Supplementary Information:

Quantifying Proton NMR Coherent Linewidth in Proteins Under Fast MAS Conditions: A Second Moment Approach

Alexander A. Malär[1], Susanne Smith-Penzel[1], Gian-Marco Camenisch [1], Thomas Wiegand[1], Ago Samoson[2], Anja Böckmann[3*], Matthias Ernst[1*], Beat H. Meier [1*]

[1]*Physical Chemistry, ETH Zürich, Vladimir-Prelog-Weg 2, 8093 Zurich, Switzerland*

[2] *School of Information Technologies, Tallinn University of Technology, Tallinn, Estonia, NMR Institute MTÜ, Tallinn, Estonia*

[3]*Institut de Biologie et Chimie des Protéines, Bases Moléculaires et Structurales des Systèmes Infectieux, Labex Ecofect, UMR 5086 CNRS, Université de Lyon, 7 passage du Vercors, 69367 Lyon, France*

** Corresponding Authors*

Table of Contents:

S1: Motivation of factor 3 in the derivation of analytical formulas with an $M_2(F_xI_{kx})$ approach

Working with second-moment expressions calculated with an F_xF_x approach is well established in the literature. Furthermore we already validated our analytic expressions for $M_2(F_xF_x)$ under MAS on CaF_2, as it can be seen in Fig. 2 of the main text. In this section we adress how to find the correct normalization factor in a *local* $M_2(F_xI_{kx})$ approach.

We know thath the total M_2 of a four-spin system (1,2,3,4) $M_2(1234)$ is calculated in the F_xF_x case by summing over all possible three-spin systems and dividing by the number of resonant spins

$$M_2(1234) = \frac{1}{4}\big(M_2(123) + M_2(124) + M_2(134) + M_2(234)\big), \tag{S1}$$

where each $M_2(ijk) = \mathrm{tr}\,([F_x\, H_{\mathrm{eff}}][F_x\, H_{\mathrm{eff}}])/\mathrm{tr}(F_xF_x)$. We note that there are four terms contributing. In an F_xI_{kx} approach, we calculate the *local* second moment of one of the spins, by a weighted sum over all three spin systems, containing the spin of interest

$$M_2(1{,}1234) = \frac{1}{R}(M_2(1{,}123) + M_2(1{,}124) + M_2(1{,}134)), \tag{S2}$$

where each $M_2(i, ijk) = \mathrm{tr}\,([F_x\, H_{\mathrm{eff}}][I_{ix}\, H_{\mathrm{eff}}])/\mathrm{tr}(F_xI_{ix})$. It is clear that the sum of the local moments of each of the 4 spins, divided by the number of resonant spins (four in this case) should give the total second moment of the system $M_2(1234)$, as it has been calculated in an F_xF_x approach:

$$\begin{aligned} M_2(1234)' = \frac{1}{4R}(&M_2(1{,}123) + M_2(1{,}124) + M_2(1{,}134) + \\ &M_2(2{,}213) + M_2(2{,}214) + M_2(2{,}234) + \\ &M_2(3{,}312) + M_2(2{,}314) + M_2(2{,}324) + \\ &M_2(4{,}412) + M_2(4{,}413) + M_2(4{,}423)). \end{aligned} \tag{S3}$$

By numerical simulation on a four-spin system in CaF_2 we find that $R = 3$, such that $M_2(1234)' = M_2(1234)$. This factor can be understood by the fact that there are in total $4 \times 3 = 12$ terms contributing. If we look at the problem from a combinatorial point of view, we can calculate the amount of terms contributing to the total second moment in the F_xF_x case $\binom{4}{3} = \frac{4!}{3!1!} = 4$ and in the F_xI_{kx} case $4 \cdot \binom{3}{2} = \frac{4 \times 3!}{2!1!} = 12$. Dividing 12/4 we find that R corresponds indeed to how many more terms contributing to the *total* second moment in an $M_2(F_xF_x)$ compared to a $M_2(F_xI_{kx})$ approach, namely three.

This argument can be extended to a five-spin system. In an $M_2(F_xF_x)$ approach, there are $\binom{5}{3} = \frac{5!}{3!2!} = 10$ terms, in an $M_2(F_xI_{kx})$ approach, there are $5 \cdot \binom{4}{2} = 5 \cdot \frac{4!}{2!2!} = 30$ terms contributing to the total second moment. According to our argument $R = \frac{30}{10} = 3$, which we checked by a numerical simulation on a five spin system in CaF_2.

For a general N spin system, the amount of terms contributing to the total second moment the $M_2(F_xF_x)$ and the $M_2(F_xI_{kx})$ approach respectively is

$$\binom{N}{3} = \frac{N!}{3!\,(N-3)!}, \quad N\binom{N-1}{2} = \frac{N(N-1)!}{2!\,(N-1-2)!}. \tag{S4}$$

Thus in an $M_2(F_x I_{kx})$ approach, there will always be

$$\frac{N(N-1)!}{2!\,(N-3)!}\frac{3!\,(N-3)!}{N!} = 3 \tag{S5}$$

more terms contributing to the total second moment, compared to an $M_2(F_x F_x)$ approach, such that for an arbitrary-sized spin system the normalization factor will always be $R = 3$.

S2: CaF_2 ^{19}F spectra & Gaussian Fit

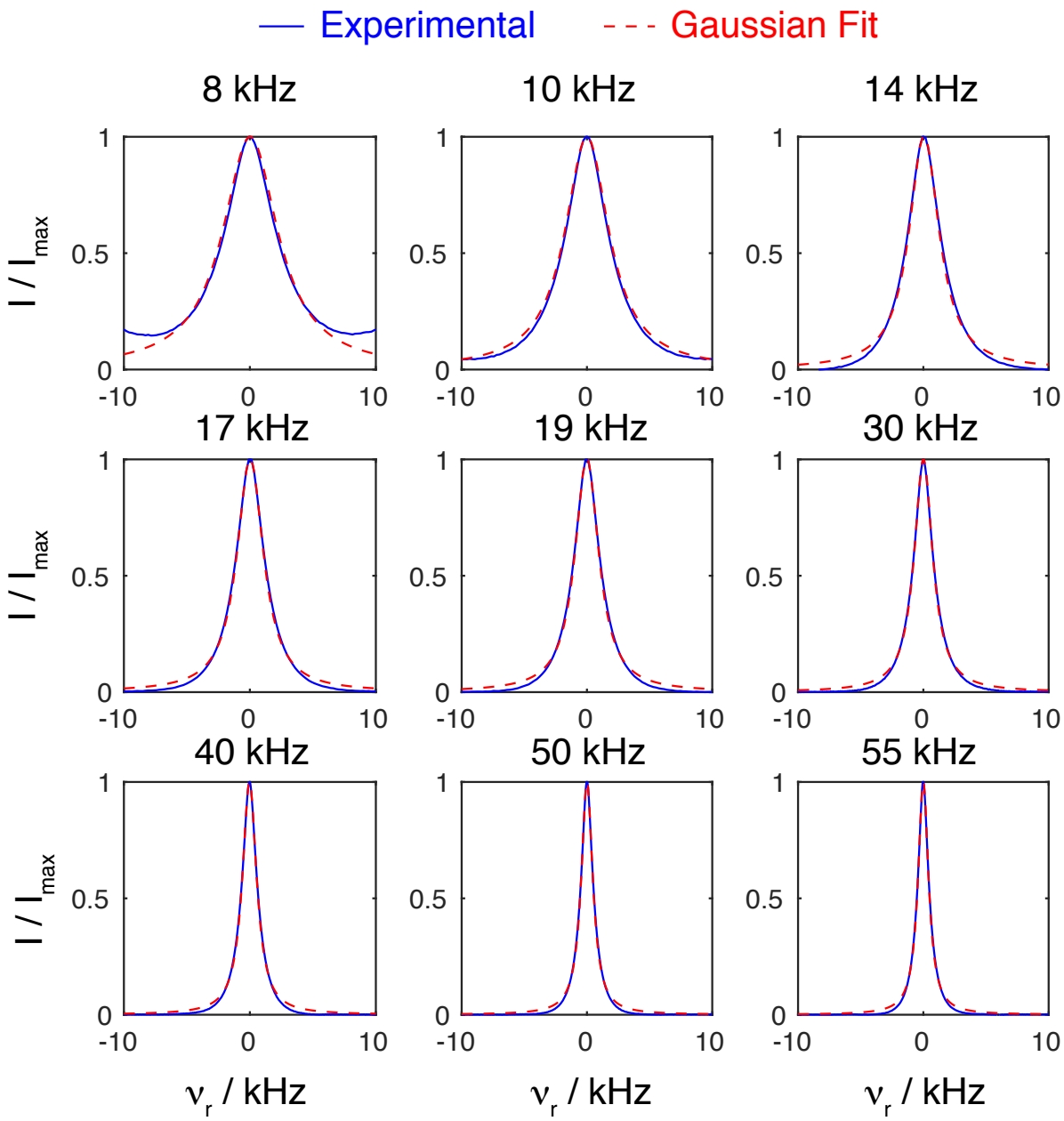


Fig. S1 Experimental ^{19}F CaF_2 spectra at different MAS frequencies (blue) and corresponding Gaussian fit (red dashed line).

S3: Strong vs. weak coupling regimes in chemical shift inclusion:

For inclusion of chemical shift effects in the M_2 simulations, we compute a numerical spectrum from the Hamiltonian $H_{eff} = H_{DD} + H_{CS}$, where H_{DD} is calculated from second-order Floquet theory and $H_{CS} = +\sum_k \Omega_k I_{kz}$. In order to construct the spectrum (as explained in the main text), we use the I_{1x} operator as detection operator rather than the total spin operator F_x. In the weak coupling regime, the I_{1x} operator is a "good" detection operator in the sense that only transitions of spin 1 can be seen. In the strong coupling regime, transitions of all spins can be observed and the individual transitions cannot be attributed to spin 1, since the individual spin states are mixed in different linear combinations. The chemical shift inclusion is therefore expected to be exact within the weak coupling regime and slightly biased in the intermediate regime of the Hamiltonian. The 8×8 matrix representation of the H_{eff} considering only the 12,13 couplings is given by

$$\begin{pmatrix}
\frac{\Omega_1}{2}+\frac{\Omega_2}{2}+\frac{\Omega_3}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\Omega_1}{2}+\frac{\Omega_2}{2}-\frac{\Omega_3}{2} & -iW & 0 & -2iW & 0 & 0 & 0 \\
0 & iW & \frac{\Omega_1}{2}-\frac{\Omega_2}{2}+\frac{\Omega_3}{2} & 0 & 2iW & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\Omega_1}{2}-\frac{\Omega_2}{2}-\frac{\Omega_3}{2} & 0 & -2iW & 2iW & 0 \\
0 & 2iW & -2iW & 0 & -\frac{\Omega_1}{2}+\frac{\Omega_2}{2}+\frac{\Omega_3}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 2iW & 0 & -\frac{\Omega_1}{2}+\frac{\Omega_2}{2}-\frac{\Omega_3}{2} & iW & 0 \\
0 & 0 & 0 & -2iW & 0 & -iW & -\frac{\Omega_1}{2}-\frac{\Omega_2}{2}+\frac{\Omega_3}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\Omega_1}{2}-\frac{\Omega_2}{2}-\frac{\Omega_3}{2}
\end{pmatrix}$$

with two 3×3 blocks which need to be diagonalized. The diagonal elements are given by sums and differences between the individual chemical shifts, while the off-diagonal elements are given by the effective couplings W, which have been computed from Floquet theory and which are equal to

$$W = \frac{1}{4 \cdot 64\nu_r} \cdot \delta_{13}\delta_{12} \sin(\alpha) \sin(\beta) \sin(\theta_{1213})((9 + 7\cos 2\beta) \cdot \cos\theta_{1213} - 7\cos\alpha \sin(2\beta) \sin(\theta_{1213})).$$

Typical values for W at 110 kHz MAS are in the order of 10^{-4}-10 Hz and much smaller than typical chemical-shift differences between coupled spins (0.1 ppm is already 85 Hz at 850 MHz). The ratio between chemical shift differences and effective couplings is quantified by the parameter $k = W/\Delta\Omega$. A small k ($k < 0.01$) is an indicator for weak couplings, a large k is an indicator for strong couplings.

Fig S1(a) shows the k statistics over all Ubiquitin H_N. One can see that the overwhelming majority of k parameters lies in orders of magnitude $\leq 10^{-2}$. Fig S1(b) shows that the weak coupling regime starts at around 10^{-2}, where the chemical-shift scaling factor is almost constant with k. Most Ubiquitin protons are in the weak coupling regime and therefore I_{kx} is a good detection operator. From the insert in Fig. S1(a) one sees that only a few proton spins are in the intermediate coupling regime, where chemical shift scaling factors are biased by sattelite transitions.

Note that this result is guaranteed by the fast MAS condition. In the static case, spins would be in weak/intermediate and strong coupling regimes.

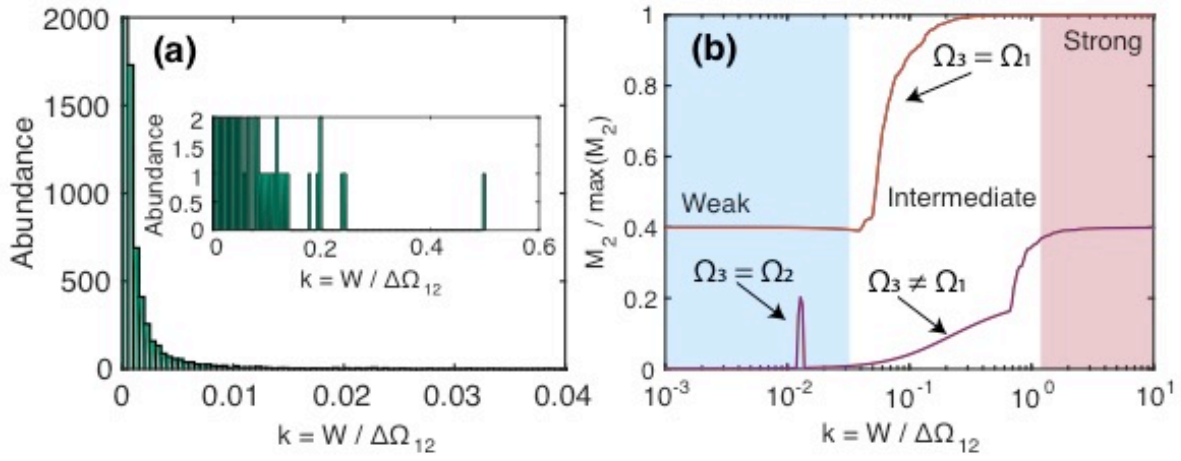


Fig. S2(a) Statistic of all H_N k values in DUL 100% back-exchanged Ubiquitin (H_N+H_{ex}). The majority of observed k values is $\leq 10^{-2}$. **(b)** Chemical shift scaling factor as function of k in the cases of spectrally overlapping spin 1-3 (red) and for all 3 chemical shifts different. The blue shaded area shows the weak coupling regime where most k values of the H_N are observed, the red shaded area shows the strong coupling regime (almost coinciding spins), while the white area indicates an intermediate coupling regime, where estimation of chemical shift scaling factors may be biased due to I_{1x} not being a good detection operator anymore.

S4: Chemical Shift Scaling Factors using Liouville-von Neumann Simulations

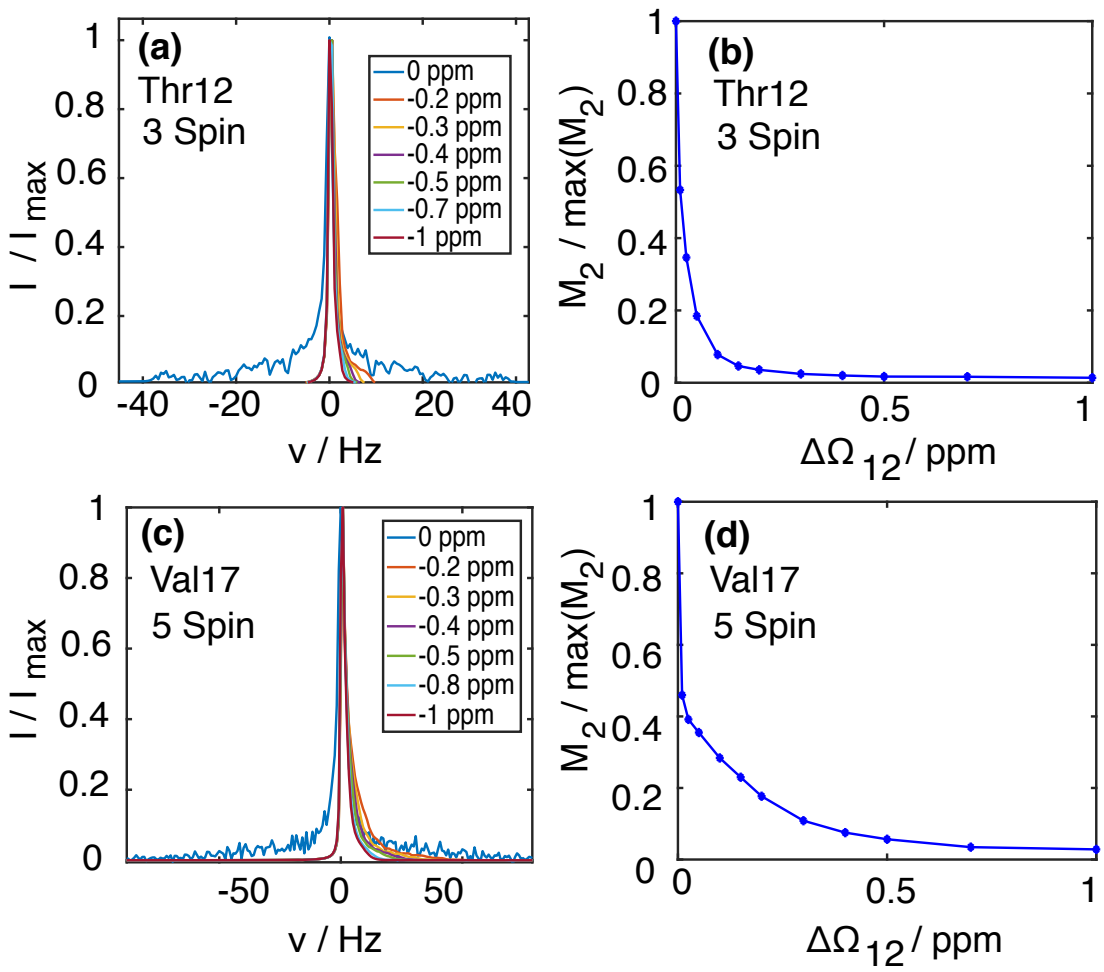


Fig. S3: Liouville-von Neumann spectra for different chemical shift offsets of spin 2 in a 3 spin geometry of the H_N in Thr12 (a) and in a 5 spin geometry of Val17 (b). M_2 extracted from the numerical spectra as function of the chemical shift offset of spin 2 for Thr12 (c) and Val17 (d). These results coincide with the findings of the M_2+CS method described in the main text.

S5: Convergence of Line Width Simulations

S5a: Second-Moment Simulations

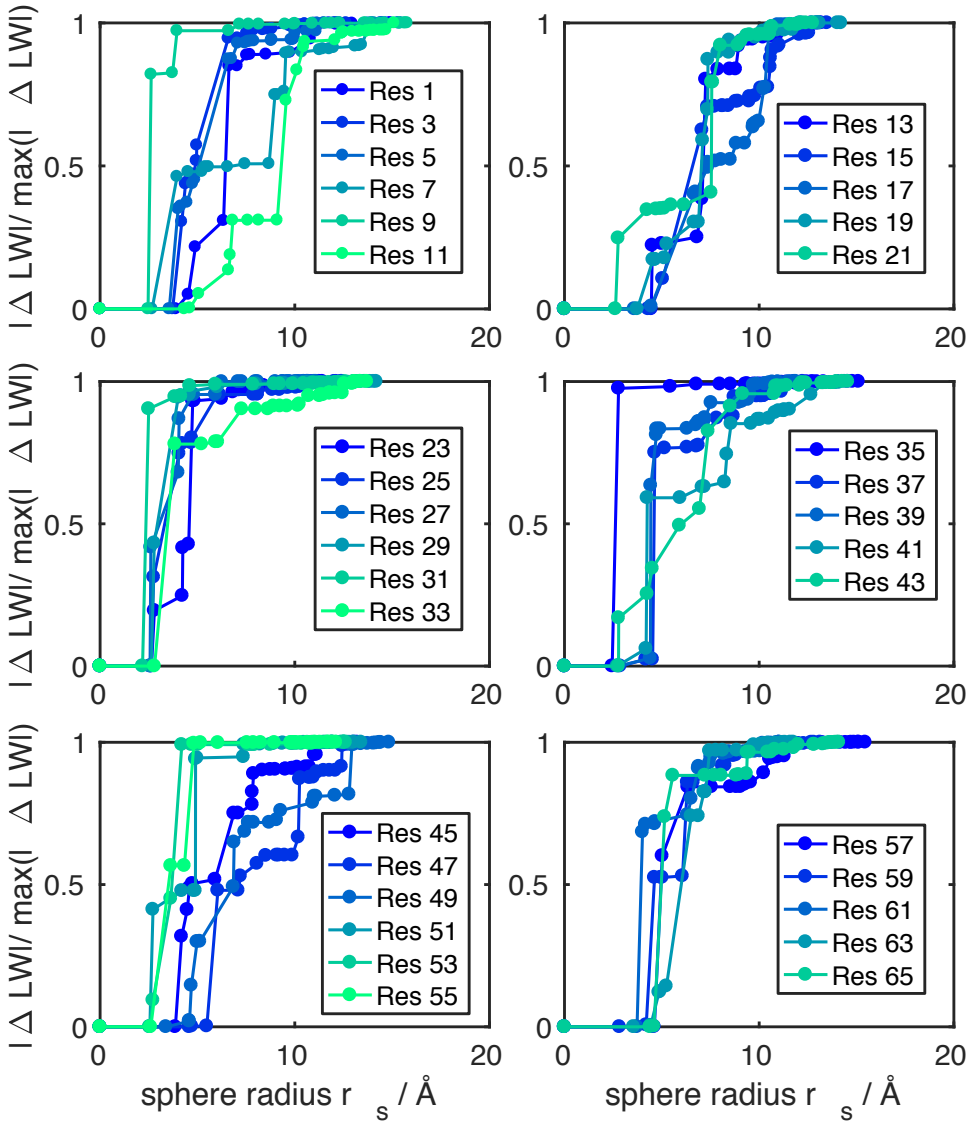


Fig. S4: M_2 convergence test for selected residues in DUL 100% B.E. Ubiquitin (H_N + H_{ex}) at 110 kHz MAS. LW_{prot} denotes the line width of the M_2 simulation considering all protons in a unit cell. $LW(r_s)$ denotes the line width obtained by considering all spins within a distance of r_s around the spin-of-interest. Convergence seems to be reached between 5-7 Å (approximately 10-12 spins). However, line width contributions of 1-2 Hz can still be observed after this radius, which can make a difference in a deuterated sample.

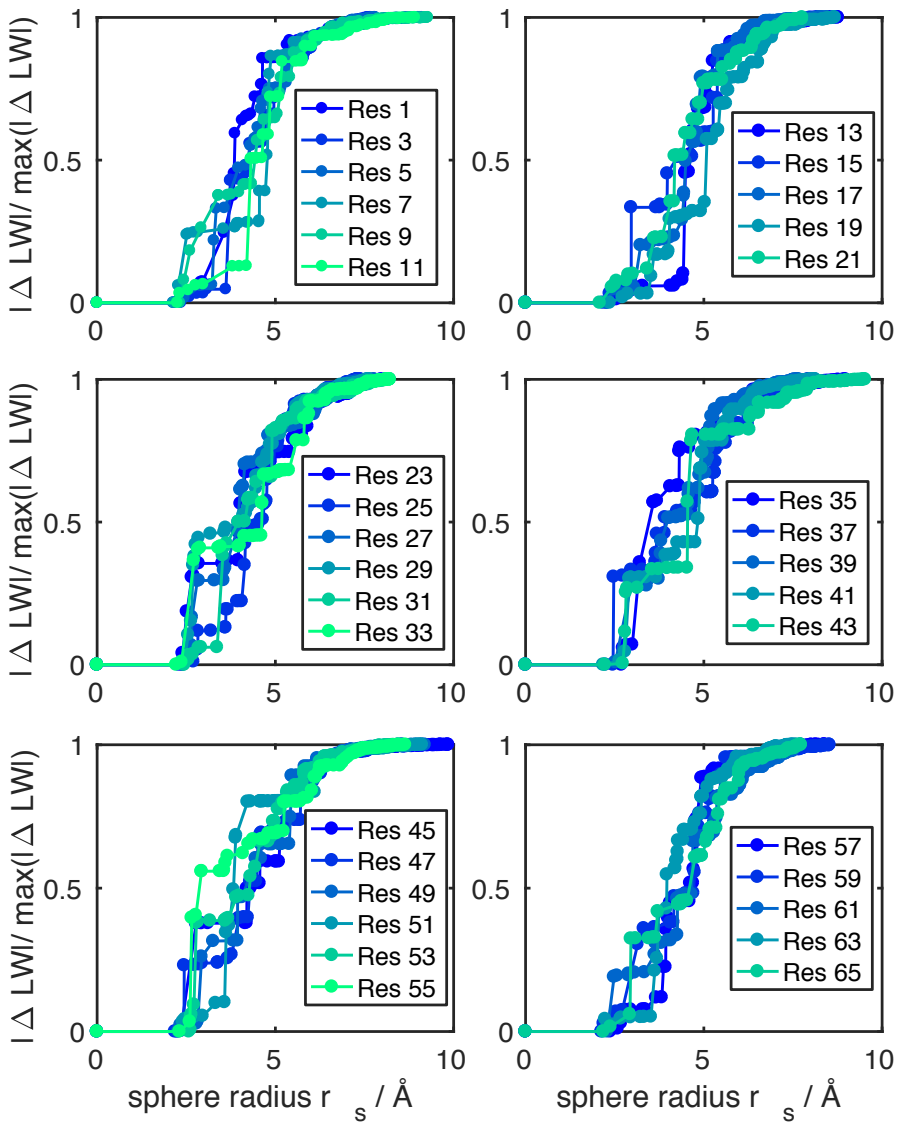


Fig. S5: M_2 convergence test for selected residues in FP Ubiquitin at 125 kHz MAS. LW_{prot} denotes the line width of the M_2 simulation considering all protons in a unit cell. $LW(r_s)$ denotes the line width obtained by considering all spins within a distance of r_s around the spin-of-interest. The main contribution to the simulated line width is achieved within a sphere of radius r_s corresponding to around 60 spins.

S5b: Liouville-von Neumann Simulations

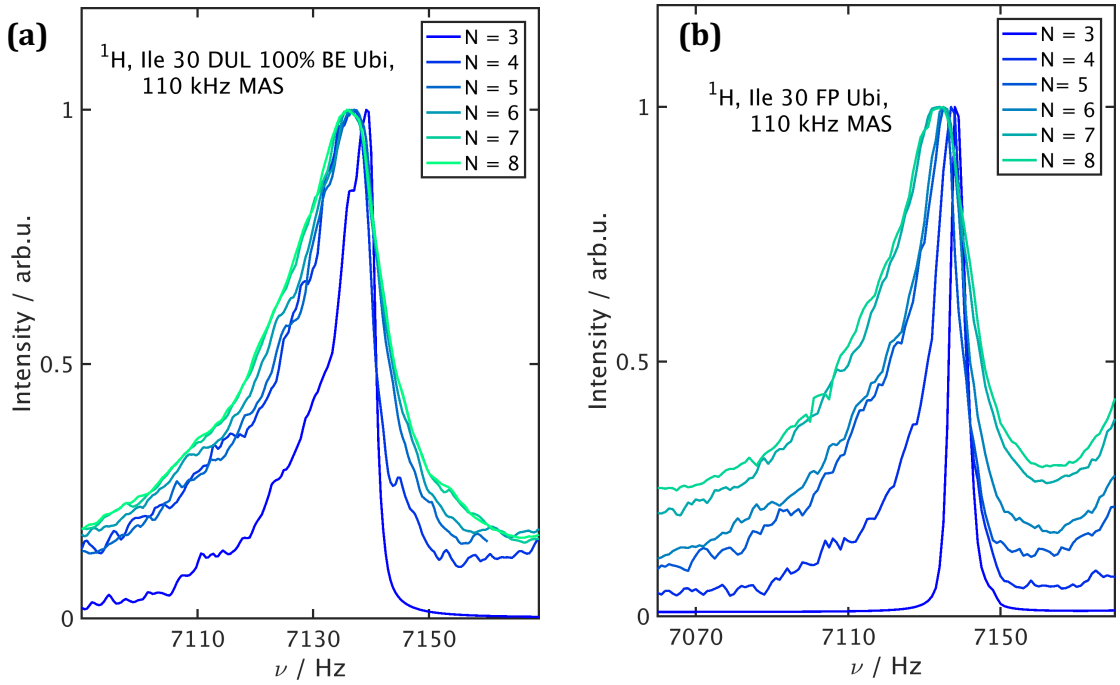


Fig. S6: Illustration of convergence behavior in LvN simulations, showing the numerical line for N = 3-8 in the example of Ile30 in DUL 100% BE (a) and Fully-protonated (b) Ubiquitin at 110 kHz MAS. LvN simulations have been performed in Matlab. The code is provided as electronic supplementary information. Line shapes seem to become more symmetric with increasing N but still a considerable asymmetry is observed.

S6: Additional line shape analysis

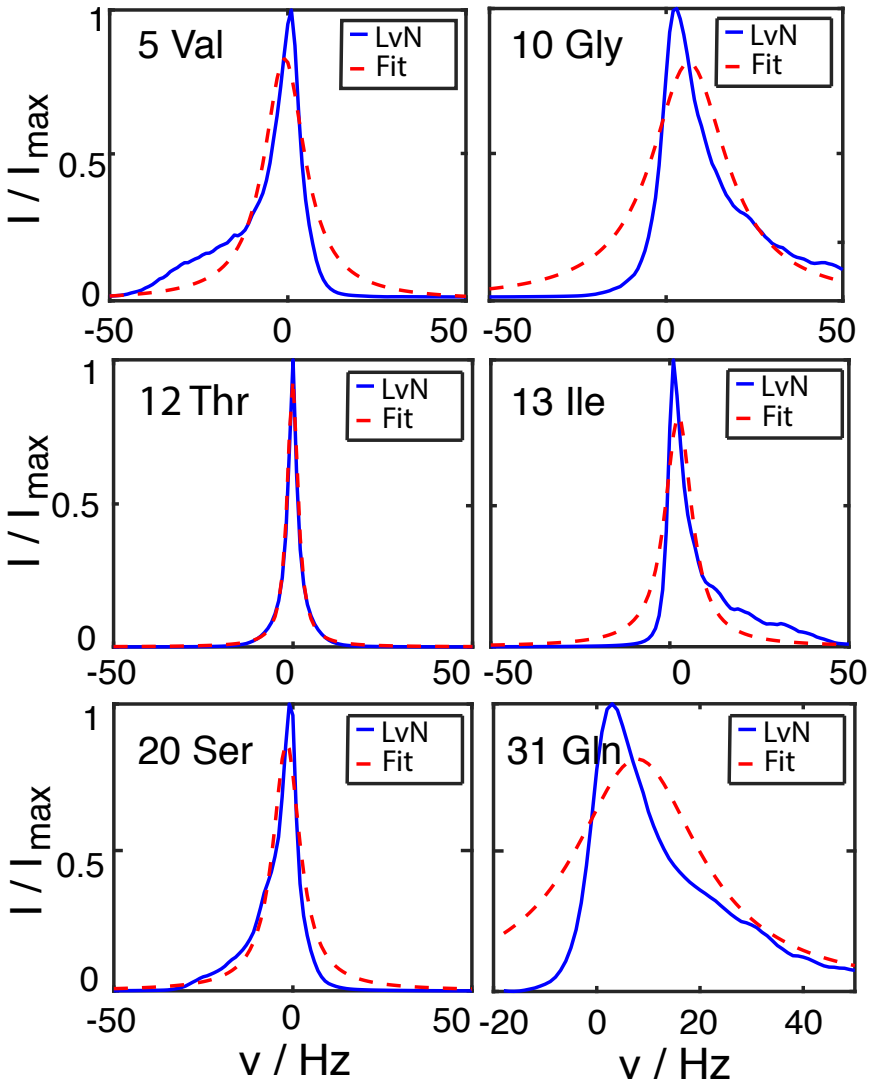


Fig. S7: Line shapes simulated with the Liouville- von Neumann approach and performance of the Gaussian fit for illustrative residues. Thr12 shows a case in which the fit perfectly matches the numerical line, all other cases show, how line shapes obtained from numerical simulations are strongly asymmetric and can't be reliably fitted with Lorentzian or Gaussian (symmetric) models.

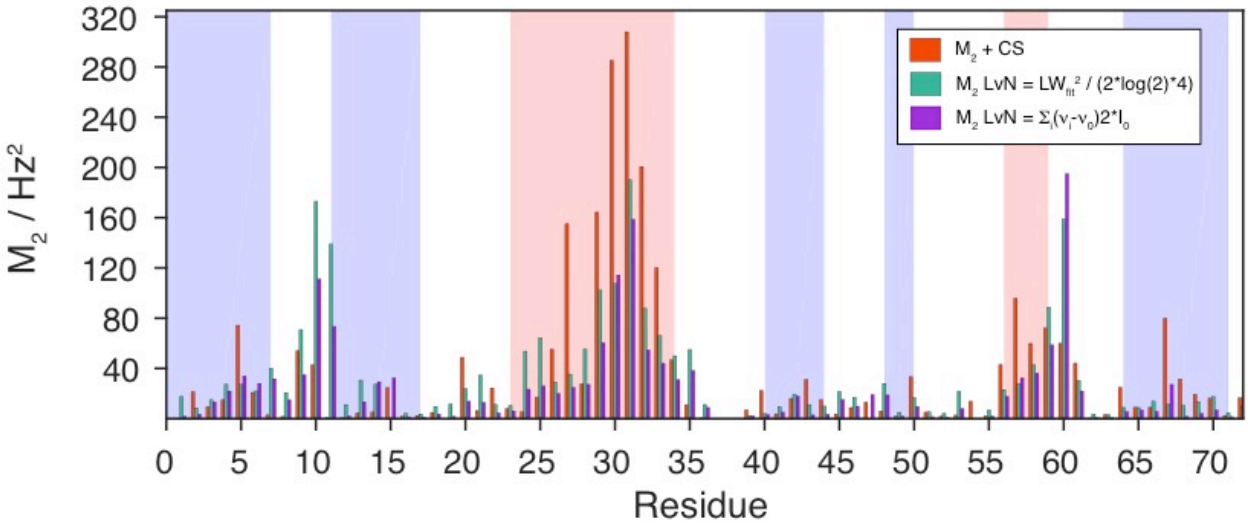


Fig S8: Comparison of simulated second moments. Red: Second moment from M_2+CS simulations. Green : Second moment extracted from the spectral line shape of the numerical spectra simulated with the LvN approach. Violet: M_2 from LvN spectra, converting the fitted Gaussian line width into a second moment. Second moments obtained from the fitted line width matches well order of magnitude and general trends obtained from the line shape. Discrepancies are mainly observed, in areas of high asymmetry (where the Gaussian fit is not the best option for fitting the line width, *e.g.* Gln31) and in regions where spectral overlap with neighboring peaks can bias the M_2 determination. Note that discrepancies are more pronounced in the second moment, but due to the $\sqrt{M_2}$ proportionality of the line width, less significant when directly comparing line widths (as it is done in the main text).

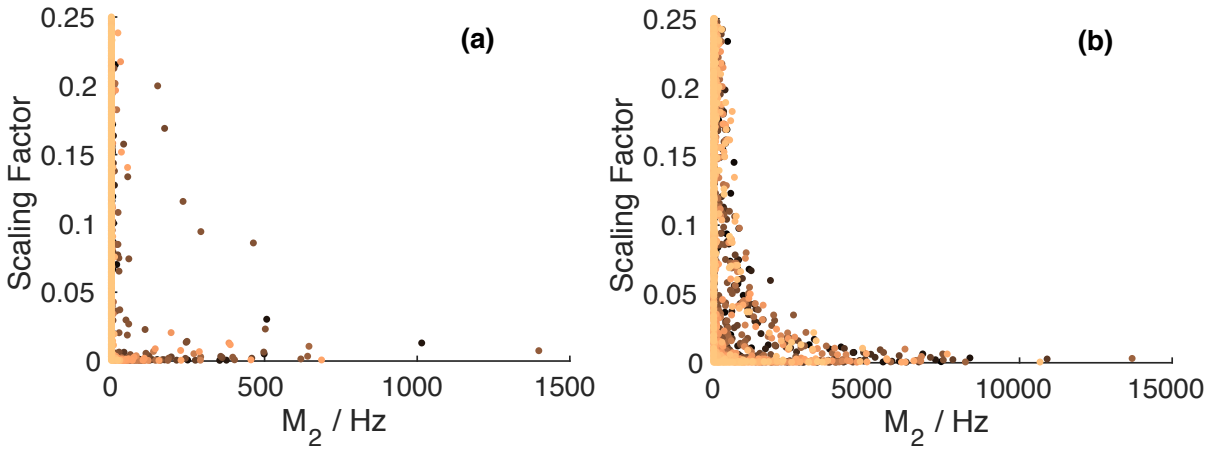


Fig. S9: Scatter Plots M_2 vs. chemical shift scaling factor for all three spin system contributions in DUL 100% back-exchanged Ubiquitin H_N +H_{ex} (a) and fully-protonated Ubiquitin (b) to enlucidate how many of them lead to an actual contribution to the proton line widths.

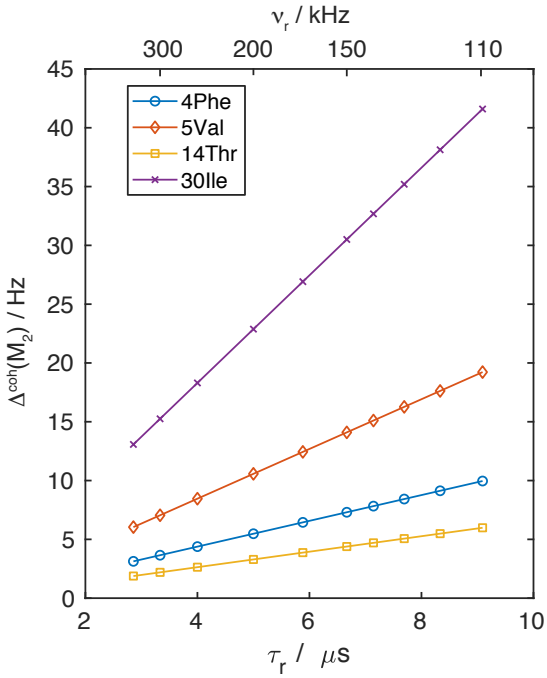


Fig. S10: Simulated MAS dependence for a selection of four representative residues (narrow/intermediate/broad linewidths) in DUL 100% back-exchanged Ubiquitin considering only the H_N protons. The simulations have been performed at MAS frequenceis of 110, 120, 130, 140, 150, 170, 200, 250, 350 kHz. Spinning frequencies of roughly 300 Hz are required to get into the order of magnitude of a solution-state sample (6-9 Hz).

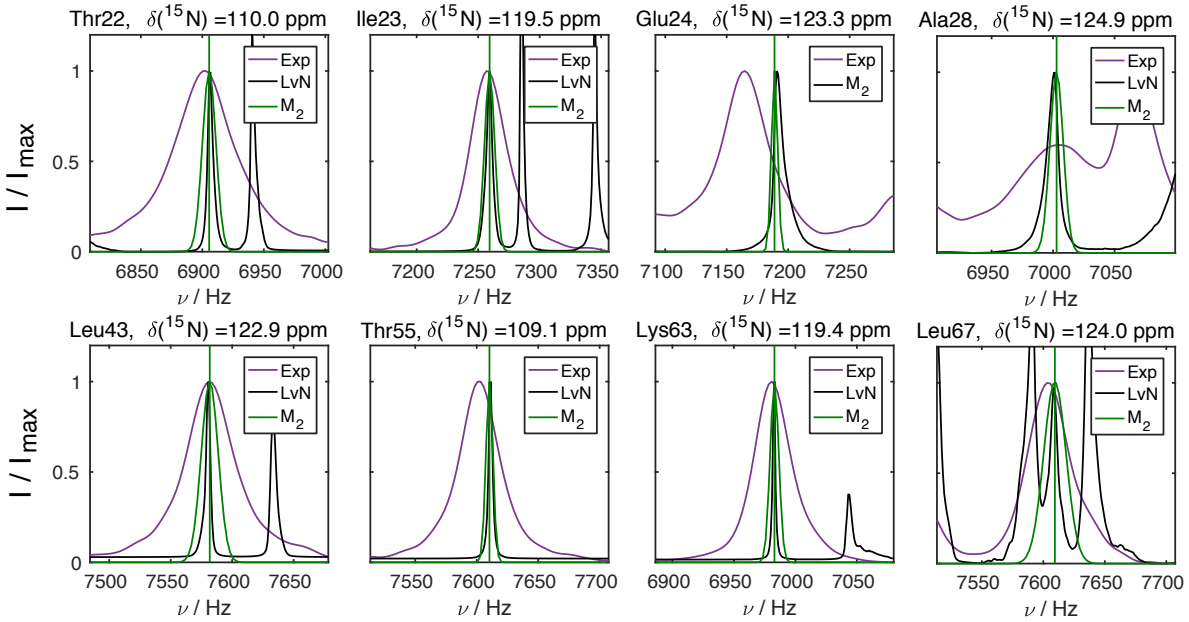


Fig. S11: Comparison of experimental (blue) and simulated (LvN black, M_2 red) line shapes at 110 kHz for the remaining experimentally broad residues, which cannot be explained, considering coherent effects (H_N + H_{ex}). Note that the LvN simulations return a 1D spectrum. The additional resonances observed in the LvN spectra are given by protons with close ^{1}H shift, which in a ^{1}H-^{15}N would however appear at a different ^{15}N shift.

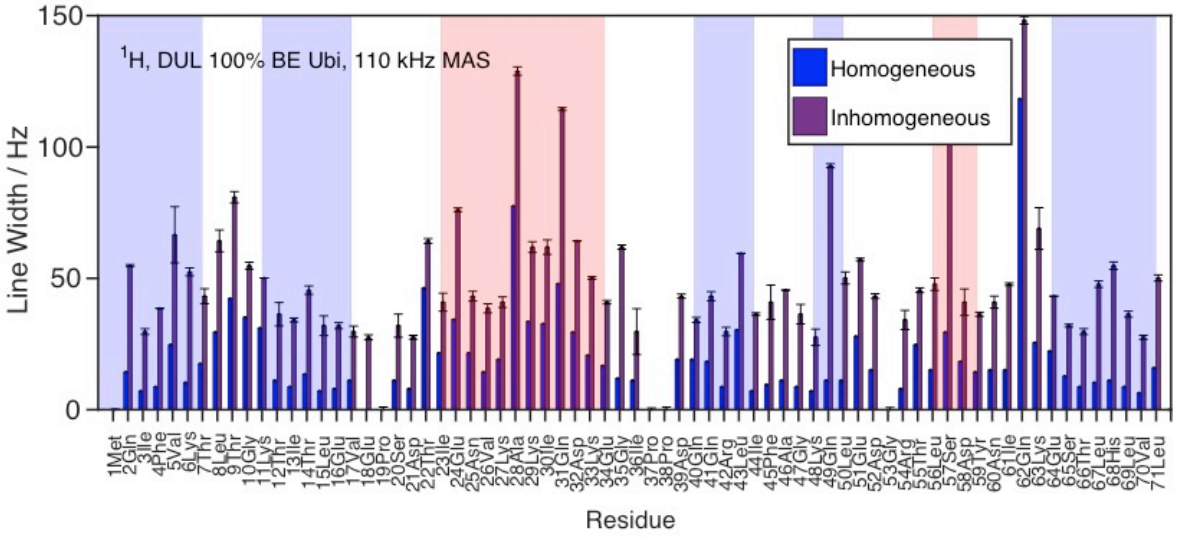


Fig. S12: Comparison between homogeneous (blue) and total experimental line widths (viole). Former is extracted from T_2' relaxation time, the second one is obtained by directly fitting the peaks in the hNH spectrum with a Gaussian function and contains as well effects from sample and magnetic field inhomogeneity (e.g. shim).

S7: Molecular Dynamic post-optimization of proton coordinates:

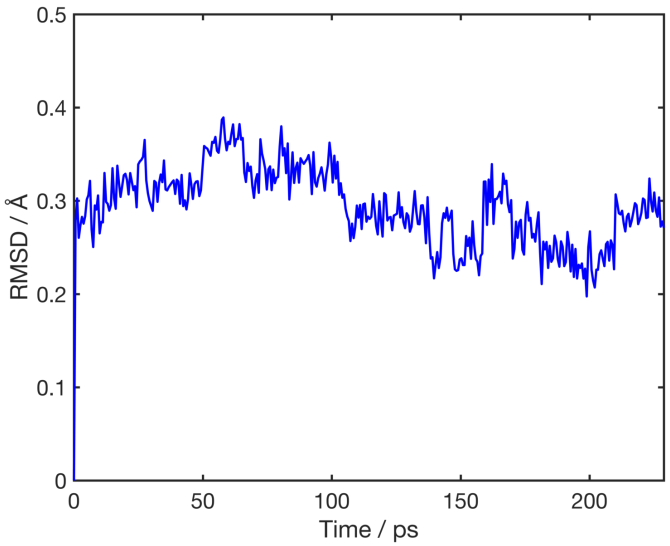
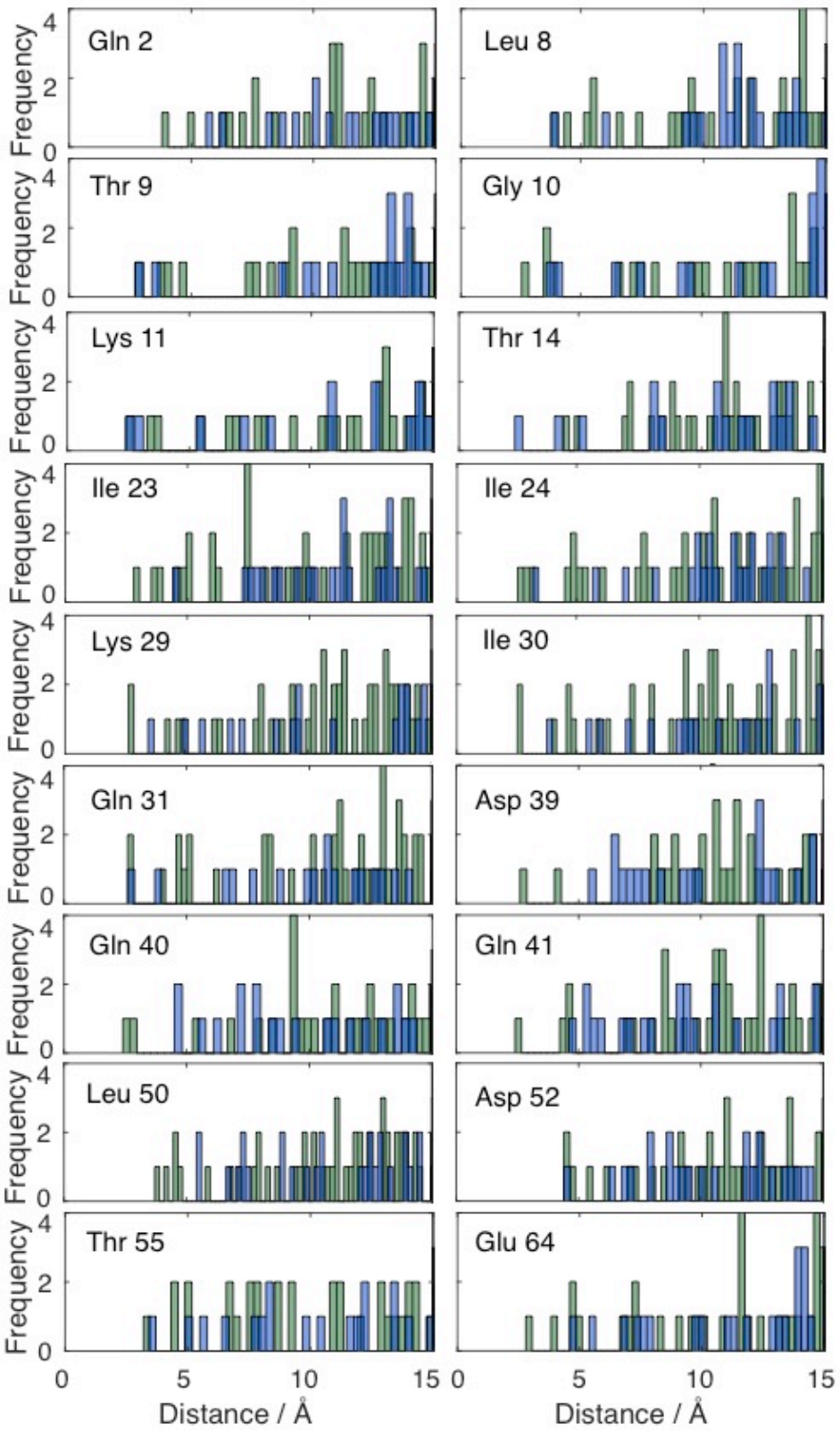


Fig. S13: RMSD trajectory for proton coordinates compared to the starting coordinates, during the MD simulation. Proton positions have been generated and post-optimized in VMD, using the NAMD software. We used a force field of the CHARMM type and periodic boundary conditions. The protein has been solvated in a water box of basic dimension 10 Å, to ensure that the protein does not leave the solvation shell, during the simulation and minimize surface effects. The simulation trajectory has been performed over 250 ps in basic time steps of 2 fs. Fig S11. shows a typical mean RMSD trajectory comparing all proton positions to the initial proton positions over time. The RMSD oscillation flattens rather rapidly, resulting in RMSD oscillations of ±0.2 Å, which are acceptably small for our purposes.

S8: Further considerations on coherent broadening by exchangeable side-chain atoms:



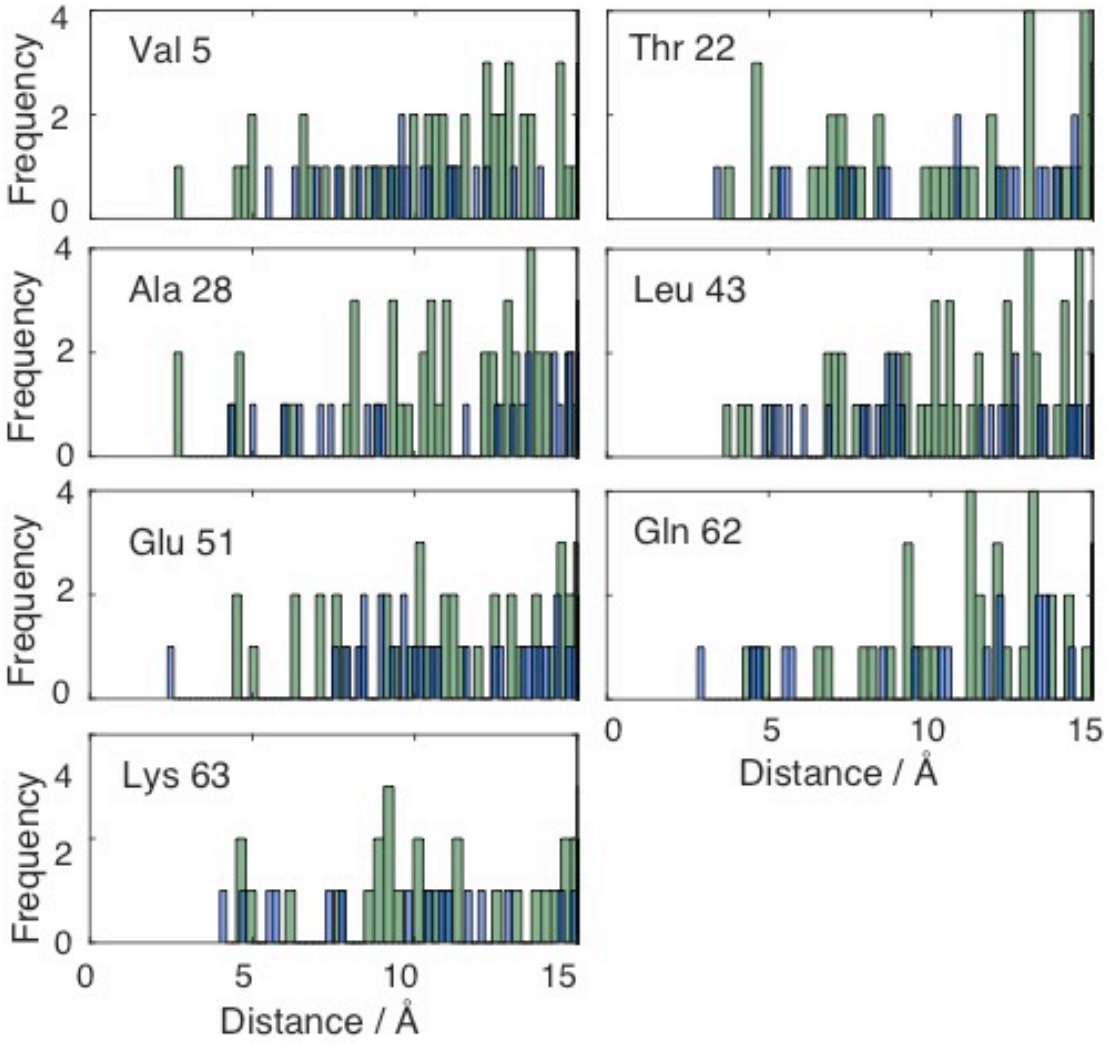


Fig. S14: Distance distributions for residues, where including back-exchanged site-chain protons, can change the line width prediction. In green H_N distances, in light blue H_{ex} distances. Note that close contacts do not necessarily imply a large effect on the simulated line width: since side chain protons are spectrally further away from amide protons, which can quench the purely geometrical line width contribution.

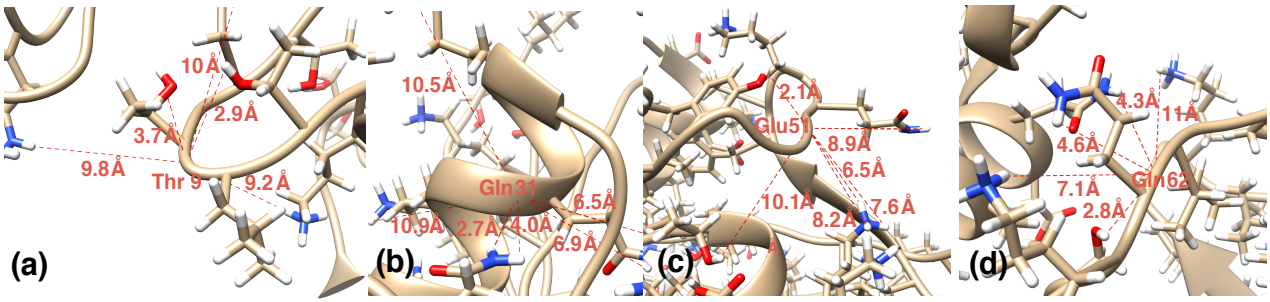


Fig. S15: Illustrative H_N – H_{ex} distances depicted on the Ubiquitin crystal structure for (a) Thr9, (b) Gln31, (c) Glu51, (d) Gln62.

S9: $R_{1\rho}$ (^{15}N) measurements for missing Ubiquitin residues:

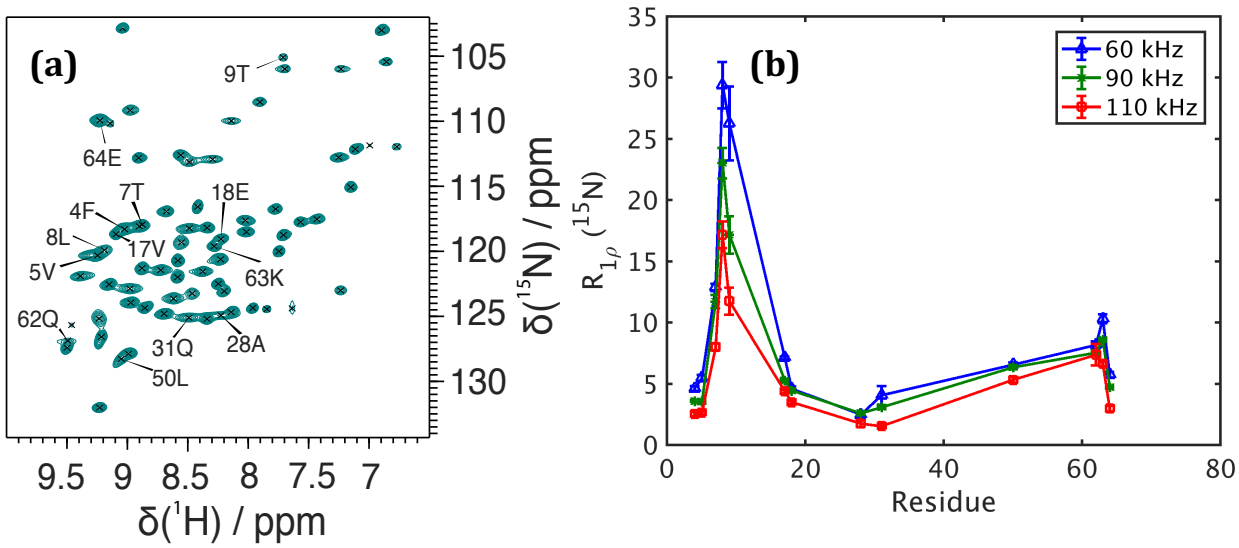


Fig. S16: (a) DUL Ubiquitin HSQC spectrum at 110 kHz MAS. The labeled peaks correspond to the residues for which $R_{1\rho}(\nu_r)$ rate information was missing in (Lakomek, Penzel, Cadalbert, Ernst, & Meier*, 2017). (b) Measured site-specific $R_{1\rho}$ rates for the missing Ubiquitin residues at 60 kHz (blue), 90 kHz (green) and 110 kHz MAS (red). Error bars have been obtained using the bootstrapping method.

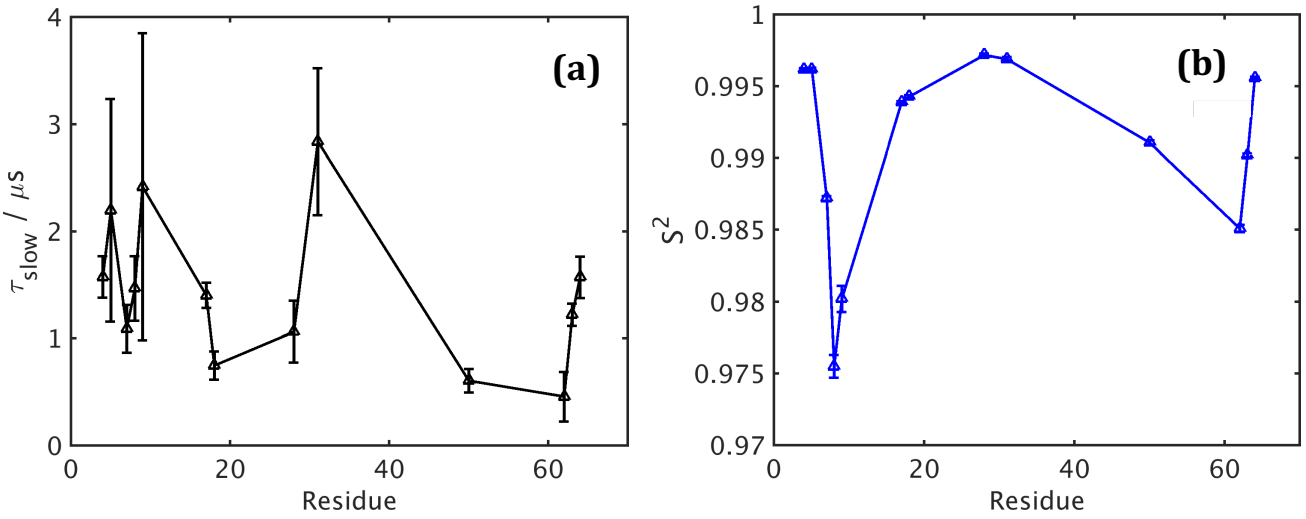


Fig. S17: Correlation times (a) and order parameters (b) for missing Ubiquitin residues extracted from the experimental ratios by using a simplified model-free approach and simultaneous χ^2 fitting of the data.

S10: MATLAB Codes

S10.1 Calculation of M2(FxI1x,DD+CS) for DUL 100% Ubiquitin

```
%% Calculation of site-specific M2(FxI1x, DD+CS) for DUL 100% B.E.
Ubiquitin
% As presented in the paper:
% "Quantifying Expected Coherent Proton Linewidth in Proteins Under Fast
% MAS Conditions: A Second Moment Approach"
% A.A.Malär, S. Penzel, A. A. Smith, A.Samoson,
% A.Böckmann, M.Ernst, B.H. Meier
% 02.02.2019 - A.A.Malär

clearvars -except filevector  % Prevent vector of CS scaling tables to
                              % be deleted after it has been loaded
close all

%% Physical constants & NMR Parameters
mu04pi  = 1e-7;             % vacuum permeability mu0/4pi in H/m
hbar    = 1.05457266e-34;   % reduced Planck constant
gammah  = 2.67522128e8;     % Proton gyromagnetic ratio Rad/(sT)
delta = -2*mu04pi*gammah^2*hbar/(1e-10^3*2*pi); % 1H-1H dipolar coupling
                                                % prefactor in Hz (no
1/r^3)
omegar=110e3;               % MAS frequency in Hz
field=850;                  % Magnetic field strength in MHz

%% Routine to load CS scaling factor (CSSF) tables (! slow process !)

if exist('filevector')  % Hint: avoid deleting CSSF vector, once loaded
else

filevector=zeros(71,71,45,45,66);  % Preallocate Vector for CSSF tables

for m = 1:1:71
for q=1:1:71

load(sprintf('./CS_table_FxI1xd/CS_FxIx_scale_table_O12_%0.0f_O13_%0.0f.mat
',m,q));
    size(scale);
    filevector(m,q,:,:,:)=scale(:,:,:);
end
end
end

%% Load Proton coordinates and corresponding CS vector

load('DUL_ubi_xyz_and_CS')

omega0=CoordAndCS(:,4)*field;
leD=length(omega0);
coordnhD2=CoordAndCS(:,1:3);

%% Start calculation of second moments

M2Dopt=zeros(1,leD);   % Preallocate Ref M2 (without CS) vector (68
residues)
M2DoptCS=zeros(1,leD); % Preallocate M2 + CS vector (68 residues)

for i1=1:leD  % Summation over all residues (site-specific M2)
    i1        % Index of spin of interest
```

```
% Summation over all 3 spin systems containing the spin of interest i1
  for i2=1:(leD-1)
    for i3=(i2+1):leD
      if(i1 ~= i2 && i1 ~= i3) % Avoid combinations like (112),(212)

        % Calculate internuclear vectors r12,r13,r23
        r12=sqrt(sum((coordnhD2(i1,:)-coordnhD2(i2,:)).^2));
        r13=sqrt(sum((coordnhD2(i1,:)-coordnhD2(i3,:)).^2));
        r23=sqrt(sum((coordnhD2(i2,:)-coordnhD2(i3,:)).^2));
        % Calculate angles between dipolar couplings
(12,13),(13,23),(12,23)
        theta1213=acos((coordnhD2(i1,:)-coordnhD2(i2,:))*(coordnhD2(i1,:)-
coordnhD2(i3,:)).'/(r12*r13));
        theta1323=acos((coordnhD2(i1,:)-coordnhD2(i3,:))*(coordnhD2(i2,:)-
coordnhD2(i3,:)).'/(r13*r23));
        theta1223=-acos((coordnhD2(i1,:)-coordnhD2(i2,:))*(coordnhD2(i2,:)-
coordnhD2(i3,:)).'/(r12*r23));
        % Attention to "-" sign for (12,23) coupling, by using this method.
        % (comes from vector algebra and definition of 23 vector)

        % Calculate absolute Chemical shift differences
        dO12=abs(omega0(i1)-omega0(i2));
        dO13=abs(omega0(i1)-omega0(i3));
        dO23=abs(omega0(i2)-omega0(i3));

        % Combine delta*/rik^3 to get delta_ik for (ik) coupling
        d12=delta/(r12^3);
        d13=delta/(r13^3);
        d23=delta/(r23^3);

    % Perform scaling factor look-out for closest geometry in filevector

SCF=find_FxI1x_CS_scaling_all_filesb(dO12,dO13,r12,r13,theta1213,field,file
vector);

    if (SCF==-Inf || SCF==+Inf) % Exclude (very rare!) cases where SCF is
undefinied or miscalculated
    elseif (SCF<0)
    else

    % Analytical Formula for M2(FxI1x) reference
    M2Dopt(i1)=M2Dopt(i1)-(1/(655360*omegar^2))*(-122*d12^2*d13^2-
28*d12*d13*(d12+d13)*d23-
61*(d12^2+d13^2)*d23^2+28*d13*cos(2*theta1213)*(d12*(2*d12*d13+(d12+d13)*d2
3)+...
    d23*(-
d12^2+d12*d13+d13*d23)*cos(2*theta1223))+33*d13^2*cos(4*theta1213)*(2*d12*(
d12+d23*cos(2*theta1223))+...
    d23^2*cos(4*theta1223))+d23*(d12*((-
94*d13^2+28*d12*(d13+d23))*cos(2*theta1223)+33*d12*d23*cos(4*theta1223))+..
.
    4*d13*(-
40*d12^2+7*d12*d13+7*d13*d23)*sin(2*theta1213)*sin(2*theta1223)+66*d13^2*(d
12+d23*cos(2*theta1223))*sin(4*theta1213)*sin(2*theta1223)));

    % Calculation of M2(DD+CS) calculated by SCF*M2(FxI1x)
    M2DoptCS(i1)=M2DoptCS(i1)-(SCF/(655360*omegar^2))*(-122*d12^2*d13^2-
28*d12*d13*(d12+d13)*d23-
61*(d12^2+d13^2)*d23^2+28*d13*cos(2*theta1213)*(d12*(2*d12*d13+(d12+d13)*d2
3)+...
    d23*(-
d12^2+d12*d13+d13*d23)*cos(2*theta1223))+33*d13^2*cos(4*theta1213)*(2*d12*(
d12+d23*cos(2*theta1223))+...
```

```
    d23^2*cos(4*theta1223))+d23*(d12*((-
94*d13^2+28*d12*(d13+d23))*cos(2*theta1223)+33*d12*d23*cos(4*theta1223))+..
.
    4*d13*(-
40*d12^2+7*d12*d13+7*d13*d23)*sin(2*theta1213)*sin(2*theta1223)+66*d13^2*(d
12+d23*cos(2*theta1223))*sin(4*theta1213)*sin(2*theta1223)));
    end
    end
  end
end
end
```

S10.2 Calculation of M2(FxI1x,DD+CS) for UL Ubiquitin

```
%% Calculation of site-specific M2(FxI1x, DD+CS) for UL Ubiquitin
% As presented in the paper:
% "Quantifying Expected Coherent Proton Linewidth in Proteins Under Fast
% MAS Conditions: A Second Moment Approach"
% A.A.Malär, S. Penzel, A. A. Smith, A.Samoson,
% A.Böckmann, M.Ernst, B.H. Meier

% Note: This code doesn't perform a simulation over all spins included
% in the coordinate file, but instead it simulates for each spin-of
% interest and each selection of input parameter N the spin systems
containing
% the N-1 closest spins to significantly reduce the calculation time.
% In practice N can be gradually increased until convergence is observed.
% 02.02.2019 - A.A.Malär

clearvars -except filevector  % Prevent vector of CS scaling tables to
                              % be deleted after it has been loaded
close all

%% Physical constants & NMR Parameters
mu04pi  = 1e-7;             % vacuum permeability mu0/4pi in H/m
hbar    = 1.05457266e-34;   % reduced Planck constant
gammah  = 2.67522128e8;     % Proton gyromagnetic ratio Rad/(sT)
delta = -2*mu04pi*gammah^2*hbar/(1e-10^3*2*pi); % 1H-1H dipolar coupling
                                                % prefactor in Hz (no
1/r^3)
omegar=125e3;               % MAS frequency in Hz
field=850;                  % Magnetic field strength in MHz

%% Routine to load CS scaling factor (CSSF) tables (! slow process !)

if exist('filevector')  % Hint: avoid deleting CSSF vector, once loaded
else

filevector=zeros(71,71,45,45,66);  % Preallocate Vector for CSSF tables

for m = 1:1:71
for q=1:1:71

load(sprintf('./CS_table_FxI1xc/CS_FxIx_scale_table_O12_%0.0f_O13_%0.0f.mat
',m,q));
    size(scale);
    filevector(m,q,:,:,:)=scale(:,:,:);
end
end
end

%% Load Coordinates and Chemical Shifts
```

```
load('FP_ubi_xyz_and_CS')
coordubis=CoordAndCS(:,1:3);
omega0=CoordAndCS(:,4)*field;
load('FP_ubi_type')
leD2=length(coordubis); % Number of 1H atoms over multiple molecules
leD= 599;       % Number of 1H atoms in one molecule

%% Sphere Parameters over which simulation should run (adjustable)

maxR=300;           % Number of closest spins N included in the analysis
                    % (change this number to simulate over more spins!)
ENDsphere=1:300;    % Used if convergence analysis' are performed
M2sphere=zeros(maxR,leD); % preallocation of
distSort=zeros(leD2,leD2);

%%

for r=maxR % Summation over size of N spin system. Note: if want to perform
           % convergence analysis, set r=1:maxR, which will give M2
           % calculated for each spin system of size between N=1 and maxR
           % If want only M2 of spin system with N=maxR, leave r=maxR.

% Preallocate for each sphere shell the M2 calculated over that sphere
M2D=zeros(1,leD);
M2Dref=zeros(1,leD);
r
for i1=1:1:leD % Summation over all residues (site-specific M2) in one
molecule
  if strcmp(typetot{i1},'HN') % Calculate M2 only for amide protons
     i1         % Index of spin of interest

% Routine for sorting the coordinates and CS, such that spin i1 is at the
% center and the first N-1 entries are the N-1 closest spins to spin i1.
       count=0;
        for k2=1:leD2
      count=count+1;
      % Calculate all distances to spin 1
      distInt(count)=sqrt(sum((coordubis(i1,:)-coordubis(k2,:)).^2));
        end
[distSort(i1,:), Isort]= sort(distInt); % Sort distances in descending
order

% Sort all relevant vectors (note: spin-of interest i1 has now index 1 and
% index N-1 is the N-1th closest spin to spin 1)
coordnhDsort=coordubis(Isort,:); % Obtain sorted xyz coordinate vector
omega0sort=omega0(Isort);        % Obtain sorted CS vector
typesort=typetot(Isort);         % Obtain sorted type vector

% Summation over all 3 spin systems containing the spin of interest i1
% within the N spin system
  for i2=1:(ENDsphere(r)-1)
    for i3=(i2+1):(ENDsphere(r))
        if(i2 ~= 1 && i3 ~= 1) % Avoid combinations like (112),(212)

        % Calculate internuclear vectors r12,r13,r23
        r12=sqrt(sum((coordnhDsort(1,:)-coordnhDsort(i2,:)).^2));
        r13=sqrt(sum((coordnhDsort(1,:)-coordnhDsort(i3,:)).^2));
        r23=sqrt(sum((coordnhDsort(i2,:)-coordnhDsort(i3,:)).^2));
        % Calculate angles between dipolar couplings
(12,13),(13,23),(12,23)
        theta1213=acos((coordnhDsort(1,:)-
```

```
coordnhDsort(i2,:))*(coordnhDsort(1,:)-coordnhDsort(i3,:)).'/(r12*r13));
        theta1323=acos((coordnhDsort(1,:)-
coordnhDsort(i3,:))*(coordnhDsort(i2,:)-coordnhDsort(i3,:)).'/(r13*r23));
        theta1223=-acos((coordnhDsort(1,:)-
coordnhDsort(i2,:))*(coordnhDsort(i2,:)-coordnhDsort(i3,:)).'/(r12*r23));
        % Attention to "-" sign for (12,23) coupling, by using this method.
        % (comes from vector algebra and definition of 23 vector)

        % Calculate absolute Chemical shift differences
        dO12=abs(omega0sort(1)-omega0sort(i2));
        dO13=abs(omega0sort(1)-omega0sort(i3));
        dO23=abs(omega0sort(1)-omega0sort(i3));

        % Combine delta*/rik^3 to get delta_ik for (ik) coupling
        d12=delta/(r12^3);
        d13=delta/(r13^3);
        d23=delta/(r23^3);

    % Perform scaling factor look-out for closest geometry in filevector

SCF=find_FxI1x_CS_scaling_all_filesb(dO12,dO13,r12,r13,theta1213,field,file
vector);

    if (SCF==-Inf || SCF==+Inf) % Exclude (very rare!) cases where SCF is
undefinied or miscalculated
    elseif (SCF<0)
    else

    % Analytical Formula for M2(FxI1x) reference
           M2Dref(i1)=M2Dref(i1)-(1/(3*655360*omegar^2))*(-122*d12^2*d13^2-
28*d12*d13*(d12+d13)*d23-
61*(d12^2+d13^2)*d23^2+28*d13*cos(2*theta1213)*(d12*(2*d12*d13+(d12+d13)*d2
3)+...
    d23*(-
d12^2+d12*d13+d13*d23)*cos(2*theta1223))+33*d13^2*cos(4*theta1213)*(2*d12*(
d12+d23*cos(2*theta1223))+...
    d23^2*cos(4*theta1223))+d23*(d12*((-
94*d13^2+28*d12*(d13+d23))*cos(2*theta1223)+33*d12*d23*cos(4*theta1223))+..
.
    4*d13*(-
40*d12^2+7*d12*d13+7*d13*d23)*sin(2*theta1213)*sin(2*theta1223)+66*d13^2*(d
12+d23*cos(2*theta1223))*sin(4*theta1213)*sin(2*theta1223)));

    % Calculation of M2(DD+CS) calculated by SCF*M2(FxI1x)
    M2D(i1)=M2D(i1)-(SCF/(655360*omegar^2))*(-122*d12^2*d13^2-
28*d12*d13*(d12+d13)*d23-
61*(d12^2+d13^2)*d23^2+28*d13*cos(2*theta1213)*(d12*(2*d12*d13+(d12+d13)*d2
3)+...
    d23*(-
d12^2+d12*d13+d13*d23)*cos(2*theta1223))+33*d13^2*cos(4*theta1213)*(2*d12*(
d12+d23*cos(2*theta1223))+...
    d23^2*cos(4*theta1223))+d23*(d12*((-
94*d13^2+28*d12*(d13+d23))*cos(2*theta1223)+33*d12*d23*cos(4*theta1223))+..
.
    4*d13*(-
40*d12^2+7*d12*d13+7*d13*d23)*sin(2*theta1213)*sin(2*theta1223)+66*d13^2*(d
12+d23*cos(2*theta1223))*sin(4*theta1213)*sin(2*theta1223)));
    end
     end
       end
    end
  end
end
```

```
M2sphere(r,:)=M2D; % Progressively insert M2 of the N spin system at the
Nth index

end

L=M2D(M2D~=0); % M2 of the amide protons (M2D contains only non-zero
entries for amide protons)
```

S10.3 Tabulation of Chemical Shift Scaling Factors (CSSF)

S10.3.1 Frame Routine for CSSF tabulation

```
%% Frame Routine for Calculation & Tabulation of Chemical Shift Scaling
Factors (CSSF)
% As presented in the paper:
% "Quantifying Expected Coherent Proton Linewidth in Proteins Under Fast
% MAS Conditions: A Second Moment Approach"
% A.A.Mal‰r, S. Penzel, A. A. Smith, A.Samoson,
% A.Bˆckmann, M.Ernst, B.H. Meier

% Note: This script tabulates the CSSF within the parameter
% space(r12,r13,theta1213) for one combination of input CS differences
% from the vector [0:0.025:0.2, 0.25:0.05:1, 1:0.2:10] ppm. Which chemical
% shifts are used for the simulation is selected using the indices zw1,zw2.

clear all
close all

%% Preallocate CSSF table in (r12,r13,theta1213) space
M23spin=zeros(45,45,66);    % M2(DD+CS)
M2FxIx=zeros(45,45,66);     % M2(DD)
scale=zeros(45,45,66);      % Scaling Factor

B0=850;  % Magnetic field in MHz;
vr=110;  % MAS frequency in kHz;
bw=50;   % Filter Bandwidth in Hz;

% Vector of CS differences used for the simulation
dO2=[0:0.025:0.2, 0.25:0.05:1, 1:0.2:10];
dO3=[0:0.025:0.2, 0.25:0.05:1, 1:0.2:10];

% Indices to select which CS differeneces are used in the simulation
% (replace xxx by the index you want)
zw1=xxx;
zw2=xxx;

% Simulated (r12,r13,theta1213) parameter space
r12=[1:0.2:8.2,9:16];
r13=[1:0.2:8.2,9:16];
theta1213=linspace(0.05,pi,66);

% Calculate M2(DD+CS),M2(DD) and the scaling factor for each parameter
% combination
for i=1:length(r12)
for j=1:length(r13)
for k=1:length(theta1213)
```

```
[r23,theta1223,theta1323]=TriangleSolver1213b(r12(i),r13(j),theta1213(k));

[M23spin(i,j,k),scale(i,j,k),M2FxIx(i,j,k)]=DD_and_CS_M2_computerc(r12(i),r
13(j),r23,theta1213(k), theta1223, dO2(zw1),dO3(zw2),B0,vr,bw);
end
end
end

%end
%end
%% Save the table
name1=sprintf('CS_FxIx_scale_table_O12_%0.0f_O13_%0.0f.mat',zw1,zw2)
name2=sprintf('CS_FxIx_M2_table_O12_%0.0f_O13_%0.0f.mat',zw1,zw2)

save(name1,'scale');
save(name2,'M23spin');
```

S10.3.2 Main calculation of CSSF for given input parameters

```
%% Main Routine to Calculate CSSF for 3spin system with input parameters
(r12,r13,theta1213,dO12,dO13)

function
[M23spin,scale,M2FxIx]=DD_and_CS_M2_computerc(r12,r13,r23,theta1213,
theta1223, CS2,CS3,B0,vr,bw)

%% Physical constants and NMR parameters
mu04pi  = 1e-7;
hbar    = 1.05457266e-34;
gammah  = 2.67522128e8;
delta = -2*mu04pi*gammah^2*hbar/(1e-10^3*2*pi);
omegar=vr*1e3;   % vr in kHz;

%% Transformation of distance into dipolar coupling
d12=delta/(r12^3);
d13=delta/(r13^3);
d23=delta/(r23^3);

%% Create spin operators

[S1,S2,S3]=n_spin_system;

%% Starting and detection operators
sigma0=S1.x+S2.x+S3.x; %(Fx starting operator);
Fdet=S1.x;             %(I1x detection operator);

Omega1=0*B0; % chemical shift of spin 1 in ppm

% Insert chemical shift difference

    Omega2=Omega1+B0*CS2;
    Omega3=Omega1+B0*CS3;

% Calculate Chemical Shift Hamiltonian (sum_k omega_k*S_kz)
CSHam=S1.z*Omega1+S2.z*Omega2+S3.z*Omega3;

% Analytical on-resonance M2 (FxI1x case):

M2FxIx=-(1/(655360*omegar^2))*(-122*d12^2*d13^2-28*d12*d13*(d12+d13)*d23-
61*(d12^2+d13^2)*d23^2+28*d13*cos(2*theta1213)*(d12*(2*d12*d13+(d12+d13)*d2
```

```
3)+...
    d23*(-
d12^2+d12*d13+d13*d23)*cos(2*theta1223))+33*d13^2*cos(4*theta1213)*(2*d12*(
d12+d23*cos(2*theta1223))+...
    d23^2*cos(4*theta1223))+d23*(d12*((-
94*d13^2+28*d12*(d13+d23))*cos(2*theta1223)+33*d12*d23*cos(4*theta1223))+..
.
    4*d13*(-
40*d12^2+7*d12*d13+7*d13*d23)*sin(2*theta1213)*sin(2*theta1223)+66*d13^2*(d
12+d23*cos(2*theta1223))*sin(4*theta1213)*sin(2*theta1223)));

% Create angular distirbution for powder averaging
pwd=pwd_avg(3);

M2D=zeros(1,1);
M2Dvar=zeros(1,1);
M4D=zeros(1,1);
M2=0;
M2pwd=zeros(1,pwd.N);
M2w=zeros(1,pwd.N);

transitionf=zeros(8,8);
Intensity=zeros(8,8);

        M2w=zeros(1,pwd.N);  % preallocate for internal loop (for powder
average points)
        for k=1:pwd.N  % loop over powder averaging points

         % Effective analytical dipolar strengths (required for calculation
of effective Hamiltonian)

WA(k)=(1/(256*omegar))*d12*d13*sin(theta1213)*sin(pwd.alpha(k))*sin(pwd.bet
a(k))*(cos(theta1213)*(9+7*cos(2*pwd.beta(k)))-
7*cos(pwd.alpha(k))*sin(theta1213)*sin(2*pwd.beta(k)));


WB(k)=(1/(4096*omegar))*d13*d23*sin(pwd.alpha(k))*sin(pwd.beta(k))*(-
14*sin(2*theta1223)*((3+cos(2*pwd.alpha(k)))*cos(2*pwd.beta(k))+2*sin(pwd.a
lpha(k))^2)+...
             sin(2*theta1213)*(7*cos(2*pwd.alpha(k))*(-
1+2*cos(2*theta1223)+2*cos(2*pwd.beta(k)))+2*(7+29*cos(2*theta1223)-...
             7*cos(pwd.alpha(k)-
2*pwd.beta(k))+7*cos(2*pwd.beta(k))*(3+2*cos(2*theta1223)*sin(pwd.alpha(k))
^2)))+...
             7*sin(2*(-theta1213+pwd.alpha(k)))-14*sin(2*theta1213-
pwd.alpha(k)-2*pwd.beta(k))+...
             cos(2*theta1213)*(-
2*sin(2*theta1223)*(29+7*cos(2*pwd.alpha(k))+14*cos(2*pwd.beta(k))*sin(pwd.
alpha(k))^2)-...
             7*(sin(2*pwd.alpha(k))+2*sin(pwd.alpha(k)-
2*pwd.beta(k))))+28*cos(pwd.alpha(k))*(-
2*cos(2*theta1223)*sin(2*pwd.beta(k))+sin(2*(theta1213+pwd.beta(k)))));


WC(k)=(1/(256*omegar))*d12*d23*sin(theta1223)*sin(pwd.alpha(k))*sin(pwd.bet
a(k))*(cos(theta1223)*(9+7*cos(2*pwd.beta(k)))-
7*cos(pwd.alpha(k))*sin(theta1223)*sin(2*pwd.beta(k)));

         Heff=zeros(8,8);   % Preallocate Effective Hamiltonian of 3 spin
system

% Calculate effective Hamiltonian of 3 spin system (DD part)
```

```
Heff(2,3)=-i*(WA(k)+2*(WB(k)+WC(k)));
Heff(2,5)=-i*(2*WA(k)-2*WB(k)+WC(k));
Heff(3,5)=i*(2*WA(k)+WB(k)-2*WC(k));
Heff(4,6)=-i*(2*WA(k)+WB(k)-2*WC(k));
Heff(4,7)=i*(2*WA(k)-2*WB(k)+WC(k));
Heff(6,7)=i*(WA(k)+2*(WB(k)+WC(k)));

Heff(3,2)=i*(WA(k)+2*(WB(k)+WC(k)));
Heff(5,2)=i*(2*WA(k)-2*WB(k)+WC(k));
Heff(5,3)=-i*(2*WA(k)+WB(k)-2*WC(k));
Heff(6,4)=i*(2*WA(k)+WB(k)-2*WC(k));
Heff(7,4)=-i*(2*WA(k)-2*WB(k)+WC(k));
Heff(7,6)=-i*(WA(k)+2*(WB(k)+WC(k)));

        Heff2=Heff;

        % Add CS Hamiltonian to effective Hamiltonian and generate full
Hamiltonian
        HeffCS=Heff2+CSHam;

        % Diagonalize Effective Hamiltonian
        [V,D]=eig(HeffCS);

        % Normalize Eigenvectors
        for m=1:length(V)
            V(:,m)=V(:,m)/norm(V(:,m));
        end
       % Transform Starting and Detection operators into the Eigenbase of
the Hamiltonian
       FEB=inv(V)*Fdet*V;
       FEB2=inv(V)*sigma0*V;

       % Generate Stick Spectrum from Hamiltonian and Compute
       % M2(DD+CS) using the definition from Abgraam
        count=1;
         for m=1:8
             for s=(m+1):8
                   transitionf(m,s)=-D(m,m)+D(s,s); % Transition frequency
                   % extracted from difference of eigenvalues
                   Intensity(m,s)=real(FEB(m,s)*(FEB2(s,m)));
                   % Intensity extracted from off-diagonal elements of
                   % Starting and Detection operators in the eigenbasis of
                   % the Hamiltonian

           if (CS2==0 || CS3==0)
                  fw=bw;
           else
                  fw=bw;
           end


                   if (CS2==0 || CS3==0)
                  % Routine to only include all resonances within the
                  % frequency bandwidth filter fw (in our case fw=50 Hz)
                  % Note: no intensity filtering applied, but the option
exists.
                       if (abs(transitionf(m,s)-Omega1)< fw &&
Intensity(m,s)>0.0)

                           M2w(k)=M2w(k)+(-D(m,m)+D(s,s)-
Omega1)^2*Intensity(m,s);
                       else
```

```
                    end
                else

                if (abs(transitionf(m,s)-Omega1)< fw &&
Intensity(m,s)>0.0)

                        M2w(k)=M2w(k)+(-D(m,m)+D(s,s)-
Omega1)^2*Intensity(m,s);

                else
                    M2w(k)=M2w(k)+(-D(m,m)+D(s,s)-Omega1)^2*0;
                end
                end

              count=count+1;

            end
         end

         M2w;
        end

% Perform Powder averaging
for k=1:pwd.N
         M2=M2+M2w(k)*pwd.weight(k);
end

% Calculate scaling factor scale=M2(DD+CS)/M2(DD)
M23spin=M2;
scale=M23spin/M2FxIx;

end
```

S10.3.3 Additional Useful Functions

Calculation of Full Triangle Geometry, given (r12,r13,theta1213) and using the Cosine Law

```
%% CosLaw For 1213 triangle with
% r12=a; r13=b; theta1213=gamma;
% r23=c; theta1323=alpha; theta1223=beta

function [r23,theta1223,theta1323]=TriangleSolver1213b(r12,r13,theta1213)

r23=sqrt(r12^2+r13^2-2*r12*r13*cos(theta1213));
%r23=c;
theta1223=acos((r12^2+r23^2-r13^2)/(2*r12*r23));

theta1323=pi-theta1213-theta1223;

%theta1323=alpha;

end
```

Calculation of Spin Operators for N spin system (n_spin_system.m)

Note: Written and Published by A.A.Smith (see references within the code)

```
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
%
%
%   Supplementary information for :
%   Characterization of Fibril Dynamics on Three Timescales by Solid-State
%   NMR
%
%   J. Biomol. NMR
%
%   A. Smith, E. Testori, R. Cadalbert, B. Meier*, M. Ernst*
%
%   B.M.: beme@ethz.ch
%   M.E.: maer@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
```

Calculation of powder Average for given Quality Factor (pwd_avg.)
Note: Written and Published by A.A.Smith (see references within the code)

```
function [ varargout ] = pwd_avg( q,sym )
%PWD_AVG Generates a powder average with quality q (q=1 to 12).
%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
%
%
%   Supplementary information for :
%   Characterization of Fibril Dynamics on Three Timescales by Solid-State
%   NMR
%
%   J. Biomol. NMR
%
%   A. Smith, E. Testori, R. Cadalbert, B. Meier*, M. Ernst*
%
%   B.M.: beme@ethz.ch
%   M.E.: maer@ethz.ch

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
```