Supplementary Information for:

Characterization of Fibril Dynamics on Three Timescales by Solid-State NMR

Albert A. Smith, Emilie Testori, Riccardo Cadalbert, Beat H. Meier,* Matthias Ernst*

ETH Zürich, Physical Chemistry, Vladimir-Prelog-Weg 2, 8093 Zürich, Switzerland

B.M.: beme@ethz.ch

M.E.: maer@ethz.ch

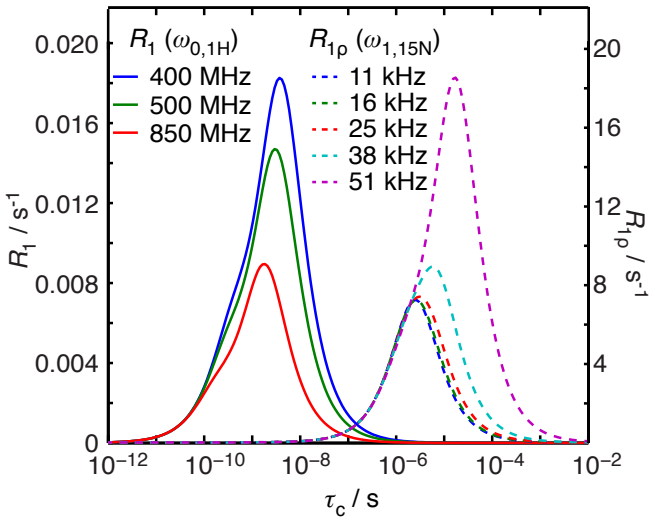
Definition of symbols

S^2 :	Order parameter of the motion. Usually denotes the total order parameter
S_s^2, S_i^2, S_f^2	Order parameters for slow, intermediate, and fast timescale motion
τ_c :	Correlation time of the motion (s)
τ_s, τ_i, τ_f	Correlation times of slow, intermediate and fast motions (s)
δ^{IS} :	Dipole coupling anisotropy (rad/s)
$\omega_I \sigma_{zz}$:	Chemical shift anisotropy (rad/s)
ω_I, ω_S :	Larmor frequency (rad/s)
ω_1	Field strength of spin locking field (rad/s)
ω_r	Rotor frequency (rad/s)
R_1	Total spin-lattice relaxation rate (s^{-1})
R_1^{IS} :	Spin-lattice relaxation induced by dipole coupling (s^{-1})
R_1^{CSA} :	Spin-lattice relaxation induced by the CSA (s^{-1})
$R_{1\rho}$	Total transverse relaxation rate under a spin-lock (s^{-1})
$R_{1\rho}^{\mathrm{IS}}$:	Transverse relaxation under a spin-lock induced by dipole coupling (s^{-1})
$R_{1\rho}^{\mathrm{CSA}}$:	Transverse relaxation under a spin-lock induced by the CSA (s^{-1})

Introduction

Sensitivity Range of R_1 and $R_{1\rho}$ measurements

SI Figure 1 shows the range where R_1 and $R_{1\rho}$ are sensitive to different timescales. However, note that the sensitive range can be strongly affected depending on what motions are present and the amplitudes of motion- for example, if all motions have $\tau_c < 10^{-7}$ s, then $R_{1\rho}$ is determined by faster motions, and further, a large amplitude motion in a less sensitive range may overwhelm a slower motion.



SI Figure 1: Calculation of R_1 (solid lines, left axis) and $R_{1\rho}$ (dotted lines, right axis) for S^2 = 0.977 as a function of correlation time assuming a CSA of Δσ=150 ppm, and a dipole coupling of δ=22.9 kHz (^{15}N values). All $R_{1\rho}$ values were calculated with $\omega_{0,1H}$ = 850 MHz. R_1 and $R_{1\rho}$ were calculated using equations (4) and (7) in the main text.

1 Theory and practical aspects of relaxation measurements

1.1 S^2 for different models

S^2 is related to the amplitude of motion, where larger amplitude motions cause S^2 to become smaller and (S^2 = 1) when no motion is present. S^2 can be calculated when a motional model is defined. Here, we give S^2 for the two-site hop and wobbling in a cone.

Two-site hop: The two-site hop (Wittebort and Szabo 1978) is a stochastic re-orientation of an anisotropic interaction between two orientations. This is defined by the angle, θ, between the principle axis of the interaction before and after the hop (assuming the asymmetry of the interaction is 0).

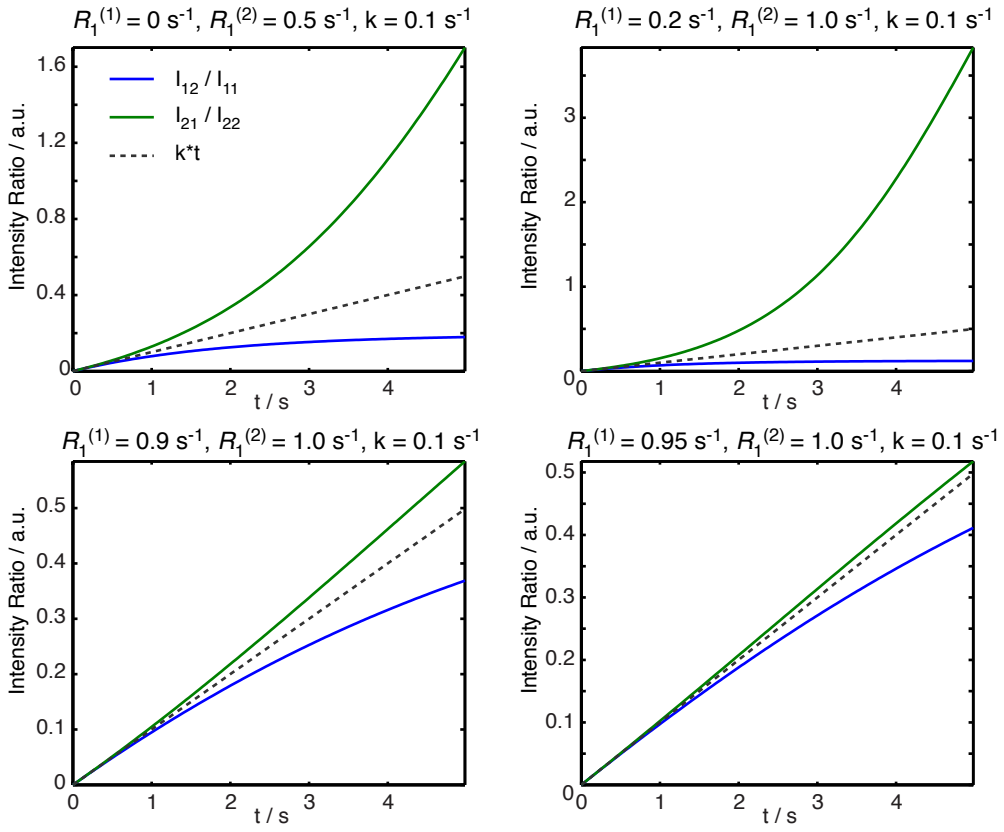
$$S^2 = \frac{1}{4}\left(3\cos^2(\theta)+1\right) \qquad (1)$$

Restricted diffusion in a cone: The restricted diffusion in a cone (wobbling in a cone) model assumes the orientation of an anisotropic motion moves stochastically, but is bounded by a cone (London and Avitabile 1978; Wittebort and Szabo 1978). This is again defined with an angle, θ, which is the opening angle of the cone (angle between the center of the cone and the outer edge).

$$S = \frac{1}{2}\cos(\theta)(1+\cos\theta) \tag{2}$$

1.2 ^{13}C–^{13}C Transfer Rates Bounded by Cross Peak Intensity

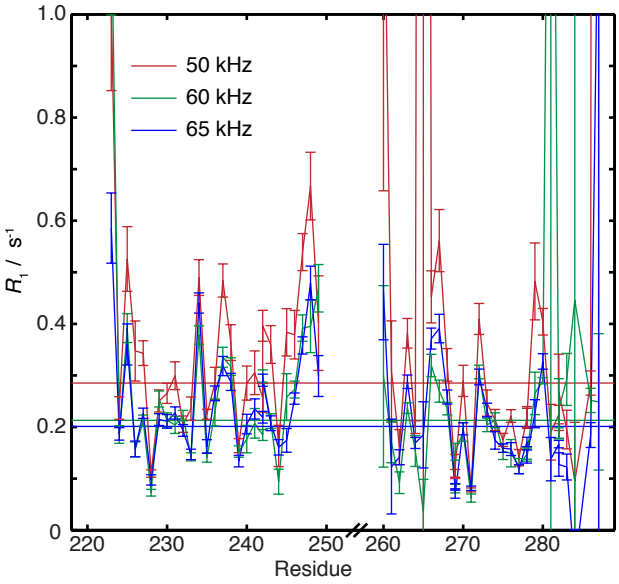
Using text equation (10) for magnetization exchange between two spins, we can calculate the cross-peak intensity for a cross-peak which results from magnetization starting on spin 1 and ending on the spin 2 (I_{12}) and compare it to the diagonal peak for spin 1 (I_{11}, starts on spin 1 and remains there). Similarly, we can compare I_{21} to I_{22}. When taking the ratios, we find if $R_1^{(1)} < R_1^{(2)}$, then $I_{12}/I_{11} < k*\tau < I_{21}/I_{22}$, which can be seen in SI Figure 2. Note that this relationship will fail when $R_1^{(1)} \approx R_1^{(2)}$.



SI Figure 2: Ratios of I_{12}/I_{11} (blue) and I_{21}/I_{22} (green) for various conditions. We also plot $k\tau$ (black, dashed), to demonstrate that $I_{12}/I_{11} < k\tau < I_{21}/I_{22}$ in each case. Note that if R_1^I and R_1^S are nearly equal, this will no longer be the case.

1.3 MAS Dependence of ^{13}Cα R_1

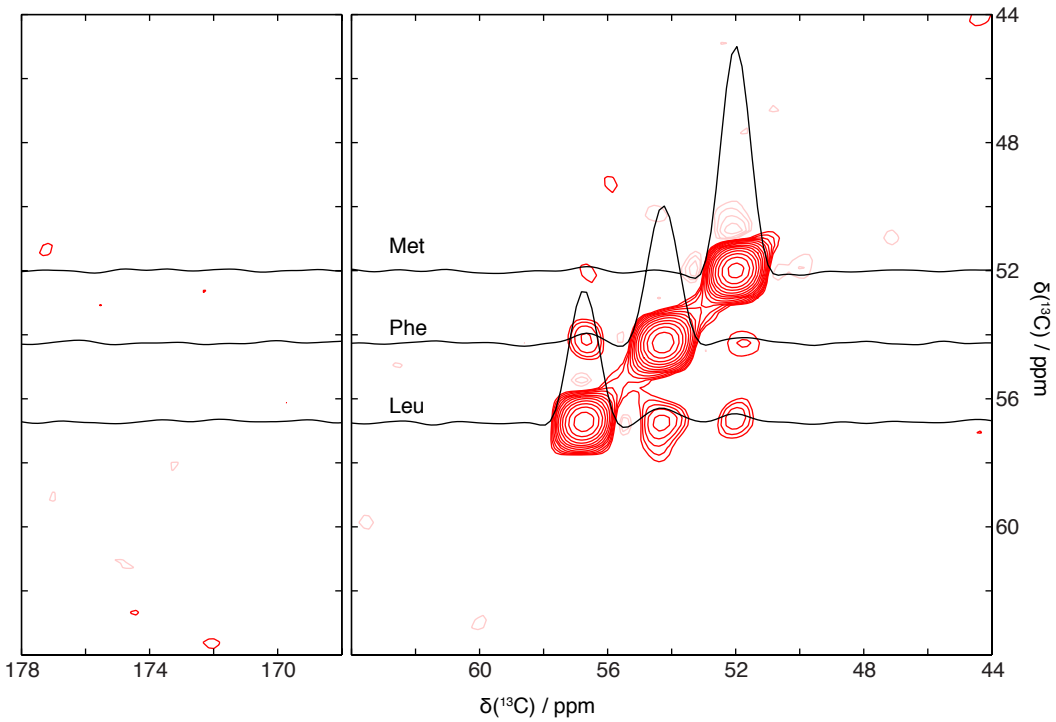
To verify that our measurement of ^{13}Cα R_1 is not dominated by spin-diffusion of ^{13}Cα magnetization to neighboring spins, we measure R_1 as a function of spinning frequency. We find that whereas the R_1 drops significantly between 50 kHz and 60 kHz (median values of 0.29 s^{-1} and 0.21 s^{-1}, respectively), the change between 60 and 65 kHz is nearly negligible (median value of 0.20 s^{-1} at 65 kHz). This indicates that significant contributions to spin-diffusion are removed between 50 kHz and 60 kHz MAS, whereas improvement with additional increases in the MAS is more limited.



SI Figure 3: Site specific ^{13}Cα R_1 measurements at 850 MHz. We plot R_1 measured at MAS frequencies of 50 kHz (red), 60 kHz (green), and 65 kHz (blue). Error bars indicate one standard deviation (68% confidence). Lines indicate the median value of each data set.

1.4 ^{13}C–^{13}C Transfer in MLF

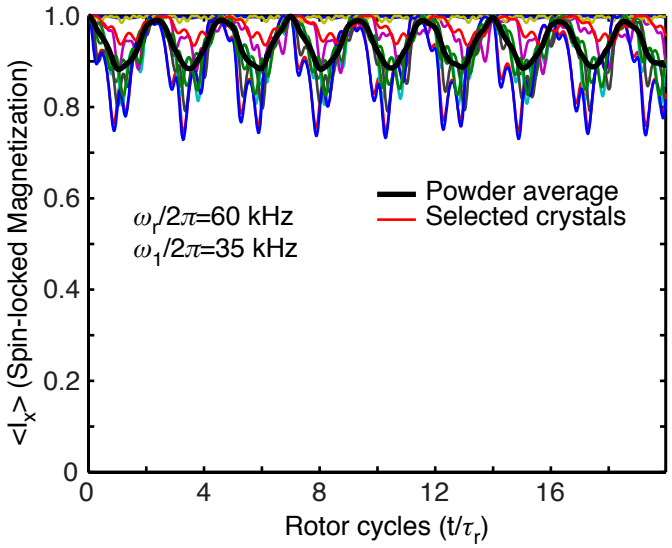
To show that ^{13}C–^{13}C PDSD between Cα and C' should be fully suppressed at 60 kHz, we measure a ^{13}C–^{13}C 2D PDSD spectrum of crystalline MLF at 60 kHz MAS (fully protonated). If any residual transfer exists, it should be visible in this sample, both due to the high signal to noise, and because full protonation should make the PDSD transfer optimal. Mixing was performed for 0.5 s. As one sees in SI Figure 4, traces through the three Cα peaks show no magnetization transfer to the C'. We then argue that for samples which do show Cα to C' transfer, especially those which are sparsely protonated, must have another mechanism of transfer- most likely C–C NOE.



SI Figure 4: ^{13}C–^{13}C PDSD with 0.5 s mixing of MLF. The Cα–Cα and Cα–C' regions are shown, with traces going through the maximum of each Cα peak, showing that no transfer to the C' is observable.

1.5 Trajectory of Spin-Locked Magnetization

In SI Figure 5, we show the simulated spin-locked magnetization of several crystallite orientations as a function of the spin-lock time (without any motion). Although the magnetization periodically refocuses, we see that the evolution of the density operator is significant on the timescale of the rotor period, and, therefore, the Redfield limit is not satisfied near this timescale.



SI Figure 5: Simulated time evolution of magnetization along a 35 kHz spin lock at 60 kHz MAS. Traces are shown for the powder average (black), and selected orientations (colored), and correspond to a two-spin system with no CSA, a dipole coupling of $\delta^{IS}/2\pi$ = 46.6 kHz ($^{1}H\alpha$–$^{13}C\alpha$ coupling), and the carrier placed on resonance.

1.6 Introduction to Liouville-Space Simulation

We briefly summarize the methods of Liouville-space simulation used in this paper. See (Abergel and Palmer 2003; Schneider and Freed 1989) for many more details.

The evolution of magnetization in NMR can be calculated in Hilbert space, using the familiar Liouville von-Neumann equation

$$\frac{d}{dt}\hat{\sigma}(t) = -i\left[\hat{H}(t), \hat{\sigma}(t)\right] \tag{3}$$

where $H(t)$ is the time-dependent Hamiltonian, and $\rho(t)$ is the density matrix (a square matrix) describing the state of the system. For a time-independent Hamiltonian, the time dependence is then given by

$$\hat{\sigma}(t) = \exp(-i\hat{H}t)\hat{\sigma}(0)\exp(i\hat{H}t)\,. \tag{4}$$

The time dependence of the Hamiltonian (due to MAS) is accounted for by taking short time steps during which the Hamiltonian can be treated as time-independent. Then, equivalent equations can be written in Liouville space, as

$$\begin{aligned} &\frac{d}{dt}\hat{\sigma}(t) = -i\hat{\hat{H}}\hat{\sigma}(t) \\ &\hat{\sigma}(t) = \exp\left(-i\hat{\hat{H}}t\right)\hat{\sigma}(0) \end{aligned}\,. \tag{5}$$

Here, $\hat{\hat{H}}$ is the Hamiltonian superoperator, which can be calculated by

$$\hat{\hat{H}} = \hat{H} \otimes \hat{E}_N - \hat{E}_N \otimes \tilde{\hat{H}}\,. \tag{6}$$

where $\hat{E}_N$ is the *N*x*N* identity matrix with the same dimensions as $\hat{H}$, $\tilde{\hat{H}}$ is the transpose of $\hat{H}$ (not the complex transpose), and $\otimes$ is the direct (Kronecker) product. Also in (5), $\hat{\sigma}(t)$ is treated as a column vector instead of as a square matrix, although with the same elements ordered columnwise. This representation has the advantage that it is now possible to introduce exchange processes to a simulation. In this paper, we use exchange between orientations of the dipole and CSA interactions, and also use exchange of the state of a ^{1}H. In the first case, one has a separate Hamiltonian for each orientation, which then are coupled by an exchange matrix, to generate the full Liouville matrix.

$$\begin{aligned} &L = i\begin{bmatrix} \hat{\hat{H}}_1 & 0_{N^2} \\ 0_{N^2} & \hat{\hat{H}}_2 \end{bmatrix} - \hat{K} \otimes \hat{E}_{N^2} \\ &\frac{d}{dt}\hat{\sigma}(t) = -L\hat{\sigma}(t) \end{aligned}\,. \tag{7}$$

Here, $\hat{\hat{H}}_1,\hat{\hat{H}}_2$ are the Hamiltonian superoperators for the two orientations, 0_{N^2} and $\hat{E}_{N^2}$ are a zero-filled matrix and an identity matrix with the same dimensions as $\hat{\hat{H}}_1,\hat{\hat{H}}_2$, and $\hat{K}$ is an exchange matrix between the two orientations (2x2 for two-site exchange).

To generate exchange of the state of a spin (spin-up or spin down) one calculates the Liouville matrix as follows, assuming that spin *S* undergoes the inversion.

$$\begin{aligned} L &= i\hat{\hat{H}} + \Gamma \\ \Gamma &= \frac{k}{2}\left(\hat{\hat{S}}_x^2 + \hat{\hat{S}}_y^2 + \hat{\hat{S}}_z^2\right). \\ \frac{d}{dt}\hat{\sigma}(t) &= -L\hat{\sigma}(t) \end{aligned} \tag{8}$$

Here, the $\hat{\hat{S}}_x,\hat{\hat{S}}_y,\hat{\hat{S}}_z$ are the superoperators of the usual spin-matrices, calculated the same way as the Hamiltonian superoperator in (6). *k* is the rate of spin inversion. The term $\hat{\hat{S}}_z^2$ is included to achieve spatial symmetry of the applied relaxation operation, although does not contribute to the inversion rate of the spin.

Using these equations, one may simulate the effects of motion or spin-inversion on magnetization. Note that it is then necessary to introduce MAS by re-calculating the Liouville matrix periodically throughout the rotor period.

1.7 Laboratory and Rotating-Frame Simulations of Two-Site Hopping Model

We provide MATLAB scripts (SI Section 5) to perform the simulations shown in Figure 2. 'tc_v_MF.m' will calculate $R_{1\rho}$ using Redfield theory and model-free correlation functions (Kurbanov et al. 2011), and compare it to $R_{1\rho}$ simulated using the stochastic Liouville equation in the rotating frame (20-25 min CPU time, real time is ~3 min on mid-2012 MacBook Pro Retina on 8 cores). The simulation generates Hamiltonians for the two-spin system before and after the spatial orientation has been changed by the hopping angle (5º, via rotation around the β-angle of the dipole coupling). Each Hamiltonian is then transformed into Liouville space. Each of the two orientations corresponds to a subspace of the full Liouville space, and the two subspaces are coupled by an exchange constant. The Hamiltonians and Liouville space are recalculated for different powder orientations, and in time to account for MAS (see code for details). 'LF_sim.m' will perform the same calculation using laboratory-frame stochastic Liouville simulations; however, the CPU time for this is 200-300 hours. The calculation is easily parallelized, so may be run in reasonable time on a cluster.

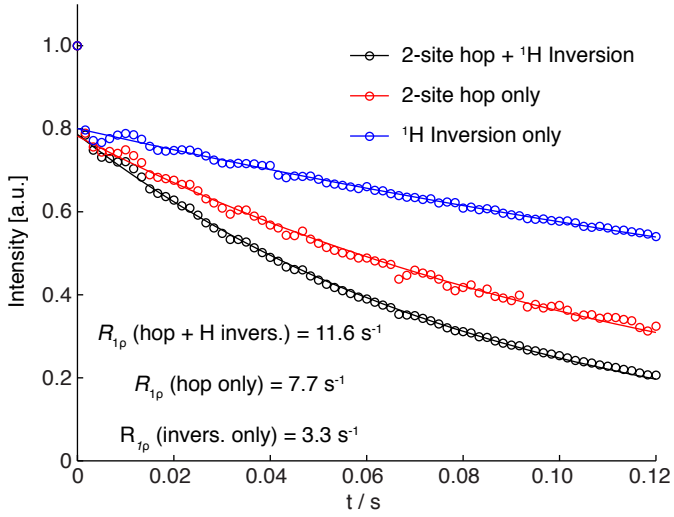
1.8 Simulations of ^{1}H Inversion Induced $R_{1\rho}$

We provide the MATLAB script 'Hflip_ind_R1p.m' (SI Section 5) which calculates the time dependency of the magnetization for a spin-locked nucleus, while the ^{1}H inverts with a given rate (75 s CPU time,

real time ~10 s on mid-2012 MacBook Pro Retina on 8 cores). This program uses a single Hamiltonian, which is then transformed into Liouville space. In Liouville space, the ^{1}H spin-up and spin-down states are then coupled by an exchange constant. As with calculation of the two-site hopping model, the Liouville space is recalculated for different orientations and in time to account for MAS.

1.9 Combined Motion and ^{1}H Inversion Induced $R_{1\rho}$

We compensate for $R_{1\rho}$ induced by inversion of the bonded ^{1}H by simply simulating the decay of the spin-locked magnetization resulting from a given inversion rate, and subtracting this from the experimental $R_{1\rho}$ value. It is critical that the observed $R_{1\rho}$ value is a sum of $R_{1\rho}$ induced by stochastic motion and the decay induced by the ^{1}H inversion. We verify this using a simulation that includes both effects, and compare this rate to the sum of rates for only a motion (a two-site hop in this case), and for only ^{1}H inversion. The results are shown in SI Figure 6, for the case of a ^{1}Hα–^{13}Cα spin pair with MAS of $\omega_r/2\pi$ = 60 kHz and field strength of $\omega_1/2\pi$ = 48 kHz. We see that $R_{1\rho}$ including both effects gives a relaxation rate of 11.6 s^{-1}, whereas the sum of the individual effects is 11.0 s^{-1}, which is reasonably good agreement. We test this near rotary-resonance, where the effect of ^{1}H inversion is strongest. The script for this simulation is given in 'hop_and_1Hinvert.m' (SI Section 5).

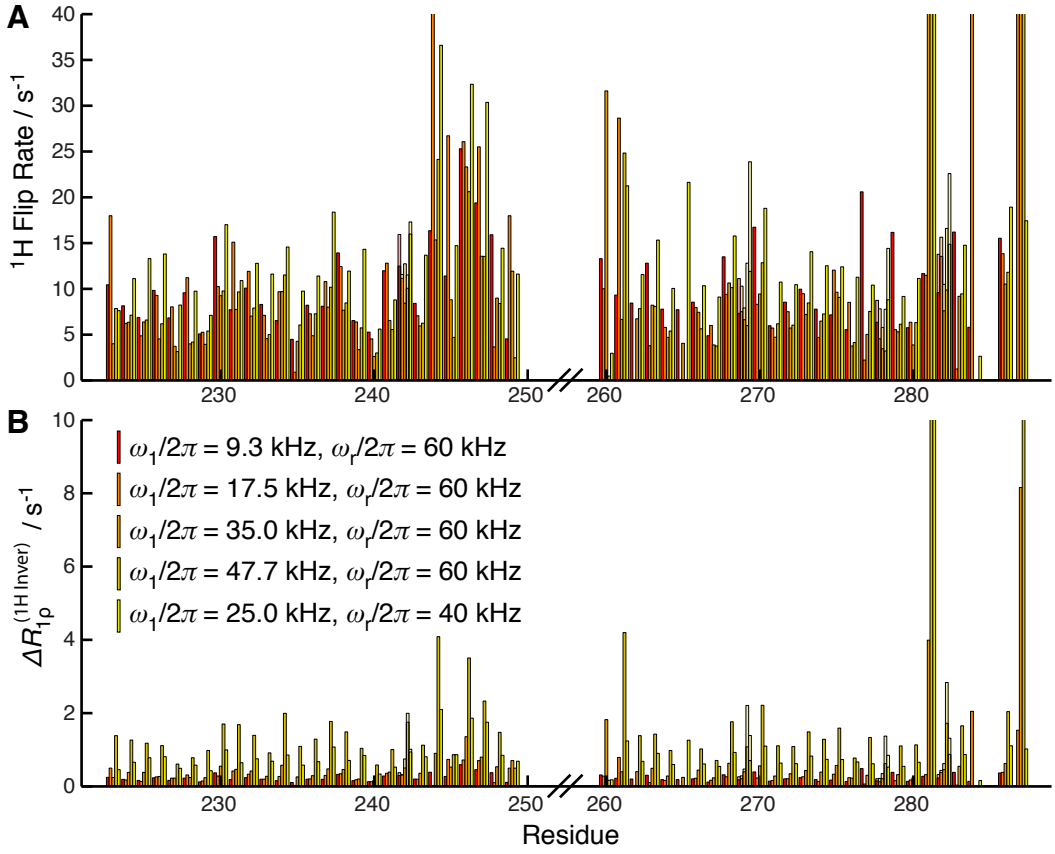


SI Figure 6: $R_{1\rho}$ relaxation is calculated as a result of a combination of slow, two-site hopping motion (τ_c = 7.2 μs, S^2 = 0.999) and stochastic ^{1}H inversion (k = 20 s^{-1}). Simulation is performed for a ^{1}Hα–^{13}Cα spin pair ($\delta^{IS}/2\pi$ = 46.6 kHz), with MAS of $\omega_r/2\pi$ = 60 kHz and field strength of $\omega_1/2\pi$ = 48 kHz. Black curves show the combination of both two-site hopping and ^{1}H inversion, whereas red shows only two-site hopping, and blue shows only ^{1}H inversion. Circles indicate calculated data, and lines indicate a fit to an exponential, of the form $I = A\exp(-R_{1\rho}t)$, where the time point t = 0 s was omitted.

1.10 Correction of $R_{1\rho}$ from ^{1}H Inversion

For each $R_{1\rho}$ measurement recorded, we have measured the ^{1}H inversion rate under the same conditions, and from this value, calculated $\Delta R_{1\rho}$ induced by ^{1}H inversion, and subtracted this away

from the observed value, the final value being used in fitting to dynamic models. Correction for ^{15}N data is given in text Figure 5, and for ^{13}Cα data in SI Figure 7.



SI Figure 7: In (A), we show the measured ^{1}H flip rate for each spin-lock and spinning frequency, matching the conditions used for measurement of ^{13}Cα $R_{1\rho}$ relaxation. The rates shown in (A) are then used in (B) to calculate the correction, $\Delta R_{1\rho}^{(1H\ inver)}$. The different experimental conditions are shown, with a different colored bar for each measurement condition, following the color code shown in (B).

2 Experimental Methods

2.1 Sample Preparation

Two samples of HET-s (218-289) were prepared for this study. Sample #1 is used for measurement of dynamics along the Hα–Cα bond. Sample #1 is partially protonated (25%) at the Hα positions, and also partially protonated at other positions, but all exchangeable sites are fully deuterated. Sample #2 is used for measurement of dynamics along the H–N bond. Sample #2 is fully deuterated except at the exchangeable sites, where it is fully protonated. The samples are prepared according to the same procedure, with exceptions noted below.

Uniformly ^{2}H,^{13}C,^{15}N-labeled histidine-tagged HET-s(218-289) was recombinantly expressed in *Escherichia coli* BL21 in M9 minimal medium, containing ^{2}H,^{13}C glucose as the sole carbon source and $^{15}NH_4Cl$ as the sole nitrogen source. For sample #1, the growth buffer contains 75% D_2O and 25% H_2O, yielding 25% protonation at the Hα position, and 25% or less protonation at other positions. For sample #2, the growth buffer is 100% D_2O, yielding full deuteration at all non-exchangeable sites. The purification steps were performed in H_2O-based buffer solutions. The standard expression steps are described in Balguerie et al.(Balguerie et al. 2003) After expression, the cells were lysed in 150 mM NaCl and 50 mM Tris-HCl, pH = 8 and disrupted using a microfluidizer (Microfluidics). The lysate was centrifuged for 90 min at 8,250 g. The pellet was resuspended in buffer (150 mM NaCl and 50 mM Tris-HCl, pH = 8) and guanidinium HCl powder was added until the solution had doubled in volume. The sample was incubated overnight at 60ºC. The supernatant was cleared by centrifugation for 3 h at 186,000 g and filtered over 0.2 µm pore-size filters. Equilibrated Ni-Sepharose was added to the sample and the protein was binding over one hour to the resin. The sample was transferred to a column and the resin was washed with eight times column volumes buffer (6 M guanidinium HCl, 20 mM Imidazole, 0.5 M NaCl, 50 mM Tris-HCl, pH = 8). The protein was eluted with elution buffer (6 M guanidinium HCl, 500 mM Imidazole, 0.5 M NaCl, 50 mM Tris-HCl, pH = 8). 1 mL fractions were taken and the fractions with protein were pooled in one Falcon tube. Sample #1 was then lyophilized and resuspended in D_2O three times, in order to deuterate all exchangeable sites. For both samples, the buffer was exchanged with 150 mM acetic acid pH = 2.5 using a HiPrep 26/10 desalting column (GE Healthcare). The pH was then immediately adjusted to 7.4 by addition of 3 M Tris at 25ºC, which triggers fibrilization. Sample #1 was fibrilized in D_2O, and sample #2 was fibrilized in H_2O, yielding full protonation at the exchangeable sites. The sample was placed on a rotator during the fibril formation. The aggregation of HET-s (218-289) into fibrils was visible after about ten minutes. After three days, the fibrils were washed three times with D_2O for sample #1 and three times with H_2O for sample #2 and one time with a buffer containing 50 mM citric acid pH = 5 (also with D_2O for sample #1 and H_2O for sample #2 and #3).

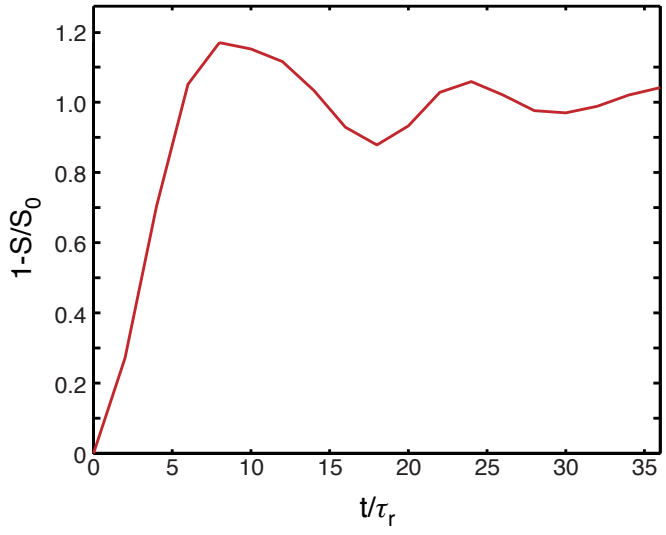
2.2 Experimental Parameters

We describe here the experimental parameters for measurement of dynamic quantities (R_1, $R_{1\rho}$, S^2). All experiments are performed at 60 kHz MAS unless otherwise noted (see last $^{13}C\alpha$ $R_{1\rho}$ measurement). For all 2D experiments, the initial and final CP conditions were set such that the ^{1}H field strength has a mean amplitude of 97 kHz, and used an adiabatic tangent shape, running from 80% of the mean amplitude to 120% of the mean amplitude, and covering tangent angles from -40º to +40º. Note that the final CP runs the reverse direction, from 120% to 80% of the mean amplitude, which gives higher overall efficiency. The ^{13}C or ^{15}N CP field strength was optimized for maximum

signal near the n = 1 zero-quantum CP condition (theoretical rf-field amplitude of 37 kHz on the heteronucleus for experiments at 60 kHz MAS but experimental optimum typically around 42 kHz). For ^{1}Hα–^{13}Cα, the CP transfer is performed for 900 μs, and for ^{1}H–^{15}N, the CP transfer is performed for 1.1-1.2 ms. For all experiments, WALTZ16 decoupling was used both during the indirect and direct dimensions, with 10 kHz for ^{1}H decoupling, 5 kHz for ^{13}C decoupling, and 5 kHz for ^{15}N decoupling). MISSISSIPPI water suppression was performed for 150 ms in all experiments, with ~15 kHz field strength on ^{1}H. All ^{1}H hard pulses (except REDOR π-pulses) use 125 kHz field strength, and all ^{15}N and ^{13}C hard pulses use 50 kHz field strength. REDOR experiments use ^{1}H π-pulses with length of 3.33 μs, and strength of 150 kHz. For ^{1}Hα–^{13}Cα REDOR, the selective ^{1}Hα pulse utilizes a Q3 Gaussian cascade, with a length of 1376 μs. For ^{1}H–^{15}N REDOR, the pulses are shifted so that the delay between the first and second pulse is 9.87 μs, and between the second and third is 0.13 μs (as opposed to having equal delays of 5 μs).

2.3 Finite-Pulse REDOR Time-Trace

We use MATLAB (The MathWorks 2013) to simulate finite-pulse REDOR, used for measurement of S^2 for ^{1}Hα–^{13}Cα motion (text Figure 9E). We fit experimental data by adjusting the dipole coupling used to simulate the curve shown in SI Figure 8. We also include an amplitude scaling factor, because imperfect inversion of the ^{1}H by the selective pulse results in a decrease in the dephasing amplitude, although does not otherwise affect the shape of the REDOR curve significantly.

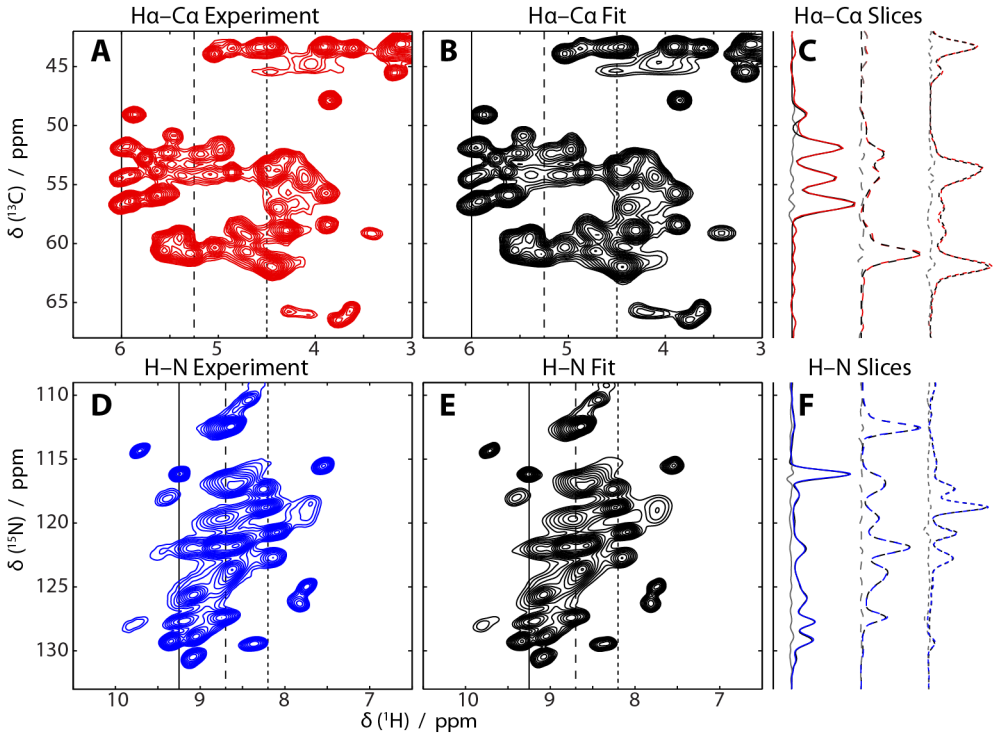


SI Figure 8: Simulation of finite-pulse REDOR, using $\omega_r/2\pi$ = 60 kHz and a dipole coupling of $\delta^{IS}/2\pi$ = 44.77 kHz, with π-pulse strengths of 150 kHz, and a selective Q3 pulse with length of 1376 μs.

3 Data Analysis

3.1 Spectrum Fitting

A significant amount of overlap among resonances makes it difficult to extract peak maxima or to perform accurate peak integration. Therefore, we use spectrum fitting to extract peak amplitudes. SI Figure 9 shows example 2D ^{1}H–^{15}N and ^{1}H–^{13}C correlation spectra, and the corresponding fitted spectrum, along with traces extracted from the experimental and fitted spectra, as well as from the error.



SI Figure 9: ^{1}Hα–^{13}Cα and ^{1}H–^{15}N Correlation Spectra and Fits. (A) and (D) show examples of the experimental ^{1}Hα–^{13}Cα and ^{1}H–^{15}N spectra acquired at 850 MHz, respectively. (B) and (E) show calculated spectra that have been fitted to the experiment. (C) and (F) show traces of the experimental spectra (red/blue), traces of the calculated spectra (black), and fit error (experimental - calculated, grey). Traces are marked on the 2D spectra with solid or dashed lines, matching those in (C) and (F).

3.2 Fitting Procedure

The process of fitting experimental data to motional parameters (correlation times, τ_c and order parameters, S^2) is performed by χ^2 minimization. The minimization is achieved by generating a grid of all the motional parameters to be fitted and calculating all experimentally measured parameters for each grid point via the methods discussed in the main text. The experimental and calculated parameters are then used to calculate χ^2 for all elements in the grid. The minimum value of χ^2 is

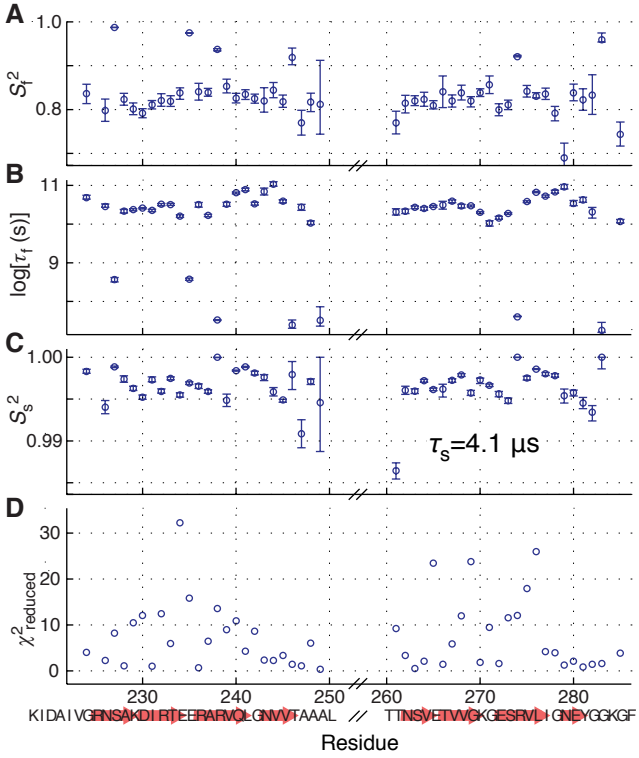
determined, and then a simplex minimization is performed starting from this point, in order to refine the values of the fit parameters.

The error on the dynamics parameters can also be estimated using χ^2. If the best-fit value of χ^2 is given by χ^2_{min}, then statistically, we expect a probability of 0.68 that the real χ^2 (χ^2 obtained if the correct parameters are known) is within 1 of χ^2_{min}. Therefore, we can obtain the 68% confidence interval for a given parameter by searching for the maximum and minimum values of the parameter for which $\chi^2 < \chi^2_{min}+1$. Note that the other fit parameters should be allowed to vary when searching for the maximum and minimum of a particular parameter (Clifford 1973). The error is also determined using the grid of χ^2 values, but in this case, one searches for either the maximum or minimum of the parameter where $\chi^2 < \chi^2_{min}+1$ is still satisfied. This is also followed by a simplex minimization, requiring that the parameter is maximized or minimized and $\chi^2 < \chi^2_{min}+1$ is satisfied.

4 Results and Discussion

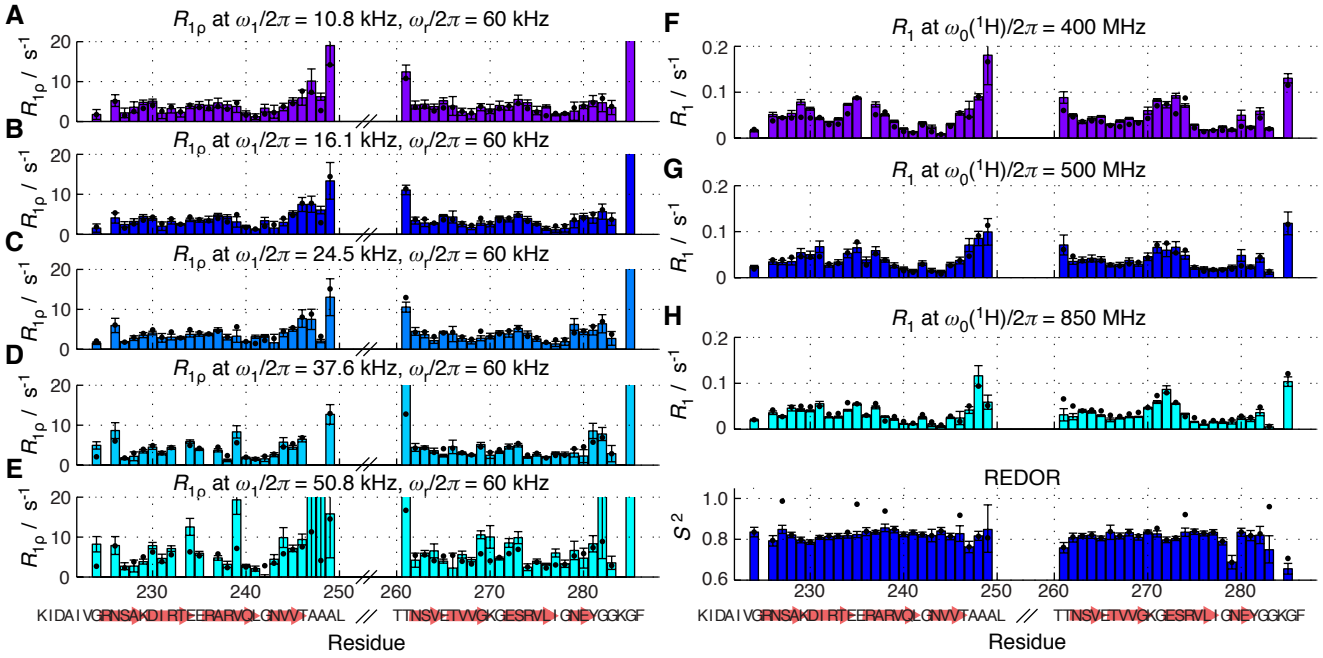
4.1 ^{1}H–^{15}N Two-Timescale Fit

We determined that the three-timescale fit of ^{1}H–^{15}N data better fits the experimental data for most residues. We also perform the two-timescale fit, with the parameters shown here in SI Figure 10, and a comparison of the experimental and calculated parameters shown in



SI Figure 10: Fit parameters for ^{1}H–^{15}N dynamics using a two-timescale model. (A) gives the residue specific fast motional amplitude (S_f^2). (B) gives the fast motional timescale (τ_f), plotted logarithmically. (C) gives the slow motional

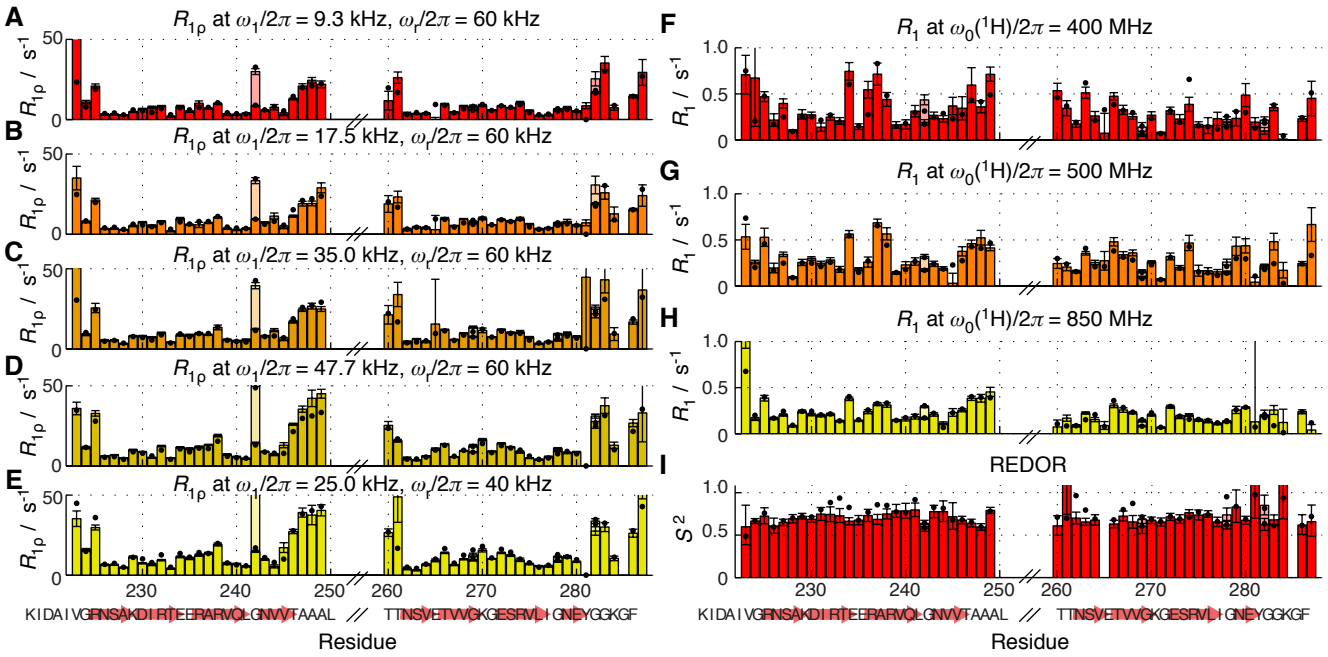
amplitude (S_s^2), using a correlation time of τ_s = 4.1 µs. (FD gives the reduced-χ^2 of the fit. (A-C) show error bars, which approximate a 68% confidence interval for the parameter.



SI Figure 11: Comparison of experimental results (colored bars) with calculated values (black dots) for ^{1}H–^{15}N data, fitted to a two-timescale fit.

4.2 ^{1}Hα–^{13}Cα Two-Timescale Fit

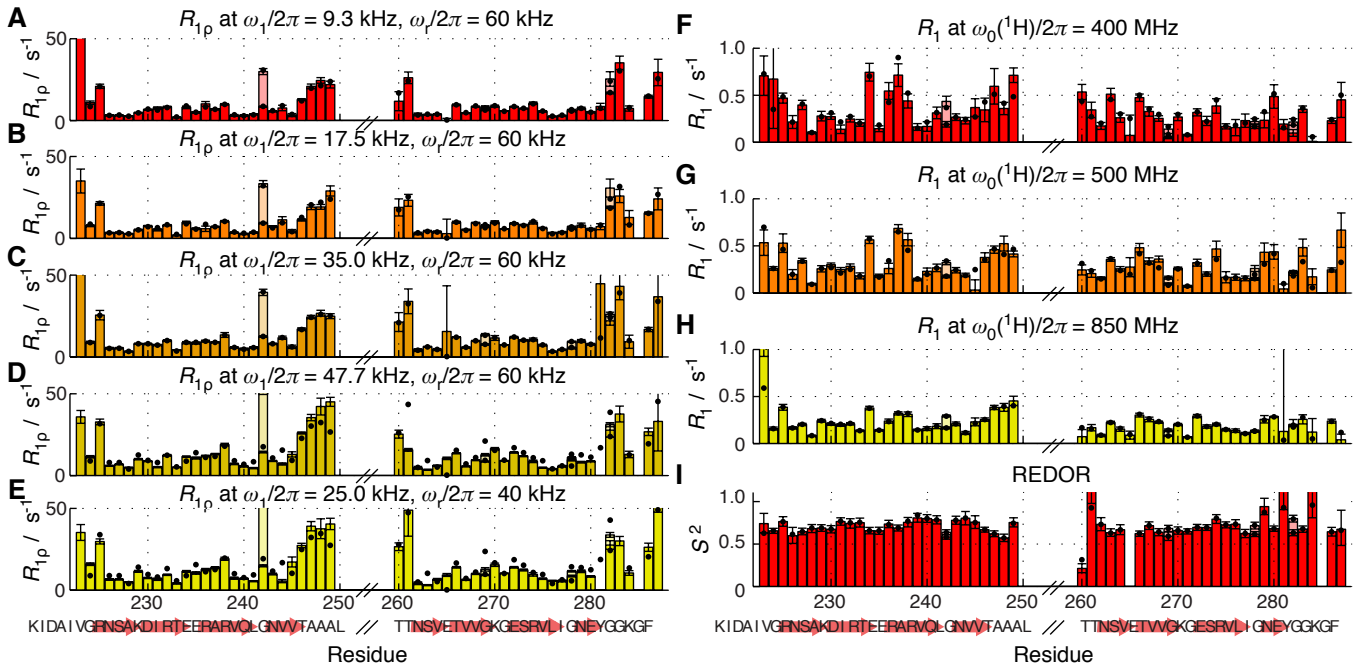
We determined that the three-timescale fit of ^{1}Hα–^{13}Cα data better fits the experimental data for most residues. However, we also perform the two-timescale fit, with the fit parameters shown in the main text, Figure 16. In SI Figure 12, we show the comparison of experimental data to calculated parameters.



SI Figure 12: Comparison of experimental results (colored bars) with calculated values (black dots) for ^{1}Hα–^{13}Cα data, fitted to a two-timescale fit.

4.3 ^{1}Hα–^{13}Cα Partial Fit with $R_{1\rho}$ Prediction

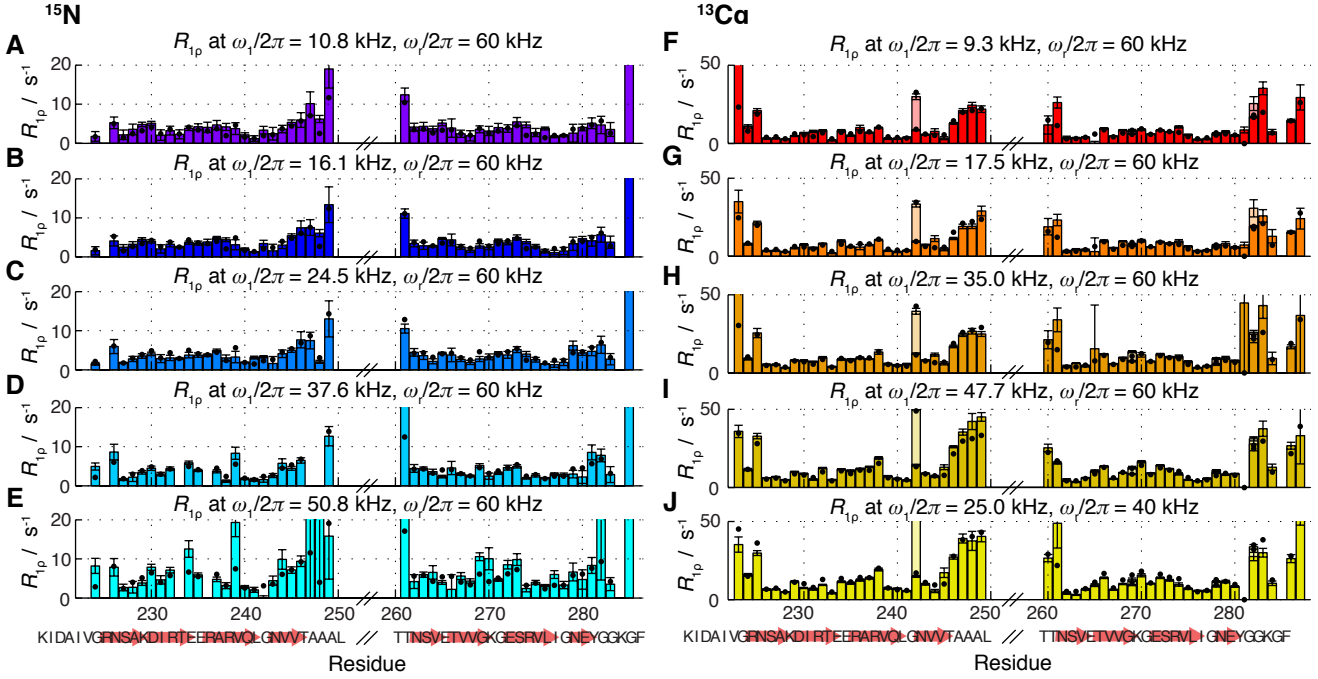
In order to verify that we have obtained a correct timescale for slow motion, we re-fit data obtained for ^{13}Cα, by omitting $R_{1\rho}$ data that was obtained at $\omega_1/2\pi$ = 47.7 kHz, $\omega_r/2\pi$ = 60 kHz and at $\omega_1/2\pi$ = 25.0 kHz, $\omega_r/2\pi$ = 40 kHz. We do this using the three-timescale model. We then calculate what the $R_{1\rho}$ values should be, and compare these to the experimental values. This, as well as the rest of the fit results, is plotted in SI Figure 13. We see that the experimental data is usually well reproduced. This supports our choice of a slow correlation time of 8.7 μs. One also notes that ^{1}H inversion induced $R_{1\rho}$ does not fit well to a motional model when multiple spinning frequencies are utilized (see main text, Figure 8). By omitting data acquired at 40 kHz MAS, we would expect to see disagreement if ^{1}H inversion were misinterpreted as a slow motion. However, the strong agreement of the experimental to predicted values indicates this is not the case, as expected since we have already corrected the $R_{1\rho}$ for ^{1}H inversion induced $R_{1\rho}$.



SI Figure 13: Comparison of experimental results (colored bars) with calculated values (black dots) for ^{1}Hα–^{13}Cα data. We have obtained this fit by omitting $R_{1\rho}$ data acquired at $\omega_1/2\pi$ = 47.7 kHz, $\omega_r/2\pi$ = 60 kHz and $\omega_1/2\pi$ = 25.0 kHz, $\omega_r/2\pi$ = 40 kHz, and fitting the remaining data, in order to see if the omitted data is still in good agreement.

4.4 Analysis of $R_{1\rho}$ Data without R_1 and S^2

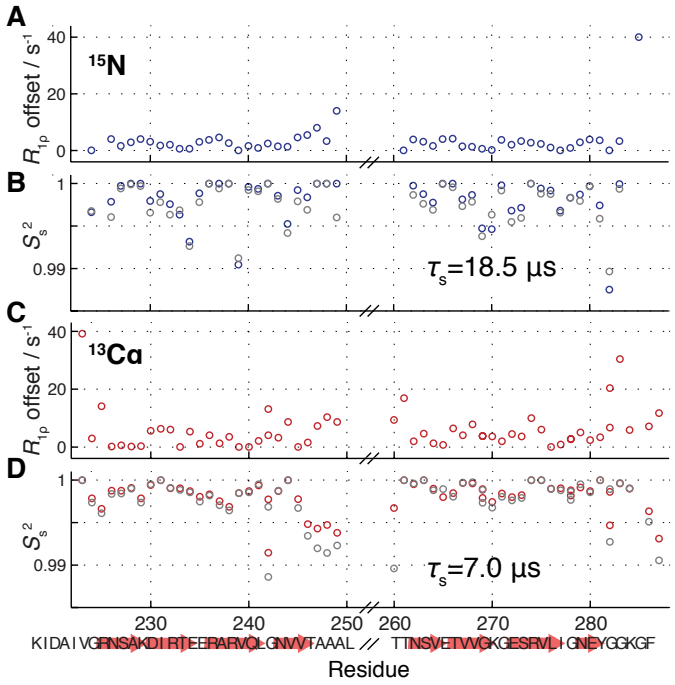
It is possible to fit $R_{1\rho}$ data without R_1 and S^2 data. SI Figure 14 shows the fit of the $R_{1\rho}$ data for ^{15}N and ^{13}Cα data using only a single, slow timescale (corresponding to text figures 17A and 18).



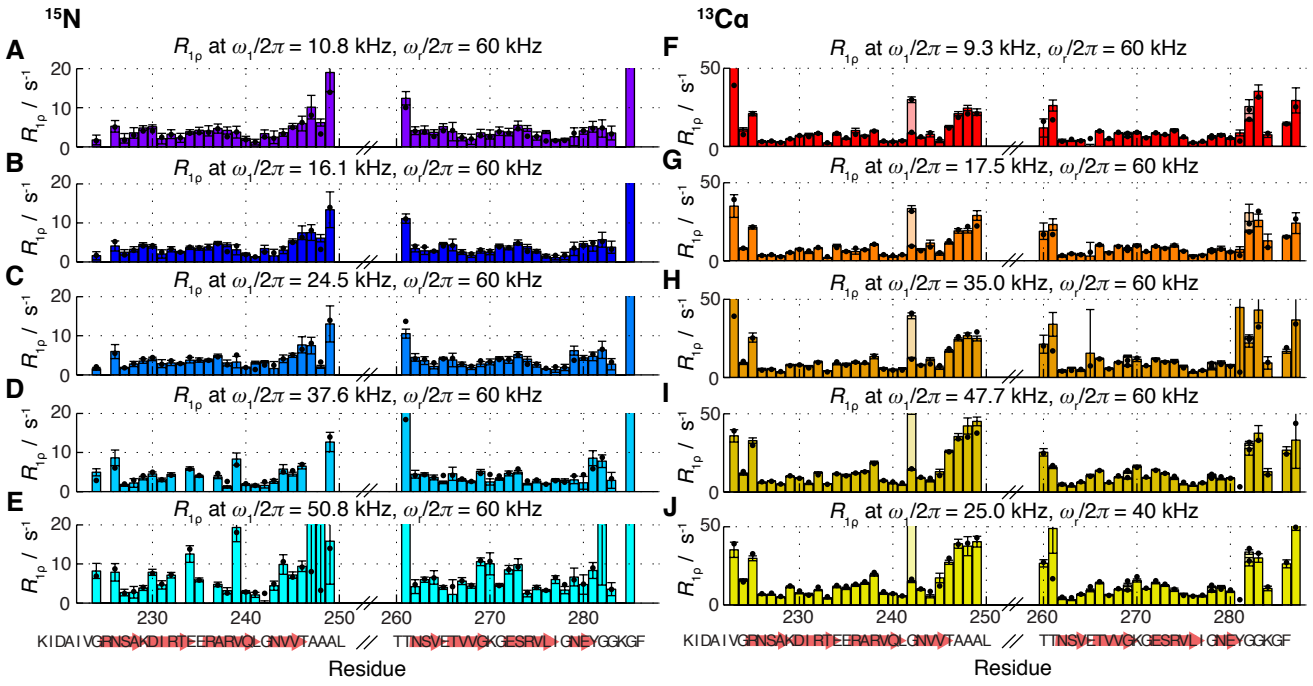
SI Figure 14: Comparison of experimental results (colored bars) with calculated values (black dots), fitted to a single slow motion. A-E give the fit for ^{15}N $R_{1\rho}$ using a correlation time of τ_s = 6.2 μs, and F-J give the fit for ^{13}Cα using τ_s = 4.1 μs.

It is also possible to fit $R_{1\rho}$ data to two timescales, a slow timescale for which a global slow correlation time can be determined (τ_s, see text Figure 17B for optimization of τ_s) as well as site-

specific values for S_s^2 and a contribution to the $R_{1\rho}$ from intermediate timescale motion ($R_{1\rho}$ offset). Note that the correlation time and amplitude of the intermediate timescale motion (τ_i, S_i^2) cannot be determined explicitely from only $R_{1\rho}$. SI Figure 15 shows fitted values of S_s^2 and the $R_{1\rho}$ correction resulting from intermediate timescale motion for ^{15}N and ^{13}Cα $R_{1\rho}$ data. SI Figure 16 shows the fit of the $R_{1\rho}$ data for ^{15}N and ^{13}Cα to the two-timescale model.



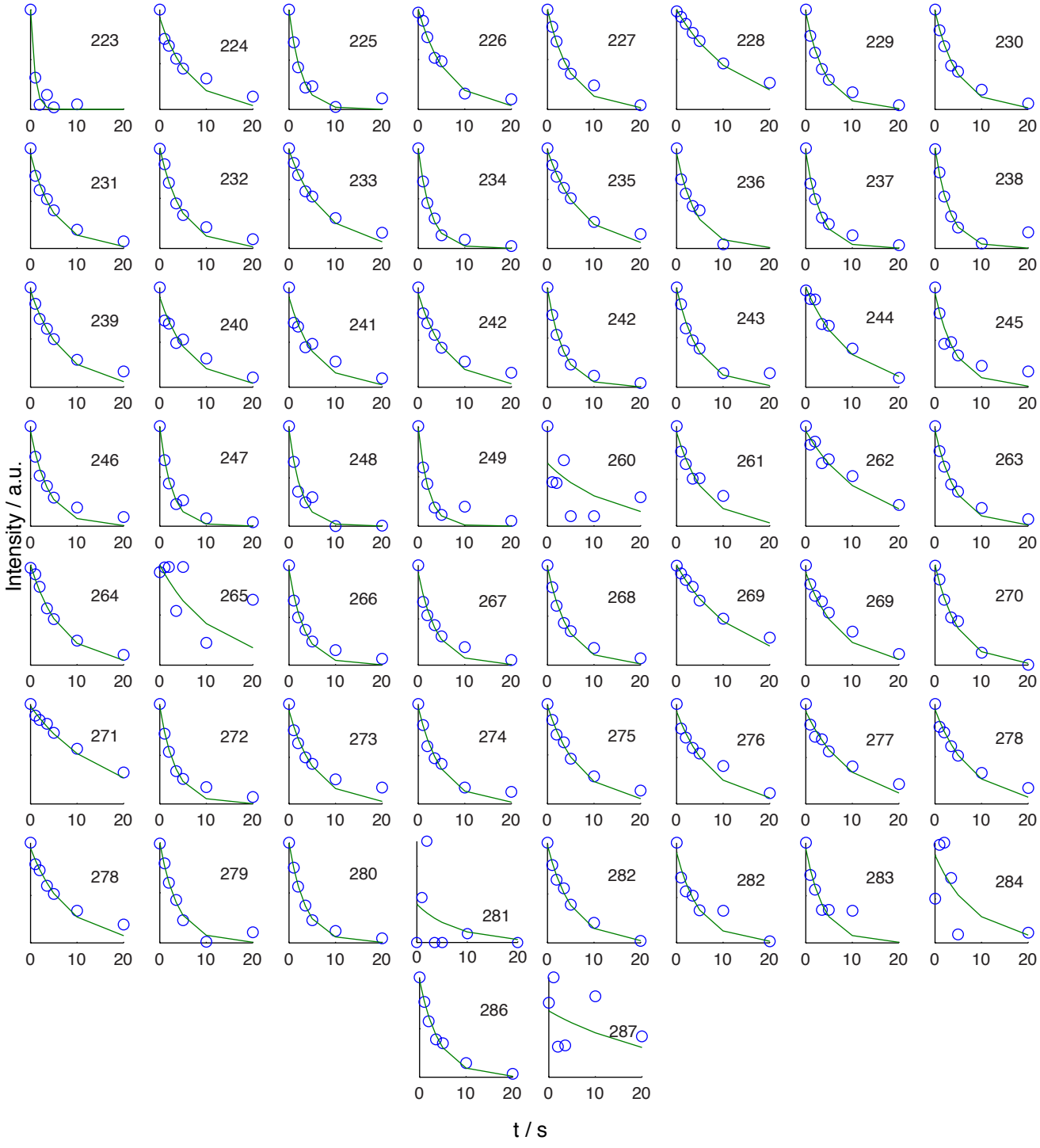
SI Figure 15: S_s^2 determined for ^{1}H–^{15}N and ^{1}Hα–^{13}Cα motion using only $R_{1\rho}$ data. $R_{1\rho}$ data was fitted to a slow motion in addition to an offset of the $R_{1\rho}$, which is the result of intermediate timescale motion. Colored dots in (B) and (D) show S_s^2, for ^{15}N and ^{13}Cα respectively, and the $R_{1\rho}$ offset is shown in (A) and (C). The correlation time of the slow motion is τ_s = 6.2 µs for ^{15}N data and 4.1 µs for ^{13}Cα data. Grey dots in (B) and (D) show S_s^2 determined while fitting the full data set to a three-timescale model, showing that usually fitting to the full data set predicts a lower S_s^2 (larger motion).



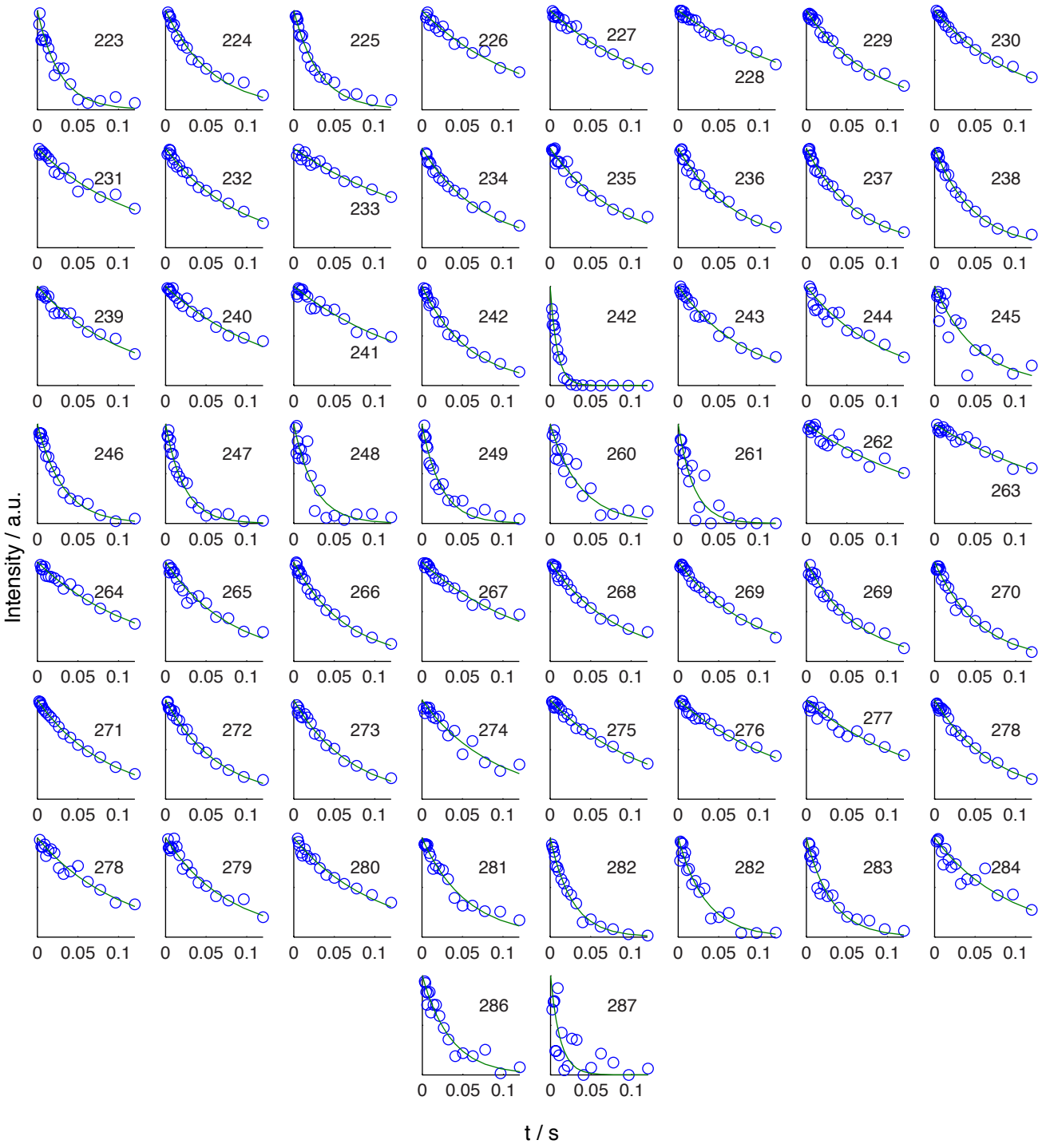
SI Figure 16: Comparison of experimental results (colored bars) with calculated values (black dots), fitted to a slow motion and intermediate motion. A-E give the fit for ^{15}N $R_{1\rho}$ using a correlation time of τ_s = 18.5 μs, and F-J give the fit for ^{13}Cα using τ_s = 7.0 μs.

4.5 Fitting of Experimental Data

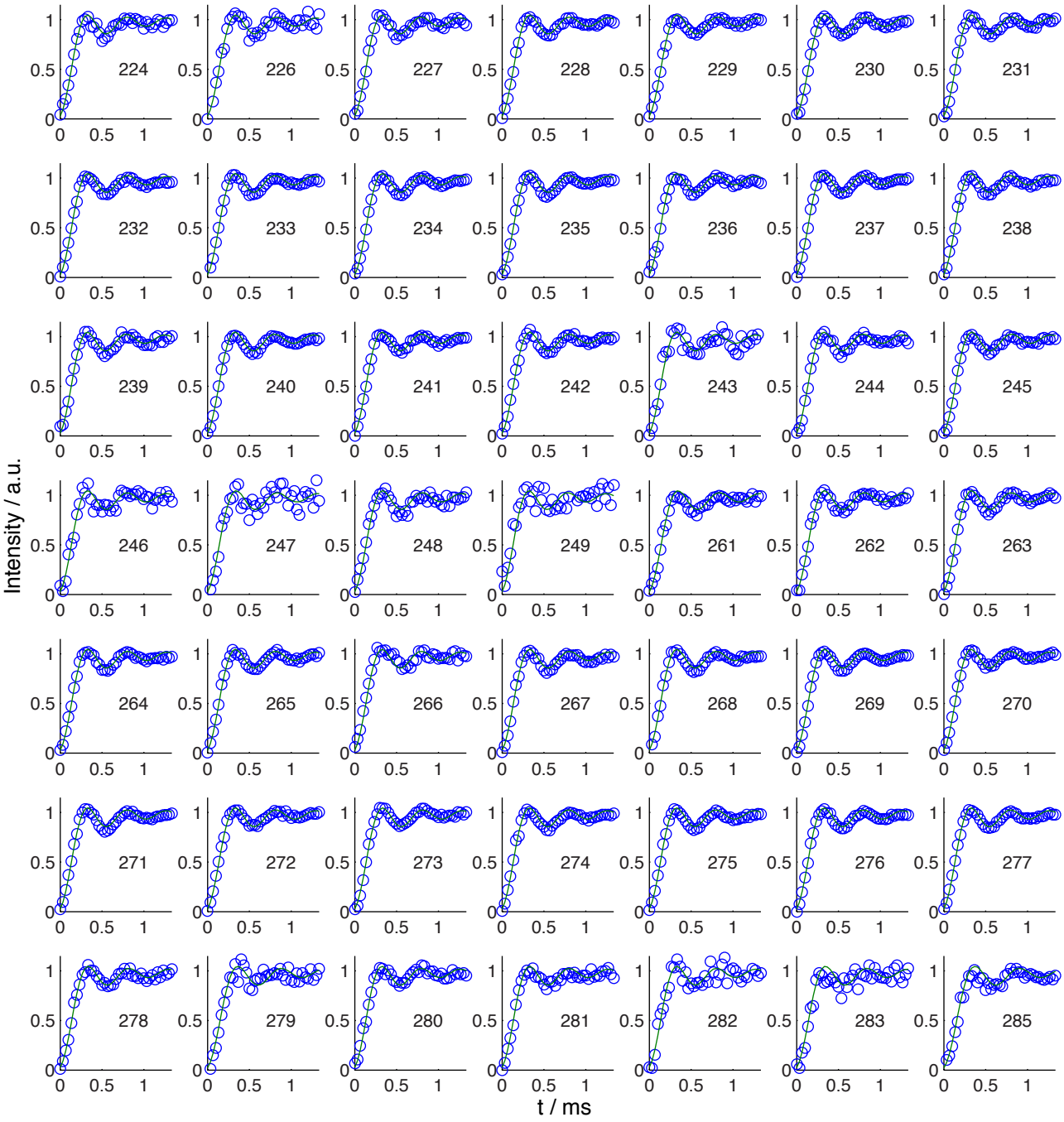
Below we show the curve fitting for several of the experiments used in this study. We show all residues that were analyzed for the given experiment.



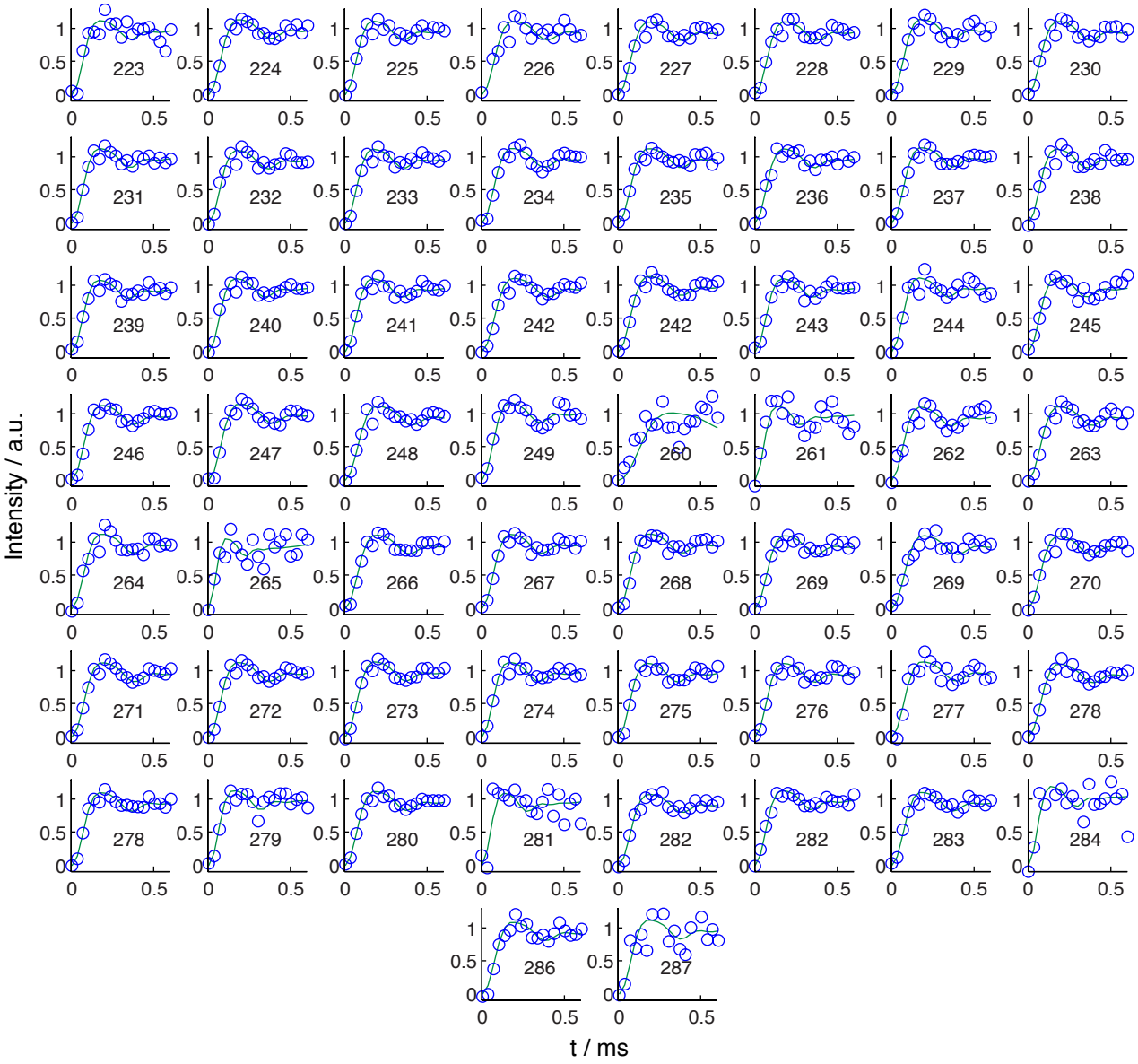
SI Figure 17: ^{13}Cα R_1 Measured at 850 MHz.



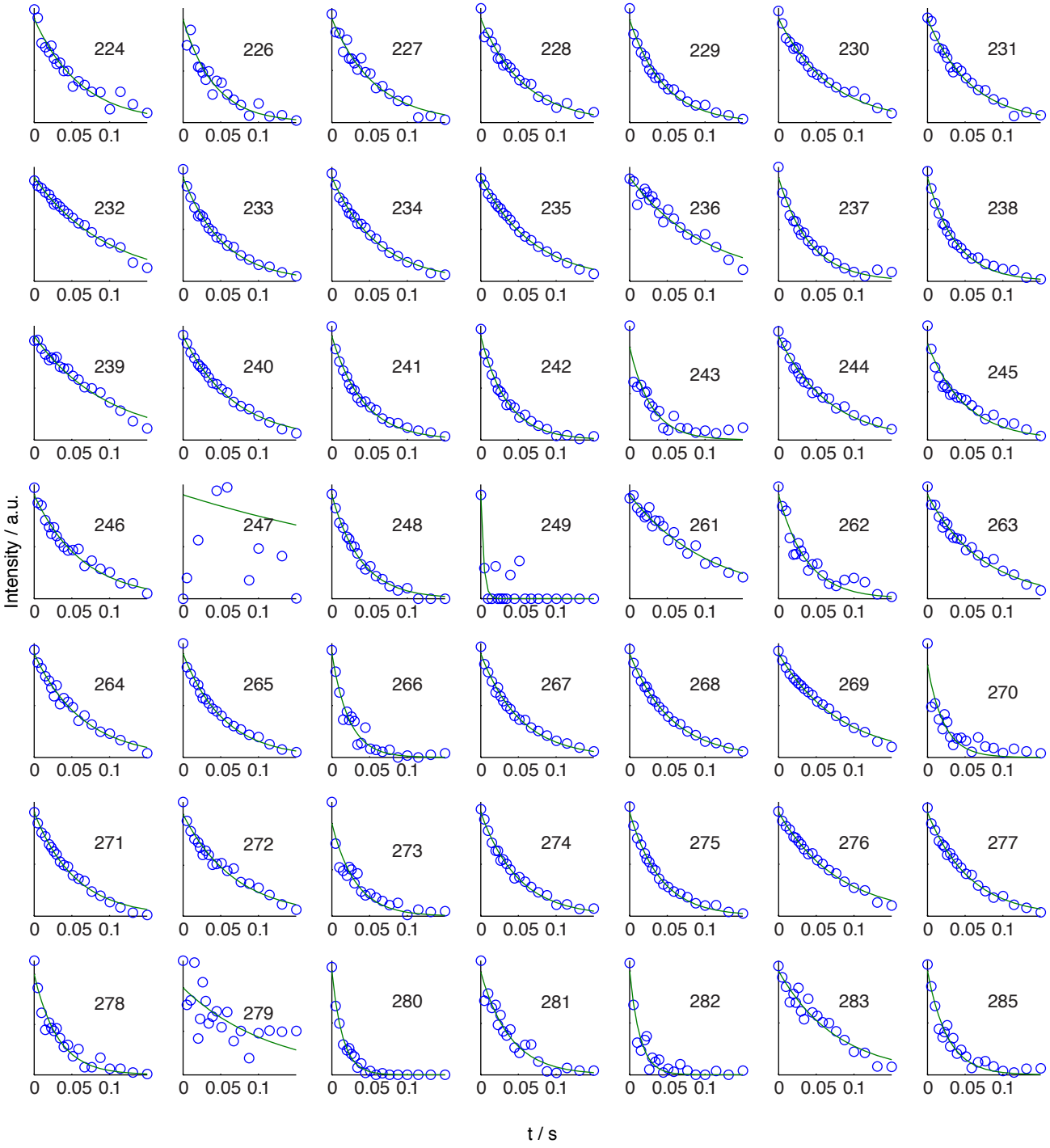
SI Figure 18: ^{13}Cα $R_{1\rho}$ Measured at 850 MHz, with 40 kHz MAS frequency and 25.0 kHz spin-locking strength.



SI Figure 19: $^{1}H–^{15}N$ REDOR measured at 850 MHz.



SI Figure 20: ^{1}Hα–^{13}Cα REDOR measured at 850 MHz.



SI Figure 21: ^{1}H flip rate, measured for correction of the ^{15}N $R_{1\rho}$ at 500 MHz, with 60 kHz MAS frequency and 50.8 kHz spin-locking strength.

4.6 Tables of Data and Calculated Parameters

Table 1: R_1 and S ^{15}N. Error is given for each measurement, and the calculated value is given after each measurement. Note that we give S, not S^2.

Residue	R_1 (400 MHz)	Calc.	R_1 (500 MHz)	Calc.	R_1 (850 MHz)	Calc.	S	Calc.
224	0.016±0.002	0.018	0.024±0.003	0.018	0.020±0.002	0.021	0.911±0.012	0.907
226	0.051±0.005	0.048	0.034±0.005	0.041	0.036±0.004	0.034	0.888±0.015	0.888

227	0.045±0.002	0.044	0.033±0.005	0.036	0.027±0.002	0.027	0.920±0.008	0.921
228	0.054±0.006	0.049	0.035±0.008	0.045	0.046±0.006	0.043	0.906±0.008	0.906
229	0.078±0.006	0.076	0.054±0.009	0.06	0.042±0.003	0.043	0.892±0.008	0.894
230	0.064±0.003	0.064	0.050±0.008	0.052	0.040±0.002	0.04	0.887±0.006	0.887
231	0.043±0.002	0.045	0.067±0.013	0.045	0.055±0.005	0.051	0.899±0.006	0.899
232	0.035±0.001	0.035	0.027±0.004	0.03	0.026±0.001	0.026	0.903±0.009	0.903
233	0.042±0.002	0.042	0.034±0.005	0.034	0.026±0.002	0.026	0.904±0.008	0.904
234	0.073±0.002	0.073	0.053±0.009	0.059	0.041±0.002	0.041	0.905±0.007	0.905
235	0.087±0.002	0.087	0.065±0.010	0.071	0.055±0.002	0.055	0.909±0.008	0.909
236	0.395±0.058	0.049	0.039±0.006	0.039	0.029±0.002	0.029	0.913±0.010	0.913
237	0.073±0.006	0.073	0.059±0.008	0.06	0.048±0.003	0.047	0.914±0.005	0.914
238	0.051±0.002	0.051	0.038±0.005	0.039	0.027±0.002	0.027	0.923±0.010	0.92
239	0.037±0.002	0.036	0.027±0.005	0.03	0.021±0.002	0.021	0.921±0.009	0.921
240	0.021±0.001	0.021	0.020±0.003	0.017	0.011±0.001	0.013	0.908±0.007	0.916
241	0.013±0.001	0.013	0.015±0.002	0.012	0.011±0.001	0.011	0.914±0.006	0.914
242	0.033±0.002	0.035	0.032±0.005	0.03	0.025±0.003	0.025	0.908±0.006	0.909
243	0.024±0.004	0.021	0.017±0.004	0.017	0.009±0.003	0.013	0.906±0.016	0.917
244	0.008±0.001	0.008	0.013±0.002	0.008	0.006±0.002	0.009	0.914±0.010	0.92
245	0.030±0.002	0.031	0.030±0.004	0.027	0.025±0.002	0.025	0.901±0.007	0.901
246	0.054±0.004	0.054	0.037±0.007	0.039	0.024±0.014	0.022	0.910±0.021	0.91
247	0.063±0.017	0.072	0.071±0.014	0.058	0.042±0.007	0.044	0.874±0.017	0.875
248	0.090±0.007	0.091	0.085±0.016	0.089	0.116±0.022	0.09	0.901±0.016	0.9
249	0.180±0.064	0.149	0.099±0.029	0.107	0.059±0.015	0.058	0.926±0.069	0.898
261	0.088±0.013	0.086	0.071±0.022	0.066	0.031±0.013	0.04	0.870±0.016	0.87
262	0.051±0.002	0.05	0.035±0.006	0.038	0.027±0.007	0.026	0.900±0.012	0.9
263	0.035±0.003	0.035	0.039±0.006	0.036	0.040±0.003	0.04	0.904±0.007	0.904
264	0.043±0.003	0.043	0.042±0.008	0.039	0.038±0.003	0.038	0.904±0.009	0.904
265	0.047±0.002	0.048	0.040±0.005	0.038	0.029±0.002	0.028	0.899±0.007	0.895
266	0.029±0.002	0.03	0.027±0.004	0.025	0.020±0.004	0.021	0.913±0.020	0.913
267	0.037±0.003	0.037	0.029±0.004	0.03	0.024±0.002	0.024	0.906±0.008	0.906
268	0.043±0.002	0.043	0.038±0.006	0.035	0.026±0.001	0.026	0.910±0.010	0.91
269	0.037±0.001	0.037	0.029±0.004	0.032	0.027±0.001	0.027	0.898±0.008	0.898
270	0.060±0.008	0.057	0.045±0.008	0.051	0.046±0.002	0.046	0.914±0.005	0.914
271	0.083±0.003	0.083	0.066±0.012	0.07	0.059±0.003	0.059	0.909±0.008	0.909
272	0.073±0.007	0.073	0.060±0.014	0.074	0.087±0.008	0.081	0.894±0.007	0.894
273	0.092±0.005	0.092	0.066±0.012	0.074	0.055±0.001	0.055	0.897±0.006	0.897
274	0.071±0.004	0.071	0.048±0.008	0.053	0.033±0.001	0.033	0.916±0.011	0.911
275	0.029±0.001	0.029	0.022±0.004	0.023	0.016±0.001	0.016	0.915±0.008	0.915
276	0.017±0.001	0.017	0.021±0.003	0.014	0.010±0.001	0.012	0.910±0.007	0.922
277	0.017±0.001	0.017	0.017±0.003	0.017	0.015±0.001	0.019	0.915±0.008	0.92
278	0.024±0.002	0.023	0.019±0.003	0.019	0.014±0.002	0.016	0.888±0.010	0.894
279	0.017±0.002	0.019	0.024±0.003	0.018	0.016±0.004	0.02	0.828±0.019	0.829
280	0.050±0.011	0.053	0.048±0.013	0.04	0.025±0.003	0.025	0.915±0.010	0.915
281	0.024±0.002	0.024	0.024±0.004	0.022	0.021±0.004	0.021	0.908±0.017	0.908
282	0.059±0.007	0.057	0.043±0.009	0.047	0.036±0.007	0.035	0.906±0.024	0.906
283	0.020±0.002	0.019	0.012±0.005	0.016	0.005±0.005	0.014	0.867±0.033	0.9
285	0.130±0.010	0.13	0.118±0.025	0.117	0.104±0.010	0.104	0.808±0.014	0.808

Table 2: R_1 and S ^{13}Cα. Error is given for each measurement, and the calculated value is given after each measurement. Note that we give S, not S^2.

Resi.	R_1(400 MHz)	Calc.	R_1 (500 MHz)	Calc.	R_1 (850 MHz)	Calc.	S	Calc.
223	0.71±0.21	0.75	0.53±0.14	0.71	1.09±0.17	0.6	0.86±0.07	0.8
224	0.67±0.53	0.35	0.26±0.02	0.26	0.16±0.02	0.16	0.81±0.02	0.81
225	0.47±0.05	0.49	0.53±0.10	0.46	0.38±0.03	0.38	0.87±0.03	0.87
226	0.21±0.07	0.22	0.20±0.05	0.19	0.17±0.01	0.17	0.77±0.06	0.78
227	0.40±0.05	0.43	0.34±0.02	0.33	0.20±0.01	0.2	0.80±0.02	0.81
228	0.10±0.01	0.1	0.09±0.01	0.09	0.08±0.01	0.08	0.83±0.03	0.83
229	0.28±0.05	0.27	0.26±0.03	0.26	0.24±0.02	0.24	0.83±0.02	0.83
230	0.27±0.03	0.31	0.30±0.02	0.27	0.21±0.01	0.21	0.82±0.03	0.82
231	0.14±0.05	0.22	0.25±0.02	0.21	0.20±0.01	0.2	0.87±0.03	0.88
232	0.25±0.03	0.27	0.28±0.03	0.25	0.21±0.01	0.21	0.86±0.03	0.88
233	0.21±0.04	0.21	0.18±0.03	0.18	0.14±0.01	0.14	0.87±0.03	0.88
234	0.74±0.09	0.71	0.56±0.04	0.57	0.38±0.02	0.38	0.81±0.02	0.81
235	0.14±0.03	0.18	0.18±0.02	0.17	0.14±0.01	0.14	0.81±0.03	0.81
236	0.55±0.09	0.43	0.26±0.06	0.34	0.24±0.02	0.23	0.86±0.03	0.85
237	0.71±0.12	0.9	0.69±0.04	0.65	0.32±0.02	0.32	0.83±0.02	0.83
238	0.44±0.08	0.52	0.57±0.07	0.45	0.31±0.02	0.32	0.87±0.03	0.88
239	0.16±0.04	0.15	0.14±0.01	0.15	0.14±0.01	0.14	0.90±0.03	0.9
240	0.17±0.05	0.21	0.23±0.03	0.19	0.16±0.02	0.16	0.89±0.03	0.89
241	0.31±0.06	0.31	0.27±0.04	0.26	0.18±0.03	0.19	0.89±0.03	0.89
242	0.19±0.03	0.19	0.18±0.01	0.18	0.17±0.01	0.17	0.76±0.02	0.76
242	0.43±0.06	0.37	0.32±0.02	0.34	0.29±0.01	0.29	0.80±0.02	0.8
243	0.27±0.03	0.26	0.24±0.03	0.24	0.21±0.02	0.21	0.88±0.03	0.88
244	0.23±0.03	0.24	0.19±0.02	0.19	0.11±0.02	0.11	0.90±0.04	0.91
245	0.37±0.09	0.23	0.03±0.11	0.23	0.23±0.04	0.22	0.87±0.04	0.85
246	0.34±0.13	0.43	0.38±0.05	0.35	0.25±0.02	0.26	0.82±0.02	0.82
247	0.59±0.19	0.51	0.46±0.04	0.47	0.38±0.02	0.38	0.79±0.02	0.79
248	0.36±0.06	0.41	0.52±0.08	0.41	0.38±0.04	0.39	0.76±0.03	0.76
249	0.71±0.08	0.48	0.41±0.03	0.47	0.45±0.05	0.4	0.87±0.03	0.86
260	0.53±0.08	0.45	0.24±0.05	0.32	0.07±0.07	0.17	0.46±0.06	0.56
261	0.34±0.08	0.27	0.21±0.04	0.23	0.17±0.03	0.15	1.09±0.10	0.96
262	0.17±0.04	0.2	0.16±0.01	0.15	0.09±0.01	0.09	0.86±0.04	0.85
263	0.51±0.06	0.46	0.36±0.02	0.35	0.22±0.01	0.23	0.80±0.03	0.81
264	0.26±0.05	0.3	0.26±0.02	0.24	0.15±0.02	0.16	0.82±0.04	0.82
265	0.07±0.21	0.24	0.27±0.10	0.19	0.09±0.04	0.1	-	0.96
266	0.47±0.04	0.5	0.48±0.04	0.42	0.30±0.02	0.31	0.79±0.02	0.79
267	0.33±0.05	0.35	0.34±0.03	0.32	0.26±0.02	0.26	0.85±0.03	0.85
268	0.25±0.03	0.3	0.36±0.03	0.27	0.23±0.01	0.23	0.80±0.03	0.8
269	0.05±0.01	0.08	0.10±0.01	0.08	0.08±0.01	0.08	0.83±0.02	0.83
269	0.14±0.05	0.17	0.16±0.02	0.16	0.14±0.01	0.14	0.77±0.03	0.77
270	0.27±0.04	0.3	0.26±0.01	0.25	0.20±0.01	0.2	0.81±0.03	0.81
271	0.08±0.01	0.08	0.07±0.01	0.07	0.07±0.01	0.07	0.80±0.02	0.81
272	0.32±0.04	0.32	0.32±0.04	0.31	0.29±0.02	0.29	0.83±0.02	0.84
273	0.23±0.05	0.22	0.20±0.02	0.2	0.18±0.02	0.18	0.83±0.02	0.83
274	0.39±0.08	0.43	0.47±0.08	0.34	0.20±0.01	0.21	0.89±0.03	0.89

275	0.17±0.03	0.17	0.17±0.06	0.16	0.15±0.01	0.15	0.85±0.02	0.85
276	0.16±0.09	0.18	0.17±0.03	0.16	0.14±0.02	0.14	0.86±0.03	0.85
277	0.23±0.07	0.2	0.15±0.03	0.16	0.11±0.01	0.11	0.79±0.03	0.79
278	0.17±0.02	0.17	0.15±0.01	0.15	0.13±0.01	0.13	0.78±0.02	0.79
278	0.22±0.03	0.25	0.26±0.03	0.2	0.13±0.01	0.13	0.85±0.02	0.85
279	0.23±0.08	0.32	0.43±0.10	0.3	0.26±0.04	0.25	0.97±0.05	0.93
280	0.49±0.13	0.52	0.44±0.08	0.41	0.28±0.01	0.28	0.82±0.02	0.83
281	0.19±0.05	0.13	0.04±0.05	0.11	0.13±9.96	0.07	1.11±0.16	0.99
282	0.22±0.04	0.24	0.23±0.01	0.22	0.19±0.01	0.19	0.79±0.02	0.8
282	0.10±0.04	0.18	0.21±0.02	0.18	0.20±0.05	0.17	0.89±0.02	0.89
283	0.35±0.03	0.36	0.48±0.09	0.33	0.26±0.05	0.27	0.82±0.02	0.82
284	0.01±0.05	0.06	0.17±0.08	0.06	0.12±0.14	0.05	1.07±0.17	0.98
286	0.23±0.03	0.24	0.24±0.02	0.24	0.24±0.02	0.24	0.80±0.03	0.8
287	0.45±0.19	0.5	0.67±0.18	0.32	0.04±0.06	0.11	0.82±0.13	0.82

Table 3: $R_{1\rho}$ ^{15}N. Error is given for each measurement, and the calculated value is given after each measurement.

Resi.	$\omega_1/2\pi$= 10.8 kHz $\omega_r/2\pi$= 60 kHz	Calc.	$\omega_1/2\pi$= 16.1 kHz $\omega_r/2\pi$= 60 kHz	Calc.	$\omega_1/2\pi$= 24.5 kHz $\omega_r/2\pi$= 60 kHz	Calc.	$\omega_1/2\pi$= 37.6 kHz $\omega_r/2\pi$= 60 kHz	Calc.	$\omega_1/2\pi$= 50.8 kHz $\omega_r/2\pi$= 60 kHz	Calc.
224	1.76±1.24	1.61	1.56±1.04	1.75	1.53±0.34	2.13	4.93±0.92	2.67	8.20±1.93	5.78
226	5.24±1.48	5.01	4.08±1.29	5.16	5.95±1.79	5.59	8.63±2.00	5.33	7.86±2.24	8.96
227	2.16±1.22	1.7	2.46±0.69	1.73	1.78±0.15	1.83	1.61±0.24	1.61	2.68±0.86	2.45
228	3.56±1.35	3.04	3.22±0.65	3.05	2.77±0.65	3.07	2.18±1.10	2.3	2.74±1.49	2.54
229	4.80±0.58	4.31	4.45±0.48	4.32	3.54±0.58	4.37	3.45±0.44	3.34	3.83±0.62	3.81
230	5.10±0.50	4.09	3.97±0.44	4.23	4.05±0.43	4.61	4.48±0.57	4.48	7.85±0.79	7.63
231	2.08±1.35	2.37	2.03±1.07	2.47	2.85±1.02	2.75	3.00±0.44	2.81	4.64±1.05	4.99
232	3.24±1.12	3.12	3.22±0.66	3.27	3.04±0.68	3.68	4.37±0.44	3.9	7.10±0.71	7.37
233	2.28±1.17	2.37	2.50±0.19	2.5	2.87±0.25	2.87	-	3.18	-	6.21
234	3.81±0.57	3.53	3.60±0.42	3.81	3.73±0.71	4.6	5.89±0.43	5.57	12.54±2.13	12.66
235	4.25±0.76	3.68	3.53±0.51	3.79	3.77±0.54	4.07	3.93±0.26	3.8	5.92±0.48	6.03
236	4.07±1.32	3.76	3.78±0.74	3.76	3.74±0.33	3.76	-	2.73	-	2.71
237	4.67±1.17	4.77	4.86±0.36	4.8	4.91±0.36	4.9	3.69±0.39	3.84	4.76±0.58	4.66
238	4.17±0.90	3.04	4.38±0.50	3.04	3.02±0.75	3.04	1.17±0.32	2.21	3.15±0.79	2.19
239	3.75±1.51	4.08	3.10±1.30	4.43	3.25±1.47	5.41	8.31±1.54	6.65	19.33±3.66	15.48
240	2.17±0.49	1.74	1.98±0.29	1.78	1.87±0.07	1.91	1.80±0.23	1.74	2.83±0.34	2.77
241	1.19±0.74	1.24	1.21±0.08	1.29	3.14±0.36	1.43	1.54±0.36	1.45	2.08±0.63	2.63
242	3.33±0.98	3.02	3.39±0.83	3.02	3.33±0.29	3.02	1.56±0.68	2.19	-0.02±0.57	2.17
243	2.44±1.61	2.15	1.55±1.65	2.24	1.62±1.76	2.49	2.61±0.50	2.53	4.43±1.18	4.45
244	3.62±0.95	3.3	2.98±0.86	3.53	4.10±0.98	4.18	5.74±1.14	4.9	9.85±2.49	10.67
245	5.35±0.60	5.06	5.40±0.47	5.15	4.99±0.54	5.42	4.45±0.49	4.72	7.15±0.82	6.81
246	5.95±1.89	6.86	7.45±1.76	6.99	7.67±2.06	7.36	6.48±0.69	6.44	9.39±1.37	9.46
247	10.13±3.03	8.01	7.50±2.03	8.01	7.47±2.12	8	-	5.82	6e+01±8e+02	5.8
248	6.21±0.90	3.05	6.06±0.93	3.05	2.04±0.37	3.04	-	2.22	1e+02±5e+02	2.21
249	19.02±4.90	14.72	13.35±4.58	14.88	13.04±4.61	15.33	12.68±2.45	12.45	15.81±11.01	16.19
261	12.42±1.71	10	11.08±1.19	10.85	10.54±1.15	13.32	44.02±5.16	16.25	68.54±37.24	39.11
262	4.13±1.01	3.95	3.41±0.86	4.02	4.46±0.98	4.21	4.42±1.00	3.61	4.19±1.77	5.09

263	4.34±1.02	3.83	2.82±0.92	3.94	3.62±0.99	4.24	4.41±0.40	3.96	5.97±0.66	6.32
264	3.72±0.64	2.63	2.74±0.17	2.76	2.15±0.64	3.13	3.61±0.62	3.36	6.54±1.73	6.35
265	5.19±0.74	4.39	4.70±0.24	4.4	4.24±0.26	4.41	2.36±0.33	3.25	3.97±0.50	3.39
266	3.87±2.36	4.23	4.33±1.50	4.25	3.80±1.74	4.33	4.55±1.68	3.35	2.21±3.77	4.01
267	2.42±0.91	2.33	2.51±0.62	2.45	2.67±0.58	2.78	2.98±0.31	2.98	5.60±0.88	5.58
268	1.91±1.30	1.76	1.71±0.56	1.86	1.88±0.54	2.14	2.49±0.22	2.37	4.33±0.55	4.55
269	3.58±0.89	2.88	2.79±0.55	3.11	2.75±0.65	3.77	5.15±0.51	4.63	10.56±1.00	10.5
270	3.14±1.17	2.43	2.49±0.64	2.58	3.31±0.76	3.01	2.47±0.80	3.45	10.03±2.83	7
271	4.01±1.09	4.05	4.21±0.36	4.09	4.28±0.35	4.22	3.22±0.30	3.45	4.64±0.37	4.52
272	3.78±0.96	3.47	3.58±0.50	3.64	3.81±0.88	4.13	4.65±0.61	4.44	8.51±1.10	8.65
273	5.49±1.11	4.7	4.98±0.65	4.86	5.17±0.75	5.31	4.97±0.42	5.19	9.84±1.50	9.01
274	4.70±0.72	3.19	3.91±0.55	3.19	4.02±0.64	3.19	1.91±0.20	2.33	2.43±0.85	2.31
275	2.74±1.05	2.61	2.74±0.45	2.67	2.57±0.62	2.85	2.67±0.42	2.59	3.97±0.53	4
276	3.77±0.38	1.41	1.47±0.48	1.48	1.52±0.07	1.66	1.98±0.13	1.75	3.20±0.24	3.23
277	1.67±0.24	1.75	1.11±0.57	1.89	1.32±0.69	2.31	2.97±0.32	2.88	6.01±1.00	6.27
278	1.95±0.24	1.97	1.49±0.93	2.05	1.88±0.81	2.27	2.84±0.44	2.29	3.31±0.93	4
279	2.58±2.30	4.25	3.34±1.72	4.33	6.20±1.16	4.56	3.01±0.98	3.96	6.64±2.29	5.69
280	4.06±0.83	4.32	4.63±0.61	4.34	4.32±0.62	4.4	2.24±1.63	3.36	4.71±2.49	3.88
281	5.12±1.39	4.6	4.10±1.35	4.77	4.72±1.11	5.24	8.54±1.90	5.18	8.31±2.03	9.16
282	4.69±2.23	5.02	5.66±1.90	5.41	6.37±2.19	6.53	7.77±1.63	7.8	21.83±17.24	17.9
283	3.51±1.77	3.3	3.77±1.55	3.34	2.71±1.44	3.44	2.86±2.05	2.79	3.53±1.66	3.65
285	52.45±5.75	53.64	54.33±7.20	54.81	62.22±10.43	58.18	52.45±7.52	51.26	69.30±37.55	82.72

Table 4: $R_{1\rho}$ $^{13}C\alpha$. Error is given for each measurement, and the calculated value is given after each measurement. The MAS frequency for each measurement is given in parenthesis.

Resi.	**$\omega_1/2\pi$= 9.3 kHz $\omega_r/2\pi$= 60 kHz**	**Calc.**	**$\omega_1/2\pi$= 17.5 kHz $\omega_r/2\pi$= 60 kHz**	**Calc.**	**$\omega_1/2\pi$= 35.0 kHz $\omega_r/2\pi$= 60 kHz**	**Calc.**	**$\omega_1/2\pi$= 47.7 kHz $\omega_r/2\pi$= 60 kHz**	**Calc.**	**$\omega_1/2\pi$= 25.0 kHz $\omega_r/2\pi$= 40 kHz**	**Calc.**
223	128.7±14.6	38.9	35.0±7.3	38.9	35.2±4.9	39.2	788.2±624.2	38.8	35.8±4.0	38.7
224	10.9±1.0	7.7	7.7±0.6	8.1	15.9±0.8	13.9	8.9±0.7	9.9	11.4±0.7	12.9

225	20.9±1.3	21	21.3±1.1	21.5	29.7±1.5	31.7	25.5±2.9	24.8	32.8±1.7	29.9
226	3.1±0.4	3.2	3.3±0.3	3.4	6.5±0.5	6.5	5.3±0.6	4.5	6.0±0.3	6.1
227	3.2±0.4	3.5	3.5±0.4	3.7	6.6±0.3	7	5.4±0.4	4.9	6.8±0.2	6.5
228	2.6±0.3	2.6	2.8±0.3	2.7	4.8±0.2	4.9	3.3±0.3	3.6	4.9±0.2	4.7
229	4.7±0.3	5	5.2±0.5	5.3	11.5±0.6	11.4	8.0±0.4	7.3	10.0±0.4	10.4
230	6.6±0.4	7.1	7.5±0.4	7.2	7.4±0.4	8.3	8.0±0.5	7.6	8.8±0.3	8.3
231	7.6±0.7	6.1	5.2±0.6	6.1	6.7±0.7	6.1	7.2±0.4	6.1	5.1±0.4	6.1
232	8.3±0.5	8.3	8.0±0.4	8.5	9.4±0.4	10.7	10.0±0.5	9.4	12.4±0.5	10.6
233	1.9±0.3	2.1	1.8±0.2	2.3	4.1±0.4	5.1	4.1±0.2	3.3	4.9±0.2	4.9
234	8.1±0.8	8.5	9.8±0.7	8.7	11.5±0.5	11.7	8.7±0.6	9.7	11.6±0.4	11.3
235	5.0±0.5	5.5	5.9±0.5	5.8	10.5±0.7	11.3	8.5±0.7	7.5	10.7±0.6	10.4
236	10.2±1.4	8	5.8±1.7	8.3	12.5±0.7	12.5	9.6±0.6	9.7	11.9±0.5	11.9
237	6.9±0.4	6.6	7.1±0.4	7	13.8±0.6	13.9	8.9±0.3	9.2	13.1±0.4	12.8
238	9.9±0.5	9.9	10.4±0.4	10.4	19.2±0.9	19.8	13.4±1.4	13.4	18.9±0.8	18.2
239	3.1±0.5	3.4	3.5±0.4	3.7	7.3±0.6	7.7	5.7±0.5	5.1	7.1±0.4	7.1
240	3.1±0.4	2.8	2.7±0.4	3.1	7.0±0.4	6.7	4.8±0.5	4.3	5.6±0.5	6.2
241	3.7±0.4	3.7	3.6±0.4	3.8	5.5±0.4	5.2	5.7±0.5	4.4	4.5±0.4	5.2
242	8.5±0.4	9	9.4±0.4	9.4	14.9±0.6	15.6	12.4±0.4	11.3	14.3±0.5	14.5
242	29.9±1.8	30.4	33.4±1.9	31.8	113.0±9.7	57.7	39.4±1.9	40.2	51.3±1.7	53.2
243	5.6±0.4	6.2	7.3±0.5	6.5	9.8±0.6	9.5	8.0±0.4	7.6	8.9±0.3	9.2
244	7.7±1.7	8.1	11.3±1.9	8.1	5.5±1.0	8	11.7±1.0	8	7.0±0.7	8
245	3.3±0.9	4.4	4.5±1.0	4.8	17.2±3.0	13.1	6.2±1.0	7.5	12.9±1.6	11.7
246	12.8±0.5	11.8	11.3±0.7	12.7	27.3±1.3	28.2	17.0±0.8	17.7	26.0±0.6	25.5
247	20.9±1.4	18.9	19.0±1.4	19.9	39.1±2.9	37.4	24.4±1.0	25.5	35.4±1.9	34.4
248	24.5±1.7	20.8	19.2±1.5	21.8	37.5±6.1	39.1	26.6±1.8	27.4	42.1±5.1	36.1
249	21.9±2.1	21	28.9±3.1	22.1	40.5±3.4	42.3	24.9±1.5	28.6	45.1±2.8	38.7
260	11.8±5.8	16.3	18.9±5.2	16.9	26.5±2.4	27.4	21.3±5.8	20.3	25.3±2.4	25.6
261	26.1±3.6	16.9	23.2±3.7	16.9	48.8±15.9	17	33.9±7.6	16.9	15.7±0.8	16.8
262	3.8±0.8	3.3	3.0±0.7	3.4	4.2±0.8	4.5	4.2±0.8	3.9	4.7±0.8	4.5
263	3.5±0.4	4.3	4.4±0.4	4.2	3.1±0.3	4.2	6.2±0.3	4.2	3.5±0.3	4.2
264	3.8±0.3	3.7	4.0±0.6	3.9	6.6±0.5	6.4	4.4±0.6	4.8	6.1±0.3	6.1
265	-5.4±6.7	6.8	2.9±8.8	7.1	9.3±0.9	10.1	15.5±28.0	8.2	10.5±0.7	9.8
266	9.6±0.5	10.1	9.9±0.6	10.3	14.0±0.5	14.3	12.0±0.3	11.6	13.6±0.3	13.6

267	4.5±0.2	4.9	5.4±0.2	5	6.4±0.4	5.8	5.7±0.3	5.3	5.6±0.2	5.9
268	8.5±0.3	9	9.5±0.4	9.1	10.0±0.4	9.7	9.7±0.4	9.3	9.4±0.3	9.7
269	6.2±0.2	6.4	6.8±0.2	6.5	9.2±0.3	9.1	7.5±0.4	7.5	8.6±0.3	8.8
269	8.2±0.8	8.4	7.8±0.5	8.7	12.7±1.0	14.1	13.6±0.7	10.4	12.7±0.7	13.2
270	8.8±0.7	9.1	9.9±0.9	9.5	15.5±1.0	16.8	11.5±1.5	11.8	16.4±0.7	15.6
271	5.8±0.2	5.6	5.7±0.1	5.9	10.2±0.2	10	7.3±0.2	7.3	9.0±0.2	9.4
272	8.4±0.3	8.8	9.1±0.4	9.1	14.0±0.5	14.7	12.0±0.4	10.9	14.1±0.4	13.8
273	7.3±0.3	7.6	8.1±0.5	7.8	12.4±0.6	12.6	10.1±0.5	9.4	11.8±0.3	11.8
274	10.5±0.7	9.8	9.7±0.9	9.8	9.6±1.1	9.8	10.6±0.9	9.8	8.6±0.8	9.8
275	5.6±0.4	5.9	6.3±0.4	5.9	7.0±0.5	5.8	7.2±0.6	5.9	4.8±0.3	5.8
276	2.4±0.3	2.5	3.3±0.4	2.6	5.7±0.3	5	3.2±0.4	3.5	4.0±0.3	4.8
277	2.9±0.4	3.3	3.8±0.4	3.5	5.6±0.6	5.8	4.4±0.4	4.3	5.6±0.5	5.5
278	6.5±0.3	6.8	7.1±0.4	7.1	11.5±0.5	11.8	9.3±0.4	8.6	11.0±0.4	11
278	5.8±0.4	5.5	5.8±0.4	5.7	8.6±0.5	8.8	5.4±0.6	6.8	9.5±0.7	8.4
279	7.4±0.8	7.9	8.1±1.2	8	11.5±0.8	9.3	9.7±0.9	8.5	8.2±0.5	9.2
280	5.0±0.4	5.5	5.6±0.6	5.7	8.4±0.4	8.8	7.7±0.5	6.8	8.7±0.4	8.4
281	8.6±1.9	0.1	7.1±1.8	0.1	-13.7±1.2	0.1	44.7±302.4	0.1	-5.9±1.5	0.1
282	16.8±0.5	17.4	19.0±0.7	18.3	33.8±1.5	34.5	26.0±1.3	23.5	31.1±1.3	31.7
282	25.4±4.4	23.9	30.7±5.5	24.1	27.6±2.6	27.1	22.0±2.6	25	27.7±4.3	26.4
283	35.2±4.1	31.3	25.8±4.0	31.4	30.0±2.8	32.2	43.1±8.2	31.7	37.6±4.9	32.2
284	7.3±1.6	8.2	12.7±4.3	8.4	10.5±1.5	11.3	9.2±4.0	9.5	12.9±2.0	11
286	14.6±0.9	14.5	15.4±0.7	15.1	26.2±2.5	26.2	16.7±1.3	18.7	26.7±2.3	24.3
287	29.4±8.1	24.9	24.1±6.7	26.2	81.9±34.3	48.7	36.8±49.3	33.5	33.0±18.0	44.9

Table 5: $^{1}H–^{15}N$ Order Parameters for the 3-Timescale Model. Upper and Lower bounds of the 68% confidence interval are given in parenthesis. Note that we give S and not S^2.

Resi.	S_f	S_i	S_s
224	0.909 (+0.006/-0.007)	1.000 (+0.000/-0.000)	0.9984 (+0.0003/-0.0002)
226	0.912 (+0.012/-0.015)	0.976 (+0.006/-0.006)	0.9980 (+0.0009/-0.0009)
227	0.936 (+0.007/-0.007)	0.984 (+0.002/-0.002)	0.9997 (+0.0001/-0.0002)
228	0.923 (+0.008/-0.010)	0.982 (+0.007/-0.003)	0.9999 (+0.0001/-0.0003)
229	0.931 (+0.006/-0.009)	0.961 (+0.002/-0.004)	0.9998 (+0.0001/-0.0001)
230	0.913 (+0.005/-0.007)	0.973 (+0.003/-0.003)	0.9983 (+0.0002/-0.0003)
231	0.901 (+0.005/-0.006)	0.999 (+0.001/-0.001)	0.9989 (+0.0005/-0.0005)
232	0.916 (+0.009/-0.009)	0.987 (+0.002/-0.002)	0.9982 (+0.0002/-0.0002)
233	0.917 (+0.007/-0.010)	0.988 (+0.005/-0.007)	0.9984 (+0.0007/-0.0005)
234	0.918 (+0.007/-0.007)	0.989 (+0.001/-0.000)	0.9963 (+0.0002/-0.0002)
235	0.939 (+0.008/-0.006)	0.970 (+0.002/-0.002)	0.9989 (+0.0002/-0.0002)
236	0.940 (+0.010/-0.014)	0.972 (+0.015/-0.006)	1.0000 (+0.0000/-0.0023)
237	0.947 (+0.006/-0.005)	0.965 (+0.004/-0.003)	0.9997 (+0.0001/-0.0001)
238	0.946 (+0.010/-0.006)	0.973 (+0.002/-0.001)	1.0000 (+0.0000/-0.0001)
239	0.929 (+0.009/-0.009)	0.995 (+0.001/-0.001)	0.9956 (+0.0006/-0.0006)
240	0.926 (+0.004/-0.001)	0.989 (+0.001/-0.000)	0.9996 (+0.0000/-0.0000)
241	0.919 (+0.007/-0.003)	0.995 (+0.001/-0.001)	0.9995 (+0.0001/-0.0001)
242	0.927 (+0.003/-0.000)	0.981 (+0.002/-0.000)	1.0000 (+0.0000/-0.0000)
243	0.928 (+0.013/-0.014)	0.990 (+0.002/-0.005)	0.9991 (+0.0004/-0.0004)
244	0.923 (+0.007/-0.006)	0.999 (+0.001/-0.001)	0.9971 (+0.0008/-0.0007)
245	0.919 (+0.008/-0.007)	0.981 (+0.003/-0.002)	0.9990 (+0.0002/-0.0002)
246	0.949 (+0.018/-0.021)	0.960 (+0.008/-0.002)	0.9984 (+0.0006/-0.0006)
247	0.923 (+0.018/-0.040)	0.948 (+0.043/-0.014)	1.0000 (+0.0000/-0.0087)
248	0.914 (+0.011/-0.022)	0.985 (+0.006/-0.004)	1.0000 (+0.0000/-0.0002)
249	0.997 (+0.003/-0.042)	0.903 (+0.032/-0.028)	0.9980 (+0.0020/-0.0029)
261	0.897 (+0.019/-0.011)	0.982 (+0.006/-0.017)	0.9877 (+0.0017/-0.0010)
262	0.928 (+0.012/-0.012)	0.971 (+0.005/-0.005)	0.9993 (+0.0004/-0.0006)
263	0.908 (+0.010/-0.007)	0.997 (+0.001/-0.007)	0.9988 (+0.0002/-0.0003)
264	0.915 (+0.009/-0.007)	0.990 (+0.003/-0.003)	0.9984 (+0.0004/-0.0005)
265	0.924 (+0.006/-0.003)	0.968 (+0.001/-0.001)	1.0000 (+0.0000/-0.0001)
266	0.932 (+0.017/-0.018)	0.980 (+0.006/-0.005)	0.9998 (+0.0002/-0.0013)
267	0.919 (+0.008/-0.008)	0.987 (+0.004/-0.003)	0.9987 (+0.0002/-0.0003)
268	0.922 (+0.008/-0.010)	0.989 (+0.002/-0.002)	0.9989 (+0.0001/-0.0002)
269	0.905 (+0.008/-0.008)	0.996 (+0.000/-0.000)	0.9969 (+0.0002/-0.0002)
270	0.925 (+0.005/-0.006)	0.990 (+0.004/-0.004)	0.9982 (+0.0005/-0.0004)
271	0.939 (+0.007/-0.007)	0.969 (+0.002/-0.002)	0.9996 (+0.0001/-0.0001)
272	0.897 (+0.006/-0.007)	0.999 (+0.001/-0.001)	0.9977 (+0.0004/-0.0004)
273	0.931 (+0.003/-0.007)	0.966 (+0.005/-0.004)	0.9980 (+0.0004/-0.0005)
274	0.944 (+0.002/-0.004)	0.965 (+0.001/-0.001)	1.0000 (+0.0000/-0.0000)
275	0.931 (+0.007/-0.006)	0.983 (+0.002/-0.001)	0.9994 (+0.0001/-0.0002)
276	0.928 (+0.004/-0.001)	0.993 (+0.000/-0.000)	0.9994 (+0.0000/-0.0000)

277	0.922 (+0.011/-0.016)	1.000 (+0.000/-0.001)	0.9983 (+0.0003/-0.0002)
278	0.904 (+0.008/-0.006)	0.990 (+0.001/-0.001)	0.9992 (+0.0002/-0.0002)
279	0.838 (+0.019/-0.009)	0.991 (+0.004/-0.004)	0.9990 (+0.0010/-0.0011)
280	0.948 (+0.011/-0.010)	0.966 (+0.006/-0.005)	0.9998 (+0.0002/-0.0007)
281	0.921 (+0.017/-0.015)	0.988 (+0.004/-0.005)	0.9979 (+0.0008/-0.0008)
282	0.921 (+0.025/-0.026)	0.988 (+0.007/-0.017)	0.9948 (+0.0023/-0.0011)
283	0.914 (+0.032/-0.020)	0.985 (+0.004/-0.004)	0.9997 (+0.0003/-0.0006)
285	0.930 (+0.019/-0.023)	0.886 (+0.024/-0.022)	0.9808 (+0.0081/-0.0098)

Table 6: $^{1}H–^{15}N$ Correlation times for the 3-Timescale Model. Upper and Lower bounds of the 68% confidence interval are given in parenthesis. The slow timescale is fixed at 14.7 μs.

Resi.	-log[τ_f (s)]	-log[τ_i (s)]
224	10.72 (+0.03/-0.09)	6.50 (+0.42/-0.00)
226	10.61 (+0.11/-0.10)	7.36 (+0.13/-0.11)
227	10.64 (+0.10/-0.06)	7.57 (+0.07/-0.04)
228	10.39 (+0.07/-0.07)	7.27 (+0.07/-0.19)
229	10.53 (+0.12/-0.03)	7.46 (+0.03/-0.03)
230	10.60 (+0.05/-0.03)	7.51 (+0.05/-0.03)
231	10.36 (+0.03/-0.02)	6.50 (+0.21/-0.00)
232	10.71 (+0.05/-0.04)	7.47 (+0.08/-0.06)
233	10.77 (+0.11/-0.07)	7.67 (+0.18/-0.17)
234	10.72 (+0.27/-0.07)	8.14 (+0.24/-0.17)
235	10.30 (+0.06/-0.04)	7.57 (+0.02/-0.03)
236	10.60 (+0.15/-0.10)	7.37 (+0.28/-0.10)
237	10.28 (+0.06/-0.05)	7.40 (+0.02/-0.05)
238	10.67 (+0.06/-0.14)	7.45 (+0.02/-0.02)
239	10.95 (+0.05/-0.11)	8.21 (+0.10/-0.14)
240	11.00 (+0.00/-0.02)	7.40 (+0.02/-0.01)
241	11.00 (+0.00/-0.04)	7.32 (+0.04/-0.03)
242	10.65 (+0.03/-0.03)	7.28 (+0.02/-0.00)
243	11.00 (+0.00/-0.05)	7.44 (+0.22/-0.14)
244	11.00 (+0.00/-0.03)	6.51 (+0.34/-0.01)
245	10.67 (+0.05/-0.05)	7.15 (+0.05/-0.07)
246	11.00 (+0.00/-0.48)	7.38 (+0.05/-0.07)
247	10.51 (+0.49/-0.09)	7.28 (+1.20/-0.11)
248	10.03 (+0.08/-0.07)	7.19 (+0.09/-0.51)
249	9.00 (+2.00/-0.00)	7.41 (+0.14/-0.19)
261	11.00 (+0.00/-0.29)	8.05 (+0.29/-0.32)
262	10.84 (+0.16/-0.19)	7.44 (+0.08/-0.06)
263	10.44 (+0.03/-0.03)	6.55 (+0.51/-0.05)
264	10.49 (+0.07/-0.06)	7.47 (+0.15/-0.14)
265	10.73 (+0.02/-0.04)	7.34 (+0.02/-0.04)
266	10.71 (+0.18/-0.18)	7.20 (+0.16/-0.12)
267	10.79 (+0.08/-0.07)	7.59 (+0.16/-0.11)
268	10.76 (+0.08/-0.06)	7.74 (+0.22/-0.12)

269	10.79 (+0.06/-0.05)	8.14 (+0.12/-0.09)
270	10.37 (+0.04/-0.04)	7.71 (+0.10/-0.17)
271	10.20 (+0.05/-0.06)	7.44 (+0.02/-0.02)
272	10.15 (+0.04/-0.04)	6.50 (+0.13/-0.00)
273	10.37 (+0.05/-0.04)	7.57 (+0.07/-0.06)
274	10.73 (+0.03/-0.10)	7.54 (+0.02/-0.02)
275	11.00 (+0.00/-0.08)	7.43 (+0.04/-0.05)
276	11.00 (+0.00/-0.01)	7.48 (+0.02/-0.00)
277	10.72 (+0.28/-0.02)	8.22 (+0.27/-0.07)
278	11.00 (+0.00/-0.03)	7.44 (+0.04/-0.03)
279	11.00 (+0.00/-0.03)	6.82 (+0.18/-0.29)
280	10.80 (+0.20/-0.21)	7.42 (+0.04/-0.09)
281	10.70 (+0.15/-0.13)	7.13 (+0.17/-0.24)
282	10.63 (+0.37/-0.26)	7.85 (+0.50/-0.43)
283	11.00 (+0.00/-0.02)	7.17 (+0.13/-0.08)
285	9.90 (+0.10/-0.13)	6.91 (+0.10/-0.11)

Table 7: ^{1}Hα–^{13}Cα Order Parameters for the 3-Timescale Model. Upper and Lower bounds of the 68% confidence interval are given in parenthesis. Note that we give S and not S^2.

Resi.	S_f	S_i	S_s
223	0.820 (+0.039/-0.035)	0.973 (+0.005/-0.018)	1.0000 (+0.0000/-0.0003)
224	0.886 (+0.018/-0.018)	0.914 (+0.012/-0.006)	0.9987 (+0.0001/-0.0001)
225	0.891 (+0.016/-0.026)	0.979 (+0.007/-0.003)	0.9981 (+0.0003/-0.0002)
226	0.799 (+0.057/-0.049)	0.972 (+0.016/-0.002)	0.9992 (+0.0001/-0.0001)
227	0.858 (+0.011/-0.020)	0.942 (+0.005/-0.009)	0.9992 (+0.0000/-0.0000)
228	0.841 (+0.008/-0.019)	0.985 (+0.005/-0.004)	0.9995 (+0.0001/-0.0000)
229	0.847 (+0.015/-0.014)	0.982 (+0.010/-0.010)	0.9987 (+0.0000/-0.0001)
230	0.884 (+0.013/-0.015)	0.931 (+0.006/-0.008)	0.9998 (+0.0001/-0.0000)
231	0.890 (+0.010/-0.000)	0.986 (+0.003/-0.000)	1.0000 (+0.0000/-0.0000)
232	0.916 (+0.008/-0.004)	0.956 (+0.003/-0.009)	0.9995 (+0.0000/-0.0001)
233	0.892 (+0.018/-0.021)	0.988 (+0.004/-0.006)	0.9994 (+0.0000/-0.0000)
234	0.915 (+0.011/-0.013)	0.886 (+0.017/-0.017)	0.9993 (+0.0001/-0.0001)
235	0.837 (+0.016/-0.014)	0.969 (+0.007/-0.007)	0.9987 (+0.0001/-0.0001)
236	0.931 (+0.017/-0.026)	0.919 (+0.018/-0.018)	0.9991 (+0.0001/-0.0001)
237	0.939 (+0.018/-0.018)	0.885 (+0.006/-0.010)	0.9985 (+0.0000/-0.0001)
238	0.929 (+0.011/-0.005)	0.945 (+0.023/-0.012)	0.9982 (+0.0001/-0.0001)
239	0.903 (+0.025/-0.004)	0.998 (+0.001/-0.001)	0.9992 (+0.0000/-0.0001)
240	0.905 (+0.017/-0.017)	0.988 (+0.007/-0.009)	0.9993 (+0.0001/-0.0000)
241	0.929 (+0.024/-0.026)	0.955 (+0.010/-0.015)	0.9997 (+0.0001/-0.0001)
242	0.795 (+0.016/-0.008)	0.962 (+0.007/-0.004)	0.9984 (+0.0001/-0.0001)
242	0.882 (+0.018/-0.018)	0.910 (+0.023/-0.019)	0.9943 (+0.0005/-0.0005)
243	0.914 (+0.011/-0.027)	0.966 (+0.009/-0.009)	0.9994 (+0.0001/-0.0000)
244	0.978 (+0.005/-0.012)	0.931 (+0.006/-0.011)	1.0000 (+0.0000/-0.0000)
245	0.850 (+0.044/-0.012)	1.000 (+0.000/-0.025)	0.9983 (+0.0002/-0.0001)
246	0.889 (+0.019/-0.018)	0.923 (+0.013/-0.015)	0.9967 (+0.0002/-0.0001)

247	0.867 (+0.020/-0.027)	0.911 (+0.024/-0.021)	0.9960 (+0.0005/-0.0004)
248	0.771 (+0.056/-0.021)	0.985 (+0.008/-0.062)	0.9957 (+0.0007/-0.0008)
249	0.873 (+0.033/-0.008)	0.992 (+0.002/-0.054)	0.9962 (+0.0005/-0.0003)
260	0.750 (+0.021/-0.000)	0.750 (+0.026/-0.000)	0.9952 (+0.0015/-0.0004)
261	0.973 (+0.002/-0.009)	0.987 (+0.003/-0.016)	1.0000 (+0.0000/-0.0000)
262	0.898 (+0.044/-0.032)	0.952 (+0.011/-0.009)	0.9998 (+0.0001/-0.0001)
263	0.890 (+0.009/-0.000)	0.911 (+0.001/-0.000)	1.0000 (+0.0000/-0.0000)
264	0.861 (+0.027/-0.015)	0.948 (+0.010/-0.007)	0.9994 (+0.0001/-0.0000)
265	0.982 (+0.018/-0.146)	0.977 (+0.023/-0.093)	0.9995 (+0.0004/-0.0008)
266	0.885 (+0.009/-0.011)	0.894 (+0.013/-0.013)	0.9990 (+0.0000/-0.0001)
267	0.891 (+0.013/-0.006)	0.951 (+0.005/-0.004)	0.9998 (+0.0000/-0.0000)
268	0.858 (+0.006/-0.005)	0.933 (+0.005/-0.015)	0.9999 (+0.0001/-0.0001)
269	0.830 (+0.011/-0.018)	0.996 (+0.000/-0.000)	0.9994 (+0.0000/-0.0001)
269	0.804 (+0.029/-0.016)	0.960 (+0.013/-0.013)	0.9987 (+0.0001/-0.0002)
270	0.869 (+0.022/-0.014)	0.936 (+0.011/-0.011)	0.9984 (+0.0001/-0.0001)
271	0.824 (+0.004/-0.000)	0.980 (+0.000/-0.000)	0.9990 (+0.0000/-0.0000)
272	0.856 (+0.011/-0.012)	0.977 (+0.002/-0.006)	0.9988 (+0.0001/-0.0001)
273	0.869 (+0.013/-0.006)	0.961 (+0.003/-0.015)	0.9990 (+0.0001/-0.0000)
274	0.966 (+0.007/-0.003)	0.920 (+0.027/-0.023)	1.0000 (+0.0000/-0.0000)
275	0.874 (+0.005/-0.007)	0.968 (+0.014/-0.001)	1.0000 (+0.0000/-0.0000)
276	0.861 (+0.023/-0.025)	0.983 (+0.007/-0.007)	0.9995 (+0.0001/-0.0001)
277	0.826 (+0.029/-0.020)	0.956 (+0.009/-0.011)	0.9994 (+0.0001/-0.0001)
278	0.819 (+0.019/-0.007)	0.961 (+0.005/-0.005)	0.9988 (+0.0001/-0.0001)
278	0.898 (+0.014/-0.007)	0.943 (+0.005/-0.008)	0.9993 (+0.0001/-0.0001)
279	0.935 (+0.011/-0.007)	0.997 (+0.001/-0.001)	0.9998 (+0.0001/-0.0001)
280	0.897 (+0.017/-0.018)	0.922 (+0.013/-0.015)	0.9993 (+0.0000/-0.0000)
281	0.986 (+0.014/-0.020)	1.000 (+0.000/-0.013)	1.0000 (+0.0000/-0.0000)
282	0.853 (+0.016/-0.016)	0.935 (+0.011/-0.010)	0.9964 (+0.0002/-0.0003)
282	0.902 (+0.028/-0.015)	0.989 (+0.002/-0.025)	0.9995 (+0.0005/-0.0007)
283	0.915 (+0.031/-0.041)	0.899 (+0.036/-0.033)	0.9998 (+0.0002/-0.0007)
284	0.981 (+0.019/-0.089)	0.995 (+0.002/-0.031)	0.9995 (+0.0003/-0.0003)
286	0.810 (+0.040/-0.013)	0.992 (+0.001/-0.026)	0.9975 (+0.0004/-0.0002)
287	1.000 (+0.000/-0.062)	0.822 (+0.057/-0.072)	0.9952 (+0.0031/-0.0016)

Table 8: ^{1}Hα–^{13}Cα Correlation times for the 3-Timescale Model. Upper and Lower bounds of the 68% confidence interval are given in parenthesis. The slow timescale is fixed at 8.7 μs.

Resi.	-log[τ_f (s)]	-log[τ_i (s)]
223	9.90 (+0.13/-0.12)	6.58 (+0.33/-0.08)
224	10.62 (+0.07/-0.07)	8.07 (+0.05/-0.04)
225	9.89 (+0.12/-0.09)	6.93 (+0.26/-0.24)
226	10.68 (+0.07/-0.10)	8.07 (+0.13/-0.68)
227	10.60 (+0.09/-0.07)	8.38 (+0.06/-0.04)
228	10.88 (+0.07/-0.04)	7.88 (+0.08/-0.12)
229	10.36 (+0.06/-0.03)	7.85 (+0.13/-0.38)
230	10.35 (+0.03/-0.03)	7.84 (+0.02/-0.05)

231	10.29 (+0.02/-0.04)	7.18 (+0.02/-0.08)
232	10.15 (+0.04/-0.03)	7.65 (+0.09/-0.12)
233	10.59 (+0.17/-0.11)	8.73 (+0.27/-0.33)
234	9.91 (+0.04/-0.16)	8.07 (+0.04/-0.03)
235	10.65 (+0.05/-0.05)	7.87 (+0.05/-0.13)
236	10.11 (+0.12/-0.14)	8.01 (+0.08/-0.09)
237	10.23 (+0.11/-0.13)	8.45 (+0.02/-0.03)
238	9.75 (+0.12/-0.17)	7.89 (+0.06/-0.23)
239	10.39 (+0.04/-0.02)	7.24 (+0.13/-0.03)
240	10.36 (+0.12/-0.10)	8.29 (+0.32/-0.31)
241	10.20 (+0.21/-0.20)	8.06 (+0.09/-0.13)
242	10.66 (+0.02/-0.02)	7.53 (+0.10/-0.09)
242	10.16 (+0.05/-0.05)	7.49 (+0.06/-0.05)
243	10.18 (+0.08/-0.04)	7.75 (+0.08/-0.15)
244	9.88 (+0.39/-0.26)	7.82 (+0.07/-0.06)
245	10.37 (+0.37/-0.16)	6.51 (+2.49/-0.01)
246	10.28 (+0.08/-0.04)	8.04 (+0.10/-0.12)
247	10.07 (+0.03/-0.05)	7.71 (+0.07/-0.08)
248	10.28 (+0.06/-0.08)	6.78 (+0.71/-0.28)
249	9.96 (+0.05/-0.18)	6.67 (+0.87/-0.17)
260	11.00 (+0.00/-0.11)	7.96 (+0.04/-0.11)
261	9.27 (+0.42/-0.27)	6.76 (+0.35/-0.26)
262	10.86 (+0.14/-0.26)	8.08 (+0.08/-0.07)
263	10.40 (+0.02/-0.02)	8.13 (+0.01/-0.00)
264	10.64 (+0.09/-0.14)	8.19 (+0.07/-0.09)
265	9.00 (+2.00/-0.00)	7.58 (+1.42/-1.08)
266	10.18 (+0.03/-0.04)	7.93 (+0.02/-0.03)
267	10.19 (+0.04/-0.07)	7.87 (+0.06/-0.08)
268	10.37 (+0.02/-0.02)	7.66 (+0.03/-0.02)
269	10.89 (+0.03/-0.03)	6.70 (+0.06/-0.05)
269	10.71 (+0.06/-0.03)	7.58 (+0.12/-0.36)
270	10.43 (+0.06/-0.03)	7.87 (+0.03/-0.07)
271	11.00 (+0.00/-0.02)	7.53 (+0.01/-0.05)
272	10.23 (+0.03/-0.02)	7.40 (+0.11/-0.16)
273	10.45 (+0.06/-0.03)	7.70 (+0.06/-0.12)
274	9.48 (+0.24/-0.44)	7.79 (+0.11/-0.11)
275	10.52 (+0.04/-0.04)	7.53 (+0.14/-0.25)
276	10.61 (+0.12/-0.11)	8.12 (+0.13/-0.43)
277	10.92 (+0.08/-0.09)	8.14 (+0.05/-0.07)
278	10.73 (+0.03/-0.02)	7.71 (+0.05/-0.05)
278	10.59 (+0.08/-0.07)	8.02 (+0.04/-0.04)
279	9.83 (+0.07/-0.07)	6.50 (+0.08/-0.00)
280	10.22 (+0.06/-0.03)	8.17 (+0.07/-0.06)
281	9.41 (+1.59/-0.41)	6.55 (+2.45/-0.05)
282	10.47 (+0.06/-0.05)	7.59 (+0.05/-0.06)
282	10.30 (+0.07/-0.15)	6.50 (+0.55/-0.00)
283	10.00 (+0.14/-0.16)	7.33 (+0.15/-0.23)

284	10.10 (+0.83/-1.10)	6.76 (+0.86/-0.26)
286	10.45 (+0.03/-0.09)	6.68 (+0.73/-0.05)
287	9.00 (+2.00/-0.00)	7.97 (+0.17/-0.18)

5 MatLab Programs

```
%%%%%%%%%%%%%%% d2.m %%%%%%%%%%%%%%%%%%%
function [ out ] = d2( beta,mp,m )
%D2 Generates the rank 2 wigner rotation matrix elements for the d_{m',m}^l
%   matrix, for rank 2 tensors. This is called with the angle beta, and
%   returns the full 5x5 matrix, allowing a transformation from m' to m.
%   Note that matrix ordering is given such that m and m' run from +2 to -2
%
%   dl=d2(beta)
%
%   dl=d2(beta,mp,m)
%
%   A. Smith, Nov. 2012.
%   alsi@nmr.phys.chem.ethz.ch
%
%
%


if nargin==1
    out=[((1+cos(beta))/2)^2 (1+cos(beta))/2*sin(beta) ...
        sqrt(3/8)*sin(beta)^2 (1-cos(beta))/2*sin(beta) ...
        ((1-cos(beta))/2)^2;
        -(1+cos(beta))/2*sin(beta) cos(beta)^2-(1-cos(beta))/2 ...
        sqrt(3/8)*sin(2*beta) (1+cos(beta))/2-cos(beta)^2 ...
        (1-cos(beta))/2*sin(beta);
        sqrt(3/8)*sin(beta)^2 -sqrt(3/8)*sin(2*beta) ...
        (3*cos(beta)^2-1)/2 sqrt(3/8)*sin(2*beta) sqrt(3/8)*sin(beta)^2;
        -(1-cos(beta))/2*sin(beta) -cos(beta)^2+(1+cos(beta))/2 ...
        -sqrt(3/8)*sin(2*beta) -(1-cos(beta))/2+cos(beta)^2 ...
        (1+cos(beta))/2*sin(beta);
        ((1-cos(beta))/2)^2 -(1-cos(beta))/2*sin(beta) ...
        sqrt(3/8)*sin(beta)^2 -(1+cos(beta))/2*sin(beta) ...
        ((1+cos(beta))/2)^2];
elseif nargin==3
    test=m+1i*mp;
    switch test
```

```
case (2)+1i*(2)
    out=((1+cos(beta))/2).^2;
case (2)+1i*(1)
    out=(1+cos(beta))/2.*sin(beta);
case (2)+1i*(0)
    out=sqrt(3/8)*sin(beta).^2;
case (2)+1i*(-1)
    out=(1-cos(beta))/2.*sin(beta);
case (2)+1i*(-2)
    out=((1-cos(beta))/2).^2;
case (1)+1i*(2)
    out=-(1+cos(beta))/2.*sin(beta);
case (1)+1i*(1)
    out=cos(beta).^2-(1-cos(beta))/2;
case (1)+1i*(0)
    out=sqrt(3/8)*sin(2*beta);
case (1)+1i*(-1)
    out=(1+cos(beta))/2-cos(beta).^2;
case (1)+1i*(-2)
    out=(1-cos(beta))/2.*sin(beta);
case (0)+1i*(2)
    out=sqrt(3/8)*sin(beta).^2;
case (0)+1i*(1)
    out=-sqrt(3/8)*sin(2*beta);
case (0)+1i*(0)
    out=(3*cos(beta).^2-1)/2;
case (0)+1i*(-1)
    out=sqrt(3/8)*sin(2*beta);
case (0)+1i*(-2)
    out=sqrt(3/8)*sin(beta).^2;
case (-1)+1i*(2)
    out=-(1-cos(beta))/2.*sin(beta);
case (-1)+1i*(1)
    out=-cos(beta).^2+(1+cos(beta))/2;
case (-1)+1i*(0)
    out=-sqrt(3/8)*sin(2*beta);
case (-1)+1i*(-1)
    out=-(1-cos(beta))/2+cos(beta).^2;
case (-1)+1i*(-2)
    out=(1+cos(beta))/2.*sin(beta);
```

```
        case (-2)+1i*(2)
            out=((1-cos(beta))/2).^2;
        case (-2)+1i*(1)
            out=-(1-cos(beta))/2.*sin(beta);
        case (-2)+1i*(0)
            out=sqrt(3/8)*sin(beta).^2;
        case (-2)+1i*(-1)
            out=-(1+cos(beta))/2.*sin(beta);
        case (-2)+1i*(-2)
            out=((1+cos(beta))/2).^2;
    end
else
    error('d2 must have either 1 or 3 inputs')
end

end
%%%%%%%%%%%%%%%%%%%% end d2.m %%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%% dipolar_coup.m %%%%%%%%%%%%%%%%%%
function [ D ] = dipolar_coup( dist,spin1,spin2 )
%Returns the dipolar coupling between two spins (delta=2*b_{12}),
%given in Hz, from a distance in [nm] and the identity of the two spins.
%The definition of the coupling is half of the full width of a pake pattern
%for a heternuclear spin system.
%
%[ D ] = dipolar_coup( dist,spin1,spin2 )
%

hplank = 6.6260693e-34;               % Planck constant in J s
mue0 = 12.56637e-7;                   % Permeability of vacuum [T^2m^3/J]

gamma1=GyroRatio(spin1);             %Gyromagnetic ratio of spin 1 in MHz/T
gamma2=GyroRatio(spin2);             %Gyromagnetic ratio of spin 2 in MHz/T

D=hplank*2e39*mue0/(4*pi*dist^3)*gamma1*gamma2;
%%%%%%%%%%%%%%%%%%%% end dipolar_coup.m %%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%% exchange_prop.m %%%%%%%%%%%%%%%%%%
function [ U,L ] = exchange_prop( N,ex_mat,vr,nsteps,nprop,varargin )
%EXCHANGE_PROP Generates a propagator in Liouville space over the course of
%one rotor period for a system exchanging between multiple states. One must
%provide the number of states, the exchange matrix between the states
%(described below), the rotor frequency, the number of steps taken to
%calculate a rotor period, the number of samples in a rotor period, and the
%Hamiltonians for each state.
%
%   N:      Number of states
%   ex_mat: Exchange matrix. The rates in each element are as follows:
%                [  1->1  2->1  3->1  ...  N->1
%                   1->2  2->2  3->2  ...  N->2
%                   1->3  2->3  3->3  ...  N->3
%                   ...
%                   1->N  2->N  3->N  ...  N->N  ]
%           Note that the columns should always sum to zero, diagonal
%           elements should be negative, and all other elements positive or
%           zero.
%           (If a single number is given instead of a matrix, exchange
%           rates between each pair of states are set to that number).
%   vr:     Rotor frequency. Make sure units of vr and Hamiltonians match.
%           For static calculation, set to zero, then set the following
%           argument (nsteps) to the desired time step, and the argument
%           after that (nprop) is ignored.
%   nsteps: Number of steps to divide the rotor period into. Note that this
%           is the total number propagators calculated, but the number
%           returned may be less
%   nprop:  This is the number of propagators returned by the program,
%           which is less than or equal to nsteps (set to one for a
%           propagator for the whole rotor period)
%   H1..HN: Hamiltonian corresponding to each site. Each Hamiltonian needs
%           to be a cell with three elements, giving components of the
%           Hamiltonian rotating at different multiples of the rotor
%           frequency.
%           H{1}:   Components not modulated by rotor (n=0)
%           H{2}:   Components modulated by -1x rotor (n=1). n=-1 are
```

```
%                   obtained from complex conjugate of H{2}
%           H{3}:   Components modulated by -2x rotor (n=2). n=-2 are
%                   obtained from complex conjugate
%           If using static calculation (vr=0), one must still include all
%           three components. One may put the full Hamiltonian in H{1},
%           leaving H{2} and H{3} as zeros, but it is also ok to do the
%           calculations the usual way and populate H{2} and H{3}.
%   L:      One may optionally set the last argument to a Louiville
%           relaxation matrix, which needs have the dimension of the
%           Hamiltonian squared. This will be applied to all N sites (for
%           H1...HN. This matrix exchanges spin states rather than
%           orientations. For example:
%           T2 relaxation of spin I:
%               L_Iz=Ham2Super(Iz);
%               L=L_Iz^2*1/T2;
%           T1 relaxation of spin I (note that this also induces some T2
%           relaxation)
%               L_Ix=Ham2Super(Ix);
%               L_Iy=Ham2Super(Iy);
%               L_Iz=Ham2Super(Iz);
%               L=(Lx^2+Ly^2+Lz^2)/(2*T1);
%
%
%   Output is a cell, U, with nprop elements, such that the propagator for
%   one rotor period is U{N}*U{N-1}*...U{3}U{2}U{1}. Optionally, one may
%   also output the individual Liouville matrices, as a cell with nsteps
%   elements.
%
%       [ U,L ] = exchange_prop( N,ex_mat,vr,nsteps,nprop,H1,H2,...,HN )
%
%           if a Liouville relation matrix is included
%
%       [ U,L ] = exchange_prop( N,ex_mat,vr,nsteps,nprop,H1,H2,...HN,L)
%
%           or for vr=0 (specify time step instead of spinning speed)
%
%       [ U,L ] = exchange_prop( N,ex_mat,0,deltat,[],H1,H2,...,HN )
%
%   A. Smith, April 2015
%
```

```
%% Get the Hamiltonians
if length(varargin)>N
    %If there is an extra entry, then assume this is a Liouville matrix
    L=varargin{end};        %Store the Liouville matrix
    H=varargin(1:N);        %Store the N Hamiltonians
else
    %If the number of entries in varargin is N, then store as the Hamiltonians
    H=varargin;
end

%% Determine the Hilbert space dimensionality
ndim=size(H{1}{1},1);       %Length of a dimension of the Hamiltonians
ndim2=ndim^2;
%Length of a dimension of the resulting Liouville space

%% Generate the exchange Liouville matrix
if numel(ex_mat)==1
    %If only single element given, set all rates to that element
    ex=ones(N)*ex_mat;      %Set all elements to ex_mat;
    ex(1:(N+1):end)=0;      %Reset the diagonal to zeros
    ex(1:(N+1):end)=-sum(ex);
    %Set the diagonal to the negative sum of its column s.t. column sum=0
    L0=kron(ex,eye(ndim2)); %Expand to the size of the Liouville space
else
    L0=kron(ex_mat,eye(ndim2));
    %Expand to the size of the Liouville space
end

%% Add the relaxation matrix in if it exists
if exist('L','var')
    %Test if a Liouville matrix for spin-exchange was given
    for k=1:N
        %Loop over the N sites, placing Liouville matrix in for each site
        L0(1+(k-1)*ndim2:k*ndim2,1+(k-1)*ndim2:k*ndim2)=...
            L0(1+(k-1)*ndim2:k*ndim2,1+(k-1)*ndim2:k*ndim2)-L;
        %Add given Liouville matrix to L0
    end
end
```

```
%% Determine if spinning or static, setup time axis
if vr==0    %Check if not spinning
    deltat=nsteps;
    %deltat must be given explicitly. We using the nsteps argument for this
    nsteps=1;           %nsteps is necessarily 1 if no spinning
    nprop=1;          %Only a single propagator is returned if no spinning.
else
    deltat=1/vr/nsteps;
    %If spinning, then the deltat is the rotor period divided by nsteps
end
t=0:deltat:(deltat*(nsteps-1));
%the time axis is set from 0 to the end of the rotor period (unless static,
%in which case t=0)

%% Pre-allocate cell for propagators
U=cell(nprop,1);
%Pre-allocate a cell to store the propagators for one rotor period
if nargout==2
    Lout=cell(nsteps,1);
    %Pre-allocate cell to store Liouville matrices if a second output
    %requested
end



for k=1:nprop
    U{k}=eye(ndim2*N);      %Set initial propagator values to the identity
end
%% Propagate the Hamiltonian

for k=1:nsteps      %Loop over the number of steps
    index=floor((k-1)/nsteps*nprop)+1;
    %Index for separating the propagator into parts
    %All steps with the same index get
    %multiplied together.

    L=zeros(ndim2);     %Pre-allocate the Liouville matrix for this step

    for j=1:N
```

```
        Ham=H{j}{1}+conj(H{j}{2})*exp(1i*2*pi*vr*t(k))+...
            H{j}{2}*exp(-1i*2*pi*vr*t(k))+...
            conj(H{j}{3})*exp(1i*4*pi*vr*t(k))+...
            H{j}{3}*exp(-1i*4*pi*vr*t(k));
            %Calculate the Hamiltonian for site j and time t(k).
        L((j-1)*ndim2+1:j*ndim2,(j-1)*ndim2+1:j*ndim2)=Ham2Super(Ham);
            %Expand this Hamiltonian into Liouville space and place into
            %the full Liouvillian
    end

    L=1i*2*pi*L-L0;     %Add the exchange matrix to the full Liouvillian,
                        %and multiply coherent part by 1i*2*pi

    if nargout==2
        Lout{k}=L;      %If second output requested, store L to Lout{k}
    end

    U{index}=expm(-deltat*L)*U{index};
    %Calculate propagator with expm(-deltat*L)
    %and multiply by accumulated propagator for that index
end

if nargout==2
    L=Lout;     %Set L to Lout if second output requested
end


end


%%%%%%%%%%%%%%%%%%%% end exchange_prop.m %%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%% GyroRatio.m %%%%%%%%%%%%%%%%%%
function [ gyro,I ] = GyroRatio( spin )
%GYRORATIO Produces the gyromagnetic ratio of various spins in MHz/T
%   GyroMagneticRatio = GyroRatio('nuc')- where 'nuc'='1H','13C',etc
%   [gyro I] = GyroRatio( 'nuc' )
```

```
switch spin
    case {'1H','H'}
        gyro=42.576;
        I=1/2;
    case '2H'
        gyro=6.536;
        I=1;
    case '3He'
        gyro=-32.434;
        I=1/2;
    case '7Li'
        gyro=16.546;
        I=3/2;
    case {'13C','C'}
        gyro=10.705;
        I=1/2;
    case '14N'
        gyro=3.077;
        I=1;
    case {'15N','N'}
        gyro=-4.3156;
        I=1/2;
    case '17O'
        gyro=-5.772;
        I=5/2;
    case '19F'
        gyro=40.053;
        I=1/2;
    case '23Na'
        gyro=11.262;
        I=3/2;
    case '31P'
        gyro=17.235;
        I=1/2;
    case '129Xe'
        gyro=-11.777;
        I=1/2;
    case '79Br'
        gyro=10.7008;
```

```
        I=3/2;
    case '81Br'
        gyro=11.5347;
        I=3/2;
    case 'Zn67'
        gyro=2.6685318;
        I=5/2;
    case 'e'
        gyro=-28.0250e3;
        I=NaN;
    case 'e-'
        gyro=-28.0250e3;
        I=NaN;
    case '13C/1H'
        gyro=.251449530;
        I=NaN;
    case '1H/13C'
        gyro=1/.251449530;
        I=NaN;
    case '15N/1H'
        gyro=.101329118;
        I=NaN;
    case '1H/15N'
        gyro=1/.101329118;
        I=NaN;
    case '15N/13C'
        gyro=1/2.481513063204596;
    case '13C/15N'
        gyro=2.481513063204596;
    otherwise
        error('Nucleus not recognized!')
end

end



%%%%%%%%%%%%%%%%%%%% end GyroRatio.m %%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%% Ham2Super.m %%%%%%%%%%%%%%%%%%
function [ HS ] = Ham2Super( Ham )
%HAM2SUPER Takes a Hamiltonian, and generates its Hamiltonian
%Superoperator. Returns the same result as:
%
%   HS=kron(H,eye(size(H)))-kron(eye(size(H)),conj(H))
%
%However, the algorithm used here is much faster
%
%   HS = Ham2Super( Ham )
%
%
%   A. Smith

n=size(Ham,1);
n2=n^2;
HS=zeros(n2);
HS1=zeros(n2);
Hamt=transpose(Ham);

for k=1:n
    HS(k:n:n2,k:n:n2)=Ham;
    HS1((n*(k-1)+1):(n*k),(n*(k-1)+1):(n*k))=Hamt;
end
HS=HS-HS1;

end


%%%%%%%%%%%%%%%%%%%% end Ham2Super.m %%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%% Hflip_ind_R1p.m %%%%%%%%%%%%%%%%%%
% This script calculates the time trace for R1p relaxation induced by
% inversion of the coupled 1H.
```

```
%% Set Parameters
dHC=dipolar_coup(0.109,'1H','13C'); %1H-13C dipole coupling (Hz)

k0=20;          %1H inversion rate (s^-1)
vr=60e3;        %Rotor frequence (Hz)
v1=48e3;        %Spin-lock strength (Hz)

% Time axis parameters
np=6001;    %Number of points in the time axis
nr=10;      %Number of rotor periods for each time point


%% Setup the powder average
pwd=pwd_avg(8); %Generate the powder structure (structure with fields
                %alpha,beta,gamma,weight, N)
AHC=rotate_inter([dHC 0],pwd); %Calculate the n=0,1,2 components of the
                               %dipole coupling in the lab frame

%% Setup the spin system, detection and initial density operators
[I S]=n_spin_system;    %Generate spin matrices for two spin-1/2 system
detect=I.x(:)';         %Gives the detection operator in Liouville space
rho0=I.x(:);            %Gives the initial density op. in Liouville space
norm=1/(detect*rho0);   %Calculate a norm. constant (signal starts at 1)


%% Setup the Liouville relaxation matrix
% Here we calculate the relaxation matrix that induces inversion of the
% proton. The first term applies the superoperator [Sx,[Sx,sigma]], the
% second term applies the superoperator [Sy,[Sy,sigma]], and the third term
% applies [Sz,[Sz,sigma]]. The superoperator is scaled to the appropriate
% rate in the next step. Note the last term is not really importand here.
L00=Ham2Super(S.x)^2+Ham2Super(S.y)^2+Ham2Super(S.z)^2;

L0=L00*k0/2;        %Scale the relaxation superoperator by k0/2
                    %(2 accounts for relaxation from both Sx and Sy terms)
signal0=cell(1,pwd.N);  %Pre-allocate a cell to store signal from each
                        %crystal orientation.
```

```
%% Loop over the powder average
parfor k=1:pwd.N
    Ham=cell(1,3);
    %Pre-allocate cell to store n=0,1,2 components of the Hamiltonian
    for j=1:3
        Ham{j}=AHC(k,j)*I.z*S.z;   %#ok<*PFBNS>
        %Multiply the interaction strength by sqrt(3/2)*T_(2,0)
        %(SzIz for heternuclear dipole)
    end

    Ham{1}=v1*I.x;      %Add the spin-locking strength to n=0 component of
                        %Hamiltonian (non-rotating component)

    U=exchange_prop(1,1,vr,120,1,Ham,L0);
      %Generate the propagator for 1 rotor period
    [V,D]=eig(U{1}^nr);     %Get the eigenvalues of the propagator for nr
                            %rotor periods, as well as the eigenframe

    D=diag(D);      %Turn diagonal matrix, D, into a column vector

    rho_det=(transpose(detect*V).*(V\rho0)*norm)*ones(1,np);
    %The above line constructs the product of the initial density
    %matrix and the detection operator in the eigenbasis of the
    %propagator. It then repeats it for every point in the time
    %axis (all the columns are the same-> faster multiplication for
    %signal calculation)
    U1=[ones(size(detect,2),1) cumprod(D*ones(1,np-1),2)];
    %Because the propagator is diagonalized, we can calculate its
    %value at all time points by repeatedly multiplying it by
    %itself- as done in the cumprod.
    signal0{k}=sum(rho_det.*U1,1);
    %We take the product of the propagator with the initial density
    %matrix and the detection operator (all in the eigenframe of
    %the propagator). Technically, this is the dot product at each
    %time point. However, to all time points at once, we have to do
    %an elementwise multiplication followed by a summation down the
    %first dimension. (a*b=sum(transpose(a).*b,1))
end

%% Add up the different powder orientations
```

```
signal=zeros(1,np);
%We will sum up the signal for all orientations into this vector
for k=1:pwd.N
    signal=signal+signal0{k}*pwd.weight(k);
    %Add the signal from the kth orientation to the summed signal, using
    %the weighting factor
end

%% Calculate time axis
deltat=1/vr*nr;         %Calculate the time step
t=0:deltat:(np-1)*deltat;   %Calculate the time axis

%% Plot the results
plot(t,signal)
axis([0 1 0 1])
xlabel('t / s')
ylabel('I')


%%%%%%%%%%%%%%%%%%%% end Hflip_ind_R1p.m %%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%% hop_and_1Hinvert.m %%%%%%%%%%%%%%%%%%
%This script simulates a two-site hopping model in addition to changes in
%R1p as induced by stochastic 1H inversion


%% Parameters
tc=7.2e-6;      %Correlation time of motion (s)
angle=2.2;      %Angle of hop in degrees

k0=20;          %1H flip rate (s^-1)

vr=60e3;        %MAS frequency (Hz)
v1=48e3;        %Spin-lock field strength (Hz)

dHC=dipolar_coup(.109,'1H','13C');  %1H-13C dipolar coupling
```

```
%% Setup spin matrices
[I S]=n_spin_system;        %Spin system

rho0=[I.x(:);I.x(:)];       %Initial density operator in Liouville space
detect=[I.x(1:end) I.x(1:end)]; %Detection operator in Liouville space
norm=1/(detect*rho0);   %Normal intensity so initial value is 1

rho00=I.x(:);           %Same as above, but for 1H flip only model,
                        %where we only have 1 hamiltonian
detect0=I.x(1:end);
norm0=1/(detect0*rho00);

%% Setup powder average
pwd=pwd_avg(7); %Generate the powder average (structure with fields
                %alpha,beta,gamma,weight, N)
AHC1=rotate_inter([dHC 0],pwd);
%Calculate n=0,1,2 components of the dipole coupling in the lab frame
AHC2=rotate_inter([dHC 0],pwd,[0 angle 0]*pi/180);
%As above, but first we tilt the dipole coupling by the hop angle

%% Setup Liouville matrix
L0=k0/2*(Ham2Super(S.x)^2+Ham2Super(S.y)^2+Ham2Super(S.z)^2);
%Liouville relaxation matrix which will stochastically invert the 1H spins
%at a rate of k0

%% Pre-allocate signal vectors
signal0=cell(pwd.N,1);          %Signal with both two-site hop and 1H inversion
signal_hop0=cell(pwd.N,1);      %Signal with only two-site hop
signal_flip0=cell(pwd.N,1);     %Signal with only 1H inversion

%% Setup time axis
nr=100;      %Number of rotor periods per time point
deltat=nr/vr;   %Time step
maxt=0.12;       %Total time propagation
t=0:deltat:maxt;    %Time axis
np=length(t);       %Number of time points

%% Loop over the powder average
%Open a parallel pool for faster calculation (embarrasingly parallel)
```

```
parfor k=1:pwd.N
    Ham1=cell(1,3);     %Pre-allocate cell to store parts of Hamiltonian
                        %rotating at 0,1, and 2 times the rotor frequency
    Ham2=cell(1,3);     %As above, but the dipole of this Hamiltonian is
                        %tilted by the hop angle relative to Ham1
    for j=1:3
        Ham1{j}=AHC1(k,j)*S.z*I.z; %#ok<*PFBNS> %Multiply the interaction
                    %by sqrt(3/2)*T_(2,0) (SzIz for heternuclear dipole)
        Ham2{j}=AHC2(k,j)*S.z*I.z;  %Same as above, for tilted Hamiltonian
    end

    Ham1{1}=v1*I.x;
    %Add the spin-locking strength to the non-rotating part of the Hamilton
    Ham2{1}=v1*I.x;     %Same as above.

    kex=1/2/tc;      %Calculate the exchange rate (1/(2*tc))
    %% First, calculate result for combined hopping, 1H inversion
    U=exchange_prop(2,kex,vr,100,1,Ham1,Ham2,L0);
    %Calculate the propagator in Liouville space for one rotor period
     % Number of sites, exchange rate, spinning speed,
     % number of steps over rotor period,number of propagators returned for
     %each rotor period, Ham for site 1, Ham for site 2,Liouville matrix
    U=U{1}^nr;          %Calculate the propagator for nr rotor periods
    [V,D]=eig(U);       %Diagonalize the propagator (returns eigenvectors
                        %in V, eigenvalues in D (in diagonal square matrix)
    D=diag(D);    %Turn square matrix with eigenvalues into a single vector

    rho_det=(transpose(detect*V).*(V\rho0)*norm)*ones(1,np);
        %The above line constructs the product of the initial density
        %matrix and the detection operator in the eigenbasis of the
        %propagator. It then repeats it for every point in the time
        %axis (all the columns are the same-> faster multiplication for
        %signal calculation)

    U1=[ones(size(detect,2),1) cumprod(D*ones(1,np-1),2)];
        %Because the propagator is diagonalized, we can calculate its
        %value at all time points by repeatedly multiplying it by
        %itself- as done in the cumprod.

    signal0{k}=sum(rho_det.*U1,1);
```

```
        %We take the product of the propagator with the initial density
        %matrix and the detection operator (all in the eigenframe of
        %the propagator). Technically, this is the dot product at each
        %time point. However, to all time points at once, we have to do
        %an elementwise multiplication followed by a summation down the
        %first dimension. (a*b=sum(transpose(a).*b,1))

    %% Second, calculate result for only hopping
    %(same as above, but leave out Liouville matrix)
    U=exchange_prop(2,kex,vr,100,1,Ham1,Ham2);
    U=U{1}^nr;
    [V,D]=eig(U);
    D=diag(D);
    rho_det=(transpose(detect*V).*(V\rho0)*norm)*ones(1,np);
    U1=[ones(size(detect,2),1) cumprod(D*ones(1,np-1),2)];
    signal_hop0{k}=sum(rho_det.*U1,1);
    %% Finally, calculate result for only 1H inversion
    %(same as above, but only include one Hamiltonian)
    U=exchange_prop(1,kex,vr,100,1,Ham1,L0);
    U=U{1}^nr;
    [V,D]=eig(U);
    D=diag(D);
    rho_det=(transpose(detect0*V).*(V\rho00)*norm0)*ones(1,np);
    U1=[ones(size(detect0,2),1) cumprod(D*ones(1,np-1),2)];
    signal_flip0{k}=sum(rho_det.*U1,1);
end

%% Sum together the powder average
signal=zeros(size(t));
%Pre-allocate signal to sum up contributions from different powder elements
signal_hop=zeros(size(t));
signal_flip=zeros(size(t));

for k=1:pwd.N       %Loop over powder elements
    signal=signal0{k}*pwd.weight(k)+signal;
    %Add together, accounting for the weighting of each element
    signal_hop=signal_hop0{k}*pwd.weight(k)+signal_hop;
    signal_flip=signal_flip0{k}*pwd.weight(k)+signal_flip;
end
```

```
 %% Fit results to exponentials
ti0=2;  %First time point to be fitted
%(Initial drop in magnetization due to rot. resonance is omitted from fit)
fun0=@(X)X(1)*exp(-t(ti0:end)*X(2));
% Exponential function to fit data to exponential decay


fun=@(X)sum((real(signal(ti0:end))-fun0(X)).^2);  % Sum of squares function
[~,b]=min(abs(exp(-1)-real(signal)));
% Find the element closest to 1/e, use as initial guess for 1/R1p
fit=fminsearch(fun,real([1 min([1/t(b) 1/t(2)])]));
%Fit signal vector to exponential using simplex minimization
R1p=fit(2);                      %Store the rate
A=fit(1);

fun=@(X)sum((real(signal_hop(ti0:end))-fun0(X)).^2);
% Sum of squares function
[~,b]=min(abs(exp(-1)-real(signal_hop)));
% Find the element closest to 1/e, use as initial guess for 1/R1p
fit=fminsearch(fun,real([1 min([1/t(b) 1/t(2)])]));
%Fit signal vector to exponential using simplex minimization
R1p_hop=fit(2);                      %Store the rate
A_hop=fit(1);

fun=@(X)sum((real(signal_flip(ti0:end))-fun0(X)).^2);
% Sum of squares function
[~,b]=min(abs(exp(-1)-real(signal_flip)));
% Find the element closest to 1/e, use as initial guess for 1/R1p
fit=fminsearch(fun,real([1 min([1/t(b) 1/t(2)])]));
%Fit signal vector to exponential using simplex minimization
R1p_flip=fit(2);                      %Store the rate
A_flip=fit(1);


%% Plotting
figure(1)
clf(1)
scatter(t,signal,'MarkerEdgeColor','Red')
hold all
plot(t,A*exp(-t*R1p),'Color','Red')
```

```
scatter(t,signal_hop,'MarkerEdgeColor','Green')
plot(t,A_hop*exp(-t*R1p_hop),'Color','Green')
scatter(t,signal_flip,'MarkerEdgeColor','Blue')
plot(t,A_flip*exp(-t*R1p_flip),'Color','Blue')
axis([0 t(end) 0 1.1])
xlabel('t / s')
ylabel('I')
legend('Two-site hop+H invers.','fit','Two-site hop only',...
    'fit','H invers. only','fit')
text(.01,.3,['R_{1\rho}(both)=' num2str(R1p,'%3.2f') ' s^{-1}'])
text(.01,.2,['R_{1\rho}(hop)=' num2str(R1p_hop,'%3.2f') ' s^{-1}'])
text(.01,.1,['R_{1\rho}(flip)=' num2str(R1p_flip,'%3.2f') ' s^{-1}'])
title(['\tau_c=' num2str(tc*1e6) ' \mu s, S^2=' ...
    num2str(1/4*(3*cos(angle*pi/180)^2+1)) ...
    ', k(H-flip)=' num2str(k0) 's^{-1}'])
%%%%%%%%%%%%%%%%%%%% end hop_and_1Hinvert.m %%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%% LF_sim.m %%%%%%%%%%%%%%%%%%
% This script calculates R1p relaxation of a two-site hop in the laboratory
% frame, therefore being valid for all correlation times.
%
% Computationally intensive- we have run this on the Brutus computer
% cluster of the ETH, taking ~2.5 hours on 99 cores. The output is for
% comparison with the results of tc_v_MF.m
%

%% Setup parameters

tc=logspace(-2,-11,100);    %Correlation time to be simulated/calculated
dHC=dipolar_coup(.109,'1H','13C');  %1H-13C dipole coupling (Hz)

vr=60e3;        %Rotor frequency (Hz)
v1=35e3;        %Spin lock strength(Hz)
v0H=850e6;      %1H Larmor frequency
v0C=v0H*GyroRatio('13C/1H');    %13C Larmor frequency
```

```
angle=5;        %hop angle (degrees)

%% Calculate powder average, term sizes
pwd=pwd_avg(3); %powder average



AHC0=rotate_inter_full([dHC 0],pwd);
%Calculate components of the dipole coupling for first site
AHC1=rotate_inter_full([dHC 0],pwd,[0 angle 0]*pi/180);
%Calculate components of dipole coupling for second site

%% Calculate spin tensors
[I S]=n_spin_system;        %Generate the spin matrices
T10_I=I.z;                  %Calculate 1-spin (rank-1) spherical tensors
T10_S=S.z;
T11_I=-1/sqrt(2)*I.p;
T1m1_I=1/sqrt(2)*I.m;
T11_S=-1/sqrt(2)*S.p;
T1m1_S=1/sqrt(2)*S.m;

T=cell(5,1);            %Calculate 2-spin spherical tensors (rank-2)
T{3}=1/sqrt(6)*(2*T10_I*T10_S+T11_I*T1m1_S+T1m1_I*T11_S); %T_(2,0)
T{4}=1/sqrt(2)*(T11_I*T10_S+T10_I*T11_S);   %T_(2,1)
T{2}=1/sqrt(2)*(T1m1_I*T10_S+T10_I*T1m1_S); %T_(2,-1)
T{5}=T11_I*T11_S;   %T_(2,2)
T{1}=T1m1_I*T1m1_S; %T_(2,-2)

%% Setup time step, time axis
nsteps=16;          %Number of steps in one cycle of the RF irradiation
deltat=1/v0C/nsteps;
                    %Time step (calculated out of the C13 larmor frequency,
                    %since this is the frequency we irradiate at- RF
                    %now time dependent)

vr=1/(round(1/(vr*deltat*nsteps))*nsteps*deltat);
% Round the rotor frequency so that deltat fits exactly into 1 rotor period

t=0:deltat:1/vr;                    % Time axis for one rotor period
t=t(1:end-1);                       % Discard last time point (same as t=0)
```

```
rps=100;        %Rotor cycles per sample
ns=1e3;         %Number of samples
t1=0:rps/vr:((ns-1)*rps/vr);    %Time axis for full propagation

%% Setup initial density operator and detection operator
rho0=[S.x(:);S.x(:)];          %Initial density operator in Liouville space
detect=[S.x(1:end) S.x(1:end)]; %Detection operator in Liouville space
norm=1/trace(rho0'*rho0);       %Normalization constant


%% Preallocation, setup
signal0=cell(pwd.N,1);       %Cell to store signal for each powder element

%% Setup exchange matrix, RF field matrices, H0 matrix
LE=[-1 1;1 -1];      %Two site exchange matrix with correlation time of 0.5
LE=kron(LE,eye(16));
%Expand matrix into correct size for two-site Liouville space

Hrf=cell(nsteps,1);
%Generate cell to store all nsteps of the time-dependent RF Hamiltonian
for k=1:nsteps          %Loop over nsteps for calculating RF Hamiltonian
    Hrf{k}=v1*2*cos(2*pi*(k-1)/nsteps)*S.x;
    %Calculate RF Hamiltion for kth step (v1x2 for linear polarized irradiation)
    Hrf{k}=kron(Hrf{k},eye(4))-kron(eye(4),transpose(Hrf{k}));
    %Expand into Liouville space for 1-site
    Hrf{k}=kron(eye(2),Hrf{k});
    %Expand again for two-site hop
end

H0=v0H*I.z+v0C*S.z;     %Calculate the Zeeman Hamiltonian
H0=kron(H0,eye(4))-kron(eye(4),transpose(H0)); %Expand into Liouville space
H0=kron(eye(2),H0);         %Expand again for two-site hop

U00=cell(pwd.N,1);
%Pre-allocate cell to store propagators for each powder element
%% Begin powder loop

parfor k=1:pwd.N
    %Loop over powder average
```

```
%(parallel computing essential for reasonable computational time)
U=cell(1,length(tc));   %Pre-allocate propagator for each tc
for m=1:length(tc)
    %Set initial propagators to the identity for each correlation time
    U{m}=eye(32);
end

% Calculate Hamiltonian components. There are 25 separate components,
% accounting for products of 5 components in the rotor frame than are
% then transformed to each of 5 components in the lab frame.
Ham=cell(5,5);  %Pre-allocate cell to store all elements of Hamiltonian
for m=1:5       %Loop over rotor frame component (add 3 to m)
    for n=1:5   %Loop over lab frame component (add 3 to n)
        Ham0=AHC0(k,m,n)*T{n}; %#ok<*PFBNS>
        %Calculate dipole Hamiltonian for first site
        Ham1=AHC1(k,m,n)*T{n};
        %Calculate dipole Hamiltonian for second site
        Ham0=kron(Ham0,eye(4))-kron(eye(4),transpose(Ham0));
        %Expand into Liouville space
        Ham1=kron(Ham1,eye(4))-kron(eye(4),transpose(Ham1));
        %Expand into Liouville space
        Ham{m,n}=[Ham0 zeros(16);zeros(16) Ham1];
        %Store into single matrix for both sites
        %Note that these are technically not Hamiltonians anymore, but
        %rather are Liouville matrices
    end
end

for j=1:length(t)
    %Calculate propagator for each element in t (~57000 elements)
    L0=zeros(32);
    %Pre-allocate matrix to store the full coherent Liouville matrix
    for m=-2:2     %Loop over the rotor frame components
        for n=1:5   %Loop over the lab frame components
            L0=L0+Ham{m+3,n}*exp(-1i*2*pi*vr*m*t(j));
            %Add to Liouville matrix
        end
    end
    L0=L0+Hrf{mod(j-1,nsteps)+1};      %Add the RF Liouville matrix
    L0=L0+H0;                          %Add the Zeeman Liouville matrix
```

```
    for m=1:length(tc) %Loop over all correlation times
        %Note that L0 doesn't change fore different correlation times,
        %so it is more efficient to construct this matrix once and
        %re-use in the inner loop.
        U0=expm(deltat*(-1i*2*pi*L0+1/tc(m)/2*LE));
        %Calculate the propagator for t(j) and tc(m)
        U{m}=U0*U{m};     %Multiply by accumulated propagator for tc(m)
    end
end

U00{k}=U;
%After generating propagator for all correlation times and 1 rotor period,
%store it for use later

signal0{k}=zeros(length(tc),ns);
%Pre-allocate signal vector for each correlation time
%and each time point in t1

for m=1:length(tc)          %Loop over each correlation time
    [V,D]=eig(U{m}^rps);        %Calculate eigenvectors and eigenvalues
                                %for propagator over rps rotor periods
    D=diag(D);                  %Turn into a single column vector

    rho_det=(transpose(detect*V).*(V\rho0)*norm)*ones(1,size(t,2));
        %The above line constructs the product of the initial density
        %matrix and the detection operator in the eigenbasis of the
        %propagator. It then repeats it for every point in the time
        %axis (all the columns are the same-> faster multiplication for
        %signal calculation)

    U1=[ones(size(detect,2),1) cumprod(D*ones(1,size(t,2)-1),2)];
        %Because the propagator is diagonalized, we can calculate its
        %value at all time points by repeatedly multiplying it by
        %itself- as done in the cumprod.

    signal0{k}(m,:)=sum(rho_det.*U1,1);
        %We take the product of the propagator with the initial density
        %matrix and the detection operator (all in the eigenframe of
```

```
            %the propagator). Technically, this is the dot product at each
            %time point. However, to all time points at once, we have to do
            %an elementwise multiplication followed by a summation down the
            %first dimension. (a*b=sum(transpose(a).*b,1))
    end
    disp([k pwd.N])
end

signal=zeros(length(tc),ns);
%Pre-allocate signal to sum up contributions from different powder elements
for k=1:pwd.N   %Loop over powder elements
    signal=signal+signal0{k}*pwd.weight(k);
    %Add together, accounting for the weighting of each element
end

%% Fitting to exponential
fun0=@(X)X(1)*exp(-t1*X(2));   % Exponential function to fit data to exponential
decay

R1p=zeros(length(tc),1);    %Pre-allocate vector to store R1p values
for k=1:length(tc)  %Loop over all correlation times
    fun=@(X)sum((real(signal(k,:))-fun0(X)).^2);  % Sum of squares function
    [~,b]=min(abs(exp(-1)-real(signal(k,:))));
    % Find the element closest to 1/e, use as initial guess for 1/R1p
    fit=fminsearch(fun,[1 min([1/t1(b) 1/t1(2)])]);
    %Fit signal vector to exponential using simplex minimization
    R1p(k)=fit(2);
end


save LF_sim_results tc signal t1 R1p U00        %Save the results

%% Plotting
figure(1)
loglog(tc,R1p)
grid on
legend('Lab Frame Two-Site Hop')
```

```
%%%%%%%%%%%%%%%%%%%% end LF_sim.m %%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%% n_spin_system.m %%%%%%%%%%%%%%%%%%
function [ varargout ] = n_spin_system(varargin)
%N_SPIN_SYSTEM This function produces the spin matrices for a spin system
%of arbitrary size. The number of spins in the spin system will be
%determined by the number of output arguments. The input arguments are
%the spin quantum numbers of the individual spins. One does not need to
%specify all spin quantum numbers, but those that are unspecified will be
%assumed to be spin 1/2. Spin numbers may be listed as multiple inputs or
%as a single vector.
%
%   [S1 S2 S3 ...] = n_spin_system(1,3/2,2,...);
%
%   or if all spin-1/2, then
%
%   [S1 S2 S3 ...] = n_spin_system;
%
%Alternatively, one may use only one output, and list all spin quantum
%numbers,in which case all spin matrices will be put out in one cell. In
%this case, the number of spins is equal to the number of inputs. This may
%be useful for more general programming.
%
%   S = n_spin_system(1,3/2,2,...)
%
%For very large spin systems, one may request sparse matrices, one may
%specify a final entry 'sparse', so that the matrices are generated with
%sparse allocation. This works with both methods above.
%
%   S = n_spin_system(1/2,1/2)
%
%   A. Smith
%   alsi@nmr.phys.chem.ethz.ch
%

if nargin~=0&&ischar(varargin{end})&&strcmpi(varargin{end},'sparse')
    issparse=true;
```

```
    temp=cell(1,nargin-1);
    for k=1:nargin-1
        temp{k}=varargin{k};
    end
else
    temp=varargin;
    issparse=false;
end

if nargout==1&&nargin==0
    spins=1/2;
    nspins=1;
elseif nargout==1
    spins=cell2mat(temp);
    nspins=length(spins);
else
    nspins=nargout;
    spins=cell2mat(temp);
    spins=[spins 1/2*ones(1,nargout-length(spins))];
end

mat_sizes=2*spins+1;

out=cell(1,nspins);

if prod(mat_sizes)^2*nspins>5e12
    error('A system this large is likely to exceed the memory')
end

if prod(mat_sizes)^2*nspins>1.5e6||issparse
    for k=1:nspins
        out{k}.x=kron(speye(prod(mat_sizes(1:k-1))),...
            kron(sparse(Jx(spins(k))),speye(prod(mat_sizes(k+1:end)))));
        out{k}.y=kron(speye(prod(mat_sizes(1:k-1))),...
            kron(sparse(Jy(spins(k))),speye(prod(mat_sizes(k+1:end)))));
        out{k}.z=kron(speye(prod(mat_sizes(1:k-1))),...
            kron(sparse(Jz(spins(k))),speye(prod(mat_sizes(k+1:end)))));
        out{k}.p=out{k}.x+1i*out{k}.y;
        out{k}.m=out{k}.x-1i*out{k}.y;
        if spins(k)==1/2
```

```
            out{k}.alpha=speye(prod(mat_sizes))/2+out{k}.z;
            out{k}.beta=speye(prod(mat_sizes))/2-out{k}.z;
        end
    end

else

    for k=1:nspins
        out{k}.x=kron(eye(prod(mat_sizes(1:k-1))),...
            kron(Jx(spins(k)),eye(prod(mat_sizes(k+1:end)))));
        out{k}.y=kron(eye(prod(mat_sizes(1:k-1))),...
            kron(Jy(spins(k)),eye(prod(mat_sizes(k+1:end)))));
        out{k}.z=kron(eye(prod(mat_sizes(1:k-1))),...
            kron(Jz(spins(k)),eye(prod(mat_sizes(k+1:end)))));
        out{k}.p=out{k}.x+1i*out{k}.y;
        out{k}.m=out{k}.x-1i*out{k}.y;
        if spins(k)==1/2
            out{k}.alpha=eye(prod(mat_sizes))/2+out{k}.z;
            out{k}.beta=eye(prod(mat_sizes))/2-out{k}.z;
        end
    end

end


if nargout==1&&nspins~=1
    varargout{1}=out;
else
    varargout=out;
end


end
%% Subfunction Jm
function Out = Jm(j)
% General spin operator J_minus
% Input: Jm(j) with j = 1/2, 1, 3/2, ....
% With no input argument the j = 1/2 operator is calculated
% last change: tm, 16.07.03
```

```
if nargin == 0

      j = 1/2;

end

Mult = round(2*j+1);
Out = zeros(Mult,Mult);

for k =-j+1:j

      Out(k+j+1,k+j)= sqrt(j*(j+1) - k*(k-1));

end

end

%% Subfunction Jp

function Out = Jp(j)
% General spin operator J_plus
% Input: Jp(j) with j = 1/2, 1, 3/2, ....
% With no input argument the j = 1/2 operator is calculated
% last change: tm, 16.07.03

if nargin == 0

      j = 1/2;

end

Mult = round(2*j+1);
Out = zeros(Mult,Mult);

for k =-j:j-1

      Out(k+j+1,k+j+2)= sqrt(j*(j+1) - k*(k+1));

end
```

```
end

%% Subfunction Jx

function Out = Jx(j)
% General spin operator Jx
% Input: Jx(j) with j = 1/2, 1, 3/2, ....
% With no input argument the j = 1/2 operator is calculated
% last change: tm, 16.07.03

if nargin == 0

      j = 1/2;

end

Out = 0.5*(Jp(j)+Jm(j));

end

%% Subfunction Jy

function Out = Jy(j)
% General spin operator Jy
% Input: Jx(j) with j = 1/2, 1, 3/2, ....
% With no input argument the j = 1/2 operator is calculated
% last change: tm, 16.07.03

if nargin == 0

      j = 1/2;

end

Out = -0.5*1i*(Jp(j)-Jm(j));


end

%% Subfunction Jz
```

```
function Out = Jz(j)
% General spin operator Jz
% Input: Jx(j) with j = 1/2, 1, 3/2, ....
% With no input argument the j = 1/2 operator is calculated
% last change: tm, 16.07.03

if nargin == 0

      j = 1/2;

end

Mult = round(2*j+1);
Out = zeros(Mult,Mult);

for k =-j:j

      Out(k+j+1,k+j+1)= - k;

end



end


%%%%%%%%%%%%%%%%%%%% end n_spin_system.m %%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%% pwd_avg.m %%%%%%%%%%%%%%%%%%
function [ varargout ] = pwd_avg( q,sym )
%PWD_AVG Generates a powder average with quality q (q=1 to 8).
%According to JCP 59 (8) 3992 (1973)
%
%   pwd = pwd_avg(q);
%
%   or
```

```
%
%   [alpha beta gamma weight] = pwd_avg(q);
%
%Alternatively, one may use only alpha and beta averaging (gamma included,
%but always 0). This is triggered if the symmetry is provided (Ci, C2h,
%D2h, D4h), in which case a powder average over either a hemi-sphere, a
%quarter-sphere, 8th-sphere, or 16th sphere is calculated. In this usage, q
%indicates the number of contours for the powder average used.
%
%   pwd = pwd_avg(q,sym)
%
%   or
%
%   [alpha beta gamma weight] = pwd_avg(q,sym)
%
%Finally, one may produce only beta angles by specifying the second
%argument as 'beta' and the first argument as the number of desired
%angles (alpha,gamma set to zero).
%
%
% A. Smith
%

if nargin==2&&strcmp(sym,'beta')
    alpha=zeros(q,1);
    beta=linspace(0,pi/2,q+1)';
    beta=beta(2:end);
    gamma=zeros(q,1);
    weight=sin(beta);
    weight=weight/sum(weight);
    if nargout==1
        varargout{1}.alpha=alpha;
        varargout{1}.beta=beta;
        varargout{1}.gamma=gamma;
        varargout{1}.weight=weight;
        varargout{1}.N=q;
    else
        varargout{1}=alpha;
        varargout{2}=beta;
        varargout{3}=gamma;
```

```
        varargout{4}=weight;
    end
elseif nargin==2

    switch sym
        case {'Ci'}
            SymPhi = 1;
        case {'C2h'}
            SymPhi = 2;
        case {'D2h'}
            SymPhi = 4;
        case {'D4h'}
            SymPhi = 8;
        otherwise
            disp('Symmetry not recognized, use Ci')
            SymPhi =1;
    end

    np = ceil(q*sin((0.5:q - 0.5)*pi/(2*q)))/SymPhi;
    nF = sum(np);

    alpha=zeros(nF*4,1);
    beta=zeros(nF*4,1);
    gamma=zeros(nF*4,1);

    theta_j = 0;
    count = 1;

    for j = 1 : q,
        dtheta = acos(cos(theta_j) - np(j)/nF ) - theta_j;
        beta( count:count+4*np( j ) - 1) = theta_j + dtheta/2;
        dphi = pi/(2*SymPhi*np(j));
        alpha(count:count+4*np(j) - 1) =0.5*dphi:dphi:(4*np(j) - 0.5)*dphi;
        count = count + 4*np(j);
        theta_j = theta_j + dtheta;
    end;
    if nargout==1
        varargout{1}.alpha=alpha;
        varargout{1}.beta=beta;
        varargout{1}.gamma=gamma;
```

```
        varargout{1}.weight=ones(nF*4,1)/nF/4;
        varargout{1}.N=nF*4;
    else
        varargout{1}=alpha;
        varargout{2}=beta;
        varargout{3}=gamma;
        varargout{4}=ones(nF*4,1)/nF/4;
    end
else
    value1=[2 50 100 144 200 300 538 1154 3000 5000 7000 10000];
    value2=[1 7 27 11 29 37 55 107 637 1197 1083 1759];
    value3=[1 11 41 53 79 61 229 271 933 1715 1787 3763];

    count=1:(value1(q)-1);

    alpha=2*pi*mod(value2(q)*count,value1(q))/value1(q);
    beta=pi*count/value1(q);
    gamma=2*pi*mod(value3(q)*count,value1(q))/value1(q);

    weight=sin(beta);
    weight=weight/sum(weight);

    if nargout==1
        varargout{1}.alpha=alpha';
        varargout{1}.beta=beta';
        varargout{1}.gamma=gamma';
        varargout{1}.weight=weight';
        varargout{1}.N=length(count);
    else
        varargout{1}=alpha';
        varargout{2}=beta';
        varargout{3}=gamma';
        varargout{4}=weight';
    end
end

end

%%%%%%%%%%%%%%%%%%%% end pwd_avg.m %%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%% rotate_inter.m %%%%%%%%%%%%%%%%%%
function [ A ] = rotate_inter( A0,pwd,varargin )
%ROTATE_INT Input is a rank 2 interaction in its principle axis system
%(PAS) by given by its anisotropy and the asymmetry (A0=[delta eta]), a
%structure containing a powder average (elements alpha,beta,gamma), and
%optionally, euler angles for the interaction (more than one set may be
%given, in which case they will be executed in order, starting from the
%principle axis system (PAS)), and the rotor angle (if omitted, assumed to
%be acos(sqrt(1/3))). Returns a matrix containing constants such that for
%the kth element of the powder average, one may write the time dependent
%Hamiltonian for the interaction in the rotating frame (keeping only the
%T_(2,0) component of the interaction) as:
%
%  H = (conj(A(k,3))*exp(2*1i*omega_r*t)+conj(A(k,2))*exp(1i*omega_r*t)+...
%         A(k,1)+A(k,2)*exp(-1i*omega_r*t)+A(k,3)*exp(-2*1i*omega_r*t))*...
%         T_(2,0)*sqrt(3/2)
%
%Then, the term T_{2,0}*sqrt(3/2) depends on the interaction, for example:
%   Heteronuclear dipole:
%       T_(2,0)*sqrt(3/2) = Iz*Sz
%   Homonuclear dipole:
%       T_(2,0)*sqrt(3/2) = (I1z*I2z-1/2*(I1x*I2x+I1y*I2y))
%   Chemical shift anisotropy:
%       T_(2,0)*sqrt(3/2) = Iz
%
%We include multiplication by a factor of sqrt(2/3) in the calculation to
%try to reduce the number of constants required when constructing the
%Hamiltonians. This is to makes things easier for sloppy programmers and
%drive the careful ones nuts. One may choose to not perform the final
%tilting from the rotor frame to the lab frame (therefore omitting the
%scaling by d2(theta_r,m,0), see d2.m)
%
%Then, the function call looks like the following:
%
%   A = rotate_inter( A0,pwd,euler1,euler2,...,theta_r )
%
%The euler rotations may be omitted if all interactions in the Hamiltonian
```

```
%are co-linear:
%
%   A = rotate_inter( A0,pwd,theta_r)
%
%And furthermore, the specification of the rotor angle is only necessary if
%spinning off the magic angle (defaults to acos(sqrt(1/3)))
%
%   A = rotate_inter( A0,pwd)
%       or
%   A = rotate_inter( A0,pwd,euler1,euler2,...)
%
%The form of the arguments are
%
%   A0=[delta, eta]    (delta is the anisotropy, eta is the asymmetry)
%
%   pwd: Structure with fields alpha,beta,gamma (radians)
%
%   euler=[alpha beta gamma] (radians)
%
%   theta_r: Rotor angle (radians). Note: to omit rotation from rotor frame
%            to lab frame, set to NaN.
%
%   A. Smith, Mar. 2015
%

%% Determine if theta_r is given, and how many sets of euler rotations
% In the following section, we determine what arguments were given. The
% rotor angle and initial frame transformations are both optional, and are
% differentiated by the size of the argument (frame transformation is a
% 3-element vector, and rotor angle is a single number.
if nargin>2&&numel(varargin{end})==1
    theta_r=varargin{end};
    euler=varargin(1:end-1);
    nea=nargin-3;
elseif nargin>2
    euler=varargin(1:end);
    nea=nargin-2;
    theta_r=acos(sqrt(1/3));
else
    euler={[0 0 0]};
```

```
    theta_r=acos(sqrt(1/3));
    nea=1;
end

if isempty(euler)
    euler={[0 0 0]};
    nea=1;
end

%% Calculate tensor in the principle axis system.
PAS2=zeros(1,3);  %Pre-allocate (note that in PAS, n=1 component always =0)
PAS2(3)=-0.5*A0(1)*A0(2);   %n=2 component (same as n=-2 component)
PAS2(1)=sqrt(3/2)*A0(1);    %n=0 component

%% Set default rotor angle if not given
if not(exist('theta_r','var'))||isempty(theta_r)
    theta_r=acos(sqrt(1/3));
end

%% Rotate into the molecular frame
MOL2=zeros(1,5);    %Pre-allocate the tensor for the molecular frame

for k=-2:2     %Calculate the k=-2,1,0,1,2 components in the molecuar frame
    index=k+3;
    for j=[-2 0 2]  %Calculate parts from the j=-2,0,2 components
                    %(n=+/- 1 PAS component is zero)
        MOL2(index)=MOL2(index)+...
            exp(-1i*(euler{1}(1)*j+euler{1}(3)*k))*...
            d2(euler{1}(2),j,k)*PAS2(abs(j)+1);
    end
 %A2_(2,k)=sum_over_j (D_(j,k)(alpha,beta,gamma)*A1_(2,j)
 %  where
 %D_(j,k)(alpha,beta,gamma)=exp(-1i*k*gamma)*d2(beta,j,k)*exp(-1i*j*alpha))
end

if nea>1
    %If extra euler angles are specified, perform additional transformations
    for kk=2:nea
        temp=MOL2;  %Store current MOL2 in temp, perform roations as above
        MOL2=zeros(1,5);
```

```
        for k=-2:2
            index=k+3;
            for j=-2:2  %Since we no longer start from PAS,
                        %must consider all components j=-2,1,0,1,2
                MOL2(index)=MOL2(index)+...
                    exp(-1i*(euler(Lundström et al.)(1)*j+euler(Lundström et
al.)(3)*k))*d2(euler(Lundström et al.)(2),j,k)*temp(j+3);
            end
        end
    end
end

%% Rotate into the lab frame
A=zeros(length(pwd.alpha),3);
%Pre-allocate output vector. #Rows is size of the powder average,
%and 3 columns for n=0,1,2
alpha=pwd.alpha(:);               %Make sure angle vectors are all columns
beta=pwd.beta(:);
gamma=pwd.gamma(:);

for k=0:2           %Calculate the 3 components, k=0,1,2 in the lab frame
    % Sum up components
    index=k+1;
    %Zero index not allowed in matlab, so shift storage to 1,2,3 locations
    for j=-2:2      %Sum over components, transform into the rotor frame
        A(:,index)=A(:,index)+...
            exp(-1i*(alpha*j+gamma*k)).*d2(beta,j,k)*MOL2(j+3);
    end
    % Do final scaling, including rotation into lab frame.
    if isnan(theta_r)
        A(:,index)=A(:,index)*2/sqrt(6);
        %If leaving in rotor frame, don't scale by d2. Scale by sqrt(2/3)
    else
        A(:,index)=A(:,index)*2/sqrt(6)*d2(theta_r,k,0);
        %To go to the lab frame, scale by sqrt(2/3) and by d2(theta_r,k,0)
    end
end
```

```
end


%%%%%%%%%%%%%%%%%%%% end rotate_inter.m %%%%%%%%%%%%%%%%%%%




%%%%%%%%%%%%%%%%%%%% rotate_inter_full.m %%%%%%%%%%%%%%%%%%%
function [ A ] = rotate_inter_full( A0,pwd,varargin )
%ROTATE_INT_FULL: Performs same function as rotate_inter, but returns a
%Nx5x5, and then includes all components of the dipole coupling, not only
%the terms required for high-field approximation. rotate_inter help entry i
%is below (note that we do not rescale the coupling definitions by
%1/sqrt(6) here as was done in rotate_inter- since not all interactions
%scale the same way).
%
%   2nd index gives component in rotor frame, third index is lab frame
%
%Input an interaction of the form [iso,delta,asym], a structure
%containing a powder average (elements alpha,beta,gamma), and
%optionally, euler angles for the interaction (more than one set may be
%given, in which case they will be executed in order, starting from the
%PAS), and the rotor angle (if omitted, assumed to be 54.7356). Output is
%then a matrix, containing each of the five components in the rotor frame
%(A(2,n), n=-2:2) but multiplied by the wigner rotation matrix to switch
%into the lab frame (if NaN is given for theta_r, then the components will
%be left in the rotor frame). For usage in simulations, each component must
%be additionally multiplied by exp(-1i*2*pi*n*vr*t).
%
%   A = rotate_inter( A0,pwd,euler1,euler2,...,theta_r )
%
%   A0=[delta, eta]    (input delta is same as output delta from
%                           dipolar_coup)
%   pwd: Structure with fields alpha,beta,gamma (radians)
%   euler=[alpha beta gamma] (radians)
%   theta_r: Rotor angle (radians). Note: to obtain full tensor, set
%       theta_r to NaN (setting 0 will only return A00 components).
```

```
%
%   A. Smith, Mar. 2015- (expansion to all terms Jun. 2015)
%


%% Determine if theta_r is given
if nargin>2&&numel(varargin{end})==1
    theta_r=varargin{end};
    euler=varargin(1:end-1);
    nea=nargin-3;
elseif nargin>2
    euler=varargin(1:end);
    nea=nargin-2;
    theta_r=acos(sqrt(1/3));
else
    euler={[0 0 0]};
    theta_r=acos(sqrt(1/3));
    nea=1;
end

if isempty(euler)
    euler={[0 0 0]};
    nea=1;
end

%% Calculate principle components
PAS2=zeros(1,3);
PAS2(3)=-0.5*A0(1)*A0(2);
PAS2(1)=sqrt(3/2)*A0(1);

%% Set some defaults
if not(exist('euler','var'))||isempty(euler)
    euler=[0 0 0];
end

if not(exist('theta_r','var'))||isempty(theta_r)
    theta_r=acos(sqrt(1/3));
end

%% Rotate into the molecular frame
```

```
MOL2=zeros(1,5);

for k=-2:2
    index=k+3;
    for j=[-2 0 2]
        MOL2(index)=MOL2(index)+...
            exp(-1i*(euler{1}(1)*j+euler{1}(3)*k))*...
            d2(euler{1}(2),j,k)*PAS2(abs(j)+1);
    end
end

if nea>1
    for kk=2:nea
        temp=MOL2;
        MOL2=zeros(1,5);
        for k=-2:2
            index=k+3;
            for j=-2:2
                MOL2(index)=MOL2(index)+...
                    exp(-1i*(euler(Lundström    et    al.)(1)*j+euler(Lundström    et
al.)(3)*k))*...
                    d2(euler(Lundström et al.)(2),j,k)*temp(j+3);
            end
        end
    end
end

%% Rotate into the lab frame
A0=zeros(length(pwd.alpha),5);
A=zeros(length(pwd.alpha),5,5);
alpha=pwd.alpha(:);
beta=pwd.beta(:);
gamma=pwd.gamma(:);

for k=-2:2
    index=k+3;
    % Sum up components
    for j=-2:2
        A0(:,index)=A0(:,index)+...
            exp(-1i*(alpha*j+gamma*k)).*d2(beta,j,k)*MOL2(j+3);
```

```
    end
    % Do final scaling, including rotation into lab frame.
    if isnan(theta_r)
        A(:,index,3)=A0(:,index);
    else
        for j=-2:2
            A(:,index,j+3)=A0(:,index)*d2(theta_r,k,j);
        end
    end
end



end

%%%%%%%%%%%%%%%%%%%% end rotate_inter_full.m %%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%% tc_v_MF.m %%%%%%%%%%%%%%%%%%
% This script calculates the Redfield model-free R1p values, as given by
% Kurbanov et al. J. Chem. Phys. 135, 184104 (2011), and simulates R1p
% using a two-site hoping model. Note that we only include the dipole
% coupling for the 1H-13C pair, as it is the main contributor to 13Ca R1p
% relaxation.
%
% Computational time on an 8-core Macbook Pro Retina (mid-2012)
% is ~175 seconds (running 8 jobs in parallel).
%


%% Set Parameters
tc=logspace(-2,-11,100);    %Correlation times to be simulated/calculated

dHC=dipolar_coup(.109,'1H','13C');  %1H-13C dipole coupling (Hz)

vr=60e3;        %Rotor frequency (Hz)
v1=35e3;        %Spin lock strength (Hz)
```

```
angle=5;        %Angle between H-CA vectors for two-site hop (degrees)

vH=850e6;       %1H Larmor frequency (only used for model free calculation)
vC=vH*GyroRatio('13C/1H');  %13C Larmor frequency

%% First, calculate R1p using a simulation

%% Set up time axis
nr=1e2;      %Number of rotor periods per propagation step
deltat=nr/vr;   %Time point step size
maxt=5;         %Maximum time to be calculated (s)
t=0:deltat:maxt;    %Time axis

%% Generate the spin matrices
[I S]=n_spin_system;    %Generate spin matrices

rho0=[I.x(:);I.x(:)];
%Gives the initial state of the system in Liouville space
%(spread spin matrices into a vector)
detect=[I.x(:)' I.x(:)'];  %Gives the detection operator in Liouville space
norm=1/(detect*rho0);
%Calculate the normalization constant (so magnetization starts at 1)

%% Setup the powder average
pwd=pwd_avg(3); %Generate the powder average
                %(structure with fields alpha,beta,gamma,weight, N)
AHC1=rotate_inter([dHC 0],pwd);
%Calculate n=0,1,2 components of the dipole coupling in the lab frame
AHC2=rotate_inter([dHC 0],pwd,[0 angle 0]*pi/180);
%As above, but first we tilt the dipole coupling by the hop angle

signal0=cell(pwd.N,1);
%Pre-allocate a cell to store the signal for each element of the pwd. avg.

%% Loop over powder average
%Open a parallel pool for faster calculation (embarrassingly parallel)
parfor k=1:pwd.N
    %Calculate time traces for each element of the powder average
    Ham1=cell(1,3);
```

```
%Pre-allocate cell to store parts of Hamiltonian rotating at 0,1, and 2
%times the rotor frequency
Ham2=cell(1,3);
%As above, but the dipole of this Hamiltonian is tilted by the hop
%angle relative to Ham1
for j=1:3
    Ham1{j}=AHC1(k,j)*S.z*I.z; %#ok<*PFBNS>
    %%Multiply the interaction by sqrt(3/2)*T_(2,0)
    %(SzIz for heternuclear dipole)
    Ham2{j}=AHC2(k,j)*S.z*I.z;
    %Same as above, for tilted Hamiltonian
end

Ham1{1}=v1*I.x;     %Add the spin-locking strength to the non-rotating
                    %part of the Hamiltonian
Ham2{1}=v1*I.x;     %Same as above.

signal0{k}=zeros(size(tc,2),size(t,2));
%Pre-allocate signal vector for this element of powder average

for m=1:size(tc,2)      %Loop over all correlation times
    kex=1/2/tc(m);      %Calculate the exchange rate (1/(2*tc))
    U=exchange_prop(2,kex,vr,100,1,Ham1,Ham2);
    %Calculate the propagator in Liouville space for one rotor period
     % Number of sites, exchange rate, spinning speed,
     %number of steps over rotor period,
     %number of propagators returned for each rotor period,
     %Ham for site 1, Ham for site 2

    U=U{1}^nr;          %Calculate the propagator for nr rotor periods
    [V,D]=eig(U);       %Diagonalize the propagator
    %(returns eigenvectors in V, eigenvalues in D
    D=diag(D);%Turn square matrix with eigenvalues into a single vector

    rho_det=(transpose(detect*V).*(V\rho0)*norm)*ones(1,size(t,2));
        %The above line constructs the product of the initial density
        %matrix and the detection operator in the eigenbasis of the
        %propagator. It then repeats it for every point in the time
        %axis (all the columns are the same-> faster multiplication for
        %signal calculation)
```

```
        U1=[ones(size(detect,2),1) cumprod(D*ones(1,size(t,2)-1),2)];
            %Because the propagator is diagonalized, we can calculate its
            %value at all time points by repeatedly multiplying it by
            %itself- as done in the cumprod.

        signal0{k}(m,:)=sum(rho_det.*U1,1);
            %We take the product of the propagator with the initial density
            %matrix and the detection operator (all in the eigenframe of
            %the propagator). Technically, this is the dot product at each
            %time point. However, to all time points at once, we have to do
            %an elementwise multiplication followed by a summation down the
            %first dimension. (a*b=sum(transpose(a).*b,1))
    end
end

%% Sum together the powder average
signal=zeros(size(tc,2),size(t,2));
%Pre-allocate signal to sum up contributions from different powder elements

for k=1:pwd.N       %Loop over powder elements
    signal=signal0{k}*pwd.weight(k)+signal;
    %Add together, accounting for the weighting of each element
end

%% Fit results to exponentials

R1phop=zeros(1,size(tc,2));  % Pre-allocate vector to store actual hoping rates.

fun0=@(X)X(1)*exp(-t*X(2));   % Exponential function to fit data to exponential
decay

for k=1:size(tc,2)  % Loop over all correlation times
    fun=@(X)sum((real(signal(k,:))-fun0(X)).^2);
    % Sum of squares function
    [~,b]=min(abs(exp(-1)-real(signal(k,:))));
    % Find the element closest to 1/e, use as initial guess for 1/R1p
    fit=fminsearch(fun,[1 min([1/t(b) 1/t(2)])]);
    %Fit signal vector to exponential using simplex minimization
    R1phop(k)=fit(2);                       %Store the rate
```

```
end

%% Calculate with model free (see Kurbanov et al.)

S2=1/4*(3*cos(angle*pi/180)^2+1);
% Calculate the model-free S2 value for 2-site hop

omega0=2*pi*[vH-vC vC vH+vC]';
% Calculate angular frequencies needed for R1 calculation

J0=2/5*(1-S2)*(ones(3,1)*tc)./(1+(omega0*tc).^2);
%Calculate spectral densities for R1 calculation

R1=(pi*dHC/2)^2*(J0(1,:)+3*J0(2,:)+6*J0(3,:));    % Calculate R1

omega=2*pi*[vH 2*vr-v1 vr-v1 vr+v1 2*vr+v1]';
%Calculate angular frequencies from MAS, RF strength
J=2/5*(1-S2)*(ones(5,1)*tc)./(1+(omega*tc).^2);
%Calculate corresponding spectral densities


R1del=(pi*dHC/2)^2*(3*J(1,:)+1/3*J(2,:)+2/3*J(3,:)+2/3*J(4,:)+1/3*J(5,:));
%Calculate R1del (see Kurbanov et al.

R1pMF=R1/2+R1del;  %Calculate R1p from R1 and R1del


%% Plot the results
figure(2)
loglog(tc,R1phop)
hold all
loglog(tc,R1pMF)
grid on
legend('Two-Site Hop','Model-Free')

%%%%%%%%%%%%%%%%%%%%% end tc_v_MF.m %%%%%%%%%%%%%%%%%%
```

References

Abergel D, Palmer AGI (2003) On the Use of the Stochastic Liouville Equation in Nuclear Magnetic Resonance: Application to R1 Relaxation in the Presence of Exchange. Concepts Magn Reson A 19A:134-148

Balguerie A, Dos Reis S, Ritter C, Chaignepain S, Bénédicte C-S, Forge V, Bathany K, Lascu I, Schmitter J-M, Riek R, Saupe SJ (2003) Domain organization and structure±function relationship of the HET-s prion protein of Podospora anserina. EMBO J 22:2071-2081

Clifford AA (1973) Multivariate error analysis: a handbook of error propagation and calculation in many-parameter systems. Wiley, New York, NY

Kurbanov R, Zinkevich T, Krushelnitsky A (2011) The nuclear magnetic resonance relaxation data analysis in solids: General R 1/R 1 equations and the model-free approach. J Chem Phys 135:184104 (184101-184109)

London RE, Avitabile J (1978) Calculated 13C NMR Relaxation Parameters for a Restricted Internal Diffusion Model. Application to Methionine Relaxation in Dihydrofolate Reductase. J Am Chem Soc 100:7159-7165

Lundström P, Teilum K, Carstensen T, Bezsonova I, Wiesner S, Hansen DF, Religa T, Akke M, Kay LE (2007) Fractional 13C enrichment of isolated carbons using [1-13C]- or [2-13C]-glucose facilitates the accurate measurement of dynamics at backbone Ca and side-chain methyl positions in proteins. J Biomol NMR 38:199-212

Schneider DJ, Freed JH (1989) Spin Relaxation and Motional Dynamics. Adv Chem Phys 73:387-527

The MathWorks I (2013) MATLAB, 2013b edn., Natick, Massachusetts, USA

Wittebort RJ, Szabo A (1978) Theory of NMR relaxation in macromolecules: Restricted diffusion and jump models for multiple internal rotations in amino acid side chains. J Chem Phys 69:1722-1736