **Figure S1** and **Figure 2** of the main manuscript.

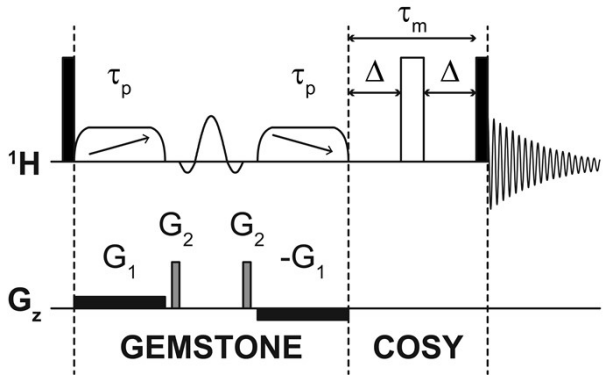

Figure S1. Schematic pulse sequence for GEMSTONE COSY. Filled and open rectangles, elements with diagonal arrows, and sinc shapes represent hard 90° and 180°, adiabatic 180° and band-selective 180° pulses, respectively. The two adiabatic pulses (of duration τ_p) control the selectivity of GEMSTONE; higher selectivity is achieved with a longer τ_p. The COSY mixing time is denoted τ_m and is two times the transfer delay Δ. The pulse program and all data files are available from DOI: 10.48420/21909600.

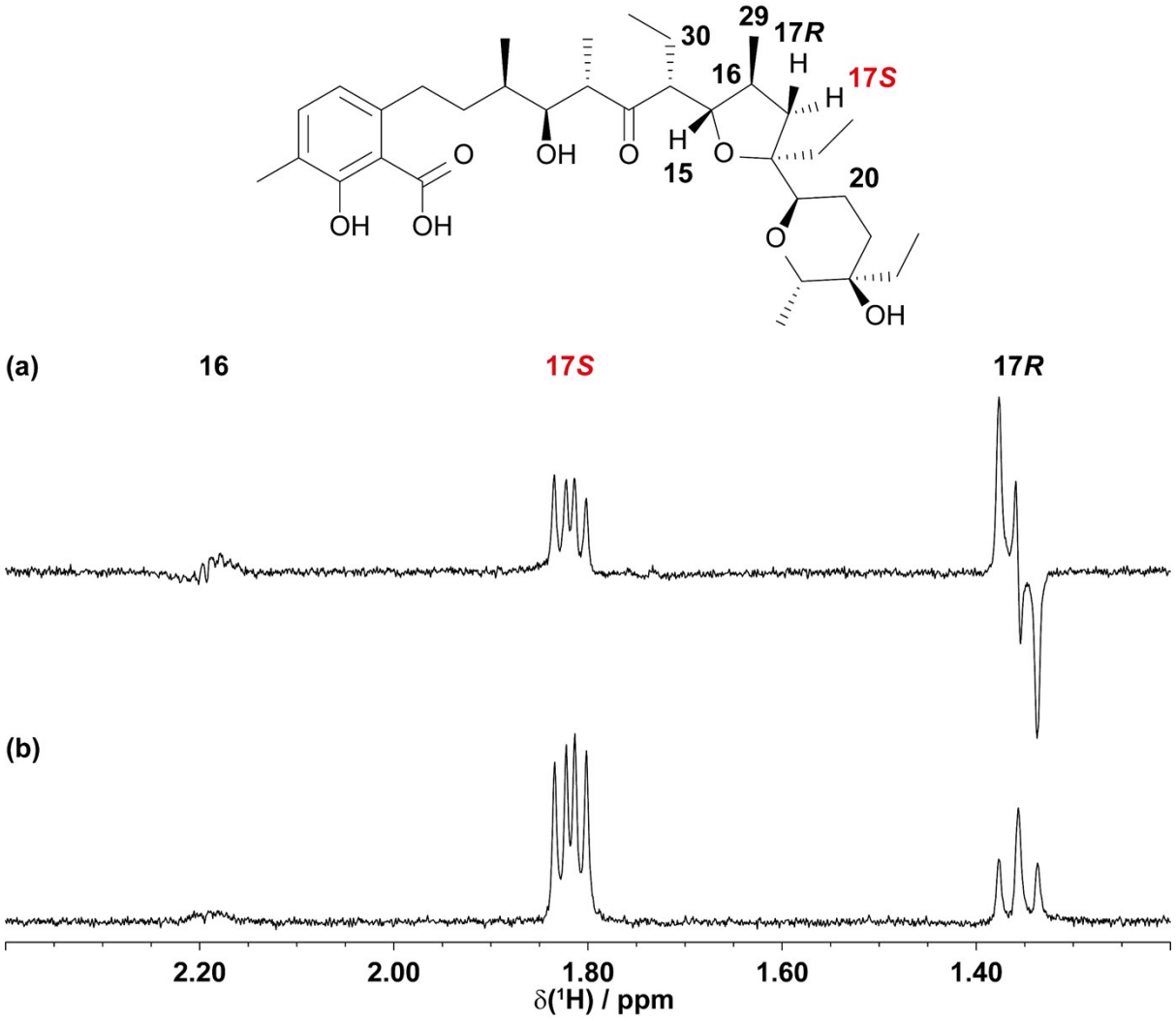

Figure S2. (a) GEMSTONE 1D COSY and **(b)** GEMSTONE 1D CLIP COSY spectra of lasalocid in $CDCl_3$ using the pulse sequences depicted in **Figure S1** and **Figure 2** of the main manuscript with COSY mixing time (τ_m) of 120 ms. The transmitter frequency was placed to excite signal **17*S***. Spectra were acquired using a 600 MHz JEOL JNM-ECZR (Delta 6.2) spectrometer with an HFX Royal probe equipped with Z-gradients.

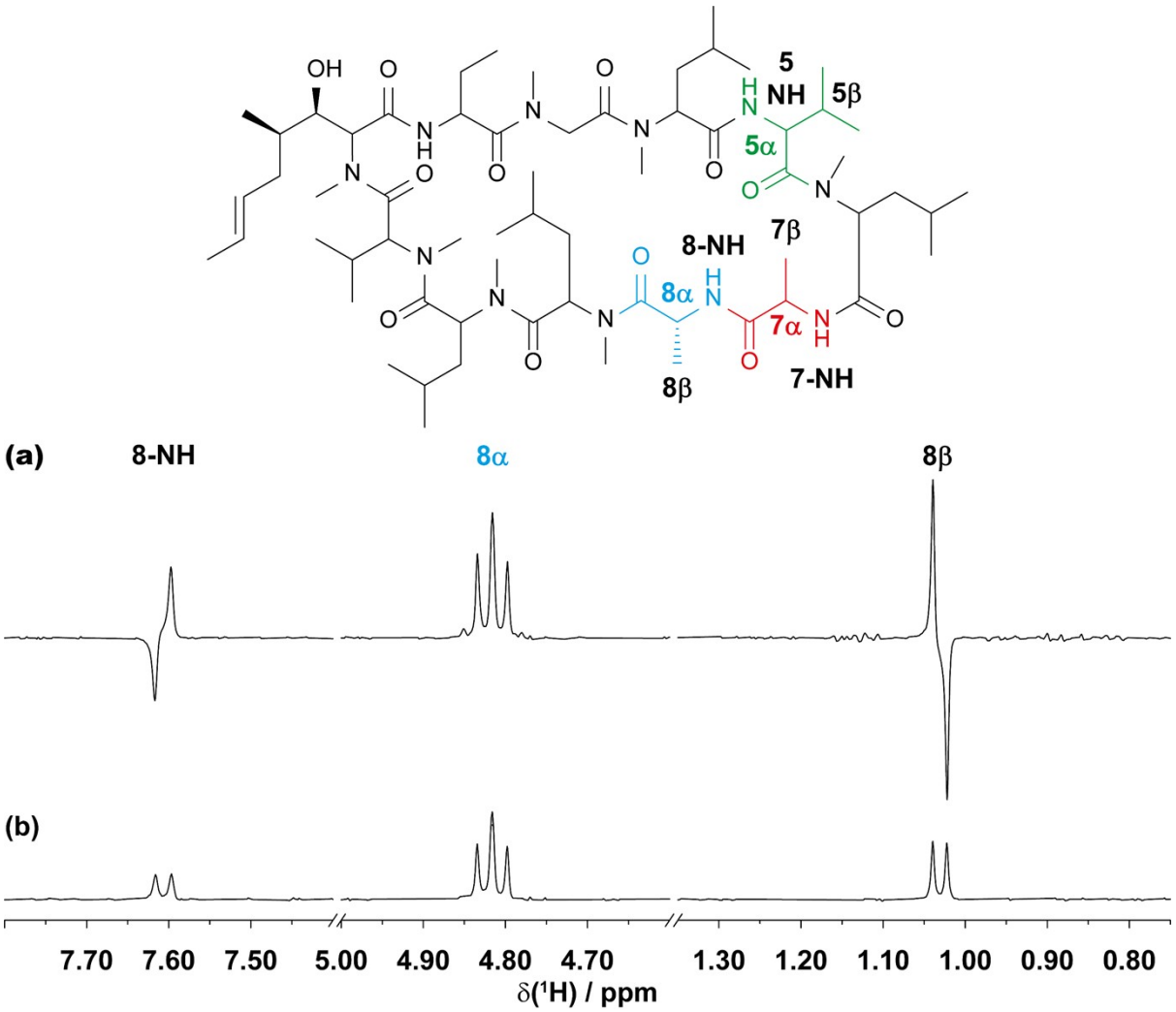

Figure S3. (a) GEMSTONE 1D COSY and **(b)** GEMSTONE 1D CLIP COSY spectra of cyclosporin in C_6D_6 using the pulse sequences depicted in **Figure S1** and **Figure 2** of the main manuscript with COSY mixing time (τ_m) of 40 ms . The transmitter frequency was placed to excite signal **8α**. Spectra were acquired using a 400 MHz Bruker Avance III (TopSpin 3.6.0) spectrometer with a BBFO H+F/X probe equipped with Z-gradients.

In the GEMSTONE COSY spectra of lasalocid and cyclosporin (**Figures S2a** and **S3a**), the respective excited signal is phased to pure absorption. Correlated signals thus appear in anti-phase, shifted in phase by 90°. Destructive interference between the lines of an anti-phase multiplet can cause signal cancellation, particularly when *J* is smaller than the linewidth.

In contrast, in the GEMSTONE CLIP-COSY spectra (**Figures S2b** and **S3b**), both excited and correlated signals have in-phase absorption mode lineshapes. The data is easily processed and interpreted.

S3. Simulation details

Amplitude profiles for the conventional selective spin echo CLIP-COSY and GEMSTONE CLIP-COSY experiments were evaluated (section S4) by using spin simulations in MATLAB with Spinach.[3] An AMX spin system, with scalar couplings J_{AM} = 6.93 Hz and J_{AX} = 7.71 Hz was used in the simulations. In the selective spin echo CLIP-COSY simulation, the A spin was excited using a 20 Hz REBURP[4] 180° pulse of duration 290.7 ms. The frequency at which the REBURP pulse was applied was changed in steps of 4 Hz over ±40 Hz from resonance, correlations to the directly coupled (M and X) spins were observed with a CLIP-COSY mixing time (τ_m) of 40 ms. In the GEMSTONE CLIP-COSY simulation, the A spin was excited using a 240 ms (τ_p = 120 ms) GEMSTONE element. The frequency at which the GEMSTONE element was applied was changed in steps of 4 Hz over ±40 Hz from resonance, correlations to the directly coupled (M and X) spins were observed with a CLIP-COSY mixing time (τ_m) of 40 ms.

Signal amplitudes as a function of CLIP-COSY transfer delay were evaluated (section S5) by using spin simulations in MATLAB with Spinach.[3] An AMX spin system with scalar couplings $J_{AM} = J_{AX}$ = 10 Hz and J_{MX} = 0 Hz, and an AMX spin system with scalar couplings $J_{AM} = J_{AX}$ = 10 Hz and J_{MX} = 5 Hz were used in the simulations. In each simulation, the A spin was excited using a 50 Hz rSNOB[5] pulse of duration 37.0 ms, correlations to the directly coupled (M and X) spins were observed under CLIP-COSY transfer delays changed in steps of 5 ms between 0 and 100 ms.

Spin system, parameter and experiment files used in the Spinach simulations are available at DOI: 10.48420/21909600.

S4. Simulation: Comparison between REBURP and GEMSTONE amplitude profiles

Conventional frequency-selective pulses[5,6] provide clean excitation of signals that are well-resolved. However, they are unable to selectively excite a signal when it is overlapped by other resonances; these neighbouring signals would also fall inside the effective bandwidth of the pulse and would be co-excited. Furthermore, signals close to, but outside of, the effective bandwidth of a frequency-selective pulse are often also co-excited; the excitation profile of a selective pulse contains transition regions, where excitation is neither uniform nor negligible. The REBURP pulse[4] provides the narrowest transition regions and is therefore the most suitable for selective excitation of a single signal where other signals are close in chemical shift, but there is still appreciable excitation (**Figure S4a**) outside the effective bandwidth specified, which causes poor frequency discrimination between signals close in chemical shift. Co-excitation of multiple signals can cause uncertainty in the assignment of COSY correlations (**Figure S4b**). If multiple signals are excited simultaneously then there is ambiguity in the assignment of coupling partners.

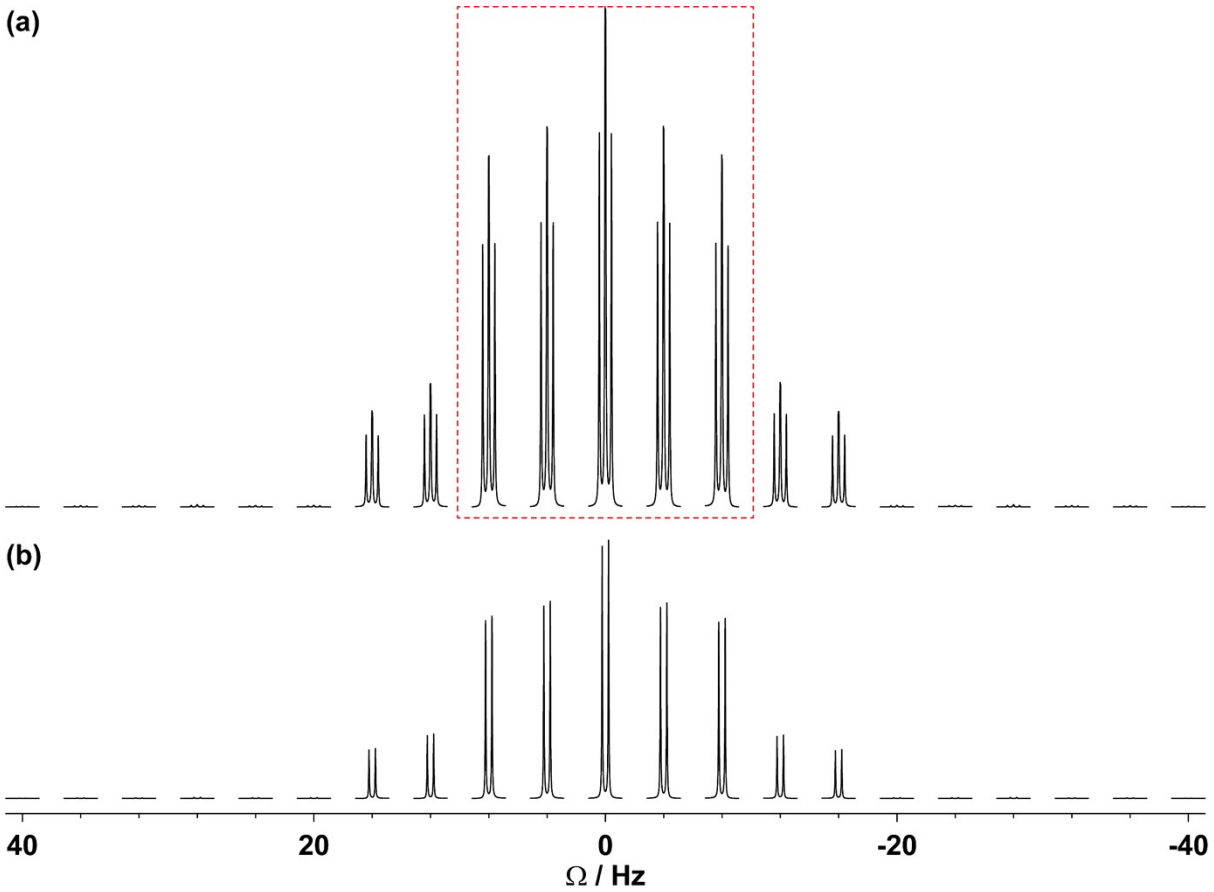

Figure S4. Selective CLIP-COSY amplitude profiles simulated using Spinach where a 20 Hz bandwidth REBURP pulse was used for excitation via a selective spin echo. An AMX spin system, with scalar couplings J_{AM} = 6.93 Hz, J_{AX} = 7.71 Hz and J_{MX} = 0 Hz was used in the simulation. (a) Excitation profile for spin A, and (b) amplitude profile for the M/X spin CLIP-COSY correlations where spin A was excited and a CLIP-COSY mixing time (τ_m) of 40 ms was used. The frequency at which the REBURP pulse was applied was changed in steps of 4 Hz over ± 40 Hz from resonance. The dashed red box in (a)

Conversely, GEMSTONE is chemical shift selective and can distinguish between overlapping multiplets, as long as there is a difference in chemical shift. The excitation profile (**Figure S5a**) takes the form of a sinc curve and provides narrow "central lobe" excitation close to resonance. Off-resonance excitation is minimal and, in practical terms, often indiscernible from thermal noise. GEMSTONE provides improved frequency discrimination between signals close in chemical shift and is more suitable for use with COSY, as its ability to select individual signals from overlapping multiplets enables the unambiguous identification of coupling partners.

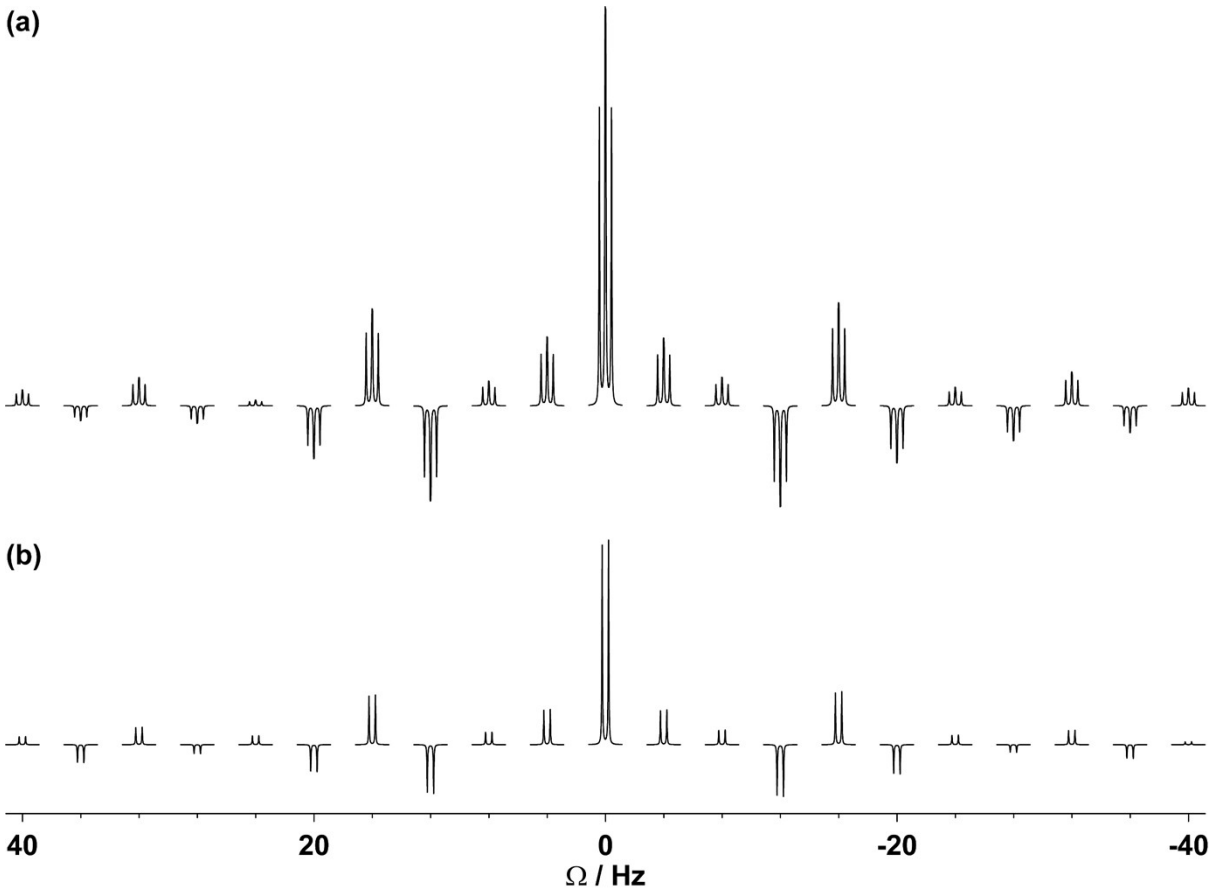

Figure S5. GEMSTONE CLIP-COSY amplitude profiles simulated using Spinach where a 240 ms (τ_p = 120 ms) GEMSTONE element was used for selective excitation. An AMX spin system, with scalar couplings J_{AM} = 6.93 Hz, J_{AX} = 7.71 Hz and J_{MX} = 0 Hz, was used in the simulation. (a) Excitation profile for the A spins, and (b) amplitude profile for the M/X spin CLIP-COSY correlations where A spins were excited and a CLIP-COSY mixing time (τ_m) of 40 ms was used. The frequency at which GEMSTONE was applied was changed in steps of 4 Hz over ± 40 Hz from resonance.

S5. Simulation: Optimisation of CLIP-COSY transfer delay

For a spin system containing a single active *J*-coupling (J_{12}) between two spins (1 and 2), the diagonal- (I_{diag}) and cross-peak (I_{cross}) signal intensities in CLIP-COSY are proportional to $\cos^2(2\pi J_{12}\Delta)$ and $\sin^2(2\pi J_{12}\Delta)$, respectively: complete magnetisation transfer from the excited to the coupled spin is therefore achieved with $\Delta = 1/4J_{12}$.

In most cases, any given spin is *J*-coupled to multiple other spins. Unless these couplings are identical, no single CLIP-COSY transfer delay (Δ) is optimum for magnetisation transfer to all coupled spins; a compromise value must be used. Spinach[3] spin simulations were used to determine the relationship between the magnitude of Δ and the amplitude of the excited and correlated spin signals. In the regime selected, there is minimal strong coupling. The effects of strong coupling on the performance of GEMSTONE and CLIP-COSY experiments are discussed in references 16, 17 and 21 of the main manuscript.

In an AMX spin system, with $J_{AM} = J_{AX} = 10$ Hz and $J_{MX} = 0$ Hz, the intensity of the excited signal (I_A) (**Figure S6a**) is modulated by the product of its active couplings (J_{AM}/J_{AX}) and closely matches the equation $I_A = \cos^2(2\pi J_{AM}\Delta)\cos^2(2\pi J_{AX}\Delta)$, see overlaid dashed red curve. The intensities of each of the coupled signals (I_M/I_X) (**Figure S6b**) are modulated by their respective active coupling to the excited signal (J_{AM}/J_{AX}), and the respective passive couplings of the excited signal (J_{AX}/J_{AM}) and closely matches the equation $I_M/I_X = \sin^2(2\pi J_{AM/AX}\Delta)\cos(2\pi J_{AX/AM}\Delta)$, see overlaid dashed blue curve.

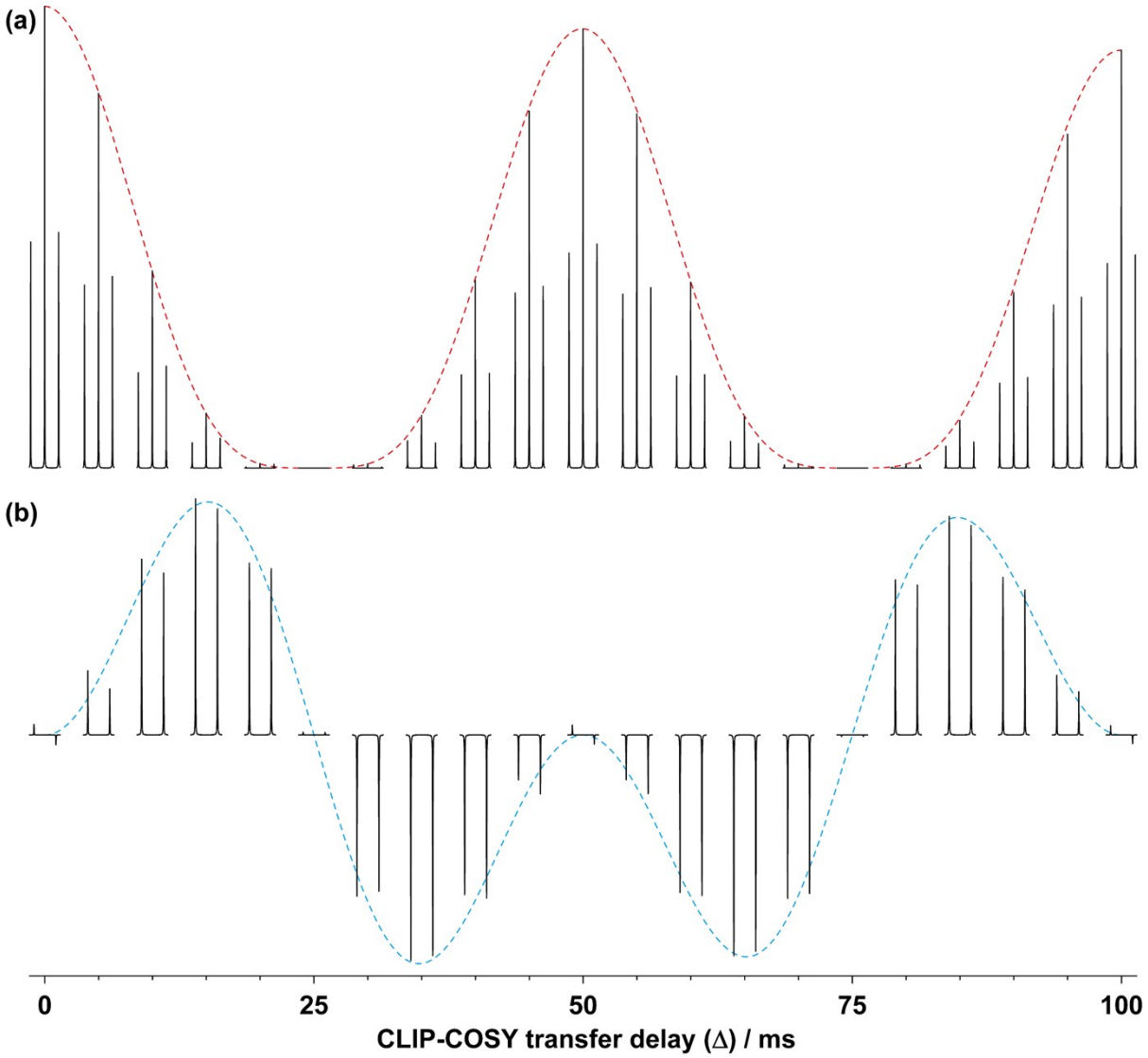

Figure S6. Signal intensities for the (a) excited and (b) coupled spins as a function of CLIP-COSY transfer delay (Δ) simulated using Spinach. An AMX spin system, with $J_{AM} = J_{AX} = 10$ Hz and $J_{MX} = 0$ Hz, was used in the simulation.

In an AMX spin system, with $J_{AM} = J_{AX} = 10$ Hz and $J_{MX} = 5$ Hz, the intensity of the excited signal (I_A) (**Figure S7a**) is modulated by the product of its active couplings (J_{AM}/J_{AX}) and closely matches the equation $I_A = \cos^2(2\pi J_{AM}\Delta)\cos^2(2\pi J_{AX}\Delta)$, see overlaid dashed red curve. The intensities (I_M/I_X) of each of the coupled signals (**Figure S7b**) are modulated by their respective active coupling to the excited signal (J_{AM}/J_{AX}), the respective passive couplings of the excited signal (J_{AX}/J_{AM}), and their couplings to any other signals (J_{MX}) and closely matches

the equation $I_M/I_X = \sin^2(2\pi J_{AM/AX}\Delta)\cos(2\pi J_{AX/AM}\Delta)\cos(2\pi J_{MX}\Delta)$, see overlaid dashed blue curve.

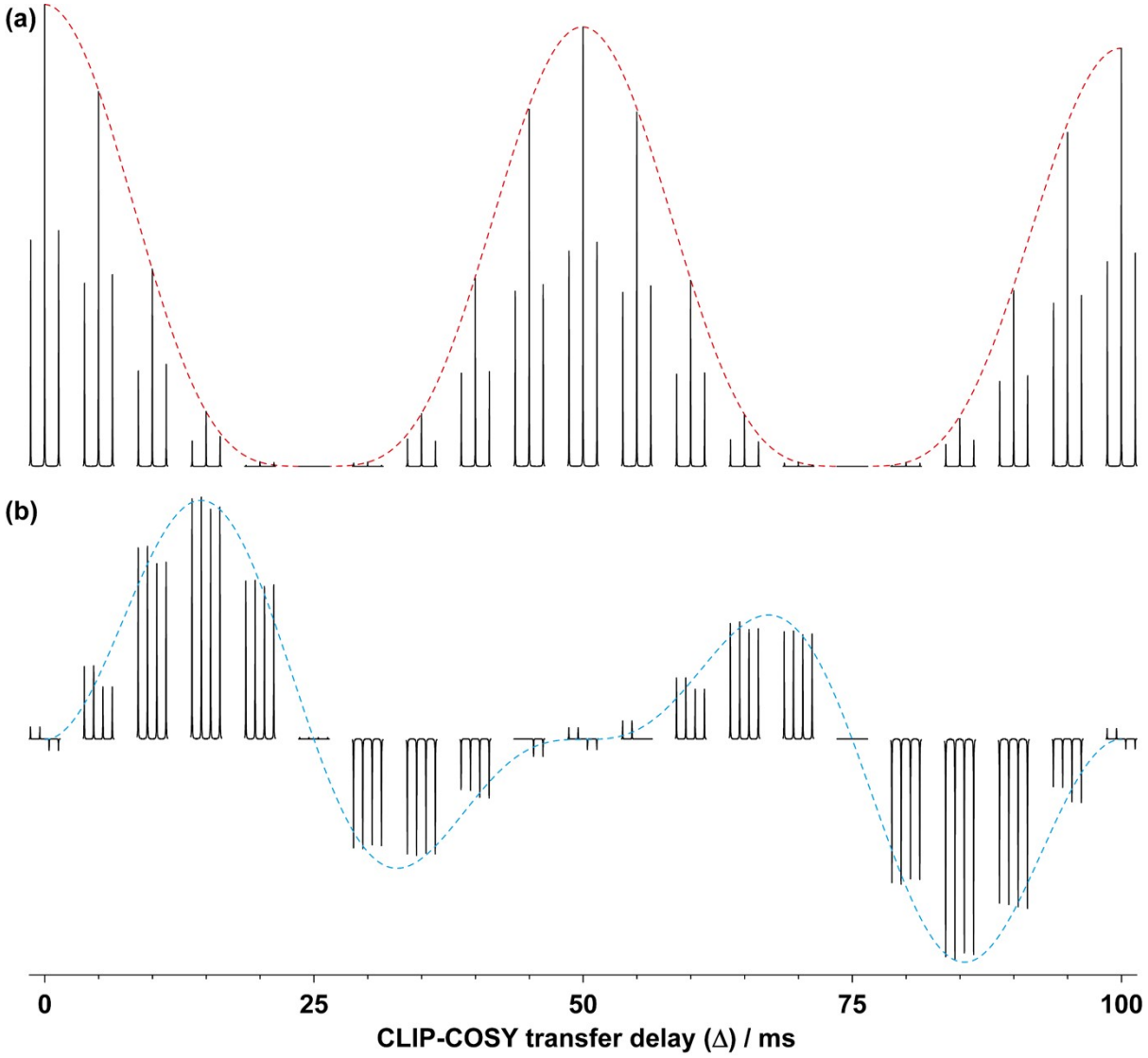

Figure S7. Signal intensities for the (a) excited and (b) coupled spins as a function of CLIP-COSY transfer delay (Δ) simulated using Spinach. An AMX spin system, with $J_{AM} = J_{AX} = 10$ Hz and $J_{MX} = 5$ Hz, was used in the simulation.

In the general case where spin 1 is *J*-coupled to spin 2 and *m* other spins *a*, and spin 2 is *J*-coupled to spin 1 and to *n* other spins *b*, the efficiency of magnetisation transfer from spin 1 to spin 2 via the active coupling J_{12} during the perfect echo mixing element of CLIP-COSY is proportional to equation [1].[2,7]

$$I_{cross} = sin^2(2\pi J_{12}\Delta) \prod_{a \neq 1,2}^{m+1} cos(2\pi J_{1a}\Delta) \prod_{b \neq 1,2}^{n+1} cos(2\pi J_{2b}\Delta) \quad [1]$$

Short transfer delays allow the signals of excited and coupled spins to be observed simultaneously and reduces the impact of relaxational attenuation. A transfer delay of 10 ms, which corresponds to a CLIP-COSY mixing time (τ_m) of 40 ms, generally works well for small molecules with typical $^3J_{HH}$ values of up to 10 Hz, and allows the signals of excited and coupled spins to be observed simultaneously.

S6. Pulse program codes

The original pulse program codes used for collecting data in this paper can be found in the comprehensive experimental data repository, see DOI: 10.48420/21909600. Example codes are listed here for convenience.

GEMSTONE CLIP-COSY experiment for Bruker Avance II+ spectrometers using TopSpin 3.

```
; Selective 1D CLIP-COSY experiment using GEMSTONE for ultra-selective
; excitation.
;
; For Bruker Avance II+ spectrometers using TopSpin 3. Automatic generation
; of waveforms using WaveMaker.
;
; Daniel Taylor
; Manchester NMR Methodology Group
; Department of Chemistry
; University of Manchester
;
; 27/05/2020

;$CLASS=HighRes
;$DIM=1D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

;WaveMaker definitions
;cnst50: band-width of the band-selective rSNOB pulse [Hz]
;cnst51: sweep-width of the adiabatic (WURST-80) pulse [Hz]
;cnst52: duration of the adiabatic (WURST-80) pulse [t1max/2: 30-100 ms]

;sp2(p12):wvm:dat_CSSF_a:f1 userA1(cnst50 Hz) ss=10.0us;
;sp41:wvm:dat_CSSF_bb1:f1  wurst-80(cnst51  Hz,  cnst52  ms;  L2H,  Q=11)
ss=20.0u;
;sp42:wvm:dat_CSSF_bb2:f1  wurst-80(cnst51  Hz,  cnst52  ms;  H2L,  Q=11)
ss=20.0u;

; ZQS elements
;cnst53: bandwidth of first ZQS pulse [kHz]
;cnst54: duration of first ZQS pulse [ms]
;cnst55: bandwidth of second ZQS pulse [kHz]
;cnst56: duration of second ZQS pulse [ms]

;sp29(p32):wvm:dat_ZQS1:f1  wurst-80(cnst53  kHz,  cnst54  ms;  NPOINTS=10000,
L2H, Q=11) ss=5.0us;
;sp30(p33):wvm:dat_ZQS2:f1  wurst-80(cnst55  kHz,  cnst56  ms;  NPOINTS=10000,
L2H, Q=11) ss=5.0us;

"d11=30m"
"d11=30m+1s/(1+cnst50)"
"d11=30m+1s/(1+cnst51)"
"d11=30m+1s/(1+cnst52)"
"d11=30m"

"p2=p1*2"

"acqt0=0.0"

baseopt_echo

1 ze
```

```
2 30m

  50u BLKGRAMP
  50u LOCKH_OFF
  d1 pl1:f1
  50u LOCKH_ON
  50u UNBLKGRAMP

  /* Hard 90deg pulse */
  (p1 ph1):f1

  /* Low to high adiabatic sweep */
  d16
  ( center (p41:sp41 ph2):f1 (p41:gp11) )
  d16

  /* Selective 180 */
  p16:gp1
  d16
  p12:sp2:f1 ph3:r
  p16:gp1
  d16 pl1:f1

  /* High to low adiabatic sweep */
  d16
  ( center (p42:sp42 ph4):f1 (p42:gp12) )
  d16 pl1:f1

  /* CLIP COSY */
  (p1 ph5):f1

  10u gron20
  (p32:sp29 ph6):f1
  20u groff
  d16 pl1:f1

  (p1 ph7):f1
  d4
  (p2 ph8):f1
  d4
  (p1 ph9):f1
  d4
  (p2 ph10):f1
  d4
  (p1 ph11):f1

  10u gron21
  (p33:sp30 ph12):f1
  20u groff
  d16 pl1:f1

  p16:gp2
  d16
  (p1 ph13):f1

  /* Acquisition */
  go=2 ph31
  30m mc #0 to 2 F0(zd)

  50u BLKGRAD
```

```
exit

ph1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
ph2 = 0
ph3 = 0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3
ph4 = 0
ph5 = 0 0 2 2
ph6 = 0
ph7 = 0
ph8 = 1
ph9 = 1
ph10 = 3
ph11 = 0
ph12 = 0
ph13 = 0 2
ph31 = 0 2 2 0 2 0 0 2 0 2 2 0 2 0 0 2 2 0 0 2 0 2 2 0 2 0 0 2 0 2 2 0


;pl1 : f1 channel - power level for pulse (default)
;sp2: f1 channel - shaped pulse, rSNOB
;sp29: f1 channel - shaped pulse (adiabatic), zero-quantum suppression
;sp30: f1 channel - shaped pulse (adiabatic), zero-quantum suppression
;sp41: f1 channel – shaped pulse (adiabatic), GEMSTONE L2H sweep
;sp42: f1 channel – shaped pulse (adiabatic), GEMSTONE H2L sweep

;p1 : f1 channel -  90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p12: f1 channel - 180 degree shaped pulse
;p16: homospoil/gradient pulse                                    [1 msec]
;p32: f1 channel – adiabatic pulse, zero-quantum suppression      [10 msec]
;p33: f1 channel – adiabatic pulse, zero-quantum suppression      [30 msec]
;p41: f1 channel – adiabatic pulse, GEMSTONE L2H sweep
;p42: f1 channel – adiabatic pulse, GEMSTONE H2L sweep

;d1 : relaxation delay; 1-5 * T1
;d4 : COSY mixing time; 1/(4*J(HH))
;d16: delay for homospoil/gradient recovery

;ns: 2 * n, total number of scans: NS * TD0
;ds: 4

;USERA1: Name of selective 180 pulse                              [rsnob]

;phcor 2 : phase difference between power levels sp1 and pl1

;for z-only gradients:
;gpz1: 15 %
;gpz2: -2 %
;gpz11: +0.5 %
;gpz12: -0.5 %

;use gradient files:
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100
;gpnam11: RECT.1
;gpnam12: RECT.1

; For sweepwidth of adiabatic shape and adjusting gpz0
; see supplementary material of M.J. Thrippleton & J. Keeler,
; Angew. Chem. Int. Ed. 42, 3938-3941 (2003)
; use pulse program zs_setup
```

```
                                          ;preprocessor-flags-start
;CALC_SPOFFS: automatically calcuate spoffs for selective pulses
;             start experiment with option -DCALC_SPOFFS (eda: ZGOPTNS)
;FLAG_BLK: for BLKGRAD before d1 rather than go
;             option -DFLAG_BLK: (eda: ZGOPTNS)
                                          ;preprocessor-flags-end

;$Id: dat_GEMSTONE_CLIPCOSY, v3, 2020/05/27 00:00:00 dat Exp $
```

GEMSTONE CLIP COSY experiment for Bruker DRX spectrometers using TopSpin 1.3.

```
; Selective 1D CLIP-COSY experiment using GEMSTONE for ultra-selective
; excitation.
;
; For Bruker DRX spectrometers using TopSpin 1.3. Manual generation of
; waveforms using Bruker shape tool.
;
; Daniel Taylor
; Manchester NMR Methodology Group
; Department of Chemistry
; University of Manchester
;
; 27/05/2020

;$CLASS=HighRes
;$DIM=1D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"d11=30m"

"p2=p1*2"

"acqt0=0.0"

1 ze
2 30m

  50u BLKGRAMP
  50u LOCKH_OFF
  d1 pl1:f1
  50u LOCKH_ON
  50u UNBLKGRAMP

  /* Hard 90deg pulse */
  (p1 ph1):f1

  /* Low to high adiabatic sweep */
  d16
  10u gron11
  (p24:sp24 ph2):f1
  10u groff
  d16

  /* Selective 180 */
  p16:gp1
  d16
  p12:sp12:f1 ph3:r
  p16:gp1
  d16 pl1:f1

  /* High to low adiabatic sweep */
  d16
  10u gron12
```

```
  (p25:sp25 ph4):f1
  10u groff
  d16 pl1:f1

  /* CLIP COSY */
  (p1 ph5):f1

  10u gron20
  (p32:sp29 ph6):f1
  20u groff
  d16 pl1:f1

  (p1 ph7):f1
  d4
  (p2 ph8):f1
  d4
  (p1 ph9):f1
  d4
  (p2 ph10):f1
  d4
  (p1 ph11):f1

  10u gron21
  (p33:sp30 ph12):f1
  20u groff
  d16 pl1:f1

  p16:gp2
  d16
  (p1 ph13):f1

  /* Acquisition */
  go=2 ph31
  30m mc #0 to 2 F0(zd)

  50u BLKGRAD

exit

ph1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
ph2 = 0
ph3 = 0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3
ph4 = 0
ph5 = 0 0 2 2
ph6 = 0
ph7 = 0
ph8 = 1
ph9 = 1
ph10 = 3
ph11 = 0
ph12 = 0
ph13 = 0 2
ph31 = 0 2 2 0 2 0 0 2 0 2 2 0 2 0 0 2 2 0 0 2 0 2 2 0 2 0 0 2 0 2 2 0


;pl1 : f1 channel - power level for pulse (default)
;sp12: f1 channel - shaped pulse, rSNOB
;sp24: f1 channel - shaped pulse (adiabatic), GEMSTONE L2H sweep
;sp25: f1 channel - shaped pulse (adiabatic), GEMSTONE H2L sweep
;sp29: f1 channel - shaped pulse (adiabatic), zero-quantum suppression
;sp30: f1 channel - shaped pulse (adiabatic), zero-quantum suppression
```

```
;p1 : f1 channel -  90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p12: f1 channel - 180 degree shaped pulse
;p16: homospoil/gradient pulse                                      [1 msec]
;p24: f1 channel - adiabatic pulse, GEMSTONE L2H sweep
;p25: f1 channel - adiabatic pulse, GEMSTONE H2L sweep
;p32: f1 channel - adiabatic pulse, zero-quantum suppression      [10 msec]
;p33: f1 channel - adiabatic pulse, zero-quantum suppression      [30 msec]

;d1 : relaxation delay; 1-5 * T1
;d4 : COSY mixing time; 1/(4*J(HH))
;d16: delay for homospoil/gradient recovery

;ns: 2 * n, total number of scans: NS * TD0
;ds: 4

;phcor 2 : phase difference between power levels sp1 and pl1

;for z-only gradients:
;gpz1: 15 %
;gpz2: -2 %
;gpz11: +0.5 %
;gpz12: -0.5 %

;use gradient files:
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100

; For sweepwidth of adiabatic shape and adjusting gpz0
; see supplementary material of M.J. Thrippleton & J. Keeler,
; Angew. Chem. Int. Ed. 42, 3938-3941 (2003)
; use pulse program zs_setup

                                          ;preprocessor-flags-start
;CALC_SPOFFS: automatically calcuate spoffs for selective pulses
;             start experiment with option -DCALC_SPOFFS (eda: ZGOPTNS)
;FLAG_BLK: for BLKGRAD before d1 rather than go
;             option -DFLAG_BLK: (eda: ZGOPTNS)
                                          ;preprocessor-flags-end

;$Id: dat_GEMSTONE_CLIPCOSY, v3, 2020/05/27 00:00:00 dat Exp $
```

Magnitude mode, GEMSTONE COSY experiment for Bruker Avance II+ spectrometers using TopSpin 3.

```
; Selective 1D COSY experiment using GEMSTONE for ultra-selective
; excitation.
;
; For Bruker Avance II+ spectrometers using TopSpin 3. Automatic generation
; of waveforms using WaveMaker.
;
; Daniel Taylor
; Manchester NMR Methodology Group
; Department of Chemistry
; University of Manchester
;
; 27/05/2020

;$CLASS=HighRes
;$DIM=1D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

;WaveMaker definitions
;cnst50: band-width of the band-selective rSNOB pulse [Hz]
;cnst51: sweep-width of the adiabatic (WURST-80) pulse [Hz]
;cnst52: duration of the adiabatic (WURST-80) pulse [t1max/2: 30-100 ms]

;sp2(p12):wvm:dat_CSSF_a:f1 userA1(cnst50 Hz) ss=10.0us;
;sp41:wvm:dat_CSSF_bb1:f1  wurst-80(cnst51  Hz,  cnst52  ms;  L2H,  Q=11)
ss=20.0u;
;sp42:wvm:dat_CSSF_bb2:f1  wurst-80(cnst51  Hz,  cnst52  ms;  H2L,  Q=11)
ss=20.0u;

"d11=30m"
"d11=30m+1s/(1+cnst50)"
"d11=30m+1s/(1+cnst51)"
"d11=30m+1s/(1+cnst52)"
"d11=30m"
“d5=d4-d16”

"p2=p1*2"

"acqt0=0.0"

baseopt_echo

1 ze
2 30m

  50u BLKGRAMP
  50u LOCKH_OFF
  d1 pl1:f1
  50u LOCKH_ON
  50u UNBLKGRAMP
```

```
  /* Hard 90deg pulse */
  (p1 ph1):f1

  /* Low to high adiabatic sweep */
  d16
  ( center (p41:sp41 ph2):f1 (p41:gp11) )
  d16

  /* Selective 180 */
  p16:gp1
  d16
  p12:sp2:f1 ph3:r
  p16:gp1
  d16 pl1:f1

  /* High to low adiabatic sweep */
  d16
  ( center (p42:sp42 ph4):f1 (p42:gp12) )
  d16 pl1:f1

  /* COSY */
  d5
  (p2 ph5):f1
  d16
  d5
  (p1 ph6:f1)

  /* Acquisition */
  go=2 ph31
  30m mc #0 to 2 F0(zd)

  50u BLKGRAD

exit

ph1 = 0
ph2 = 0 0 0 0 1 1 1 1
ph3 = 0 1
ph4 = 0 0 1 1 0 0 1 1
ph5 = 0
ph6 = 1
ph31 = 2 0 0 2 0 2 2 0

;pl1 : f1 channel - power level for pulse (default)
;sp2: f1 channel - shaped pulse, rSNOB
;sp41: f1 channel – shaped pulse (adiabatic), GEMSTONE L2H sweep
;sp42: f1 channel – shaped pulse (adiabatic), GEMSTONE H2L sweep

;p1 : f1 channel -  90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p12: f1 channel - 180 degree shaped pulse
;p16: homospoil/gradient pulse                                    [1 msec]
;p41: f1 channel – adiabatic pulse, GEMSTONE L2H sweep
;p42: f1 channel – adiabatic pulse, GEMSTONE H2L sweep

;d1 : relaxation delay; 1-5 * T1
;d4 : COSY mixing time; 1/(4*J(HH))
;d16: delay for homospoil/gradient recovery

;ns: 2 * n, total number of scans: NS * TD0
```

```
;ds: 4

;USERA1: Name of selective 180 pulse                                [rsnob]

;phcor 2 : phase difference between power levels sp1 and pl1

;for z-only gradients:
;gpz1: 15 %
;gpz11: +0.5 %
;gpz12: -0.5 %

;use gradient files:
;gpnam1: SMSQ10.100
;gpnam11: RECT.1
;gpnam12: RECT.1

; For sweepwidth of adiabatic shape and adjusting gpz0
; see supplementary material of M.J. Thrippleton & J. Keeler,
; Angew. Chem. Int. Ed. 42, 3938-3941 (2003)
; use pulse program zs_setup

                                          ;preprocessor-flags-start
;CALC_SPOFFS: automatically calcuate spoffs for selective pulses
;             start experiment with option -DCALC_SPOFFS (eda: ZGOPTNS)
;FLAG_BLK: for BLKGRAD before d1 rather than go
;             option -DFLAG_BLK: (eda: ZGOPTNS)
                                          ;preprocessor-flags-end

;$Id: dat_GEMSTONE_COSY, v2, 2020/05/27 00:00:00 dat Exp $
```

Magnitude mode, GEMSTONE COSY experiment for Bruker DRX spectrometers using TopSpin 1.3.

```
; Selective 1D COSY experiment using GEMSTONE for ultra-selective
; excitation.
;
; For Bruker DRX spectrometers using TopSpin 1.3. Manual generation of
; waveforms using Bruker shape tool.
;
; Daniel Taylor
; Manchester NMR Methodology Group
; Department of Chemistry
; University of Manchester
;
; 27/05/2020

;$CLASS=HighRes
;$DIM=1D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"d11=30m"
"d5=d4-d16"

"p2=p1*2"

"acqt0=0.0"

1 ze
2 30m

  50u BLKGRAMP
  50u LOCKH_OFF
  d1 pl1:f1
  50u LOCKH_ON
  50u UNBLKGRAMP

  /* Hard 90deg pulse */
  (p1 ph1):f1

  /* Low to high adiabatic sweep */
  d16
  10u gron11
  (p24:sp24 ph2):f1
  10u groff
  d16

  /* Selective 180 */
  p16:gp1
  d16
  p12:sp12:f1 ph3:r
  p16:gp1
  d16 pl1:f1
```

```
  /* High to low adiabatic sweep */
  d16
  10u gron12
  (p25:sp25 ph4):f1
  10u groff
  d16 pl1:f1

  /* CLIP COSY */
  d5
  (p2 ph5):f1
  d16
  d5
  (p1 ph6):f1

  /* Acquisition */
  go=2 ph31
  30m mc #0 to 2 F0(zd)

  50u BLKGRAD

exit

ph1 = 0
ph2 = 0 0 0 0 1 1 1 1
ph3 = 0 1
ph4 = 0 0 1 1 0 0 1 1
ph5 = 0
ph6 = 1
ph31 = 2 0 0 2 0 2 2 0


;pl1 : f1 channel - power level for pulse (default)
;sp12: f1 channel - shaped pulse, rSNOB
;sp24: f1 channel – shaped pulse (adiabatic), GEMSTONE L2H sweep
;sp25: f1 channel – shaped pulse (adiabatic), GEMSTONE H2L sweep

;p1 : f1 channel -  90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p12: f1 channel - 180 degree shaped pulse
;p16: homospoil/gradient pulse                                    [1 msec]
;p24: f1 channel – adiabatic pulse, GEMSTONE L2H sweep
;p25: f1 channel – adiabatic pulse, GEMSTONE H2L sweep

;d1 : relaxation delay; 1-5 * T1
;d4 : COSY mixing time; 1/(4*J(HH))
;d16: delay for homospoil/gradient recovery

;ns: 2 * n, total number of scans: NS * TD0
;ds: 4

;phcor 2 : phase difference between power levels sp1 and pl1

;for z-only gradients:
;gpz1: 15 %
;gpz11: +0.5 %
;gpz12: -0.5 %

;use gradient files:
;gpnam1: SMSQ10.100
```

```
; For sweepwidth of adiabatic shape and adjusting gpz0
; see supplementary material of M.J. Thrippleton & J. Keeler,
; Angew. Chem. Int. Ed. 42, 3938-3941 (2003)
; use pulse program zs_setup

                                          ;preprocessor-flags-start
;CALC_SPOFFS: automatically calcuate spoffs for selective pulses
;             start experiment with option -DCALC_SPOFFS (eda: ZGOPTNS)
;FLAG_BLK: for BLKGRAD before d1 rather than go
;             option -DFLAG_BLK: (eda: ZGOPTNS)
                                          ;preprocessor-flags-end

;$Id: dat_GEMSTONE_COSY, v2, 2020/05/27 00:00:00 dat Exp $
```

GEMSTONE CLIP-COSY experiment for JEOL ECZ/ECZL spectrometers using Delta 6.

```
------------------------------------------------------------------------
----
--                                                                    --
--                        Experiment Source Code                      --
--            Delta NMR Experiment & Machine Control Interface        --
--                                                                    --
--                    Copyright (c) 2021 JEOL Ltd                     --
--                           All Rights Reserved                      --
--                                                                    --
------------------------------------------------------------------------
----
-- HELP.eng: GEMSTONE CLIP-COSY
-- Category: 1D, CLIP-COSY, GEMSTONE,liquid_advanced
-- File name : GEMSTONE_CLIP-COSY
--
-- Sequence name : GEMSTONE CLIP-COSY
--
-- Reference :
-- P. Kiraly et al, Angew. Chem. Int. Ed. 10.1002/anie.202011642
-- P. Kiraly et al, Chem. Commun. , 2021,57, 2368-2371
-- D. Taylor et al,
--
-- Parameters
-- x_pulse          : 90[deg] pulse width
-- x_atn            : attenuator of x_pulse
--
-- obs_sel_180      : 180[deg] selective pulse
-- obs_sel_offset   : offset for obs_sel_180
-- obs_sel_atn      : attenuator of obs_sel_180
-- obs_sel_shape    : pulse shape of obs_sel_180
--
--
-- relaxation_delay : inter-pulse delay
-- repetition_time  : pulse repetition_time (= relaxation_delay+x_acq_time)
--
--
-- dante_presat     : select TRUE or FALSE for obs_dante_presaturation.
-- irr_mode            : select irr_mode, "OFF","Presaturation" or "Homo
Decouple".
-- tri_mode            : select tri_mode, "OFF","Presaturation" or "Homo
Decouple".
-- presat_time      : duration of dante presaturation.
--
-- dante parameters
--
-- dante_pulse      : pulse width of dante presaturation
-- dante_interval   : pulse interval of dante presaturation
-- dante_attenuator : attenuator of dante presaturation
-- dante_loop       : real loop number/100
--
-- irr/tri decoupling parameters
--
-- irr_domain       : nucleus of irr presaturation or homo_decoupling
-- irr_offset       : offset of irr presaturation or homo_decoupling
-- irr_attenuator   : attenuator of irr presaturation or homo_decoupling
--
-- tri_domain       : nucleus of tri presaturation or homo_decoupling
-- tri_offset       : offset of tri presaturation or homo_decoupling
```

```
-- tri_attenuator   : attenuator of tri presaturation or homo_decoupling
--
-- Note :
-- scans = 16*n
--
-- x90-PFG1-sel180-PFG1-PFG2-sel180-PFG2-mix_time-acq
-- PFG1 not = PFG2
--
--
-- END HELP

header
    filename        => "GEMSTONE_CLIP-COSY";
    sample_id       => "";
    comment         => "GEMSTONE CLIP-COSY";
    process         =  "proton_autophase.list";
    auto_dwell      =  TRUE;
    auto_filter     =  TRUE;
    auto_gain       => FALSE;
    filter_limit    =  12;
    force_dual_mode => FALSE;
    decimation_rate =  0;
    force_tune      => FALSE;
    power_2         =  FALSE;
    round_points    =  FALSE;
    save_aborted    =  TRUE;
    vector          =  TRUE;
    mod_save        => 0;
end header;

instrument
    recvr_gain          => 50;
    get_90              =  "pulse_service::get_90_value";
    get_atn             =  "pulse_service::get_atn_value";
    get_freq            =  "pulse_service::get_freq_value";
    get_spin            =  "pulse_service::get_spin";
    get_gamma           =  "pulse_service::get_gamma";
    get_value_of_field  =  "pulse_service::get_value_of_field";
    get_probe_parameter =  "probe_service::get_probe_parameter";
    get_s_mparam        =  "pulse_service::get_s_mparam";
    get_d_mparam        =  "pulse_service::get_d_mparam";
    spin_state          => "SPIN OFF";
end instrument;

acquisition
    x_domain             => "Proton";
    x_offset             => 5[ppm];
    x_sweep              => 18[ppm];
    default_x_resolution = _get_value_of_field(x_domain, "x_1D_resolution");
    x_data_points        =  _get_value_of_field(x_domain, "x_data_points");
    x_points_default       =? if upper (x_sweep / default_x_resolution) >
x_data_points
                             then x_data_points
                             else upper (x_sweep / default_x_resolution);
    x_points             => 16384;
    scans                => 32, help "scans = 4*n";
    x_prescans           => 4;
    mod_return           => 8;
    x_acq_time           =? x_points / x_sweep;
    x_resolution         =? 1 / x_acq_time;
```

```
    x_sweep_input               =    lower  (x_sweep  *  transition_ratio  /
_get_freq(x_domain) * 1[Mppm]);
    x_points_input       =  lower (x_points * transition_ratio);
end acquisition;

pulse
    collect COMPLEX,OBS;

    macro    obs_presat_time(    dante_presat,    dante_loop,    dante_pulse,
dante_interval, phase_dante, phs_shft, dante_attenuator, presat_time ) is
        when dante_presat do
            loop 100 * dante_loop times
                dante_pulse, (OBS.GATE, OBS.PHS.phase_dante.lstep(phs_shft),
OBS.ATN.dante_attenuator);
                dante_interval;
            end loop;
        end when;
        when not dante_presat do
            presat_time;
        end when;
    end obs_presat_time;

    macro acq( dead_time, delay, phase ) is
        dead_time, OBS.PHS.0;
        delay, (OBS.PHS.0, RCV.GATE);
        acquire (OBS.PHS.0, RCV.PHS.phase);
    end acq;

    experiment               =  "GEMSTONE_clip-cosy.jxp";
    obs_domain               =  x_domain;
    obs_offset               =  x_offset;
    obsfreq                  =  _get_freq(obs_domain);
    comment_1                =? "*** Pulse ***";
    x_pulse                  => x90, help "observe 90[deg] pulse";
    x_atn                    =? xatn, help "attenuator for x_pulse";
    obs_pulse                =  x_pulse;
    obs_atn                  =  x_atn;
    comment_56               =? "*** GEMSTONE ***";
    obs_chpgs_shape          => "CHIRP", ad_shape_names, help "shape pulse";
    obs_chpgs_m_pulse        => 50[ms], help "ZQfilter pulse width";
    obs_chpgs_m_fsweep       => 2.5[kHz];
    obs_chpgs_sweep_ppm      =  obs_chpgs_m_fsweep / obsfreq * 1[Mppm];
    obs_chpgs_chirp_smooth   => 10[%];
    obs_chpgs_m_q0           => 11, help "q0 >= 11";
    obs_chpgs_m_ph_rev       =  1;
    obs_chpgs_b1_calc        =  sqrt (obs_chpgs_m_q0 * obs_chpgs_m_fsweep /
(6.28319 * obs_chpgs_m_pulse));
    obs_chpgs_m_b1max        => obs_chpgs_b1_calc;
    obs_chpgs_pw90           =  1 / (4 * obs_chpgs_m_b1max);
    obs_chpgs_atn_calc        =   obs_atn + 20[dB] * log (obs_chpgs_pw90 /
obs_pulse);
    obs_chpgs_atn            => obs_chpgs_atn_calc;
    obs_chpgs2_m_fsweep      =  obs_chpgs_m_fsweep;
    obs_chpgs2_m_pulse       =  obs_chpgs_m_pulse;
    obs_chpgs2_chirp_smooth  =  obs_chpgs_chirp_smooth;
    obs_chpgs2_m_q0          =  obs_chpgs_m_q0;
    obs_chpgs2_m_ph_rev      =  -1, (1, -1), help "phase reverse";
    obs_chpgs2_atn           =  obs_chpgs_atn_calc;
    grad_sl1                 =? obs_chpgs_m_pulse;
    grad_sl1_amp            => 2.5[mT/m], help "Amplitude of gradient during
1st chirp";
```

```
    grad_sl2                =  obs_chpgs_m_pulse;
    grad_sl2_amp           =? -1 * grad_sl1_amp, help "Amplitude of gradient
during 2nd chirp";
    grad_shape_type            => "SQUARE", fg_shape_names, help "gradient
shape";
    comment_32              =? "*** Selective Pulse 180deg  ***";
    obs_sel_shape          => "RSNOB_180", std_shape_names, help "shape of
obs_sel_180";
    soft_shape_input        =  obs_sel_shape;
    obs_sel_offset          =? x_offset, help "offset of obs_sel_180";
    soft_bw_input          => 100[Hz], help "bandwidth in [ppm] or [Hz] of
obs_sel_180";
    xfreq                   =  _get_freq(x_domain);
    soft_angle               =? _get_s_mparam(soft_shape_input, "m_angle",
90[deg]);
    int_soft_pc              =  _get_s_mparam(soft_shape_input, "m_integ",
100[%]);
    int_r_soft_pc              =  int_soft_pc / _get_s_mparam("gauss_90",
"m_integ", 100[%]);
    unit_soft_pc            =  if soft_bw_input * 0 = 0[ppm]
                               then 1
                               else if soft_bw_input * 0 = 0[Hz]
                               then 2
                               else 3;
    soft_bw_hz              =  if unit_soft_pc = 1
                               then soft_bw_input * xfreq / 1[Mppm]
                               else if unit_soft_pc = 2
                               then soft_bw_input
                               else 1[Hz];
    type_soft_pc             =  _get_s_mparam(soft_shape_input, "m_type",
"std");
    is_std_pc                =  if type_soft_pc = "std" and unit_soft_pc <
3
                               then TRUE
                               else FALSE;
    soft_pw_calc            =? if is_std_pc
                               then  _get_s_mparam(soft_shape_input,  "bw",
1) / soft_bw_hz
                               else 1[us];
    soft_atn_calc           =? if is_std_pc
                               then xatn_soft + 20[dB] * log (soft_pw_calc
/ x90_soft * 90[deg] / soft_angle * int_r_soft_pc)
                               else 79[dB];
    obs_sel_180                 => soft_pw_calc, help "selective 180[deg]
pulse";
    obs_sel_atn          => soft_atn_calc, help "attenuator of obs_sel_180";
    obs_sel_slp             =  obs_sel_offset;
    comment_50              =? "*** CLIP-COSY ***";
    delta                  => 15[ms], help "transfer delay for CLIP-COSY";
    initial_wait            =  1[s];
    rpt_min                 =  x_points * filter_limit * 3.64[us] + 10[ms];
    rpt_def_temp            =  if x_domain = "Proton"
                               then 7[s]
                               else if x_domain = "Fluorine19"
                               then 7[s]
                               else if x_domain = "Carbon13"
                               then 3[s]
                               else 5[s];
    rd_def_temp             =  if rpt_def_temp - x_acq_time > 0.1[s]
                               then rpt_def_temp - x_acq_time
                               else 0.1[s];
```

```
    relaxation_delay_default =  if rpt_def_temp > rpt_min
                                then rd_def_temp
                                else if rpt_min - x_acq_time > 0[s]
                                then rpt_min - x_acq_time
                                else 0.1[s];
    relaxation_delay         => 0.5[s], help "inter-pulse delay";
    repetition_time                =? relaxation_delay + x_acq_time, help
"relaxation_delay+x_acq_time";
    comment_8                =? "*** Pulse Field Gradient ***";
    gradient_max             =? z_gradient_max, help "Maximum amplitude for
a given probe as defined in the probe file";
    grad_1                   => 1[ms], help "pulse width of PFG1";
    grad_1_amp                  => 0.1[T/m], help "amplitude of PFG1(not =
PFG2)";
    grad_2                   =  grad_1, help "pulse width of PFG2";
    grad_2_amp                  => 60[mT/m], help "amplitude of PFG2(not =
PFG1)";
    grad_shape                 => "SQUARE", fg_shape_names, help "shape of
gradients";
    grad_recover             => 0.2[ms];
    grad_recover_zq          => 5[ms];
    comment_41               =? "*** Zero-Quantum Dephasing ***";
    obs_chp_shape           =? "CHIRP", ad_shape_names, help "shape pulse";
    obs_chp1_m_pulse            => 50[ms], (30[ms], 40[ms], 50[ms]), help
"ZQfilter pulse width";
    obs_chp1_m_fsweep        =? if obs_chp1_m_pulse = 30[ms]
                                then x_sweep * 2
                                else if obs_chp1_m_pulse = 40[ms]
                                then x_sweep * 2
                                else x_sweep * 2;
    obs_chp1_sweep_ppm     =? round (obs_chp1_m_fsweep * 1[Mppm] / obsfreq),
help "obs_chp1_ad_sweep_ppm";
    obs_chp1_chirp_smooth    =  10[%];
    obs_chp1_m_q0               =   5, help "q >= 5 On resonance adiabatic
Quality factor";
    obs_chp1_m_ph_rev        =  1, (1, -1), help "phase reverse";
    obs_chp1_b1_calc          =  sqrt (obs_chp1_m_q0 * obs_chp1_m_fsweep /
(6.28319 * obs_chp1_m_pulse));
    obs_chp1_m_b1max         =? obs_chp1_b1_calc;
    obs_chp1_pw90            =  1 / (4 * obs_chp1_m_b1max);
    obs_chp1_atn_calc         =  if obs_atn + 20[dB] * log (obs_chp1_pw90 /
obs_pulse) < obs_atn + 6[dB]
                                then obs_atn + 6[dB]
                                else obs_atn + 20[dB] * log (obs_chp1_pw90
/ obs_pulse);
    obs_chp1_atn             =? obs_chp1_atn_calc;
    obs_chp2_m_pulse            => 30[ms], (30[ms], 40[ms], 50[ms]), help
"ZQfilter pulse width";
    obs_chp2_m_fsweep        =? if obs_chp2_m_pulse = 30[ms]
                                then x_sweep * 2
                                else if obs_chp2_m_pulse = 40[ms]
                                then x_sweep * 2
                                else x_sweep * 2;
    obs_chp2_sweep_ppm     =? round (obs_chp2_m_fsweep * 1[Mppm] / obsfreq),
help "obs_chirp2_ad_sweep_ppm";
    obs_chp2_chirp_smooth    =  10[%];
    obs_chp2_m_q0               =   5, help "q >= 5 On resonance adiabatic
Quality factor";
    obs_chp2_m_ph_rev        =  1, (1, -1), help "phase reverse";
    obs_chp2_b1_calc          =  sqrt (obs_chp2_m_q0 * obs_chp2_m_fsweep /
(6.28319 * obs_chp2_m_pulse));
```

```
    obs_chp2_m_b1max         =? obs_chp2_b1_calc;
    obs_chp2_pw90            =  1 / (4 * obs_chp2_m_b1max);
    obs_chp2_atn_calc         =  if obs_atn + 20[dB] * log (obs_chp2_pw90 /
obs_pulse) < obs_atn + 6[dB]
                               then obs_atn + 6[dB]
                               else obs_atn + 20[dB] * log (obs_chp2_pw90
/ obs_pulse);
    obs_chp2_atn             =? obs_chp2_atn_calc;
    g_factor                 =  1, help "obs_gamma/1H_gamma";
    coil_factor              =  1, help "coil_length/20mm";
    f_factor_zqf1             =   obs_chp1_m_fsweep / 40[kHz], help "ref
10ppm*8/500MHz=40kHz";
    grad_zqf1_amp_calc        =  47[mT/m] * f_factor_zqf1 / coil_factor /
g_factor, help "gradient amplitude of grad_sl_ps";
    grad_zqf1_amp           =? grad_zqf1_amp_calc, help "Enter amplitude in
Tesla/meter units";
    f_factor_zqf2               =   obs_chp2_m_fsweep / 40[kHz], help "ref
10ppm*8/500MHz=40kHz";
    grad_zqf2_amp_calc        =  47[mT/m] * f_factor_zqf2 / coil_factor /
g_factor, help "gradient amplitude of grad_sl_ps";
    grad_zqf2_amp           =? grad_zqf2_amp_calc, help "Enter amplitude in
Tesla/meter units";
    comment_900              =? "*** lock hold ***";
    lock_hold                 => FALSE, help "select TRUE or FALSE for lock
hold";
    comment_201              =? "*** obs_dante_presaturation ***";
    dante_presat                          =>  FALSE,  help  "True/False  for
obs_dante_presaturation";
    when dante_presat do
        dante_offset                  => x_offset, help "offset of dante
presaturation";
        dante_pulse                  => 4[us], help "pulse width of dante
presaturation";
        dante_interval          => 0.1[ms] - dante_pulse, help "interval of
dante pulse train";
        dante90_temp             =  x90;
        danteatn_temp            =  xatn;
        dante_temp           =  dante_pulse / (dante_pulse + dante_interval);
        dante_atn_default        =? danteatn_temp + 20[dB] * log (8.3[ms] *
dante_temp / dante90_temp), help "calculate a value of B1=30Hz";
        dante_attenuator           => dante_atn_default, 10[dB]->119[dB] :
10[mdB], help "attenuator of dante presaturation";
        proton_freq              =  _get_freq(x_domain);
        diff_offset              =  (dante_offset - x_offset) * 1[Mppm-1] *
proton_freq;
        phs_shft             =  diff_offset * (dante_pulse + dante_interval)
* 360;
        phase_dante              =  {0};
    end when;
    comment_202            =? "*** irr_ presaturation & homo spin decoupling
***";
    irr_mode                      => "Off", ("Off", "Presaturation", "Homo
Decouple"), help "Select Presaturation or Homo Decouple";
    when irr_mode = "Homo Decouple" do
        note_irr_homodec         =? "Set [filter_limit = 2-4] in the Header
parameters";
    end when;
    when irr_mode /= "Off" do
        irr_domain           =? x_domain, help "nucleus of irr presaturation
or homo_decoupling";
```

```
        irr_offset              => 5[ppm], help "offset of irr presaturation
or homo_decoupling";
        irr90_temp               =  x90;
        irratn_temp              =  xatn;
        irr_atn_default           =? irratn_temp + 20[dB] * log (8.3[ms] /
irr90_temp), help "calculate a value of B1=30Hz";
        irr_attenuator               => irr_atn_default, 10[dB]->119[dB] :
10[mdB], help "attenuator of irr presaturation or homo_decoupling";
    end when;
    comment_203            =? "*** tri_ presaturation & homo spin decoupling
***";
    tri_mode                      => "Off", ("Off", "Presaturation", "Homo
Decouple"), help "Select Presaturation or Homo Decouple";
    when tri_mode = "Homo Decouple" do
        note_tri_homodec         =? "Set [filter_limit = 2-4] in the Header
parameters";
    end when;
    when tri_mode /= "Off" do
        tri_domain           =? x_domain, help "nucleus of tri presaturation
or homo_decoupling";
        tri_offset              => 5[ppm], help "offset of tri presaturation
or homo_decoupling";
        tri90_temp               =  x90;
        triatn_temp              =  xatn;
        tri_atn_default           =? triatn_temp + 20[dB] * log (8.3[ms] /
tri90_temp), help "calc.value of B1=30Hz";
        tri_attenuator               => tri_atn_default, 10[dB]->119[dB] :
10[mdB], help "attenuator of tri presaturation or homo_decoupling";
    end when;
    comment_111              =? "*** presat_time ***";
    presat_time_flag         => FALSE, help "select TRUE or FALSE for using
presat_time";
    when presat_time_flag do
        presat_time_temp       => relaxation_delay, help "presaturation_time
<= relaxation_delay";
    end when;
    presat_time              =? if relaxation_delay > presat_time_temp
                                then presat_time_temp
                                else relaxation_delay;
    relaxation_delay_calc    =  relaxation_delay - presat_time;
    dante_loop                =  lower (presat_time / (100 * (dante_pulse +
dante_interval))), help "real loop number/100";
    dead_time                =  if not auto_dwell
                                then 0.2 / x_sweep
                                else 0[us];
    delay                    =  if not auto_dwell
                                then 0.8 / x_sweep
                                else 0[us];
    phase_1                  =  {16(0), 16(180)};
    phase_2                  =  {0};
    phase_3                  =  {4(0), 4(90), 4(180), 4(270)};
    phase_4                  =  {0};
    phase_5                  =  {2(0), 2(180)};
    phase_6                  =  {0};
    phase_7                  =  {0};
    phase_8                  =  {90};
    phase_9                  =  {90};
    phase_10                 =  {270};
    phase_11                 =  {0};
    phase_12                 =  {0};
    phase_13                 =  {0, 180};
```

```
    phase_acq                =  {2(0, 180, 180, 0, 180, 0, 0, 180), 2(180,
0, 0, 180, 0, 180, 180, 0)};
    module_config            =  if irr_mode = "Homo Decouple"
                                then if tri_mode = "Homo Decouple"
                                     then   "irr.time_share   tri.time_share
obs.blank_inhibit"
                                     else "irr.time_share obs.blank_inhibit"
                                else if tri_mode = "Homo Decouple"
                                then "tri.time_share obs.blank_inhibit"
                                else "";
begin
    initial_wait;
    relaxation_delay_calc;
    when irr_mode /= "Off" do
        on (IRR.GATE, IRR.PHS.0, IRR.ATN.irr_attenuator);
    end when;
    when tri_mode /= "Off" do
        on (TRI.GATE, TRI.PHS.0, TRI.ATN.tri_attenuator);
    end when;
    obs_presat_time( dante_presat, dante_loop, dante_pulse, dante_interval,
phase_dante, phs_shft, dante_attenuator, presat_time );
    when irr_mode /= "Off" do
        off IRR.GATE;
    end when;
    when tri_mode /= "Off" do
        off TRI.GATE;
    end when;
    when lock_hold do
        on LOCKHOLD;
    end when;
    1[us];
    x_pulse, (OBS.GATE, OBS.PHS.phase_1, OBS.ATN.x_atn);
    3[ms];
    parallel
        justify center
            grad_sl1, (FGZ.GATE, FGZ.AMP.grad_sl1_amp, FGZ.SHAPE."SQUARE");
        justify center
            obs_chpgs_m_pulse,          (OBS.GATE,          OBS.PHS.phase_2,
OBS.ATN.obs_chpgs_atn, OBS.SHAPE.{obs_chpgs_shape, "obs_chpGS"});
    end parallel;
    3[ms];
    grad_1, (FGZ.GATE, FGZ.SHAPE.grad_shape, FGZ.AMP.grad_1_amp);
    grad_recover;
    obs_sel_180,     (OBS.GATE,     OBS.PHS.phase_3,     OBS.ATN.obs_sel_atn,
OBS.LAMINAR.obs_sel_slp, OBS.SHAPE.obs_sel_shape);
    grad_1, (FGZ.GATE, FGZ.SHAPE.grad_shape, FGZ.AMP.grad_1_amp);
    grad_recover;
    3[ms];
    parallel
        justify center
            grad_sl2, (FGZ.GATE, FGZ.AMP.grad_sl2_amp, FGZ.SHAPE."SQUARE");
        justify center
            obs_chpgs_m_pulse,          (OBS.GATE,          OBS.PHS.phase_4,
OBS.ATN.obs_chpgs2_atn, OBS.SHAPE.{obs_chpgs_shape, "obs_chpGS2"});
    end parallel;
    3[ms];
    x_pulse, (OBS.GATE, OBS.PHS.phase_5, OBS.ATN.x_atn);
    on (FGZ.GATE, FGZ.AMP.grad_zqf1_amp);
    obs_chp1_m_pulse,   (OBS.GATE,   OBS.PHS.phase_6,   OBS.ATN.obs_chp1_atn,
OBS.SHAPE.{obs_chp_shape, "obs_chp1"});
    off (FGZ.GATE, FGZ.AMP.grad_zqf1_amp);
```

```
    grad_recover;
    x_pulse, (OBS.GATE, OBS.PHS.phase_7, OBS.ATN.x_atn);
    delta / 2;
    2 * x_pulse, (OBS.GATE, OBS.PHS.phase_8, OBS.ATN.x_atn);
    delta / 2;
    x_pulse, (OBS.GATE, OBS.PHS.phase_9, OBS.ATN.x_atn);
    delta / 2;
    2 * x_pulse, (OBS.GATE, OBS.PHS.phase_10, OBS.ATN.x_atn);
    delta / 2;
    x_pulse, (OBS.GATE, OBS.PHS.phase_11, OBS.ATN.x_atn);
    0.1[ms];
    on (FGZ.GATE, FGZ.AMP.grad_zqf2_amp);
    obs_chp2_m_pulse,  (OBS.GATE,  OBS.PHS.phase_12,  OBS.ATN.obs_chp2_atn,
OBS.SHAPE.{obs_chp_shape, "obs_chp2"});
    off (FGZ.GATE, FGZ.AMP.grad_zqf2_amp);
    0.1[ms];
    grad_2, (FGZ.GATE, FGZ.SHAPE.grad_shape, FGZ.AMP.grad_2_amp);
    grad_recover_zq;
    x_pulse, (OBS.GATE, OBS.PHS.phase_13, OBS.ATN.x_atn);
    acq( dead_time, delay, phase_acq );
    when lock_hold do
        off LOCKHOLD;
    end when;
end pulse;
```

Magnitude mode, GEMSTONE COSY experiment for JEOL ECZ/ECZL spectrometers using Delta 6.

```
------------------------------------------------------------------------
----
--                                                                    --
--                        Experiment Source Code                      --
--            Delta NMR Experiment & Machine Control Interface        --
```

```
--                                                                          --
--                    Copyright (c) 2021 JEOL Ltd                           --
--                          All Rights Reserved                             --
--                                                                          --
------------------------------------------------------------------------------
----
-- HELP.eng: GEMSTONE COSY
-- Category: 1D, COSY, GEMSTONE,liquid_advanced
-- File name : GEMSTONE_COSY
--
-- Sequence name : GEMSTONE COSY
--
-- Reference :
-- P. Kiraly et al, Angew. Chem. Int. Ed. 10.1002/anie.202011642
-- P. Kiraly et al, Chem. Commun. , 2021,57, 2368-2371
-- D. Taylor et al,
--
-- Parameters
-- x_pulse          : 90[deg] pulse width
-- x_atn            : attenuator of x_pulse
--
-- obs_sel_180      : 180[deg] selective pulse
-- obs_sel_offset   : offset for obs_sel_180
-- obs_sel_atn      : attenuator of obs_sel_180
-- obs_sel_shape    : pulse shape of obs_sel_180
--
--
-- relaxation_delay : inter-pulse delay
-- repetition_time  : pulse repetition_time (= relaxation_delay+x_acq_time)
--
--
-- dante_presat     : select TRUE or FALSE for obs_dante_presaturation.
-- irr_mode            : select irr_mode, "OFF","Presaturation" or "Homo
Decouple".
-- tri_mode            : select tri_mode, "OFF","Presaturation" or "Homo
Decouple".
-- presat_time      : duration of dante presaturation.
--
-- dante parameters
--
-- dante_pulse      : pulse width of dante presaturation
-- dante_interval   : pulse interval of dante presaturation
-- dante_attenuator : attenuator of dante presaturation
-- dante_loop       : real loop number/100
--
-- irr/tri decoupling parameters
--
-- irr_domain       : nucleus of irr presaturation or homo_decoupling
-- irr_offset       : offset of irr presaturation or homo_decoupling
-- irr_attenuator   : attenuator of irr presaturation or homo_decoupling
--
-- tri_domain       : nucleus of tri presaturation or homo_decoupling
-- tri_offset       : offset of tri presaturation or homo_decoupling
-- tri_attenuator   : attenuator of tri presaturation or homo_decoupling
--
-- Note :
-- scans = 16*n
--
-- x90-PFG1-sel180-PFG1-PFG2-sel180-PFG2-mix_time-acq
-- PFG1 not = PFG2
--
```

```
--
-- END HELP

header
    filename  => "GEMSTONE_COSY";
    sample_id => "";
    comment   => "GEMSTONE COSY";
    process   =  "proton_autophase.list";
    include "header";
end header;

instrument
    include "instrument";
    SPIN_STATE => "SPIN OFF";
end instrument;

acquisition
    x_domain         => "Proton";
    x_offset         => 5[ppm];
    x_sweep          => 18[ppm];
    include "x_points_default_1d_calc";
    x_points         => 16384;

    scans            => 32, help "scans = 4*n";
    x_prescans       => 4;
    mod_return       => 4;
    include "acquisition";
end acquisition;

pulse
    collect COMPLEX,OBS;

    obs_domain = x_domain;
    obs_offset = x_offset;
    obsfreq = _get_freq( obs_domain );

comment_1 =? "*** Pulse ***";
    x_pulse => x90, help "observe 90[deg] pulse";
    x_atn   =? xatn, help "attenuator for x_pulse";

    obs_pulse  = x_pulse;
    obs_atn    = x_atn;

comment_56 =? "*** GEMSTONE ***";
    obs_chpGS_shape      => "CHIRP", ad_shape_names, help "shape pulse";
    obs_chpGS_m_pulse   => 50[ms], help "ZQfilter pulse width";
    obs_chpGS_m_fsweep  => 2.5[kHz];
    obs_chpGS_sweep_ppm = (obs_chpGS_m_fsweep / obsfreq) * 1000000[ppm];
    obs_chpGS_chirp_smooth =  10[%];
    obs_chpGS_m_q0      => 11, help "q0 >= 11";
    obs_chpGS_m_ph_rev  = 1;
    obs_chpGS_b1_calc    = sqrt( obs_chpGS_m_q0 * obs_chpGS_m_fsweep / (2 *
pi * obs_chpGS_m_pulse) );
    obs_chpGS_m_b1max   => obs_chpGS_b1_calc;
    obs_chpGS_pw90      =  1 / (4 * obs_chpGS_m_b1max);
    obs_chpGS_atn_calc  = obs_atn + 20[dB] * log( obs_chpGS_pw90 / obs_pulse
);
    obs_chpGS_atn       => obs_chpGS_atn_calc;

      obs_chpGS2_m_fsweep     =  obs_chpGS_m_fsweep;
    obs_chpGS2_m_pulse      =  obs_chpGS_m_pulse;
```

```
    obs_chpGS2_chirp_smooth =  obs_chpGS_chirp_smooth;
    obs_chpGS2_m_q0        =  obs_chpGS_m_q0;
    obs_chpGS2_m_ph_rev     =  -1, (1,-1), help "phase reverse";
    obs_chpGS2_atn          =  obs_chpGS_atn_calc;

      grad_sl1           =? obs_chpGS_m_pulse;
    grad_sl1_amp       => 2.5[mT/m], help "Amplitude of gradient during 1st
chirp";
    grad_sl2           = obs_chpGS_m_pulse;
    grad_sl2_amp         =? -1.0*grad_sl1_amp, help "Amplitude of gradient
during 2nd chirp";
      grad_shape_type    => "SQUARE", fg_shape_names, help "gradient shape";


comment_32 =? "*** Selective Pulse 180deg  ***";
    obs_sel180_shape        => "RSNOB_180",std_shape_names, help "shape of
obs_sel_180";
    soft_shape_input    =  obs_sel180_shape;
    obs_sel_offset      =? x_offset, help "offset of obs_sel_180";
    soft_bw_input         => 100[Hz], help "bandwidth in [ppm] or [Hz] of
obs_sel_180";
    include "soft_pulse_calc";    -- input : soft_shape_input, soft_bw_input,
output : soft_angle, soft_pw_calc, soft_atn_calc, soft_bw_hz
    obs_sel_180         => soft_pw_calc, help "selective 180[deg] pulse";
    obs_sel180_atn       => soft_atn_calc, help "attenuator of obs_sel_180";
    obs_sel180_slp         =  obs_sel_offset;


comment_7 =? "*** Pulse Delay ***";
      initial_wait      =  1[s];
    j_hh => 10[Hz], help "1H-1H J constant ~ 8 - 12 [Hz]";
    include "relaxation_delay_1d_calc";
    relaxation_delay => relaxation_delay_default, help "inter-pulse delay";
    repetition_time       =?   relaxation_delay   +   x_acq_time,   help
"relaxation_delay+x_acq_time";

comment_41 =? "*** Zero-Quantum Dephasing ***";
    obs_chp_shape         =? "CHIRP", ad_shape_names, help "shape pulse";

    obs_chp1_m_pulse      => 50[ms], (30[ms],40[ms],50[ms]), help "ZQfilter
pulse width";
    obs_chp1_m_fsweep     =? if obs_chp1_m_pulse = 30[ms] then x_sweep * 2
                             else if obs_chp1_m_pulse = 40[ms] then x_sweep
* 2
                             else x_sweep * 2;
    obs_chp1_sweep_ppm   =? round( obs_chp1_m_fsweep * 1000000[ppm] / obsfreq
), help "obs_chp1_ad_sweep_ppm";
    obs_chp1_chirp_smooth =  10[%];
    obs_chp1_m_q0         =  5, help "q >= 5 On resonance adiabatic Quality
factor";
    obs_chp1_m_ph_rev     =  1, (1,-1), help "phase reverse";
    obs_chp1_b1_calc      =  sqrt( (obs_chp1_m_q0 * obs_chp1_m_fsweep) / (2
* pi * obs_chp1_m_pulse) );
    obs_chp1_m_b1max      =? obs_chp1_b1_calc;
    obs_chp1_pw90         =  1 / (4 * obs_chp1_m_b1max);
    obs_chp1_atn_calc      =   if obs_atn + 20[dB] * log( obs_chp1_pw90 /
obs_pulse ) < obs_atn + 6[dB] then obs_atn + 6[dB]
                             else obs_atn + 20[dB] * log( obs_chp1_pw90 /
obs_pulse );
    obs_chp1_atn          =? obs_chp1_atn_calc;
```

```
    obs_chp2_m_pulse      => 30[ms], (30[ms],40[ms],50[ms]), help "ZQfilter
pulse width";
    obs_chp2_m_fsweep     =? if obs_chp2_m_pulse = 30[ms] then x_sweep * 2
                             else if obs_chp2_m_pulse = 40[ms] then x_sweep
* 2
                             else x_sweep * 2;
    obs_chp2_sweep_ppm   =? round( obs_chp2_m_fsweep * 1000000[ppm] / obsfreq
), help "obs_chirp2_ad_sweep_ppm";
    obs_chp2_chirp_smooth =  10[%];
    obs_chp2_m_q0         =  5, help "q >= 5 On resonance adiabatic Quality
factor";
    obs_chp2_m_ph_rev     =  1, (1,-1), help "phase reverse" ;
    obs_chp2_b1_calc      =  sqrt( (obs_chp2_m_q0 * obs_chp2_m_fsweep) / (2
* pi * obs_chp2_m_pulse) );
    obs_chp2_m_b1max      =? obs_chp2_b1_calc;
    obs_chp2_pw90         =  1 / (4 * obs_chp2_m_b1max);
    obs_chp2_atn_calc    =  if obs_atn + 20[dB]*log(obs_chp2_pw90/obs_pulse)
<     obs_atn+6[dB]     then     obs_atn+6[dB]     else     obs_atn     +
20[dB]*log(obs_chp2_pw90/obs_pulse);
    obs_chp2_atn          =? obs_chp2_atn_calc;

    g_factor              =  1, help "obs_gamma/1H_gamma";
    coil_factor           =  1, help "coil_length/20mm";
    f_factor_zqf1              =   obs_chp1_m_fsweep  /  40[kHz],  help  "ref
10ppm*8/500MHz=40kHz";
    grad_zqf1_amp_calc   =  47[mT/m] * f_factor_zqf1 / coil_factor /g_factor,
help "gradient amplitude of grad_sl_ps";
    grad_zqf1_amp          =? grad_zqf1_amp_calc, help "Enter amplitude in
Tesla/meter units";
    f_factor_zqf2              =   obs_chp2_m_fsweep  /  40[kHz],  help  "ref
10ppm*8/500MHz=40kHz";
    grad_zqf2_amp_calc      =   47[mT/m]  *  f_factor_zqf2  /  coil_factor  /
g_factor, help "gradient amplitude of grad_sl_ps";
    grad_zqf2_amp          =? grad_zqf2_amp_calc, help "Enter amplitude in
Tesla/meter units";

comment_8 =? "*** Pulse Field Gradient ***";
    gradient_max      =? z_gradient_max, help "Maximum amplitude for a given
probe as defined in the probe file";

    grad_1       => 1[ms], help "pulse width of PFG1";
    grad_1_amp   => 0.06[T/m], help "amplitude of PFG1";

    grad_2       =? grad_1, help "pulse width of PFG2";
    grad_2_amp   => 0.09[T/m], help "amplitude of PFG2 (not = PFG1)";

    grad_shape    => "SINE", fg_shape_names, help "shape type of PFG pulse
PFG1,PFG2";
    grad_recover => 0.1[ms], help "gradient recovery time";

    grad_spoil       => 1[ms], help "duration of grad_1 during mixing";
    grad_spoil_amp  => 90[mT/m], help "Enter amplitude in Tesla/meter units";

    include "lockhold";
    include "obs_dante_presat";
    include "obs_presat_time";
    -- I think homo-decoupling should be removed, we can't use it with
antiphase signals observed
    include "irr_presat_homo_dec";
    include "tri_presat_homo_dec";
    include "presat_relaxation_delay_calc";
```

```
    include "pulse";

      phase_x      = {0,90};      -- Paul,what is this one
    phase_1      = {0,180};     --hard90 excitation
      phase_shape1 = {0};         --1st adiabaic pulse
      phase_shape2 = {0,0,90,90}; --rsnob
      phase_shape3 = {0};         --2nd adiabatic pulse
      phase_2      = {0};         --hard180 in COSY transfer
    phase_3      = {90};         --COSY read pulse
      phase_acq    = {0,180,180,0};


    module_config = if irr_mode = "Homo Decouple" then
                        if tri_mode = "Homo Decouple" then
                             "irr.time_sharetri.time_shareobs.blank_inhibit"
                        else "irr.time_share obs.blank_inhibit"
                    else if tri_mode = "Homo Decouple" then
                         "tri.time_share obs.blank_inhibit"
                    else "";
begin
    initial_wait;

    relaxation_delay_calc;

    --irr_presaturation block on
    when irr_mode /= "Off" do
        on (irr.gate, irr.phs.0, irr.atn.irr_attenuator);
    end when;
    --tri_presaturation block on
    when tri_mode /= "Off" do
        on (tri.gate, tri.phs.0, tri.atn.tri_attenuator);
    end when;

    obs_presat_time(dante_presat, dante_loop, dante_pulse, dante_interval,
                    phase_dante, phs_shft, dante_attenuator, presat_time);
    --irr presaturation block off
    when irr_mode /= "Off" do
        off (irr.gate);
    end when;
    --tri presaturation block off
    when tri_mode /= "Off" do
        off (tri.gate);
    end when;

    --lock_hold on
    when lock_hold do
        on(lockhold);
    end when;

    1[us];

      x_pulse, (obs.gate, obs.phs.phase_1, obs.atn.x_atn);

      3[ms];

    parallel
    --begin
      justify center
           grad_sl1, (fgz.gate, fgz.amp.grad_sl1_amp, fgz.shape."SQUARE");
    justify center
```

```
         obs_chpGS_m_pulse,         (obs.gate,         obs.phs.phase_shape1,
obs.atn.obs_chpGS_atn, obs.shape.{obs_chpGS_shape,"obs_chpGS"});
    end parallel;

      3[ms];

    grad_1, (fgz.gate, fgz.amp.grad_1_amp, fgz.shape.grad_shape_type);
    grad_recover;


    obs_sel_180, (obs.gate, obs.phs.phase_shape2, obs.atn.obs_sel180_atn,
                  obs.laminar.obs_sel180_slp, obs.shape.obs_sel180_shape);


    grad_1, (fgz.gate, fgz.amp.grad_1_amp, fgz.shape.grad_shape_type);
    grad_recover;

    3[ms];
    parallel
    --begin
      justify center
          grad_sl2, (fgz.gate, fgz.amp.grad_sl2_amp, fgz.shape."SQUARE");
    justify center
          obs_chpGS_m_pulse,         (obs.gate,         obs.phs.phase_shape3,
obs.atn.obs_chpGS2_atn, obs.shape.{obs_chpGS_shape,"obs_chpGS2"});
    end parallel;
    3[ms];


      1/(4*j_hh);
    x_pulse*2, (obs.gate, obs.phs.phase_2, obs.atn.x_atn);
    1/(4*j_hh);
    x_pulse, (obs.gate, obs.phs.phase_3, obs.atn.x_atn);

    acq( dead_time, delay, phase_acq );

    --lock_hold off
    when lock_hold do
        off(lockhold);
    end when;
end pulse;
```

GEMSTONE TOCSY experiment for JEOL ECZ/ECZL spectrometers using Delta 6.

```
--------------------------------------------------------------------------
----
--                                                                      --
--                        Experiment Source Code                        --
--            Delta NMR Experiment & Machine Control Interface          --
--                                                                      --
--                    Copyright (c) 2021 JEOL Ltd                       --
--                          All Rights Reserved                         --
--                                                                      --
--------------------------------------------------------------------------
----
-- HELP.eng: GEMSTONE TOCSY
-- Category: 1D, tocsy, GEMSTONE,liquid_advanced
-- File name : GEMSTONE_TOCSY
--
-- Sequence name : GEMSTONE TOCSY
```

```
--
-- Reference :
-- P. Kiraly et al, Angew. Chem. Int. Ed. 10.1002/anie.202011642
-- P. Kiraly et al, Chem. Commun. , 2021,57, 2368-2371
--
-- Parameters
-- x_pulse          : 90[deg] pulse width
-- x_atn            : attenuator of x_pulse
--
-- obs_sel_180      : 180[deg] selective pulse
-- obs_sel_offset   : offset for obs_sel_180
-- obs_sel_atn      : attenuator of obs_sel_180
-- obs_sel_shape    : pulse shape of obs_sel_180
--
-- x_spinlock_pulse : 90 deg pulse width of isotropic mixing sequence
-- x_spinlock_atn   : attenuator of x_pulse of x_spinlock_pulse
--
--  trim            : trim pulse width
--  mix_time        : mixting time of TOCSY
--
-- relaxation_delay : inter-pulse delay
-- repetition_time  : pulse repetition_time (= relaxation_delay+x_acq_time)
--
--
-- dante_presat     : select TRUE or FALSE for obs_dante_presaturation.
-- irr_mode            : select irr_mode, "OFF","Presaturation" or "Homo
Decouple".
-- tri_mode            : select tri_mode, "OFF","Presaturation" or "Homo
Decouple".
-- presat_time      : duration of dante presaturation.
--
-- dante parameters
--
-- dante_pulse      : pulse width of dante presaturation
-- dante_interval   : pulse interval of dante presaturation
-- dante_attenuator : attenuator of dante presaturation
-- dante_loop       : real loop number/100
--
-- irr/tri decoupling parameters
--
-- irr_domain       : nucleus of irr presaturation or homo_decoupling
-- irr_offset       : offset of irr presaturation or homo_decoupling
-- irr_attenuator   : attenuator of irr presaturation or homo_decoupling
--
-- tri_domain       : nucleus of tri presaturation or homo_decoupling
-- tri_offset       : offset of tri presaturation or homo_decoupling
-- tri_attenuator   : attenuator of tri presaturation or homo_decoupling
--
-- Note :
-- scans = 16*n
--
-- x90-PFG1-sel180-PFG1-PFG2-sel180-PFG2-mix_time-acq
-- PFG1 not = PFG2
--
--
--


header
    filename        => "GEMSTONE_TOCSY";
    sample_id       => "";
```

```
    comment          => "GEMSTONE TOCSY";
    process          =  "proton_autophase.list";
    auto_dwell       =  TRUE;
    auto_filter      =  TRUE;
    auto_gain        => FALSE;
    filter_limit     =  12;
    force_dual_mode => FALSE;
    decimation_rate =  0;
    force_tune       => FALSE;
    power_2          =  FALSE;
    round_points     =  FALSE;
    save_aborted     =  TRUE;
    vector           =  TRUE;
    mod_save         => 0;
end header;

instrument
    recvr_gain           => 50;
    get_90               =  "pulse_service::get_90_value";
    get_atn              =  "pulse_service::get_atn_value";
    get_freq             =  "pulse_service::get_freq_value";
    get_spin             =  "pulse_service::get_spin";
    get_gamma            =  "pulse_service::get_gamma";
    get_value_of_field   =  "pulse_service::get_value_of_field";
    get_probe_parameter =  "probe_service::get_probe_parameter";
    get_s_mparam         =  "pulse_service::get_s_mparam";
    get_d_mparam         =  "pulse_service::get_d_mparam";
    spin_state           => "SPIN OFF";
end instrument;

acquisition
    x_domain             => "Proton";
    x_offset             => 5[ppm];
    x_sweep              => 18[ppm];
    default_x_resolution = _get_value_of_field(x_domain, "x_1D_resolution");
    x_data_points        =  _get_value_of_field(x_domain, "x_data_points");
    x_points_default       =? if upper (x_sweep / default_x_resolution) >
x_data_points
                             then x_data_points
                             else upper (x_sweep / default_x_resolution);
    x_points             => 16384;
    scans                => 64, help "scans = 16*n";
    x_prescans           => 8;
    mod_return           => 8;
    x_acq_time           =? x_points / x_sweep;
    x_resolution         =? 1 / x_acq_time;
    x_sweep_input                =    lower  (x_sweep  *  transition_ratio  /
_get_freq(x_domain) * 1[Mppm]);
    x_points_input       =  lower (x_points * transition_ratio);
end acquisition;

pulse
    collect COMPLEX,OBS;

    macro dipsi2_rc( pulse_90, phase, sl_atn, tau ) is
        on OBS.ATN.sl_atn;
        pulse_90 * 180 / 90, (OBS.GATE, OBS.PHS.phase);
        tau;
        pulse_90 * 140 / 90, (OBS.GATE, OBS.PHS.phase);
        pulse_90 * 320 / 90, (OBS.GATE, OBS.PHS.phase + 180);
        tau;
```

```
pulse_90 * 90 / 90, (OBS.GATE, OBS.PHS.phase + 180);
pulse_90 * 270 / 90, (OBS.GATE, OBS.PHS.phase);
tau;
pulse_90 * 20 / 90, (OBS.GATE, OBS.PHS.phase);
pulse_90 * 200 / 90, (OBS.GATE, OBS.PHS.phase + 180);
tau;
pulse_90 * 85 / 90, (OBS.GATE, OBS.PHS.phase + 180);
pulse_90 * 30 / 90, (OBS.GATE, OBS.PHS.phase);
pulse_90 * 125 / 90, (OBS.GATE, OBS.PHS.phase + 180);
tau;
pulse_90 * 120 / 90, (OBS.GATE, OBS.PHS.phase + 180);
pulse_90 * 300 / 90, (OBS.GATE, OBS.PHS.phase);
tau;
pulse_90 * 75 / 90, (OBS.GATE, OBS.PHS.phase);
pulse_90 * 255 / 90, (OBS.GATE, OBS.PHS.phase + 180);
tau;
pulse_90 * 10 / 90, (OBS.GATE, OBS.PHS.phase + 180);
pulse_90 * 190 / 90, (OBS.GATE, OBS.PHS.phase);
tau;
pulse_90 * 180 / 90, (OBS.GATE, OBS.PHS.phase);
loop 2 times
    tau;
    pulse_90 * 180 / 90, (OBS.GATE, OBS.PHS.phase + 180);
    tau;
    pulse_90 * 140 / 90, (OBS.GATE, OBS.PHS.phase + 180);
    pulse_90 * 320 / 90, (OBS.GATE, OBS.PHS.phase);
    tau;
    pulse_90 * 90 / 90, (OBS.GATE, OBS.PHS.phase);
    pulse_90 * 270 / 90, (OBS.GATE, OBS.PHS.phase + 180);
    tau;
    pulse_90 * 20 / 90, (OBS.GATE, OBS.PHS.phase + 180);
    pulse_90 * 200 / 90, (OBS.GATE, OBS.PHS.phase);
    tau;
    pulse_90 * 85 / 90, (OBS.GATE, OBS.PHS.phase);
    pulse_90 * 30 / 90, (OBS.GATE, OBS.PHS.phase + 180);
    pulse_90 * 125 / 90, (OBS.GATE, OBS.PHS.phase);
    tau;
    pulse_90 * 120 / 90, (OBS.GATE, OBS.PHS.phase);
    pulse_90 * 300 / 90, (OBS.GATE, OBS.PHS.phase + 180);
    tau;
    pulse_90 * 75 / 90, (OBS.GATE, OBS.PHS.phase + 180);
    pulse_90 * 255 / 90, (OBS.GATE, OBS.PHS.phase);
    tau;
    pulse_90 * 10 / 90, (OBS.GATE, OBS.PHS.phase);
    pulse_90 * 190 / 90, (OBS.GATE, OBS.PHS.phase + 180);
    tau;
    pulse_90 * 180 / 90, (OBS.GATE, OBS.PHS.phase + 180);
end loop;
tau;
pulse_90 * 180 / 90, (OBS.GATE, OBS.PHS.phase);
tau;
pulse_90 * 140 / 90, (OBS.GATE, OBS.PHS.phase);
pulse_90 * 320 / 90, (OBS.GATE, OBS.PHS.phase + 180);
tau;
pulse_90 * 90 / 90, (OBS.GATE, OBS.PHS.phase + 180);
pulse_90 * 270 / 90, (OBS.GATE, OBS.PHS.phase);
tau;
pulse_90 * 20 / 90, (OBS.GATE, OBS.PHS.phase);
pulse_90 * 200 / 90, (OBS.GATE, OBS.PHS.phase + 180);
tau;
pulse_90 * 85 / 90, (OBS.GATE, OBS.PHS.phase + 180);
```

```
        pulse_90 * 30 / 90, (OBS.GATE, OBS.PHS.phase);
        pulse_90 * 125 / 90, (OBS.GATE, OBS.PHS.phase + 180);
        tau;
        pulse_90 * 120 / 90, (OBS.GATE, OBS.PHS.phase + 180);
        pulse_90 * 300 / 90, (OBS.GATE, OBS.PHS.phase);
        tau;
        pulse_90 * 75 / 90, (OBS.GATE, OBS.PHS.phase);
        pulse_90 * 255 / 90, (OBS.GATE, OBS.PHS.phase + 180);
        tau;
        pulse_90 * 10 / 90, (OBS.GATE, OBS.PHS.phase + 180);
        pulse_90 * 190 / 90, (OBS.GATE, OBS.PHS.phase);
        tau;
        pulse_90 * 180 / 90, (OBS.GATE, OBS.PHS.phase);
        tau;
        off OBS.ATN.sl_atn;
    end dipsi2_rc;

    macro   obs_presat_time(   dante_presat,   dante_loop,   dante_pulse,
dante_interval, phase_dante, phs_shft, dante_attenuator, presat_time ) is
        when dante_presat do
            loop 100 * dante_loop times
                dante_pulse, (OBS.GATE, OBS.PHS.phase_dante.lstep(phs_shft),
OBS.ATN.dante_attenuator);
                dante_interval;
            end loop;
        end when;
        when not dante_presat do
            presat_time;
        end when;
    end obs_presat_time;

    macro acq( dead_time, delay, phase ) is
        dead_time, OBS.PHS.0;
        delay, (OBS.PHS.0, RCV.GATE);
        acquire (OBS.PHS.0, RCV.PHS.phase);
    end acq;

    experiment               =  "GEMSTONE_TOCSY.jxp";
    obs_domain               =  x_domain;
    obs_offset               =  x_offset;
    obsfreq                  =  _get_freq(obs_domain);
    comment_1                =? "*** Pulse ***";
    x_pulse                  => x90, help "observe 90[deg] pulse";
    x_atn                    =? xatn, help "attenuator for x_pulse";
    obs_pulse                =  x_pulse;
    obs_atn                  =  x_atn;
    comment_56               =? "*** GEMSTONE ***";
    obs_chpgs_shape          => "CHIRP", ad_shape_names, help "shape pulse";
    obs_chpgs_m_pulse        => 50[ms], help "ZQfilter pulse width";
    obs_chpgs_m_fsweep       => 2.5[kHz];
    obs_chpgs_sweep_ppm      =  obs_chpgs_m_fsweep / obsfreq * 1[Mppm];
    obs_chpgs_chirp_smooth   =  10[%];
    obs_chpgs_m_q0           => 11, help "q0 >= 11";
    obs_chpgs_m_ph_rev       =  1;
    obs_chpgs_b1_calc        =  sqrt (obs_chpgs_m_q0 * obs_chpgs_m_fsweep /
(6.28319 * obs_chpgs_m_pulse));
    obs_chpgs_m_b1max        => obs_chpgs_b1_calc;
    obs_chpgs_pw90           =  1 / (4 * obs_chpgs_m_b1max);
    obs_chpgs_atn_calc         =  obs_atn + 20[dB] * log (obs_chpgs_pw90 /
obs_pulse);
    obs_chpgs_atn            => obs_chpgs_atn_calc;
```

```
    obs_chpgs2_m_fsweep      =  obs_chpgs_m_fsweep;
    obs_chpgs2_m_pulse       =  obs_chpgs_m_pulse;
    obs_chpgs2_chirp_smooth  =  obs_chpgs_chirp_smooth;
    obs_chpgs2_m_q0          =  obs_chpgs_m_q0;
    obs_chpgs2_m_ph_rev      =  -1, (1, -1), help "phase reverse";
    obs_chpgs2_atn           =  obs_chpgs_atn_calc;
    grad_sl1                 =? obs_chpgs_m_pulse;
    grad_sl1_amp           => 2.5[mT/m], help "Amplitude of gradient during
1st chirp";
    grad_sl2                 =  obs_chpgs_m_pulse;
    grad_sl2_amp           =? -1 * grad_sl1_amp, help "Amplitude of gradient
during 2nd chirp";
    grad_shape_type            => "SQUARE", fg_shape_names, help "gradient
shape";
    comment_32               =? "*** Selective Pulse 180deg  ***";
    obs_sel180_shape         => "RSNOB_180", std_shape_names, help "shape of
obs_sel_180";
    soft_shape_input         =  obs_sel180_shape;
    obs_sel_offset           =? x_offset, help "offset of obs_sel_180";
    soft_bw_input            => 100[Hz], help "bandwidth in [ppm] or [Hz] of
obs_sel_180";
    xfreq                    =  _get_freq(x_domain);
    soft_angle               =? _get_s_mparam(soft_shape_input, "m_angle",
90[deg]);
    int_soft_pc              =  _get_s_mparam(soft_shape_input, "m_integ",
100[%]);
    int_r_soft_pc               =  int_soft_pc / _get_s_mparam("gauss_90",
"m_integ", 100[%]);
    unit_soft_pc             =  if soft_bw_input * 0 = 0[ppm]
                                then 1
                                else if soft_bw_input * 0 = 0[Hz]
                                then 2
                                else 3;
    soft_bw_hz               =  if unit_soft_pc = 1
                                then soft_bw_input * xfreq / 1[Mppm]
                                else if unit_soft_pc = 2
                                then soft_bw_input
                                else 1[Hz];
    type_soft_pc              =  _get_s_mparam(soft_shape_input, "m_type",
"std");
    is_std_pc                 =  if type_soft_pc = "std" and unit_soft_pc <
3
                                then TRUE
                                else FALSE;
    soft_pw_calc             =? if is_std_pc
                                then  _get_s_mparam(soft_shape_input,  "bw",
1) / soft_bw_hz
                                else 1[us];
    soft_atn_calc            =? if is_std_pc
                                then xatn_soft + 20[dB] * log (soft_pw_calc
/ x90_soft * 90[deg] / soft_angle * int_r_soft_pc)
                                else 79[dB];
    obs_sel_180                 => soft_pw_calc, help "selective 180[deg]
pulse";
    obs_sel180_atn        => soft_atn_calc, help "attenuator of obs_sel_180";
    obs_sel180_slp           =  obs_sel_offset;
    comment_12               =? "*** TOCSY mixing time ***";
    x_spinlock_pulse        => x90_spin, help "90deg pulse width of isotropic
mixing sequence(clean-DIPSI2)";
    x_spinlock_atn                    =>  xatn_spin,  help  "attenuator  of
x_spinlock_pulse";
```

```
    tau_interval                => 10[us], help "tau interval of isotropic
mixing sequence(clean-DIPSI2) about 10-20use";
    mix_time                    => 60[ms], help "mixing time of TOCSY(clean-
dipsi2)";
    mix_time_loop             =? round (mix_time / (x_spinlock_pulse * 10240
/ 90 + 36 * tau_interval));
    total_mix_time               =? (x_spinlock_pulse * 10240 / 90 + 36 *
tau_interval) * mix_time_loop;
    comment_7                 =? "*** Pulse Delay ***";
    initial_wait              =  1[s];
    rpt_min                   =  x_points * filter_limit * 3.64[us] + 10[ms];
    rpt_def_temp              =  if x_domain = "Proton"
                                 then 7[s]
                                 else if x_domain = "Fluorine19"
                                 then 7[s]
                                 else if x_domain = "Carbon13"
                                 then 3[s]
                                 else 5[s];
    rd_def_temp               =  if rpt_def_temp - x_acq_time > 0.1[s]
                                 then rpt_def_temp - x_acq_time
                                 else 0.1[s];
    relaxation_delay_default =  if rpt_def_temp > rpt_min
                                 then rd_def_temp
                                 else if rpt_min - x_acq_time > 0[s]
                                 then rpt_min - x_acq_time
                                 else 0.1[s];
    relaxation_delay          => 1[s], help "inter-pulse delay";
    repetition_time              =? relaxation_delay + x_acq_time, help
"relaxation_delay+x_acq_time";
    comment_41                =? "*** Zero-Quantum Dephasing ***";
    obs_chp_shape             =? "CHIRP", ad_shape_names, help "shape pulse";
    obs_chp1_m_pulse             => 50[ms], (30[ms], 40[ms], 50[ms]), help
"ZQfilter pulse width";
    obs_chp1_m_fsweep         =? if obs_chp1_m_pulse = 30[ms]
                                 then x_sweep * 2
                                 else if obs_chp1_m_pulse = 40[ms]
                                 then x_sweep * 2
                                 else x_sweep * 2;
    obs_chp1_sweep_ppm      =? round (obs_chp1_m_fsweep * 1[Mppm] / obsfreq),
help "obs_chp1_ad_sweep_ppm";
    obs_chp1_chirp_smooth     =  10[%];
    obs_chp1_m_q0                =   5, help "q >= 5 On resonance adiabatic
Quality factor";
    obs_chp1_m_ph_rev         =  1, (1, -1), help "phase reverse";
    obs_chp1_b1_calc           =  sqrt (obs_chp1_m_q0 * obs_chp1_m_fsweep /
(6.28319 * obs_chp1_m_pulse));
    obs_chp1_m_b1max          =? obs_chp1_b1_calc;
    obs_chp1_pw90             =  1 / (4 * obs_chp1_m_b1max);
    obs_chp1_atn_calc          =  if obs_atn + 20[dB] * log (obs_chp1_pw90 /
obs_pulse) < obs_atn + 6[dB]
                                 then obs_atn + 6[dB]
                                 else obs_atn + 20[dB] * log (obs_chp1_pw90
/ obs_pulse);
    obs_chp1_atn              =? obs_chp1_atn_calc;
    obs_chp2_m_pulse             => 30[ms], (30[ms], 40[ms], 50[ms]), help
"ZQfilter pulse width";
    obs_chp2_m_fsweep         =? if obs_chp2_m_pulse = 30[ms]
                                 then x_sweep * 2
                                 else if obs_chp2_m_pulse = 40[ms]
                                 then x_sweep * 2
                                 else x_sweep * 2;
```

```
    obs_chp2_sweep_ppm      =? round (obs_chp2_m_fsweep * 1[Mppm] / obsfreq),
help "obs_chirp2_ad_sweep_ppm";
    obs_chp2_chirp_smooth    =  10[%];
    obs_chp2_m_q0               =   5, help "q >= 5 On resonance adiabatic
Quality factor";
    obs_chp2_m_ph_rev        =  1, (1, -1), help "phase reverse";
    obs_chp2_b1_calc          =  sqrt (obs_chp2_m_q0 * obs_chp2_m_fsweep /
(6.28319 * obs_chp2_m_pulse));
    obs_chp2_m_b1max         =? obs_chp2_b1_calc;
    obs_chp2_pw90            =  1 / (4 * obs_chp2_m_b1max);
    obs_chp2_atn_calc         =  if obs_atn + 20[dB] * log (obs_chp2_pw90 /
obs_pulse) < obs_atn + 6[dB]
                               then obs_atn + 6[dB]
                               else obs_atn + 20[dB] * log (obs_chp2_pw90
/ obs_pulse);
    obs_chp2_atn             =? obs_chp2_atn_calc;
    g_factor                 =  1, help "obs_gamma/1H_gamma";
    coil_factor              =  1, help "coil_length/20mm";
    f_factor_zqf1               =   obs_chp1_m_fsweep / 40[kHz], help "ref
10ppm*8/500MHz=40kHz";
    grad_zqf1_amp_calc        =   47[mT/m] * f_factor_zqf1 / coil_factor /
g_factor, help "gradient amplitude of grad_sl_ps";
    grad_zqf1_amp           =? grad_zqf1_amp_calc, help "Enter amplitude in
Tesla/meter units";
    f_factor_zqf2               =   obs_chp2_m_fsweep / 40[kHz], help "ref
10ppm*8/500MHz=40kHz";
    grad_zqf2_amp_calc        =   47[mT/m] * f_factor_zqf2 / coil_factor /
g_factor, help "gradient amplitude of grad_sl_ps";
    grad_zqf2_amp           =? grad_zqf2_amp_calc, help "Enter amplitude in
Tesla/meter units";
    comment_8                =? "*** Pulse Field Gradient ***";
    gradient_max             =? z_gradient_max, help "Maximum amplitude for
a given probe as defined in the probe file";
    grad_1                   => 1[ms], help "pulse width of PFG1";
    grad_1_amp               => 60[mT/m], help "amplitude of PFG1";
    grad_2                   =? grad_1, help "pulse width of PFG2";
    grad_2_amp                 => 90[mT/m], help "amplitude of PFG2 (not =
PFG1)";
    grad_shape                => "SQUARE", fg_shape_names, help "shape type
of PFG pulse PFG1,PFG2";
    grad_recover             => 2[ms], help "gradient recovery time";
    grad_spoil                  => 1[ms], help "duration of grad_1 during
mixing";
    grad_spoil_amp         => 90[mT/m], help "Enter amplitude in Tesla/meter
units";
    comment_900              =? "*** lock hold ***";
    lock_hold                 => FALSE, help "select TRUE or FALSE for lock
hold";
    comment_201              =? "*** obs_dante_presaturation ***";
    dante_presat                          =>  FALSE,  help  "True/False  for
obs_dante_presaturation";
    when dante_presat do
        dante_offset                  => x_offset, help "offset of dante
presaturation";
        dante_pulse                  => 4[us], help "pulse width of dante
presaturation";
        dante_interval          => 0.1[ms] - dante_pulse, help "interval of
dante pulse train";
        dante90_temp             =  x90;
        danteatn_temp            =  xatn;
        dante_temp           =  dante_pulse / (dante_pulse + dante_interval);
```

```
        dante_atn_default        =? danteatn_temp + 20[dB] * log (8.3[ms] *
dante_temp / dante90_temp), help "calculate a value of B1=30Hz";
        dante_attenuator           => dante_atn_default, 10[dB]->119[dB] :
10[mdB], help "attenuator of dante presaturation";
        proton_freq              =  _get_freq(x_domain);
        diff_offset              =  (dante_offset - x_offset) * 1[Mppm-1] *
proton_freq;
        phs_shft             =  diff_offset * (dante_pulse + dante_interval)
* 360;
        phase_dante              =  {0};
    end when;
    comment_202            =? "*** irr_ presaturation & homo spin decoupling
***";
    irr_mode                      => "Off", ("Off", "Presaturation", "Homo
Decouple"), help "Select Presaturation or Homo Decouple";
    when irr_mode = "Homo Decouple" do
        note_irr_homodec         =? "Set [filter_limit = 2-4] in the Header
parameters";
    end when;
    when irr_mode /= "Off" do
        irr_domain           =? x_domain, help "nucleus of irr presaturation
or homo_decoupling";
        irr_offset             => 5[ppm], help "offset of irr presaturation
or homo_decoupling";
        irr90_temp              =  x90;
        irratn_temp             =  xatn;
        irr_atn_default           =? irratn_temp + 20[dB] * log (8.3[ms] /
irr90_temp), help "calculate a value of B1=30Hz";
        irr_attenuator               => irr_atn_default, 10[dB]->119[dB] :
10[mdB], help "attenuator of irr presaturation or homo_decoupling";
    end when;
    comment_203            =? "*** tri_ presaturation & homo spin decoupling
***";
    tri_mode                      => "Off", ("Off", "Presaturation", "Homo
Decouple"), help "Select Presaturation or Homo Decouple";
    when tri_mode = "Homo Decouple" do
        note_tri_homodec         =? "Set [filter_limit = 2-4] in the Header
parameters";
    end when;
    when tri_mode /= "Off" do
        tri_domain           =? x_domain, help "nucleus of tri presaturation
or homo_decoupling";
        tri_offset             => 5[ppm], help "offset of tri presaturation
or homo_decoupling";
        tri90_temp              =  x90;
        triatn_temp             =  xatn;
        tri_atn_default           =? triatn_temp + 20[dB] * log (8.3[ms] /
tri90_temp), help "calc.value of B1=30Hz";
        tri_attenuator               => tri_atn_default, 10[dB]->119[dB] :
10[mdB], help "attenuator of tri presaturation or homo_decoupling";
    end when;
    comment_111             =? "*** presat_time ***";
    presat_time_flag        => FALSE, help "select TRUE or FALSE for using
presat_time";
    when presat_time_flag do
        presat_time_temp       => relaxation_delay, help "presaturation_time
<= relaxation_delay";
    end when;
    presat_time             =? if relaxation_delay > presat_time_temp
                               then presat_time_temp
                               else relaxation_delay;
```

```
    relaxation_delay_calc    =  relaxation_delay - presat_time;
    dante_loop                =  lower (presat_time / (100 * (dante_pulse +
dante_interval))), help "real loop number/100";
    dead_time                =  if not auto_dwell
                                then 0.2 / x_sweep
                                else 0[us];
    delay                    =  if not auto_dwell
                                then 0.8 / x_sweep
                                else 0[us];
    phase_x                  =  {0, 90};
    phase_1                  =  {0};
    phase_shape1             =  {4(0), 4(90)};
    phase_shape2             =  {0, 90};
    phase_shape3             =  {2(0), 2(90)};
    phase_lock               =  {0, 90};
    phase_acq                =  {180, 2(0), 180, 0, 2(180), 0};
    module_config            =  if irr_mode = "Homo Decouple"
                                then if tri_mode = "Homo Decouple"
                                     then   "irr.time_share   tri.time_share
obs.blank_inhibit"
                                     else "irr.time_share obs.blank_inhibit"
                                else if tri_mode = "Homo Decouple"
                                then "tri.time_share obs.blank_inhibit"
                                else "";
begin
    initial_wait;
    relaxation_delay_calc;
    when irr_mode /= "Off" do
        on (IRR.GATE, IRR.PHS.0, IRR.ATN.irr_attenuator);
    end when;
    when tri_mode /= "Off" do
        on (TRI.GATE, TRI.PHS.0, TRI.ATN.tri_attenuator);
    end when;
    obs_presat_time( dante_presat, dante_loop, dante_pulse, dante_interval,
phase_dante, phs_shft, dante_attenuator, presat_time );
    when irr_mode /= "Off" do
        off IRR.GATE;
    end when;
    when tri_mode /= "Off" do
        off TRI.GATE;
    end when;
    when lock_hold do
        on LOCKHOLD;
    end when;
    1[us];
    x_pulse, (OBS.GATE, OBS.PHS.phase_1, OBS.ATN.x_atn);
    3[ms];
    parallel
        justify center
            grad_sl1, (FGZ.GATE, FGZ.AMP.grad_sl1_amp, FGZ.SHAPE."SQUARE");
        justify center
            obs_chpgs_m_pulse,       (OBS.GATE,       OBS.PHS.phase_shape1,
OBS.ATN.obs_chpgs_atn, OBS.SHAPE.{obs_chpgs_shape, "obs_chpGS"});
    end parallel;
    3[ms];
    grad_1, (FGZ.GATE, FGZ.AMP.grad_1_amp, FGZ.SHAPE.grad_shape_type);
    grad_recover;
    obs_sel_180,  (OBS.GATE,  OBS.PHS.phase_shape2,  OBS.ATN.obs_sel180_atn,
OBS.LAMINAR.obs_sel180_slp, OBS.SHAPE.obs_sel180_shape);
    grad_1, (FGZ.GATE, FGZ.AMP.grad_1_amp, FGZ.SHAPE.grad_shape_type);
    grad_recover;
```

```
    3[ms];
    parallel
        justify center
            grad_sl2, (FGZ.GATE, FGZ.AMP.grad_sl2_amp, FGZ.SHAPE."SQUARE");
        justify center
            obs_chpgs_m_pulse,       (OBS.GATE,       OBS.PHS.phase_shape3,
OBS.ATN.obs_chpgs2_atn, OBS.SHAPE.{obs_chpgs_shape, "obs_chpGS2"});
    end parallel;
    3[ms];
    x_pulse, (OBS.GATE, OBS.PHS.phase_1, OBS.ATN.x_atn);
    3[ms];
    on (FGZ.GATE, FGZ.AMP.grad_zqf1_amp);
    obs_chp1_m_pulse,   (OBS.GATE,   OBS.PHS.phase_1,   OBS.ATN.obs_chp1_atn,
OBS.SHAPE.{obs_chp_shape, "obs_chp1"});
    off (FGZ.GATE, FGZ.AMP.grad_zqf1_amp);
    3[ms];
    loop mix_time_loop times
        dipsi2_rc( x_spinlock_pulse, phase_lock, x_spinlock_atn, tau_interval
);
    end loop;
    3[ms];
    grad_spoil, (FGZ.GATE, FGZ.AMP.grad_spoil_amp);
    grad_recover;
    on (FGZ.GATE, FGZ.AMP.grad_zqf2_amp);
    obs_chp2_m_pulse,   (OBS.GATE,   OBS.PHS.phase_1,   OBS.ATN.obs_chp2_atn,
OBS.SHAPE.{obs_chp_shape, "obs_chp2"});
    off (FGZ.GATE, FGZ.AMP.grad_zqf2_amp);
    grad_recover;
    x_pulse, (OBS.GATE, OBS.PHS.phase_1, OBS.ATN.x_atn);
    acq( dead_time, delay, phase_acq );
    when lock_hold do
        off LOCKHOLD;
    end when;
end pulse;
```

Band-selective version of the interferogram Zangger-Sterk pure shift NMR experiment for JEOL ECZ/ECZL spectrometers using Delta 6.

```
------------------------------------------------------------------------
----
--                                                                      --
--                        Experiment Source Code                        --
--            Delta NMR Experiment & Machine Control Interface          --
--                                                                      --
--                   Copyright (c) 2021 JEOL Ltd                        --
--                          All Rights Reserved                         --
--                                                                      --
------------------------------------------------------------------------
----
-- HELP.eng: pure shift sequence (homonuclear broadband-decoupling )
-- File name : pureshift_1d_bandselective
-- Category: 1D, liquid_advanced, pureshift
-- Sequence name : pureshift 1d band-selective method
--
-- Reference :
--
-- K. Zangger, H. Sterk, J. Magn. Reson. 124, 486-489 (1997)
-- J. A. Aguilar, S. Faulkner, M. Nilsson, G. A. Morris, Angew. Chem. Int.
Ed. 49, 3901-3903 (2010)
-- Laura Castanar , Magn.Reson.Chem.55,46-53(2017)
--
-- Parameters
-- x_atn                : attenuator of x_pulse
-- x_pulse              : pulse width
--
-- relaxation_delay     : inter-pulse delay
```

```
--    repetition_time                    :    pulse    repetition_time    (=
relaxation_delay+x_acq_time)
--
-- dante_presat         : select TRUE or FALSE for obs_dante_presaturation.
-- presat_time          : duration of dante presaturation.
--
-- dante parameters
-- dante_pulse          : pulse width of dante presaturation
-- dante_interval       : pulse interval of dante presaturation
-- dante_attenuator     : attenuator of dante presaturation
-- dante_loop           : real loop number/100
--
-- Note :
--  scans = 1*n
--
-- END HELP
--
header
    filename        => "pureshift_1d_bandselective";
    sample_id       => "";
    comment         => "Band-selective 1D pureshift";
    process         =  "pureshift_1d.list";
    include "header";
    auto_filter     =  FALSE;
end header;

instrument
    include "instrument";
    spin_state  => "SPIN OFF";
end instrument;

acquisition
    x_domain               => "Proton";
    x_offset               => 5[ppm];
    spectrum_width         => 20[ppm];
    x_sweep                =  spectrum_width*4;
    x_acq_time             => 20[ms];
    J_calc                 =? 1/(2*x_acq_time);
    x_points               =? upper (x_sweep * x_acq_time);
    x_resolution           =? x_sweep/x_points;
      actual_x_acq_time    =? 1.0/x_resolution;
    scans                  => 4, help "scans = 4*n";
    x_prescans             => 4;
    mod_return             => 4;

    y_points               => 32;

    total_acq_time         =? x_acq_time*y_points;
    pureshift_x_points     =? x_points*y_points ;
    pureshift_x_resolution =? x_sweep/pureshift_x_points;

end acquisition;

pulse
    collect COMPLEX,OBS REAL;

    obs_domain = x_domain;
    obs_offset = x_offset;
    obsfreq = _get_freq( obs_domain );

comment_1 =? "*** Pulse ***";
```

```
    scramble  =  1[ms];

    x_pulse   => x90, help "set 90deg pulse width";
    x_atn     =? xatn, help "attenuator for x_pulse";

    obs_pulse =  x_pulse;
    obs_atn   =  x_atn;

comment_32 =? "*** Selective Pulse 180deg  ***";
    obs_sel_shape          => "RSNOB_180",std_shape_names, help "shape of
obs_sel_180";
    soft_shape_input    =  obs_sel_shape;
    obs_sel_offset      => x_offset, help "offset of obs_sel_180";
    soft_bw_input         => 100[Hz], help "bandwidth in [ppm] or [Hz] of
obs_sel_180";
    include "soft_pulse_calc";   -- input : soft_shape_input, soft_bw_input,
output : soft_angle, soft_pw_calc, soft_atn_calc, soft_bw_hz
    obs_sel_180         => soft_pw_calc, help "selective 180[deg] pulse";
    obs_sel_atn         => soft_atn_calc, help "attenuator of obs_sel_180";
    obs_sel_slp         =  obs_sel_offset;

comment_7 =? "*** Pulse Delay ***";
    initial_wait      =  1[s];
    t1                =  1.0[us];
    relaxation_delay  => 1[s], help "inter-pulse delay";
    repetition_time          =?   relaxation_delay   +   x_acq_time,   help
"relaxation_delay+x_acq_time";

comment_8 =? "*** Pulse Field Gradient ***";
    gradient_max           =? z_gradient_max, help "Maximum amplitude for a
given probe as defined in the probe file";

    grad_1_ps             => 1[ms], help "duration of grad_1_ps";
    grad_1_ps_amp       => 0.10[T/m], help "gradient amplitude of grad_1_ps";

    grad_2_ps             => grad_1_ps, help "duration of grad_2_ps";
    grad_2_ps_amp       => -0.05[T/m], help "gradient amplitude of grad_2_ps";

    grad_3_ps             => grad_1_ps, help "duration of grad_2_ps";
    grad_3_ps_amp       => -0.15[T/m], help "gradient amplitude of grad_2_ps";

    g_factor             = 1;
    coil_factor          = 1;
    f_factor_ps          = (x_sweep*0.25)/5000[Hz];
    grad_sl_ps_amp_calc  =?  6[mT/m]*f_factor_ps/coil_factor/g_factor,  help
"gradient amplitude of grad_sl_ps";
    grad_sl_ps_amp       => 0[T/m], help "gradient amplitude of grad_sl_ps";
    delta_SN             =? soft_bw_hz/(x_sweep*0.25), help "about0.05";

    grad_shape_type_ps   => "SINE", fg_shape_names, help "gradient shape";
    grad_recover_ps      => 0.5[ms], help "gradient recovery time";

    include "lockhold";
    include "obs_dante_presat";
    include "obs_presat_time";

    include "irr_presat_homo_dec";
    irr_mode => "Off", ("Off", "Presaturation"), help "Select Presaturation
or Homo Decouple";

    include "tri_presat_homo_dec";
```

```
    tri_mode => "Off", ("Off", "Presaturation"), help "Select Presaturation
or Homo Decouple";

    include "presat_relaxation_delay_calc";
    include "pulse";

    phase_1      = {2(0, 180), 2(90, 270)};
    phase_2      = {16(0)};
    phase_shape1 = {2(0, 0, 90, 90), 2(180, 180, 270, 270)};
    phase_acq    = {2(0, 180, 180, 0, 90, 270, 270, 90)};

begin
    initial_wait;

    1[us];
    scramble,   (obs.gate, obs.phs.0, obs.atn.x_atn + 6[dB]);
    scramble*2, (obs.gate, obs.phs.90, obs.atn.x_atn + 6[dB]);

    relaxation_delay_calc;

    --irr_presaturation block on
    when irr_mode /= "Off" do
        on (irr.gate, irr.phs.0, irr.atn.irr_attenuator);
    end when;
    --tri_presaturation block on
    when tri_mode /= "Off" do
        on (tri.gate, tri.phs.0, tri.atn.tri_attenuator);
    end when;
    obs_presat_time(dante_presat, dante_loop, dante_pulse, dante_interval,
                    phase_dante, phs_shft, dante_attenuator, presat_time);
    --irr presaturation block off
    when irr_mode /= "Off" do
        off (irr.gate);
    end when;
    --tri presaturation block off
    when tri_mode /= "Off" do
        off (tri.gate);
    end when;

    1[us];

    --lock_hold on
    when lock_hold do
        on(lockhold);
    end when;

    x_pulse, (obs.gate, obs.phs.phase_1, obs.atn.x_atn);

    t1 ystep x_acq_time/2 ;

      actual_x_acq_time/4 - grad_1_ps - grad_recover_ps;

    grad_1_ps,(fgz.gate,fgz.amp.grad_1_ps_amp,fgz.shape.grad_shape_type_ps);
    grad_recover_ps;
    x_pulse*2, (obs.gate, obs.phs.phase_2, obs.atn.x_atn);

      actual_x_acq_time/4 - grad_1_ps - grad_recover_ps;

    grad_1_ps,(fgz.gate,fgz.amp.grad_1_ps_amp,fgz.shape.grad_shape_type_ps);
    grad_recover_ps;
```

```
    grad_2_ps,(fgz.gate,fgz.amp.grad_2_ps_amp,fgz.shape.grad_shape_type_ps);
    grad_recover_ps;
    on (fgz.gate, fgz.amp.grad_sl_ps_amp );
    obs_sel_180, (obs.gate, obs.phs.phase_shape1, obs.atn.obs_sel_atn,
                  obs.laminar.obs_sel_slp, obs.shape.obs_sel_shape);
    off (fgz.gate, fgz.amp.grad_sl_ps_amp );
    grad_2_ps,(fgz.gate,fgz.amp.grad_2_ps_amp,fgz.shape.grad_shape_type_ps);
    grad_recover_ps;

    t1 ystep x_acq_time/2;

    acq( dead_time, delay, phase_acq );

    --lock_hold off
    when lock_hold do
        off(lockhold);
    end when;
end pulse;
```

S7. References

1 P. Kiraly, N. Kern, M. P. Plesniak, M. Nilsson, D. J. Procter, G. A. Morris and R. W. Adams, *Angew. Chem., Int. Ed.*, 2021, **60**, 666–669.

2 M. R. M. Koos, G. Kummerlöwe, L. Kaltschnee, C. M. Thiele and B. Luy, *Angew. Chem., Int. Ed.*, 2016, **55**, 7655–7659.

3 H. J. Hogben, M. Krzystyniak, G. T. P. Charnock, P. J. Hore and I. Kuprov, *J. Magn. Reson.*, 2011, **208**, 179–194.

4 H. Geen and R. Freeman, *J. Magn. Reson.*, 1991, **93**, 93–141.

5 Ē. Kupče, J. Boyd and I. D. Campbell, *J. Magn. Reson. Ser. B*, 1995, 106, 300–303.

6 C. Bauer, R. Freeman, T. Frenkiel, J. Keeler and A. J. Shaka, *J. Magn. Reson.*, 1984, **58**, 442–457.

7 Y. Xia, G. Legge, K. Y. Jun, Y. Qi, H. Lee and X. Gao, *Magn. Reson. Chem.*, 2005, **43**, 372–379.