Electronic Supplementary Material (ESI) for Analyst.

Reliable, high-quality suppression of NMR signals arising from water and macromolecules: application to bio-fluid analysis

Juan A. Aguilar,[1*] Julia Cassani,[1,2] Fay Probert,[3] Jacqueline Palace,[4] Tim D. W. Claridge,[5] Adolfo Botana[6] and Alan M. Kenwright[1]

1 Department of Chemistry. Durham University. South Road, Durham, DH1 3LE (UK).
2 Departamento de Sistemas Biológicos. Universidad Autnoma Metropolitana Unidad. Xochimilco Col. Villa Quietud, CP 04960, D.F. (México).
3 Department of Pharmacology, University of Oxford, Mansfield Road, Oxford, OX1 3QT (UK).
4 Nuffield Department of Clinical Neurosciences, John Radcliffe Hospital, University of Oxford, Level 3, West Wing, Head-ley Way, Oxford, OX3 9DU (UK).
5 Department of Chemistry, University of Oxford, Chemistry Research Laboratory, Mansfield Road, Oxford, OX1 3TA (UK).
6 JEOL House, Silvert Court, Watchmead Welwyn Garden City, Herts AL7 1LT (UK).

This supplementary information section contains:

- The Robust-5 code for Varian/Agilent, Bruker and Jeol spectrometers.

- The Wasted-I and II code for Varian/Agilent, Bruker and Jeol spectrometers.

Robust-5 code for Varian/Agilent spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Robust-5 for Varian/Agilent spectrometers START: remove this line */

```
#ifndef LINT
static char SCCSid[] = "@(#)Robust5";
#endif

/* J. A. Aguilar. Durham University (UK). 20/12/2012.
   j.a.aguilar@durham.ac.uk

This code is provided as a record of the pulse programmes used to obtain the spectra reported
in this publication.  There is no warranty implied or explicit) that it is optimal or bug-
free.  Anyone using this code does so at their own risk.

gt2 - gradient duration (1 ms)
gzlvl1 - gradient power for the 1st solvent suppression echo
gzlvl2 - gradient power for the 2nd solvent suppression echo
A good gzlvl1/gzlvl2= 5.8. See below.
gstab  - gradient stabilization delay (1 ms)
pe - flag for perfect echo pulse. Default='y'
d3 - delay in WATERGATE. Position of the secondary notches=1/d3 Hz from the centre of the
spectrum.
alt_grd. Flag to alternate the pulsed field gradients every other scan. Default='y'. This
seems to improve the suppression slightly. To determine whether your spectrometer benefits
from this option, just run alt_grd='y','n, 'y','n'

Typical conditions as used in the manuscript:

The phase cycle requires 32 scans and that is what we recommend, but often, a minimum of 8
produces good results. Always set the number of scans to a multiple of two.

It is important, if very high levels of suppression are sought, to optimize the duration and
strength of the pulsed field gradients, as well as its ratio. We have found that a ratio of
5.8 between the first and the second gradient pairs produces excellent results when 1 ms long
pulses of 28.3 G cm-1 (first pair) are used. The optimum gradient stabilization delay varies
among probes but 0.5 to 1 ms delays are often adequate. It was found that in order to minimize
signal distortions the pre-focusing gradient pules and the first radio-frequency pulse should
be separated by, at least, 1.5 ms, although this may vary among probes. Finally, alternating
the polarity of the gradient pulses every other scan seems to slightly improve results. Again,
the latter is probably probe dependent.

Because the W5 elements produces extra notches separated by 1/d3 Hz, the W5 inter-pulse delay
(d3) was set to 240 µs when using the 600 MHz spectrometer, and to 287 when using the 700 MHz
one.

*/

#include <standard.h>
pulsesequence()
{
static int     ph1[2]  = {0,2},
               ph2[32] = {0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3},
               ph3[32] = {2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1},
               ph4[1]  = {1},
               ph5[8]  = {0,0,1,1,2,2,3,3},
               ph6[8]  = {2,2,3,3,0,0,1,1},
               rec[16] = {0,2,2,0,0,2,2,0,2,0,0,2,2,0,0,2},
               ph9[2] = {0,1};

       double  gstab = getval("gstab"),
               gzlvl1 = getval("gzlvl1"),
               gt2 = getval("gt2"),
               gzlvl2 = getval("gzlvl2"),
               d3 = getval("d3"),
               mult=getval("mult"),
               ddrtc = getval("ddrtc");
```

```
        char    sspul[MAXSTR],
                pe[MAXSTR],
                alt_grd[MAXSTR];

        rof1 = getval("rof1"); if(rof1 > 2.0e-6) rof1 = 2.0e-6;
        getstr("pe",pe);
        getstr("sspul",sspul);
        getstr("lockcomp",lockcomp);
        getstr("alt_grd",alt_grd);


        settable(t1,2,ph1);
        settable(t2,32,ph2);
        settable(t3,32,ph3);
        settable(t4,1,ph4);
        settable(t5,8,ph5);
        settable(t6,8,ph6);
        settable(t7,16,rec);
        mod2(ct,v9);

status(A);


delay(rof1);

     delay(d1-gt2*4.0-gstab*2.0);
     obspower(tpwr);
     obsoffset(tof);

if (alt_grd[0] == 'y')
{
ifzero(v9); rgradient('z',gzlvl1);
elsenz(v9); rgradient('z',-1.0*gzlvl1);
endif(v9);
}
else  rgradient('z',gzlvl1);
      delay(2.0*gt2);

if (alt_grd[0] == 'y')
{
ifzero(v9); rgradient('z',gzlvl2);
elsenz(v9); rgradient('z',-1.0*gzlvl2);
endif(v9);
}
else  rgradient('z',gzlvl2);
       delay(2.0*gt2);
       rgradient('z',0);
       delay(1.5e-3);

status(B);

        rgpulse(pw, t1, rof1, rof1);          /* 90d initial excitation pulse */

if (alt_grd[0] == 'y')
{
ifzero(v9); zgradpulse(-1.0*gzlvl1,gt2);
elsenz(v9); zgradpulse(gzlvl1,gt2);
endif(v9);
}
else zgradpulse(-1.0*gzlvl1,gt2);
                                                    /* First WATERGATE */
       delay(gstab-rof1-rof1);
       rgpulse(pw*0.087,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.087 + pw*0.206)/3.1416);
       rgpulse(pw*0.206,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.413)/3.1416);
       rgpulse(pw*0.413,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.778)/3.1416);
       rgpulse(pw*0.778,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.778 + pw*1.491)/3.1416);
       rgpulse(pw*1.491,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*1.491 + pw*1.491)/3.1416);
       rgpulse(pw*1.491,t3,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*1.491 + pw*0.778)/3.1416);
       rgpulse(pw*0.778,t3,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.778 + pw*0.413)/3.1416);
       rgpulse(pw*0.413,t3,rof1,rof1);
```

```
       delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.206)/3.1416);
       rgpulse(pw*0.206,t3,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.087)/3.1416);
       rgpulse(pw*0.087,t3,rof1,rof1);

if (alt_grd[0] == 'y')
{
ifzero(v9); zgradpulse(-1.0*gzlvl1,gt2);
elsenz(v9); zgradpulse(gzlvl1,gt2);
endif(v9);
}
else zgradpulse(-1.0*gzlvl1,gt2);
       delay(gstab-rof1);

if (pe[A] == 'y')                                    /* Use perfect echo pulse */
  {
       rgpulse(pw,t4,rof1,rof1);                     /* 90d refocusing pulse */
   }

if (alt_grd[0] == 'y')
{
ifzero(v9); zgradpulse(-1.0*gzlvl2,gt2);
elsenz(v9); zgradpulse(gzlvl2,gt2);
endif(v9);
}
else zgradpulse(-1.0*gzlvl2,gt2);
                                                     /* Second WATERGATE */
       delay(gstab-rof1-rof1);
       rgpulse(pw*0.087,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.087 + pw*0.206)/3.1416);
       rgpulse(pw*0.206,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.413)/3.1416);
       rgpulse(pw*0.413,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.778)/3.1416);
       rgpulse(pw*0.778,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.778 + pw*1.491)/3.1416);
       rgpulse(pw*1.491,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*1.491 + pw*1.491)/3.1416);
       rgpulse(pw*1.491,t6,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*1.491 + pw*0.778)/3.1416);
       rgpulse(pw*0.778,t6,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.778 + pw*0.413)/3.1416);
       rgpulse(pw*0.413,t6,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.206)/3.1416);
       rgpulse(pw*0.206,t6,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.087)/3.1416);
       rgpulse(pw*0.087,t6,rof1,rof2);


if (alt_grd[0] == 'y')
{
ifzero(v9); zgradpulse(-1.0*gzlvl2,gt2);
elsenz(v9); zgradpulse(gzlvl2,gt2);
endif(v9);
}
else zgradpulse(-1.0*gzlvl2,gt2);
       delay(gstab-rof2+ddrtc);

setreceiver(t7);
status(C);
}
```

/* Robust-5 for Varian/Agilent spectrometers END: remove this line */

Robust-5 code for Bruker spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Robust-5 for Bruker spectrometers START: remove this line */

```
;Robust5.pp
;avance-version (20/12/2012)

;Robust NMR water signal suppression for demanding analytical applications
;Juan A. Aguilar and Simon. J. Kenwright
;Analyst, 141, pp236-242 (2016)
;doi: 10.1039/c5an02121a

;Excitation scilpting W5 reference:
;M. Liu, X. Mao, C. He, H. Huang, J.K. Nicholson & J.C. Lindon,
;J. Magn. Reson. 132, 125 - 129 (1998)

;Using the W5 time correction of :
;Wang et al  J. Magn. Reson. 206, 205 - 209 (2010)


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


#include <Avance.incl>
#include <Grad.incl>
"p20=p16*2"
"d19= (1/(2*cnst10))"
"p27=p1"


"d20=(d19*2)- (2*(p27*0.087 + p27*0.206)/3.1416)"
"d21=(d19*2)- (2*(p27*0.413 + p27*0.206)/3.1416)"
"d22=(d19*2)- (2*(p27*0.413 + p27*0.778)/3.1416)"
"d23=(d19*2)- (2*(p27*0.778 + p27*1.491)/3.1416)"
"d24=(d19*2)- (2*(p27*1.491 + p27*1.491)/3.1416)"
"d25=(d19*2)- (2*(p27*0.778 + p27*1.491)/3.1416)"
"d26=(d19*2)- (2*(p27*0.413 + p27*0.778)/3.1416)"
"d27=(d19*2)- (2*(p27*0.413 + p27*0.206)/3.1416)"
"d28=(d19*2)- (2*(p27*0.087 + p27*0.206)/3.1416)"
"acqt0=-4u"

1 ze
2 30m
  d1
  50u UNBLKGRAD
  p20:gp3*EA
  d16
  d16
  p20:gp4*EA
  d16
  d16
  10u pl1:f1


  p1 ph1


  p16:gp1*EA
  d16 pl18:f1
  p27*0.087 ph3
  d20
  p27*0.206 ph3
  d21
  p27*0.413 ph3
```

```
  d22
  p27*0.778 ph3
  d23
  p27*1.491 ph3
  d24
  p27*1.491 ph4
  d25
  p27*0.778 ph4
  d26
  p27*0.413 ph4
  d27
  p27*0.206 ph4
  d28
  p27*0.087 ph4
  p16:gp1*EA
  d16
  p1 ph2

  p16:gp2*EA
  d16
  p27*0.087 ph5
  d20
  p27*0.206 ph5
  d21
  p27*0.413 ph5
  d22
  p27*0.778 ph5
  d23
  p27*1.491 ph5
  d24
  p27*1.491 ph6
  d25
  p27*0.778 ph6
  d26
  p27*0.413 ph6
  d27
  p27*0.206 ph6
  d28
  p27*0.087 ph6
  p16:gp2*EA
  d16 igrad EA

  4u BLKGRAD
  go=2 ph31
  30m mc #0 to 2 F0(zd)
exit

ph1=0 2
ph2=1
ph3=0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
ph4=2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
ph5=0 0 1 1 2 2 3 3
ph6=2 2 3 3 0 0 1 1
ph31=0 2 2 0 0 2 2 0 2 0 0 2 2 0 0 2

;pl1 : f1 channel - power level for pulse (default)
;p1 : f1 channel -  90 degree high power pulse
;p16: homospoil/gradient pulse. 1 ms
;d1 : relaxation delay; 1-5 * T1
;d16: delay for homospoil/gradient recovery. 1 ms
;d19: delay for W5 binomial water suppression
;d19 = (1/(2*d)), d = distance of next null (in Hz). Calculated using cnst10.
;cnst10: distance to null in Hz for W5 binomial water suppression
;        The further apart the notches, they wider they become
;ns: 8 * n, total number of scans: NS * TD0
;ds: 4; Newer use an odd number of scans or the residual signals will be huge.
;DO NOT SPIN THE SAMPLE


;use gradient ratio:    gp 1 : gp 2 : gp 3 : gp 4
;                                    62 :   10.68 : -62 : -10.68
;for z-only gradients:
;gpz1: 53%
;gpz2: 9.14%
;gpz1: -53%
;gpz2: -9.14%
```

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

;$Id: Robust5.pp,v 1.10 20/12/2012. 17:49:32 ber Exp $
```

/* Robust-5 for Bruker spectrometers END: remove this line */

Robust-5 code for Jeol spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Robust-5 for Jeol spectrometers START: remove this line */

```
------------------------------------------------------------------------------
--                                                                          --
--                          Experiment Source Code                          --
--              Delta NMR Experiment & Machine Control Interface            --
--                                                                          --
--                                                                          --
--                                                                          --
--                                                                          --
------------------------------------------------------------------------------
-- HELP.eng: Robust-5
-- Category: 1D, water_suppression, liquids
-- File name : Robust5
--
-- Sequence name : Robust-5
--
-- Reference
-- Robust NMR water signal suppression for demanding analytical applications
-- Juan A. Aguilar and Simon. J. Kenwright
-- Analyst, 2016,141, 236-242
-- doi: 10.1039/c5an02121a
--
-- Parameters
-- x_pulse              : 90[deg] pulse width
-- x_atn                : attenuator of x_pulse
--
--
-- relaxation_delay     : inter-pulse delay
-- repetition_time      : pulse repetition_time (= relaxation_delay+x_acq_time)
--
-- 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
--
-- delta                : diffusion-encoding gradient pulse width
-- g                    : diffusion-encoding gradient pulse amplitude
-- grad_1               : pulse width of PFG1
-- grad_1_amp           : z-axis amplitude of PFG1
-- grad_shape_type      : shape of PFG pulse
-- grad_recover         : gradient recovery time
--
--
--
-- END HELP

header
    filename  => "Robust5";
    sample_id => "";
    comment   => "Robust-5 NMR pulse sequence for solvent suppression";
    process   =  "proton.list";
    include "header";
end header;

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

acquisition
    x_domain         => "Proton";
    x_offset         => 4.68[ppm];
    x_sweep          => 12[ppm];
```

```
    x_points         => 16384;
    scans            => 32;
    x_prescans       => 4;
    mod_return       => 1;
    include "acquisition";
end acquisition;

pulse
    include "watergate_hard";

    collect COMPLEX,OBS;

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

    grad_1       =? 1.0[ms], help "duration of grad_1";
    grad_2       =? 1.0[ms], help "duration of grad_2";
    gradient_max => z_gradient_max, help "Maximum amplitude for a given probe as defined in
the probe file";
    Note         =? "Do not exceed gradient_max value for grad_1_amp";

    grad_1_amp   => 250[mT/m], help "Amplitude first gradient pair in Tesla/meter units";
    grad_2_amp   => 60[mT/m], help "Amplitude second gradient pair in Tesla/meter units";
    grad_shape   => "SQUARE",("SQUARE","SINE"), help "shape of grad_1";
    grad_recover => 1[ms], help "gradient recovery time";


comment_312 =? "*** Watergate hard solvent suppression ***";
    --watergate parameters
    wgh_x_pulse         => x90, help "90[deg] pulse used for watergate composite pulse";
    include "services";
    xfreq               =?  _get_freq( x_domain );
    wgh_null            => 4000[Hz], help "null frequency";
    wgh_null_ppm        =? (wgh_null/xfreq)*1000000[ppm];
    wgh_tau             =? 1/wgh_null, help "wagtergate pulse interval 1/wgh_null";


comment_7 =? "*** Pulse Delay ***";
    initial_wait     =  1.0[s];
    relaxation_delay => 1.0[s], help "inter-pulse delay";

    include "pulse";


    phase_1   = {0,180};
    phase_2   = {90};
    phase_wgh1 = {8(0), 8(90), 8(180), 8(270)};
    phase_wgh2 = {2(0), 2(90), 2(180), 2(270)};
    phase_acq = {0,180,180,0,0,180,180,0,180,0,0,180,180,0,0,180};


begin
    initial_wait;
    relaxation_delay;
   -- Lock prefocussing
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_1_amp);
    grad_recover;
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_1_amp);
    grad_recover;
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_2_amp);
    grad_recover;
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_2_amp);
    grad_recover;
    grad_recover;
        grad_recover;
        grad_recover;

   -- Perfect echo
    x_pulse, (obs.gate, obs.phs.phase_1, obs.atn.x_atn);
    grad_1, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_1_amp);
    grad_recover;

    wgh5_comp180(wgh_x_pulse, phase_wgh1, x_atn, wgh_tau);

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

```
    x_pulse, (obs.gate, obs.phs.phase_2, obs.atn.x_atn);
    grad_2, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_2_amp);
    grad_recover;

    wgh5_comp180(wgh_x_pulse, phase_wgh2, x_atn, wgh_tau);

    grad_2, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_2_amp);
    grad_recover;

    acq (dead_time, delay, phase_acq);
end pulse;
```

/* Robust-5 for Jeol spectrometers END: remove this line */

Wasted-I code for Varian/Agilent spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Wasted-I for Varian/Agilent spectrometers START: remove this line */

```
#ifndef LINT
static char SCCSid[] = "@(#)Wasted_I";
#endif

/* J. A. Aguilar. Durham. 20/12/2012.

       gt1            - gradient duration for the solvent suppression echo
        gzlvl1         - gradient power for the solvent suppression echo
       gt2            - gradient duration for the solvent suppression echo
        gzlvl2         - gradient power for the solvent suppression echo
        gstab          - gradient stabilization delay
        d3             - determines the location the secondary notches (at 1/d3 Hz from the
centre of the spectrum). The further apart, the wider the notches.
        tof            - transmitter offset (set on resonance for H2O)
       slock          - spinlock (y/n). Default='y'. Helps keeping the water signal small.
       d3             - creates extra notches at a distance delta=1/d3 Hz from the centre of
the window. The further apart the notches, the wider they become
       cycles         - 1 cycle = 2 spin echoes
       cycle_time     - 4 times tau (2 spin echoes, see paper). The duration for the whole
cycle (2 spin echoes) is. Keep cycle_time << 0.25/(2PiJ).

Typical conditions as used in the manuscript:
Always set the number of scans to a multiple of two, but the phase cycle requires 32 scans to
produce clean results and that is what we recommend.
If very high levels of suppression are sought, it is important to optimize the duration and
strength of the pulsed field gradients, as well as their ratios.
We have found that a ratio of 5.8 between the first and the second gradient pairs produces
excellent results when 1 ms long pulses of 28.3 G cm-1 (first pair) are used.
The optimum gradient stabilization delay varies among probes but 0.5 to 1 ms delays are often
adequate.

-- Reference
-- Robust NMR water signal suppression for demanding analytical applications
-- Juan A. Aguilar and Alan. J. Kenwright
-- Analyst (2016),141, 236-242
-- doi: 10.1039/C5AN02121A

-- Spin echo NMR spectra without J modulation
-- A Aguilar, Juan & Nilsson, Mathias & Bodenhausen, Geoffrey & A Morris, Gareth.
-- Chemical communications (2011), 48, 811-3. DOI: 10.1039/c1cc16699a.

*/


#include <standard.h>
pulsesequence()
{

static int     ph1[2]  = {0,2},
               ph2[32] = {0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3},
               ph3[32] = {2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1},
               ph4[1]  = {1},
               ph5[8]  = {0,0,1,1,2,2,3,3},
               ph6[8]  = {2,2,3,3,0,0,1,1},
               rec[16] = {0,2,2,0,0,2,2,0,2,0,0,2,2,0,0,2};

        double  gstab = getval("gstab"),
               gt1 = getval("gt1"),
               gt2 = getval("gt2"),
                     gzlvl1 = getval("gzlvl1"),
                     gzlvl2 = getval("gzlvl2"),
                     gzlvl3 = getval("gzlvl3"),
```

```
               d3 = getval("d3"),     /* Creates extra notches at a distance delta=1/d3 Hz
from the center of the window. The further apart the notches, the wider they become */

               atfrq = getval("satfrq"),
               cycles = getval("cycles"),      /* 1 cycle = 2 spin echoes */
               cycle_time = getval("cycle_time"), /* cycle_time (4 time tau) = time duration
for the whole cycle (2 spin echoes). << 0.25/(2PiJ). Usually 4 ms, but up to 20 ms can be used
sometimes  */
               ddrtc = getval("ddrtc");


        char    sspul[MAXSTR],
                wg[MAXSTR],
                pe[MAXSTR],
                slock[MAXSTR];

        rof1 = getval("rof1"); if(rof1 > 2.0e-6) rof1 = 2.0e-6;
        getstr("wg",wg);
        getstr("pe",pe);
        getstr("sspul",sspul);
        getstr("slock",slock);


        settable(t1,2,ph1);
        settable(t2,32,ph2);
        settable(t3,32,ph3);
        settable(t4,1,ph4);
        settable(t5,8,ph5);
        settable(t6,8,ph6);
        settable(t7,16,rec);

  cycles=cycles-2;
  initval(cycles,v5);

status(A);

delay(rof1);

        if (satmode[0] == 'y')         /* This option has not been tested */
       {
        if (d1 - satdly > 0)
          delay(d1 - satdly);
        else
        delay(0.02);
        obspower(satpwr);
         if (satfrq != tof)
          obsoffset(satfrq);
         rgpulse(satdly,zero,rof1,rof1);
         if (satfrq != tof)
          obsoffset(tof);
        obspower(tpwr);
        delay(1.0e-5);
       }
      else
      {  delay(d1); }

   if (getflag("wet"))        /* This option has not been tested */
     wet4(zero,one);

status(B);

        rgpulse(pw, t1, rof1, rof1);                                                    /* 90d
initial excitation pulse */

  /*  ************** 1st cycle START *********************/

      delay(cycle_time*0.25-rof1-rof1);
          rgpulse(pw*2.0,zero,rof1,rof1);                            /* 180d */
      delay(cycle_time*0.25-rof1-rof1);

           rgpulse(pw,t4,rof1,rof1);                         /* 90d refocusing pulse */

      delay(cycle_time*0.25-rof1-rof1);
          rgpulse(pw*2.0,zero,rof1,rof1);                       /* 180d */
      delay(cycle_time*0.25-rof1);

/*  ************** 1st cycle END  *********************/
```

```
/*  ************** Core cycles START ********************/

starthardloop(v5);


      delay(cycle_time*0.25-rof1);
          rgpulse(pw*2.0,zero,rof1,rof1);                    /* 180d */
      delay(cycle_time*0.25-rof1-rof1);

                    rgpulse(pw,t4,rof1,rof1);                 /* 90d refocusing pulse */

      delay(cycle_time*0.25-rof1-rof1);
          rgpulse(pw*2.0,zero,rof1,rof1);                    /* 180d */
      delay(cycle_time*0.25-rof1);


endhardloop();

/*  ************** Core cycles END ********************/

/*  ************** Last cycle START ********************/

     delay(cycle_time*0.25-rof1);
         rgpulse(pw*2.0,zero,rof1,rof1);                              /* 180d */
     delay(cycle_time*0.25-rof1-rof1);

                    rgpulse(pw,t4,rof1,rof1);                 /* 90d refocusing pulse */

     delay(cycle_time*0.25-rof1-rof1);
         rgpulse(pw*2.0,zero,rof1,rof1);                       /* 180d */
     delay(cycle_time*0.25-rof1);

if (slock[0] == 'y') {
         rgpulse(pw*150,t4,rof1,rof1);                         /* Spin-Lock */
}

/*  ************** Last cycle END ********************/

/*  ************** WATERGATE ********************/


        zgradpulse(gzlvl1,gt1);
        delay(gstab-rof1-rof1);
                                                  /* First WATERGATE */
        rgpulse(pw*0.087,t2,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.087 + pw*0.206)/3.1416);
        rgpulse(pw*0.206,t2,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.413)/3.1416);
        rgpulse(pw*0.413,t2,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.778)/3.1416);
        rgpulse(pw*0.778,t2,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.778 + pw*1.491)/3.1416);
        rgpulse(pw*1.491,t2,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*1.491 + pw*1.491)/3.1416);
        rgpulse(pw*1.491,t3,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*1.491 + pw*0.778)/3.1416);
        rgpulse(pw*0.778,t3,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.778 + pw*0.413)/3.1416);
        rgpulse(pw*0.413,t3,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.206)/3.1416);
        rgpulse(pw*0.206,t3,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.087)/3.1416);
        rgpulse(pw*0.087,t3,rof1,rof1);

        zgradpulse(gzlvl1,gt1);
        delay(gstab-rof1);

        rgpulse(pw,t4,rof1,rof1);

        zgradpulse(gzlvl2,gt1);                               /* Second WATERGATE */
        delay(gstab-rof1-rof1);

        rgpulse(pw*0.087,t5,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.087 + pw*0.206)/3.1416);
        rgpulse(pw*0.206,t5,rof1,rof1);
```

```
        delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.413)/3.1416);
        rgpulse(pw*0.413,t5,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.778)/3.1416);
        rgpulse(pw*0.778,t5,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.778 + pw*1.491)/3.1416);
        rgpulse(pw*1.491,t5,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*1.491 + pw*1.491)/3.1416);
        rgpulse(pw*1.491,t6,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*1.491 + pw*0.778)/3.1416);
        rgpulse(pw*0.778,t6,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.778 + pw*0.413)/3.1416);
        rgpulse(pw*0.413,t6,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.206)/3.1416);
        rgpulse(pw*0.206,t6,rof1,rof1);
        delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.087)/3.1416);
        rgpulse(pw*0.087,t6,rof1,rof2);

        zgradpulse(gzlvl2,gt1);
        delay(gstab-rof2+ddrtc);


setreceiver(t7);

status(C);

}
```

/* Wasted-I for Varian/Agilent spectrometers END: remove this line */

Wasted-II code for Varian/Agilent spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Wasted-II for Varian/Agilent spectrometers START: remove this line */

```
#ifndef LINT
static char SCCSid[] = "Painles_II";
#endif

/* J. A. Aguilar. Durham. 20/12/2012.

       gt1            - gradient duration for the solvent suppression echo
        gzlvl1         - gradient power for the solvent suppression echo
       gt2            - gradient duration for the solvent suppression echo
        gzlvl2         - gradient power for the solvent suppression echo
        gstab          - gradient stabilization delay
        d3             - determines the location the secondary notches (at 1/d3 Hz from the
centre of the spectrum). The further apart, the wider the notches.
        tof            - transmitter offset (set on resonance for H2O)
       slock          - 1st spinlock (y/n). Default='y'. Helps keeping the water signal small
       slock_a        - 2nd spinlock (y/n). Default='y'. Eliminates dispersive signals that
could have been produced by COSY-TYPE transfers or by the use of long echo times
       d3             - creates extra notches at a distance delta=1/d3 Hz from the centre of
the window. The further apart the notches, the wider they become
       cycles         - 1 cycle = 2 spin echoes
       cycle_time     - 4 times tau (2 spin echoes, see paper). The duration for the whole
cycle (2 spin echoes) is. Keep cycle_time << 0.25/(2PiJ).

Typical conditions as used in the manuscript:
Always set the number of scans to a multiple of two, but the phase cycle requires 32 scans to
produce clean results and that is what we recommend.
If very high levels of suppression are sought, it is important to optimize the duration and
strength of the pulsed field gradients, as well as their ratios.
We have found that a ratio of 5.8 between the first and the second gradient pairs produces
excellent results when 1 ms long pulses of 28.3 G cm-1 (first pair) are used.
The optimum gradient stabilization delay varies among probes but 0.5 to 1 ms delays are often
adequate.
It was found that in order to minimize signal distortions the pre-focusing gradient pules and
the first radio-frequency pulse should be separated by at least 1.5 ms,
although this may vary among probes.
Finally, alternating the polarity of the gradient pulses every other scan seems to slightly
improve results. Again, the latter is probably probe dependent.

-- Reference
-- Robust NMR water signal suppression for demanding analytical applications
-- Juan A. Aguilar and Alan. J. Kenwright
-- Analyst (2016),141, 236-242
-- doi: 10.1039/C5AN02121A

-- Spin echo NMR spectra without J modulation
-- A Aguilar, Juan & Nilsson, Mathias & Bodenhausen, Geoffrey & A Morris, Gareth.
-- Chemical communications (2011), 48, 811-3. DOI: 10.1039/c1cc16699a.

*/

#include <standard.h>

pulsesequence()
{
static int     ph1[2]  = {0,2},
               ph2[32] = {0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3},
               ph3[32] = {2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1},
               ph4[1]  = {1},
               ph5[8]  = {0,0,1,1,2,2,3,3},
               ph6[8]  = {2,2,3,3,0,0,1,1},
               rec[16] = {0,2,2,0,0,2,2,0,2,0,0,2,2,0,0,2},
               ph9[2] = {0,1};
```

```
        double  gstab = getval("gstab"),
                gt0 = getval("gt0"),
                gt1 = getval("gt1"),
                gt2 = getval("gt2"),
                gzlvl0 = getval("gzlvl0"),
                gzlvl1 = getval("gzlvl1"),
                gzlvl2 = getval("gzlvl2"),
                gzlvl3 = getval("gzlvl3"),
                d3 = getval("d3"),     /* second notch at delta=1/d3 */
                Dtau,Ddelta,Deltac,dosytimecubed, dosyfrq,del,
                atfrq = getval("satfrq"),
                cycles = getval("cycles"),      /* 1 cycle = 2 spin echoes = 2 diffusion
encoding periods. Keep in mind that each cycle uses 4 gradients, so keep it to a minumum. It
also helps keeping signal. */
                cycle_time = getval("cycle_time"), /* cycle_time= time duration for the whole
cycle (2 spin echoes). 20-30, even 40 ms can be used depending on the actual coupling
constants. Not the whole cycle_time is used for diffusion encoding, as gstabs are used   */
                ddrtc = getval("ddrtc"),
                itof = getval("itof");



        char    lockcomp[MAXSTR],
                sspul[MAXSTR],
                slock[MAXSTR],
                slock_a[MAXSTR],
                alt_grd[MAXSTR];

        rof1 = getval("rof1"); if(rof1 > 2.0e-6) rof1 = 2.0e-6;
        getstr("sspul",sspul);
        getstr("lockcomp",lockcomp);
        getstr("slock",slock);
        getstr("slock_a",slock_a);
        getstr("alt_grd",alt_grd);

        settable(t1,2,ph1);
        settable(t2,32,ph2);
        settable(t3,32,ph3);
        settable(t4,1,ph4);
        settable(t5,8,ph5);
        settable(t6,8,ph6);
        settable(t7,16,rec);

        mod2(ct,v9);

   dosyfrq = sfrq;
   cycles=cycles-2;
   initval(cycles,v5);

status(A);

delay(rof1);

        if (satmode[0] == 'y')                /* This has not been tested */
       {
        if (d1 - satdly > 0)
          delay(d1 - satdly);
        else
        delay(0.02);
        obspower(satpwr);
        obsoffset(satfrq);
         rgpulse(satdly,zero,rof1,rof1);

        delay(1.0e-5);
       }
     else
     {      delay(d1-gt2*4.0-gstab*2.0);  }

       obspower(tpwr);
       obsoffset(tof);

if (alt_grd[0] == 'y')
{
ifzero(v9); rgradient('z',gzlvl1);
elsenz(v9); rgradient('z',-1.0*gzlvl1);
endif(v9);
}
```

```
else  rgradient('z',gzlvl1);

       delay(2.0*gt2);

if (alt_grd[0] == 'y')
{
ifzero(v9); rgradient('z',gzlvl2);
elsenz(v9); rgradient('z',-1.0*gzlvl2);
endif(v9);
}
else  rgradient('z',gzlvl2);

       delay(2.0*gt2);

       rgradient('z',0);
       delay(1.5e-3);

status(B);

       rgpulse(pw, t1, rof1, rof1);                /* 90d initial excitation pulse */

if (alt_grd[0] == 'y')
{
ifzero(v9); zgradpulse(-1.0*gzlvl1,gt2);
elsenz(v9); zgradpulse(gzlvl1,gt2);
endif(v9);
}
else zgradpulse(-1.0*gzlvl1,gt2);

       delay(gstab-rof1-rof1);
                                                          /* 1st WATERGATE */
       rgpulse(pw*0.087,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.087 + pw*0.206)/3.1416);
       rgpulse(pw*0.206,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.413)/3.1416);
       rgpulse(pw*0.413,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.778)/3.1416);
       rgpulse(pw*0.778,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.778 + pw*1.491)/3.1416);
       rgpulse(pw*1.491,t2,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*1.491 + pw*1.491)/3.1416);
       rgpulse(pw*1.491,t3,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*1.491 + pw*0.778)/3.1416);
       rgpulse(pw*0.778,t3,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.778 + pw*0.413)/3.1416);
       rgpulse(pw*0.413,t3,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.206)/3.1416);
       rgpulse(pw*0.206,t3,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.087)/3.1416);
       rgpulse(pw*0.087,t3,rof1,rof1);

if (alt_grd[0] == 'y')
{
ifzero(v9); zgradpulse(-1.0*gzlvl1,gt2);
elsenz(v9); zgradpulse(gzlvl1,gt2);
endif(v9);
}
else zgradpulse(-1.0*gzlvl1,gt2);
       delay(gstab-rof1);

       rgpulse(pw,t4,rof1,rof1);            /* 90d refocusing pulse, perfect echo */

if (alt_grd[0] == 'y')
{
ifzero(v9); zgradpulse(-1.0*gzlvl2,gt2);
elsenz(v9); zgradpulse(gzlvl2,gt2);
endif(v9);
}
else zgradpulse(-1.0*gzlvl2,gt2);            /* Second WATERGATE */
       delay(gstab-rof1-rof1);

       rgpulse(pw*0.087,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.087 + pw*0.206)/3.1416);
       rgpulse(pw*0.206,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.413)/3.1416);
       rgpulse(pw*0.413,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.778)/3.1416);
```

```
       rgpulse(pw*0.778,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.778 + pw*1.491)/3.1416);
       rgpulse(pw*1.491,t5,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*1.491 + pw*1.491)/3.1416);
       rgpulse(pw*1.491,t6,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*1.491 + pw*0.778)/3.1416);
       rgpulse(pw*0.778,t6,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.778 + pw*0.413)/3.1416);
       rgpulse(pw*0.413,t6,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.413 + pw*0.206)/3.1416);
       rgpulse(pw*0.206,t6,rof1,rof1);
       delay(d3-rof1-rof1-2*(pw*0.206 + pw*0.087)/3.1416);
       rgpulse(pw*0.087,t6,rof1,rof1);

if (alt_grd[0] == 'y')
{
ifzero(v9); zgradpulse(-1.0*gzlvl2,gt2);
elsenz(v9); zgradpulse(gzlvl2,gt2);
endif(v9);
}
else zgradpulse(-1.0*gzlvl2,gt2);
       delay(gstab-rof1);

if (slock[0] == 'y') {
        rgpulse(pw*150,t4,0,0);                        /* SL */
}

  /*  ************** 1st cycle START *********************/

      delay(cycle_time*0.25-rof1-rof1);
         rgpulse(pw*2.0,zero,rof1,rof1);                    /* 180d */
      delay(cycle_time*0.25-rof1-rof1);

           rgpulse(pw,t4,rof1,rof1);                        /* 90d refocusing pulse */


      delay(cycle_time*0.25-rof1-rof1);
         rgpulse(pw*2.0,zero,rof1,rof1);                     /* 180d */
      delay(cycle_time*0.25-rof1);



/*  ************** 1st cycle END  *********************/

/*  ************** Core cycles START *********************/

starthardloop(v5);

      delay(cycle_time*0.25-rof1);
         rgpulse(pw*2.0,zero,rof1,rof1);                    /* 180d */
      delay(cycle_time*0.25-rof1-rof1);

                    rgpulse(pw,t4,rof1,rof1);             /* 90d refocusing pulse */

      delay(cycle_time*0.25-rof1-rof1);
         rgpulse(pw*2.0,zero,rof1,rof1);                     /* 180d */
      delay(cycle_time*0.25-rof1);


endhardloop();

/*  ************** Core cycles END *********************/

/*  ************** Last cycle START *********************/

     delay(cycle_time*0.25-rof1);
        rgpulse(pw*2.0,zero,rof1,rof1);                                 /* 180d */
     delay(cycle_time*0.25-rof1-rof1);

        rgpulse(pw,t4,rof1,rof1);                     /* 90d refocusing pulse */

     delay(cycle_time*0.25-rof1-rof1);
        rgpulse(pw*2.0,zero,rof1,rof1);                         /* 180d */
     delay(cycle_time*0.25-rof1);


/*  ************** Last cycle END *********************/
```

```
if (slock_a[0] == 'y') {
         obsoffset(tof+itof);
         rgpulse(pw*250,one,rof1,rof2);
         obsoffset(tof);}

setreceiver(t7);

status(C);

}
```

/* Wasted-II for Varian/Agilent spectrometers END: remove this line */

Wasted-I code for Bruker spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Wasated-I for Bruker spectrometers START: remove this line */

```
;There is no warranty (implied or explicit) that it is optimal or bug-free.
;Anyone using this code does so at their own risk.

; Wasted-I: Project + Spinlock + Robust5 (no pre-focussing)
;Juan A. Aguilar. 1st draft: 20/12/2012 Durham (UK).
;Avance-version (13/05/16)

;Water suppression using Robust-5
;Broad signal suppression using Project


;Not yet published.

;Robust NMR water signal suppression for demanding analytical applications
;Juan A. Aguilar and Simon. J. Kenwright
;Analyst, 141, pp236-242 (2016)
;doi: 10.1039/C5AN02121A

;Spin echo NMR spectra without J modulation
;Juan A. Aguilar, Mathias Nilsson, Geoffrey Bodenhausen and Gareth A. Morris
;Chem. Commun., 2012,48, 811-813
;doi: 10.1039/C1CC16699A


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


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

"p20=p16*2"
"d19= (1/(2*cnst10))"
"p27=p1"

"d3=l4*d2" ; Total T2 time


"d20=(d19*2)- (2*(p27*0.087 + p27*0.206)/3.1416)"
"d21=(d19*2)- (2*(p27*0.413 + p27*0.206)/3.1416)"
"d22=(d19*2)- (2*(p27*0.413 + p27*0.778)/3.1416)"
"d23=(d19*2)- (2*(p27*0.778 + p27*1.491)/3.1416)"
"d24=(d19*2)- (2*(p27*1.491 + p27*1.491)/3.1416)"
"d25=(d19*2)- (2*(p27*0.778 + p27*1.491)/3.1416)"
"d26=(d19*2)- (2*(p27*0.413 + p27*0.778)/3.1416)"
"d27=(d19*2)- (2*(p27*0.413 + p27*0.206)/3.1416)"
"d28=(d19*2)- (2*(p27*0.087 + p27*0.206)/3.1416)"
"acqt0=0u"

1 ze
2 30m
  d1

 p1 ph1

/******************** PROJECT: START ********************/

3 d2*0.25
  p1*2 ph7
  d2*0.25
```

```
  p1 ph8          /* Perfect echo */

  d2*0.25
  p1*2 ph7
  d2*0.25
  lo to 3 times l4

/******************** PROJECT: END ********************/


p1*250 ph8 /* Spin Lock */


/******************** Robust-5: START ********************/

  50u UNBLKGRAD
  p16:gp1*EA
  d16
  p27*0.087 ph3
  d20
  p27*0.206 ph3
  d21
  p27*0.413 ph3
  d22
  p27*0.778 ph3
  d23
  p27*1.491 ph3
  d24
  p27*1.491 ph4
  d25
  p27*0.778 ph4
  d26
  p27*0.413 ph4
  d27
  p27*0.206 ph4
  d28
  p27*0.087 ph4
  p16:gp1*EA
  d16
  50u

  p1 ph2         /* Perfect echo */

  50u
  p16:gp2*EA
  d16
  p27*0.087 ph5
  d20
  p27*0.206 ph5
  d21
  p27*0.413 ph5
  d22
  p27*0.778 ph5
  d23
  p27*1.491 ph5
  d24
  p27*1.491 ph6
  d25
  p27*0.778 ph6
  d26
  p27*0.413 ph6
  d27
  p27*0.206 ph6
  d28
  p27*0.087 ph6
  p16:gp2*EA
  d16 igrad EA
  50u BLKGRAD


/******************** Robust-5: END ********************/

  go=2 ph31
  30m mc #0 to 2 F0(zd)
exit
```

```
ph1=0 2
ph2=1
ph3=0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
ph4=2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
ph5=0 0 1 1 2 2 3 3
ph6=2 2 3 3 0 0 1 1
ph7=0 /* PROJECT 180 */
ph8=1 /* PROJECT 90 */
ph31=0 2 2 0 0 2 2 0 2 0 0 2 2 0 0 2


;pl1 : f1 channel - power level for pulse (default)
;p1 : f1 channel -  90 degree high power pulse
;p16: homospoil/gradient pulse. 1 ms
;d16: delay for homospoil/gradient recovery. 1 ms
;d1 : relaxation delay; 1-5 * T1

;d19: delay for W5 binomial water suppression. Calculated using cnst10
;cnst10: distance from nulls in Hz for W5 binomial water suppression. The further apart the
notches, the wider they become
;ns: 32 * n, total number of scans: NS * TD0.  Newer use an odd number of scans or the
residual water signal will be huge.

;ds: 4;
;d2 : Cycle time (2 echo times). T2 filter time= d2*l4. d2<< 0.25/(2*Pi*J).
;l4: loop for T2 filter. Min value= 1 (2 echoes). T2 filter time= d2*l4 (not counting the rf
pulses)

; DO NOT SPIN THE SAMPLE

;use gradient ratio:    gp 1 :  gp 2 : gp 3 :   gp 4
;                         62 : 10.68 :  -62 : -10.68

;for z-only gradients:
;gpz1: 62.8%
;gpz2: 10.68%
;gpz3: -62.8%
;gpz2: -10.68%

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


;$Id: PR,v 03 2012/01/31 17:49:32 ber Exp $
```

/* Wasted-I for Bruker spectrometers END: remove this line */

Wasted-II code for Bruker spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Wasted-II for Bruker spectrometers START: remove this line */

```
;There is no warranty (implied or explicit) that it is optimal or bug-free.
;Anyone using this code does so at their own risk.

;Robust5 - spinlock - Project - spinlock
;The fist spinlock help keeping the water signals attenuated.
;The second spinlock eliminates dispersive components caused by the planar black-transfer
(recovery of signal suppressed by Robust5)
;Juan A. Aguilar. 1st draft: 20/12/2012 Durham (UK).
;Avance-version (13/05/16)

;Water suppression using Robust-5
;Broad signal suppression using Project


;Not yet published. Base on the following:

;Robust NMR water signal suppression for demanding analytical applications
;Juan A. Aguilar and Simon. J. Kenwright
;Analyst, 141, pp236-242 (2016)
;doi: 10.1039/C5AN02121A

;Spin echo NMR spectra without J modulation
;Juan A. Aguilar, Mathias Nilsson, Geoffrey Bodenhausen and Gareth A. Morris
;Chem. Commun., 2012,48, 811-813
;doi: 10.1039/C1CC16699A


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


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

"p20=p16*2"
"d19= (1/(2*cnst10))"
"p27=p1"

"d3=l4*d2" ; Total T2 time


"d20=(d19*2)- (2*(p27*0.087 + p27*0.206)/3.1416)"
"d21=(d19*2)- (2*(p27*0.413 + p27*0.206)/3.1416)"
"d22=(d19*2)- (2*(p27*0.413 + p27*0.778)/3.1416)"
"d23=(d19*2)- (2*(p27*0.778 + p27*1.491)/3.1416)"
"d24=(d19*2)- (2*(p27*1.491 + p27*1.491)/3.1416)"
"d25=(d19*2)- (2*(p27*0.778 + p27*1.491)/3.1416)"
"d26=(d19*2)- (2*(p27*0.413 + p27*0.778)/3.1416)"
"d27=(d19*2)- (2*(p27*0.413 + p27*0.206)/3.1416)"
"d28=(d19*2)- (2*(p27*0.087 + p27*0.206)/3.1416)"
"acqt0=0u"

1 ze
2 30m
  d1
/******************** Robust-5: START ********************/


  50u UNBLKGRAD
  p20:gp3*EA
  d16
```

```
  d16
  p20:gp4*EA
  d16
  d16
  10u

  p1 ph1

  4u
  p16:gp1*EA
  d16 pl18:f1
  p27*0.087 ph3
  d20
  p27*0.206 ph3
  d21
  p27*0.413 ph3
  d22
  p27*0.778 ph3
  d23
  p27*1.491 ph3
  d24
  p27*1.491 ph4
  d25
  p27*0.778 ph4
  d26
  p27*0.413 ph4
  d27
  p27*0.206 ph4
  d28
  p27*0.087 ph4
  p16:gp1*EA
  d16
  4u

  p1 ph2         /* Perfect echo */

  4u
  p16:gp2*EA
  d16
  p27*0.087 ph5
  d20
  p27*0.206 ph5
  d21
  p27*0.413 ph5
  d22
  p27*0.778 ph5
  d23
  p27*1.491 ph5
  d24
  p27*1.491 ph6
  d25
  p27*0.778 ph6
  d26
  p27*0.413 ph6
  d27
  p27*0.206 ph6
  d28
  p27*0.087 ph6
  p16:gp2*EA
  d16 igrad EA
  4u BLKGRAD

/******************** Robust-5: END ********************/

p1*150 ph8    /* Spin lock, important to keep the water signal attenuated */

/******************** PROJECT: START ********************/
3 d2*0.25
  p1*2 ph7
  d2*0.25

  p1 ph8          /* Perfect echo */

  d2*0.25
  p1*2 ph7
  d2*0.25
  lo to 3 times l4
```

```
/********************* PROJECT: END *********************/

p1*250 ph8    /* Spin lock, important to suppress dispersive components */

  go=2 ph31
  30m mc #0 to 2 F0(zd)
exit


ph1=0 2
ph2=1
ph3=0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
ph4=2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
ph5=0 0 1 1 2 2 3 3
ph6=2 2 3 3 0 0 1 1
ph7=0 /* PROJECT 180 */
ph8=1 /* PROJECT 90 */
ph31=0 2 2 0 0 2 2 0 2 0 0 2 2 0 0 2


;pl1 : f1 channel - power level for pulse (default)
;p1 : f1 channel -  90 degree high power pulse
;p16: homospoil/gradient pulse. 1 ms
;d16: delay for homospoil/gradient recovery. 1 ms
;d1 : relaxation delay; 1-5 * T1

;d19: delay for W5 binomial water suppression. Calculated using cnst10
;cnst10: distance between nulls in Hz for W5 binomial water suppression. The further apart,
they wider they become.
;ns: 32 * n, total number of scans: NS * TD0.  Newer use an odd number of scans or the
residual water signal will be huge.
;The presence of the 2nd spinlock, necessary to eliminate dispersive components, can introduce
oscillatory components in the residual water signal. Completing the 32 step phase cycle
usually eliminates the problem.

;ds: 4;
;d2 : Cycle time (2 echo times). T2 filter time= d2*l4. d2<< 0.25/(2*Pi*J).
;l4: loop for T2 filter. Min value= 1 (2 echoes). T2 filter time= d2*l4

; DO NOT SPIN THE SAMPLE

;use gradient ratio:    gp 1 :  gp 2 : gp 3 :   gp 4
;                         62 : 10.68 :  -62 : -10.68

;for z-only gradients:
;gpz1: 62.8%
;gpz2: 10.68%
;gpz3: -62.8%
;gpz2: -10.68%

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

/* Wasted-II for Bruker spectrometers END: remove this line */

Wasted-I code for Jeol spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Wasted-I for Jeol spectrometers START: remove this line */

```
------------------------------------------------------------------------------
--                                                                          --
--                         Experiment Source Code                           --
--              Delta NMR Experiment & Machine Control Interface            --
--                                                                          --
--                                                                          --
--                                                                          --
--                                                                          --
------------------------------------------------------------------------------
-- HELP.eng: Wasted-I, combination of Project + Robust-5 for solvent suppression with T2
filter
-- Category: 1D, water_suppression, liquids
-- File name : Wasted-I
-- Sequence name : Wasted-I
--
-- Reference
-- Robust NMR water signal suppression for demanding analytical applications
-- Juan A. Aguilar and Alan. J. Kenwright
-- Analyst (2016),141, 236-242
-- doi: 10.1039/C5AN02121A
--
-- Spin echo NMR spectra without J modulation
-- A Aguilar, Juan & Nilsson, Mathias & Bodenhausen, Geoffrey & A Morris, Gareth.
-- Chemical communications (2011), 48, 811-3. DOI: 10.1039/c1cc16699a.
--
--
-- END HELP

header
    filename  => "Wasted-I";
    sample_id => "";
    comment   => "Wasted-I, Project+Robust-5";
    process   =  "proton.list";
    include "header";
end header;

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

acquisition
    x_domain         => "Proton";
    x_offset         => 4.68[ppm];
    x_sweep          => 12[ppm];
    x_points         => 16384;
    scans            => 32;
    x_prescans       => 4;
    mod_return       => 1;
    include "acquisition";
end acquisition;

pulse
    include "watergate_hard";

    collect COMPLEX,OBS;

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

    grad_1       =? 1.0[ms], help "duration of grad_1";
    grad_2       =? 1.0[ms], help "duration of grad_2";
    gradient_max => z_gradient_max, help "Maximum amplitude for a given probe as defined in
the probe file";
```

```
    Note         =? "Do not exceed gradient_max value for grad_1_amp";

    grad_1_amp   => 250[mT/m], help "Amplitude first gradient pair in Tesla/meter units";
    grad_2_amp   => 60[mT/m], help "Amplitude second gradient pair in Tesla/meter units";
    grad_shape   => "SINE",("SQUARE","SINE"), help "shape of grad_1";
    grad_recover => 1[ms], help "gradient recovery time";

        spinlock_enable => TRUE, help "use spinlock pulses flanking PROJECT";
        x_spinlock_pulse => x90_spin, help "spin lock 90 pulse width";
    x_spinlock_atn   => xatn_spin, help "attenuator of x_spinlock_pulse";
        sl2                  =>64, help "SL multiplier 1";

comment_2 =? "*** T2 filter ***";
       delay_list       => 0.1[s],                       help "delay time(array parameter),
must be tau_step unit step";
    tau_step         => 20[ms],                      help "tau interval minimum time ";
    tau_interval     =? (tau_step - x_pulse * 5) / 4, help "tau_interval (=(tau_step-x180)2)";
    loop_number      =? round(delay_list / tau_step), help "calculation of loop number (=
round(delay_list/tau_step))";
    calc_delay       =? loop_number * tau_step,       help "real delay time = loop_number *
tau_step";
       t2_addition           =? x_spinlock_pulse*(sl2), help "additional time from spin_lock
time that contributes to T1rho relaxation";


comment_312 =? "*** Watergate hard solvent suppression ***";
    --watergate parameters
    wgh_x_pulse         => x90, help "90[deg] pulse used for watergate composite pulse";
    include "services";
    xfreq               =?  _get_freq( x_domain );
    wgh_null            => 4000[Hz], help "null frequency";
    wgh_null_ppm        =? (wgh_null/xfreq)*1000000[ppm];
    wgh_tau             =? 1/wgh_null, help "wagtergate pulse interval 1/wgh_null";


comment_7 =? "*** Pulse Delay ***";
    initial_wait     =  1.0[s];
    relaxation_delay => 1.0[s], help "inter-pulse delay";

    include "pulse";


    phase_1   = {0,180};
    phase_2   = {90};
        phase_3   = {0};
    phase_wgh1 = {8(0), 8(90), 8(180), 8(270)};
    phase_wgh2 = {2(0), 2(90), 2(180), 2(270)};
    phase_acq = {0,180,180,0,0,180,180,0,180,0,0,180,180,0,0,180};


begin
    initial_wait;
    relaxation_delay;

        --Robust

   -- Lock prefocussing
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_1_amp);
    grad_recover;
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_2_amp);
    grad_recover;
        grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_1_amp);
    grad_recover;
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_2_amp);
    grad_recover*6;


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

        --Project
        loop loop_number times
         tau_interval;
         x_pulse*2, (obs.gate, obs.phs.phase_3, obs.atn.x_atn);
         tau_interval;
         x_pulse, (obs.gate, obs.phs.phase_2, obs.atn.x_atn);
         tau_interval;
         x_pulse*2, (obs.gate, obs.phs.phase_3, obs.atn.x_atn);
```

```
        tau_interval;
    end loop;
        --SL
        when spinlock_enable do
                x_spinlock_pulse*sl2, (obs.gate, obs.phs.phase_2, obs.atn.x_spinlock_atn);
        end when;

        -- Perfect echo
        grad_1, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_1_amp);
    grad_recover;

    wgh5_comp180(wgh_x_pulse, phase_wgh1, x_atn, wgh_tau);

    grad_1, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_1_amp);
    grad_recover;
    x_pulse, (obs.gate, obs.phs.phase_2, obs.atn.x_atn);
    grad_2, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_2_amp);
    grad_recover;

    wgh5_comp180(wgh_x_pulse, phase_wgh2, x_atn, wgh_tau);

    grad_2, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_2_amp);
    grad_recover;

    acq (dead_time, delay, phase_acq);
end pulse;
```

/* Wasted-I for Jeol spectrometers END: remove this line */

Wasted-II code for Jeol spectrometers

/* There is no warranty (implied or explicit) that it is optimal or bug-free. Anyone using this code does so at their own risk.*/

/* Wasted-II for Jeol spectrometers START: remove this line */

```
------------------------------------------------------------------------------
--                                                                          --
--                          Experiment Source Code                          --
--              Delta NMR Experiment & Machine Control Interface            --
--                                                                          --
--                                                                          --
--                                                                          --
--                                                                          --
------------------------------------------------------------------------------
-- HELP.eng: Wasted-II, combination of Robust-5 + Project for solvent suppression with T2
filter
-- Category: 1D, water_suppression, liquids
-- File name : Wasted-II
-- Sequence name : Wasted-II
--
-- Reference
-- Robust NMR water signal suppression for demanding analytical applications
-- Juan A. Aguilar and Alan. J. Kenwright
-- Analyst (2016),141, 236-242
-- doi: 10.1039/C5AN02121A
--
-- Spin echo NMR spectra without J modulation
-- A Aguilar, Juan & Nilsson, Mathias & Bodenhausen,
--
-- END HELP

header
    filename  => "Wasted-II";
    sample_id => "";
    comment   => " Wasted-II, Robust-5 + Project";
    process   =  "proton.list";
    include "header";
end header;

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

acquisition
    x_domain         => "Proton";
    x_offset         => 4.68[ppm];
    x_sweep          => 12[ppm];
    x_points         => 16384;
    scans            => 32;
    x_prescans       => 4;
    mod_return       => 1;
    include "acquisition";
end acquisition;

pulse
    include "watergate_hard";

    collect COMPLEX,OBS;

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

    grad_1       =? 1.0[ms], help "duration of grad_1";
    grad_2       =? 1.0[ms], help "duration of grad_2";
    gradient_max => z_gradient_max, help "Maximum amplitude for a given probe as defined in
the probe file";
    Note         =? "Do not exceed gradient_max value for grad_1_amp";
```

```
    grad_1_amp   => 250[mT/m], help "Amplitude first gradient pair in Tesla/meter units";
    grad_2_amp   => 60[mT/m], help "Amplitude second gradient pair in Tesla/meter units";
    grad_shape   => "SINE",("SQUARE","SINE"), help "shape of grad_1";
    grad_recover => 1[ms], help "gradient recovery time";

        spinlock_enable => TRUE, help "use spinlock pulses flanking PROJECT";
        x_spinlock_pulse => x90_spin, help "spin lock 90 pulse width";
    x_spinlock_atn   => xatn_spin, help "attenuator of x_spinlock_pulse";
        sl1                  =>128, help "SL multiplier 1";
        sl2                  =>64, help "SL multiplier 2";

comment_2 =? "*** T2 filter ***";
        delay_list       => 0.1[s],                        help "delay time(array parameter),
must be tau_step unit step";
    tau_step         => 20[ms],                      help "tau interval minimum time ";
    tau_interval     =? (tau_step - x_pulse * 5) / 4, help "tau_interval (=(tau_step-x180)2)";
    loop_number      =? round(delay_list / tau_step), help "calculation of loop number (=
round(delay_list/tau_step))";
    calc_delay       =? loop_number * tau_step,       help "real delay time = loop_number *
tau_step";
        t2_addition           =? x_spinlock_pulse*(sl1+sl2), help "additional time from
spin_lock time that contributes to T1rho relaxation";


comment_312 =? "*** Watergate hard solvent suppression ***";
    --watergate parameters
    wgh_x_pulse         => x90, help "90[deg] pulse used for watergate composite pulse";
    include "services";
    xfreq               =?  _get_freq( x_domain );
    wgh_null            => 4000[Hz], help "null frequency";
    wgh_null_ppm        =? (wgh_null/xfreq)*1000000[ppm];
    wgh_tau             =? 1/wgh_null, help "wagtergate pulse interval 1/wgh_null";


comment_7 =? "*** Pulse Delay ***";
    initial_wait     =  1.0[s];
    relaxation_delay => 1.0[s], help "inter-pulse delay";

    include "pulse";


    phase_1   = {0,180};
    phase_2   = {90};
        phase_3   = {0};
    phase_wgh1 = {8(0), 8(90), 8(180), 8(270)};
    phase_wgh2 = {2(0), 2(90), 2(180), 2(270)};
    phase_acq = {0,180,180,0,0,180,180,0,180,0,0,180,180,0,0,180};


begin
    initial_wait;
    relaxation_delay;

        --Robust

   -- Lock prefocussing
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_1_amp);
    grad_recover;
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_2_amp);
    grad_recover;
        grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_1_amp);
    grad_recover;
    grad_2, (fgz.gate, fgz.shape.grad_shape, -fgz.amp.grad_2_amp);
    grad_recover*6;

   -- Perfect echo
    x_pulse, (obs.gate, obs.phs.phase_1, obs.atn.x_atn);
    grad_1, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_1_amp);
    grad_recover;

    wgh5_comp180(wgh_x_pulse, phase_wgh1, x_atn, wgh_tau);

    grad_1, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_1_amp);
    grad_recover;
    x_pulse, (obs.gate, obs.phs.phase_2, obs.atn.x_atn);
    grad_2, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_2_amp);
    grad_recover;
```

```
    wgh5_comp180(wgh_x_pulse, phase_wgh2, x_atn, wgh_tau);

    grad_2, (fgz.gate, fgz.shape.grad_shape, fgz.amp.grad_2_amp);
    grad_recover;

        --SL
        when spinlock_enable do
                x_spinlock_pulse*sl1, (obs.gate, obs.phs.phase_2, obs.atn.x_spinlock_atn);
        end when;

        --Project
        loop loop_number times
         tau_interval;
         x_pulse*2, (obs.gate, obs.phs.phase_3, obs.atn.x_atn);
         tau_interval;
         x_pulse, (obs.gate, obs.phs.phase_2, obs.atn.x_atn);
         tau_interval;
         x_pulse*2, (obs.gate, obs.phs.phase_3, obs.atn.x_atn);
         tau_interval;
    end loop;
        --SL
        when spinlock_enable do
                x_spinlock_pulse*sl2, (obs.gate, obs.phs.phase_2, obs.atn.x_spinlock_atn);
        end when;

    acq (dead_time, delay, phase_acq);
end pulse;
```

/* Wasted-II for Jeol spectrometers END: remove this line */