Supporting Information:

SORDOR Pulses: Expansion of the Böhlen-Bodenhausen Scheme for Low-Power Broadband Magnetic Resonance

Jens D. Haller, David L. Goodwin, B. Luy

Contents

Extension of Figure 1

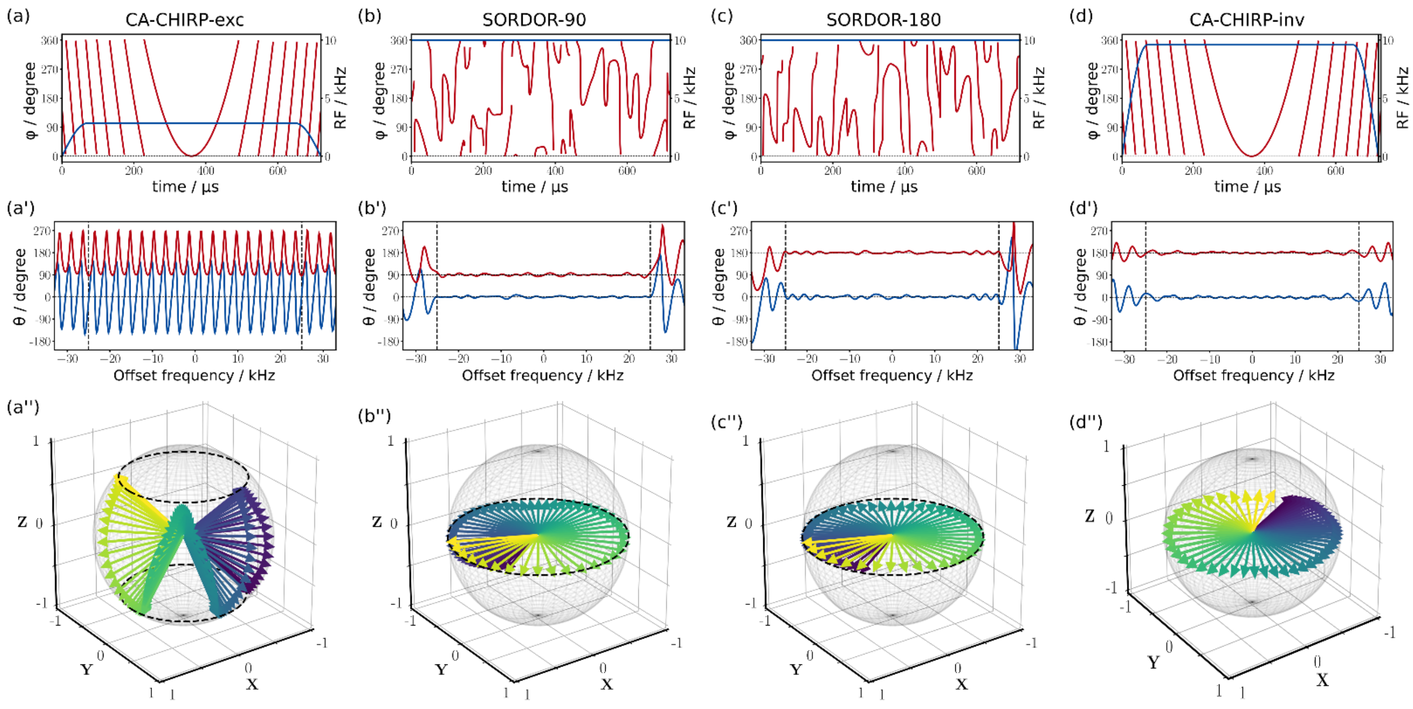

Figure S1: Comparison of a constant adiabaticity CHIRP excitation (CA-CHIRP-exc) and inversion pulse (CA-CHIRP-inv), a SORDOR-90 and the matched SORDOR-180 pulse (±10% compensation of B_1-inhomogeneity). (a-d) Pulse rf-amplitudes and phases. (a'-d') Offset dependencies of the transverse ($\sqrt(x^2 + y^2)$), and z-components of the effective rotations for the specific pulses. (a"-d") Visualizations of the rotational axes of the different pulses for a reduced offset region with color-coded offsets. Note that the CHIRP excitation and inversion pulse do not perform universal rotations while the combination of SORDOR-90 and SORDOR-180 does.

SORDOR pulse shapes

Pulse shapes are found in the .zip file with this Supporting Information

SORDOR-90 (720 µs, 50kHz bandwidth, 10kHz RF-ampl., ±5% compensation of B1-inhomog.)

File name: SORDOR_90_720us_RF10_BW50_pm5

SORDOR-180 (720 µs, 50kHz bandwidth, 10kHz RF-ampl., ±5% compensation of B1-inhomog.)

File name: SORDOR_180_720us_RF10_BW50_pm5

SORDOR-90 (720 µs, 50kHz bandwidth, 10kHz RF-ampl., ±10% compensation of B1-inhomog.)

File name: SORDOR_90_720us_RF10_BW50_pm10

SORDOR-180 (720 µs, 50kHz bandwidth, 10kHz RF-ampl., ±10% compensation of B1-inhomog.)

File name: SORDOR_180_720us_RF10_BW50_pm10

Pulse program for SORDOR 1D

```
;1D using shaped pulse

#include <Avance.incl>

1 ze
 30m pl=0[Watt]:f1
2 50m
   d1
  (p11:sp11 ph1):f1
  go=2 ph31
  50m mc #0 to 2 F0(zd)
exit

ph1=0 2 2 0 1 3 3 1
ph31=0 2 2 0 1 3 3 1


;pl1 : f1 channel - power level for pulse (default)
;p1 : f1 channel -  high power pulse
;d1 : relaxation delay; 1-5 * T1
;ns: 1 * n, total number of scans: NS * TD0
```

Pulse program for SORDOR Echo

```
;1D Echo using shaped pulse

#include <Avance.incl>

1 ze
 30m pl=0[Watt]:f1
2 50m
   d1
  (p11:sp11 ph1):f1
  d15
  (p12:sp12 ph2):f1
  d15
  go=2 ph31
  50m mc #0 to 2 F0(zd)
exit

ph1=0 2 2 0 1 3 3 1
ph2=1 3 3 1 0 2 2 0
ph31=0 2 2 0 1 3 3 1

;pl1 : f1 channel - power level for pulse (default)
;p1 : f1 channel -  high power pulse
;d1 : relaxation delay; 1-5 * T1
;d15 : echo delay
;ns: 1 * n, total number of scans: NS * TD0
```

Pulse program for ^{19}F-PROJECT using SORDOR

```
;19F-PROJECT using SORDOR pulses
;T2 measurement using PROJECT

#include <Avance.incl>

"d11=30m"

1 ze
2 d1 pl=0[Watt]:f1 rpp3
  (p11:sp11 ph1):f1
3 d20
  (p12:sp12 ph2+ph3):f1
  d20
  (p11:sp11 ph2+ph3):f1
  d20
  (p12:sp12 ph2+ph3):f1
  d20 ipp3
  lo to 3 times c
  go=2 ph31
  d11 wr #0 if #0 ivc
  lo to 1 times td1
exit

ph1=0 0 2 2 1 1 3 3
ph2=1 3 1 3 0 2 0 2

# ifdef XY16
ph3=0 1 0 1  1 0 1 0  2 3 2 3  3 2 3 2  ;XY 16
# else
ph3=0 0 2 2  2 0 0 2  2 2 0 0  0 2 2 0  ;MLEV 16
#endif /*XY16*/

ph31=0 0 2 2 1 1 3 3

;pl1 : f1 channel - power level for pulse (default)
;p1 : f1 channel -  90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;d1 : relaxation delay; 1-5 * T1
;d11: delay for disk I/O                              [30 msec]
;d20: fixed echo time to allow elimination of diffusion
;     and J-mod. effects
;vc : variable loop counter, taken from vc-list

;td1: number of experiments = number of values in vc-list
;define VCLIST
;this pulse program produces a ser-file (PARMOD = 2D)
;d20: PROJECT delay
;vc : vc should contain even numbers to provide
;     for cancellation of 180 degree pulse errors
```

Python scripts for second order phase correction

In order to execute python scripts in Topspin, please run the „edpy" command in Topspin and create new python scripts called pk2.py (1D version) and pk22D.py (pseudo-2D version). Copy and paste the subsequent code to the created scripts and execute from the Topspin command line.
For a 1D, please run „pk2 ϵ" (e.g. „pk2 3.65"), where ϵ is the value for second order phase correction and the phase φ is calculated from

$$\varphi(\nu, \epsilon) = 2\pi \cdot \left(\frac{2\nu}{sw}\right)^2 \cdot \epsilon,$$

where sw is the spectral width and ν is the frequency offset. Setting the spectral width to the band-width covered by SORDOR pulses can facilitate the procedure.
For a pseudo-2D (e.g. ^{19}F-PROJECT), the second order phase correction is applied successively and the pseudo-2D is stored as multiple 1Ds. The script is executed with „pk22D ϵ `procno`", where `procno` is the processing number of the *first* 1D spectrum and following 1Ds are stored in ascending numbers.

Second order phase correction for 1D spectra

```
# call script like: pk2 "value for pk"
# example: pk2 3.65

import sys
import math

slope = float(sys.argv[1])

spect_real = GETPROCDATA(-500, 500)
spect_imag = GETPROCDATA(-500, 500, dataconst.PROCDATA_IMAG)

for i in range(len(spect_real)):
	phase=(i-len(spect_real)/2)**2 *8*math.pi*slope/len(spect_real)**2

	dummyx=spect_real[i]*math.cos(phase) - spect_imag[i]*math.sin(phase)
	dummyy=spect_real[i]*math.sin(phase) + spect_imag[i]*math.cos(phase)

	spect_real[i]=dummyx
	spect_imag[i]=dummyy

SAVE_ARRAY_AS_1R1I(spect_real,spect_imag)
```

Second order phase correction for ^{19}F-PROJECT

```
# second order phase correction on 2D
# it will create multiple 1Ds starting at a certain "procno"
# call script like: pk22D "value for pk" "procno"
# example: pk22D 3.65 900

import sys, math

td = int(GETPAR2("s1 TD"))
curdat = CURDATA()
slope = float(sys.argv[1])

for i in range(td):
	RSR(str(i+1), str(int(sys.argv[2])+i))

	#_______________________________________________
	# pk2.py
	spect_real = GETPROCDATA(-500, 500)
	spect_imag = GETPROCDATA(-500, 500,dataconst.PROCDATA_IMAG)

	for i in range(len(spect_real)):
		phase=(i-len(spect_real)/2)**2 *8*math.pi*slope/len(spect_real)**2

		dummyx=spect_real[i]*math.cos(phase) - spect_imag[i]*math.sin(phase)
		dummyy=spect_real[i]*math.sin(phase) + spect_imag[i]*math.cos(phase)

		spect_real[i]=dummyx
		spect_imag[i]=dummyy

	SAVE_ARRAY_AS_1R1I(spect_real,spect_imag)
	#_______________________________________________

	RE(curdat)
```