Supporting Information

Triplet NOAH supersequences optimised for small molecule structure characterisation

Tim D. W. Claridge, Maksim Mayzel, and Ēriks Kupče

Figure S1: Comparison of NOAH3-BSX spectra *versus* conventional data sets of andrographolide recorded with similar acquisition parameters. Left-hand traces derive from individual experiments and right-hand traces from NOAH modules. The corresponding spectra contain comparable information. The conventional DQF-COSY experiment shows artefacts typically associated with “rapid pulsing” (marked by dashed line) which are attenuated in the NOAH module at the expense of some residual artefacts arising from the intense methyl singlets (boxed).

Figure S2: Andrographolide NOAH-3 BSN and BSR° data: NOESY (left) versus ROESY (right) data at 700 MHz (upper) and 500 MHz (lower).

Table S1: Comparison of the time requirements for NOAH-2 experiments and NOAH-3 BSX experiments *versus* the total time for the separate correlation experiments. All data sets were collected as 1K*256 points (per individual module) across 7 ppm (^1H) and 170 ppm (^{13}C) windows, acquiring 2 transients per increment with a recovery delay (d1) of 1.5 s. Coupling constants were assumed to be 145 Hz ($^1J_{\text{CH}}$) and 8 Hz ($^{2/3}J_{\text{CH}}$) with mixing times as indicated in the table. For CLIP-COSY, J_{HH} was set to a compromise value of 30 Hz.

Experiment	Module X In BSX	Experiment Duration	Individual experiment summed duration	NOAH time relative to individuals
HMBC	-	14m 36s	-	-
HSQC	-	14m 20s	-	-
COSY	-	14m 12s	-	-
CLIP-COSY	-	14m 46s	-	-
DQF-COSY	-	14m 33s	-	-
TOCSY (80 ms)	-	14m 54s	-	-
NOESY (400 ms)	-	17m 23s	-	-
ROESY (300 ms)	-	17m 11s	-	-
NOAH-2 SC		15m 35s	28m 32s	0.55
NOAH-2 BS		16m 6s	28m 56s	0.56
NOAH-3 BSC	COSY	17m 21s	43m 8s	0.40
NOAH-3 BSC ^c	CLIP-COSY	17m 48s	43m 42s	0.41
NOAH-3 BSD	DQF-COSY	17m 16s	43m 29s	0.40
NOAH-3 BST	TOCSY (80 ms)	18m 13s	43m 50s	0.42
NOAH-3 BSN	NOESY (400 ms)	20m 40s	46m 19s	0.45
NOAH-3 BSR ^o	ROESY (300 ms)	19m 48s	46m 7s	0.43

Fig S3: A comparison of ROESY spectra from NOAH-3 BSR experiments employing with either the phase-alternating spin-lock (left) or the off-resonance adiabatic sweeps (right) for ROE generation. The corresponding NOAH module is shown above each spectrum. Positive cross peaks (black) arise from exchange between hydroxyl protons and water. Data were collected at 500 MHz using a room temperature TBO probe for andrographolide (38 mM, DMSO). Both NOAH data sets were collected with 2 transients per increment with mixing times of 300 ms. The phase-alternating sequence comprised 694 cycles of pairs of 216 μ s 180° pulses, whilst the high and low offset off-resonance pulses were each 150 ms amplitude-modulated WURST sweeps with $B_1(\text{max})$ of 5 kHz applied at 9.5 and -0.5 ppm respectively.

Fig S4: A comparison of signal intensities in spectrum projections extracted from conventional spectra (lower traces) vs the NOAH BSX modules (upper traces) of andrographolide (38 mM, DMSO, 700 MHz TXI probe). All spectra were collected with 2 transients per increment.

Non-Uniform Sampling (NUS) in NOAH

The NOAH supersequences are written to be compatible with NUS. The homonuclear modules of the NOAH-3 BSX sequences employ either echo-antiecho or States F1 quadrature detection for NUS compatibility.

Currently the standard Bruker NUS implementation in TOPSPIN cannot be used directly with the supersequences and NUS sampling must be defined explicitly within the sequence. A python script (`noah_nus.py`) is available that will convert a NOAH experiment into the NUS variant and will define the necessary parameters for executing NUS. The modified pulse program is stored with a `.nus` extension and may be used for future NUS data collection. The sampling itself is defined within the pulse program using the following definitions for `d0` (heteronuclear t_1) and `d10` (homonuclear t_1) periods.

```
define list<loopcounter> t1list=<$VCLIST> ;define NUS list
"d0=in0*t1list+3u" ; calculate t1 intervals explicitly
"d10=in10*t1list+3u"
lm t1list.inc ;increment t1 list pointer on looping
```

The VCLIST (named "noah") contains the sampling intervals and is generated automatically by the python script according to the requested NusAMOUNT.

The user therefore follows the following simple protocol to set up the NUS experiment:

1. Load the appropriate NOAH experiment parameter set
2. Define the % NUS sampling required in the usual manner (NusAMOUNT [%])
3. Execute the python script ***noah_nus.py*** (given below)

The acquired NOAH data set may be processed using the standard NOAH processing AU scripts which will recognise the use of NUS sampling and will process data accordingly. A Hilbert transform is also called to allow 2D phasing.

NOAH under automation

NOAH parameter sets may be used directly under automation within ICONNMR as for conventional experiment parameter sets. Data acquisition is initiated by the Bruker standard AU program `au_zgcosy` ($AUNM = au_zgcosy$). Following this, processing is handled via the `splitx_au` AU program which must be defined in the processing parameters ($AUNMP = splitx_au$), together with the user defined AU programs USERP1, USERP2 and USERP3. See **AU Programs** below for details.

For implementing *Wavemaker* statements for shaped pulse definition under automation the line:

```
XCMD("wvm -q")
```

should be included in the acquisition AU program, prior to the ZG statement.

Implementation of NUS under automation can be achieved by running the python NUS set-up script ***noah_nus.py*** in real time by including the line:

```
XPY("noah_nus")
```

in the acquisition AU program, prior to the ZG statement. See **Python NUS script** below.

Pulse Programs

NOAH-3 BSC with magnitude-mode ASAP-COSY

```
;NOAH-3 BSC: HMBC + me-HSQC + COSY (QF)
;Topspin 3 version 2018/04/09
;WaveMaker supported version.
;With adiabatic refocussing, 2-step low-pass HMBC filter and
;optional multiplicity editing with J-compensation in HSQC
;COSY recorded as QF for magnitude processing- this is NOT NUS
compatible.
;
;Ref: E. Kupce and T. D. W. Claridge, Chem. Commun., 2018, 54, 7139-
7142, DOI: 10.1039/c8cc03296c
;Ref: Tim D. W. Claridge, Maksim Mayzel, and Eriks Kupce, submitted
;
;$CLASS=HighRes
;$DIM=2D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"p2=p1*2"
"p4=p3*2"
"d2=1s/(cnst2*2)" ;JCOMP
"d4=1s/(cnst2*4)"
"d5=1s/(cnst5*4)"
"d6=1s/(cnst13*2)"

"d0=3u"
"d10=3u"

"in0=inf1/2"
"cnst10=2*dw/inf1" ; cnst10 is a scaling factor swC/swH for COSY
"in10=inf1*cnst10"

"l3=td1/6"
"l9=(d9/(p45*20))"

"DELTA=p16+d16+p2/2+4u+d0-p3*2/PI"
"DELTA1=d4-p14/2"
"DELTA2=d4+p14/2"
"DELTA3=d6-p16-4u"
"DELTA4=DELTA-p2"
"DELTA5=DELTA1-p1*2/PI"
"DELTA6=DELTA2-p16-de"

"DELTA11=1s/(2*cnst6)-p16-d16"
"DELTA12=1s/(2*cnst7)-p16-d16"

"acqt0=0"
baseopt_echo

1 ze
30m pl12:f2
```



```

2 30m
3 4m do:f2
4 50u UNBLKGRAD
  4u pl1:f1
  p16:gp0
  4u
  (p1 ph1)
  4u
  p16:gp0*1.37
  50u BLKGRAD
  d1 st0
; ZZ-HMBC module

p1 ph1
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2
p1 ph1
DELTA1 UNBLKGRAD
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 pl2:f2
p1 ph1
DELTA11
p16:gp3
d16 pl2:f2
(p3 ph3):f2
DELTA12
p16:gp4
d16
(p3 ph3):f2
4u
p16:gp5
DELTA3
(p3 ph5):f2
DELTA
(p14:sp3 ph6):f2
4u
p16:gp1*EA*0.77
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA*0.77
d16
(p14:sp3 ph4):f2
DELTA pl2:f2
(p3 ph3):f2
(p2 ph1)
4u
p16:gp2*0.77
4u
d16 BLKGRAD
goscnp ph30

4u pl2:f2
(p3 ph1):f2
50u UNBLKGRAD

```

```

p16:gp0*1.77
5m st
; HSQC module

(p1 ph1)
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
(p1 ph2):f1
(p3 ph3):f2
DELTA
#ifdef EDIT                                ;optional multiplicity editing
(p31:sp18 ph1):f2
#else
(p14:sp3 ph1):f2
#endif
4u
p16:gp1*EA
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA
d16
#ifdef EDIT
d2
(p31:sp18 ph6):f2
DELTA4
(p2 ph2):f1
d2 p12:f2
#else
(p14:sp3 ph6):f2
DELTA p12:f2
#endif
(p3 ph5):f2
(p1 ph6):f1
DELTA5
(p14:sp3 ph6):f2
(p2 ph1):f1
p16:gp2
DELTA6 p112:f2 BLKGRAD
goscnp ph30 cpd2:f2
50u do:f2

if "d9 > 1m"    ;ASAP-COSY
{
50u ; UNBLKGRAD
p16:gp0*1.77
d16
50u ; BLKGRAD

9 (p45:sp45 ph10):f1
(p45:sp45 ph12):f1
(p45:sp45 ph11):f1
(p45:sp45 ph12):f1
(p45:sp45 ph10):f1
(p45:sp45 ph10):f1
(p45:sp45 ph12):f1

```

```

(p45:sp45 ph11):f1
(p45:sp45 ph12):f1
(p45:sp45 ph10):f1
(p45:sp45 ph13):f1
(p45:sp45 ph15):f1
(p45:sp45 ph14):f1
(p45:sp45 ph15):f1
(p45:sp45 ph13):f1
(p45:sp45 ph13):f1
(p45:sp45 ph15):f1
(p45:sp45 ph14):f1
(p45:sp45 ph15):f1
(p45:sp45 ph13):f1
lo to 9 times 19
4u pl1:f1      ;end mixing
}

```

```

50u ; UNBLKGRAD
p16:gp0
5m st

```

```

(p1 ph7)
d10      ;COSY t1-evolution
4u
p16:gp3
d16
(p1 ph1)
4u
p16:gp3
d16
go=2 ph31      ;acquire H-H COSY QF
30m wr #0 if #0 zd igrad EA
1m id10
lo to 3 times 2
1m id0
1m ip3*2
1m ip30*2
lo to 4 times 13

```

```

50u BLKGRAD
exit

```

```

ph1=0
ph2=1
ph3=0 2
ph4=0 0 0 0 2 2 2 2
ph5=0 0 2 2
ph6=0
ph7=0 2
ph10=0
ph11=(360) 60
ph12=(360) 150
ph13=(360) 180
ph14=(360) 240
ph15=(360) 330
ph30=0 2 2 0
ph31=0 2

```

```

;p11 : f1 channel - power level for pulse (default)
;p12 : f2 channel - power level for pulse (default)
;p112: f2 channel - power level for CPD/BB decoupling
;p1 : f1 channel - 90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p3 : f2 channel - 90 degree high power pulse
;p14: f2 channel - 180 degree shaped pulse for inversion
;p16: homospoil/gradient pulse [1 msec]
;p31: f2 channel - 180 degree shaped pulse for inversion with J-
compensation
;p45: f1 channel - 180 degree CAWURST-2 pulse
;sp3: f2 channel - shaped pulse 180 degree
;sp18: f2 channel - shaped pulse 180 degree with J-compensation
;sp45: f1 channel - CAWURST-2 pulse (180 degree)
;d0 : incremented delay (2D) [3 usec]
;d1 : relaxation delay; 1-5 * T1
;d6 : delay for evolution of long range couplings (1/2Jlr)
;d9: ASAP mixing duration [0 (no mixing) or 40-60 ms]
;d11: delay for disk I/O [5 msec]
;d16: delay for homospoil/gradient recovery
;cnst2: = J(XH)
;cnst6: = 1J(XH)min
;cnst7: = 1J(XH)max
;cnst13: = nJ(XH) long range
;inf1: 1/SW(X) = 2 * DW(X)
;in0: 1/(2 * SW(X)) = DW(X)
;nd0: 2
;ns: 2 * n
;ds: 16
;NBL=3: number of NOAH modules
;FnMODE: echo-antiecho

;WaveMaker shapes (optional)
; ~~~~~ use 'wvm -a' command to create these shapes
~~~~~
;sp3:wvm:wul80C13: cawurst-20(220 ppm, 0.5 ms; L2H)
;sp18:wvm:wul80Jcomp: wurst-40(280 ppm; Jcomp, L2H)
;cpd2:wvm:wudec: cawurst_d-20(220 ppm, 1.4 ms; L2H)
;sp45:wvm:wuASAP: cawurst-2(30 ppm, 1.0 ms; Q=3)

;use gradient ratio: gp 0 : gp 1 : gp 2 : gp 3 gp4 : gp5
; 17.13 : 80 : 40.2 : 15 : -10 : -5

;for z-only gradients:
;gpz0: 17.13%
;gpz1: 80%
;gpz2: 40.2% for C-13, 16.2% for N-15
;gpz3: 15%
;gpz4: -10%
;gpz5: -5%

;use gradient files:
;gpnam0: SMSQ10.100
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100
;gpnam3: SMSQ10.100
;gpnam4: SMSQ10.100
;gpnam5: SMSQ10.100

;EDIT: for C-13 multiplicity editing start experiment with

```

```
;          option -DEDIT (eda: ZGOPTNS)

;Processing
;use splitx au-program to split the data
;define user processing au programs (USERP1...P3) for automatic
processing of all data sets
;COSY requires FnTYPE QF for processing, defined by processing au
userP3 = "noah_cosy QF"
```

NOAH-3 BSC^c

```
;NOAH-3 BScC: HMBC + me-HSQC + CLIP-COSY
;Topspin 3 version 2018/04/09
;WaveMaker supported version.
;With adiabatic refocussing, 2-step low-pass HMBC filter and
;optional multiplicity editing with J-compensation in HSQC
;
;Ref: E. Kupce and T. D. W. Claridge, Chem. Commun., 2018, 54, 7139-
7142, DOI: 10.1039/c8cc03296c
;Ref: Tim D. W. Claridge, Maksim Mayzel, and Eriks Kupce, submitted
;
;$CLASS=HighRes
;$DIM=2D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"p2=p1*2"
"p4=p3*2"
"d2=1s/(cnst2*2)"      ;JCOMP
"d4=1s/(cnst2*4)"
"d5=1s/(cnst5*4)"
"d6=1s/(cnst13*2)"

"d0=3u"
"d10=3u"

"in0=inf1/2"
"cnst10=2*dw/inf1"      ; cnst10 is a scaling factor swC/swH for COSY
"in10=inf1*cnst10"

"l3=td1/6"

"DELTA=p16+d16+p2/2+4u+d0-p3*2/PI"
"DELTA1=d4-p14/2"
"DELTA2=d4+p14/2"
"DELTA3=d6-p16-4u"
"DELTA4=DELTA-p2"
"DELTA5=DELTA1-p1*2/PI"
"DELTA6=DELTA2-p16-de"

"DELTA11=1s/(2*cnst6)-p16-d16"
"DELTA12=1s/(2*cnst7)-p16-d16"

"acqt0=0"
baseopt_echo

1 ze
  30m p112:f2
2 30m
3 4m do:f2
4 50u UNBLKGRAD
  4u p11:f1
  p16:gp0
  4u
```

```

(p1 ph1)
4u
p16:gp0*1.37
50u BLKGRAD
d1 st0
; ZZ-HMBC module

p1 ph1
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2
p1 ph1
DELTA1 UNBLKGRAD
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
p1 ph1
DELTA11
p16:gp3
d16 p12:f2
(p3 ph3):f2
DELTA12
p16:gp4
d16
(p3 ph3):f2
4u
p16:gp5
DELTA3
(p3 ph5):f2
DELTA
(p14:sp3 ph6):f2
4u
p16:gp1*EA*0.77
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA*0.77
d16
(p14:sp3 ph4):f2
DELTA p12:f2
(p3 ph3):f2
(p2 ph1)
4u
p16:gp2*0.77
4u
d16 BLKGRAD
goscnp ph30

4u p12:f2
(p3 ph1):f2
50u UNBLKGRAD
p16:gp0*1.77
5m st
; HSQC module

(p1 ph1)
DELTA1

```

```

(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 pl2:f2
(p1 ph2):f1
(p3 ph3):f2
DELTA
#ifdef EDIT                                ;optional multiplicity editing
    (p31:sp18 ph1):f2
#else
    (p14:sp3 ph1):f2
#endif
4u
p16:gp1*EA
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA
d16
#ifdef EDIT
    d2
    (p31:sp18 ph6):f2
    DELTA4
    (p2 ph2):f1
    d2 pl2:f2
#else
    (p14:sp3 ph6):f2
    DELTA pl2:f2
#endif
(p3 ph5):f2
(p1 ph6):f1
DELTA5
(p14:sp3 ph6):f2
(p2 ph1):f1
p16:gp2
DELTA6 pl12:f2 BLKGRAD
goscnp ph30 cpd2:f2
50u do:f2
50u UNBLKGRAD
p16:gp0
5m st

;begin CLIP-COSY experiment
p1 ph7 ;excitation, States
d10
p1 ph10
10u gron6 pl0:f1
(p32:sp29 ph8):f1
20u groff
d16 pl1:f1
p1 ph22

;in-phase transfer (planar mixing)
d5
p2 ph20
d5
p1 ph20
d5
p2 ph23

```



```

d5

p1 ph21
10u gron6*1.333 p10:f1
(p32*0.75:sp29 ph9):f1
20u groff
p16:gp3
d16 p11:f1 ;BLKGRAD
p1 ph24

go=2 ph31      ; acquire H-H CLIP-COSY
30m wr #0 if #0 zd igrad EA

1m ip7

lo to 3 times 2
1m rp7
1m id0
1m id10
1m ip3*2
1m ip30*2
lo to 4 times 13

50u BLKGRAD
exit

ph1=0
ph2=1
ph3=0 2
ph4=0 0 0 0 2 2 2 2
ph5=0 0 2 2
ph6=0
ph7=0 2
ph8=0
ph9=2
ph10=0
ph20=1
ph21=0
ph22=0
ph23=3
ph24=0
ph30=0 2 2 0
ph31=0 2

;p11 : f1 channel - power level for pulse (default)
;p12 : f2 channel - power level for pulse (default)
;p112: f2 channel - power level for CPD/BB decoupling
;p1 : f1 channel - 90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p3 : f2 channel - 90 degree high power pulse
;p14: f2 channel - 180 degree shaped pulse for inversion
;p16: homospoil/gradient pulse [1 msec]
;p31: f2 channel - 180 degree shaped pulse for inversion with J-
compensation
;p32: f1 channel - 180 degree shaped pulse (adiabatic) [20 msec]
; smoothed chirp (sweepwidth, 20% smoothing, 10000 points)
;sp3: f2 channel - shaped pulse 180 degree
;sp18: f2 channel - shaped pulse 180 degree with J-compensation

```

```

;sp29: f1 channel - shaped pulse (adiabatic)
;d0 : incremented delay (2D) [3 usec]
;d1 : relaxation delay; 1-5 * T1
;d6 : delay for evolution of long range couplings (1/2Jlr)
;d4 : 1/(4J(HH))
;d11: delay for disk I/O [5 msec]
;d16: delay for homospoil/gradient recovery
;cnst2: = 1J(XH)
;cnst6: = 1J(XH)min
;cnst7: = 1J(XH)max
;cnst5: = > J(HH); 30 Hz
;cnst10: calculated internally; set to 1 for correct processing
;cnst13: = nJ(XH) long range
;inf1: 1/SW(X) = 2 * DW(X)
;in0: 1/(2 * SW(X)) = DW(X)
;nd0: 2
;ns: 2 * n
;ds: 16
;NBL=3: number of NOAH modules
;FnMODE: echo-antiecho

;WaveMaker shapes (optional)
; ~~~~~ use 'wvm -a' command to create these shapes
~~~~~
;sp3:wvm:wul80C13: cawurst-20(80 kHz, 0.5 ms; L2H)
;sp18:wvm:wul80Jcomp: wurst-40(280 ppm; Jcomp, L2H)
;cpd2:wvm:wudec: cawurst_d-20(220 ppm, 1.4 ms; L2H)

;use gradient ratio: gp 0 : gp1 : gp2 : gp3 : gp4 : gp5 : gp6
; 17.13 : 80 : 40.2 : 15 : -10 : -5 : 11

;for z-only gradients:
;gpz0: 17.13%
;gpz1: 80%
;gpz2: 40.2%
;gpz3: 15% for purging
;gpz4: -10%
;gpz5: -5%
;gpz6: 11% for ZQ filter

;use gradient files:
;gpnam0: SMSQ10.100
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100
;gpnam3: SMSQ10.100
;gpnam4: SMSQ10.100
;gpnam5: SMSQ10.100
;gpnam6: SMSQ10.100

;EDIT: for C-13 multiplicity editing start experiment with
; option -DEDIT (eda: ZGOPTNS)

;Processing
;use splitx au-program to split the data
;define user processing au programs (USERP1...P3) for automatic
processing of all data sets
;CLIP-COSY requires FnTYPE States defined by processing au userP3 =
"noah_clipcosy States"

```

NOAH-3 BSD

```
;NOAH-3 BSD: HMBC + me-HSQC + DQF-COSY
;Topspin 3 version 2018/04/09
;WaveMaker supported version.
;With adiabatic refocussing, 2-step low-pass HMBC filter and
;optional multiplicity editing with J-compensation in HSQC
;
;Ref: E. Kupce and T. D. W. Claridge, Chem. Commun., 2018, 54, 7139-
7142, DOI: 10.1039/c8cc03296c
;Ref: Tim D. W. Claridge, Maksim Mayzel, and Eriks Kupce, submitted
;
;$CLASS=HighRes
;$DIM=2D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"p2=p1*2"
"p4=p3*2"
"d2=1s/(cnst2*2)" ;JCOMP
"d4=1s/(cnst2*4)"
"d5=1s/(cnst5*4)"
"d6=1s/(cnst13*2)"

"d0=3u"
"d10=3u"

"in0=inf1/2"
"cnst10=2*dw/inf1" ; cnst10 is a scaling factor swC/swH for COSY
"in10=inf1*cnst10"

"l3=td1/6"

"DELTA=p16+d16+p2/2+4u+d0-p3*2/PI"
"DELTA1=d4-p14/2"
"DELTA2=d4+p14/2"
"DELTA3=d6-p16-4u"
"DELTA4=DELTA-p2"
"DELTA5=DELTA1-p1*2/PI"
"DELTA6=DELTA2-p16-de"

"DELTA11=1s/(2*cnst6)-p16-d16"
"DELTA12=1s/(2*cnst7)-p16-d16"

"DELTA15=p16+d16+d10"
"DELTA16=p16+d16+8u"

"acqt0=0"
baseopt_echo

1 ze
  30m p112:f2
2 30m
3 4m do:f2
```

```

4 50u UNBLKGRAD
4u pl1:f1
p16:gp0
4u
(p1 ph1)
4u
p16:gp0*1.37
50u BLKGRAD
d1 st0
; ZZ-HMBC module

p1 ph1
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2
p1 ph1
DELTA1 UNBLKGRAD
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 pl2:f2
p1 ph1
DELTA11
p16:gp3
d16 pl2:f2
(p3 ph3):f2
DELTA12
p16:gp4
d16
(p3 ph3):f2
4u
p16:gp5
DELTA3
(p3 ph5):f2
DELTA
(p14:sp3 ph6):f2
4u
p16:gp1*EA*0.77
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA*0.77
d16
(p14:sp3 ph4):f2
DELTA pl2:f2
(p3 ph3):f2
(p2 ph1)
4u
p16:gp2*0.77
4u
d16 BLKGRAD
goscnp ph30

4u pl2:f2
(p3 ph1):f2
50u UNBLKGRAD
p16:gp0*1.77
5m st

```

```

; HSQC module

(p1 ph1)
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
(p1 ph2):f1
(p3 ph3):f2
DELTA
#ifdef EDIT                                ;optional multiplicity editing
    (p31:sp18 ph1):f2
#else
    (p14:sp3 ph1):f2
#endif
4u
p16:gp1*EA
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA
d16
#ifdef EDIT
    d2
    (p31:sp18 ph6):f2
    DELTA4
    (p2 ph2):f1
    d2 p12:f2
#else
    (p14:sp3 ph6):f2
    DELTA p12:f2
#endif
(p3 ph5):f2
(p1 ph6):f1
DELTA5
(p14:sp3 ph6):f2
(p2 ph1):f1
p16:gp2
DELTA6 p112:f2 BLKGRAD
goscnp ph30 cpd2:f2
50u do:f2
50u UNBLKGRAD
p16:gp0
5m st

p1 ph7                                ;start DQF COSY
DELTA15
p2 ph1
d10                                ; sw1 scaling factor
p16:gp6*EA
d16
p1 ph1
DELTA16
p2 ph1
8u
p16:gp7
d16
p1 ph1

```

```

DELTA16
p2 ph1
4u
p16:gp8
d16

go=2 ph31      ; acquire H-H DQF-COSY
30m wr #0 if #0 zd igrad EA

lo to 3 times 2
1m id0
1m id10
1m ip3*2
1m ip7*2
1m ip30*2
1m ip31*2
lo to 4 times 13

50u BLKGRAD
exit

ph1=0
ph2=1
ph3=0 2
ph4=0 0 0 0 2 2 2 2
ph5=0 0 2 2
ph6=0
ph7=0 2
ph10=0
ph30=0 2 2 0
ph31=0 2

;p11 : f1 channel - power level for pulse (default)
;p12 : f2 channel - power level for pulse (default)
;p112: f2 channel - power level for CPD/BB decoupling
;p1 : f1 channel - 90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p3 : f2 channel - 90 degree high power pulse
;p14: f2 channel - 180 degree shaped pulse for inversion
;p31: f2 channel - 180 degree shaped pulse for inversion with J-
compensation
;sp3: f2 channel - shaped pulse 180 degree
;sp18: f2 channel - shaped pulse 180 degree with J-compensation
;p16: homospoil/gradient pulse [1 msec]
;d0 : incremented delay (2D) [3 usec]
;d1 : relaxation delay; 1-5 * T1
;d6 : delay for evolution of long range couplings (1/2Jlr)
;d11: delay for disk I/O [5 msec]
;d16: delay for homospoil/gradient recovery
;cnst2: = J(XH)
;cnst6: = 1J(XH)min
;cnst7: = 1J(XH)max
;cnst10: calculated internally; set to 1 for correct processing
;cnst13: = nJ(XH) long range
;inf1: 1/SW(X) = 2 * DW(X)
;in0: 1/(2 * SW(X)) = DW(X)
;nd0: 2
;ns: 2 * n

```

```

;ds: 16
;NBL=3: number of NOAH modules
;FnMODE: echo-antiecho

;WaveMaker shapes (optional)
; ~~~~~ use 'wvm -a' command to create these shapes
~~~~~
;sp3:wvm:wul80C13: cawurst-20(80 kHz, 0.5 ms; L2H)
;sp18:wvm:wul80Jcomp: wurst-40(280 ppm; Jcomp, L2H)
;cpd2:wvm:wudec: cawurst_d-20(220 ppm, 1.4 ms; L2H)

;use gradient ratio:      gp 0 : gp1 : gp2 : gp3 : gp4 : gp5 : gp6 :
gp7 : gp8
;                          17.13 : 80 : 40.2 : 15 : -10 : -5 : 15 :
5 : 25

;for z-only gradients:
;gpz0: 17.13%
;gpz1: 80%
;gpz2: 40.2%
;gpz3: 15%
;gpz4: -10%
;gpz5: -5%
;gpz6: 15%
;gpz7: 5%
;gpz8: 25%

;use gradient files:
;gpnam0: SMSQ10.100
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100
;gpnam3: SMSQ10.100
;gpnam4: SMSQ10.100
;gpnam5: SMSQ10.100
;gpnam6: SMSQ10.100
;gpnam7: SMSQ10.100
;gpnam8: SMSQ10.100

;EDIT: for C-13 multiplicity editing start experiment with
;          option -DEDIT (eda: ZGOPTNS)

;Processing
;use splitx au-program to split the data
;define user processing au programs (USERP1...P3) for automatic
processing of all data sets
;DQF-COSY requires phase-sensitive echo-antiecho defined by
processing au userP3 = "noah_noesy"

```

NOAH3-BST

```
;NOAH-3 BST: HMBC + me-HSQC + DIPSI2-TOCSY
;Topspin 3 version 2018/04/09
;WaveMaker supported version.
;With adiabatic refocussing, 2-step low-pass HMBC filter and
;optional multiplicity editing with J-compensation in HSQC
;
;Ref: E. Kupce and T. D. W. Claridge, Chem. Commun., 2018, 54, 7139-
7142, DOI: 10.1039/c8cc03296c
;Ref: Tim D. W. Claridge, Maksim Mayzel, and Eriks Kupce, submitted
;
;$CLASS=HighRes
;$DIM=2D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"p2=p1*2"
"p4=p3*2"
"d2=1s/(cnst2*2)" ;JCOMP
"d4=1s/(cnst2*4)"
"d5=1s/(cnst5*4)"
"d6=1s/(cnst13*2)"

"d0=3u"
"d10=3u"

"in0=inf1/2"
"cnst10=2*dw/inf1" ; cnst10 is a scaling factor swC/swH for COSY
"in10=inf1*cnst10"

"l3=td1/6"

"FACTOR1=(d9/(p6*115.112))/2"
"l1=FACTOR1*2"

"DELTA=p16+d16+p2/2+4u+d0-p3*2/PI"
"DELTA1=d4-p14/2"
"DELTA2=d4+p14/2"
"DELTA3=d6-p16-4u"
"DELTA4=DELTA-p2"
"DELTA5=DELTA1-p1*2/PI"
"DELTA6=DELTA2-p16-de"

"DELTA11=1s/(2*cnst6)-p16-d16"
"DELTA12=1s/(2*cnst7)-p16-d16"

"acqt0=0"
baseopt_echo

1 ze
  30m pl12:f2
2 30m
3 4m do:f2
4 50u UNBLKGRAD
```



```

4u p11:f1
p16:gp0
4u
(p1 ph1)
4u
p16:gp0*1.37
50u BLKGRAD
d1 st0
; ZZ-HMBC module

p1 ph1
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2
p1 ph1
DELTA1 UNBLKGRAD
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
p1 ph1
DELTA11
p16:gp3
d16 p12:f2
(p3 ph3):f2
DELTA12
p16:gp4
d16
(p3 ph3):f2
4u
p16:gp5
DELTA3
(p3 ph5):f2
DELTA
(p14:sp3 ph6):f2
4u
p16:gp1*EA*0.77
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA*0.77
d16
(p14:sp3 ph4):f2
DELTA p12:f2
(p3 ph3):f2
(p2 ph1)
4u
p16:gp2*0.77
4u
d16 BLKGRAD
goscnp ph30

4u p12:f2
(p3 ph1):f2
50u UNBLKGRAD
p16:gp0*1.77
5m st
; HSQC module

```

```

(p1 ph1)
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
(p1 ph2):f1
(p3 ph3):f2
DELTA
#ifdef EDIT                                ;optional multiplicity editing
    (p31:sp18 ph1):f2
#else
    (p14:sp3 ph1):f2
#endif
    4u
    p16:gp1*EA
    d16
    d0
    (p2 ph4)
    d0
    4u
    p16:gp1*EA
    d16
#ifdef EDIT
    d2
    (p31:sp18 ph6):f2
    DELTA4
    (p2 ph2):f1
    d2 p12:f2
#else
    (p14:sp3 ph6):f2
    DELTA p12:f2
#endif
    (p3 ph5):f2
    (p1 ph6):f1
    DELTA5
    (p14:sp3 ph6):f2
    (p2 ph1):f1
    p16:gp2
    DELTA6 p112:f2 BLKGRAD
    goscpn ph30 cpd2:f2
    50u do:f2
    50u UNBLKGRAD
    p16:gp0
    5m st

;begin TOCSY experiment
p1 ph7 ;excitation, States
d10
p1 ph10
10u gron6 p10:f1
(p32:sp29 ph8):f1
20u groff
d16 p110:f1

;begin DIPSI2
5 p6*3.556 ph23
p6*4.556 ph25
p6*3.222 ph23
p6*3.167 ph25

```

```

p6*0.333 ph23
p6*2.722 ph25
p6*4.167 ph23
p6*2.944 ph25
p6*4.111 ph23

p6*3.556 ph25
p6*4.556 ph23
p6*3.222 ph25
p6*3.167 ph23
p6*0.333 ph25
p6*2.722 ph23
p6*4.167 ph25
p6*2.944 ph23
p6*4.111 ph25

p6*3.556 ph25
p6*4.556 ph23
p6*3.222 ph25
p6*3.167 ph23
p6*0.333 ph25
p6*2.722 ph23
p6*4.167 ph25
p6*2.944 ph23
p6*4.111 ph25
lo to 5 times l1

;end DIPSI2

p16:gp3
d16
10u gron6*1.333 p10:f1
(p32*0.75:sp29 ph8):f1
20u groff
d16 p11:f1 ;BLKGRAD
p1 ph26

go=2 ph31 ; acquire H-H DIPSI2-TOCSY
30m wr #0 if #0 zd igrad EA

1m ip7

lo to 3 times 2
1m rp7
1m id0
1m id10
1m ip3*2
1m ip30*2
lo to 4 times l3

50u BLKGRAD
exit

```

```

ph1=0
ph2=1
ph3=0 2
ph4=0 0 0 0 2 2 2 2
ph5=0 0 2 2
ph6=0
ph7=0 2
ph8=0
ph10=0
ph23=3
ph25=1
ph26=0
ph30=0 2 2 0
ph31=0 2

;p11 : f1 channel - power level for pulse (default)
;p12 : f2 channel - power level for pulse (default)
;p112: f2 channel - power level for CPD/BB decoupling
;p1 : f1 channel - 90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p3 : f2 channel - 90 degree high power pulse
;p14: f2 channel - 180 degree shaped pulse for inversion
;sp3: f2 channel - shaped pulse 180 degree
;sp18: f2 channel - shaped pulse 180 degree with J-compensation
;sp29: f1 channel - shaped pulse (adiabatic)
;p16: homospoil/gradient pulse [1 msec]
;p31: f2 channel - 180 degree shaped pulse for inversion with J-
compensation
;p32: f1 channel - 180 degree shaped pulse (adiabatic) [20 msec]
; smoothed chirp (sweepwidth, 20% smoothing, 10000 points)
;d0 : incremented delay (2D) [3 usec]
;d1 : relaxation delay; 1-5 * T1
;d6 : delay for evolution of long range couplings (1/2Jlr)
;d9 : TOCSY mixing time
;d11: delay for disk I/O [5 msec]
;d16: delay for homospoil/gradient recovery
;cnst2: = J(XH)
;cnst6: = 1J(XH)min
;cnst7: = 1J(XH)max
;cnst13: = nJ(XH) long range
;inf1: 1/SW(X) = 2 * DW(X)
;in0: 1/(2 * SW(X)) = DW(X)
;nd0: 2
;ns: 2 * n
;ds: 16
;NBL=3: number of NOAH modules
;FnMODE: echo-antiecho

;WaveMaker shapes (optional)
; ~~~~~ use 'wvm -a' command to create these shapes
~~~~~
;sp3:wvm:wul80C13: cawurst-20(80 kHz, 0.5 ms; L2H)
;sp18:wvm:wul80Jcomp: wurst-40(280 ppm; Jcomp, L2H)
;cpd2:wvm:wudec: cawurst_d-20(220 ppm, 1.4 ms; L2H)

;use gradient ratio: gp 0 : gp1 : gp2 : gp3 : gp4 : gp5 : gp6
; 17.13 : 80 : 40.2 : 15 : -10 : -5 : 11

```

```
;for z-only gradients:
;gpz0: 17.13%
;gpz1: 80%
;gpz2: 40.2%
;gpz3: 15% for purging
;gpz4: -10%
;gpz5: -5%
;gpz6: 11% for ZQ filter

;use gradient files:
;gpnam0: SMSQ10.100
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100
;gpnam3: SMSQ10.100
;gpnam4: SMSQ10.100
;gpnam5: SMSQ10.100

;EDIT: for C-13 multiplicity editing start experiment with
;          option -DEDIT (eda: ZGOPTNS)

;Processing
;use splitx au-program to split the data
;define user processing au programs (USERP1...P3) for automatic
processing of all data sets
;TOCSY requires States to be defined by processing au userP3 =
"noah_tocsy States"
```

NOAH3-BSN

```
;NOAH-3 BSN: HMBC + me-HSQC + NOESY
;Topspin 3 version 2018/04/09
;WaveMaker supported version.
;With adiabatic refocussing, 2-step low-pass HMBC filter and
;optional multiplicity editing with J-compensation in HSQC
;
;Ref: E. Kupce and T. D. W. Claridge, Chem. Commun., 2018, 54, 7139-
7142, DOI: 10.1039/c8cc03296c
;Ref: Tim D. W. Claridge, Maksim Mayzel, and Eriks Kupce, submitted
;
;$CLASS=HighRes
;$DIM=2D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"p2=p1*2"
"p4=p3*2"
"d2=1s/(cnst2*2)" ;JCOMP
"d4=1s/(cnst2*4)"
"d5=1s/(cnst5*4)"
"d6=1s/(cnst13*2)"

"d0=3u"
"d10=3u"

"in0=inf1/2"
"cnst10=2*dw/inf1" ; cnst10 is a scaling factor swC/swH for COSY
"in10=inf1*cnst10"

"l3=td1/6"

"TAU=d8-p32-p16-d16-30u"

"DELTA=p16+d16+p2/2+4u+d0-p3*2/PI"
"DELTA1=d4-p14/2"
"DELTA2=d4+p14/2"
"DELTA3=d6-p16-4u"
"DELTA4=DELTA-p2"
"DELTA5=DELTA1-p1*2/PI"
"DELTA6=DELTA2-p16-de"

"DELTA11=1s/(2*cnst6)-p16-d16"
"DELTA12=1s/(2*cnst7)-p16-d16"

"acqt0=0"
baseopt_echo

1 ze
  30m pl12:f2
2 30m
3 4m do:f2
4 50u UNBLKGRAD
```

```

4u p11:f1
p16:gp0
4u
(p1 ph1)
4u
p16:gp0*1.37
50u BLKGRAD
d1 st0
; ZZ-HMBC module

p1 ph1
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2
p1 ph1
DELTA1 UNBLKGRAD
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
p1 ph1
DELTA11
p16:gp3
d16 p12:f2
(p3 ph3):f2
DELTA12
p16:gp4
d16
(p3 ph3):f2
4u
p16:gp5
DELTA3
(p3 ph5):f2
DELTA
(p14:sp3 ph6):f2
4u
p16:gp1*EA*0.77
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA*0.77
d16
(p14:sp3 ph4):f2
DELTA p12:f2
(p3 ph3):f2
(p2 ph1)
4u
p16:gp2*0.77
4u
d16 BLKGRAD
goscnp ph30

4u p12:f2
(p3 ph1):f2
50u UNBLKGRAD
p16:gp0*1.77
5m st
; HSQC module

```

```

(p1 ph1)
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
(p1 ph2):f1
(p3 ph3):f2
DELTA
#ifdef EDIT                                ;optional multiplicity editing
    (p31:sp18 ph1):f2
#else
    (p14:sp3 ph1):f2
#endif
4u
p16:gp1*EA
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA
d16
#ifdef EDIT
    d2
    (p31:sp18 ph6):f2
    DELTA4
    (p2 ph2):f1
    d2 p12:f2
#else
    (p14:sp3 ph6):f2
    DELTA p12:f2
#endif
(p3 ph5):f2
(p1 ph6):f1
DELTA5
(p14:sp3 ph6):f2
(p2 ph1):f1
p16:gp2
DELTA6 p112:f2 BLKGRAD
goscnp ph30 cpd2:f2
50u do:f2
50u UNBLKGRAD
p16:gp0
5m st

p1 ph7                ;start NOESY, States
d10
p1 ph1
10u gron6
(p32:sp29 ph6):f1
20u groff
p16:gp3
d16 p11:f1
; 4u BLKGRAD
TAU
p1 ph6

go=2 ph31              ; acquire H-H NOESY
30m wr #0 if #0 zd igrad EA

```



```

1m ip7

lo to 3 times 2
1m rp7
1m id0
1m id10
1m ip3*2
1m ip30*2
lo to 4 times 13

50u BLKGRAD
exit

ph1=0
ph2=1
ph3=0 2
ph4=0 0 0 0 2 2 2 2
ph5=0 0 2 2
ph6=0
ph7=0 2
ph10=0
ph30=0 2 2 0
ph31=0 2

;p11 : f1 channel - power level for pulse (default)
;p12 : f2 channel - power level for pulse (default)
;p112: f2 channel - power level for CPD/BB decoupling
;p1 : f1 channel - 90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p3 : f2 channel - 90 degree high power pulse
;p14: f2 channel - 180 degree shaped pulse for inversion
;sp3: f2 channel - shaped pulse 180 degree
;sp18: f2 channel - shaped pulse 180 degree with J-compensation
;sp29: f1 channel - shaped pulse (adiabatic)
;p16: homospoil/gradient pulse [1 msec]
;p31: f2 channel - 180 degree shaped pulse for inversion with J-
compensation
;p32: f1 channel - 180 degree shaped pulse (adiabatic) [20 msec]
; smoothed chirp (sweepwidth, 20% smoothing, 10000 points)
;d0 : incremented delay (2D) [3 usec]
;d1 : relaxation delay; 1-5 * T1
;d6 : delay for evolution of long range couplings (1/2Jlr)
;d8 : NOESY mixing time
;d11: delay for disk I/O [5 msec]
;d16: delay for homospoil/gradient recovery
;cnst2: = J(XH)
;cnst6: = 1J(XH)min
;cnst7: = 1J(XH)max
;cnst13: = nJ(XH) long range
;inf1: 1/SW(X) = 2 * DW(X)
;in0: 1/(2 * SW(X)) = DW(X)
;nd0: 2
;ns: 2 * n
;ds: 16
;NBL=3: number of NOAH modules
;FnMODE: echo-antiecho

```

```

;WaveMaker shapes (optional)
; ~~~~~ use 'wvm -a' command to create these shapes
~~~~~
;sp3:wvm:wul80C13: cawurst-20(80 kHz, 0.5 ms; L2H)
;sp18:wvm:wul80Jcomp: wurst-40(280 ppm; Jcomp, L2H)
;cpd2:wvm:wudec: cawurst_d-20(220 ppm, 1.4 ms; L2H)

;use gradient ratio:      gp 0 : gp1 : gp2 : gp3 : gp4 : gp5 : gp6
;                          17.13 : 80 : 40.2 : 15 : -10 : -5 : 11

;for z-only gradients:
;gpz0: 17.13%
;gpz1: 80%
;gpz2: 40.2%
;gpz3: 15% for purging
;gpz4: -10%
;gpz5: -5%
;gpz6: 11% for ZQ filter

;use gradient files:
;gpnam0: SMSQ10.100
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100
;gpnam3: SMSQ10.100
;gpnam4: SMSQ10.100
;gpnam5: SMSQ10.100

;EDIT: for C-13 multiplicity editing start experiment with
;      option -DEDIT (eda: ZGOPTNS)

;Processing
;use splitx au-program to split the data
;define user processing au programs (USERP1...P3) for automatic
processing of all data sets
;NOESY requires FnTYPE States defined by processing au userP3 =
"noah_noesy States"

```

NOAH-3 BSR^o with off-resonance ROESY

```
;NOAH-3 BSoR: HMBC + me-HSQC + ROESY (off-res)
;Topspin 3 version 2018/04/09
;WaveMaker supported version.
;With adiabatic refocussing, 2-step low-pass HMBC filter and
;optional multiplicity editing with J-compensation in HSQC
;ROESY uses hi/lo off-resonance adiabatic pulses: total mixing time
= 2 * p50
;
;Ref: E. Kupce and T. D. W. Claridge, Chem. Commun., 2018, 54, 7139-
7142, DOI: 10.1039/c8cc03296c
;Ref: Tim D. W. Claridge, Maksim Mayzel, and Eriks Kupce, submitted
;
;$CLASS=HighRes
;$DIM=2D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"p2=p1*2"
"p4=p3*2"
"d2=1s/(cnst2*2)" ;JCOMP
"d4=1s/(cnst2*4)"
"d5=1s/(cnst5*4)"
"d6=1s/(cnst13*2)"

"d0=3u"
"d10=3u"
"d15=2*p50"

"in0=inf1/2"
"cnst10=2*dw/inf1" ; cnst10 is a scaling factor swC/swH for COSY
"in10=inf1*cnst10"

"l3=td1/6"

"DELTA=p16+d16+p2/2+4u+d0-p3*2/PI"
"DELTA1=d4-p14/2"
"DELTA2=d4+p14/2"
"DELTA3=d6-p16-4u"
"DELTA4=DELTA-p2"
"DELTA5=DELTA1-p1*2/PI"
"DELTA6=DELTA2-p16-de"

"DELTA11=1s/(2*cnst6)-p16-d16"
"DELTA12=1s/(2*cnst7)-p16-d16"

"acqt0=0"
baseopt_echo

1 ze
  30m pl12:f2
2 30m
3 4m do:f2
4 50u UNBLKGRAD
```

```

4u p11:f1
p16:gp0
4u
(p1 ph1)
4u
p16:gp0*1.37
50u BLKGRAD
d1 st0
; ZZ-HMBC module

p1 ph1
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2
p1 ph1
DELTA1 UNBLKGRAD
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
p1 ph1
DELTA11
p16:gp3
d16 p12:f2
(p3 ph3):f2
DELTA12
p16:gp4
d16
(p3 ph3):f2
4u
p16:gp5
DELTA3
(p3 ph5):f2
DELTA
(p14:sp3 ph6):f2
4u
p16:gp1*EA*0.77
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA*0.77
d16
(p14:sp3 ph4):f2
DELTA p12:f2
(p3 ph3):f2
(p2 ph1)
4u
p16:gp2*0.77
4u
d16 BLKGRAD
goscnp ph30

4u p12:f2
(p3 ph1):f2
50u UNBLKGRAD
p16:gp0*1.77
5m st
; HSQC module

```

```

(p1 ph1)
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
(p1 ph2):f1
(p3 ph3):f2
DELTA
#ifdef EDIT                                ;optional multiplicity editing
    (p31:sp18 ph1):f2
#else
    (p14:sp3 ph1):f2
#endif
4u
p16:gp1*EA
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA
d16
#ifdef EDIT
    d2
    (p31:sp18 ph6):f2
    DELTA4
    (p2 ph2):f1
    d2 p12:f2
#else
    (p14:sp3 ph6):f2
    DELTA p12:f2
#endif
(p3 ph5):f2
(p1 ph6):f1
DELTA5
(p14:sp3 ph6):f2
(p2 ph1):f1
p16:gp2
DELTA6 p112:f2 BLKGRAD
goscnp ph30 cpd2:f2
50u do:f2
50u UNBLKGRAD
p16:gp0
5m st

p1 ph7                                ;start ROESY, States
d10
p1 ph1
10u
(p50:sp49 ph6):f1
(p50:sp50 ph6):f1
10u
p16:gp3
d16 p11:f1
p1 ph6

go=2 ph31                                ; acquire H-H ROESY
30m wr #0 if #0 zd iggrad EA

```

```

1m ip7

lo to 3 times 2
1m rp7
1m id0
1m id10
1m ip3*2
1m ip30*2
lo to 4 times 13

50u BLKGRAD
"d15=d15"           ;display total ROESY mixing time
exit

ph1=0
ph2=1
ph3=0 2
ph4=0 0 0 0 2 2 2 2
ph5=0 0 2 2
ph6=0
ph7=0 2
ph10=0
ph30=0 2 2 0
ph31=0 2

;p11 : f1 channel - power level for pulse (default)
;p12 : f2 channel - power level for pulse (default)
;p112: f2 channel - power level for CPD/BB decoupling
;p1 : f1 channel - 90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p3 : f2 channel - 90 degree high power pulse
;p14: f2 channel - 180 degree shaped pulse for inversion
;p50 : f1 channel - Adiabatic pulse for ROESY (half total mixing
time)
;
;           total mixing time = 2*p50
;sp3: f2 channel - shaped pulse 180 degree
;sp18: f2 channel - shaped pulse 180 degree with J-compensation
;sp49: f1 channel - first shaped pulse WURSTAM (adiabatic) for ROESY
mixing (high shift)
;sp50: f1 channel - second shaped pulse WURSTAM (adiabatic) for ROESY
mixing (low shift)
;p16: homospoil/gradient pulse [1 msec]
;p31: f2 channel - 180 degree shaped pulse for inversion with J-
compensation
;d0 : incremented delay (2D) [3 usec]
;d1 : relaxation delay; 1-5 * T1
;d6 : delay for evolution of long range couplings (1/2Jlr)
;d15 : Total ROESY mixing time =2*p50
;d11: delay for disk I/O [5 msec]
;d16: delay for homospoil/gradient recovery
;cnst2: = J(XH)
;cnst6: = 1J(XH)min
;cnst7: = 1J(XH)max
;cnst13: = nJ(XH) long range
;inf1: 1/SW(X) = 2 * DW(X)
;in0: 1/(2 * SW(X)) = DW(X)
;nd0: 2
;ns: 2 * n

```

```

;ds: 16
;NBL=3: number of NOAH modules
;FnMODE: echo-antiecho

;WaveMaker shapes (optional)
; ~~~~~ use 'wvm -a' command to create these shapes
~~~~~
;sp3:wvm:wul80C13: cawurst-20(80 kHz, 0.5 ms; L2H)
;cpd2:wvm:wudec: cawurst_d-20(220 ppm, 1.4 ms; L2H)
;sp18:wvm:wul80Jcomp: wurst-40(280 ppm; Jcomp, L2H)
;sp49:wvm:wul80H1SL: wurstAM(p50, 9.5 ppm; Blmax = 5.0 kHz)
;sp50:wvm:wul80H1SL2: wurstAM(p50, -0.5 ppm; Blmax = 5.0 kHz)

;use gradient ratio:      gp 0 : gp1 : gp2 : gp3 : gp4 : gp5
;                          17.13 : 80 : 40.2 : 15 : -10 : -5

;for z-only gradients:
;gpz0: 17.13%
;gpz1: 80%
;gpz2: 40.2%
;gpz3: 15% for purging
;gpz4: -10%
;gpz5: -5%

;use gradient files:
;gpnam0: SMSQ10.100
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100
;gpnam3: SMSQ10.100
;gpnam4: SMSQ10.100
;gpnam5: SMSQ10.100

;EDIT: for C-13 multiplicity editing start experiment with
;      option -DEDIT (eda: ZGOPTNS)

;Processing
;use splitx au-program to split the data
;define user processing au programs (USERP1...P3) for automatic
processing of all data sets
;ROESY requires FnTYPE States defined by processing au userP3 =
"noah_roesy States"

```

NOAH-3 BSR with phase-alternating spin-lock ROESY

```
;NOAH-3 BSR: HMBC + me-HSQC + ROESY
;Topspin 3 version 2018/04/09
;WaveMaker supported version.
;With adiabatic refocussing, 2-step low-pass HMBC filter and
;optional multiplicity editing with J-compensation in HSQC
;ROESY spinlock using 180x 180-x pulses
;
;Ref: E. Kupce and T. D. W. Claridge, Chem. Commun., 2018, 54, 7139-
7142, DOI: 10.1039/c8cc03296c
;Ref: Tim D. W. Claridge, Maksim Mayzel, and Eriks Kupce, submitted
;
;$CLASS=HighRes
;$DIM=2D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

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

"p2=p1*2"
"p4=p3*2"
"d2=1s/(cnst2*2)" ;JCOMP
"d4=1s/(cnst2*4)"
"d5=1s/(cnst5*4)"
"d6=1s/(cnst13*2)"

"d0=3u"
"d10=3u"

"in0=inf1/2"
"cnst10=2*dw/inf1" ; cnst10 is a scaling factor swC/swH for COSY
"in10=inf1*cnst10"

"l3=td1/6"
"l4=p15/(p25*2)"

"DELTA=p16+d16+p2/2+4u+d0-p3*2/PI"
"DELTA1=d4-p14/2"
"DELTA2=d4+p14/2"
"DELTA3=d6-p16-4u"
"DELTA4=DELTA-p2"
"DELTA5=DELTA1-p1*2/PI"
"DELTA6=DELTA2-p16-de"

"DELTA11=1s/(2*cnst6)-p16-d16"
"DELTA12=1s/(2*cnst7)-p16-d16"

"acqt0=0"
baseopt_echo

1 ze
  30m p112:f2
2 30m
3 4m do:f2
4 50u UNBLKGRAD
```



```

4u p11:f1
p16:gp0
4u
(p1 ph1)
4u
p16:gp0*1.37
50u BLKGRAD
d1 st0
; ZZ-HMBC module

p1 ph1
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2
p1 ph1
DELTA1 UNBLKGRAD
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
p1 ph1
DELTA11
p16:gp3
d16 p12:f2
(p3 ph3):f2
DELTA12
p16:gp4
d16
(p3 ph3):f2
4u
p16:gp5
DELTA3
(p3 ph5):f2
DELTA
(p14:sp3 ph6):f2
4u
p16:gp1*EA*0.77
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA*0.77
d16
(p14:sp3 ph4):f2
DELTA p12:f2
(p3 ph3):f2
(p2 ph1)
4u
p16:gp2*0.77
4u
d16 BLKGRAD
goscnp ph30

4u p12:f2
(p3 ph1):f2
50u UNBLKGRAD
p16:gp0*1.77
5m st
; HSQC module

```

```

(p1 ph1)
DELTA1
(p14:sp3 ph6):f2
(p2 ph1):f1
DELTA2 p12:f2
(p1 ph2):f1
(p3 ph3):f2
DELTA
#ifdef EDIT                                ;optional multiplicity editing
(p31:sp18 ph1):f2
#else
(p14:sp3 ph1):f2
#endif
4u
p16:gp1*EA
d16
d0
(p2 ph4)
d0
4u
p16:gp1*EA
d16
#ifdef EDIT
d2
(p31:sp18 ph6):f2
DELTA4
(p2 ph2):f1
d2 p12:f2
#else
(p14:sp3 ph6):f2
DELTA p12:f2
#endif
(p3 ph5):f2
(p1 ph6):f1
DELTA5
(p14:sp3 ph6):f2
(p2 ph1):f1
p16:gp2
DELTA6 p112:f2 BLKGRAD
goscnp ph30 cpd2:f2
50u do:f2
50u UNBLKGRAD
p16:gp0
5m st
4u BLKGRAD

p1 ph7                                ;start ROESY, States
d10
4u p127:f1
5 p25 ph8
p25 ph9
lo to 5 times l4

go=2 ph31                                ; acquire H-H ROESY
30m wr #0 if #0 zd iggrad EA

1m ip7

lo to 3 times 2

```

```

1m rp7
1m id0
1m id10
1m ip3*2
1m ip30*2
lo to 4 times l3

50u BLKGRAD
exit

ph1=0
ph2=1
ph3=0 2
ph4=0 0 0 0 2 2 2 2
ph5=0 0 2 2
ph6=0
ph7=0 2 2 0 ;phase cycle for axial suppn in 2 scans
ph8=0 0 2 2
ph9=2 2 0 0
ph10=0
ph30=0 2 2 0
ph31=0 2 2 0

;p11 : f1 channel - power level for pulse (default)
;p12 : f2 channel - power level for pulse (default)
;p112: f2 channel - power level for CPD/BB decoupling
;p127: f1 channel - power level for pulsed ROESY-spinlock
;p1 : f1 channel - 90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p3 : f2 channel - 90 degree high power pulse
;p14: f2 channel - 180 degree shaped pulse for inversion
;p15: f1 channel - pulse for ROESY spinlock
;p16: homospoil/gradient pulse [1 msec]
;p31: f2 channel - 180 degree shaped pulse for inversion with J-
compensation
;p25: f1 channel - 180 degree pulse at p127
;sp3: f2 channel - shaped pulse 180 degree
;sp18: f2 channel - shaped pulse 180 degree with J-compensation
;d0 : incremented delay (2D) [3 usec]
;d1 : relaxation delay; 1-5 * T1
;d6 : delay for evolution of long range couplings (1/2Jlr)
;d11: delay for disk I/O [5 msec]
;d16: delay for homospoil/gradient recovery
;cnst2: = J(XH)
;cnst6: = 1J(XH)min
;cnst7: = 1J(XH)max
;cnst13: = nJ(XH) long range
;l4: loop for spinlock = p15 / p25*2
;inf1: 1/SW(X) = 2 * DW(X)
;in0: 1/(2 * SW(X)) = DW(X)
;nd0: 2
;ns: 2 * n
;ds: 16
;NBL=3: number of NOAH modules
;FnMODE: echo-antiecho

;WaveMaker shapes (optional)

```

```

; ~~~~~ use 'wvm -a' command to create these shapes
~~~~~
;sp3:wvm:wul80C13: cawurst-20(80 kHz, 0.5 ms; L2H)
;sp18:wvm:wul80Jcomp: wurst-40(280 ppm; Jcomp, L2H)
;cpd2:wvm:wudec: cawurst_d-20(220 ppm, 1.4 ms; L2H)

;use gradient ratio:      gp 0 : gp1 : gp2 : gp3 : gp4 : gp5
;                          17.13 : 80 : 40.2 : 15 : -10 : -5

;for z-only gradients:
;gpz0: 17.13%
;gpz1: 80%
;gpz2: 40.2%
;gpz3: 15% for purging
;gpz4: -10%
;gpz5: -5%

;use gradient files:
;gpnam0: SMSQ10.100
;gpnam1: SMSQ10.100
;gpnam2: SMSQ10.100
;gpnam3: SMSQ10.100
;gpnam4: SMSQ10.100
;gpnam5: SMSQ10.100

;EDIT: for C-13 multiplicity editing start experiment with
;      option -DEDIT (eda: ZGOPTNS)

;Processing
;use splitx au-program to split the data
;define user processing au programs (USERP1...P3) for automatic
processing of all data sets
;ROESY requires FnTYPE States defined by processing au userP3 =
"noah_roesy States"

```

AU Programs

splitx_au

Use this to split the NOAH-N experiment into N separate data sets for processing. When split, each data set will call the associated processing AU program (via **xaup**) to perform all required processing steps. A NOAH experiment numbered **Z** will then yield three data sets numbered **Z001**, **Z002** and **Z003**. For use in automation, this should be defined as AUNMP. It should be placed in the directory:

TOPSPIN-HOME\exp\stan\nmr\au\src\user

```
/* *****/
/* splitx_au - TopSpin3 version */
/* Based on NBL, splits and processes */
/* NOAH expts according to AU processing */
/* programs defined by userP1-P5 parameters */
/* EK, 21.11.2015, eriks.kupce@bruker.com */
/* *****/

int size[3];
int *row;
int i, j, k, td, tdl, tds, tdlS, tdx, nus;
int oexpno, nexpno, byteorder, parmode;
int nbl = 0, fntype = 0;
char path[PATH_MAX + 64], proc_nm[8], str[32], *ch, str1[128];
FILE **fp;

/***** get dataset and parameters *****/

FETCHPAR("PARMODE", &parmode)
FETCHPAR("NBL", &nbl)

if (parmode != 1) STOPMSG("Program is only suitable for 2D")
if ((nbl<2) || (nbl>8)) STOPMSG("Program is only suitable for 8 >
NBL > 1")

FETCHPARS("BYTORDA", &byteorder)
FETCHPARS("TD", &tds)
FETCHPAR1("TD", &tdl)
FETCHPAR1S("TD", &tdlS)

nus=0;
strcpy(path, ACQUPATH("acqu"));
ch=strrchr(path, '/');
*ch='\0';
strcat(path, "/vclist");
if(access(path,0)==0)
{
    nus=1;
    FETCHPAR("userA1", str1)
}

td = ((tds + 255) / 256) * 256;
if (td <= 0 || td > 64 * 1024 * 1024)
    STOPMSG("TD{F2} out of range")
if (tdlS <= 0 || tdlS > 64 * 1024 * 1024)
    STOPMSG("TD{F1} out of range")
tdx = nbl*td;
if ((double) tdx >= 2. * 1024 * 1024 * 1024 / sizeof(int))
    STOPMSG("amount of memory requested too large")
```

```

/***** get input *****/

oexpno = expno;
nexpno = expno * 1000 + 1;
if (nexpno <= 0)
    STOPMSG("invalid expno value")

/***** check files *****/

for(i=0; i<nbl; i++)
{
    WRAPARAM(nexpno+i)
    ERRORABORT
}

fp = (FILE **) calloc(nbl+1, sizeof(FILE *));

(void)strcpy(path, ACQUPATH("ser"));
if ((fp[nbl] = fopen(path, "rb")) == 0)
    STOPMSG(strcat(path, " - cannot open file for reading"))

for(i=0; i<nbl; i++)
{
    expno = nexpno+i;
    (void)strcpy(path, ACQUPATH("ser"));
    if ((fp[i] = fopen(path, "wb")) == 0)
        STOPMSG(strcat(path, " - cannot open file for writing"))
}

expno = oexpno;

/***** allocate memory *****/

if((row = (int*)malloc(tdx * sizeof(int))) == 0)
    STOPMSG("cannot get enough memory")

/***** split *****/

Show_status("splitting data");

k = tdl1s / nbl;
for (i = 0; i < k; i++)
{
    if (fread(row, sizeof(int), tdx, fp[nbl]) != (size_t)(tdx))
        STOPMSG("read failed")

    local_swap4(row, tdx * sizeof(int), byteorder);

    for (j = 0; j < nbl; j++)
    {
        if (fwrite(row+j*td, sizeof(int), td, fp[j]) != (size_t)(td))
            STOPMSG("write failed")
    }
}

/***** free resources *****/

for(i=0; i<nbl; i++) fclose(fp[i]);
fclose(fp[i]);

```

```

free(row);

/***** store parameters and generate audit entry ***/

tdl /= nbl;
tdls /= nbl;

strcpy(path, ACQUPATH(0));
byteorder = local_endian();
size[0] = td;
size[1] = tdls;
i = 0;
do
{
    double ymin, ymax;
    char autext[PATH_MAX + 256];
    char* hashbuf = autext + sprintf(autext, "created by splitx");

    hashbuf += sprintf(hashbuf, " %d", nbl);
    hashbuf += sprintf(hashbuf, " %d. dataset from\n          %s", i +
1, path);

    DATASET(name, nexpro + i, procno, disk, user)

    sprintf(proc_nm, "userP%d", i+1);
    FETCHPAR(proc_nm, str)
    STOREPAR("AUNMP", str)
    STOREPAR("NBL", 1)
    STOREPARS("NBL", 1)
    STOREPARS("BYTORDA", byteorder)
    STOREPAR1("TD", tdl)
    STOREPAR1S("TD", tdls)

    if(nus>0)
    {
        int td_0;
        char path2[256], newname[256];

        strcpy(path2, ACQUPATH("acqu"));
        ch=strrchr(path2, '/');
        *ch='\0';
        strcpy(newname,path2);
        strcat(path2, "/vclist");
        strcat(newname, "/nuslist");
        if(access(path2,0)==0)
        {
            rename(path2,newname);
            STOREPARS("FnTYPE", 2)
            STOREPAR("Mdd_CsALG", 1)
            STOREPAR("Mdd_CsNITER", 50)
            STOREPAR("Mdd_CsVE", 0)
            td_0 = atoi(str1);
            STOREPAR1S("NusTD", td_0/nbl)
        }
    }

    if (ChecksumFileMax(ACQUPATH("ser"), 0, hashbuf, 0, 0, byteorder,
0, parmode + 1, tds, size, size, &ymin, &ymax) > 0)
    {
        (void)AuditCreate(ACQUPATH("audita.txt"), autext);
    }
}

```

```

        STOREPARS("YMIN_a", ymin)
        STOREPARS("YMAX_a", ymax)
    }
    XAUPW;

    REFRESHDATA
    XCMD("sendgui browse_update_tree")
    Show_meta(SM_RAW | SM_RAWP);
}
while (++i < nbl);

Show_status("splitx done.");

QUIT

```

AU programs for individual modules

These are defined as the processing parameters *USERP1* to *USERP3* within each data set according to the triplet BSX combination and are called by the *splitx_au* program to correctly process each separated experiment. The *proc_noah* program is called by the *USERP1-3* processing programs to perform the appropriate FT and the *fixF1* program is called to correct the *f1* chemical shifts of the homonuclear module.

All AU programs should be placed in the directory:

TOPSPIN-HOME\exp\stan\nmr\au\src\user

noah_hmbc

```

/* AU for processing NOAH HMBC experiments */

if((i_argc > 2) && (strlen(i_argv[2]) > 1))    // processing USERPn
options
{
    sprintf(cmd, "1 FnMODE %s", i_argv[2]);
    XCMD (cmd)
    sprintf(cmd, "1s FnMODE %s", i_argv[2]);
    XCMD (cmd)
}

XCMD ("2 WDW SINE")
XCMD ("1 WDW SINE")
XCMD ("1 SSB 2")
XCMD ("2 SSB 4")
XCMD ("2 PH_mod no")
XCMD ("1 PH_mod no")

XCMD ("proc_noah")

XF2M
XCMD ("2 NLEV 20")
XCMD ("2 LEV0 5")
XCMD ("sendgui levcalc")
XCMD ("2 SPECTYP HMBC")
QUIT

```


noah_hsqc

```
/* AU for processing NOAH 13C HSQC/HMQC experiments */

if((i_argc > 2) && (strlen(i_argv[2]) > 1)) // processing USERPn
options
{
    sprintf(cmd, "1 FnMODE %s", i_argv[2]);
    XCMD (cmd)
    sprintf(cmd, "1s FnMODE %s", i_argv[2]);
    XCMD (cmd)
}

XCMD ("2 WDW QSINE")
XCMD ("1 WDW QSINE")
XCMD ("1 SSB 2")
XCMD ("2 SSB 2")
XCMD ("2 PH_mod pk")
XCMD ("1 PH_mod pk")

XCMD ("proc_noah")

XCMD ("sendgui apk2d")
XCMD ("2 NLEV 20")
XCMD ("sendgui levcalc")
XCMD ("2 SPECTYP HSQC")
QUIT
```

noah_cosy

```
/* AU for processing NOAH H-H magnitude correlation experiments */

if((i_argc > 2) && (strlen(i_argv[2]) > 1)) // processing USERPn
options
{
    sprintf(cmd, "1 FnMODE %s", i_argv[2]);
    XCMD (cmd)
    sprintf(cmd, "1s FnMODE %s", i_argv[2]);
    XCMD (cmd)
}

XCMD ("2 WDW SINE")
XCMD ("1 WDW SINE")
XCMD ("1 SSB 0")
XCMD ("2 SSB 0")
XCMD ("2 PH_mod no")
XCMD ("1 PH_mod mc")
XAU ("fixF1", "")
XCMD ("proc_noah") // XFB
XCMD ("abs1")
XCMD ("1 SR 0")
XCMD ("2 NLEV 20")
XCMD ("sendgui levcalc")
XCMD ("2 SPECTYP COSY")
QUIT
```

noah_clipcosy

```
/* AU for processing phase-sensitive NOAH H-H CLIPCOSY experiments*/

if((i_argc > 2) && (strlen(i_argv[2]) > 1))    // processing USERPn
options
{
    sprintf(cmd, "1 FnMODE %s", i_argv[2]);
    XCMD (cmd)
    sprintf(cmd, "1s FnMODE %s", i_argv[2]);
    XCMD (cmd)
}

XCMD ("2 WDW QSINE")
XCMD ("1 WDW QSINE")
XCMD ("1 SSB 2")
XCMD ("2 SSB 2")
XCMD ("2 PH_mod pk")
XCMD ("1 PH_mod pk")
XAU ("fixF1", "")

XCMD ("proc_noah")    // XFB

XCMD ("sendgui apk2d")
XCMD ("1 SR 0")
XCMD ("2 NLEV 20")
XCMD ("sendgui levcalc")
XCMD ("2 SPECTYP COSY")
QUIT
```

noah_tocsy

```
/* AU for processing phase-sensitive NOAH H-H TOCSY experiments */

if((i_argc > 2) && (strlen(i_argv[2]) > 1))    // processing USERPn
options
{
    sprintf(cmd, "1 FnMODE %s", i_argv[2]);
    XCMD (cmd)
    sprintf(cmd, "1s FnMODE %s", i_argv[2]);
    XCMD (cmd)
}

XCMD ("2 WDW QSINE")
XCMD ("1 WDW QSINE")
XCMD ("1 SSB 2")
XCMD ("2 SSB 2")
XCMD ("2 PH_mod pk")
XCMD ("1 PH_mod pk")
XAU ("fixF1", "")

XCMD ("proc_noah")    // XFB

XCMD ("sendgui apk2d")
XCMD ("1 SR 0")
XCMD ("2 NLEV 20")
XCMD ("sendgui levcalc")
XCMD ("2 SPECTYP TOCSY")
QUIT
```

noah_noesy

```
/* AU for processing phase-sensitive NOAH H-H NOESY correlation
experiments */

if((i_argc > 2) && (strlen(i_argv[2]) > 1)) // processing USERPn
options
{
    sprintf(cmd, "1 FnMODE %s", i_argv[2]);
    XCMD (cmd)
    sprintf(cmd, "1s FnMODE %s", i_argv[2]);
    XCMD (cmd)
}

XCMD ("2 WDW QSINE")
XCMD ("1 WDW QSINE")
XCMD ("1 SSB 2")
XCMD ("2 SSB 2")
XCMD ("2 PH_mod pk")
XCMD ("1 PH_mod pk")
XAU ("fixF1", "")

XCMD ("proc_noah") // XFB

XCMD ("sendgui apk2d")
XCMD ("1 SR 0")
XCMD ("2 NLEV 20")
XCMD ("sendgui levcalc")
XCMD ("2 SPECTYP NOESY")
QUIT
```

noah_oesy

```
/* AU for processing phase-sensitive NOAH H-H ROESY correlation
experiments */

if((i_argc > 2) && (strlen(i_argv[2]) > 1)) // processing USERPn
options
{
    sprintf(cmd, "1 FnMODE %s", i_argv[2]);
    XCMD (cmd)
    sprintf(cmd, "1s FnMODE %s", i_argv[2]);
    XCMD (cmd)
}

XCMD ("2 WDW QSINE")
XCMD ("1 WDW QSINE")
XCMD ("1 SSB 2")
XCMD ("2 SSB 2")
XCMD ("2 PH_mod pk")
XCMD ("1 PH_mod pk")
XAU ("fixF1", "")

XCMD ("proc_noah") // XFB

XCMD ("sendgui apk2d")
XCMD ("1 SR 0")
XCMD ("2 NLEV 20")
XCMD ("sendgui levcalc")
```

```
XCMD ("2 SPECTYP ROESY")
QUIT
```

proc_noah

```
/* proc_noah - EK, TC and MM */
/* Perform 2D Transform for standard or NUS NOAH data*/
/* If NUS data, Hilbert transform is also called to allow 2D phasing*/

int  fntp, tdl;

XFB

FETCHPARS("FnTYPE", &fntp)
if(fntp==2)
{
    XHT2
    FETCHPAR1S("NusTD", &tdl)
    STOREPAR1("TD", tdl)          // for display purposes only
}

QUIT
```

fixF1

```
/* fixF1 fix F1 axis for NOAH HH correlation spectra */
/* requires cnst10 - C/H scaling factor */
/* Eriks Kupce: eriks.kupce@bruker.com */

int    eno, ceno;
double a1=0.0, sw=0.0, sw1=0.0, sf1=0.0;
float  x1=0.0;
char   c1[8];

    FETCHPAR("CNST 10", &x1);    /* GETFLOAT("cnst10 :", x1); */

    FETCHPARS("SFO1", &sf1);
    STOREPAR1S("SFO1", sf1);

    FETCHPARS("NUC1", c1);
    STOREPAR1S("NUC1", c1);

    FETCHPARS("BF1", &a1);
    STOREPAR1S("BF1", a1);

    FETCHPAR1S("SW_h", &sw1);

    if(x1 < 0.001)
    {
        sprintf(text, "scaling factor = %.4f. \Aborting...", x1);
        STOPMSG(text);
    }
    else if(x1 == 1.0)    // cnst10 not set
    {
        FETCHPARS("SW_h", &sw);
        x1=sw1/sw;
    }
}
```

```
sw1 = sw1/x1;  
STOREPAR1S("SW_h", sw1);  
STOREPAR1S("SW", sw1/sf1);
```

```
FETCHPARS("O1", &a1);  
STOREPARS("O2", a1);  
STOREPAR1S("O1", a1);
```

```
QUIT
```

Python NUS script

This is used to set-up NOAH NUS based acquisitions. It should be placed in the directory:

TOPSPIN-HOME\exp\stan\nmr\py\user

```
from __future__ import with_statement
"""
noah_nus.py

TopSpin python script to setup NUS for NOAH experiments
Author: Maksim Mayzel, maksim.mayzel@bruker.com
Date: 2019-03-19
Usage: noah_nus
"""

import os
import shutil
import sys
import java.lang.System as System
import random
import de.bruker.nmr.mfw.root.UtilPath as UtilPath
from de.bruker.nmr.prsc.dbxml.ParfileLocator import getParfileDirs

TSHOME=UtilPath.getTopspinHome()
TSPROG=os.path.join(TSHOME,'prog','bin')
VC=os.path.join(TSHOME,'exp','stan','nmr','lists','vc')

def getpulprogtext(ppnam):
    """
    Get pulprog text
    """
    ppdirs=getParfileDirs(0)
    pptext=None
    for d in ppdirs:
        pplst=os.listdir(d)
        if ppnam in pplst:
            with open(os.path.join(d,ppnam)) as f:
                pptext=f.readlines()
            break
    if not pptext:
        ERRMSG('%s not found'%ppnam)
        EXIT()
    return pptext

def savepulprog(ppnam,pptext,overwrite=True):
    """
    Save pulprog to user pp dir
    """
    ppdirs=getParfileDirs(0)
    ready=0
    for d in ppdirs:
        if 'user' not in d:
            continue
        # pplst=os.listdir(d)
        # if ppnam in pplst and not CONFIRM("NOAH NUS","%s exists.
        Overwrite?"%ppnam):
            # ready=1
            # break
        with open(os.path.join(d,ppnam),'w') as f:
```

```

        f.writelines(ppText)
        ready=1
    break
if not ready:
    ERRMSG('Something went wrong.\n%s was not saved'%ppnam)
    EXIT()
return

def editNoah(ppText):
    """
    Add NUS to NOAH
    """
    ppTextNew=[]
    ddef=''
    tlinec=0
    for line in ppText:
        if line.startswith('"in0'):
            ddef='"d0=in0*tl1list+3u"\n'
        if line.startswith('"in10'):
            ddef+='"d10=in10*tl1list+3u"\n'
        if line.startswith('"in20'):
            ddef+='"d20=in20*tl1list+3u"\n'

        if "#include <Delay.incl" in line:
            line="%s\ndefine                                list<loopcounter>
tl1list=<$VCLIST>\n"%line
        if "4 50u UNBLKGRAD" in line:
            line="%s\n%s"%(line,ddef)
        if 'id0' in line or 'id10' in line or 'id20' in line:
            line='; %s'%line
            if not tlinec: #add tl1list.inc only once
                line='lm tl1list.inc\n%s'%line
                tlinec=1

        ppTextNew.append(line)
    return ppTextNew

def makenuslist():
    """
    Create sampling scheme
    """
    try:
        td=int(GETPAR('USERA1'))
    except:
        td=int(GETPAR("1 TD"))
    PUTPAR('USERA1',str(td))

    nbl=int(GETPAR("NBL"))
    ppnam=GETPAR("PULPROG")
    if not ppnam.startswith('noah%i'%nbl):
        ERRMSG("NBL %i does not correspond to PULPROG %s"%(nbl,ppnam))
        EXIT()
    nimax=int(td/nbl/2)
    ni=int(float(GETPAR('NusAMOUNT'))*nimax/100.)
    PUTPAR('1 TD',str(ni*2*nbl))

    system=System.getProperty('os.name')
    ext='.exe' if 'Windows' in system else ''
    nussampler=os.path.join(TSPROG,'nussampler'+ext)

```

```

nuslist=os.path.join(VC, 'noah')
dim=2
seed=random.randint(0,1000)
perms=0

sptype='norepeatshuffle'
nn='nn'
zz='0 0'
t2=GETPAR('1 Nust2')+ ' 1'
sw=GETPAR('1 SWH')+ ' 1'
nimax='%i 1'%nimax
ni='%i 1'%ni

nusstr='%s -p "file=%s" "NDIM=%s" "SPARSE=y" "SEED=%s" \
"sptype %s" "f180 %s" "CT_SP %s" "CEXP %s" "phase %s" "Jsp %s"
"T2 %s" \
"FIRST_POINT_ZERO 1" "nholes -1" "SW=%s" "NIMAX= %s" "NIMIN=%s"
"NI=%s"%\

(nussampler,nuslist,dim,seed,sptype,nn,nn,nn,zz,zz,t2,sw,nimax,zz,ni
)

ecode=os.system(nusstr)
if ecode:
    ERRMSG('nussampler exit code %s'%ecode)
    EXIT()

PUTPAR('VCLIST','noah')
return

def converpp():
    """
    Convert NOAH pulse sequence to NUS compatible
    """
    ppnam=GETPAR('PULPROG')
    if ppnam.endswith('.nus'):
        return
    sourcedir=getParfileDirs(0)
    ppText=getpulprogtext(ppnam)
    ppText=editNoah(ppText)
    ppnam=ppnam+'.nus'
    savepulprog(ppnam,ppText)

    PUTPAR('PULPROG',ppnam)
    return

def main(argv=None):

    #print('--- noah.py ---')
    converpp()
    makenuslist()
    #print('-----')
    return

if __name__ == "__main__":
    e = main()

```