Supplementary Material

Sample preparation of the outer membrane protein G (OmpG)

The OmpG sample was prepared as described earlier[1, 2] with the following exceptions: after purification under denaturing conditions (8 M urea), the proton content of the backbone amide was set to 50 %. Similarly, the buffer for the subsequent refolding, reconstitution and 2D crystallization also contained 50 % H_2O and 50 % D_2O.

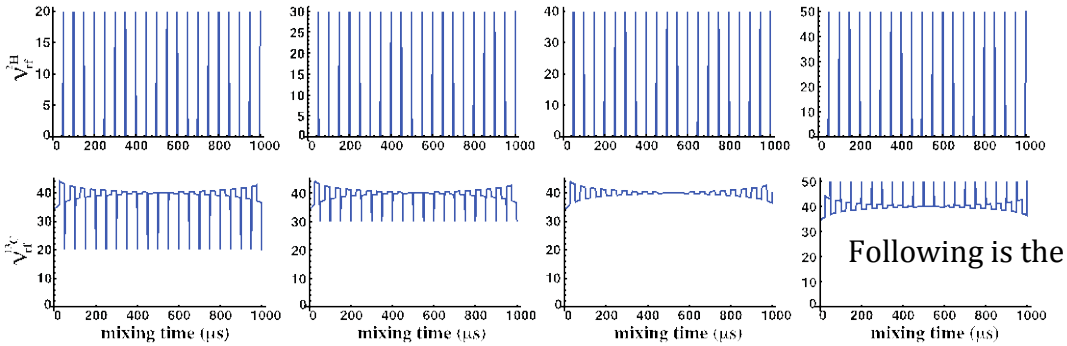
Adiabatic RESPIRATIONCP waveforms

The waveforms for the adiabatic RESPIRATIONCP were generated using a SIMPSON script that converts a list of amplitudes and phases into a shape file for Topspin. The list of the amplitudes and phases can be generated using following analytic expression

$$\nu_{rf} = 2\nu_r \pm b\tan\left(\alpha\left(\frac{\tau}{2} - t\right)\right),$$

where $\alpha = \frac{2}{\tau}\tan^{-1}\left(\frac{\Delta}{b}\right)$. In the present case, the parameters Δ = 6000 radian per sec and b = $6000/2\pi$ Hz were used. The adiabatic parameters b and Δ are dependent on the dipole-dipole coupling strength and govern the shape of the wave. τ is the total time for the pulse sequence and ν_r is the sample spinning frequency.

The various waves used for demonstrating the efficiency versus rf amplitudes plots in the paper were generated using the above scripts and the plots for the amplitude profiles on the two channels are shown in Fig. S1.



Figu. S1. Rf waves of lengths 1 ms (the short pulses are 4μs long) corresponding to different rf amplitudes for ^{2}H channel (top row) and ^{13}C channel (bottom row). Here the y-axis labels represent all the three kinds of power levels used as explained in figure-1c in the article.

Following is the SIMPSON script (.in file) that generates two lists (.store files) containing the rf amplitudes and phases for the two channels under consideration (^{2}H and ^{13}C in this case) and also generates the build up curve showing the transfer efficiency of the generated lists.

```
spinsys {
  channels 2H 13C
  nuclei 2H 13C
  shift 1 0 0 0 0 0 0
  quadrupole 1 2 0.18e6 0.1 0 0 0
  shift 1 0p 7.7p 0.65 -162.46 65.58 66.979
```

```
  dipole 1 2 -3689 0 14.167 -94.159
  jcoupling 1 2 21.5 0 0 0 0 0
}

par {
  method           direct
  proton_frequency 400e6
  start_operator   I1x
  detect_operator  I2x
  spin_rate        20000
  sw               spin_rate
  gamma_angles     5
  crystal_file     rep20

   #power level parameters
  variable   qr    2*spin_rate
   #Short pulse on C channel
  variable   vrfC   20000
  #Short pulse on D channel
  variable   vrfD   20000

   #rotor period
  variable   tr    1.0e6/spin_rate

  variable   pi           3.14
       #adiabatic parameters:
  variable   delta 6000
       #adiabatic parameter:
  variable   b1    6000.0/(2.0*pi)
       #total length of pulse sequence:
  variable   tau   1000
       #length of each pulse:
  variable   dtau  2
       #total number of pulses:
  variable   nstep tau/dtau
  #number of short pulses per rotor period
  variable   npul  1
       #adiabatic parameter - do not change!:
  variable   alfa  2.0*atan(delta/b1)/tau

 #number of recoupling pulses per rotor period
  variable n  int(tr/dtau)-npul

  np       int(tau/tr)+1
}

proc block1 {} {
  global par ptype rf_N ph_N
  maxdt 1.0

   #phase calculation for the two Continuous pulses per period
  for {set i 1} {$i<=$par(n)} {incr i} {
       if {$i==1} {
         set b 0.0
       }
       if {$i==[expr $par(n)/2+1]} {
         set b 180.0
       }
      set  ph_N_first($i)    $b
       }

    #Amplitude calculation for the two Continuous pulses (adiabtic waves)
  for {set j 1} {$j<=$par(nstep)} {incr j} {

       set jj [expr int(fmod([expr $j-1], $par(n)))+1]

       set ph_N($j) $ph_N_first($jj)

#time increment in steps of dtau
       set tt [expr $j*$par(dtau)]

       #the adiabatic expression
       set wrf_dream [expr - $par(b1)*tan($par(alfa)*(0.5*$par(tau)-$tt))]

       if {$jj==1} {
         set b 1
       }
```

```
        if {$jj==[expr $par(n)/2+1]} {

         set b -1
        }

        set rf_N($j) [expr $par(qr)+$b*$wrf_dream]
}
}

#To store the lists of Amplitudes and Phases for both the channels
proc rfstoreC {} {
  global par ptype rf_N ph_N
        set rfstoreC       [open $par(name)_rfC.store w]


  for {set j 1} {$j<$par(np)} {incr j} {

        set k   [expr ($j-1)*$par(n)]
        for {set i [expr $k+1]} {$i<=[expr $k+$par(n)]} {incr i} {
           puts $rfstoreC "$rf_N($i)"
        }
        for {set p 1} {$p <= $par(npul)} {incr p} {
           puts $rfstoreC "$par(vrfC) 0.0"
           }
  }

        close $rfstoreC
}

proc rfstoreD {} {
  global par ptype rf_N ph_N
        set rfstoreD       [open $par(name)_rfD.store w]

  for {set j 1} {$j<$par(np)} {incr j} {

        set k   [expr ($j-1)*$par(n)]
        for {set i [expr $k+1]} {$i<=[expr $k+$par(n)]} {incr i} {
           puts $rfstoreD "0.0"
        }
  for {set p 1} {$p <= $par(npul)} {incr p} {
           puts $rfstoreD "$par(vrfD) 0.0"
           }
  }

        close $rfstoreD
}


proc pulseq_exitation {} {
  global par t tcount rf_N ph_N

    block1
    store 1

  reset

    store 2
    acq

  reset

  prop 1

  for {set j 1} {$j<$par(np)} {incr j} {
  reset
        set k   [expr ($j-1)*$par(n)]
        prop 2
        for {set i [expr $k+1]} {$i<=[expr $k+$par(n)]} {incr i} {
        pulse $par(dtau) 0.0 0.0 $rf_N($i) $ph_N($i)
        }
        for {set p 1} {$p <= $par(npul)} {incr p} {
        pulse $par(dtau)  $par(vrfD) 0.0 $par(vrfC) 0.0
        }

        store 2

acq
```

```
  }
}

proc main {} {
   global par type scanpar
     set par(verbose) 1
     set par(pulse_sequence) pulseq_exitation
   set f [fsimpson]
   fsave $f $par(name).fid
   rfstoreC
   rfstoreD
fsave $f $par(name).xy -xreim
}
```

The script for converting the list of ampitudes and phases into a bruker wave:

```
#!/usr/bin/tclsh
global argv

if {[llength $argv] != 3} {
  puts stderr "error: specify basename of spin-file to open"
  puts stderr "and the time stepsize in microsec"
  puts stderr "and the nominal rf field strength on I"
  exit
}

set name [lindex $argv 0]
set deltat [lindex $argv 1]
set nomrfI [lindex $argv 2]

set file [open $name r]
set indx 0

set maxrfI -100000
set minrfI 100000
set maxfiI -7200
set minfiI 7200
set avrfI  0

while {[gets $file s] != -1} {
   incr indx
   set lst {}
   foreach i [split $s] {
      if {$i != {}} {
        lappend lst $i
      }
   }
   set rfI($indx) [lindex $lst 0]
   set fiI($indx) [lindex $lst 1]

   set avrfI [expr $avrfI + $rfI($indx)]

   if {$fiI($indx) < 0} {
      set fiI($indx) [expr $fiI($indx)+360.0]
   }
   if {$rfI($indx) > $maxrfI} {
      set maxrfI $rfI($indx)
   }
   if {$rfI($indx) < $minrfI} {
      set minrfI $rfI($indx)
   }
   if {$fiI($indx) > $maxfiI} {
      set maxfiI $fiI($indx)
   }
   if {$fiI($indx) < $minfiI} {
      set minfiI $fiI($indx)
   }
}

puts "number of steps in wave form     : $indx"
puts "deltat in wave form              : $deltat"
puts ""
puts "maximum  I-spin rf field strength: $maxrfI"
puts "minimum  I-spin rf field strength: $minrfI"
```

```
puts "average  I-spin rf field strength: [expr $avrfI/$indx]"
puts "nom. max I-spin rf field strength: $nomrfI"
puts "maximum  I-spin rf field phase   : $maxfiI"
puts "minimum  I-spin rf field phase   : $minfiI"

set outfI [open $name\_I.waves w]

set minI [expr $minrfI*100.0/$nomrfI]
set maxI [expr $maxrfI*100.0/$nomrfI]

puts $outfI "##TITLE= /u/exp/stan/nmr/lists/wave/$name\_I.wave"
puts $outfI "##JCAMP-DX= 5.00 Bruker JCAMP library"
puts $outfI "##DATA TYPE= Shape Data"
puts $outfI "##ORIGIN= Bruker Analytik GmbH"
puts $outfI "##OWNER= <ab>"
puts $outfI "##DATE= 03/11/10"
puts $outfI "##TIME= 08:15:00"
puts $outfI "##MINX= $minI"
puts $outfI "##MAXX= $maxI"
puts $outfI "##MINY= $minfiI"
puts $outfI "##MAXY= $maxfiI"
puts $outfI "##\$SHAPE_EXMODE= Excitation"
puts $outfI "##\$SHAPE_TOTROT= 0.000000e+00"
puts $outfI "##\$SHAPE_BWFAC= 0.000000e+00"
puts $outfI "##\$SHAPE_INTEGFAC= 7.460936e-01"
puts $outfI "##\$SHAPE_MODE= 0"
puts $outfI "##NPOINTS= $indx"
puts $outfI "##XYPOINTS= (XY..XY)"
for {set i 1} {$i <= $indx} {incr i} {
  set ampl [expr $rfI($i)*100/$nomrfI]
  puts $outfI "  [format "%8.6e" $ampl], [format "%8.6e" $fiI($i)]  "
}
puts $outfI "##END="
```

1. M. Hiller, L. Krabben, K. R. Vinothkumar, F. Castellani, B.-J. van Rossum, W. Kühlbrandt and H. Oschkinat, *ChemBioChem*, 2005, **6**, 1679-1684.
2. D. Lalli, P. Schanda, A. Chowdhury, J. Retel, M. Hiller, V. Higman, L. Handel, V. Agarwal, B. Reif, B. Rossum, Ü. Akbey and H. Oschkinat, *Journal of biomolecular NMR*, 2011, **51**, 477-485.