Supporting Information

Cross-Correlated Relaxation of Dipolar Coupling and Chemical-Shift Anisotropy in Magic-Angle Spinning $R_{1\rho}$ NMR Measurements: Application to Protein Backbone Dynamics Measurements

Vilius Kurauskas, Emmanuelle Weber, Audrey Hessel, Isabel Ayala, Dominique Marion* and Paul Schanda*

Supporting Figures

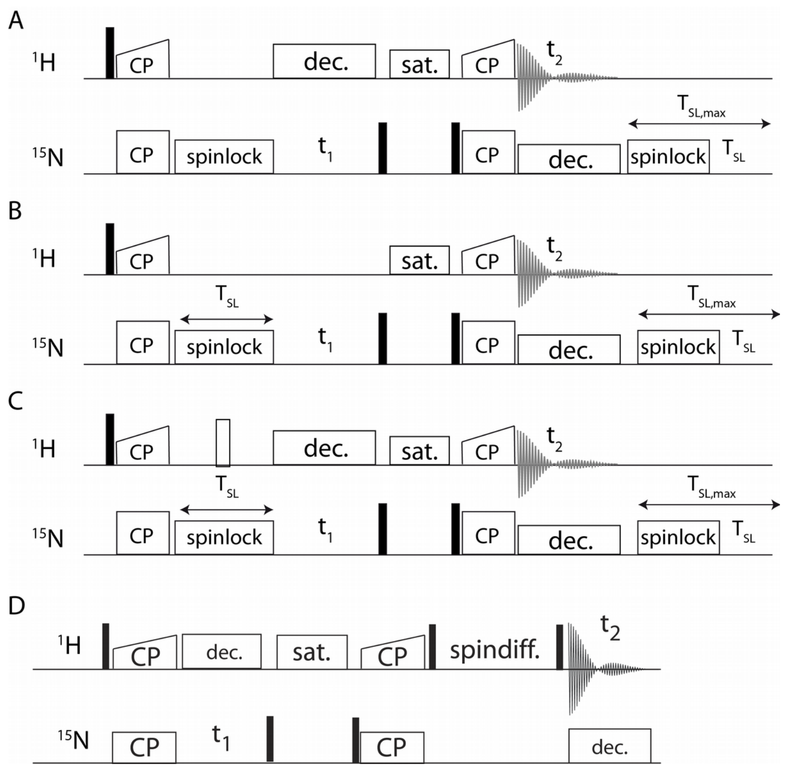

Figure S1: Pulse sequences used in this study. In all cases, narrow black rectangles and wider open rectangles denote π/2 and π pulses, respectively applied at 100 kHz (^{1}H) and 75 kHz (^{15}N). Cross-polarization (CP) was applied with an RF field strengh of approximately 60 kHz (^{15}N) and 95 kHz (^{1}H, ramped by approximately ±10%). Decoupling on ^{1}H (^{15}N) was performed WALTZ-16 at a field strength of 5 kHz (3 kHz). Solvent suppression (“sat”) was achieved by applying a two-pulse phase-modulation sequence with 850 μs length of the individual pulses and a total duration of 50 ms, and a field strength of 10 kHz. In all cases, the total duration of the spin-lock was kept constant, i.e. an extra spin-lock period was inserted after the FID, as indicated. (A) Sequence used for measurement of N_x $R_{1\rho}$ decay without ^{1}H decoupling. (B) Sequence used for acquisition of differential $R_{1\rho}$ decay of the two doublet components, achieved by omitting the ^{1}H decoupling during ^{15}N evolution. (C) Pulse sequence used for recording in-phase N_x $R_{1\rho}$ decay with ^{1}H decoupling. (D) Pulse sequence used to measure ^{1}H spin diffusion (data shown in Figure 5).

All the simulations take into account only the spin-lock part, neglecting the CP transfer steps.

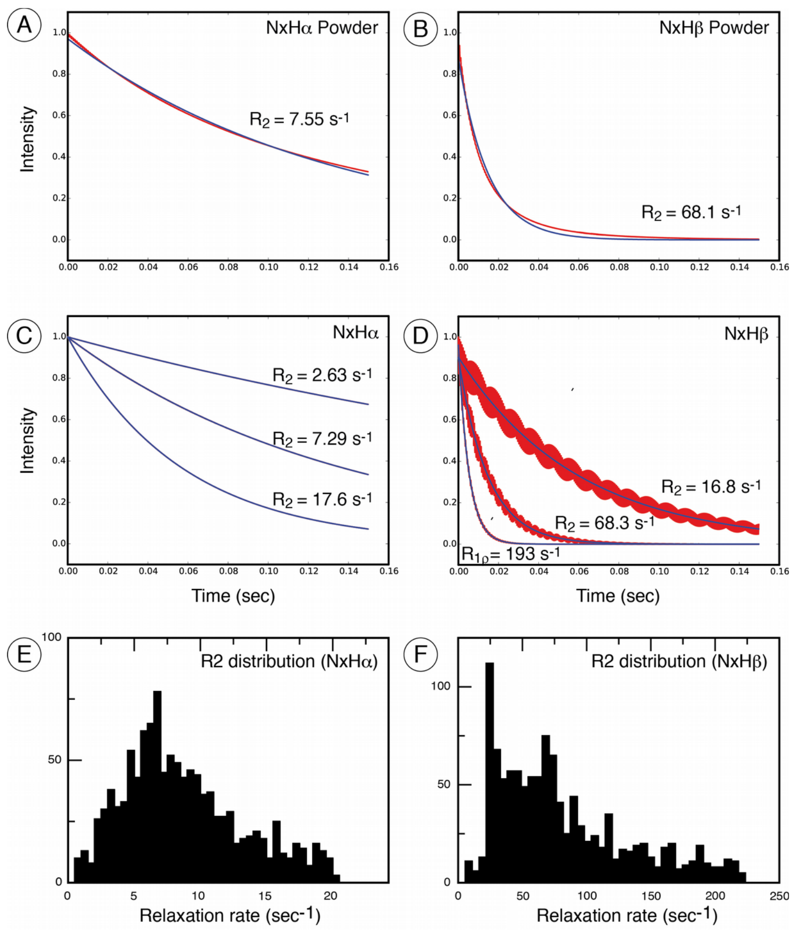

Figure S2. Simulated $R_{1\rho}$ relaxation curves of the doublet components N_x-H^α and N_x-H^β averaged over a powder of orientations (A,B) and for individual crystallites (C-F). (C) and (D) show three representative orientations, and (E) and (F) show the distribution of relaxation rate constants across the simulation of powder orientations.

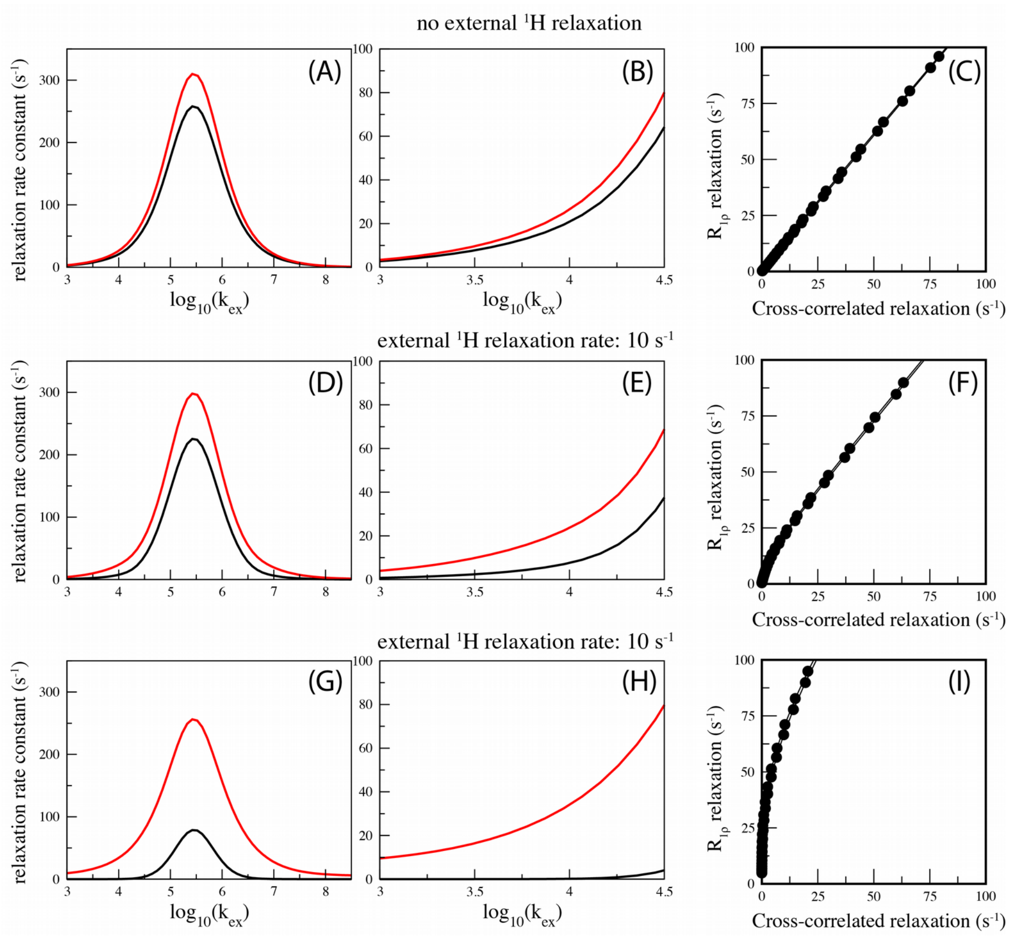

Figure S3. The effect of proton relaxation on ^{15}N transverse relaxation parameters, from numerical simulations with an *ad hoc* added random field fluctuation leading to proton relaxation. The simulations are akin to those shown in Figure 4, with a larger amplitude of motion, i.e. a jump angle of 20 degrees.

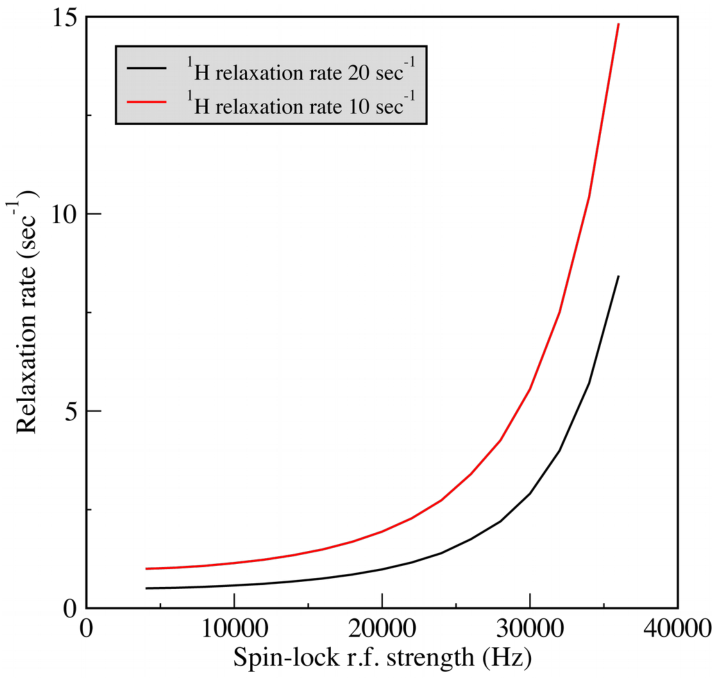

Figure S4. The increase of $R_{1\rho}$ depends on the RF field strength and the MAS frequency. Shown are numerical simulations of $R_{1\rho}$ at a MAS frequency of 40 kHz. The jump angle was set to zero (no dynamics).

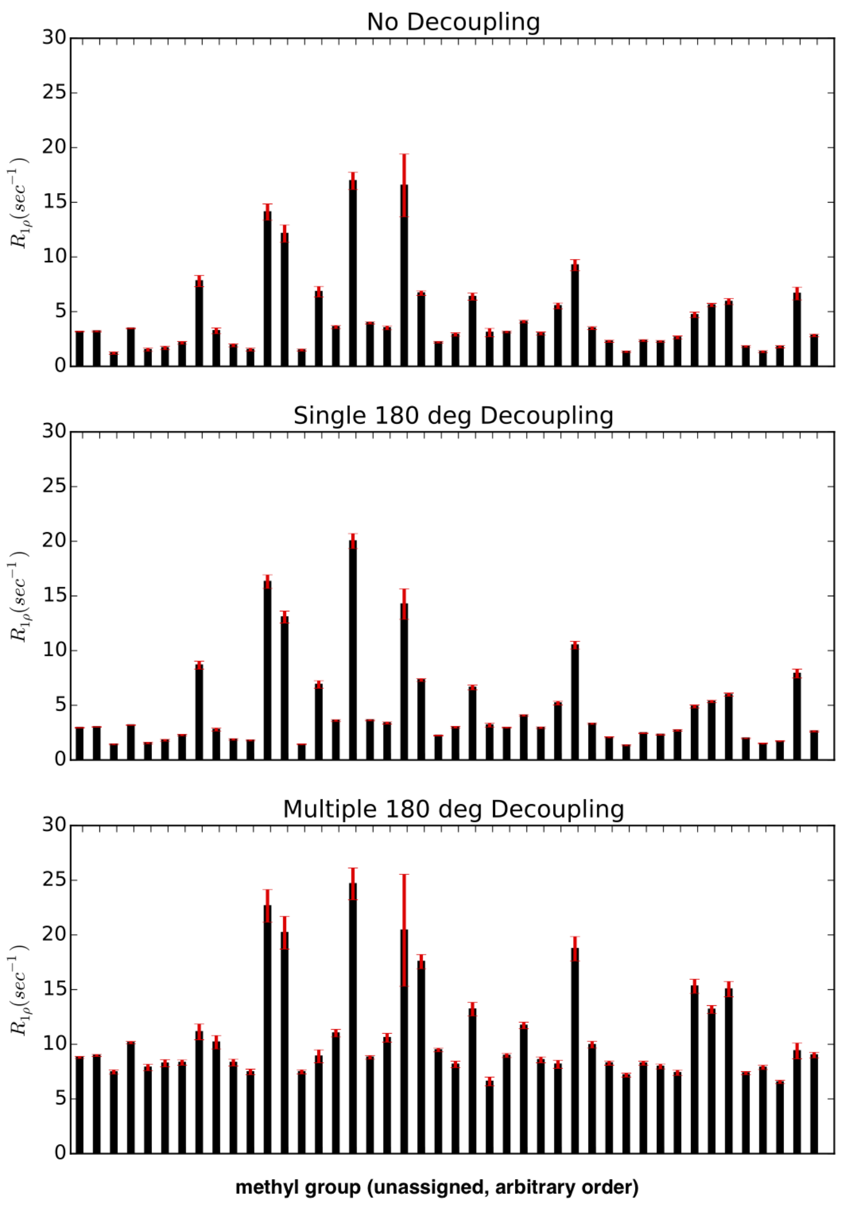

Figure S5. Experimental investigation of multi-pulse decoupling, shown for ^{13}C relaxation in CHD_2 methyl groups in the protein TET2.[1] ^{13}C $R_{1\rho}$ relaxation was measured in a deuterated sample labelled with CHD_2 groups in an otherwise deuterated background. The MAS frequency was set to 55 kHz, and the spin-lock RF field set to 15 kHz. No ^2H decoupling was used in these measurements. The three panels show the case of no ^1H decoupling during the ^{13}C spin-lock, a single ^1H π pulse and a train of ^1H π pulses, applied every 5 ms. No rotor synchronization was attempted for these latter decoupling pulses. Remarkably, even though the delay between pulses exceeds by far the rotor period, slow recoupling due to the pulse train is manifest as enhanced apparent relaxation rate constants.

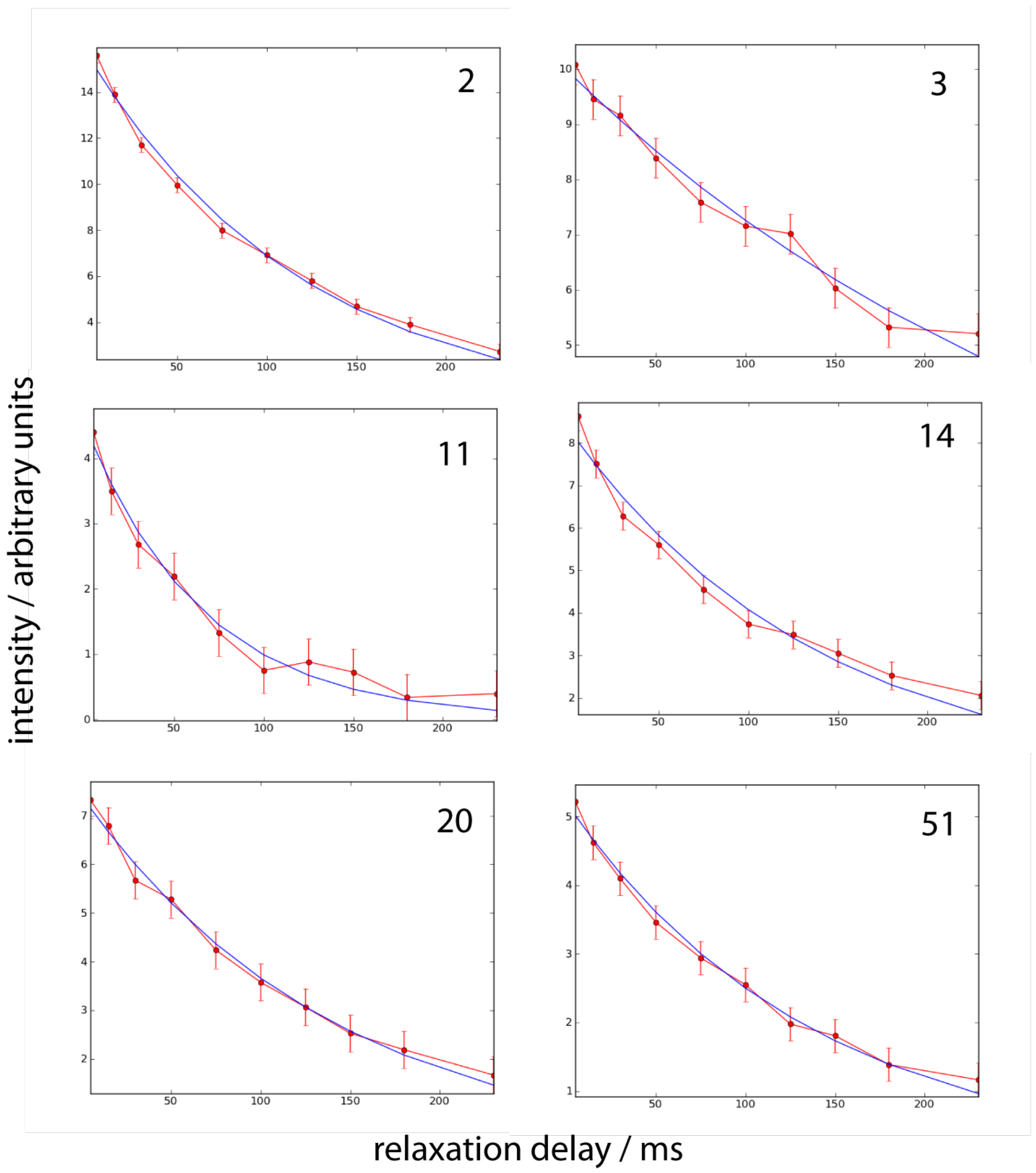

Figure S6. Examples of ^{15}N $R_{1\rho}$ decay curves in microcrystalline ubiquitin in the absence of any ^{1}H irradiation (i.e. without suppression of CSA/D CCR). Residue numbers are shown in each panel. The experiment was performed at 39.5 kHz MAS, and 12 kHz spin-lock RF field strength, using a 50% back-exchanged, deuterated ubiquitin sample.[2]

Additional discussion of the effect of ^{1}H spin flips on $R_{1\rho}$ rate constants

The simulations in Figure 4 have revealed that the proton spin flips lead to an increase in the observed $R_{1\rho}$ rate constants. This increase of $R_{1\rho}$ can be understood even without invoking dynamics, i.e. also for a static HN spin pair, and is described as follows. Consider the evolution of in-phase N_x coherence for a given ^{1}H-^{15}N spin pair in the presence of a spin-lock RF field, and assume that the RF field amplitude matches the MAS frequency, i.e. $\omega_{RF}= \omega_{MAS}$, rotary-resonance. In this situation the heteronuclear dipolar coupling is reintroduced, such that the initial coherence N_x rapidly interconverts to proton coherence and is periodically refocused. When considering a single crystallite, the coherence is fully refocused, on a time scale that depends on the orientation of the crystallite. For a powder, different crystallites oscillate at different frequencies, and the net oscillation is damped, resulting in a loss of the initial N_x coherence. This particular case, choosing the rotary-resonance for the spin-lock, is of course not of practical relevance, because this rapid decay due to the coherence evolution makes extraction of dynamic information impossible.

However, if the spin-lock RF field is chosen at a frequency that differs from the rotary resonance condition (i.e. $\nu_{RF}\neq n.\nu_{MAS}$), the described oscillation changes in its amplitude and frequency, but the situation is qualitatively similar: the main difference is that the loss of initial N_x coherence is reduced to only a small part. If the RF field and MAS frequencies are sufficiently separated, the coherent decay may in fact be negligibly small. The remaining part of the N_x coherence undergoes during each rotor period small evolution under the H-N dipolar coupling (and CSA), which is continuously refocused by MAS. Consequently, no net decay occurs. Now consider the possibility that the proton stochastically changes its spin state. As the proton spin flips, the dipolar coupling is inverted. As a consequence, the dipolar-coupling evolution, which was otherwise refocused under MAS, is not refocused any more. For a given spin pair at a given orientation with respect to the rotor-fixed frame this failure of refocusing translates to a sudden change of the mean N_x intensity, i.e. rather than oscillating in the vicinity of $<N_x>=1$, the oscillation is at a different level. In principle, subsequent proton spin flips may restore the oscillation level to a similar level as the initial starting point. However, when considering a whole ensemble and powder average of spin pairs, the failure of refocusing the dipolar evolution, brought about by the sudden jumps of the proton spin state, will inevitably lead to a net decrease of N_x, i.e. a process that appears as "relaxation", even though strictly speaking it is not ^{15}N relaxation, but an indirect effect of stochastic fluctuations of the proton spin state. This "excess $R_{1\rho}$ decay" increases the observed $R_{1\rho}$ rate constants, as shown by the numerical simulations in Figure 4 (most easily seen in the inserts).

In principle, a second possible reason for the increased $R_{1\rho}$ decay may come into play if the spin flips are fast enough such that the flipping proton constitutes itself a fluctuating magnetic field, which relaxes the ^{15}N spin. (One may consider this as similar to scalar relaxation of the second kind.) For this latter mechanism to be active, the proton spin flips would need to be of the order of thousands per second, a scenario that seems unlikely at least in the samples used here (see below).

Figure 4 and Figure S3 also reveal that in cases where $R_{1\rho}$ is very large, the inclusion of the proton flips may actually reduce the effective ^{15}N $R_{1\rho}$ relaxation rate constant. This situation appears only in cases of $R_{1\rho}$ that are at the maximum of the observed rate constants, i.e. for microsecond motions. A simple simulation of the exchange between two states, N_xH^α and N_xH^β, taking into account auto-relaxation, cross-correlated relaxation and an exchange between these two states, illustrates that such an apparent decrease of the average relaxation rate constant of the two states can be explained:

Consider the most simplistic approach for relaxation in the system of the two states, $N_x[H^\alpha]$ and $N_x[H^\beta]$. Consider that the two states relaxes with $R_{1\rho}+\Gamma_{1\rho}^{CSA/D}$ and $R_{1\rho}-\Gamma_{1\rho}^{CSA/D}$, respectively, and that the two states are interconverted through a proton spin flip, that occurs at a rate *k*. Relaxation in this system is described with the relaxation matrix:

$$\Gamma=\begin{bmatrix} R_{1\rho}+\Gamma_{1\rho}^{CSA/D}-k & k \\ k & R_{1\rho}-\Gamma_{1\rho}^{CSA/D}-k \end{bmatrix} . \qquad (1)$$

This equation shows that the evolution of the two doublet components are coupled. The $R_{1\rho}$ decay of the average of the two doublet components, i.e. the in-phase N_x coherence decay with time *t*, is then described by $I(t)=e^{(k-R_{1\rho})t}.\cosh\left(\sqrt{\left(\left(\Gamma_{1\rho}^{CSA/D}\right)^2+k^2\right)}\right)t$ (2)

(as derived with Mathematica). Figure S7 shows plausible cases, using this simplistic approach, in which the introduction of a proton flip rate, *k*, leads to an apparent longer life time of the N_x coherence.

The simulations in Figure S3 suggest that this apparent decrease of $R_{1\rho}$ occurs only in the range where the rate constants are largest, and in practice it appears that the increase of $R_{1\rho}$ through the mechanism described above dominates. Investigating these effects further is deferred to future work.

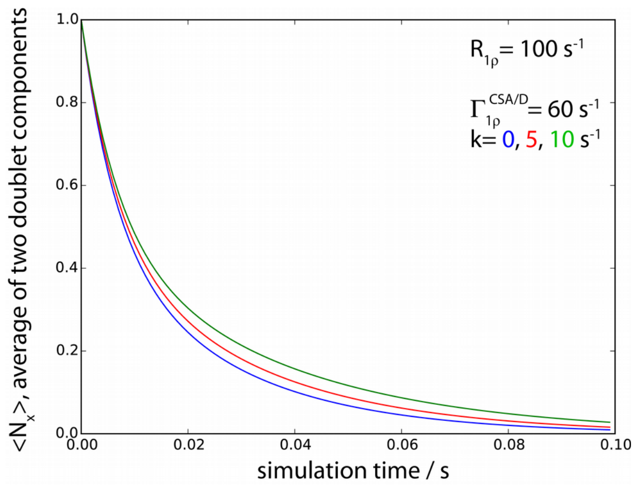

Figure S7. Simulation of the evolution of the average of the two doublet components, using the simple approach described with Equations (1) and (2) above. Three different proton flip rate constants, *k*, are shown.

Numerical spin simulation source code

The following program, to be used with the GAMMA package[3], simulates the evolution of ^{15}N coherence (either $2N_xH\alpha$ or $2N_xH_\beta$ or N_x, to be chosen using a flag) under a ^{15}N spin-lock, assuming an explicit two-state jump. The program requires a spin-system input file (shown further below), which states the number and identity of the spins in the two exchanging states.

The source code below must be saved in a .cc file (e.g. "R1rho_exchange_DMPS.cc") and compiled with GAMMA (http://scion.duhs.duke.edu/vespa/gamma/wiki). Typically the program is executed with an input file, which contains all the relevant parameters, such as the interaction strengths, the number of powder points, the spin-lock RF field strength and the available flags and options. The latter are described within the source code. An example input file is also given below.

The program outputs a MATLAB format file, which contains an array of time points and an array containing the expectation value of the relevant operator ($2N_xH\alpha$ or $2N_xH_\beta$ or N_x). Fitting such a decay curve, using an external program, allows extracting a relaxation rate constant. In this study we used scipy (python) modules to import the MATLAB files, and numpy routines for fitting.

Source code:

```
/*
 R1rho simulation program with explicit exchange between two exchanging spin systems.
 RF spin-lock on 15N and 0, 1, N 180º pulse on 1H to suppress CDA/DD cross-correlation
 Evolution computed on a single crystallite or averaged over a powder
 Detection of the 15N in-phase coherence (Nx) or of the individual lines (NxHa or NxHb)
(C) Dominique Marion, Emmanuelle Weber, Paul Schanda IBS-Grenoble 2015-2016
 Adapted from a CPMG-simulation code written by Matthias Ernst (ETH Zurich), as used in Tollinger
 et al, JACS 2012.
 If you use this code or modified code based on this we kindly ask to cite
 Kurauskas et al., J. Phys. Chem. B, 2016 (i.e., this publication)
*/

#include "gamma.h"
#include <sys/time.h>
#include <sys/resource.h>

#define MAXSPINS 8

using namespace std;

complex powl(const complex& z, int m)
{ if(norm(z) == 0)
   return complex(0);
  else
   return complex(exp(m * log(z)));
}

super_op powl(const super_op& LOp, int power)
{ super_op LOp1;
  complex z;
  LOp.set_EBR();
  LOp1=LOp;
  for(int i=0; i<LOp.LS(); i++)
  { z = LOp.get(i,i);
   LOp1.put(i,i, powl(z, power));
  }
  return LOp1;
 }

int main(int argc, char *argv[])

{
  sys_dynamic axc[2];
  multi_sys ax;
  gen_op temp, zz, Upp[4], Ham, H[2][5], sigmax, sigma1, sigma;

  gen_op detect, detectSxIa, detectSxIb, Hrf;
  super_op Iop,K,L,U,UT,Un,Un1,LO;
  space_T Adip[2][MAXSPINS][MAXSPINS], Adip_R[2][MAXSPINS][MAXSPINS];
  space_T Acsa[2][MAXSPINS], Acsa_R[2][MAXSPINS];
  double J[2][MAXSPINS][MAXSPINS], D[2][MAXSPINS][MAXSPINS];
  double iso_CSA[2][MAXSPINS], eta_CSA[2][MAXSPINS], delta_CSA[2][MAXSPINS];
  int i,j,k,m,n,Fnp,count,qu,steps;
  int coherence, decoupling, orientation, val1, val2;
  string name, names;
  const double thetam=54.73561032;
  double mas_freq;
  double ltime, time, scale;
  int nspins[2];
  double alpha,beta,gamma;
  double alpha_CSA[2][MAXSPINS], beta_CSA[2][MAXSPINS];
  double gamma_CSA[2][MAXSPINS];
  double alpha_D[2][MAXSPINS][MAXSPINS], beta_D[2][MAXSPINS][MAXSPINS];
  double gamma_D[2][MAXSPINS][MAXSPINS];
  double kex,k_ex,kd;
  double Ttotal;                                                    /* Total relax delay */

  struct rusage me;
  int sampling_taur;

  double gamB1;

/*
Investigations of a nonrandom numerical method for multidimensional integration
Vera B. Cheng, Henry H. Suzukawa Jr., and Max Wolfsberg
The Journal of Chemical Physics 59, 3992 (1973); doi: 10.1063/1.1680590
*/

  int value1[] = {1, 50, 100, 144, 200, 300, 538, 1154};
  int value2[] = {1,  7,  27,  11,  29,  37,  55,  107};
  int value3[] = {1, 11,  41,  53,  79,  61, 229,  271};

  count = 1;
  query_parameter(argc,argv,count++,"Spin System Name          ? ", names);
/*      Read a Gamma spin-system                                  */
  ax.read(names);
  axc[0] = ax.Comp(0);
  axc[1] = ax.Comp(1);

  nspins[0]=axc[0].spins();
  nspins[1]=axc[1].spins();
  for(k=0;k<2;++k)
  { for(i=0;i<nspins[k]-1;++i)
    { for(j=i+1;j<nspins[k];++j)
/*      J-coupling and dipolar coupling between 1H and 15N                  */
```

```
    { query_parameter(argc,argv,count++,"J Coupling Constant        ? ", J[k][i][j]);
      query_parameter(argc,argv,count++,"Dipolar Coupling Constant   ? ", D[k][i][j]);
      query_parameter(argc,argv,count++,"Euler angle alpha          ? ", alpha_D[k][i][j]);
      query_parameter(argc,argv,count++,"Euler angle beta           ? ", beta_D[k][i][j]);
      query_parameter(argc,argv,count++,"Euler angle gamma          ? ", gamma_D[k][i][j]);
    }
   }
  }
  for(k=0;k<2;++k)
  { for(i=0;i<nspins[k];++i)
/*      Isotropic chemical shift and CSA for 1H (spin 0) and 15N (spin 1)            */
   { query_parameter(argc,argv,count++,"isotropic chemical shift   ? ", iso_CSA[k][i]);
     query_parameter(argc,argv,count++,"CSA tensor (delta CSA)     ? ", delta_CSA[k][i]);
     query_parameter(argc,argv,count++,"CSA tensor (eta CSA)       ? ", eta_CSA[k][i]);
     query_parameter(argc,argv,count++,"Euler angle alpha          ? ", alpha_CSA[k][i]);
     query_parameter(argc,argv,count++,"Euler angle beta           ? ", beta_CSA[k][i]);
     query_parameter(argc,argv,count++,"Euler angle gamma          ? ", gamma_CSA[k][i]);
   }
  }
  query_parameter(argc,argv,count++,"Powder Quality (cheng)     ? ", qu);
  query_parameter(argc,argv,count++,"Number of time steps       ? ", steps);
  query_parameter(argc,argv,count++,"spinning speed             ? ", mas_freq);
/*      We only sample the spin-system after multiple rotations of the MAS rotor       */
  query_parameter(argc,argv,count++,"Rate of sampling, in units of tau_r ?", sampling_taur);
  query_parameter(argc,argv,count++,"Total relaxation delay     ? ", Ttotal);
/*      Random field relaxation acting on 1H                                 */
  query_parameter(argc,argv,count++,"Random field constant      ? ", kd);
/*      Kinetic exchange rate constant between the two states                         */
  query_parameter(argc,argv,count++,"Jump rate constant         ? ", k_ex);
  query_parameter(argc,argv,count++,"Output Filename            ? ", name);
  query_parameter(argc,argv,count++,"RF field strength          ? ", gamB1);
/*      0 = NxHa   1 =  NxHb    2 =  Nx                              */
  query_parameter(argc,argv,count++,"Det. coherence (Na Nb Nx)   ? ", coherence);
/*      0 (no 180º pulse)  1 (single 180º pulse)  2 (180º pulse train)          */
  query_parameter(argc,argv,count++,"NoDec Dec MultiDec         ? ", decoupling);
/*      0 (powder = average over all orientation) >0 single orientation               */
  query_parameter(argc,argv,count++,"Orientation                ? ", orientation);

  time      = (1.0/mas_freq)/steps;
  kex       = 1.0/2.0*k_ex;
/* kex = kforward+kbackward */
  Fnp = Ttotal * mas_freq / (sampling_taur * 2);         // data points to be sampled. ; factor /2 comes
from splitting the spinlock in two halves.
/*
            1/mas_freq
        |-----------------|  x sampling_taur * 2
                                                ^
            step                                echo
*/


  cout << "\nR1rho simulation under MAS with stochastic motional processes for NH       \n";
  cout << "==========================================================\n\n";
  cout << "Program version: " << __FILE__ << " compiled at " << __DATE__ ", "
      << __TIME__ << "\n\n";
  cout << "Parameters:\n";
  cout << "rotation angle thetam         : " << thetam << " Degree\n";
  cout << "size of spin system           : " << nspins[0]+nspins[1] << " spins\n";
  for(k=0;k<2;++k)
  { for(i=0;i<nspins[k]-1;++i)
```

```
    { for(j=i+1;j<nspins[k];++j)
      { cout << "J       coupling constant (" << i << "," << j << ") : " <<
               J[k][i][j] << " Hz\n";
        cout << "Dipolar coupling constant (" << i << "," << j << ") : " <<
               D[k][i][j] << " Hz\n";
        cout << "Relative orientation of D tensor: (" << alpha_D[k][i][j] << "," <<
                beta_D[k][i][j] << "," << gamma_D[k][i][j] << ")\n";
      }
    }
  }
  for(k=0;k<2;++k)
  { for(i=0;i<nspins[k];++i)
    { cout << "Isotropic chem shift(" << i << ") : " <<
             iso_CSA[k][i] << " Hz\n";
      cout << "Delta of CSA tensor (" << i << ") : " <<
             delta_CSA[k][i] << " Hz\n";
      cout << "Eta of CSA tensor   (" << i << ") : " <<
             eta_CSA[k][i] << "\n";
      cout << "Relative orientation of CSA tensor: (" << alpha_CSA[k][i] << "," <<
              beta_CSA[k][i] << "," << gamma_CSA[k][i] << ")\n";
    }
  }

  cout << "Powder Quality Number:        " << qu << "  (" << value1[qu] << " orientations)\n";
  cout << "Ttotal:                       " << Ttotal << "\n";
  cout << "Maximum relaxation delay:     " << Ttotal << " s\n";
  cout << "Sampling every                " << sampling_taur << "th rotor period\n";
  cout << "Number of data points (Fnp):  " << Fnp << " points\n";
  cout << "MAS frequency:                " << mas_freq << " Hz\n";
  cout << "Time increments:              " << time << "s\n";
  cout << "Exchange rate constant        " << 2.0*kex << " s-1\n";
  cout << "Random field constant         " << kd << " s-1\n";
  cout << "Output filename:              " << name << "\n";
  cout << "RF fields (gamma*B1, in Hz)   " << gamB1 << "\n";

  cout << "\n";
  cout << "Nb of delays = " << Fnp <<"\n";
  cout << "Delays = [0:"<< 2.0*sampling_taur/mas_freq << ":" << 2.0*(Fnp-1)*sampling_taur/mas_freq
<<"];\n";
  cout << "\n";


  cout.flush();

  ax.Kex(kex,0);
  zz = Fx(axc[0]);

  Iop = commutator(multize(zz,ax,0));

  matrix mx(ax.LS(),ax.LS(),i_matrix_type);
  Iop.put_mx(mx);

  block_2D data(2,Fnp);
/*
          The output file contains the computed amplitude (data(0,m)) and
          the corresponding delays (data(0,m)
 */
  for(m=0;m<Fnp;++m)
    {
     data(0,m) = 0;
    data(1,m) = float (m) * 2.0*sampling_taur/mas_freq;
   }


//setup for the space tensor
  matrix help(3,3,0);
  for(k=0;k<2;++k)
  { for(i=0;i<nspins[k]-1;++i)
   { for(j=i+1;j<nspins[k];++j)
     { help.put_h(-1.0/2.0,0,0);
      help.put_h(-1.0/2.0,1,1);
      help.put_h( 1.0,2,2);
      help   = - (complex) D[k][i][j] * help;
      Adip[k][i][j] = A2(help);
      Adip[k][i][j] = Adip[k][i][j].rotate(alpha_D[k][i][j],beta_D[k][i][j],gamma_D[k][i][j]);
     }
   }
  }
  for(k=0;k<2;++k)
  { for(i=0;i<nspins[k];++i)
   { help.put_h(-1.0/2.0*(1.0+eta_CSA[k][i]),0,0);
    help.put_h(-1.0/2.0*(1.0-eta_CSA[k][i]),1,1);
    help.put_h( 1.0,2,2);
    help = (complex) delta_CSA[k][i] * help;
    Acsa[k][i] = A2(help);
    Acsa[k][i] = Acsa[k][i].rotate(alpha_CSA[k][i],beta_CSA[k][i],gamma_CSA[k][i]);
   }
  }

  string name1 = name+".mat";
  string name2 = name;


          /* 1H pi pulse */
          zz=Ixypuls_U(axc[0],"1H",0.0,180.0);
          Upp[0] = multize(zz,ax,0);
          zz=Ixypuls_U(axc[1],"1H",0.0,180.0);
          Upp[0] += multize(zz,ax,1);

   zz = gamB1 * Fx(axc[0],"15N");
   Hrf = multize(zz,ax,0);
   zz = gamB1 * Fx(axc[1],"15N");
   Hrf += multize(zz,ax,1);

/* Random field contribution acting on the 1H spins                         */

   zz = Fz(axc[0],"1H");
   temp = multize(zz,ax,0);
   zz = Fz(axc[1],"1H");
   temp += multize(zz,ax,1);
   L0  = kd*d_commutator(temp);

   zz = Fx(axc[0],"1H");
   temp = multize(zz,ax,0);
   zz = Fx(axc[1],"1H");
   temp += multize(zz,ax,1);
   L0  += kd*d_commutator(temp);

   zz = Fy(axc[0],"1H");
   temp = multize(zz,ax,0);
   zz = Fy(axc[1],"1H");
   temp += multize(zz,ax,1);
   L0  += kd*d_commutator(temp);

if (orientation == 0)
          { val1 = 1;
           val2 = value1[qu];
          }
          else
          { if (orientation > value1[qu])
           {  orientation = value1[qu];
             cout << "Orientation set to " << value1[qu]<< "\n";
           }
           val1 = orientation;
           val2 = orientation;
          }


/* if orientation == 0 the observable is averaged over a powder
  if orientation >0 the observable is computed for a single orientation
*/

for(count=val1; count<=val2; ++count)                          /*          Powder loop       */
  { beta  = 180.0 * count/value1[qu];
   alpha = 360.0 * ((value2[qu]*count) % value1[qu])/value1[qu];
   gamma = 360.0 * ((value3[qu]*count) % value1[qu])/value1[qu];
   if ((count % 10 == 1) || (orientation >0))
   { getrusage(0, & me);
    cout << count << "\tbeta = " << beta << "\talpha = "
        << alpha << "\tgamma = " << gamma
        << ",\ttime used: " << me.ru_utime.tv_sec << " seconds\n";
    cout.flush();
   }
   scale = sin(beta/180.0*PI);

/*      Initial state                                        */

   sigmax = Fx(axc[0],"15N");
   sigma1 = multize(sigmax,ax,0)*ax.pop(0);
   sigmax = Fx(axc[0],"15N");
   sigma1 += multize(sigmax,ax,1)*ax.pop(1);

/*      Computing the detection operators                          */

   zz    = Fm(axc[0],"15N");
   detect = multize(zz, ax, 0);
   zz    = Ix(axc[0],1)*Ia(axc[0],0);
   detectSxIa = multize(zz, ax, 0);
   zz    = Ix(axc[0],1)*Ib(axc[0],0);
   detectSxIb = multize(zz, ax ,0);

/*      Rotation of the space tensor                         */

   for(k=0;k<2;++k)
   { for(i=0;i<nspins[k]-1;++i)
    { for(j=i+1;j<nspins[k];++j)
      { Adip_R[k][i][j] = Adip[k][i][j].rotate(alpha,beta,gamma);
      }
    }
   }
   for(k=0;k<2;++k)
   { for(i=0;i<nspins[k];++i)
    { Acsa_R[k][i] = Acsa[k][i].rotate(alpha,beta,gamma);
    }
   }

/*      Zero all components                                          */
   for(k=0;k<2;++k)
    for(i=0;i<5;++i)
     H[k][i] = gen_op();

/*      Computation of the Hamiltonian ....                               */
   for(k=0;k<2;++k)
   {  for(j=-2;j<=2;++j)
/*      ... Dipolar interaction and J-coupling                            */

    { for(m=0;m<nspins[k]-1;++m)
     { for(n=m+1;n<nspins[k];++n)
      { if(axc[k].isotope(m) != axc[k].isotope(n))
       { H[k][j+2] += Adip_R[k][m][n].component(2,j) * d2(j,0,thetam) *
1/sqrt(6.0)*2*Iz(axc[k],m)*Iz(axc[k],n);
        if(j==0)
        { H[k][j+2] += J[k][m][n]*Iz(axc[k],m)*Iz(axc[k],n);
        }
       }
       else
       { H[k][j+2] += Adip_R[k][m][n].component(2,j) * d2(j,0,thetam) * 1/sqrt(6.0)*
             (2*Iz(axc[k],m)*Iz(axc[k],n)-(Ix(axc[k],m)*Ix(axc[k],n)
+Iy(axc[k],m)*Iy(axc[k],n)));
        if(j==0)
        { H[k][j+2] += J[k][m][n]*(Iz(axc[k],m)*Iz(axc[k],n)+Ix(axc[k],m)*Ix(axc[k],n)
+Iy(axc[k],m)*Iy(axc[k],n));
        }
       }
      }
     }

/*      ... isotropic chemical shift and CSA                              */

     for(m=0;m<nspins[k];++m)
     { H[k][j+2] += Acsa_R[k][m].component(2,j) * d2(j,0,thetam) * 1/sqrt(6.0)*2*Iz(axc[k],m);
      if(j==0)
      { H[k][j+2] += iso_CSA[k][m]*Iz(axc[k],m);
      }
     }
    }
   }

   U = Iop;  // Identity Operator

/*     Calculation of the propagator over one cycle of the MAS                */

   for(ltime=0.5*time;ltime<=steps*time;ltime += time)
   {
    Ham = Hrf;
    for(k=0;k<2;++k)
               { for(i=-2;i<=2;++i)
               Ham += exp(complex(0,i*2.0*PI*ltime*mas_freq)) * multize(H[k][i+2],ax,k);
               }
    L = complex(0,2.0*PI)*commutator(Ham);
```

```
      if ( kd != 0.0 )
                    { L = L + L0;
                    }
      K = Xnm(ax);
      L += K;
      UT = exp(L,-time);
      UT.set_HBR();
      U = UT*U;
    }

/*    Places the superoperator into its original Hilbert space basis.               */
    U.set_HBR();
    sigma1.set_DBR();
    sigma=sigma1;
/*    Makes a longer propagator, in order to sample less often than every taur       */
    Un=powl(U,sampling_taur);
/*    Places the superoperator into its original Hilbert space basis.               */
         Un.set_HBR();
    Un1=Un;

/*    Loop over the complete spin-lock period
              */

    for(i=0;i<Fnp;++i)
    {
      if (coherence == 0)
      data(0,i) += proj(sigma,detectSxIa)*scale;
      if (coherence == 1)
      data(0,i) += proj(sigma,detectSxIb)*scale;
      if (coherence == 2)
      data(0,i) += proj(sigma,detect)*scale;
      if (decoupling == 0)
/*    No 1H Decoupling pulse                                      */
/*         |          Un          |          Un          | (Fnp)          */
      {
        sigma=Un*sigma;
        sigma=Un*sigma;
      }
      if (decoupling == 1)
/*    Single 1H Decoupling pulse                                    */
/*         |         Un1        180º       Un1          | (Fnp)         */
/*         with       Un1=Un*Un*Un*Un                               */
      {
/* set sigma back to initial value, before applying the whole sequence              */
        sigma=sigma1;
        sigma=Un1*sigma;
             sigma.sim_trans_ip(Upp[0]);          // 1H pi pulse
             sigma=Un1*sigma;
             Un1=Un*Un1;
      }
      if (decoupling == 2)
/*    Multiple 1H Decoupling pulse                                    */
/*         |        Un1         180º       Un1          | (Fnp)         */
/*         with       Un1=Un                              */
           {
             sigma=Un1*sigma;
        sigma.sim_trans_ip(Upp[0]);    // 1H pi pulse
        sigma=Un1*sigma;
      }
```

```
   }

  }       /*   end of powder loop          */



  MATLAB(name1,name2,data,1);
  cout << "1st point  " << data(0,0) << data(0,1) << data(0,2) << "\n";
  cout << "Delays     " << data(1,0)*1e6 << data(1,1)*1e6 << data(1,2)*1e6 << "\n";
  cout.flush();
  exit(0);
}
```

<u>Spin system file:</u>

The following file is a spin system file, which is required and called by the GAMMA program. Copy this into a file, e.g. “2spin_50perc.sys”, and use this name to the call “names” in the GAMMA program. The values of chemical shift, J coupling and Larmor frequency are all irrelevant here, because these values are called explicitly by the GAMMA program.

```
MSysName     (2) : Two-Spin Exchange     - Name of overall spin system
NComp        (0) : 2          - Number of Components in the System
Popul(0)     (1) : 0.5        - Population of Component 0
Popul(1)     (1) : 0.5        - Population of Component 1
Exch(0)      (2) : (0<=>1)    - Non-Mutual Exchange Scheme
Smap(0,0)    (2) : (0)0(1)0   - Map (Cmp 0 0th spin) to (Cmp 1 0th spin)
Smap(0,1)    (2) : (0)1(1)1   - Map (Cmp 0 1st spin) to (Cmp 1 1st spin)
Kex_nm(0)    (1) : 0.0        - Non-Mutual Exchange Rate (1/Sec)

[0]SysName   (2) : NHa        - Name of component 1
[0]NSpins    (0) : 2          - Number of Spins in the System
[0]Iso(0)    (2) : 1H         - Spin Isotope Type
[0]Iso(1)    (2) : 15N        - Spin Isotope Type
[0]v(0)      (1) :     0.0   - Chemical Shifts in Hz – irrelevant
[0]v(1)      (1) :     0.0   - Chemical Shifts in Hz – irrelevant
[0]J(0,1)    (1) :     0.0   - J coupling in Hz – irrelevant
[0]Omega     (1) :   300.0   - Spectrometer Frequency in MHz (1H based) – irrelevant

[1]SysName   (2) : NHb        - Name of component 1
[1]NSpins    (0) : 2          - Number of Spins in the System
[1]Iso(0)    (2) : 1H         - Spin Isotope Type
[1]Iso(1)    (2) : 15N        - Spin Isotope Type
[1]v(0)      (1) :     0.0   - Chemical Shifts in Hz – irrelevant
[1]v(1)      (1) :     0.0   - Chemical Shifts in Hz – irrelevant
[1]J(0,1)    (1) :     0.0   - J coupling in Hz – irrelevant
[1]Omega     (1) :   300.0   - Spectrometer Frequency in MHz (1H based) – irrelevant
```

<u>Example input file:</u>

The following example input file detects the evolution of N_x, uses one ^{1}H π pulse and averages over a powder (100 orientations). The ^{15}N CSA and the ^{1}H-^{15}N dipolar coupling span a 20 degree angle, and both jump by 5 degrees between the two orientations. The spins are on-resonance (no isotropic chemical shift) in both exchanging states. The ^{1}H chemical-shift tensor is not altered upon exchange. It is assumed that the executable GAMMA program is called R1rho_exchange_DMPS, and the spin system file “2spin_50perc.sys”.

```
set filename = kex_25000_angle_5_pop_50_RF_15000_600
./R1rho_exchange_DMPS 2spin_50perc.sys\
        90 21228 0 103 0\
        90 21228 0 103 5 \
        0 480 0 0 0 0\
        0 6300 0 0 83 0\
        0 480 0 0 0 0\
        0 6300 0 0 83 5 \
2 100 40000 1 0.12 5  25000  ${filename}\
15000 2 1 0
```

References

(1) Amero, C.; Schanda, P.; Asuncion Dura, M.; Ayala, I.; Marion, D.; Franzetti, B.; Brutscher, B.; Boisbouvier, J. Fast Two-Dimensional NMR Spectroscopy of High Molecular Weight Protein Assemblies. *J. Am. Chem. Soc.* **2009**, *131*, 3448–3449.

(2) Haller, J. D.; Schanda, P. Amplitudes and Time Scales of Picosecond-to-Microsecond Motion in Proteins Studied by Solid-State NMR: A Critical Evaluation of Experimental Approaches and Application to Crystalline Ubiquitin. *J. Biomol. NMR* **2013**, *57*, 263–280.

(3) Smith, S.; Levante, T.; Meier, B.; Ernst, R. Computer Simulations in Magnetic Resonance. An Object-Oriented Programming Approach. *J. Magn. Reson.,* **1994**, *106*, 75–105.