Contents

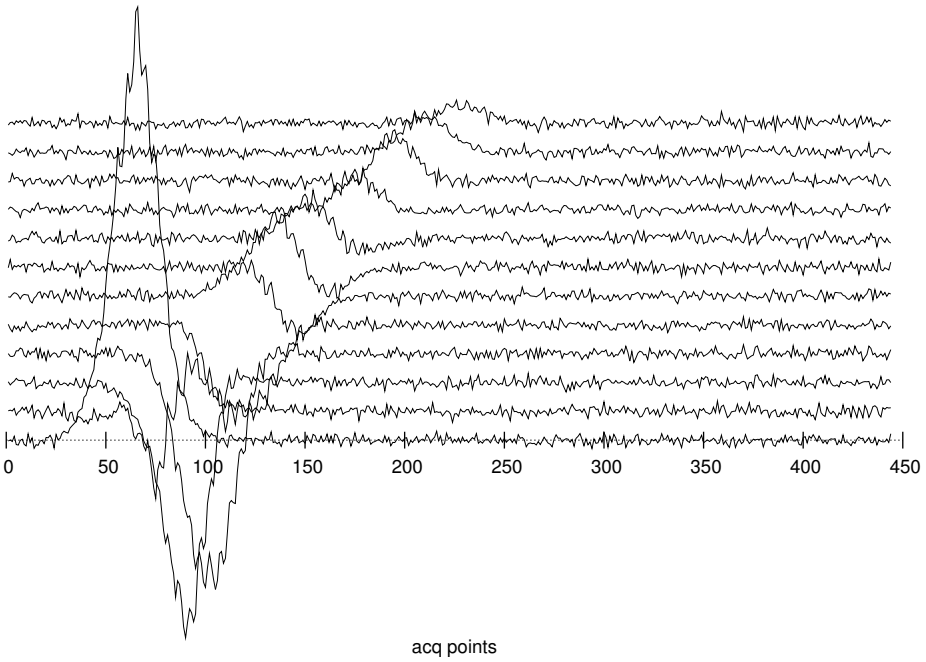

Figure 1: FID slices from a shifted-echo experiment

1 The Shifted Echo MQMAS Experiment

Many variations of the MQMAS experiment exist. Here we concentrate on the simple echo, the shifted echo, and the split-t_1 experiment (Fig. **??**). Each one is appropriate for different circumstances.

The **shifted echo** experiment is just that – it shifts the echo further forward into the acquisition time so that the initially aquired density is zero (noise), followed by the full echo. An example of the acquired FIDs is given in Fig. 1. During processing, the acquisition time = 0 point is shifted to the centre of the echo. The signal can consequently be considered to span from effectually negative infinite time to positive infinite time. Because absorptive and dispersive signal are even (symmetric) and odd (anti-symmetric) about zero, respectively (think cosine and sine), acquisition of the full echo provides the necessary sign discrimination between positive and negative frequencies. There are therefore no t_2-dependent dispersion-mode components in the 2D signal

The simple echo experiment is used for samples which are dominated by homogenous broadening, i.e. rapidly decaying lifetime. Aside of the rapid l

The shifted-echo pulse sequence is comprised of $(7\pi/12) - t_1 - (\pi/6) - \tau - (\pi/3) - acq$

1.1 Phase table

ph1:	0	30	60	90	120	150	180	210	240	270	300	330
ph2:	0											
ph3:	0	0	0	0	0	0	0	0	0	0	0	0
	45	45	45	45	45	45	45	45	45	45	45	45
	90	90	90	90	90	90	90	90	90	90	90	90
	135	135	135	135	135	135	135	135	135	135	135	135
	180	180	180	180	180	180	180	180	180	180	180	180
	225	225	225	225	225	225	225	225	225	225	225	225
	270	270	270	270	270	270	270	270	270	270	270	270
	315	315	315	315	315	315	315	315	315	315	315	315
phr:	0	270	180	90	0	270	180	90	0	270	180	90
	90	0	270	180	90	0	270	180	90	0	270	180
	180	90	0	270	180	90	0	270	180	90	0	270
	270	180	90	0	270	180	90	0	270	180	90	0

The fixed delay τ of 1 ms causes a shift in the echo of 50 points at a dwell time (Δt) of 20 μs per point, thus a first-order phase adjustment of $-18000° = -50 \times 360°$. The variable t_1 delay causes an additional delay which must be handled differently for each slice, as can be seen clearly in figure **??**. This t_1-dependent shift is accomplished via a "shear" transform of the 2D spectrum, based on the time shifting theorem. The appropriate basic value to shift by for a spin $I = 5/2$ nucleus is 19/12. The implementation of the shear transform in NMRPipe, however, requires that this value be multiplied by the ratio of the digital resolutions in the direct and indirect dimensions. It may be appropriate to shear instead by a factor of 2.965, which is (19/12 + 5/6 + 17/31); 5/6 compensates for the residual slope noted by Bodart (J Mag Res 133 p207 (1998)) and 17/31 removes the chemical shift offset contribution to the shift in the isotropic dimension as described by Massiot (Solid State NMR 6 p73 (1996) eq. 11).

Data process with the script below is shown in figures **??** for sodium salt and **??** for potassium salt samples.

Where the offset gives the difference between the NMRPipe default right edge of the spectrum and the desired center of the spectrum (along the directly detected, anisotropic dimension):

1.2 Processing of MQMAS data with NMRPipe

1.2.1 Data Conversion

Simpson simulations: in principle, the -nmrpipe output flag in Simpson generates an NMRPipe format .fid file. Generally several values in the header are incorrect, however, and these must be fixed up in accord with the parameters you used to simulate. Something like the following can be included at the top of your spectral processing file (e.g. nmrproc.com):

```
1  #!/bin/tcsh
2
3  set n="27Al_shiftedechopipe"
4
5  sethdr $n.fid\
6    -xN               512  -yN          24  \
7    -xT               256  -yT          24  \
```

```
  -xMODE        Complex  -yMODE     Real  \
  -xSW        50000.000  -ySW      10000  \
  -xOBS          156.00  -yOBS   156.000  \
  -xCAR           0.000  -yCAR     0.000  \
  -xLAB             MAS  -yLAB       Iso  \
  -ndim               2  -aq2D Magnitude
```

Varian data:

```
#!/bin/tcsh

var2pipe -in ./fid -noaswap  \
  -xN              3000  -yN              36  \
  -xT              1500  -yT              36  \
  -xMODE        Complex  -yMODE         Real  \
  -xSW       100000.000  -ySW      10000.000  \
  -xOBS         156.267  -yOBS       156.267  \
  -xCAR           0.000  -yCAR         0.000  \
  -xLAB             MAS  -yLAB           ISO  \
  -ndim               2  -aq2D     Magnitude  \
  -out ./test.fid -verb -ov

```

Bruker data:

```
#!/bin/tcsh

bruk2pipe -in ./ser -bad 0.0 -noaswap -DMX -decim 4 -dspfvs 10 -grpdly 0  \
  -xN              1024  -yN              24  \
  -xT               512  -yT              24  \
  -xMODE        Complex  -yMODE        Real  \
  -xSW         50000.00  -ySW         10000  \
  -xOBS         112.432  -yOBS      112.432  \
  -xCAR             0.0  -yCAR          0.0  \
  -xLAB       Aniso-ppm  -yLAB      Iso-MAS  \
  -ndim               2  -aq2D    Magnitude  \
  -out ./test.fid -verb -ov

```

1.2.2 Spectral Processing

Experimental Data

```
#!/bin/tcsh

nmrPipe -in test.fid\
| nmrPipe -fn MAC -macro /usr/local/NMRPipe/nmrtxt/stagger_GM.M \
      -var slope 3.1667 g2 200 initoff 50\
#
| nmrPipe  -fn ZF -size 1024\
```

```
| nmrPipe  -fn FT -auto\
| nmrPipe -fn PS -p0 -65 -p1 -18000 \
| nmrPipe -fn TP \
| nmrPipe -fn RS -rs 1 \
| nmrPipe -fn GM -g1 0 -g2 50 -c 1.0\
| nmrPipe -fn ZF -size 512\
| nmrPipe -fn FT -auto\
| nmrPipe -fn PS -p0 -95 -p1 0\
| nmrPipe  -fn MAC -macro /usr/local/NMRPipe/nmrtxt/shear.M \
      -var slope 3.9583 offset 512\
| nmrPipe -fn CS -ls 128 -sw\
| nmrPipe -fn BASE -nw 2 -nl 0% 10% 90% 100%\
| nmrPipe -fn TP\
| nmrPipe  -fn MAC -macro /opt/NMRPipe/nmrtxt/shear.M \
      -var slope 0.282353 offset 128\
| nmrPipe -fn BASE -nw 2 -nl 0% 4% 96% 100%\
   -verb -ov -out test_1912_1217.ft2

pipe2txt.tcl -index Hz test_1912_1217.ft2 > test_1912_1217.dat
```

1. In case the default shell is other than csh, run this script in tcsh (a less buggy version of csh). Type `echo $0` on the command line to determine your shell.

3. Read in the NMRPipe format.

4-5. Normal apodization functions (e.g. EM, GM) apply the same windowing function to all slices. The shifting echo requires the apodizing function to slide along t_2, so it is maximal at the top of the echo. The echo position is predicatable, and so we use a custom macro to accomplish this:

```
/* stagger_GM -- apply GM with a t1-dependent offset                          */
/* written by J. Gehman 08/2011                                               */
/* | nmrPipe -fn MAC -macro stagger_GM.M -var g2 200 slope 1.5833 initoff 50 \ */
/* install this file into /usr/local/NMRPipe/nmrtxt/ or /opt/NMRPipe/...      */
/*    or where ever NMRPipe is installed                                      */

NDSW = 1003;
ABS_XDIM = -1;
ABS_YDIM = -2;

sQsw   = getParm( fdata, NDSW, ABS_XDIM );
mQsw   = getParm( fdata, NDSW, ABS_YDIM );

/* 1 point in mQ is the same time as mQsw/sQsw points in 1Q */
shift  = initoff + slope*(yLoc-1)*sQsw/mQsw;

g  = 0.6*PI*g2/sQsw;
gg = g*g;

for( i = 0; i < size; i++ ) {
   t = shift - i;
   w = exp( -gg*t*t );
   rdata[i] *= w;
   idata[i] *= w;
}
```

The variables that must be passed to the macro are:

- `initoff`: set this to the fixed delay × sw
- `g2`: the decay constant, same as what one would otherwise use with the `GM` windowing function
- `slope`: set to -k for the given spin and coherence order selected for evolution in t_1. Also, the data is in the time domain, with no processing yet performed – for the phase-modulated shifted-echo form of the MQMAS experiment, this means that an additional factor of 2 is necessary to compensate for the fact that the direct 1Q dimension is complex (two data points per digital time point) but the indirect mQ dimension is real (one data point per digital time point). For 3QMAS of a spin $^5/_2$, this would be $2 \times {}^{19}/_{12} = 3.1667$.

7. Zero fill as desired to increase *digital* resolution (i.e. *not* real resolution).

8. Fourier transform the t_2 (directly detected, MAS) dimension.

9. Perform a first order phase correction to account for the echo shift. The `p1` value should be negative of the τ echo delay (tXechsel on Varian) × the direct dimension spectral width (same as `-xSW` in the conversion script) × 360. Perform any neceeary zero-order phase corrections as well.

10. Transpose the data plane so that subsequent commands operate on the t_2 (indirectly detected, isotropic) dimension.

11. Right-shift the indirect data as the first slice already includes a full rotor period of 3Q evolution

12. Zero fill as desired for digital resolution as for the direct dimension in step 7.

13. Gaussian apodization in the indirect dimension.

14. Fourier transform.

15. Phase correct the indirect dimension if necessary

16-17. Run the shearing transform using a macro. The macro is:

```
shift  = slope*(yLoc - offset);

p0 = 0.0;
p1 = -360.0*shift;

(void) hilbert( rdata, idata, size);
(void) ift( rdata, idata, size);
(void) phase( rdata, idata, size, p0, p1 );
(void) fft( rdata, idata, size);
```

Set offset to half of the number of points in the direct dimension (in this case set by `-size` in the ZF, step 7 above. Set the slope to the desired shearing factor a × the ratio of the digital resolutions in each dimension (computed with spectral widths sw and number of points np). For a spin $^5/_2$ nucleus one might choose $a = k = {}^{19}/_{12}$:

$$\text{slope} = a \times \frac{\text{sw}_{\text{MAS}}}{\text{sw}_{\text{iso}}} \times \frac{\text{np}_{\text{iso}}}{\text{np}_{\text{MAS}}} \tag{1}$$

18. The experiment in the first place was probably run (or should have been run) with just enough spectral width in the indirect (isotropic) dimension to fit all species and inhomogenous lineshapes, but portions of the lineshape might be aliased through the opposite edge of the spectrum due to the spectral offset from the carrier. Perform a circular shift to frame the spectra window properly. As a first guess, try `-ls` about a quarter of the spectral width.

19. Baseline correct using the left and right edges of the spectrum

20. Transpose data plane back to operate on the direct dimension.

21-22. Shear again in this dimension, if desired. One may, for example, want to use a shearing factor $b = {}^{-1}/_{(\lambda+p)}$, as discussed [**?**]. Note that if a CS is performed in the other dimension (e.g. step 18 above), The number of points left-shifted must be subtracted from the offset. Here 256 is theoretical centre of the 512-point indirect dimension, but zero was shifted by 128 points, so we use `offset 128` instead of 256.

23. Baseline correct using the left and right edges of each slice. Be careful that no real spectral intensity is included in the percentages used in to define the edges.

24. Output sheared spectrum in NMRPipe format, overwriting (`-ov`) any previous spectra with the same name.

26. Write an ascii list of frequency coordinates and intensities.

Simulated Data Processing of Simpson simulated data is similar, with exceptions noted:

```
#!/bin/tcsh

set n="27Al_shiftedechopipe"

nmrPipe -in $n.fid\
| nmrPipe -fn GM -g1 0 -g2 100 -c 1.0\
| nmrPipe  -fn ZF -size 1024\
| nmrPipe  -fn FT -auto\
| nmrPipe -fn PS -p0 0 -p1 -18000 \
| nmrPipe -fn TP \
#
| nmrPipe -fn ZF -size 512\
| nmrPipe -fn GM -g1 0 -g2 200 -c 1.0\
| nmrPipe -fn FT -auto\
#
| nmrPipe  -fn MAC -macro /opt/NMRPipe/nmrtxt/shear.M\
      -var slope 3.9583 offset 512\
| nmrPipe -fn CS -ls 2500Hz -sw\
| nmrPipe -fn TP\
| nmrPipe  -fn MAC -macro /opt/NMRPipe/nmrtxt/shear.M\
      -var slope 0.282353 offset 256\
| nmrPipe -fn BASE -nw 2 -nl 0% 10% 90% 100%\
   -verb -ov -out ${n}_1912_1217.ft2

pipe2txt.tcl -index Hz ${n}_1912_1217.ft2 > ${n}_1912_1217.dat
```

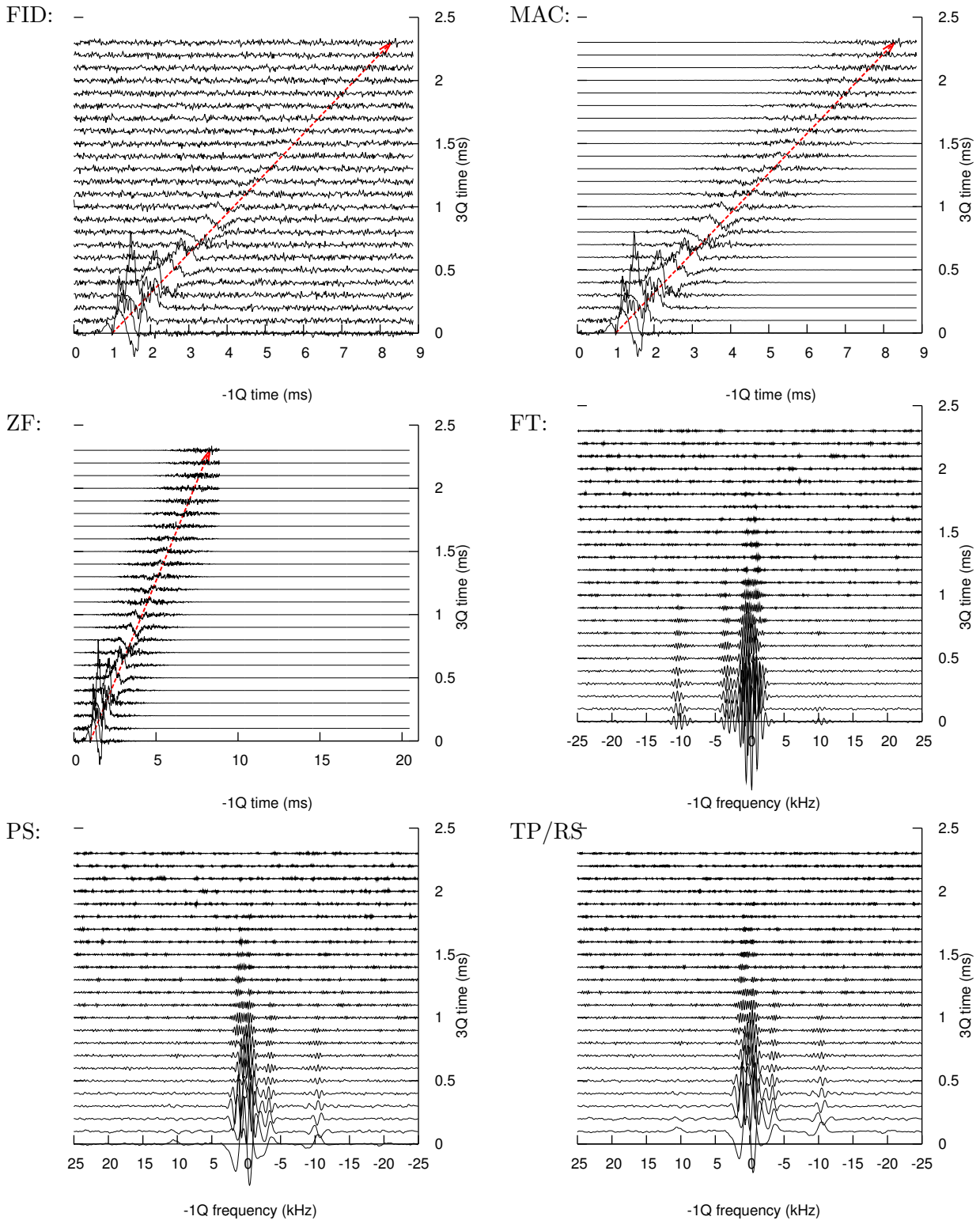

Figure 2: Graphic representation of processing steps for 3QMAS (t_1,t_2) and (t_1,ν_2) spectral domains. Sample/data are "Geopolymer K1.5 Si/Al=2.0" #308 from Chem-Eng/Tallahassee Jan 2007 data series. Red lines show the slope of the shifted echo.

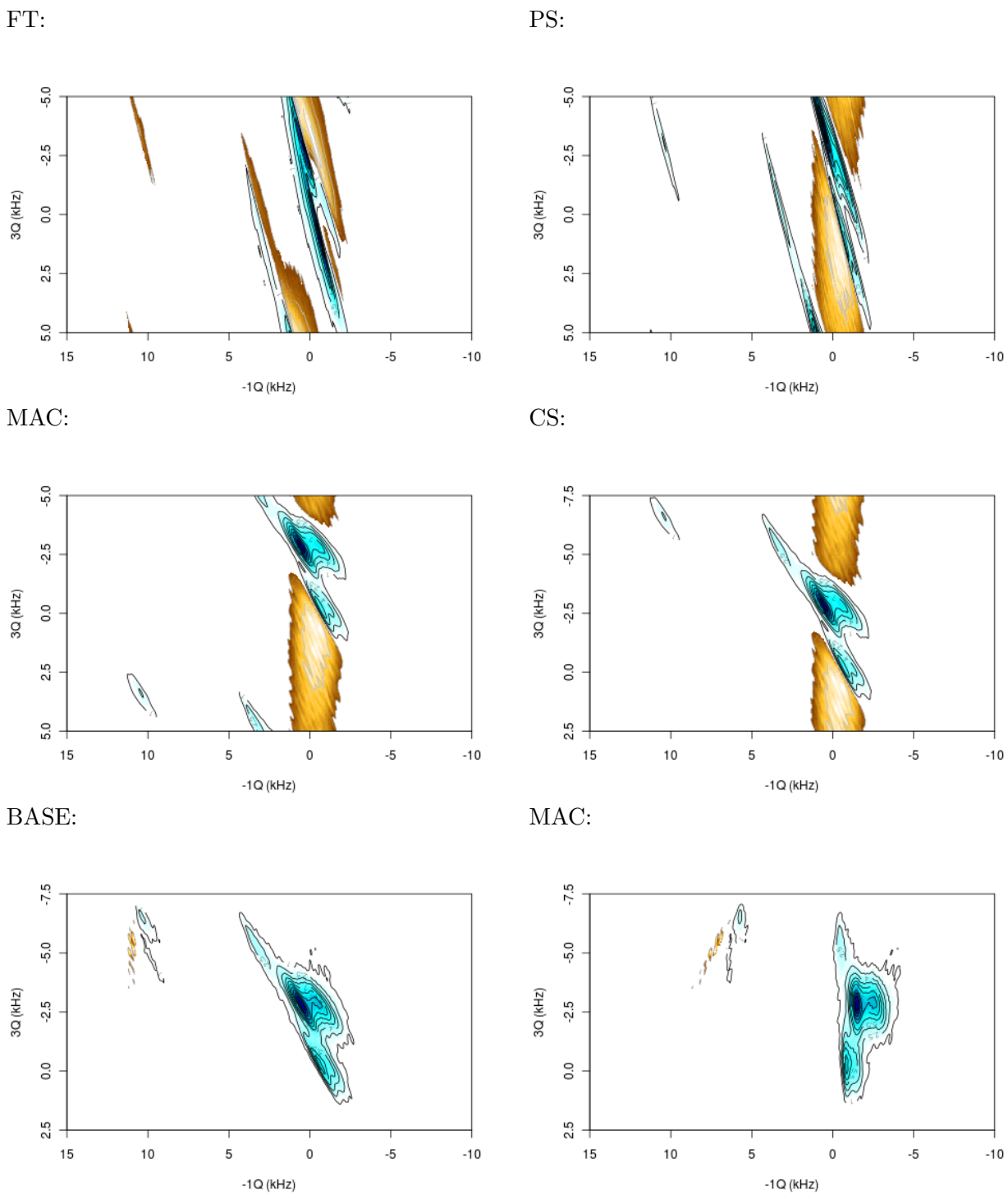

Figure 3: Graphic representation of processing steps for the 3QMAS (ν_1,ν_2) spectral domain (continued from Fig. 2. Blue intensity is positive, orange intensity is negative. Spectra is cropped along the -1Q for display purposes. A final (subtle) BASEline correction is performed at the end and shown in Fig. 4.

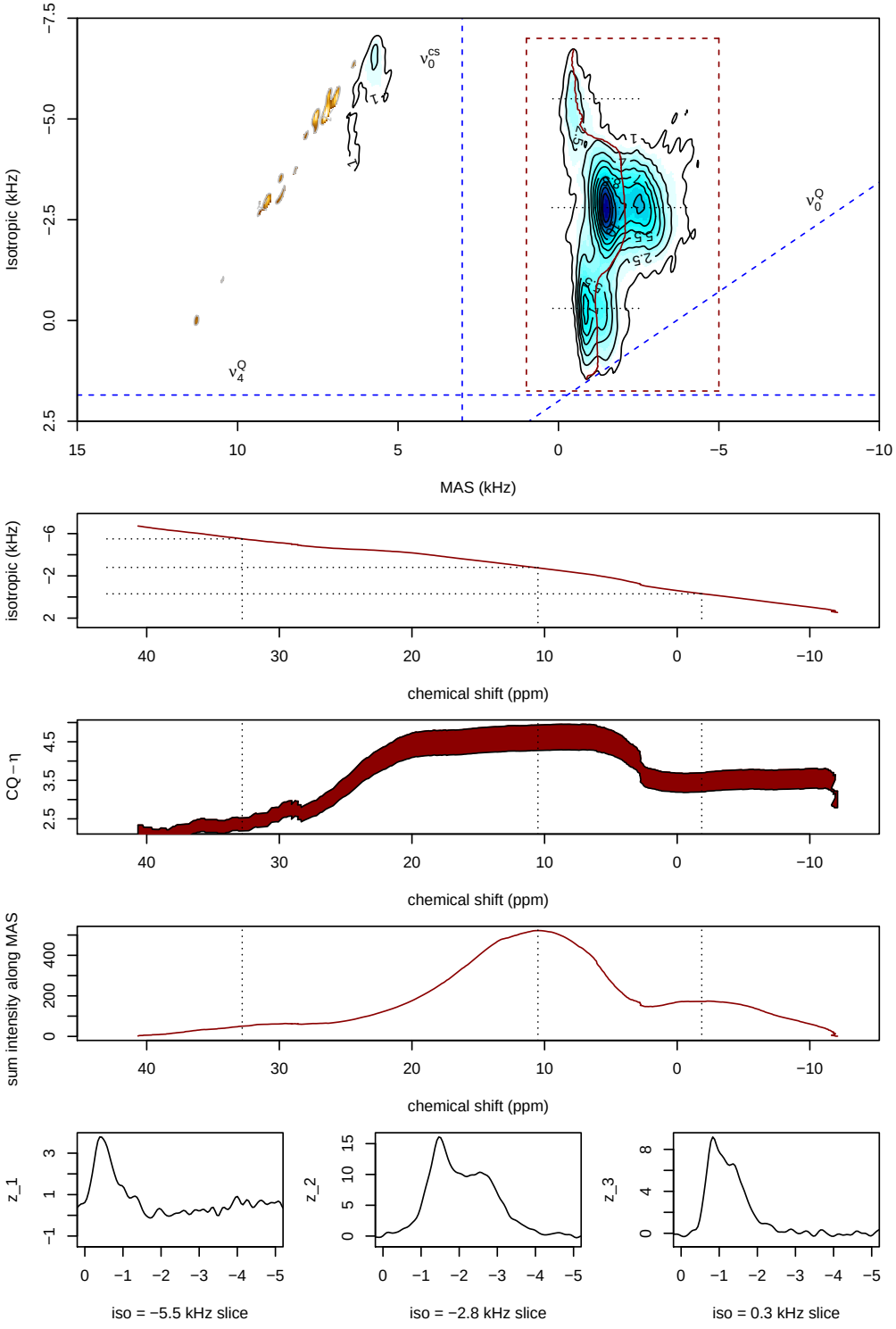

Figure 4: Final spectrum of biaxially sheared spectrum, continued from Fig. 3. The top panel also shows (dashed blue) the lines along which ν_0^{CS}, ν_0^Q and ν_4^Q vary; (dashed red) boundaries for the first moment calculation; (solid red) the first moment calculated along the MAS axis using `centreofmass2 16_Hz.dat 1024 512 1E7 1.58333 0.7058824 112.416 -5000 1000 -7000 1750`; and (dotted black) perhaps the three primary species seen in the spectrum. These three dotted black lines are also shown in the next three panels which show the correspondence to the chemical shift of (B) the isotropic MQMAS coordinate; (C) the CQ-η along the first moment line (the solid band is bounded by CQ values for $\eta = 0$ and $\eta = 1$); and (D) the total intensity of the first moment calculation. The bottom three panels show the three slices from the top panel of the identified three species. The R script to generate this plot is listed later.

3. NMRPipe output tends to just be test.fid, test.ft, etc. Simpson output, on the other hand, tends to be more specific, so this trick allows us to change the inputs throughout the processing file with a single line.

6. Simulated data isn't noisy, and at least under Simpson, it also i does not experience relaxation the way real data does. So use regular sorts of apodizing to simulate relaxation, i.e. the dwindling echo intensity at longer t_1 times.

9. First order phase correction is performed as for this step in the experimental processing, to account for the echo shift. The `p1` value should be negative of the τ echo delay (tXechsel on Varian) $\times$ the direct dimension spectral width (same as `-xSW` in the conversion script) $\times$ 360. Zero-order phase shift may not be neccessary for simulated data. If it is, it is probably either $\pm$90 or 180.

11. RS? dimension in step 7.

15. The PS in the indirect dimension shouldn't be necessary

Simulation

Simpson input and NMRPipe processing files for anisotopic dimension

Simpson can be used to simulate the projection of a defined MQMAS lineshape in the -1Q (direct, MAS, ν_2 dimension relatively quickly using the input file found in section 1.4.1. On the other hand, the full phase cycle can also be simulated, with the input file found in section 1.4.2.

Simulations of the full 2D MQMAS spectrum can be run must faster, however, if explicit coherence filters are used. The following input file takes ~26 min. on a true quad-core 3.3 MHz CPU. The data is processed as described in section 1.2

```
# VERSION 0.30
# 3QMAS shifted echo SIMPSON input file for I=5/2
# John Gehman (Chemistry)
# Univ of Melbourne Australia
# Aug 2011
#
# Corresponds to the Echo- (vs Antiecho) selection phase cycle

spinsys {
   channels 27Al
   nuclei   27Al
   shift 1 5061.46 0 0 0 0 0
   quadrupole 1 2 3.25e6 0.68 0 0 0
}

par {
   spin_rate        10000
   sw               50000
   sw1               10000
   variable dw         1e6/sw
   np               512
   ni              24
   crystal_file      rep678
```

```
   gamma_angles     37
   proton_frequency   600.0e6
   variable acqperTr   sw/spin_rate
   start_operator   I1z
   verbose          1000
   variable rf      83333.33
   variable rf3     16666.67
   variable p1      3.5
   variable p2      1.0
   variable p3      10.0
   variable Tr        1e6/spin_rate
   variable fixdel   1000.0
}

proc pulseq {} {
   global par
   maxdt 0.5

   # 3Q:
   matrix set 1 coherence {3}
   # 1Q:
   matrix set 2 coherence {1}
   # -1Q:
   matrix set 3 coherence {-1}
   # central transition
   matrix set detect elements {{3 4}}

   # propagators 100+ are dwell time delay between acq points at different
   # increments of the rotor cycle incremented at p1+p2+p3
   for {set d 0} {$d < $par(acqperTr) } {incr d} {
       reset [expr ($d)*$par(dw) + $par(p3)/2]
       delay $par(dw)
       store [expr 100+$d]
   }

   # prop 1.0 is pulse 1. Rotor period  begins in the middle of the p1 pulse
   reset [expr $par(Tr) - $par(p1)/2 ]
   pulse $par(p1) $par(rf) x
   store 10

   # prop 1.1 is indexed for end of pulse 1 and is for rest of 1/2 rotor period
   reset [expr $par(p1)/2 ]
   delay [ expr ( $par(Tr) - $par(p1) )/2 ]
   store 11

   # prop 1.2 is a rotor period delay, indexed at 0.5 Tr
   reset [expr $par(Tr)/2 ]
   delay $par(Tr)
   store 12

   # prop 2.0 is pulse 2, preceeded by 1/2 the rotor period, which ends
   # in the middle of p2
   reset [expr $par(Tr)/2 ]
   delay [ expr ( $par(Tr)- $par(p2) )/2 ]
```

```
78    pulse $par(p2) $par(rf) x
79    store 20
80
81    # prop 2.3 is indexed for end of p2 and is the fixed delay, which
82    # begins in the middle of p2 and ends at the end of p3.
83    reset [expr $par(p2)/2 ]
84    delay [ expr $par(fixdel) - ($par(p3) + $par(p2) )/2 ]
85    pulse $par(p3) $par(rf3) x
86    store 23
87
88    # Start sequence
89
90    for {set nii 1} {$nii<=$par(ni)} {incr nii} {
91
92       reset [expr $par(p1)/2 ]
93       prop 11
94       if [expr $nii > 1 ] {
95          prop 12
96          store 11
97       }
98
99       reset [expr $par(Tr)-$par(p1)/2 ]
100      prop 10
101      filter 1
102      prop 11
103      prop 20
104      filter 2
105      prop 23
106      filter 3
107
108      for {set j 0} {$j < [expr $par(np)]} {incr j} {
109         acq -y
110         prop [expr 100 + ( $j%$par(acqperTr) ) ]
111      }
112   }
113}
114
115proc main {} {
116   global par
117
118   fsave [fsimpson] $par(name).fid
119   fsave [fsimpson] $par(name)pipe.fid -nmrpipe
120   puts "Larmor frequency (Hz) of 27Al: "
121   puts [resfreq 27Al $par(proton_frequency)]
122}
```

1.3 Spectral Interpretation and Analysis

Refer to Gehman & Provis **??** for equations that help to translate spectal coordinates into isotropic chemical and quadrupolar shifts, and *vice versa*. The following perl script implements these equations (caution for spin 3/2). The inputs to be listed on the command line are

1. v0: the Larmor frequency (in MHz),

2. vmQ: the frequency coordinate (in Hz) on the ν_1 (indirect, mQ evolution) axis,
3. v1Q: the frequency coordinate (in Hz) on the ν_2 (direct, $\pm$1Q evolution) axis
4. CS: the chemical shift of the species in question (in Hz),
5. CQ: the quadrupolar coupling constant (in MHz),
6. eta: the asymmetry parameter η of the quadrupolar interaction,
7. p: the multiple quantum order that evolves in the first (indirect) dimension (3 for 3QMAS),
8. I: the spin (e.g. 2.5 for spin $^5/_2$ like ^{27}Al and ^{17}O),
9. a: shearing factor along the centre frequency of the $\pm$1Q axis (probably -k or p),
10. b: shearing factor along the centre frequency of the mQ axis,

```
use strict;

# $v0 and $CQin in MHz
if ($#ARGV != 9 ) {
   die "<v0> <v1> <v2> <CS> <CQ> <eta> <p> <I> <a> <b>\n";
}

my ($v0,$v1,$v2,$vCSin,$CQin,$eta,$p,$I,$sa,$sb) = @ARGV;

my $C0p = $p*($I*($I+1.0) - 3.0*$p*$p/4.0);
my $C01n = -1.0*($I*($I+1.0) - 0.25);
my $lambda = $C0p/$C01n;

my $C4p = $p*(18.0*$I*($I+1.0) - 8.5*$p*$p - 5);
my $C41n = -1.0*(18.0*$I*($I+1.0) - 13.5);
my $k = $C4p/$C41n;

printf("spin:                       %1d/2\n",$I*2);
printf("p-coherence evolution in t1:  %1d\n",$p);
printf("p-coherence evolution in t2: %2d\n",-1);
printf("lambda:                 %8.5f\n",$lambda);
printf("k:                      %8.5f\n",$k);

# Sanity check:
my $ca = 0;
my $cb = 0;

if ( abs($sa + $k) < 1E-4 ) {
   print "you've opted for isotropic shearing factor a = -k\n";
   $ca = 1;
} elsif ( abs($sa - $p) < 1E-4) {
   print "you've opted for isotropic shearing factor a = p\n";
   $ca = 2;
} else {
   print "you've opted for an unanticipated isotropic shearing factor.\n";

   $ca = 3;
```

```
}

if ( abs($sb) < 1E-9 ) {
   print "you've opted for ZERO MAS shearing factor b = 0\n";
} elsif ( abs($sb - 1.0/($p+$k) ) < 1E-4) {
   print "you've opted for MAS shearing factor b = 1/(p+k)\n";
   $cb = 1
} elsif ( abs($sb + 1.0/($p+$lambda) ) < 1E-4) {
   print "you've opted for MAS shearing factor b = -1/(p+lambda)\n";
   $cb = 2;
} else {
   print "you've opted for an unanticipated MAS shearing factor.\n";
   $cb = 3;
}

if ($cb > 1E-9 && abs($ca-$cb) > 1E-9) {
   print "you've opted for an unanticipated combination of shearing factors\n";
}

# Calculate sheared spectrum positions for chemical shift and CQ values given
# Eq 2 from Gehman & Provis
my $v0Qtheor = 1E5 * $CQin**2 * ($eta**2 + 3)
   / (($I*2.0)*($I*2.0-1))**2.0
   / ($v0) ;

# Eq 10 from Gehman & Provis
my $ab1 = (1.0+$sa*$sb);
my $v1theor = ($sa-$p)*$vCSin + ($C0p+$sa*$C01n)*$v0Qtheor;
my $v2theor = ($ab1-$p*$sb)*$vCSin + ($sb*$C0p+$ab1*$C01n)*$v0Qtheor;

# Calculate chemical, CQ shifts and frequences from spectral positions
# Eq 11 Gehman and Provis
my $s = 1.0/($C0p + $p*$C01n);
my $vCS = $s*( $v2*($sa*$C01n+$C0p) - $v1*($sb*$C0p+$C01n*$ab1) );
my $v0Q = $s*( $v2*($p-$sa) + $v1*($ab1-$sb*$p) );

my @eta = map {$_ * 0.1} (0..10);
my $CQtmp = 10.0*$v0Q*$v0*(2.0*$I*(2.0*$I-1))**2.0;
my @CQexp = map { ($CQtmp/($_+3.0))**0.5 } @eta;


printf("        inputs-based  freq-based\n");
printf("vmQ:   %9.3f      %9.3f Hz\n",$v1theor,$v1);
printf("v1Q:   %9.3f      %9.3f Hz\n",$v2theor,$v2);
printf("vCS:   %9.3f      %9.3f\n",$vCSin,$vCS);
printf("v0Q:   %9.3f      %9.3f Hz\n",$v0Qtheor,$v0Q);
   printf("CQ:    %9.3f MHz               (eta = %5.3f)\n",$CQin,$eta);
if ($v0Q < 0) {
   printf("CQ:                    v0Q < 0 (try going the other way with CS)\n");
} else {
   foreach my $e (0..10) {
      printf("                     %9.3f MHz (eta = %3.1f)\n",$CQexp[$e],$eta[$e]);
   }
}
```

The script is dual purpose; it can be run with the intent more of working out (1) where a chemical species with given physical parameters (ν_0^{CS}, CQ, η) would appear in a sheared spectrum, or (2) to translate sheared-spectrum frequency coordinates into physical parameters for a given chemical species. These are printed to standard-out in the "inputs-based" and "freq-based" columns, respectively.

1.3.1 Centre of Mass

The code listed in section 1.4.3 can be used to find the first moment of a given spectral feature along the -1Q, MAS, directly-detected axis. An example is given in Fig. **??**.

The code is compiled (requires the Gnu Scientific Library) into an executable (name specified by the **-o**utput option) with

```
g++ -o centreofmass centreofmass2.c -lgsl -gslcblas
```

The code must be run with 11 specifications on the command line:

1. the ascii data file to analyze
2. number of points in the horizontal (mas, -1Q', direct) dimension
3. number of points in the vertical (iso, mQ', indirect) dimension
4. threshhold value, below which data will be ignored
5. shear factor in the mQ dimension
6. shear factor in the -1Q dimension
7. the Larmor frequency, in MHz
8. the lower frequency of lineshapes in the horizontal (-1Q', direct) dimension
9. the upper freuqnecy of lineshapes in the horizontal (-1Q', direct) dimension
10. the lower frequency of lineshapes in the vertical (3Q', indirect) dimension
11. the upper freuqnecy of lineshapes in the vertical (3Q', indirect) dimension

This code will generate a data file with the following columns:

1. `-1Q' freq`: the centre of mass along the horizontal axis
2. `3Q' freq`: the sheared isotropic (indirect) frequency axis coordinate
3. `cs (Hz)`: the isotropic chemical shift frequency component
4. `v04 (Hz)`: the isotropic quadrupolar shift frequency component
5. `cs (ppm)`: the isotropic chemical shift in ppm
6. `CQ-eta0`: the upper value of the quadrupolar coupling bracket ($\eta = 0$)
7. `CQ-eta1`: the lower value of the quadrupolar coupling bracket ($\eta = 1$)
8. `total I`: the total intensity along the 1Q' axis at the y =3Q' coordinate.

1.4 Additional Code and file listings

1.4.1 Simpson simulation, projection onto -1Q axis

The projection of a 2D spectrum into the direct dimension of the shifted echo 3QMAS experiment gives the same anisotropic powder pattern spectrum irrespective of the mQ-axis shearing factor used to process the 2D spectrum. This 1D projection can be simulated using the following Simpson input file. Depending on the number of angles used (a combination of crystal file entries and gamma_angles), this code takes appr. 1-15 minutes to run on (one processor of) a Core2 Duo E6850 3 GHz Intel CPU.

```
# 17O_echo.in
# This was adapted from R. Hajjar at some stage.
spinsys {
     channels 17O
     nuclei   17O
     quadrupole 1 2 5e6 1 0 0 0
}

par {
     spin_rate         10000
     variable tsw      20
     sw                50000
     variable dw       1e6/sw
     np                512
     crystal_file      zcw4180
     gamma_angles      20
     proton_frequency  829.0834e6
     variable acqperTr sw/spin_rate
     start_operator    I1z
     verbose           1000
     variable rf       83333.33
     variable rf3      16666.67
     variable p1       3.5
     variable p2       1.0
     variable p3       10.0
     variable Tr       1e6/spin_rate
     variable fixdel   1000.0
}

proc pulseq {} {
   global par
   maxdt 0.5

   # 3Q:
   matrix set 1 coherence {3}
   # 1Q:
   matrix set 2 coherence {1}
   # central transition
   matrix set detect elements {{3 4}}


   # propagators 100+ are dwell time delay between acq points at different
   # increments of the rotor cycle incremented at p1+p2+p3
```

```
   for {set d 0} {$d < $par(acqperTr) } {incr d} {
      reset [expr ($d)*$par(dw) + $par(p2)/2 + $par(p1)  ]
      delay $par(dw)
      store [expr 100+$d]
   }


   reset

   pulse $par(p1) $par(rf) x
   filter 1

   pulse $par(p2) $par(rf) x
   filter 2

   delay [expr $par(fixdel)-($par(p2)+$par(p3))/2 ]

   pulse [expr $par(p3)/2 ] $par(rf3) x
   acq -y
   pulse [expr $par(p3)/2 ] $par(rf3) x
   delay [expr $par(dw) - $par(p3)/2]

   for {set j 1} {$j < [expr $par(np)-1]} {incr j} {
      acq -y
      prop [expr 100 + ( $j%$par(acqperTr) ) ]
   }
}

proc main {} {
       global par

     fsave [fsimpson] $par(name).fid
     fsave [fsimpson] $par(name)pipe.fid -nmrpipe
     puts "Larmor frequency (Hz) of 17O: "
     puts [resfreq 17O $par(proton_frequency)]
}
```

The NMRPipe output files can be processed with the following script

```
#!/bin/tcsh

nmrPipe -in 17O_echopipe.fid\
| nmrPipe  -fn ZF -size 1024\
| nmrPipe -fn GM -g1 0 -g2 10 -c 1.0\
| nmrPipe  -fn FT\
| nmrPipe -fn PS -p0 0 -p1 -18000.00    \
| nmrPipe -fn EXT -x1 385 -xn 640       \
| nmrPipe -fn PS -p0 0 -p1 0 -di \
   -verb -ov -out 17O_echo.ft2
```

These projection simulations give something like the 1D spectra in figure **??**

1.4.2 Simpson simulation, MQMAS full phase cycle

Simpson can also simulate the experiment with a full explicit phase cycle, which takes ages to run. The same CPU as in the last section, using rep678 with 67 gamma_angles, took 97 hours. Note that Simpson can now run on multiple CPUs simultaneously, so this could be cut down considerably.

```
#3QMAS_O17_0.23.in
spinsys {
   channels 17O
   nuclei   17O
   quadrupole 1 2 5000000 -1 0 0 0
}

par {
   variable n         1
   spin_rate         10000
   variable p1         3.5
   variable p2         1.0
   variable p3         10.0
   variable fixdel      1000.0
   variable rf7         83333.33
   variable rf20      16666.67
   variable Tr        1e6/spin_rate
   variable realnp      512
   variable realni      24
   ni               1
   np               realnp*realni*96
   sw               50000
   sw1               10000
   proton_frequency   829.0834e6
   start_operator       I1z
   detect_operator    I1p
   crystal_file      rep678
   verbose            1000
   variable acqperTr   sw/spin_rate
   variable dw         1e6/sw
   gamma_angles       67
}

proc pulseq {} {
   global par

   maxdt 1.0

   # propagators 100+ are dwell time delay between acq points at different
   # increments of the rotor cycle
   for {set d 0} {$d < $par(acqperTr) } {incr d} {
       reset [expr $d*$par(dw)]
       delay $par(dw)
       store [expr 100+$d]
   }

   # prop 2 is free evolution for 1 rotor period, but indexed at half Tr
    reset [expr $par(Tr)/2.0]
```

```
49      delay $par(Tr)
50      store 2
51
52    # propagators 3 through 14 are the 12 different phases for 1st pulse
53    # followed by remainder of half rotor period delay
54     for {set j 0} {$j<12} {incr j} {
55         set p [ expr $j+3 ]
56       reset
57       delay  [ expr $par(Tr) - $par(p1)/2.0 ]
58         pulse $par(p1) $par(rf7) [expr 30*$j]
59       delay [ expr ($par(Tr) - $par(p1) )/2.0 ]
60         store $p
61     }
62
63    # propagator 15 is half rotor period, less half p2, p2 pulse, and another
64    # half rotor period less half p2
65     reset [ expr $par(Tr)/2.0 ]
66     delay [ expr ($par(Tr)-$par(p2))/2.0]
67     pulse $par(p2) $par(rf7) 0
68     delay [ expr ($par(Tr)-$par(p2))/2.0]
69     store 15
70
71    # propagators 16 through 23 are the 8 different phases for p3
72    # preceded by half rotor period less p3
73    for {set j 0} {$j<8} {incr j} {
74         set p [expr $j+16]
75        reset [ expr $par(Tr)/2.0 ]
76         delay [ expr $par(Tr)/2 - $par(p3) ]
77         pulse $par(p3) $par(rf20) [expr 45*$j]
78         store $p
79     }
80
81    # Loop through phase cycle and (nested) indirect dimension; concatenate
82    # the 96 FIDs, to be sorted out later in the main subroutine, as per
83    # 8 Sept 2006 advice of Thomas Vosegaard simpson-simmol@yahoogroups.com
84
85    for {set i 0} {$i < 96} {incr i} {
86
87       set prop314  [expr $i%12 + 3]
88       set prop1623  [expr 16 + ($i - $i%12)/12 ]
89       # num of prop 2 used for fixed delay
90       set prop2n  [expr $par(fixdel)/$par(Tr) - 1]
91
92       # Calculate receiver phase
93       set b0   [expr $i%48]
94       set b1   [expr ($b0 - $b0%12)/12]
95       set b2   [expr ( $b0%4 + 2*($b0%2) )%4 ]
96       set b3   [expr ( $b1+$b2 )%4 ]
97       set ap    [expr 90 * $b3]
98
99       # loop for indirect dimension
100      for {set nii 0} {$nii<$par(realni)} {incr nii} {
101
102         # run pulse sequence
```

```
         reset
         prop $prop314
         for {set del 0} {$del <= $nii} {incr del} {
            if {$del>0} {prop 2}
         }
         prop 15
         for {set del 0} {$del < $prop2n} {incr del} {
            prop 2
         }
         prop $prop1623

         for {set j 0} {$j < $par(realnp)} {incr j} {
            acq $ap
            prop [expr 100 + $j%$par(acqperTr)]
         }
      }
   }
}

proc main {} {
   global par
   set f [fsimpson]
   set g [fdup $f]
   fset $g -np $par(realnp)
   fset $g -ni $par(realni)
   for {set j 2} {$j <= 96} {incr j} {
      for {set i 1} {$i <= $par(realnp)*$par(realni)} {incr i} {
         set c1 [findex $f [expr $i+($j-1)*$par(realnp)*$par(realni)] ]
         set c2 [findex $g $i ]
         fsetindex $g $i\
            [expr [lindex $c1 0]+[lindex $c2 0] ] \
            [expr [lindex $c1 1]+[lindex $c2 1] ]
      }
   }

   fsave $g CQ_10MHz_023-067-678.fid
   fsave $g CQ_10MHz_023-067-678168pipe.fid -nmrpipe
}
```

1.4.3 centreofmass2

```

#include <stdio.h>
#include <cstring>
#include <fstream>
#include <iomanip>
#include <cstdlib>
#include <cmath>
#include <ctime>
#include <iostream>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
```

```
13 #include <gsl/gsl_math.h>
14 using namespace std;
15
16
17 int main ( int argc, char *argv[]  ) {
18    if (argc != 12) {
19       cout << "./code file npH npV th a b v0 vHLO vHHI vVLO vVHI \n";
20       exit(0);
21    }
22
23    // threshhold value below which data will be ignored
24    int npH; sscanf(argv[2],"%d",&npH);
25    int npV; sscanf(argv[3],"%d",&npV);
26    double th; sscanf(argv[4],"%le",&th);
27    double shear3Q; sscanf(argv[5],"%le",&shear3Q);
28    double shear1Q; sscanf(argv[6],"%le",&shear1Q);
29    double v0; sscanf(argv[7],"%le",&v0);
30    double vHLO; sscanf(argv[8],"%le",&vHLO);
31    double vHHI; sscanf(argv[9],"%le",&vHHI);
32    double vVLO; sscanf(argv[10],"%le",&vVLO);
33    double vVHI; sscanf(argv[11],"%le",&vVHI);
34
35    gsl_matrix * spectrum = gsl_matrix_alloc(npH,npV);
36    gsl_vector * vV = gsl_vector_calloc(npV);
37    gsl_vector * vH = gsl_vector_calloc(npH);
38
39    /* sheared isotropic frequencies v^prime = (v3Q^prime,v1Q^prime) are
40    transformed from NMR parameters nmr = (v0cs,v0Q) by the shearing matrix S
41    and the admixtures of frequencies under differential evolution in t1 and t2
42    A:
43          v^prime = S.A.nmr
44
45    NMR parameters are therefore
46
47          nmr = inv(A).inv(S).v^prime
48    */
49
50    gsl_matrix * invshear;
51    invshear = gsl_matrix_calloc(2,2);
52     gsl_matrix_set(invshear,0,0,shear3Q*shear1Q+1.0);
53     gsl_matrix_set(invshear,0,1,-shear3Q);
54     gsl_matrix_set(invshear,1,0,-shear1Q);
55     gsl_matrix_set(invshear,1,1,1.0);
56
57    // hardcoded for spin 5/2, 3 -> -1, for now
58    gsl_matrix * invadmix;
59    invadmix = gsl_matrix_calloc(2,2);
60    gsl_matrix_set(invadmix,0,0,8.0);
61     gsl_matrix_set(invadmix,0,1,6.0);
62     gsl_matrix_set(invadmix,1,0,1.0);
63     gsl_matrix_set(invadmix,1,1,3.0);
64    gsl_matrix_scale(invadmix,-1.0/18.0);
65
66    gsl_matrix * invboth;
```

```
67     invboth = gsl_matrix_calloc(2,2);
68     gsl_blas_dgemm(CblasNoTrans,CblasNoTrans,1.0,invadmix,invshear,0.0,invboth);
69
70     printf("# number points -1Q dimension: %6d\n",npH);
71     printf("# number points  mQ dimension: %6d\n",npV);
72     printf("# thresshold data value: %20.5f\n",th);
73     printf("# shear factor  mQ: %8.5f\n",shear3Q);
74     printf("# shear factor -1Q: %8.5f\n",shear1Q);
75     printf("# larmor frequency: %10.6f\n",v0);
76     printf("# H axis centreofmass calc restricted to: %9.5f to %9.5f\n",
77        vHLO,vHHI);
78     printf("# V axis centreofmass calc restricted to: %9.5f to %9.5f\n",
79        vVLO,vVHI);
80  cout << "#invboth: " << endl;
81  cout <<  "# " << gsl_matrix_get(invboth,0,0) << " "
82      <<  "# " << gsl_matrix_get(invboth,0,1) << " "
83      << endl;
84  cout <<  "# " << gsl_matrix_get(invboth,1,0) << " "
85      <<  "# " << gsl_matrix_get(invboth,1,1) << " "
86      << endl;
87
88      ifstream indata;
89      indata.open(argv[1], ios::in);
90      if ( !indata.is_open() ) {
91          cout << "Can't open data file " << argv[1] << endl;
92          exit(0);
93      }
94
95     string buf;
96     // skip first blank line
97
98     double x, y, z;
99     bool first = 1;   // flag for first time through
100    int h=0;   // mas dimension point index
101    int v=-1;   // iso dimension point index
102
103     getline(indata,buf);
104    while ( !indata.eof() ) {
105       // if line is empty, grab the next line and add to the mas
106       // frequency list
107       if ( buf.empty() ) {
108          h=0;
109          v++;
110          getline(indata,buf);
111          sscanf(buf.c_str(), "%le %le %le", &x,&y,&z);
112          gsl_vector_set( vV, v, y );
113          gsl_matrix_set( spectrum, h, v, z );
114          if (first) {
115             gsl_vector_set( vH, h, x );
116          }
117          h++;
118       } else {
119          sscanf(buf.c_str(), "%le %le %le", &x,&y,&z);
120          gsl_matrix_set( spectrum, h,v, z );
```

```
121         if (first) {
122            gsl_vector_set( vH, h, x );
123         }
124         if (h>=npH) {
125            first = 0;
126         }
127         h++;
128      }
129    getline(indata,buf);
130   }
131
132   gsl_vector * freq;
133   gsl_vector * nmr;
134   freq = gsl_vector_alloc(2);
135   nmr = gsl_vector_alloc(2);
136
137   printf("#%9s %10s %10s %10s %10s %7s %7s %10s\n",
138      "H freq","V freq","cs (Hz)","v04 (Hz)","cs (ppm)",
139      "CQ-eta0","CQ-eta1","total I");
140   for ( v=0; v< npV; v++) {
141      double wsum = 0;
142      double sum = 0;
143
144      if ( gsl_vector_get(vV,v) >= vVLO
145         && gsl_vector_get(vV,v) <= vVHI ) {
146
147         for ( h=0; h< npH; h++) {
148            if ( gsl_matrix_get(spectrum,h,v) > th
149               && gsl_vector_get(vH,h) >= vHLO
150                  && gsl_vector_get(vH,h) <= vHHI ) {
151               wsum += gsl_vector_get(vH,h) * gsl_matrix_get(spectrum,h,v);
152               sum += gsl_matrix_get(spectrum,h,v);
153            }
154         }
155      }
156      if (sum > 0 ) {
157         double CQ_h0, CQ_h1;
158         double com = wsum/sum;
159
160         gsl_vector_set(freq,0, gsl_vector_get(vV,v) );
161         gsl_vector_set(freq,1, com );
162         gsl_blas_dgemv(CblasNoTrans,1.0,invboth,freq,0.0,nmr);
163
164         CQ_h0 = pow( v0 * gsl_vector_get(nmr,1) / 7.5E2 , 0.5 );
165         CQ_h1 = pow( v0 * gsl_vector_get(nmr,1) / 1E3  , 0.5 );
166         printf("%10.3f %10.3f %10.3f %10.3f %10.3f %7.3f %7.3f %15.5g\n",
167            com,
168            gsl_vector_get(vV,v),
169            gsl_vector_get(nmr,0),
170            gsl_vector_get(nmr,1),
171            gsl_vector_get(nmr,0)/v0,
172            CQ_h0,
173            CQ_h1,
174            sum);
```

```
      }
   }

}
```

1.4.4 Example R script for plotting

```
require(grDevices);
require(graphics);

# IMAGE
spectrum <- scan("16_BASE_step.dat", list(0,0,0) );
sQ <- spectrum[[1]];
mQ <- spectrum[[2]];
z <- spectrum[[3]];

sQ2 <- unique(sQ);
mQ2 <- unique(mQ);
sQ2l = length(sQ2);
mQ2l = length(mQ2);

z2<-matrix(z,nrow=sQ2l,ncol=mQ2l);
scale(z2,center=FALSE);
z2p <- z2;
z2n <-z2;

z2p[z2p<1] = NA;
z2n[z2n>-1] = NA;

# COM
com <- scan("16_com2.dat", list(0,0,0,0,0,0,0,0), comment.char="#" );
comx <-com[[1]];
comy <-com[[2]];
comx=comx/1000.0;
comy=comy/1000.0;

sQ.at <- seq(-25,25,by=5);
mQ.at <- seq(-7.5,2.5,by=2.5);

# NMR PARAMETER PLOT
ppm <- com[[5]];
CQ0 <- com[[6]];
CQ1 <- com[[7]];
arb <- com[[8]]/1E7;

# COLOURS
blues <- colorRampPalette(c("white","cyan","darkblue"),space="rgb");
oranges <- colorRampPalette(c("white","goldenrod1","darkorange4"),space="rgb");

#setEPS();
#postscript("16_R.eps",width=8,height=6)
#png(filename="16_BASE.png",width=480,height=360,pointsize=12,units="px");
pdf("16_R.pdf",width=6.7,height=9.5)
```

```
47 par(omi=c(0.1,0.1,0.1,0.1));
48 par(mai=c(0.55,0.55,0.1,0.1));
49 layout( matrix(c(1,1,1,2,2,2,3,3,3,4,4,4,5,6,7),5,3,byrow=TRUE),
50    widths=c(2.167,2.167,2.167),
51    heights=c(3.6,1.5,1.5,1.5,1.5),
52    respect=TRUE
53 );
54
55 image(sQ2,mQ2,z2p,
56    xlim=c(15,-10),
57    ylim=c(2.5,-7.5),
58    xlab="MAS (kHz)",
59    ylab="Isotropic (kHz)",
60    col=blues(20),
61    axes=FALSE);
62 image(sQ2,mQ2,z2n,
63    xlim=c(15,-10),
64    ylim=c(2.5,-7.5),
65    col=oranges(20),
66    add=TRUE,
67    axes=FALSE);
68 contour(sQ2,mQ2,z2,
69    levels=seq(1,15,by=1.5),
70    add=TRUE,
71    col="black");
72 contour(sQ2,mQ2,z2,
73    levels=seq(-1,-15,by=-1.5),
74    add=TRUE,
75    col="gray");
76
77 text(-8.0,-3,expression(nu[0]^Q))
78 abline(2.0,0.54269,lty=2,col="blue")
79
80 text(10.0,1.25,expression(nu[4]^Q))
81 abline(h=1.85,lty=2,col="blue")
82
83 text(4.0,-6.5,expression(nu[0]^{cs}))
84 abline(v=3,lty=2,col="blue")
85
86 lines(comx,comy,col="darkred"); # com
87 arrows(-5,-7,x1=1,col="darkred",lty=2,code=0); # frame for com calc
88 arrows(-5,1.750,x1=1,col="darkred",lty=2,code=0);
89 arrows(-5,-7,y1=1.750,col="darkred",lty=2,code=0);
90 arrows(1,-7,y1=1.750,col="darkred",lty=2,code=0);
91 arrows(-4,-2.8,x1=0.25,col="black",lty=3,code=0); # horiz lines through each species
92 arrows(-2.5,-5.5,x1=0.25,col="black",lty=3,code=0);
93 arrows(-2.5,-0.3,x1=0.25,col="black",lty=3,code=0);
94
95 axis(1,at=sQ.at);
96 axis(2,at=mQ.at);
97 box()
98
99 # isotropic axis vs ppm
100 plot(ppm,comy,
```

```
   type="l",
   xlim=c(43,-13),
   ylim=c(2.5,-7.5),
   col="darkred",
   ylab="isotropic (kHz)",
   xlab="chemical shift (ppm)");
arrows(43,-2.8,x1=10.5,col="black",lty=3,code=0); # horiz lines through each species
arrows(43,-5.5,x1=32.8,col="black",lty=3,code=0);
arrows(43,-0.3,x1=-1.85,col="black",lty=3,code=0);
arrows(10.5,-2.8,y1=2.5,col="black",lty=3,code=0); # horiz lines through each species
arrows(32.8,-5.5,y1=2.5,col="black",lty=3,code=0);
arrows(-1.85,-0.3,y1=2.5,col="black",lty=3,code=0);

plot(ppm,CQ0,
   type="l",
   xlim=c(43,-13),
   col="darkred",
   ylab=expression(CQ-eta),
   xlab="chemical shift (ppm)");
xx <- c(ppm,rev(ppm));
yy <- c(CQ0,rev(CQ1));
polygon(xx,yy,col='darkred');
abline(v=10.5,col="black",lty=3); # horiz lines through each species
abline(v=32.8,,col="black",lty=3);
abline(v=-1.85,col="black",lty=3);

plot(ppm,arb,
   type="l",
   col="darkred",
   xlim=c(43,-13),
   xlab="chemical shift (ppm)",
   ylab="sum intensity along MAS axis");
abline(v=10.5,col="black",lty=3); # horiz lines through each species
abline(v=32.8,,col="black",lty=3);
abline(v=-1.85,col="black",lty=3);

# Individual slices
sp.df <- data.frame(spectrum);
names(sp.df) <- c("spx","spy","spz");

#sp_1 <- subset(sp.df,sp.df[[2]]==-5507.812);
#sQ_1 <- sp_1[[1]];
ztmp <- subset(sp.df, spy < -5.500 & spy > -5.510);
z_1 <- ztmp[,3];
plot(sQ2,z_1,
   type="l",
   col="black",
   xlim=c(0,-5),
   xlab="iso = -5.5 kHz slice"
);

ztmp <- subset(sp.df, spy < -2.792 & spy > -2.794);
z_2 <- ztmp[,3];
plot(sQ2,z_2,
```

```
   type="l",
   col="black",
   xlim=c(0,-5),
   xlab="iso = -2.8 kHz slice"
);

ztmp <- subset(sp.df, spy < -0.311 & spy > -0.313);
z_3 <- ztmp[,3];
plot(sQ2,z_3,
   type="l",
   col="black",
   xlim=c(0,-5),
   xlab="iso = 0.3 kHz slice"
);

dev.off()
```