MANCHESTER
1824
The University of Manchester

Practical Implementations

Mathias Nilsson

NMR Methodology Group
University of Manchester

Manchester
September 12th , 2017

Practical Implementations: overview

UoM implementation packages for Varian and Bruker

Practical considerations

Linear prediction

(Pulse sequence code examples)

Practical Implementations: overview

UoM implementation packages for Varian and Bruker

Practical considerations

Linear prediction

(Pulse sequence code examples)

Download from our website:

http://nmr.chemistry.manchester.ac.uk/pureshift

The Bruker and Varian/Agilent pure shift data and software archives can also be downloaded from DOI:10.17632/w9nz44cyft.1 and DOI:10.17632/rgj4jwcsnz.1 respectively.

UoM package (Varian and Bruker)

	Bruker Avance III/Neo (TopSpin 3/4)	Varian VNMRS (VnmrJ 4)
Example data/parameters (500 MHz, quinine in DMSO-d_6)	X	X
Pulse sequences	X	X
Setup macros		X
Processing macros	X	X
Pulse shapes	X	X
Manual	X	X

UoM package: 1D experiments

	Bruker	Varian
Interferogram		
PSYCHE	X	X
TSE-PSYCHE	X	X
Zangger-Sterk	X	X
BS	X	X
BS (multiple frequencies)		X
BIRD		X
Real time		
BIRD	X	X
BS	X	X
Zangger-Sterk	X	X

Experimental data provided includes both raw data and assembled pure shift data

UoM package: 2D experiments

	Bruker	Varian
Interferogram		
PSYCHE 2DJ	X	
F_1-PSYCHE-TOCSY	X	
PSYCHE-iDOSY	X	
Real time		
HSQC-BIRD	X	
edHSQC-BIRD	X	X

We have various other experiments available, not yet part of the package.

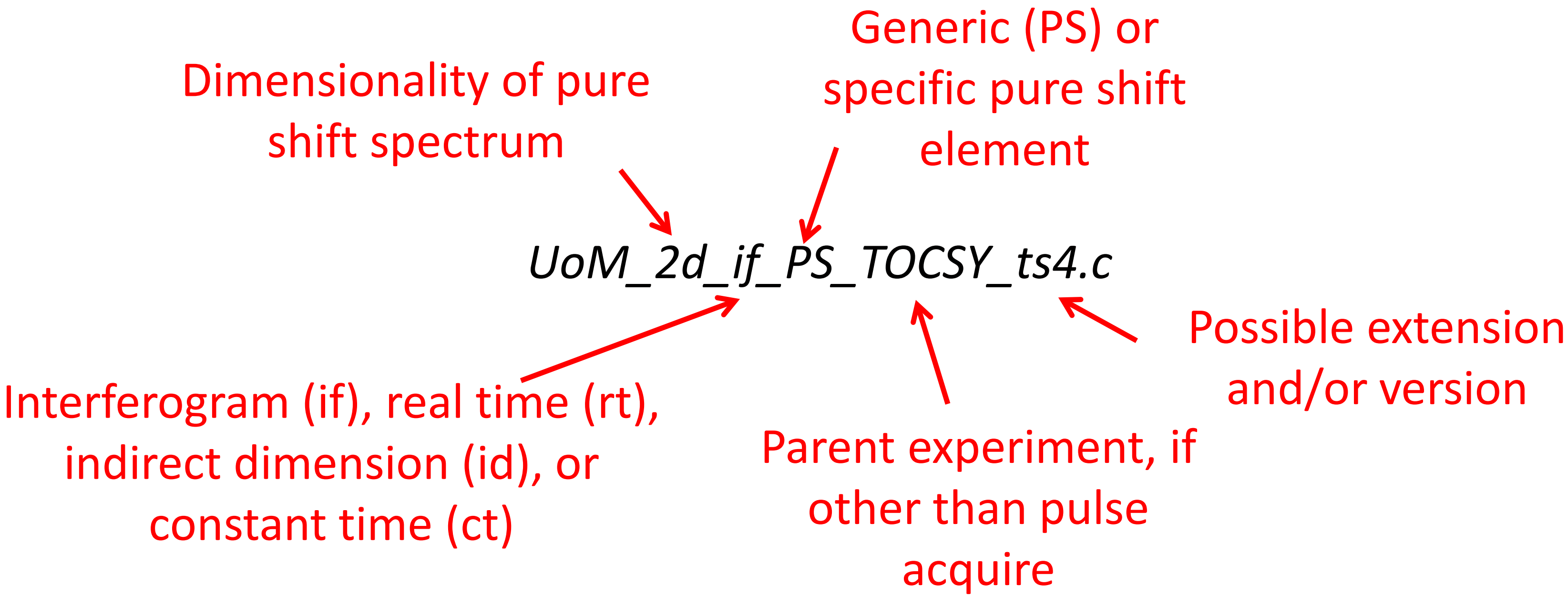
Check for updates on our website:

http://nmr.chemistry.manchester.ac.uk/

UoM package: naming convention

Pulse sequences

Convention is a general guide, not all sequences fit in logically



Dimensionality of pure shift spectrum

Generic (PS) or specific pure shift element

UoM_2d_if_PS_TOCSY_ts4.c

Interferogram (if), real time (rt), indirect dimension (id), or constant time (ct)

Parent experiment, if other than pulse acquire

Possible extension and/or version

Examples:

1D interferogram using BIRD:	*UoM_1d_if_BIRD*
2D real time HSQC:	*UoM_2d_rt_BIRD_gHSQC*
2D TOCSY with F_1 PSYCHE decoupling:	*UoM_2d_id_PSYCHE_TOCSY*

UoM package: naming convention

Macros

<u>Setup macros (currently Varian only):</u>

As pulse sequence, but starting with '*UoM_setup*'
PS is replaced with specific pure shift element

<u>Processing macros:</u>

if or *rt* for each dimensionality (not needed if *id* or *ct*):
UoM_proc_1d_if, *UoM_proc_1d_rt*,
UoM_proc_2d_if, *UoM_proc_2d_rt*,

Examples:
BIRD real time gradient HSQC
setup: *UoM_setup_2d_rt_BIRD_gHSQC*
process: *UoM_proc_2d_rt*

UoM package: setup macros

Varian only (for now):

e.g. *UoM_setup_1d_rt_ZS*, *UoM_setup_2d_BIRD_gHSQC*

Run the macro from a standard ^{1}H experiment

Sets up an experiment with (hopefully) sensible acquisition parameters as listed in the manual

Sets wexp=‘<*processing macro*>’ to produce a pure shift spectrum when “au” is used for acquisition

Saving of results is left to user

UoM package: Varian shaped pulses

Calculated on the fly if kp_auto=‘y’ [default]
based on user parameters including (*a* for *active* spin)**:**

bw_a	bandwidth
offset	
kp_wave_a	pulse shape (e.g. rsnob or psyche)
kp_beta_a	flip angle (default: 180° for ZS; 4 for PSYCHE*)

Setting the parameters (user responsibility if kp_auto=‘n’) **:**

shp_a	shapefile name in shapelib
pw180_a	duration of the pulse [μs]
pwr180_a	power of the pulse [dB]

Additions to wavelib:

/vnmr/wavelib/inversion/psyche
/vnmr/wavelib/inversion/kp2_wurst180
/vnmr/wavelib/decoupling/kp_WURST40

Detailed information in the manual
*nominal value is confusing, refer to manual

UoM package: Bruker shaped pulses

Calculating pulse shapes

- Done using standard tools
- On the fly implementation in the future?

Shapes provided

- PSYCHE elements of different durations
  - compromise between artefact suppression and T_2 weighting
  - flip angle set by CNST 20
- 180° CHIRP
  - for TSE (triple spin echo) experiments, and ZQC suppression

Detailed information in the manual

UoM package: processing macros

Varian

Keep a copy of the raw data in expX/pureshift

can be recalled using ‘*UoM_unpureshift*’

Bruker

Create a new experiment: current +1000 (warns before overwrite)

Names

Interferogram 1D experiments

UoM_proc_1d_if (Bruker: same as pshift)

Real-time 1D experiments (only Varian in current implementation)

UoM_proc_1d_rt

Interferogram 2D experiments (only Bruker in current implementation)

UoM_proc_2d_if (Bruker: same as pshift)

Real-time 2D experiments (only Varian in current implementation)

UoM_proc_2d_rt

Practical Implementations: overview

UoM implementation package for Varian and Bruker

Practical considerations

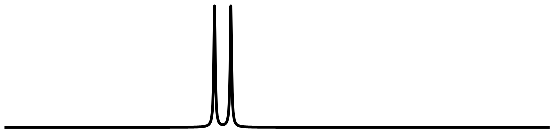
Linear prediction

(Pulse sequence code examples)

Pure shift: how do we get a clean spectrum?

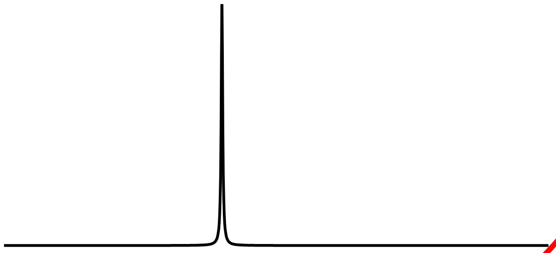
Conventional ^{1}H spectrum

This is our starting point



Ideal pure shift spectrum

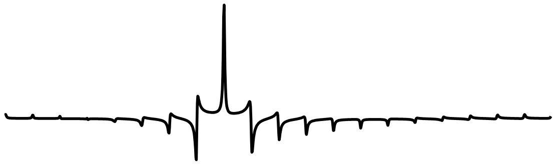
Clean, sharp, sensitive



With SAPPHIRE we are pretty close to the ideal

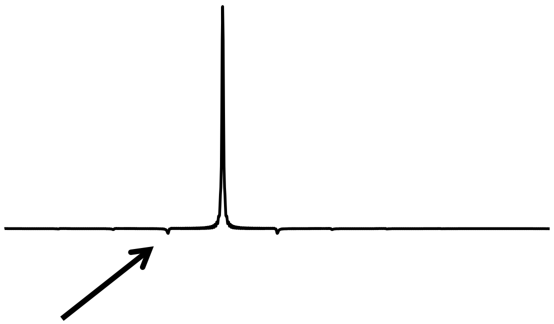
First attempt pure shift spectrum

Severe problems with artefacts/sidebands



Typical current pure shift spectrum

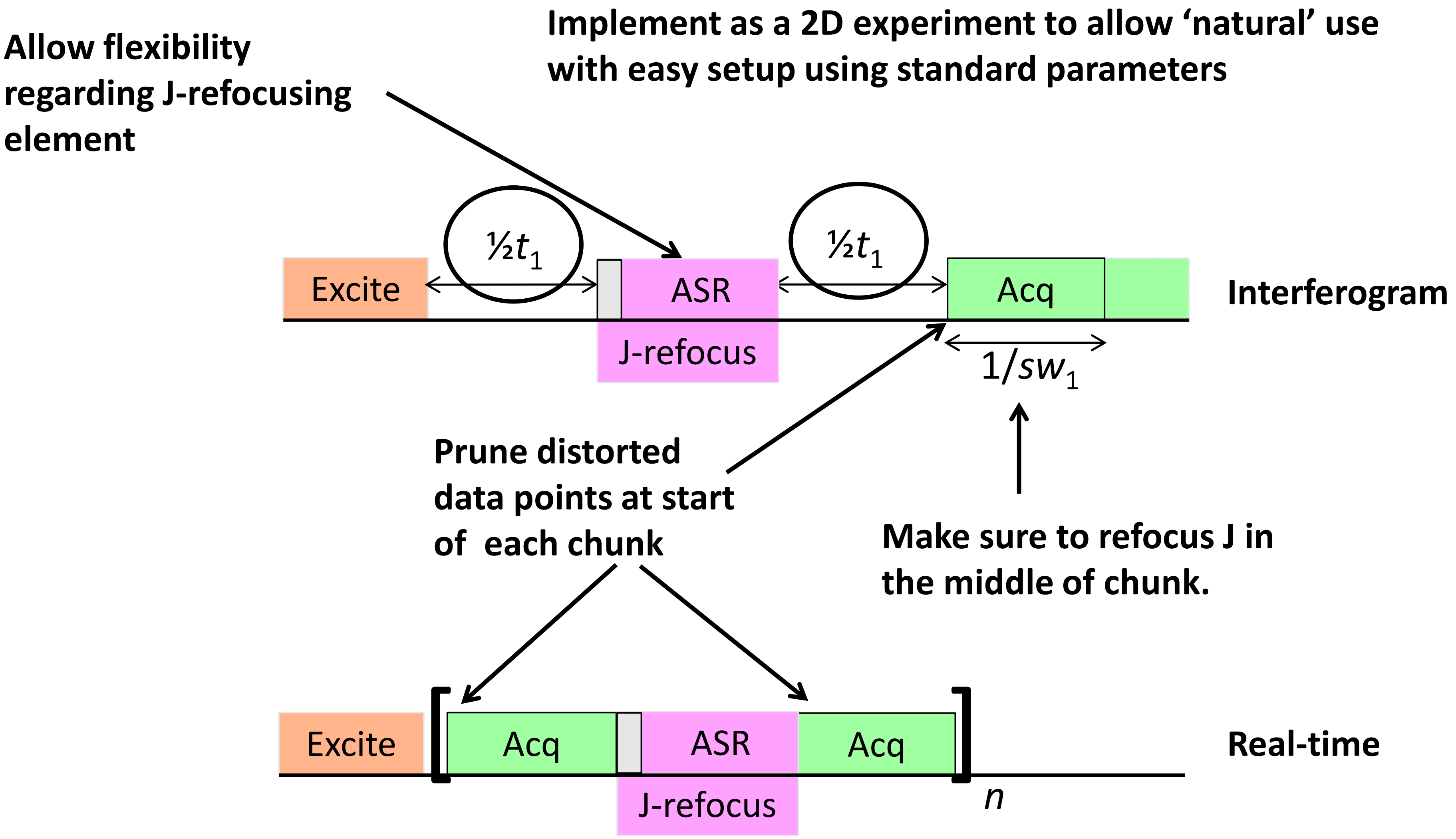
Clean enough for most applications



approx. 3%

Prototypical pure shift sequences

How do we implement practical pulse sequences?



Allow flexibility regarding J-refocusing element

Implement as a 2D experiment to allow 'natural' use with easy setup using standard parameters

Prune distorted data points at start of each chunk

Make sure to refocus J in the middle of chunk.

Interferogram

Real-time

Pure shift: practical experiments

Implement interferogram acquisition as a ’standard’ 2D experiment

The use of SW_1 to determine chunk size is easily translated to chunk duration: chunk duration = $1/SW_1$

SW/SW_1 needs to be an integer to avoid phase discontinuities between chunks

$1/SW_1$ should be short compared to $1/J$ to avoid artefacts, but is a compromise with experiment time

Allow flexibility regarding J-refocusing element

One sequence or many?

Refocus J in the middle of the chunk

Sideband shapes and phases determined by J-refocusing position

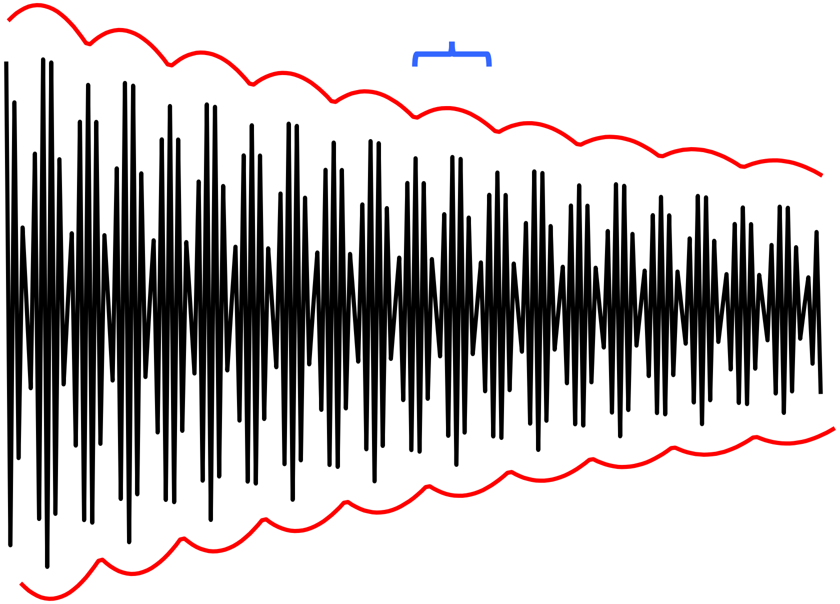
Prune distorted data points from start of each chunk

Distorted early data points cause severe chunking artefacts

Chunking sidebands

Acquiring pure shift data in chunks of duration $1/SW_1$ gives rise to J-sidebands with a spacing SW_1 in the spectrum

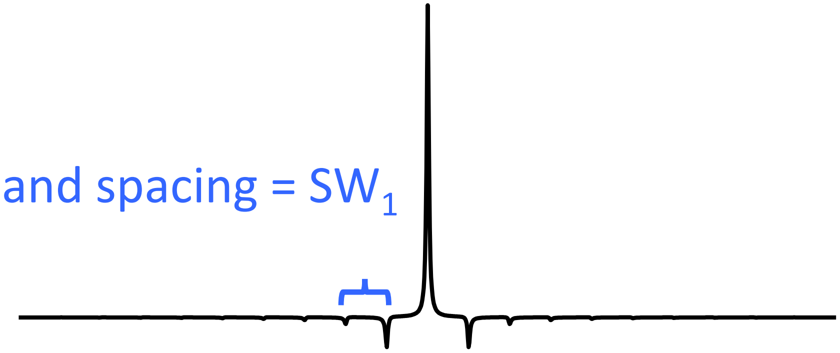
Pure shift FID



Envelope

Chunk duration = $1/SW_1$

Pure shift spectrum



Sideband spacing = SW_1

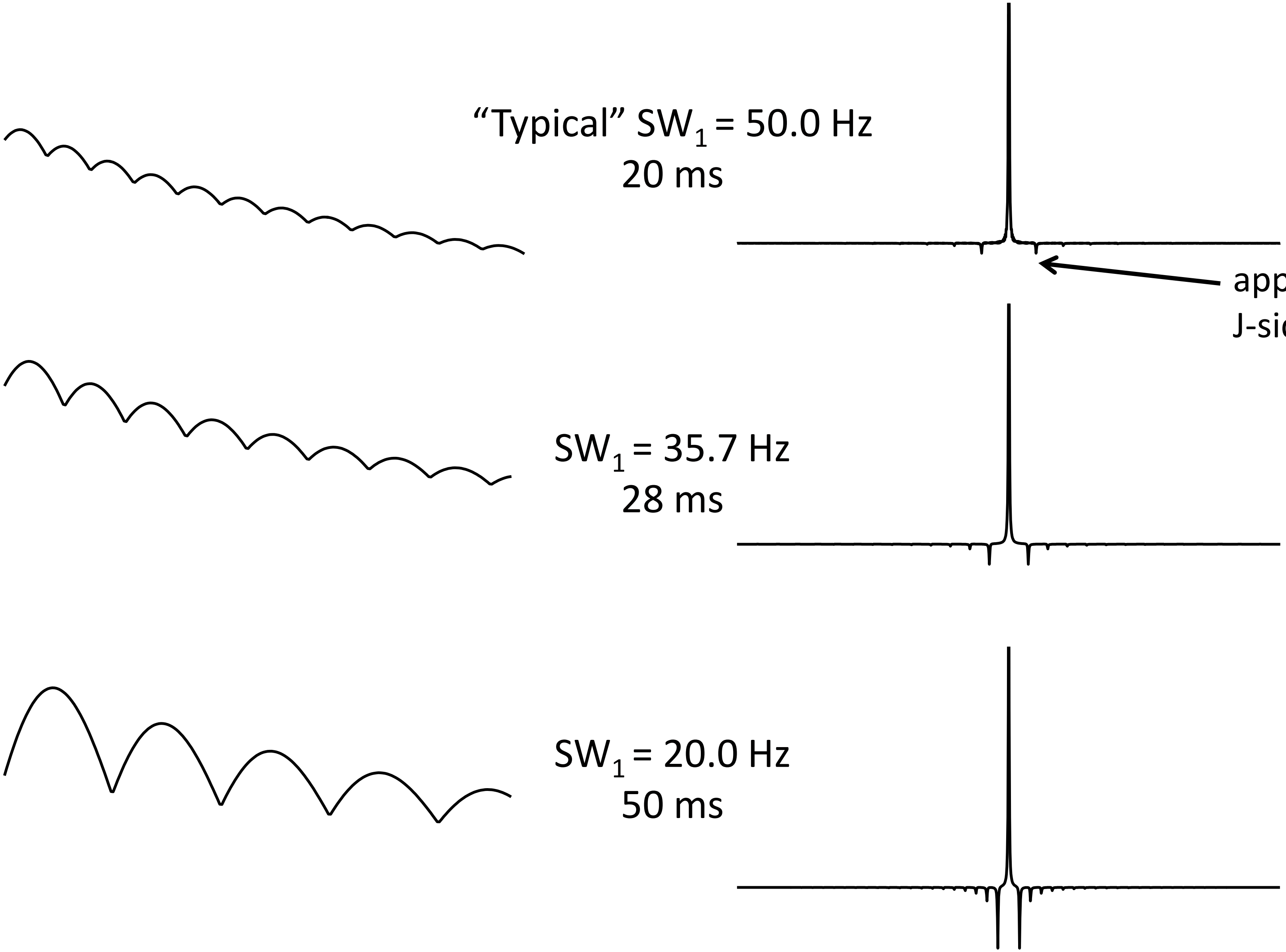
Amplitude of the n^{th} sideband for an AX system:

$$(\cos^2[\pi n] - \alpha n \cot[\pi/2\alpha] \sin[2\pi n]) / (1 - 4\alpha n^2)$$

$\alpha = SW_1/J$

J. Magn. Reson. **271**, 99 (2016)

Effect of SW$_1$ / chunk duration; J = 7.5 Hz



FID envelope

Pure shift spectrum

"Typical" SW$_1$ = 50.0 Hz
20 ms

approx. 3%
J-sidebands

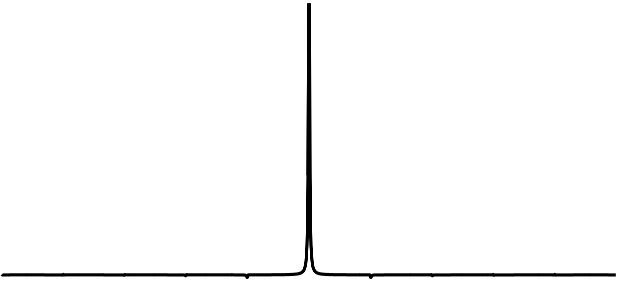
SW$_1$ = 35.7 Hz
28 ms

SW$_1$ = 20.0 Hz
50 ms

Effect of $SW/SW_1 \neq$ an integer

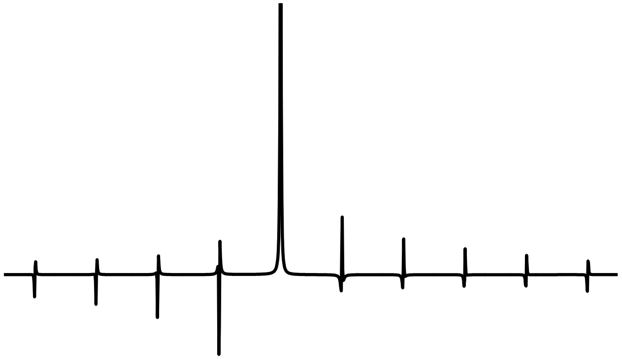
SW/SW_1 = **19.6** (SW_1 = 102 Hz, SW = 1000 Hz)

δ= 0 Hz (on resonance)
Only weak J-sidebands

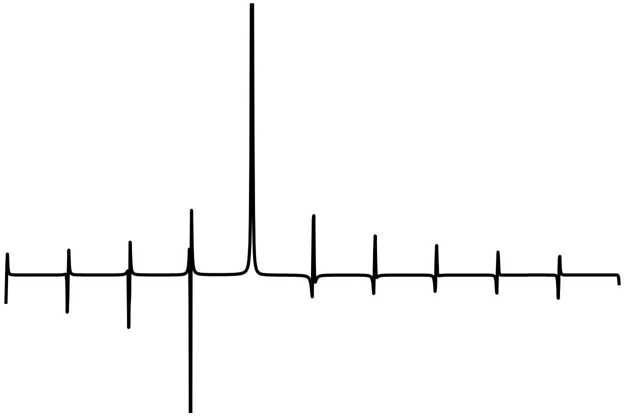


Frequency-dependent phase discontinuity between chunks

δ= 50 Hz



δ= 100 Hz

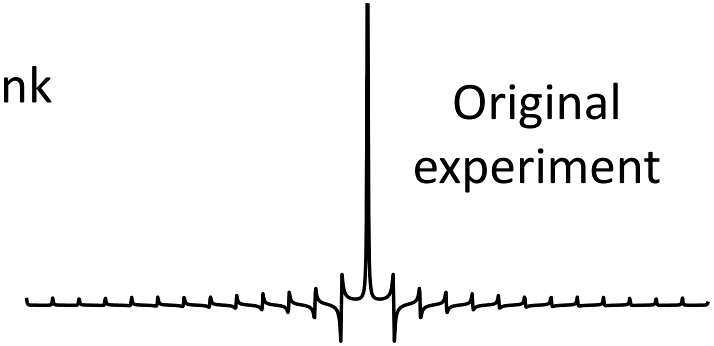


Effect of J refocusing position

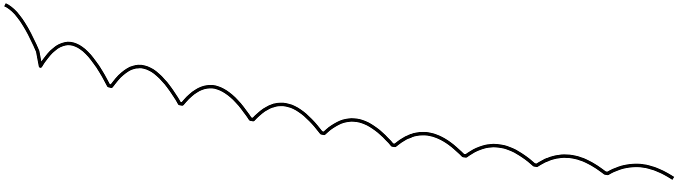
FID envelope

Pure shift spectrum

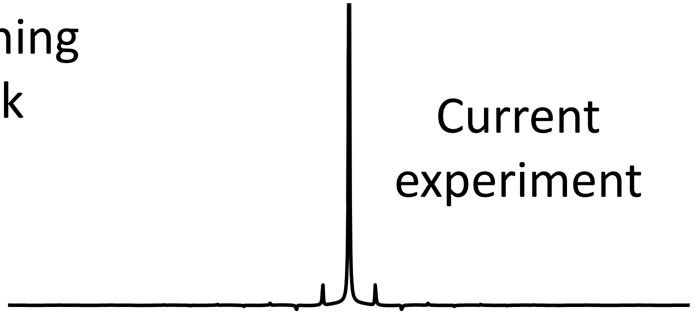
J refocused at the beginning of each chunk



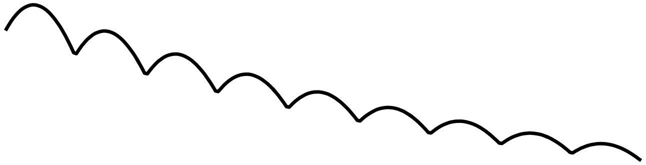
Original experiment



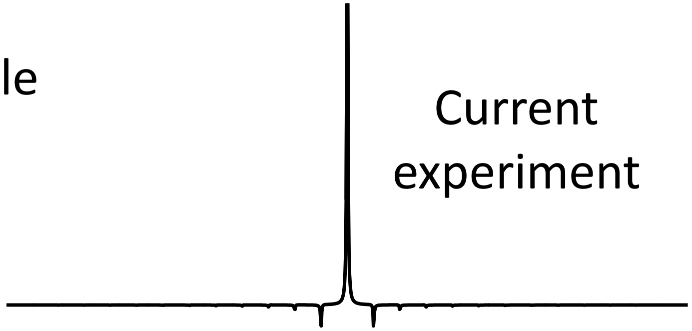
J refocused at the beginning of the first (half) chunk and in the middle of successive ones



Current experiment



J refocused in the middle of each chunk



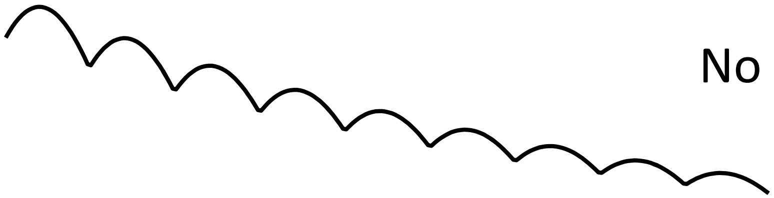
Current experiment

Effect of distorted early data points

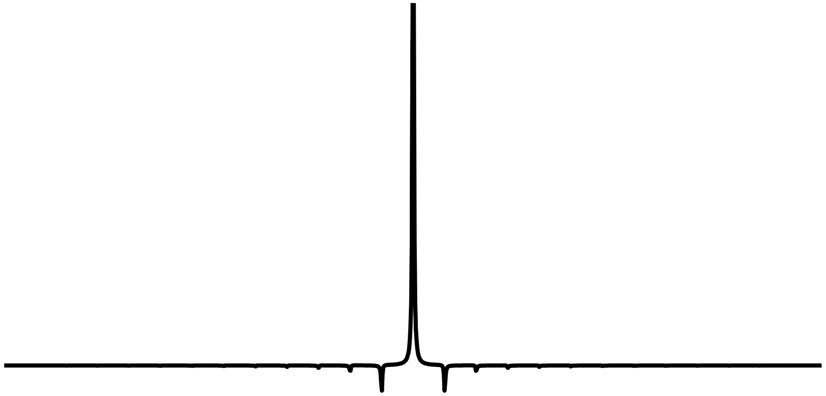
In practice the first few data points of a FID are always distorted
Modern digital filtering methods often make this worse

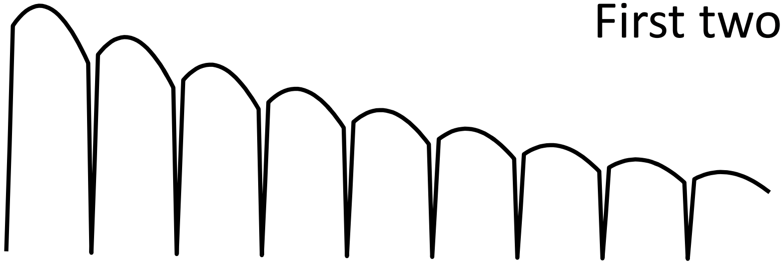
FID envelope

Pure shift spectrum

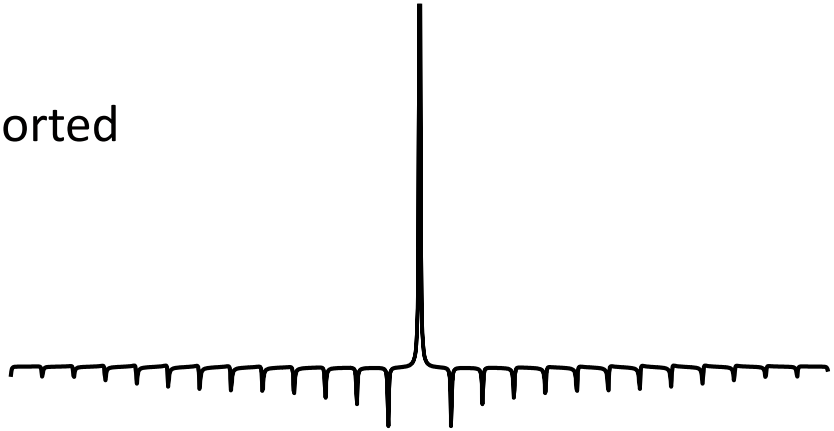


No distortion



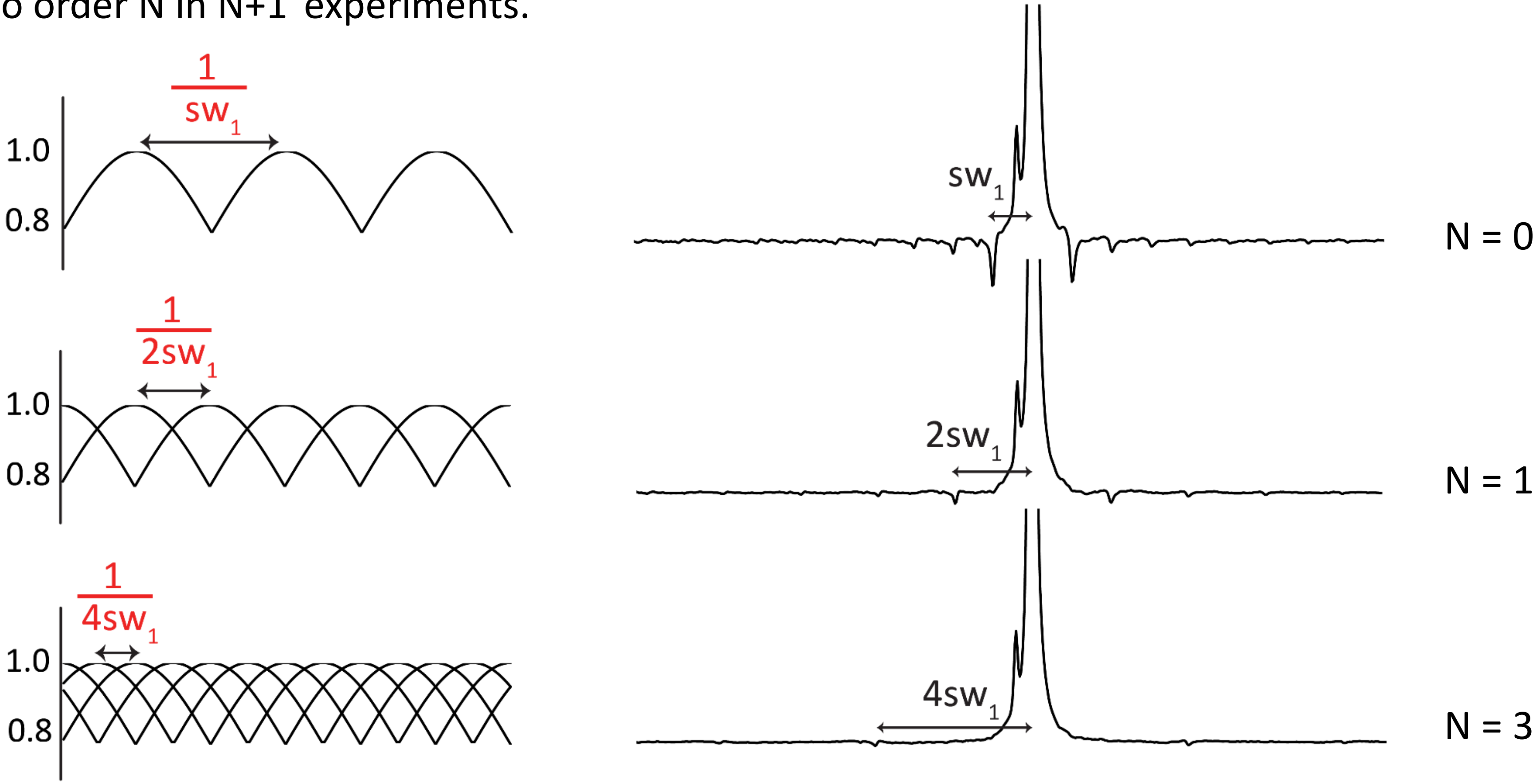


First two points distorted



SAPPHIRE: getting rid of sidebands by modulation averaging

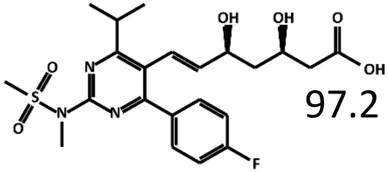
Systematically varying the timing of the first chunk suppresses sidebands to order N in N+1 experiments.



Averaging spectra measured with different SW_1 can reduce sidebands, but does not suppress them
J. Magn. Reson. **259**, 207 (2015)

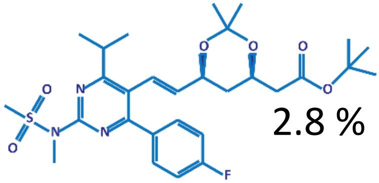
Chem. Commun. In press (2017), *DOI: 10.1039/c7cc04423b*

SAPPHIRE application: impurity analysis

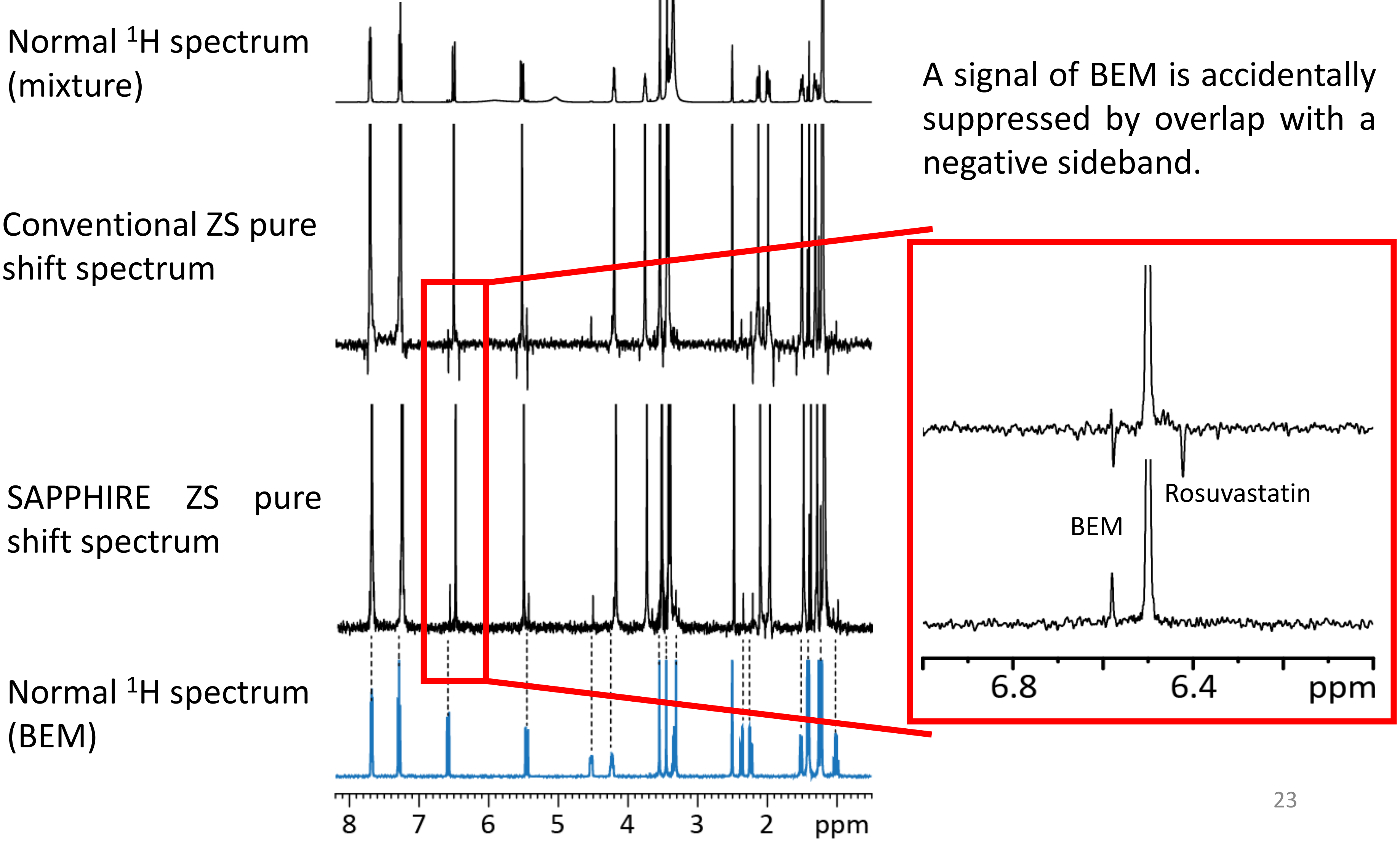


97.2 % rosuvastatin

+



2.8 % BEM (precursor)



A signal of BEM is accidentally suppressed by overlap with a negative sideband.

Practical Implementations: overview

UoM implementation package for Varian and Bruker

Practical considerations

Linear prediction

(Pulse sequence code examples)

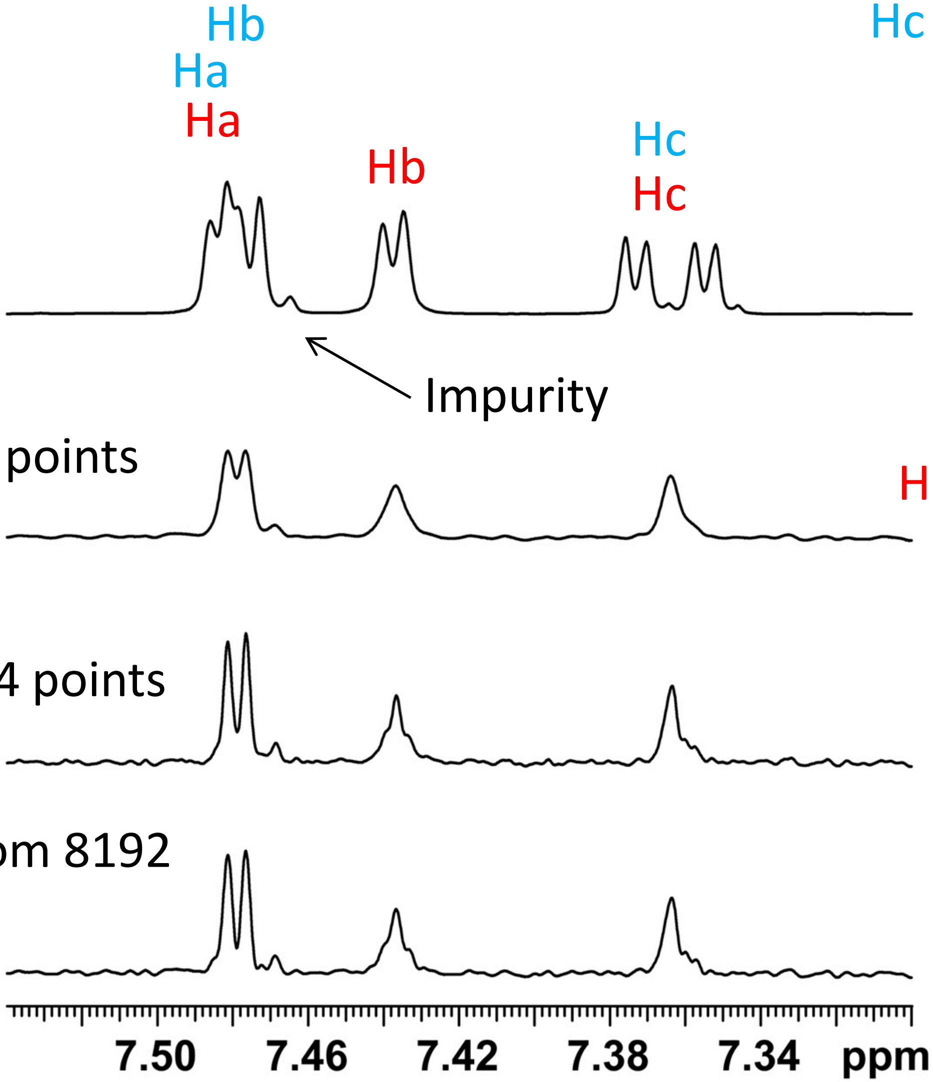
Linear prediction: 500 MHz ^{1}H spectra of a quinine and quinidine mixture

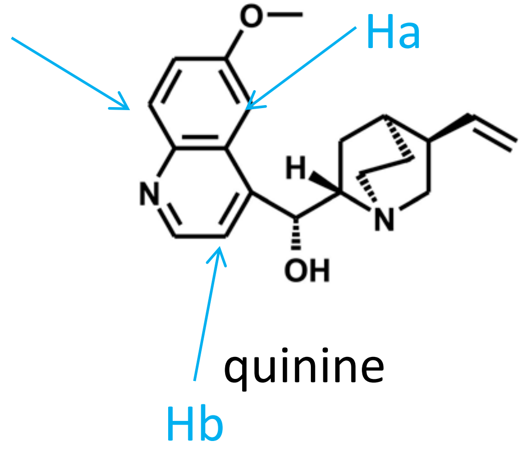
Conventional

Pure shift: 8192 points

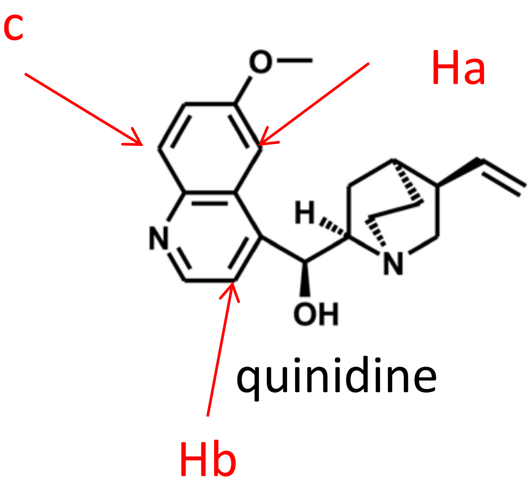
Pure shift: 16384 points

Pure shift: LP from 8192 to 16384 points





quinine



quinidine

A Pure Shift NMR Workshop

11.00	Gareth Morris	**Welcome, introduction and history**
11.30	Peter Kiraly	**Interferogram and real-time acquisition methods**
12.00	Laura Castañar	**Zangger-Sterk and band-selective methods**
12.30	Mohammadali Foroozandeh	**PSYCHE**
13.00		*Lunch and poster session*
14.00	Ralph Adams	**Other pure shift and related methods**
14.30	Mathias Nilsson	**Practical implementations**
15.00	Adolfo Botana	**JEOL pure shift implementation**
15.10	Vadim Zorin	**MestreNova pure shift implementation**
15.20	Ēriks Kupče	**Bruker shaped pulse implementation**
15.30		*Question and answer session*

University of Manchester, 12th September 2017

Practical Implementations: overview

UoM implementation packages for Varian and Bruker

Practical considerations

Pulse sequence code examples

Linear prediction

Interferogram implementation: Varian (stripped down sequence)

```
pulsesequence()
{
           droppts = getval("droppts"),                 /* number of dummy points to acquire */

status(B);

           rgpulse(pw,v1,rof1,0.0);                     / *hard 90 pulse*/
           delay(d2/2.0);
           obspower(pplvl);
           delay(0.25/sw1-rof1-gt1-gstab);              /* refocus mid-chunk*/
           zgradpulse(gzlvl1,gt1);
           delay(gstab);
           rgpulse(pp,v2,rof1,rof1);                    /* hard 180 pulse; start ZS element */
           obspower(pwr180_a);
           delay(gstab);
           zgradpulse(gzlvl1,gt1);
           delay(0.25/sw1-rof1-gt1-gstab);              /* refocus mid-chunk*/
           delay(droppts/sw);                           /* droppoints */
           delay(tau_a-rof1-gt2-gstab+rof2);
           zgradpulse(gzlvl2,gt2);
           delay(gstab);
           rgradient('z',gzlvl7);
           shaped_pulse(shp_a,pw180_a,v3,rof1,rof1);   /* soft 180 pulse */
           rgradient('z',0.0);                          /* end ZS element */
           delay(gstab);
           zgradpulse(gzlvl2,gt2);
           delay(tau_a-rof1-gt2-gstab);
           delay(d2/2.0);
status(C);


}
```

Implemented as a 2D experiment ☐
Allow different J-refocusing ☐
Refocus J in the middle of the chunk ☐
Prune distorted data points ☐

Interferogram implementation: Varian (stripped down sequence)

```
pulsesequence()
{
           droppts = getval("droppts"),                /* number of dummy points to acquire */

status(B);

           rgpulse(pw,v1,rof1,0.0);                    / *hard 90 pulse*/
           delay(d2/2.0);
           obspower(pplvl);
           delay(0.25/sw1-rof1-gt1-gstab);             /* refocus mid-chunk*/
           zgradpulse(gzlvl1,gt1);
           delay(gstab);
           rgpulse(pp,v2,rof1,rof1);                   /* hard 180 pulse; start ZS element */
           obspower(pwr180_a);
           delay(gstab);
           zgradpulse(gzlvl1,gt1);
           delay(0.25/sw1-rof1-gt1-gstab);             /* refocus mid-chunk*/
           delay(droppts/sw);                          /* droppoints */
           delay(tau_a-rof1-gt2-gstab+rof2);
           zgradpulse(gzlvl2,gt2);
           delay(gstab);
           rgradient('z',gzlvl7);
           shaped_pulse(shp_a,pw180_a,v3,rof1,rof1);   /* soft 180 pulse */
           rgradient('z',0.0);                         /* end ZS element */
           delay(gstab);
           zgradpulse(gzlvl2,gt2);
           delay(tau_a-rof1-gt2-gstab);
           delay(d2/2.0);
status(C);


}
```

Implemented as a 2D experiment ☑
Allow different J-refocusing ☐
Refocus J in the middle of the chunk ☐
Prune distorted data points ☐

Interferogram implementation: Varian (stripped down sequence)

```
pulsesequence()
{
           droppts = getval("droppts"),                /* number of dummy points to acquire */

status(B);

           rgpulse(pw,v1,rof1,0.0);                    / *hard 90 pulse*/
           delay(d2/2.0);
           obspower(pplvl);
           delay(0.25/sw1-rof1-gt1-gstab);             /* refocus mid-chunk*/
           zgradpulse(gzlvl1,gt1);
           delay(gstab);
           rgpulse(pp,v2,rof1,rof1);                   /* hard 180 pulse; start ZS element */
           obspower(pwr180_a);
           delay(gstab);
           zgradpulse(gzlvl1,gt1);
           delay(0.25/sw1-rof1-gt1-gstab);             /* refocus mid-chunk*/
           delay(droppts/sw);                          /* droppoints */
           delay(tau_a-rof1-gt2-gstab+rof2);
           zgradpulse(gzlvl2,gt2);
           delay(gstab);
           rgradient('z',gzlvl7);
           shaped_pulse(shp_a,pw180_a,v3,rof1,rof1);   /* soft 180 pulse */
           rgradient('z',0.0);                         /* end ZS element */
           delay(gstab);
           zgradpulse(gzlvl2,gt2);
           delay(tau_a-rof1-gt2-gstab);
           delay(d2/2.0);
status(C);


}
```

Implemented as a 2D experiment ☑

Allow different J-refocusing ☑

Refocus J in the middle of the chunk ☐

Prune distorted data points ☐

Interferogram implementation: Varian (stripped down sequence)

```
pulsesequence()
{
           droppts = getval("droppts"),                /* number of dummy points to acquire */

status(B);

           rgpulse(pw,v1,rof1,0.0);                    / *hard 90 pulse*/
           delay(d2/2.0);
           obspower(pplvl);
           delay(0.25/sw1-rof1-gt1-gstab);             /* refocus mid-chunk*/
           zgradpulse(gzlvl1,gt1);
           delay(gstab);
           rgpulse(pp,v2,rof1,rof1);                   /* hard 180 pulse; start ZS element */
           obspower(pwr180_a);
           delay(gstab);
           zgradpulse(gzlvl1,gt1);
           delay(0.25/sw1-rof1-gt1-gstab);             /* refocus mid-chunk*/
           delay(droppts/sw);                          /* droppoints */
           delay(tau_a-rof1-gt2-gstab+rof2);
           zgradpulse(gzlvl2,gt2);
           delay(gstab);
           rgradient('z',gzlvl7);
           shaped_pulse(shp_a,pw180_a,v3,rof1,rof1);   /* soft 180 pulse */
           rgradient('z',0.0);                         /* end ZS element */
           delay(gstab);
           zgradpulse(gzlvl2,gt2);
           delay(tau_a-rof1-gt2-gstab);
           delay(d2/2.0);
status(C);


}
```

Implemented as a 2D experiment ☑

Allow different J-refocusing ☑

Refocus J in the middle of the chunk ☑

Prune distorted data points ☐

Interferogram implementation: Varian (stripped down sequence)

```
pulsesequence()
{
           droppts = getval("droppts"),                /* number of dummy points to acquire */

status(B);

           rgpulse(pw,v1,rof1,0.0);                    / *hard 90 pulse*/
           delay(d2/2.0);
           obspower(pplvl);
           delay(0.25/sw1-rof1-gt1-gstab);             /* refocus mid-chunk*/
           zgradpulse(gzlvl1,gt1);
           delay(gstab);
           rgpulse(pp,v2,rof1,rof1);                   /* hard 180 pulse; start ZS element */
           obspower(pwr180_a);
           delay(gstab);
           zgradpulse(gzlvl1,gt1);
           delay(0.25/sw1-rof1-gt1-gstab);             /* refocus mid-chunk*/
           delay(droppts/sw);                          /* droppoints */
           delay(tau_a-rof1-gt2-gstab+rof2);
           zgradpulse(gzlvl2,gt2);
           delay(gstab);
           rgradient('z',gzlvl7);
           shaped_pulse(shp_a,pw180_a,v3,rof1,rof1);   /* soft 180 pulse */
           rgradient('z',0.0);                         /* end ZS element */
           delay(gstab);
           zgradpulse(gzlvl2,gt2);
           delay(tau_a-rof1-gt2-gstab);
           delay(d2/2.0);
status(C);


}
```

Implemented as a 2D experiment ☑

Allow different J-refocusing ☑

Refocus J in the middle of the chunk ☑

Prune distorted data points ☑

Interferogram implementation: Bruker (stripped down sequence)

```
define delay tauA
define delay tauC
"d0=0"
"in0=inf1/2"
"tauA=in0/2-p16-d16-50u”                   ;refocus mid-chunk
"tauC=dw*2*cnst4”                          ;use cnst4 for droppoints

1 ze
2 d11
3 d1 pl1:f1
  p1 ph1                                   ;hard 90
  d0
  tauA                                     ;refocus mid-chunk
  p16:gp1
  d16
  p2 ph2                                   ;hard 180 pulse; start ZS element
  p16:gp1
  d16
  50u
  tauA                                     ;refocus mid-chunk
  tauC                                     ;droppoints
  50u
  d17
  p17:gp2
  d17
  20u gron0 pl0:f1
  (p12:sp12 ph3):f1  ;soft 180
  20u groff pl1:f1   ;end ZS element
  d17
  p17:gp2
  d17
  d0
  go=2 ph31
  d11 mc #0 to 2 F1QF(id0)
```

Implemented as a 2D experiment ☐
Allow different J-refocusing ☐
Refocus J in the middle of the chunk ☐
Prune distorted data points ☐

Interferogram implementation: Bruker (stripped down sequence)

```
define delay tauA
define delay tauC
"d0=0"
"in0=inf1/2"
"tauA=in0/2-p16-d16-50u”                  ;refocus mid-chunk
"tauC=dw*2*cnst4”                         ;use cnst4 for droppoints

1 ze
2 d11
3 d1 pl1:f1
  p1 ph1                                  ;hard 90
  d0
  tauA                                    ;refocus mid-chunk
  p16:gp1
  d16
  p2 ph2                                  ;hard 180 pulse; start ZS element
  p16:gp1
  d16
  50u
  tauA                                    ;refocus mid-chunk
  tauC                                    ;droppoints
  50u
  d17
  p17:gp2
  d17
  20u gron0 pl0:f1
  (p12:sp12 ph3):f1  ;soft 180
  20u groff pl1:f1   ;end ZS element
  d17
  p17:gp2
  d17
  d0
  go=2 ph31
  d11 mc #0 to 2 F1QF(id0)
```

Implemented as a 2D experiment ☑
Allow different J-refocusing ☐
Refocus J in the middle of the chunk ☐
Prune distorted data points ☐

Interferogram implementation: Bruker (stripped down sequence)

```
define delay tauA
define delay tauC
"d0=0"
"in0=inf1/2"
"tauA=in0/2-p16-d16-50u”                    ;refocus mid-chunk
"tauC=dw*2*cnst4”                           ;use cnst4 for droppoints

1 ze
2 d11
3 d1 pl1:f1
  p1 ph1                                    ;hard 90
  d0
  tauA                                      ;refocus mid-chunk
  p16:gp1
  d16
  p2 ph2                                    ;hard 180 pulse; start ZS element
  p16:gp1
  d16
  50u
  tauA                                      ;refocus mid-chunk
  tauC                                      ;droppoints
  50u
  d17
  p17:gp2
  d17
  20u gron0 pl0:f1
  (p12:sp12 ph3):f1  ;soft 180
  20u groff pl1:f1   ;end ZS element
  d17
  p17:gp2
  d17
  d0
  go=2 ph31
  d11 mc #0 to 2 F1QF(id0)
```

Implemented as a 2D experiment ☑

Allow different J-refocusing ☑

Refocus J in the middle of the chunk ☐

Prune distorted data points ☐

Interferogram implementation: Bruker (stripped down sequence)

```
define delay tauA
define delay tauC
"d0=0"
"in0=inf1/2"
"tauA=in0/2-p16-d16-50u”                     ;refocus mid-chunk
"tauC=dw*2*cnst4”                            ;use cnst4 for droppoints

1 ze
2 d11
3 d1 pl1:f1
  p1 ph1                                     ;hard 90
  d0
  tauA                                       ;refocus mid-chunk
  p16:gp1
  d16
  p2 ph2                                     ;hard 180 pulse; start ZS element
  p16:gp1
  d16
  50u
  tauA                                       ;refocus mid-chunk
  tauC                                       ;droppoints
  50u
  d17
  p17:gp2
  d17
  20u gron0 pl0:f1
  (p12:sp12 ph3):f1  ;soft 180
  20u groff pl1:f1   ;end ZS element
  d17
  p17:gp2
  d17
  d0
  go=2 ph31
  d11 mc #0 to 2 F1QF(id0)
```

Implemented as a 2D experiment ☑

Allow different J-refocusing ☑

Refocus J in the middle of the chunk ☑

Prune distorted data points ☐

Interferogram implementation: Bruker (stripped down sequence)

```
define delay tauA
define delay tauC
"d0=0"
"in0=inf1/2"
"tauA=in0/2-p16-d16-50u”                    ;refocus mid-chunk
"tauC=dw*2*cnst4”                           ;use cnst4 for droppoints

1 ze
2 d11
3 d1 pl1:f1
  p1 ph1                                    ;hard 90
  d0
  tauA                                      ;refocus mid-chunk
  p16:gp1
  d16
  p2 ph2                                    ;hard 180 pulse; start ZS element
  p16:gp1
  d16
  50u
  tauA                                      ;refocus mid-chunk
  tauC                                      ;droppoints
  50u
  d17
  p17:gp2
  d17
  20u gron0 pl0:f1
  (p12:sp12 ph3):f1  ;soft 180
  20u groff pl1:f1   ;end ZS element
  d17
  p17:gp2
  d17
  d0
  go=2 ph31
  d11 mc #0 to 2 F1QF(id0)
```

Implemented as a 2D experiment ☑

Allow different J-refocusing ☑

Refocus J in the middle of the chunk ☑

Prune distorted data points ☑

Varian pulse sequence for real-time acquisition (stripped down)

```
pulsesequence()
{
     droppts = getval("droppts"),                           /* number of dummy points to acquire */
                                                            /* will be stripped by post */
                                                            /* acquisition macro */

     kp_npoints = getval("kp_npoints ")                     /* number of points per chunk*/
     kp_cycles = getval("kp_cycles ")                       /* number of chunks*/

     setacqmode(WACQ|NZ);                                   /* stop DSP to collect zeros*/


     loop(v10,v11);                                         /* loop over nchunks*/


            rcvron();
            delay(rof1);
            acquire(droppts,1.0/sw);                        /* drop points */
            if (dm[2]=='y')                                 /* start decoupler for chunk */
            {
                        decprgon(dseq, 1.0/dmf, dres);
                        decon(); decunblank();
            }
            acquire(kp_npoints,1.0/sw);                     /* Acquire chunk*/
            if (dm[2]=='y')                                 /* stop decoupler during droppoints */
            {
                        decoff();
                        decprgoff();
            }
            acquire(droppts,1.0/sw);                        /* drop points */

            recoff();   }

     /* Pure shift ELEMENT HERE */

     endloop(v11);
```

Bruker pulse sequence for real time acquisition (stripped-down)

Avance III/Topspin 3

```
  dwellmode explicit                 ;before starting actual sequence

  ACQ_START(ph30,ph31)
4 0.1u REC_UNBLK                     ; Start receiver for acquisition
  0.05u DWL_CLK_ON                   ; Start DSP for acquisition
  d62                                ; Acquire data points (d62 = chunk duration)
  0.05u DWL_CLK_ON                   ; Stop DSP so we don’t acquire zeros during J refocus
  0.1u REC_BLK                       ; Stop Reciever
; J REFOCUS ELEMENT HERE
  lo to 4 times l1

d11 do:f2 mc #0 to 2
     F1EA(calgrad(EA ACQ_START(ph30,ph31)), caldel(d0, +in0) & calph(ph3, +180) & calph(ph6, +180) &
calph(ph31, +180))
```

Neo/TopSpin 4

```
 ACQ_START(ph30,ph31)
; J REFOCUS ELEMENT HERE
4 0.1u REC_UNBLK                     ; Start receiver for acquisition
  0.05u DWELL_RELEASE                ; Start DSP for acquisition
  d62                                ; Acquire data points d62 = chunk duration)
  0.05u DWELL_HOLD                   ; Stop DSP so we don’t acquire zeros during J refocus
  0.1u REC_BLK                       ; Stop Reciever
; J REFOCUS ELEMENT HERE
  lo to 4 times l1

d11 do:f2 mc #0 to 2
     F1EA(calgrad(EA ACQ_START(ph30,ph31)), caldel(d0, +in0) & calph(ph3, +180) & calph(ph6, +180) &
calph(ph31, +180))
```