Electronic Supplementary Material (ESI) for RSC Advances

"Pure shift" 1H NMR, a robust method for revealing heteronuclear couplings in complex spectra.

Juan A. Aguilar,[a] Gareth A. Morris[b] and Alan M. Kenwright[*a]

Supplementary Information

Index:

/* The pulse sequence STARTS here. Delete this line */

```
#ifndef LINT
#endif
/* Pure shift BIRD. Durham 2013.  Juan A. Aguilar, Mathias Nilsson, Gareth A. Morris.
; Please cite:
J. A. Aguilar, M. Nilsson and G. A. Morris, Angew. Chem., Int. Ed. Engl., 2011, 50, 9716 -
9717.
and
J. R. Garbow, D. P. Weitekamp and A. Pines, Chem. Phys. Lett., 1982, 93, 504 - 509.

; This sequence comes without warranty.
; Check the sequence and your parameters carefully before use.

        Paramters:
               Minimum number of scans: 2
                sspul   :       selects magnetization randomization option
                glvl_ss :     homospoil gradient level (DAC units)
                gtss :        homospoil gradient duration
                gt1     :       homospoil gradient time
                gzlvl1  :       encoding Gradient level
                gt1     :       encoding gradient time
                gstab   :       recovery delay
                j1xh    :       One-bond XH coupling constant

               pw at tpwr:             pulse duration (pw) (1H) for a 90 rotation using a tpwr
power level.
       pwxlvl : power level (13C) for a 180 rotation using pwCBIP
       pwCBIP : pulse length of the 13C 180  pulse using pwxlvl power.
      shp_CBIP: name of the shaped pulse used in conjunction with pwCBIP.

This version supports the following 13C pulses: BIP pulses (shp_CBIP='BIP125us') or Wurst2i
pulses.
40 KHz, 400 us long 13C wurst2i pulses can be produced using
ref_pw90=your_value ref_pwr=your_value pxshape('wurst2i 40000/0.4m','pulsename')

A value of sw1 around 50 Hz produces good results. Make sure sw/sw1 is an integer.

Use table = 'PSB1D.tbl' . The file should be placed here: /vnmrsys/tablib

The timing of the experiment (done in some spectrometer using setlp0) can be adjusted setting
lsfid=1 before using setlp0.

Use the macro "pureshift" to assemble the raw data into a 1D pure shift form.

*/

#include <standard.h>

pulsesequence()
{

static int     ph9[2] = {0,1};

double
       gzlvl_ss=getval("gzlvl_ss"),
       gtss=getval("gtss"),
       gzlvl1=getval("gzlvl1"),
       gt1=getval("gt1"),
       gstab=getval("gstab"),
       pwxlvl = getval("pwxlvl"),
       pwCBIP = getval("pwCBIP"),
       tau = 1/(2*(getval("j1xh"))); /* J1xh One-bond XH coupling constant */

  char   sspul[MAXSTR];
  char   shp_CBIP[MAXSTR];
  char   table[MAXSTR];

  getstr("table",table);
  getstr("sspul",sspul);
  getstr("shp_CBIP",shp_CBIP);

/* Check that sw1 is an integer submultiple of sw */

  if (fabs((sw/sw1)-(double)((int)((sw/sw1)+0.5)))> 0.01)
   {
```

```
      text_message("WARNING: sw1 should be an integer submultiple of sw\n");
   }

loadtable(table);
settable(t9,2,ph9); getelem(t9,ct,v9);

getelem(t1,ct,v1);
getelem(t2,ct,v2);
getelem(t3,ct,v3);
getelem(t4,ct,v4);
getelem(t5,ct,v5);
getelem(t6,ct,v6);
getelem(t7,ct,v7);
getelem(t8,ct,v8);
getelem(t10,ct,oph);
getelem(t11,ct,v11);

add(v2,oph,oph);
add(v2,oph,oph);
add(v6,oph,oph);
add(v6,oph,oph);
sub(oph,v3,oph);
sub(oph,v3,oph);
add(oph,v1,oph);
mod2(ct,v9);

   /* equilibrium period */
   status(A);

        if (sspul[0] == 'y')
      {
                zgradpulse(gzlvl_ss,gtss);
                obspower(tpwr);
          rgpulse(pw, zero, 0.0, 0.0);
          rgpulse(pw, one , 0.0, 0.0);
                zgradpulse(gzlvl_ss,gtss);
      }
        delay(d1);

   status(B);

        decpower(pwxlvl);
        obspower(tpwr);
        decoffset(dof);
        delay(rof1);

    rgpulse(pw,v1,rof1,rof1);               /* 1H12C suppression      1H 90y    */

    delay(tau);                                     /* 1H12C suppression               */


ifzero(v9);
simshaped_pulse("",shp_CBIP,pw*2.0,pwCBIP,v2,v2,rof1,rof1);
elsenz(v9);
if (pw*2.0 > pwCBIP) {
shaped_pulse("",pw*2.0,v2,rof1,rof1);
}
else {
delay((pwCBIP-pw*2.0)*0.5);
shaped_pulse("",pw*2.0,v2,rof1,rof1);
delay((pwCBIP-pw*2.0)*0.5);
}
endif(v9);                                      /* 1H12C suppression - END         */
delay(0.25/sw1- gt1- gstab + d2/2.0);           /* 0.25 chunktime  + time incrementation - gt1 -
gstab */
zgradpulse(gzlvl1*0.25,gt1);                    /* PFG * 1.0 */
delay(gstab);
rgpulse(pw*2.0,v6,rof1,rof1);                           /* H 180 rf */
delay(0.25/sw1- gt1- gstab );                   /* 0.25 chunk - gt1 - gstab*/

zgradpulse(gzlvl1,gt1);                         /* BIRD. PFG * 4.0 */
delay(gstab);

/*  ************************ BIRD(d)  ************************ */
/*  ************************ BIRD(d)  ************************ */
    rgpulse(pw,v3,rof1,rof1);
    delay(tau);
```

```
    simshaped_pulse("",shp_CBIP,pw*2.0,pwCBIP,v4,v4,rof1,rof1);
    delay(tau);
    rgpulse(pw,v5,rof1,rof2);

/*  ************************ BIRD(d)  ************************ */
/*  ************************ BIRD(d)  ************************ */

zgradpulse(gzlvl1*0.75,gt1);                /* BIRD. PFG * 3.0 */
delay(gstab);
delay(tau - gt1 - gstab - 1/sw);            /* 0.5/j1xh - gt1 - gstab  - 1 drop pointt*/
delay(d2/2.0);                              /* Time incrementation */

       decpower(dpwr);
       decoffset(dof);

   /* --- observe period --- */
   status(C);
}
```

/* The pulse sequence ENDS here. Delete this line */

/* The phase-cycle table STARTS here. Delete this line and save the file as vnmrsys/tablib/PSB1D.tbl*/

```
t1 = 0
t2 = [0 1 2 3]8
t3 = [0 1 2 3]2
t4 = [0 1 2 3]2
t11 = [0 1 2 3]2
t5 = [0 1 2 3]2

t6 = [0 1 2 3]32
t7 = 0
t8 = 0 2
 t10 = 0 2

/* t12 = 0 2 */
```

/* The phase-cycle table ENDS here */

/* The assembler macro STARTS here. Delete this line and save the file as /vnmrsys/maclib/pureshift */

```
if (($#>0)) then
       write('error','Usage: pureshiftZS_proc;  takes no arguments')
       abort
endif

jexp:$exp,$expname

cptmp('pureshift')
$nfid=ni
if (lsfid>0) then
       $droppts=lsfid+1
else
       $droppts=1
ENDIF
exists('droppts','parameter'):$ex
IF $ex>0 then
       $droppts=droppts
```

```
ENDIF
exists('sw1','parameter'):$ex
IF $ex<1 then          "make sure 2D parameters available"
        par2d
ENDIF
exists('nchunk','parameter'):$ex
IF $ex>0 then          "backward compatibility"
        sw1=sw*2/nchunk
        groupcopy('current','processed','acquisition')
ENDIF

$npoint=trunc((sw/sw1)+0.5)
$chunk1=$npoint/2
exists('chunk1','parameter'):$ex
IF $ex>0 then
        $chunk1=chunk1
ENDIF
if $chunk1=0 then
        $chunk1=$npoint
endif
$tmpfile=userdir+'/'+$expname+'/homodec_writefid'
$tmpfile2=userdir+'/'+$expname+'/homodec_fid'
beepoff

$imag=0.0
$real=0.0

exists($tmpfile,'file'):$ex1
IF $ex1>0 then
    shell('rm',$tmpfile)
ENDIF

$i=1
REPEAT
        writefid($tmpfile,$i)
        lookup('file',$tmpfile)
        $k=1
        repeat
                lookup('read'):$temp   "read dummy points"
                lookup('read'):$temp   "read dummy points"
        $k=$k+1
        until $k>$droppts

        $j=1
        IF $i<2 THEN
                REPEAT
                        lookup('read'):$imag[$j]
                        lookup('read'):$real[$j]
                        $j=$j+1
                UNTIL ($j>($chunk1))
        ELSE
                REPEAT
                        lookup('read'):$imag[($i-2)*$npoint+$j+$chunk1]
                        lookup('read'):$real[($i-2)*$npoint+$j+$chunk1]
                        $j=$j+1
                UNTIL ($j>($npoint))
        ENDIF
        exists($tmpfile,'file'):$ex1
        IF $ex1>0 then
            shell('rm',$tmpfile)
        ENDIF
        $i=$i+1
UNTIL ($i>$nfid)

exists($tmpfile2,'file'):$ex1
IF $ex1>0 then
    shell('rm',$tmpfile2)
ENDIF

$i=1
REPEAT
        write('file',$tmpfile2,'%d %d',$imag[$i],$real[$i])
        $i=$i+1
UNTIL ($i>(($nfid-1)*$npoint+$chunk1))

rm(curexp+'/acqfil/fid')
shell('sleep 1')
```

```
makefid($tmpfile2)

setvalue('np',2.0*($npoint*($nfid-1)+$chunk1))
setvalue('fn',np)
setvalue('at', 0.5*np/sw)
groupcopy('current','processed','acquisition')
exists($tmpfile2,'file'):$ex1
IF $ex1>0 then
  shell('rm',$tmpfile2)
ENDIF

exists('nchunk','parameter'):$ex
IF $ex>0 then          "backward compatibility"
        destroy('nchunk','current')
        destroy('nchunk','processed')
ENDIF
lb='n' gf=at/2 lsfid=0
fn=4*np ni=0
groupcopy('current','processed','acquisition')
wft aph full vsadj
```

/* The assembler macro ENDS here */

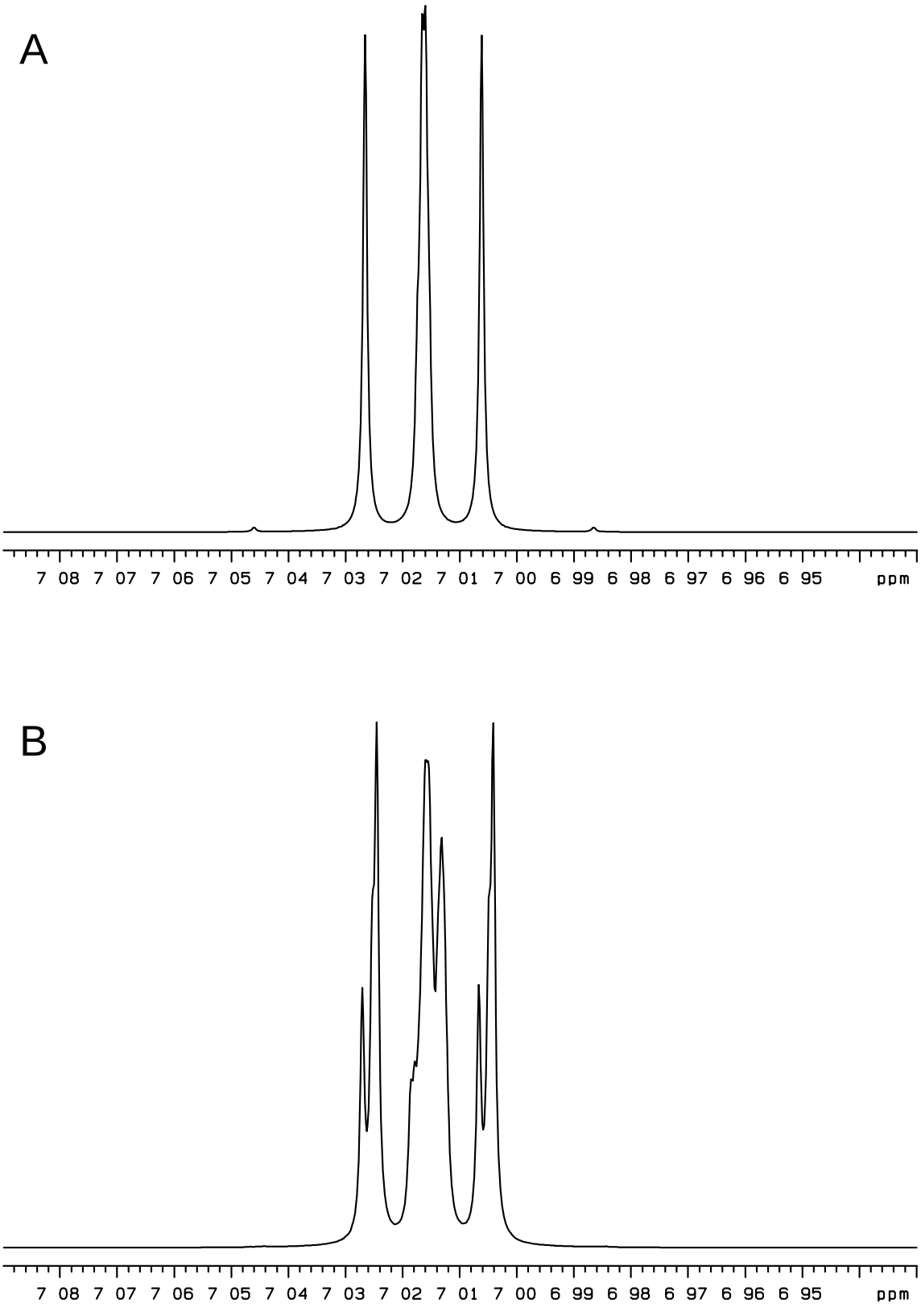

Figure S1. Spin system simulations (refs 20 & 21), using the coupling constants reported in the literature (ref 23), of the coupling patterns expected in the ^{1}H

spectrum of 1,4-difluorobenzene. Spectrum A is for the per- ^{12}C isotopologue, spectrum B is for the isotopologue with ^{13}C directly bonded to the ^{1}H nucleus being observed (all other carbons being ^{12}C). Isotope shifts for both the ^{1}H and ^{19}F nuclei as a result of the presence of the ^{13}C are included in the simulation of spectrum B (refs 24 & 25). While the agreement is not perfect, spectrum B reproduces the additional complexity (splitting) and asymmetry observed in the ^{1}H pure shift BIRD spectrum of 1,4-difluorobenzene (Figure 6). Discrepancies in detail may be due to the fact that coupling constants and shifts in the literature were determined in a different solvent to that used to record the spectra in Figure 6. Due to the relatively low signal to noise ratio and presence of decoupling artefacts in the observed spectrum (Figure 6A), no attempts were made to optimise the simulation.