Electronic Supplementary Information for

Rocking motion in solid proteins studied by the ^{15}N proton-decoupled $R_{1\rho}$ relaxometry.

Alexey Krushelnitsky,* Günter Hempel, Hannes Jurack, Tiago Ferreira

Institut für Physik, Martin-Luther-Universität Halle-Wittenberg, Betty-Heimann-Str. 7, 06120, Halle (Saale), Germany

krushelnitsky@physik.uni-halle.de

1 ANALYSIS OF THE $R_{1\rho}$ DECAYS

As a typical example, we show the analysis of the standard and proton-decoupled $R_{1\rho}$ decays measured in the microcrystalline sample at T=19 ºC and spin lock field 14 kHz. Usually, we measured the signal amplitudes after 128 logarithmically spaced spin lock pulse delays from 0.4-0.8 ms to 24-28 ms. Such a high time resolution of the measurement enables decreasing the "dead time" of the decays caused by the initial oscillations using the adjacent averaging (smoothing) algorithm. Fig. S1 compares the raw and smoothed decays, from which it can be seen that the "dead time" can be decreased from ~2 ms to ~1 ms and from ~3.5 ms to ~2 ms for the proton-decoupled and standard $R_{1\rho}$ decays, respectively.

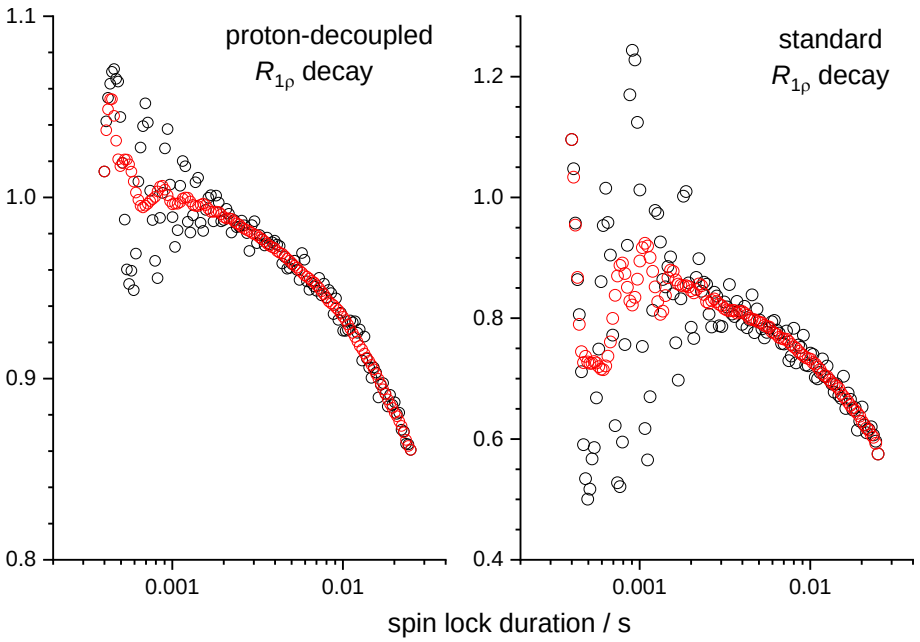

Figure S1. Experimental (black circles) and smoothed (red circles) $R_{1\rho}$ decays. Smoothing was performed as averaging over 15 adjacent points using Origin software. The decays were arbitrarily normalised.

After smoothing, the decays were fitted using a formula consisting of sum of five components:

$$A(t) = A_0\left[0.1\cdot\exp(-\sigma^2 Rt) + 0.2\cdot\exp(-\sigma Rt) + 0.4\cdot\exp(-Rt) + \right.$$
$$\left. +0.2\cdot\exp(-Rt/\sigma) + 0.1\cdot\exp(-Rt/\sigma^2)\right] \qquad \text{(S1)}$$

with A_0, R and σ being variable fitting parameters. Parameter σ plays a role of a width of the relaxation rates distribution. The mean relaxation rate (initial slope of the decay) is determined as

$$\left\langle R_{1\rho}\right\rangle = R\cdot\left(0.1\cdot\sigma^2 + 0.2\cdot\sigma + 0.4 + 0.2/\sigma + 0.1/\sigma^2\right) \qquad \text{(S2)}$$

The coefficients 0.1, 0.2, 0.4, 0.2, and 0.1 were chosen arbitrarily. However, this practically does not affect the final result. Another set of these coefficients (we tried e.g. 0.05, 0.15, 0.6, 0.15. 0.05) provides essentially the same mean relaxation rate.

The phenomenological formula (Eq. S1), in many cases, could not describe the whole measured decay very well since the shape of the decay can be more complicated than the one described by Eq. S1. In this case, the fitting range of the decay was truncated down to 10-15 ms, which was usually enough for reasonably good fitting. Fitting the smoothed decays provides quite a small fitting error of $\langle R_{1\rho} \rangle$. The small error, however, looks unrealistic since the initial slope is invisible because of the initial oscillations. While fitting, we extrapolate the decay to zero time, and such extrapolation is obviously uncertain. To estimate a more realistic fitting uncertainty, we fitted the decays at various fixed values of the parameter σ that provide visually the same fitting quality of the decay. The difference between the mean relaxation rates obtained at the minimum and maximum values of σ that provide reasonably good fitting, comprised the fitting uncertainty indicated in Fig. 11 of the main paper. This is illustrated in Fig. S2, which shows two options for fitting the smoothed decays with two different σ values. Still, we need to admit that because of the "dead time", determination of the mean relaxation rate may have a certain systematic error, which is nevertheless obviously relatively small.

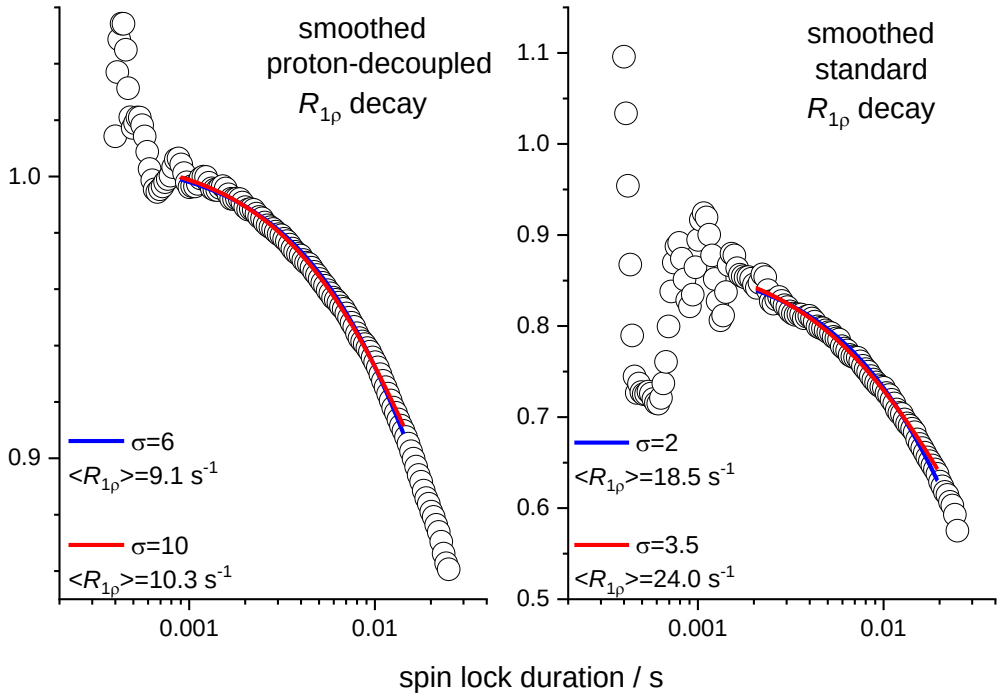

Figure S2. Fitting the smoothed decays using Eq. S1. The colour lines are the fitting curves, the corresponding σ and $\langle R_{1\rho} \rangle$ values are indicated in the figure.

2 FITTING THE DATA FOR THE MICROCRYSTALLINE SAMPLE ASSUMING A DISTRIBUTION OF THE CORRELATION TIMES FOR THE FAST COMPONENT OF THE CORRELATION FUNCTION

For fitting the data assuming a τ_F -distribution, Eq. 8 of the main paper should be modified as:

$$J(\omega) = \left(1 - S^2\right)\left[A\int P(\tau, \tau_F)\frac{\tau}{1+\left(\omega\tau\right)^2} d\tau + (1-A)\frac{\tau_S}{1+\left(\omega\tau_S\right)^2}\right] \quad \text{(S3)}$$

where $P(\tau, \tau_F)$ is the distribution function, which we assume to be log-normal:

$$P(\tau, \tau_F) = \frac{1}{\sqrt{2\pi}\tau\sigma}\exp\left[-\frac{\left(\ln(\tau) - \ln(\tau_F)\right)^2}{2\sigma^2}\right] \quad \text{(S4)}$$

In this way, we introduce one more fitting parameter, the distribution width σ. Then, using Eqs. S3 and S4, we fitted the $R_{1\rho}$ data measured at 29 ºC with the fixed value S^2=0.975, which corresponds to the order parameter obtained at T= 9 ºC (see Tab. 2 of the main paper). We got practically the same fitting error and the following set of the fitting parameters: A=0.67±0.1, τ_F=(0.23±0.12) μs, τ_S=(0.78±0.25) ms, σ=1.95±0.2. Thus, the unnatural temperature dependence of the order parameter in the microcrystalline sample (the decrease of the motional amplitude with increasing temperature) can be explained. The τ_F-distribution can also explain the very weak temperature dependence of the correlation time τ_F. To demonstrate this, we fitted the $R_{1\rho}$ data measured at T= 9 ºC using Eqs. S3 and S4 with fixed S^2=0.975 and σ=1.95; that is, we assumed that at lower temperature the τ_F-distribution has the same distribution width. We could also obtain good fitting with the following set of the remaining fitting parameters: A=0.73±0.03, τ_F=(0.43±0.03) μs, τ_S=(0.66±0.07) ms. So, now τ_F decreases about two times, and S^2 stays the same upon increasing temperature from 9 ºC to 29 ºC.

Qualitatively, the apparent effect of decreasing motional amplitude with increasing temperature (see Tab.2 of the main paper) can be explained by a limited dynamic window of the $R_{1\rho}$ experiment. This window is determined by the combinations of the MAS and spin-lock frequencies, at which the spectral density function is being sampled in the $R_{1\rho}$ experiment, see Eqs. 1-3 of the main paper. If a temperature variation shifts a correlation time of a motion out of this window, the motion becomes "invisible" for the experiment. This is what we most probably observe in the microcrystalline sample: τ_F at higher temperature becomes too short to contribute to the relaxation rate and thus, the apparent amplitude of the motion becomes smaller. This is not the case for the powder sample since τ_F for this sample is about one order of magnitude longer than for the microcrystalline one. Hence, temperature increase up to 29 ºC cannot shift the fast component out of the dynamic window.

Fig. S3 presents the simulated $R_{1\rho}$ dispersions for two cases, assuming no correlation times distributions (Eq. 8 of the main paper) and assuming log-normal τ_F-distribution (Eqs. S3 and S4) for two temperatures. One may see that solid and dashed curves are practically indistinguishable within the range of the spin lock frequencies used in our experiments. They differ

only at much larger frequency differences (ν_{SL}-ν_{MAS}), which are more difficult to achieve in a real experiment. We note that the values of the fitting parameters obtained assuming τ_F-distribution are not "more correct" compared to the data presented in Tab. 2 of the main paper since fitting was performed with quite arbitrary fixing some of the parameters and using phenomenological distribution function. This is only an example demonstrating how correlation time distribution may explain the apparent unnatural temperature behaviour of some dynamic parameters.

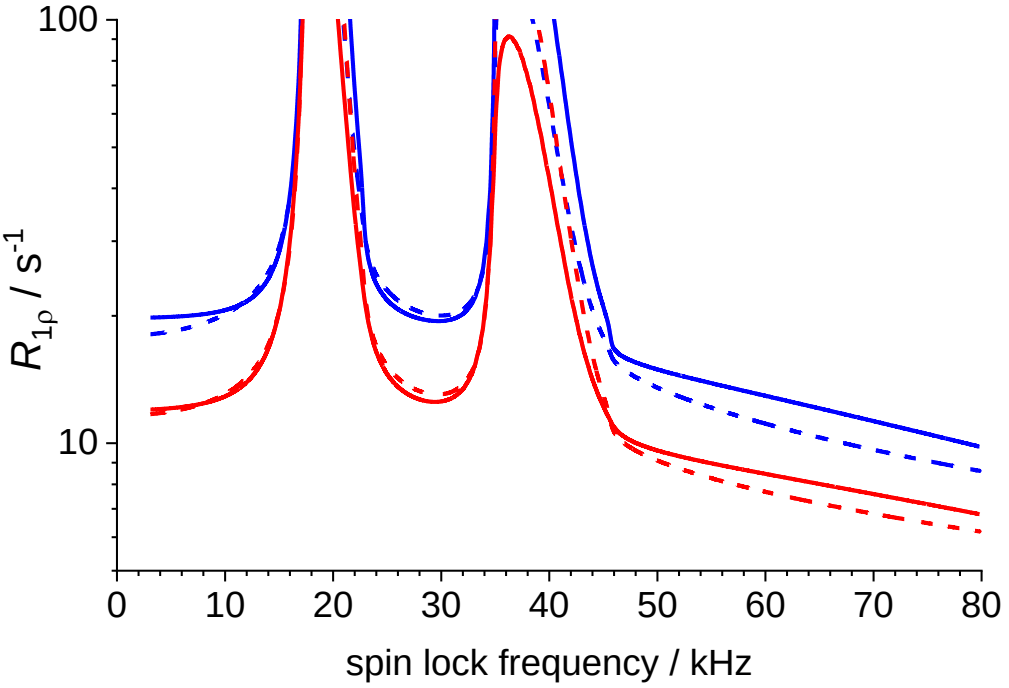

Figure S3. Simulated $R_{1\rho}$ dispersions (standard $R_{1\rho}$, without proton decoupling) for the microcrystalline sample at temperatures 9 ºC (blue lines) and 29 ºC (red lines). Solid and dashed lines correspond to the simulations assuming no correlation times distributions (Eq. 8 of the main paper) and assuming log-normal τ_F-distribution (Eqs. S3 and S4), respectively. The simulation (fitting) parameters for the solid curves are shown in Tab.2 of the main paper, the ones for the dashed curves are mentioned in the text.

3 MEASUREMENTS OF THE MOTIONALLY AVERAGED ^{15}N-^{1}H DIPOLAR COUPLINGS

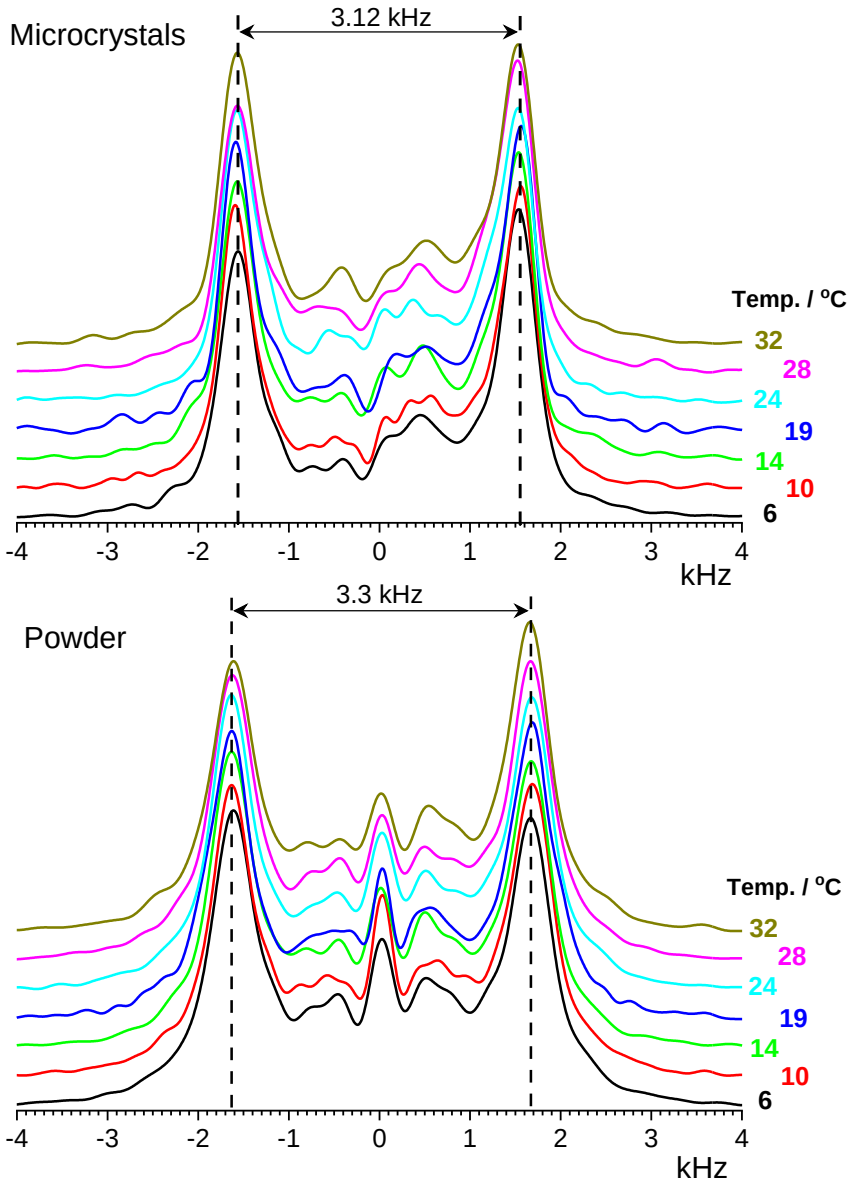

Figure S4. Projections of the Fourier-transformed dipolar modulations measured using R-PDLF pulse sequence for the microcrystalline and powder samples at different temperatures. The residual ^{15}N-^{1}H dipolar couplings were obtained by the division of the peak splittings (indicated in the figure) by the scaling factor 0.315. The parameter C_{NH}^2 (Eqs. 1 and 3 of the main paper) was calculated by multiplication of the dipolar coupling by 2π and taking square of this value. The dipolar couplings measured at the CP contact time 0.5 ms (data not shown) are practically the same.

4 HEATING EFFECT OF THE LONG ^{15}N SPIN-LOCK AND ^{1}H DECOUPLING PULSES

To quantify the heating effect caused by long RF pulses, we measured the sample temperature of microcrystalline GB1 protein after applying these pulses. This sample had a complex composition of the crystalline liquor, resulting in a ^{1}H spectrum with many sharp peaks, see Fig. S5.

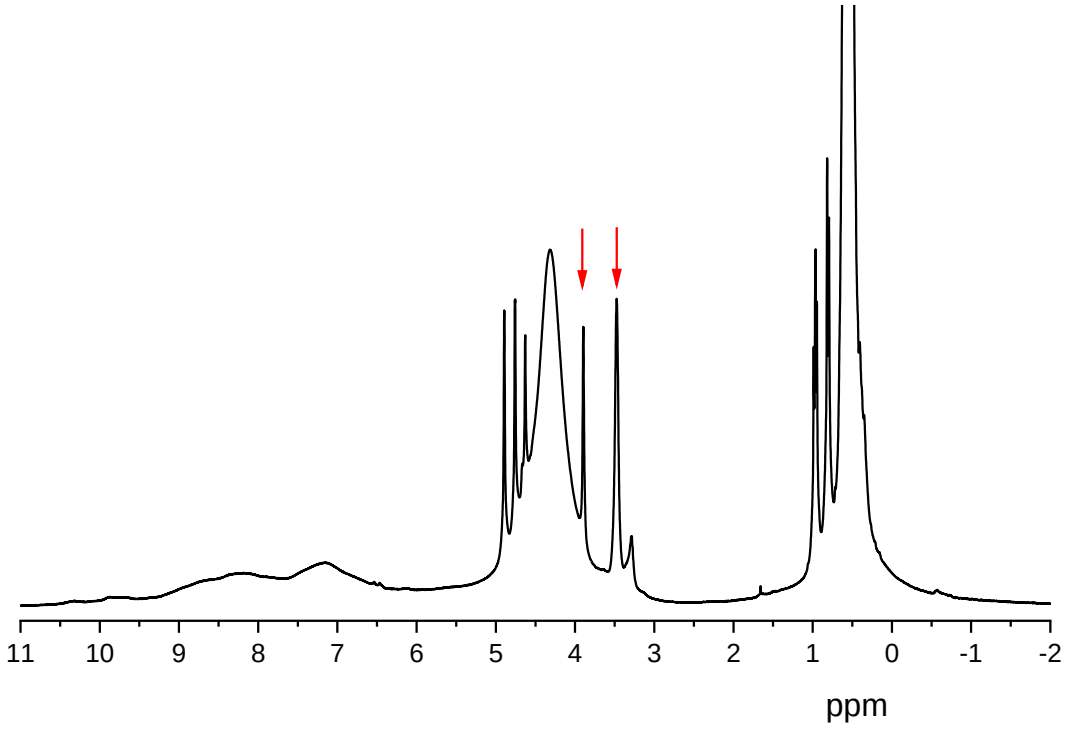

Figure S5. ^{1}H spectrum of the microcrystalline GB1 sample measured at t= 9 °C. The absolute ppm-scale has only been approximately calibrated.

Two peaks were selected (marked by red arrows in Fig. S5) and used as an internal thermometer, with the chemical shift difference between them revealing the temperature dependence (Fig. S6). The temperature slope of this distance was found to be 7.9 Hz/degree.

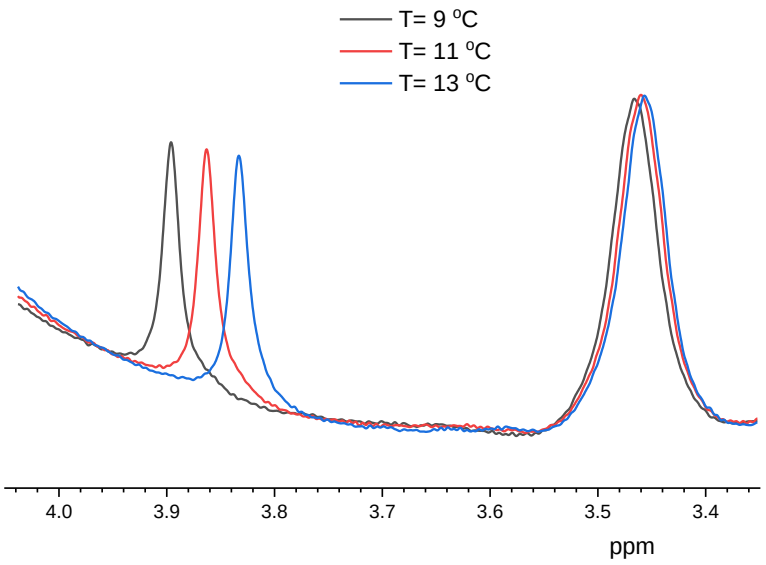

Figure S6. Zoomed region of the ^{1}H spectrum measured at three different temperatures.

The sample temperature was measured using the pulse sequence shown in Fig. S7, with maximum duration of the ^{15}N spin-lock and proton decoupling pulses used in the $R_{1\rho}$ measurements.

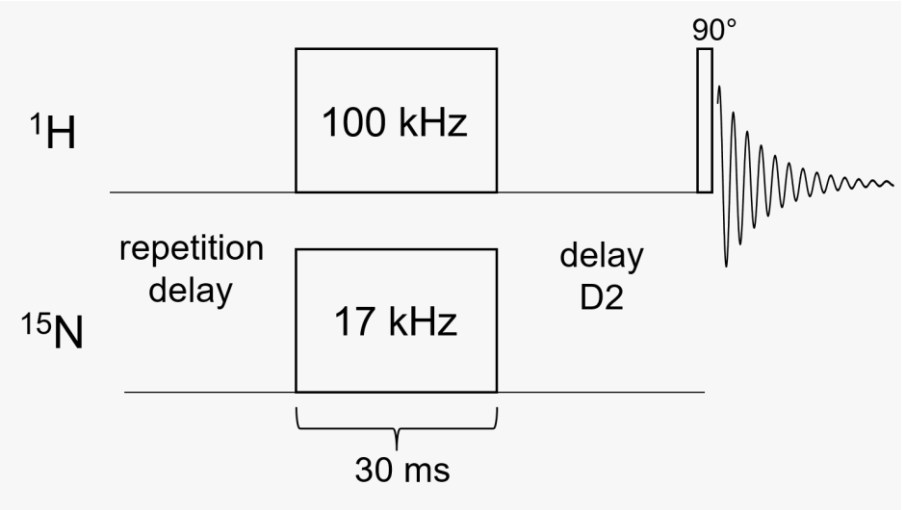

Figure S7. The pulse sequence for checking the heating effect. The repetition delay 4 s (as in the $R_{1\rho}$ experiments), number of accumulations 128.

The spectrum was measured after delay D2, which was varied from 50 ms to 4 s. The heating effect (ΔT) as a function of D2 is shown in Fig. S8, with ΔT=0 corresponding to the spectrum measured without long ^{1}H and ^{15}N pulses. These data show that the cumulative ^{1}H and ^{15}N heating at maximum pulse duration is less than 1 degree. The open circle in Fig. S8 shows that ^{15}N spin-lock heating is smaller but comparable to the proton pulse heating. Heating of the powder GB1 sample is expected to be even smaller since it contains no dissolved ions.

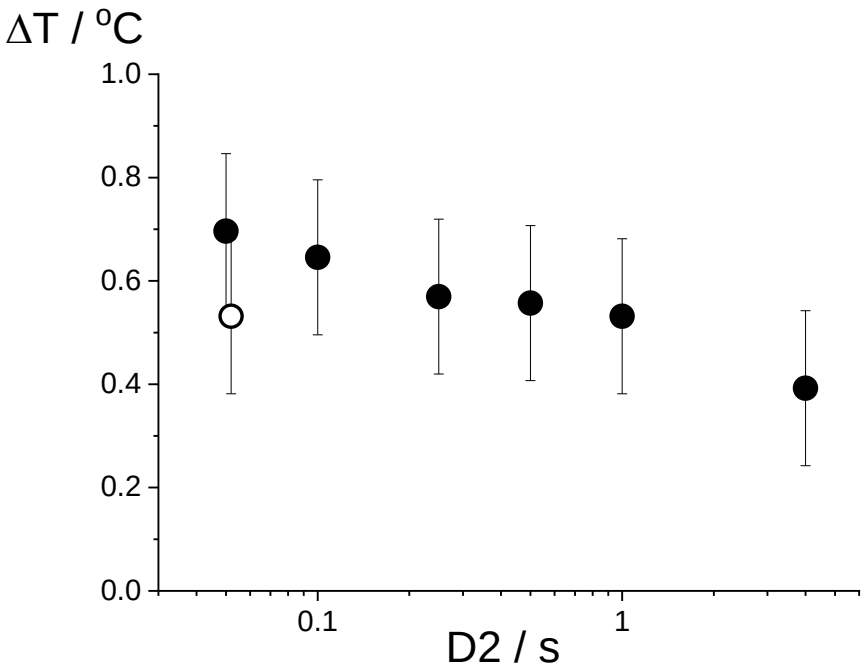

Figure S8. The heating effect (increased sample temperature after the long pulses) as a function of the delay D2 in the pulse sequence shown in Fig. S7. The open circle indicates the sample temperature after only ^{1}H pulse (without ^{15}N spin-lock). The experiments were conducted at a temperature 9 °C.

5 R_{1P} EQUATION FOR HETERONUCLEAR DIPOLAR INTERACTION BETWEEN SPINS ½ UNDER PROTON DECOUPLING AND MAS

5.1 WAY OF CALCULATION

We use quantum-mechanical master equation for the weak-collision case. That means that coherent processes are not important for the time evolution. We use the Bloch-Redfield-Wangsness approach; for some basics see [1,2,3], for an overview: [4].

$$\frac{\mathrm{d}\hat{\rho}}{\mathrm{d}t} = -\int_0^\infty \mathrm{d}\tau \left\langle \left[\hat{H}(t), \left[\hat{H}(t-\tau), \hat{\rho}(t) \right] \right] \right\rangle \tag{S5}$$

Often it can be expected that a system of equation of observables contains less scalar quations than that of the full density matrix. Suppose an observable O is represented by the operator $\hat{O}$, then

$$O = \mathrm{Tr}\left\{ \rho\, \hat{O} \right\} \quad \text{and}$$

$$\frac{\mathrm{d}O}{\mathrm{d}t} = -\int_0^\infty \mathrm{d}\tau \left\langle \mathrm{Tr}\left\{ \left[\hat{H}(t), \left[\hat{H}(t-\tau), \hat{\rho}(t) \right] \right] \hat{O} \right\} \right\rangle \tag{S6}$$

$$= -\int_0^\infty \mathrm{d}\tau\ \mathrm{Tr}\left\{ \left\langle \left[\hat{H}(t-\tau), \left[\hat{H}(t), \hat{O} \right] \right] \right\rangle \hat{\rho}(t) \right\}$$

The latter expression has been obtained under the condition that the trace of a product of matrices is invariant with respect to a cyclic permutation of its factors even if they do not commute.

The magnetization of *I* spins as well as that of *S* spins are the observables of interest here:

$$M_I = \hbar\gamma_I \,\mathrm{Tr}\left\{ \hat{\rho}\, \hat{I}_x \right\} \quad ; \quad M_S = \hbar\gamma_S \,\mathrm{Tr}\left\{ \hat{\rho}\, \hat{S}_x \right\} \tag{S7}$$

The generic average symbol ⟨⟩ includes both orientational average and time average. Following the arguments of Abragam [3], we assume that the *t*-averaging over the operator expression is performed independently on that over the density matrix because of the stationary character of the former. We emphasize it by using an extra time variable:

$$\frac{\mathrm{d}M_I}{\mathrm{d}t} = -\gamma_I \hbar \int_0^\infty \mathrm{d}\tau\ \mathrm{Tr}\left\{ \left\langle \left[\hat{H}(t_1-\tau), \left[\hat{H}(t_1), \hat{I}_x \right] \right] \right\rangle_{t_1,\theta,\varphi} \hat{\rho}(t) \right\} \tag{S8}$$

[1] N. Bloembergen, E. M. Purcell and R. V. Pound, Phys. Rev. 73 (1948) 679
[2] R. K. Wangsness, F. Phys. ev. 89 (1953) 278
[3] A. Abragam, „The Principles of Nuclear Magnetism“, Clarendon Press, Oxford 1961
[4] M. Goldman, J. Magn. Res. 149 (2001) 160

5.2 CONDITIONS OF VALIDITY

Validity of the QM master equation:

1. The basis of the wave-function space is chosen so that both $\hat{H}$ and $[\hat{H}, \hat{\rho}_0]$ average to zero over the ensemble; this corresponds to isotropic samples. The initial state corresponds to the magnetization after $\pi/2$ pulse, i.e. $\hat{\rho}_0 \sim \hat{I}_x$;
2. The motional correlation time is much shorter than the relaxation time, $\tau_c \ll T_{1\rho}$.

Additional preconditions for this derivation

- In-resonance irradiation of *S* spins as well as *I* spins;
- Spin ½ (larger spin quantum number: in most cases, quadrupolar interaction dominates the relaxation);
- Ensemble of isolated spin pairs.

5.3 CALCULATION STEPS

1. Transformation of the Hamiltonian into a basis where the first condition of 4.2 is fulfilled (details of the transformations can be found in numerous textbooks);
2. Averaging;
3. Evaluation of the commutators occurring on insertion of the selected Hamiltonian into Eq.(S6);
4. Integration;
5. Deriving the relaxation time(s) from the differential equation of magnetization(s);
6. Inserting the appropriate expressions for the spectral densities. The sample spinning (MAS) is included in this step.

5.4 HAMILTONIANS IN THE LABORYTORY FRAME

5.4.1 Total Hamiltonian

Two spin species: *I* and *S*

$$\begin{aligned} \hat{H}_\mathrm{t} &= \hat{H}_\mathrm{Z} + \hat{H}_\mathrm{rf} + \hat{H}_\mathrm{DD}\,; \\ \hat{H}_\mathrm{Z} &= -\omega_{0I}\hat{I}_z - \omega_{0S}\hat{S}_z && \leftarrow \text{Zeeman interaction} \\ \hat{H}_\mathrm{rf} &= -2\omega_{1I}\hat{I}_x \cos\omega_{0I}t - 2\omega_{1S}\hat{S}_x \cos\omega_{0S}t && \leftarrow \text{rf irradiation along } x \\ \hat{H}_\mathrm{DD} & && \leftarrow \text{dipolar interaction; see below} \end{aligned} \tag{S9}$$

5.4.2 Dipolar Hamiltonian

In their famous paper from 1948, Bloembergen, Purcell and Pound [1] decomposed the Hamiltonian in the laboratory frame into six terms („dipolar alphabet“); for spin pairs:

$$\hat{H}_\mathrm{DD} = C_{IS}\left(\hat{A} + \hat{B} + \hat{C} + \hat{D} + \hat{E} + \hat{F}\right) \quad ; \quad C_{IS} := \frac{\hbar\gamma_I\gamma_S}{r^3} \tag{S10}$$

which have different effect on the spin system:

$$\begin{array}{ll}
\hat{A} = \left(1-3\cos^2\theta\right)\hat{I}_z\hat{S}_z & \Delta m = 0 \\
\hat{B} = -\frac{1}{4}\left(1-3\cos^2\theta\right)\left(\hat{I}_+\hat{S}_- + \hat{I}_-\hat{S}_+\right) & \Delta m = 0 \\
\hat{C} = -\frac{3}{2}\sin\theta\cos\theta\, e^{-i\varphi}\left(\hat{I}_+\hat{S}_z + \hat{I}_z\hat{S}_+\right) & \Delta m = +1 \\
\hat{D} = -\frac{3}{2}\sin\theta\cos\theta\, e^{+i\varphi}\left(\hat{I}_-\hat{S}_z + \hat{I}_z\hat{S}_-\right) & \Delta m = -1 \\
\hat{E} = -\frac{3}{4}\sin^2\theta\, e^{-2i\varphi}\ \hat{I}_+\hat{S}_+ & \Delta m = +2 \\
\hat{F} = -\frac{3}{4}\sin^2\theta\, e^{+2i\varphi}\ \hat{I}_-\hat{S}_- & \Delta m = -2
\end{array} \qquad \text{(S11)}$$

Each of these terms can be separated into both geometrical function F_q and operator component $\hat{A}_q$:

$$\hat{H}_{\mathrm{DD}} = C_{IS}\sum_{q=-2}^{2} F_q\hat{A}_q \qquad \text{(S12)}$$

where

$$F_0(t) = \left(1-3\cos^2\theta\right); \qquad F_1(t) = -\frac{3}{2}\sin\theta\cos\theta\, e^{-i\varphi}; \qquad F_2(t) = -\frac{3}{4}\sin^2\theta\, e^{-2i\varphi} \qquad \text{(S13)}$$
$$F_{-q} = F_q{}^*$$

$$\begin{aligned}
\hat{A}_0 &= \hat{I}_z\hat{S}_z - \frac{1}{4}\left(\hat{I}_+\hat{S}_- + \hat{I}_-\hat{S}_+\right) = \hat{I}_z\hat{S}_z - \frac{1}{2}\left(\hat{I}_x\hat{S}_x + \hat{I}_y\hat{S}_y\right) \\
\hat{A}_1 &= \hat{I}_+\hat{S}_z + \hat{I}_z\hat{S}_+ \\
\hat{A}_2 &= \hat{I}_+\hat{S}_+ \\
\hat{A}_{-q} &= \hat{A}_q^{\dagger}
\end{aligned} \qquad \text{(S14)}$$

5.5 HAMILTONIANS IN THE ROTATING FRAME

5.5.1 Transformation

The static Zeeman part of the Hamiltonian is removed by the following transformation applied to Hamiltonian as well as density matrix:

$$\begin{aligned}
\hat{\rho}_{\mathrm{R}} &= \exp\left(i\,\hat{H}_Z t\right)\hat{\rho}_{\mathrm{LKS}}\exp\left(-i\,\hat{H}_Z t\right) \\
\hat{H}_{\mathrm{R}} &= \exp\left(i\,\hat{H}_Z t\right)\left(\hat{H}_{\mathrm{rf}} + \hat{H}_{\mathrm{DD}}\right)\exp\left(-i\,\hat{H}_Z t\right)
\end{aligned} \qquad \text{(S15)}$$

Product-operator rules can be applied to transform linear and bilinear terms of the spin-operator components.

5.5.2 rf part

$$\hat{H}_{\mathrm{R,rf}} = -\omega_{1I}\hat{I}_x - \omega_{1S}\hat{S}_x \quad + \text{ components rotating with } 2\omega_{0I;S} \qquad \text{(S16)}$$

The counter-rotating components are neglected as usually.

5.5.3 Components of the dipolar Hamiltonian

The transformation of the five operator parts gives

$$\begin{aligned}
\hat{A}_0 &\rightarrow \hat{A}_{\mathrm{R}0} = \hat{I}_z\hat{S}_z - \frac{1}{4}\hat{I}_+\hat{S}_- \, e^{-i(\omega_{0I}-\omega_{0S})t} - \frac{1}{4}\hat{I}_-\hat{S}_+ \, e^{-i(\omega_{0S}-\omega_{0I})t} \\
\hat{A}_1 &\rightarrow \hat{A}_{\mathrm{R}1} = \hat{I}_+\hat{S}_z \, e^{-i\omega_{0I}t} + \hat{I}_z\hat{S}_+ \, e^{-i\omega_{0S}t} \\
\hat{A}_{-1} &\rightarrow \hat{A}_{\mathrm{R}\text{-}1} = \hat{I}_-\hat{S}_z \, e^{i\omega_{0I}t} + \hat{I}_z\hat{S}_- \, e^{i\omega_{0S}t} \\
\hat{A}_2 &\rightarrow \hat{A}_{\mathrm{R}2} = \hat{I}_+\hat{S}_+ \cdot e^{-i(\omega_{0I}+\omega_{0S})t} \\
\hat{A}_{-2} &\rightarrow \hat{A}_{\mathrm{R}\text{-}2} = \hat{I}_-\hat{S}_- \cdot e^{i(\omega_{0I}+\omega_{0S})t}
\end{aligned} \tag{S17}$$

5.6 DOUBLE-ROTATING FRAME

5.6.1 Transformation

To remove any constant part of $\hat{H}_{\mathrm{R}}$, the rf part must be eliminated. This is performed by the transformation

$$\hat{\sigma} = \exp\left(-i\left(\omega_{1I}\hat{I}_x + \omega_{1S}\hat{S}_x\right)t\right)\hat{\rho}\,\exp\left(+i\left(\omega_{1I}\hat{I}_x + \omega_{1S}\hat{S}_x\right)t\right) \tag{S18}$$

$$\hat{X}_{\mathrm{DR}} = \exp\left(-i\left(\omega_{1I}\hat{I}_x + \omega_{1S}\hat{S}_x\right)t\right)\hat{X}_{\mathrm{R}}\,\exp\left(+i\left(\omega_{1I}\hat{I}_x + \omega_{1S}\hat{S}_x\right)t\right) \tag{S19}$$

where $\hat{X}$ represents a generic operator.

5.6.2 Components of the dipolar Hamiltonian

This transformation gives

$$\begin{aligned}
\hat{A}_{\mathrm{DR}0} &= a + b\,e^{i(\omega_{0I}-\omega_{0S})t} + b^{\dagger}\,e^{-i(\omega_{0I}-\omega_{0S})t} \\
\hat{A}_{\mathrm{DR}1} &= c\,e^{i\omega_{0I}t} + d\,e^{i\omega_{0S}t} \\
\hat{A}_{\mathrm{DR}2} &= e^{i(\omega_{0I}+\omega_{0S})t}\left\{2\left[\hat{I}_x\hat{S}_x + \hat{Q}_+\cos\left(\omega_{1I}+\omega_{1S}\right)t + \hat{R}_+\sin\left(\omega_{1I}+\omega_{1S}\right)t\right.\right. \\
&\qquad \left.\left. - \hat{Q}_-\cos\left(\omega_{1I}-\omega_{1S}\right)t - \hat{R}_-\sin\left(\omega_{1I}-\omega_{1S}\right)t\right] \right. \\
&\qquad \left. + i\left(\hat{I}_y\hat{S}_x\cos\omega_{1I}t - \hat{I}_z\hat{S}_x\sin\omega_{1I}t + \hat{I}_x\hat{S}_y\cos\omega_{1S}t - \hat{I}_x\hat{S}_z\sin\omega_{1S}t\right)\right\} \\
\hat{A}_{\mathrm{DR}\,-q} &= \hat{A}^{\dagger}_{\mathrm{DR}\,q}
\end{aligned} \tag{S20}$$

with

$$\begin{aligned}
\hat{a} := {} & \hat{Q}_+\cos\left(\omega_{1I}+\omega_{1S}\right)t + \hat{Q}_-\cos\left(\omega_{1I}-\omega_{1S}\right)t \\
& + \hat{R}_+\sin\left(\omega_{1I}+\omega_{1S}\right)t + \hat{R}_-\sin\left(\omega_{1I}-\omega_{1S}\right)t
\end{aligned} \tag{S21}$$

$$\hat{b} := -\frac{1}{4}\Big\{\hat{I}_x\hat{S}_x - \hat{Q}_+\cos\big(\omega_{1I}+\omega_{1S}\big)t + \hat{Q}_-\cos\big(\omega_{1I}-\omega_{1S}\big)t$$
$$- \hat{R}_+\sin\big(\omega_{1I}+\omega_{1S}\big)t + \hat{R}_-\sin\big(\omega_{1I}-\omega_{1S}\big)t \qquad \text{(S22)}$$
$$+ i\Big(\hat{I}_y\hat{S}_x\cos\omega_{1I}t - \hat{I}_z\hat{S}_x\sin\omega_{1I}t - \hat{I}_x\hat{S}_y\cos\omega_{1S}t + \hat{I}_x\hat{S}_z\sin\omega_{1S}t\Big)\Big\}$$

$$\hat{c} := \Big\{i\Big[\hat{R}_+\cos\big(\omega_{1I}+\omega_{1S}\big)t - \hat{Q}_+\sin\big(\omega_{1I}+\omega_{1S}\big)t$$
$$+ \hat{R}_-\cos\big(\omega_{1I}-\omega_{1S}\big)t - \hat{Q}_-\sin\big(\omega_{1I}-\omega_{1S}\big)t\Big] \qquad \text{(S23)}$$
$$+\hat{I}_x\hat{S}_z\cos\omega_{1S}t + \hat{I}_x\hat{S}_y\sin\omega_{1S}t\Big\}$$

$$\hat{d} := \Big\{i\Big[\hat{R}_+\cos\big(\omega_{1I}+\omega_{1S}\big)t - \hat{Q}_+\sin\big(\omega_{1I}+\omega_{1S}\big)t$$
$$- \hat{R}_-\cos\big(\omega_{1I}-\omega_{1S}\big)t + \hat{Q}_-\sin\big(\omega_{1I}-\omega_{1S}\big)t\Big] \qquad \text{(S24)}$$
$$+\hat{I}_z\hat{S}_x\cos\omega_{1I}t + \hat{I}_y\hat{S}_x\sin\omega_{1I}t\Big\}$$

and the shortcuts

$$\hat{Q}_\pm := \frac{1}{2}\big(\hat{I}_z\hat{S}_z \mp \hat{I}_y\hat{S}_y\big); \quad \hat{R}_\pm := \frac{1}{2}\big(\hat{I}_y\hat{S}_z \pm \hat{I}_z\hat{S}_y\big) \qquad \text{(S25)}$$

5.7 ENSEMBLE AVERAGE AND INTEGRATION

5.7.1 Averaging toward correlation function

Insertion of the Hamiltonian into Eq. (S8):

$$\frac{\mathrm{dM}_I}{\mathrm{d}t} = -\gamma_I\hbar\int_0^\infty \mathrm{d}\tau\ \mathrm{Tr}\Big\{\Big\langle\Big[\hat{H}(t_1-\tau),\Big[\hat{H}(t_1),\hat{I}_x\Big]\Big]\Big\rangle_{t_1,\theta,\varphi}\rho(t)\Big\}$$
$$= -\hbar\gamma_I C_{IS}^2\int_0^\infty \mathrm{d}\tau\ \mathrm{Tr}\Big\{\sum_{p,q=-2}^{2}\Big\langle\big\langle F_q(t-\tau)F_p(t)\big\rangle_{\theta,\varphi}\Big[\hat{A}_{\mathrm{DR}\,q}(t_1-\tau),\Big[\hat{A}_{\mathrm{DR}\,p}(t_1),\hat{I}_x\Big]\Big]\Big\rangle_{t_1}\rho(t)\Big\} \qquad \text{(S26)}$$

The orientation average is restricted to the geometrical functions only.

5.7.2 Correlation functions

We assume stationary behaviour of the thermal motion. This means that the orientation average must not depend on *t*. The geometrical parts form the products $F_q(0)F_p(0)$. Their φ dependency is described by the factor $\exp\big(-i(p+q)\varphi\big)$ which follows from the definition of the F_p. Hence $\big\langle F_q(0)F_p(0)\big\rangle$ is zero for an isotropic sample unless $p = -q$. Consequently, only the following terms which are the autocorrelation functions survive the averaging:

$$\begin{aligned}
\left\langle F_0(t-\tau)F_0(t)\right\rangle &= \left\langle \left|F_0\right|^2\right\rangle = K_0(\tau); & K_0(0) &= \left\langle \left|F_0\right|^2\right\rangle = \frac{4}{5} \\
\left\langle F_1(t-\tau)F_1(t)\right\rangle &= \left\langle F_1(t-\tau)F_{-1}{}^*(t)\right\rangle = K_1(\tau); & K_1(0) &= \left\langle \left|F_1\right|^2\right\rangle = \frac{3}{10} \\
\left\langle F_2(t-\tau)F_{-2}(t)\right\rangle &= \left\langle F_2(t-\tau)F_2{}^*(t)\right\rangle = K_2(\tau); & K_2(0) &= \left\langle \left|F_2\right|^2\right\rangle = \frac{3}{10}
\end{aligned} \tag{S27}$$

Therefore the orientational averaging reduces the number of terms if the sample is isotropic:

$$\frac{\mathrm{dM}_I}{\mathrm{d}t} = -C_{IS}^2 \sum_{q=-2}^{2} \int_0^\infty K_{|q|}(\tau)\,\mathrm{d}\tau \cdot \mathrm{Tr}\left\{\left\langle \left[\hat{A}_{\mathrm{DR}\,q}(t_1-\tau), \left[\hat{A}_{\mathrm{DR}\,-q}(t_1), \hat{I}_x\right]\right]\right\rangle_{t_1} \rho(t)\right\} \tag{S28}$$

5.7.3 Particular case for $q = 0$

$$\begin{aligned}
&\left[\hat{A}_{\mathrm{DR}\,0}(t_1-\tau), \left[\hat{A}_{\mathrm{DR}\,0}(t_1), \hat{I}_x\right]\right] \\
&\quad = \left[\hat{a}(t_1-\tau), \left[\hat{a}(t_1), \hat{I}_x\right]\right] + \left[\hat{b}(t_1-\tau), \left[\hat{b}^\dagger(t_1), \hat{I}_x\right]\right] e^{-i(\omega_{0I}-\omega_{0S})\tau} \\
&\qquad + \left[\hat{b}^\dagger(t_1-\tau), \left[\hat{b}(t_1), \hat{I}_x\right]\right] e^{i(\omega_{0I}-\omega_{0S})\tau} \\
&\quad + \text{terms oscillating with } (\omega_{0I}-\omega_{0S})t_1 \text{ or } (\omega_{0I}-\omega_{0S})2t_1
\end{aligned} \tag{S29}$$

5.7.4 Particular case for $q = 1$

$$\begin{aligned}
&\left[\hat{A}_{\mathrm{DR}1}^\dagger(t_1-\tau), \left[\hat{A}_{\mathrm{DR}1}(t_1), \hat{I}_x\right]\right] = \\
&= \left[\hat{c}^\dagger(t_1-\tau)e^{-i\omega_{0I}(t_1-\tau)} + \hat{d}^\dagger(t_1-\tau)e^{-i\omega_{0S}(t_1-\tau)}, \left[\hat{c}\,e^{i\omega_{0I}t_1} + \hat{d}\,e^{i\omega_{0S}t_1}, \hat{I}_x\right]\right] \\
&= \left[\hat{c}^\dagger(t_1-\tau), \left[\hat{c}(t_1), \hat{I}_x\right]\right] e^{i\omega_{0I}\tau} + \left[\hat{d}^\dagger(t_1-\tau), \left[\hat{d}(t_1), \hat{I}_x\right]\right] e^{i\omega_{0S}\tau} \\
&\quad + \text{terms oscillating vs. } t_1
\end{aligned} \tag{S30}$$

5.8 EVALUATION OF THE COMMUTATORS

5.8.1 Helpful commutator rules

Commutator rules for the shortcuts defined above can be derived from the well-known spin-operator rules:

$$\left[\hat{Q}_-, \hat{I}_x\right] = i\hat{R}_-; \qquad \left[\hat{R}_-, \hat{I}_x\right] = -i\hat{Q}_-; \qquad \left[\hat{Q}_+, \hat{I}_x\right] = i\hat{R}_+; \qquad \left[\hat{R}_+, \hat{I}_x\right] = -i\hat{Q}_+$$

$$\left[\hat{Q}_+, \hat{R}_+\right] = -\frac{i}{8}\left(\hat{I}_x + \hat{S}_x\right); \qquad \left[\hat{Q}_-, \hat{R}_-\right] = -\frac{i}{8}\left(\hat{I}_x - \hat{S}_x\right)$$

$$\left[\hat{Q}_+, \hat{R}_-\right] = \left[\hat{Q}_-, \hat{R}_+\right] = \left[\hat{Q}_+, \hat{Q}_-\right] = \left[\hat{R}_-, \hat{R}_+\right] = 0$$

5.8.2 Inner commutators

$$\begin{aligned}\left[\hat{a}(t_1),\hat{I}_x\right] &= \Big[\hat{Q}_+\cos(\omega_{1I}+\omega_{1S})t_1+\hat{Q}_-\cos(\omega_{1I}-\omega_{1S})t_1 \\ &\qquad +\hat{R}_+\sin(\omega_{1I}+\omega_{1S})t_1+\hat{R}_-\sin(\omega_{1I}-\omega_{1S})t_1\,,\,\hat{I}_x\Big] \\ &= i\Big\{\hat{R}_+\cos(\omega_{1I}+\omega_{1S})t_1+\hat{R}_-\cos(\omega_{1I}-\omega_{1S})t_1 \\ &\qquad -\hat{Q}_+\sin(\omega_{1I}+\omega_{1S})t_1-\hat{Q}_-\sin(\omega_{1I}-\omega_{1S})t_1\Big\}\end{aligned} \tag{S31}$$

$$\begin{aligned}\left[\hat{b}(t_1),\hat{I}_x\right] &= -\frac{1}{4}\Big[\hat{I}_x\hat{S}_x-\hat{Q}_+\cos(\omega_{1I}+\omega_{1S})t_1+\hat{Q}_-\cos(\omega_{1I}-\omega_{1S})t_1 \\ &\quad -\hat{R}_+\sin(\omega_{1I}+\omega_{1S})t_1+\hat{R}_-\sin(\omega_{1I}-\omega_{1S})t_1 \\ &\quad +i\left(\hat{I}_y\hat{S}_x\cos\omega_{1I}t_1-\hat{I}_z\hat{S}_x\sin\omega_{1I}t_1-\hat{I}_x\hat{S}_y\cos\omega_{1S}t_1+\hat{I}_x\hat{S}_z\sin\omega_{1S}t_1\right),\hat{I}_x\Big] \\ &= -\frac{i}{4}\Big\{-\hat{R}_+\cos(\omega_{1I}+\omega_{1S})t_1+\hat{R}_-\cos(\omega_{1I}-\omega_{1S})t_1+\hat{Q}_+\sin(\omega_{1I}+\omega_{1S})t_1 \\ &\quad -\hat{Q}_-\sin(\omega_{1I}-\omega_{1S})t_1\Big\}+\frac{1}{4}\left(\hat{I}_z\cos\omega_{1I}t_1+\hat{I}_y\sin\omega_{1I}t_1\right)\hat{S}_x\end{aligned} \tag{S32}$$

$$\begin{aligned}\left[\hat{c}(t_1)\,,\,\hat{I}_x\right] &= \Big[i\Big\{\hat{R}_+\cos(\omega_{1I}+\omega_{1S})t_1-\hat{Q}_+\sin(\omega_{1I}+\omega_{1S})t_1 \\ &\qquad +\hat{R}_-\cos(\omega_{1I}-\omega_{1S})t_1-\hat{Q}_-\sin(\omega_{1I}-\omega_{1S})t_1\Big\} \\ &\qquad +\hat{I}_x\hat{S}_z\cos\omega_{1S}t_1+\hat{I}_x\hat{S}_y\sin\omega_{1S}t_1\,,\,\hat{I}_x\Big] \\ &= \hat{Q}_+\cos(\omega_{1I}+\omega_{1S})t_1+\hat{R}_+\sin(\omega_{1I}+\omega_{1S})t_1 \\ &\quad +\hat{Q}_-\cos(\omega_{1I}-\omega_{1S})t_1+\hat{R}_-\sin(\omega_{1I}-\omega_{1S})t_1\end{aligned} \tag{S33}$$

$$\begin{aligned}\left[\hat{d}(t_1)\,,\,\hat{I}_x\right] &= \Big[i\Big\{\hat{R}_+\cos(\omega_{1I}+\omega_{1S})t_1-\hat{Q}_+\sin(\omega_{1I}+\omega_{1S})t_1 \\ &\qquad -\hat{R}_-\cos(\omega_{1I}-\omega_{1S})t_1+\hat{Q}_-\sin(\omega_{1I}-\omega_{1S})t_1\Big\} \\ &\qquad +\hat{I}_z\hat{S}_x\cos\omega_{1S}t_1+\hat{I}_y\hat{S}_x\sin\omega_{1S}t_1\,,\,\hat{I}_x\Big] \\ &= \hat{Q}_+\cos(\omega_{1I}+\omega_{1S})t_1+\hat{R}_+\sin(\omega_{1I}+\omega_{1S})t_1 \\ &\quad -\hat{Q}_-\cos(\omega_{1I}-\omega_{1S})t_1-\hat{R}_-\sin(\omega_{1I}-\omega_{1S})t_1 \\ &\quad +i\,\hat{I}_y\hat{S}_x\cos\omega_{1S}t_1-i\,\hat{I}_z\hat{S}_x\sin\omega_{1S}t_1\end{aligned} \tag{S34}$$

$$
\begin{aligned}
&\left[\hat{A}_{\mathrm{DR2}}(t_1)\,,\hat{I}_x\right]=\\
&=e^{i(\omega_{0I}+\omega_{0S})t_1}\Big[2\Big(\hat{I}_x\hat{S}_x+\hat{Q}_+\cos\big(\omega_{1I}+\omega_{1S}\big)t_1+\hat{R}_+\sin\big(\omega_{1I}+\omega_{1S}\big)t_1\\
&\qquad-\hat{Q}_-\cos\big(\omega_{1I}-\omega_{1S}\big)t_1-\hat{R}_-\sin\big(\omega_{1I}-\omega_{1S}\big)t_1\Big)\\
&\qquad+i\Big(\hat{I}_y\hat{S}_x\cos\omega_{1I}t_1-\hat{I}_z\hat{S}_x\sin\omega_{1I}t_1+\hat{I}_x\hat{S}_y\cos\omega_{1S}t_1-\hat{I}_x\hat{S}_z\sin\omega_{1S}t_1\Big)\,,\hat{I}_x\Big]\\
&=\Big\{2i\Big(\hat{R}_+\cos\big(\omega_{1I}+\omega_{1S}\big)t_1-\hat{Q}_+\sin\big(\omega_{1I}+\omega_{1S}\big)t_1-\hat{R}_-\cos\big(\omega_{1I}-\omega_{1S}\big)t_1\\
&\qquad+\hat{Q}_-\sin\big(\omega_{1I}-\omega_{1S}\big)t_1\Big)+\hat{I}_z\hat{S}_x\cos\omega_{1I}t_1+\hat{I}_y\hat{S}_x\sin\omega_{1I}t_1\Big\}\,e^{i(\omega_{0I}+\omega_{0S})t_1}
\end{aligned}
\tag{S35}
$$

5.8.3 Outer commutators for $q = 0$

$$
\begin{aligned}
&\Big[\hat{a}\big(t_1-\tau\big),\Big[\hat{a}\big(t_1\big),\hat{I}_x\Big]\Big]=\\
&=i\Big[\hat{Q}_+\cos\big(\omega_{1I}+\omega_{1S}\big)\big(t_1-\tau\big)+\hat{Q}_-\cos\big(\omega_{1I}-\omega_{1S}\big)\big(t_1-\tau\big)\\
&\qquad+\hat{R}_+\sin\big(\omega_{1I}+\omega_{1S}\big)\big(t_1-\tau\big)+\hat{R}_-\sin\big(\omega_{1I}-\omega_{1S}\big)\big(t_1-\tau\big)\,,\\
&\qquad\hat{R}_+\cos\big(\omega_{1I}+\omega_{1S}\big)t_1+\hat{R}_-\cos\big(\omega_{1I}-\omega_{1S}\big)t_1\\
&\qquad-\hat{Q}_+\sin\big(\omega_{1I}+\omega_{1S}\big)t_1-\hat{Q}_-\sin\big(\omega_{1I}-\omega_{1S}\big)t_1\Big]\\
&=\Big\{\frac{1}{8}\Big(\hat{I}_x+\hat{S}_x\Big)\cos\big(\omega_{1I}+\omega_{1S}\big)\big(t_1-\tau\big)\cos\big(\omega_{1I}+\omega_{1S}\big)t_1\\
&\qquad+\frac{1}{8}\Big(\hat{I}_x-\hat{S}_x\Big)\cos\big(\omega_{1I}-\omega_{1S}\big)\big(t_1-\tau\big)\cos\big(\omega_{1I}-\omega_{1S}\big)t_1\\
&\qquad+\frac{1}{8}\Big(\hat{I}_x+\hat{S}_x\Big)\sin\big(\omega_{1I}+\omega_{1S}\big)\big(t_1-\tau\big)\sin\big(\omega_{1I}+\omega_{1S}\big)t_1\\
&\qquad+\frac{1}{8}\Big(\hat{I}_x-\hat{S}_x\Big)\sin\big(\omega_{1I}-\omega_{1S}\big)\big(t_1-\tau\big)\sin\big(\omega_{1I}-\omega_{1S}\big)t_1\\
&=\underline{\underline{\frac{1}{8}\Big(\hat{I}_x+\hat{S}_x\Big)\cos\big(\omega_{1I}+\omega_{1S}\big)\tau+\frac{1}{8}\Big(\hat{I}_x-\hat{S}_x\Big)\cos\big(\omega_{1I}-\omega_{1S}\big)\tau}}
\end{aligned}
\tag{S36}
$$

$$\left[\hat{b}^{\dagger}(t_1-\tau)\left[\hat{b}(t_1),\hat{I}_x\right]\right]=$$

$$\begin{aligned}
&=\frac{1}{4}\Big[-\hat{I}_x\hat{S}_x+\hat{Q}_+\cos(\omega_{1I}+\omega_{1S})(t_1-\tau)-\hat{Q}_-\cos(\omega_{1I}-\omega_{1S})(t_1-\tau)\\
&\quad+\hat{R}_+\sin(\omega_{1I}+\omega_{1S})(t_1-\tau)-\hat{R}_-\sin(\omega_{1I}-\omega_{1S})(t_1-\tau)\\
&\quad+i\Big\{\hat{I}_y\hat{S}_x\cos\omega_{1I}(t_1-\tau)-\hat{I}_x\hat{S}_y\cos\omega_{1S}(t_1-\tau)\\
&\qquad+\hat{I}_x\hat{S}_z\sin\omega_{1S}(t_1-\tau)-\hat{I}_z\hat{S}_x\sin\omega_{1I}(t_1-\tau)\Big\},\\
&\quad+\frac{i}{4}\Big\{\hat{R}_+\cos(\omega_{1I}+\omega_{1S})t_1-\hat{R}_-\cos(\omega_{1I}-\omega_{1S})t_1\\
&\qquad-\hat{Q}_+\sin(\omega_{1I}+\omega_{1S})t_1+\hat{Q}_-\sin(\omega_{1I}-\omega_{1S})t_1\Big\}\\
&\quad+\frac{1}{4}\left(\hat{I}_z\cos\omega_{1I}t_1+\hat{I}_y\sin\omega_{1I}t_1\right)\hat{S}_x\Big]
\end{aligned}\tag{S37}$$

$$\begin{aligned}
&=\frac{1}{16}\Big\{\frac{i}{4}\left(\hat{I}_y\cos\omega_{1I}t_1-\hat{I}_z\sin\omega_{1I}t_1\right)\\
&\quad+\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\cos(\omega_{1I}+\omega_{1S})(t_1-\tau)\cos(\omega_{1I}+\omega_{1S})t_1\\
&\quad+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\cos(\omega_{1I}-\omega_{1S})(t_1-\tau)\cos(\omega_{1I}-\omega_{1S})t_1\\
&\quad+\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\sin(\omega_{1I}+\omega_{1S})(t_1-\tau)\sin(\omega_{1I}+\omega_{1S})t_1\\
&\quad+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\sin(\omega_{1I}-\omega_{1S})(t_1-\tau)\sin(\omega_{1I}-\omega_{1S})t_1\\
&\quad-\frac{1}{4}\hat{I}_x\cos\omega_{1I}(t_1-\tau)\cos\omega_{1I}t_1-\frac{1}{4}\hat{I}_x\sin\omega_{1I}(t_1-\tau)\sin\omega_{1I}t_1\\
&\quad+\text{ further terms which contain a } t_1 \text{ dependence.}\\
&=\underline{\underline{\frac{1}{128}\left(\hat{I}_x+\hat{S}_x\right)\cos(\omega_{1I}+\omega_{1S})\tau+\frac{1}{128}\left(\hat{I}_x-\hat{S}_x\right)\cos(\omega_{1I}-\omega_{1S})\tau-\frac{1}{64}\hat{I}_x\cos\omega_{1I}\tau}}\\
&+\text{ terms oscillating vs. } t_1
\end{aligned}\tag{S38}$$

Insertion into Eq. (S29) and t_1-averaging:

$$\boxed{\begin{aligned}
&\left\langle\left[\hat{A}_{\mathrm{DR}0}(t-\tau),\left[\hat{A}_{\mathrm{DR}0}(t),\hat{I}_x\right]\right]\right\rangle_{t_1}=\\
&=\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\cos(\omega_{1I}+\omega_{1S})\tau+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\cos(\omega_{1I}-\omega_{1S})\tau\\
&\quad+\left\{\frac{1}{64}\left(\hat{I}_x+\hat{S}_x\right)\cos(\omega_{1I}+\omega_{1S})\tau+\frac{1}{64}\left(\hat{I}_x-\hat{S}_x\right)\cos(\omega_{1I}-\omega_{1S})\tau\right.\\
&\qquad\left.-\frac{1}{32}\hat{I}_x\cos\omega_{1I}\tau\right\}\times\cos(\omega_{0I}-\omega_{0S})\tau
\end{aligned}}\tag{S39}$$

5.8.4 Outer commutators for $q = 1$

$$
\begin{aligned}
&\left[\hat{c}^{\dagger}(t_1-\tau),\left[\hat{c}(t_1),\hat{I}_x\right]\right]= \\
&=\Big[-i\Big\{\hat{R}_+\cos(\omega_{1I}+\omega_{1S})(t_1-\tau)-\hat{Q}_+\sin(\omega_{1I}+\omega_{1S})(t_1-\tau) \\
&\quad+\hat{R}_-\cos(\omega_{1I}-\omega_{1S})(t_1-\tau)-\hat{Q}_-\sin(\omega_{1I}-\omega_{1S})(t_1-\tau)\Big\} \\
&\quad+\hat{I}_x\hat{S}_z\cos\omega_{1S}(t_1-\tau)+\hat{I}_x\hat{S}_y\sin\omega_{1S}(t_1-\tau), \\
&\quad\hat{Q}_+\cos(\omega_{1I}+\omega_{1S})t_1+\hat{R}_+\sin(\omega_{1I}+\omega_{1S})t_1 \\
&\quad+\hat{Q}_-\cos(\omega_{1I}-\omega_{1S})t_1+\hat{R}_-\sin(\omega_{1I}-\omega_{1S})t_1\Big] \\
&=\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\cos(\omega_{1I}+\omega_{1S})(t_1-\tau)\cos(\omega_{1I}+\omega_{1S})t_1 \\
&\quad+\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\sin(\omega_{1I}+\omega_{1S})(t_1-\tau)\sin(\omega_{1I}+\omega_{1S})t_1 \\
&\quad+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\cos(\omega_{1I}-\omega_{1S})(t_1-\tau)\cos(\omega_{1I}-\omega_{1S})t_1 \\
&\quad+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\sin(\omega_{1I}-\omega_{1S})(t_1-\tau)\sin(\omega_{1I}-\omega_{1S})t_1+T \qquad \text{(S40)} \\
&=\underline{\underline{\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\cos(\omega_{1I}+\omega_{1S})\tau+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\cos(\omega_{1I}-\omega_{1S})\tau+T}}
\end{aligned}
$$

$$
\begin{aligned}
&\left[\hat{d}^{\dagger}(t_1-\tau),\left[\hat{d}(t_1),\hat{I}_x\right]\right] \\
&=\Big[-i\Big\{\hat{R}_+\cos(\omega_{1I}+\omega_{1S})(t_1-\tau)-\hat{Q}_+\sin(\omega_{1I}+\omega_{1S})(t_1-\tau) \\
&\quad-\hat{R}_-\cos(\omega_{1I}-\omega_{1S})(t_1-\tau)+\hat{Q}_-\sin(\omega_{1I}-\omega_{1S})(t_1-\tau)\Big\} \\
&\quad+\hat{I}_z\hat{S}_x\cos\omega_{1S}(t_1-\tau)+\hat{I}_y\hat{S}_x\sin\omega_{1S}(t_1-\tau), \\
&\quad\hat{Q}_+\cos(\omega_{1I}+\omega_{1S})t_1+\hat{R}_+\sin(\omega_{1I}+\omega_{1S})t_1 \\
&\quad-\hat{Q}_-\cos(\omega_{1I}-\omega_{1S})t_1-\hat{R}_-\sin(\omega_{1I}-\omega_{1S})t_1\Big] \\
&=\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\cos(\omega_{1I}+\omega_{1S})(t_1-\tau)\cos(\omega_{1I}+\omega_{1S})t_1 \\
&\quad+\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\sin(\omega_{1I}+\omega_{1S})(t_1-\tau)\sin(\omega_{1I}+\omega_{1S})t_1 \\
&\quad+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\cos(\omega_{1I}-\omega_{1S})(t_1-\tau)\cos(\omega_{1I}-\omega_{1S})t_1 \\
&\quad+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\sin(\omega_{1I}-\omega_{1S})(t_1-\tau)\sin(\omega_{1I}-\omega_{1S})t_1+T \qquad \text{(S41)} \\
&=\underline{\underline{\frac{1}{8}\left(\hat{I}_x+\hat{S}_x\right)\cos(\omega_{1I}+\omega_{1S})\tau+\frac{1}{8}\left(\hat{I}_x-\hat{S}_x\right)\cos(\omega_{1I}-\omega_{1S})\tau+T}}
\end{aligned}
$$

Insertion into Eq. (S30) and t_1-averaging:

$$\left[\hat{A}^{\dagger}_{\mathrm{DR1}}(t_1-\tau),\left[\hat{A}_{\mathrm{DR1}}(t_1),\hat{I}_x\right]\right]+\left[\hat{A}_{\mathrm{DR1}}(t_1-\tau),\left[\hat{A}^{\dagger}_{\mathrm{DR1}}(t_1),\hat{I}_x\right]\right]$$
$$=2\left[\hat{c}^{\dagger}(t_1-\tau),\left[\hat{c}(t_1),\hat{I}_x\right]\right]\cos\omega_{0I}t_1+2\left[\hat{d}^{\dagger}(t_1-\tau),\left[\hat{d}(t_1),\hat{I}_x\right]\right]\cos\omega_{0S}t_1+T \quad \text{(S42)}$$
$$=\boxed{\begin{aligned}&\frac{1}{4}\left\{\left(\hat{I}_x+\hat{S}_x\right)\cos\left(\omega_{1I}+\omega_{1S}\right)\tau+\left(\hat{I}_x-\hat{S}_x\right)\cos\left(\omega_{1I}-\omega_{1S}\right)\tau\right\}\times\\&\qquad\times\left(\cos\omega_{0I}\tau+\cos\omega_{0S}\tau\right)+T\end{aligned}}$$

5.8.5 Outer commutator for $q = 2$

$$\left[\hat{A}^{\dagger}_{\mathrm{DR2}}(t_1-\tau),\left[\hat{A}_{\mathrm{DR2}}(t_1),\hat{I}_x\right]\right]=$$
$$=e^{i(\omega_{0I}+\omega_{0S})\tau}\cdot\Big[2\hat{I}_x\hat{S}_x+\hat{Q}_+\cos\left(\omega_{1I}+\omega_{1S}\right)(t_1-\tau)+2\hat{R}_+\sin\left(\omega_{1I}+\omega_{1S}\right)(t_1-\tau)$$
$$-2\hat{Q}_-\cos\left(\omega_{1I}-\omega_{1S}\right)(t_1-\tau)-2\hat{R}_-\sin\left(\omega_{1I}-\omega_{1S}\right)(t_1-\tau)$$
$$-i\Big(\hat{I}_y\hat{S}_x\cos\omega_{1I}(t_1-\tau)-\hat{I}_z\hat{S}_x\sin\omega_{1I}(t_1-\tau)$$
$$+\hat{I}_x\hat{S}_y\cos\omega_{1S}(t_1-\tau)-\hat{I}_x\hat{S}_z\sin\omega_{1S}(t_1-\tau)\Big),$$
$$2i\Big(\hat{R}_+\cos\left(\omega_{1I}+\omega_{1S}\right)t_1-\hat{Q}_+\sin\left(\omega_{1I}+\omega_{1S}\right)t_1$$
$$-\hat{R}_-\cos\left(\omega_{1I}-\omega_{1S}\right)t_1+\hat{Q}_-\sin\left(\omega_{1I}-\omega_{1S}\right)t_1\Big)$$
$$+\hat{I}_z\hat{S}_x\cos\omega_{1I}t_1+\hat{I}_y\hat{S}_x\sin\omega_{1I}t_1\Big]$$
$$=e^{i(\omega_{0I}+\omega_{0S})\tau}\cdot\Big\{\frac{1}{2}\left(\hat{I}_x+\hat{S}_x\right)\cos\left(\omega_{1I}+\omega_{1S}\right)(t_1-\tau)\cos\left(\omega_{1I}+\omega_{1S}\right)t_1$$
$$+\frac{1}{2}\left(\hat{I}_x+\hat{S}_x\right)\sin\left(\omega_{1I}+\omega_{1S}\right)(t_1-\tau)\sin\left(\omega_{1I}+\omega_{1S}\right)t_1$$
$$+\frac{1}{2}\left(\hat{I}_x-\hat{S}_x\right)\cos\left(\omega_{1I}-\omega_{1S}\right)(t_1-\tau)\cos\left(\omega_{1I}-\omega_{1S}\right)t_1$$
$$+\frac{1}{2}\left(\hat{I}_x-\hat{S}_x\right)\sin\left(\omega_{1I}-\omega_{1S}\right)(t_1-\tau)\sin\left(\omega_{1I}-\omega_{1S}\right)t_1$$
$$+\frac{1}{4}\hat{I}_x\cos\omega_{1I}(t_1-\tau)\cos\omega_{1I}t_1+\frac{1}{4}\hat{I}_x\sin\omega_{1I}(t_1-\tau)\sin\omega_{1I}t_1$$
$$=e^{i(\omega_{0I}+\omega_{0S})\tau}\cdot\Big\{\frac{1}{2}\left(\hat{I}_x+\hat{S}_x\right)\cos\left(\omega_{1I}+\omega_{1S}\right)\tau+\frac{1}{2}\left(\hat{I}_x-\hat{S}_x\right)\cos\left(\omega_{1I}-\omega_{1S}\right)\tau$$
$$+\frac{1}{4}\hat{I}_x\cos\omega_{1I}\tau\Big\} \quad \text{(S43)}$$

$$\left[\hat{A}^{\dagger}_{\mathrm{DR}2}(t_1-\tau),\left[\hat{A}_{\mathrm{DR}2}(t_1),\hat{I}_x\right]\right]+\left[\hat{A}_{\mathrm{DR}2}(t_1-\tau),\left[\hat{A}^{\dagger}_{\mathrm{DR}2}(t_1),\hat{I}_x\right]\right]$$

$$= \boxed{\left\{\left(\hat{I}_x+\hat{S}_x\right)\cos\left(\omega_{1I}+\omega_{1S}\right)\tau+\left(\hat{I}_x-\hat{S}_x\right)\cos\left(\omega_{1I}-\omega_{1S}\right)\tau+\frac{1}{2}\hat{I}_x\cos\omega_{1I}\tau\right\}\times \times\cos\left(\omega_{0I}+\omega_{0S}\right)\tau} \quad \text{(S44)}$$

5.9 INTEGRATION TOWARD SPECTRAL DENSITY

5.9.1 General expression

In Eq. (S26) we have to integrate expressions of the type

$$\int_0^{\infty} K(\tau)\cos\Omega\tau\,\mathrm{d}\tau = J(\Omega)$$

$$\int_0^{\infty} K(\tau)\cos\Omega\tau\,\cos\tilde{\Omega}\tau\,\mathrm{d}\tau = \frac{1}{2}\int_0^{\infty} K(\tau)\left[\cos\left(\Omega+\tilde{\Omega}\right)\tau+\cos\left(\Omega-\tilde{\Omega}\right)\tau\right]\mathrm{d}\tau \quad \text{(S45)}$$

$$= \frac{1}{2}\left[J\left(\Omega+\tilde{\Omega}\right)+J\left(\Omega-\tilde{\Omega}\right)\right]$$

which gives the function $J(\Omega)$ known as spectral density.

5.9.2 Influence of MAS on a general correlation function

The correlation loss of the orientations of internuclear vectors caused by thermal motion is superimposed by a periodic modulation caused by sample rotation [5]:

$$K_{\mathrm{rot}}(\tau) = K_{\mathrm{stat}}(\tau)\left(\frac{2}{3}\cos\omega_{\mathrm{rot}}\tau+\frac{1}{3}\cos 2\omega_{\mathrm{rot}}\tau\right) \quad \text{(S46)}$$

where K_{stat} represents the correlation function of the same vectors if the sample were non-rotating.

5.9.3 Influence of MAS on a general spectral density

$$J_{\mathrm{rot}} = \int_0^{\infty} K_{\mathrm{rot}}(\tau)\cos\Omega\tau\,\mathrm{d}\tau = \int_0^{\infty} K_{\mathrm{stat}}(\tau)\left(\frac{2}{3}\cos\omega_{\mathrm{rot}}\tau+\frac{1}{3}\cos 2\omega_{\mathrm{rot}}\tau\right)\cos\Omega\tau\,\mathrm{d}\tau$$

$$= \int_0^{\infty} K_{\mathrm{stat}}(\tau)\left(\frac{1}{3}\cos\left(\Omega+\omega_{\mathrm{rot}}\right)\tau+\frac{1}{3}\cos\left(\Omega-\omega_{\mathrm{rot}}\right)\tau\right. \quad \text{(S47)}$$

$$\left.+\frac{1}{6}\cos\left(\Omega+2\omega_{\mathrm{rot}}\right)\tau+\frac{1}{6}\cos\left(\Omega-2\omega_{\mathrm{rot}}\right)\tau\right)\mathrm{d}\tau$$

hence

$$J_{\mathrm{rot}}(\Omega) = \frac{1}{3}J_{\mathrm{stat}}\left(\Omega+\omega_{\mathrm{rot}}\right)+\frac{1}{3}J_{\mathrm{stat}}\left(\Omega-\omega_{\mathrm{rot}}\right)+\frac{1}{6}J_{\mathrm{stat}}\left(\Omega+2\omega_{\mathrm{rot}}\right)+\frac{1}{6}J_{\mathrm{stat}}\left(\Omega-2\omega_{\mathrm{rot}}\right) \quad \text{(S48)}$$

[5] S. Clough, K.W. Gray, Proc. Royal Soc. 79 (1962) 457

5.10 RESULT

5.10.1 Differential equation of observables

Inserting Eqs.(S39), (S42) and (S44) into Eq. (S26) yields

$$\begin{aligned}\frac{\mathrm{dM}_I}{\mathrm{d}t} = -C_{IS}^2 \Bigg\{ & \frac{1}{8}\Big\{\big(M_I + \tilde{M}_S\big) J_0\big(\omega_{1I} + \omega_{1S}\big) + \big(M_I - \tilde{M}_S\big) J_0\big(\omega_{1I} - \omega_{1S}\big)\Big\} \\ & + \frac{1}{128}\big(M_I + \tilde{M}_S\big)\Big[J_0\big(\omega_{0I} - \omega_{0S} + \omega_{1I} + \omega_{1S}\big) + J_0\big(\omega_{0I} - \omega_{0S} - \omega_{1I} - \omega_{1S}\big)\Big] \\ & + \frac{1}{128}\big(M_I - \tilde{M}_S\big)\Big[J_0\big(\omega_{0I} - \omega_{0S} + \omega_{1I} - \omega_{1S}\big) + J_0\big(\omega_{0I} - \omega_{0S} - \omega_{1I} + \omega_{1S}\big)\Big] \\ & - \frac{1}{64} M_I \Big[J_0\big(\omega_{0I} - \omega_{0S} + \omega_{1I}\big) + J_0\big(\omega_{0I} - \omega_{0S} - \omega_{1I}\big)\Big] \\ & + \frac{1}{8}\big(M_I + \tilde{M}_S\big)\Big[J_1\big(\omega_{0I} + \omega_{1I} + \omega_{1S}\big) + J_1\big(\omega_{0I} - \omega_{1I} - \omega_{1S}\big) \\ & \qquad + J_1\big(\omega_{0S} + \omega_{1I} + \omega_{1S}\big) + J_1\big(\omega_{0S} - \omega_{1I} - \omega_{1S}\big)\Big] \\ & + \frac{1}{8}\big(M_I - \tilde{M}_S\big)\Big[J_1\big(\omega_{0I} + \omega_{1I} - \omega_{1S}\big) + J_1\big(\omega_{0I} - \omega_{1I} + \omega_{1S}\big) \\ & \qquad + J_1\big(\omega_{0S} + \omega_{1I} - \omega_{1S}\big) + J_1\big(\omega_{0S} - \omega_{1I} + \omega_{1S}\big)\Big] \\ & + \frac{1}{2}\big(M_I + \tilde{M}_S\big)\Big[J_2\big(\omega_{0I} + \omega_{0S} + \omega_{1I} + \omega_{1S}\big) + J_2\big(\omega_{0I} + \omega_{0S} - \omega_{1I} - \omega_{1S}\big)\Big] \\ & + \frac{1}{2}\big(M_I - \tilde{M}_S\big)\Big[J_2\big(\omega_{0I} + \omega_{0S} + \omega_{1I} - \omega_{1S}\big) + J_2\big(\omega_{0I} + \omega_{0S} - \omega_{1I} + \omega_{1S}\big)\Big] \\ & + \frac{1}{4} M_I \Big[J_2\big(\omega_{0I} + \omega_{0S} + \omega_{1I} - \omega_{1S}\big) + J_2\big(\omega_{0I} + \omega_{0S} - \omega_{1I} + \omega_{1S}\big)\Big]\Bigg\}\end{aligned} \tag{S49}$$

with $\tilde{M}_S = M_S \cdot (\gamma_I / \gamma_S)$.

An identical equation is obtained for M_S by interchanging indexes "I" and "S".

5.10.2 Extraction of the relaxation times

Eq. (S49) and its analogue for M_S can be written in the form

$$\left|\begin{array}{ll} \dfrac{\mathrm{d}}{\mathrm{d}t} M_I = & -\dfrac{1}{T_{1\rho II}} M_I \quad -\dfrac{1}{T_{1\rho IS}} M_S \\ \dfrac{\mathrm{d}}{\mathrm{d}t} M_S = & -\dfrac{1}{T_{1\rho SI}} M_I \quad -\dfrac{1}{T_{1\rho SS}} M_S \end{array}\right| \tag{S50}$$

with

$$\frac{1}{T_{1\rho II}} = C_{IS}^2 \left\{ \frac{1}{8}\left\{ J_0\left(\omega_{1I}+\omega_{1S}\right) + J_0\left(\omega_{1I}-\omega_{1S}\right)\right\} \right.$$
$$+\frac{1}{128}\Big[J_0\left(\omega_{0I}-\omega_{0S}+\omega_{1I}+\omega_{1S}\right) + J_0\left(\omega_{0I}-\omega_{0S}-\omega_{1I}-\omega_{1S}\right)$$
$$+ J_0\left(\omega_{0I}-\omega_{0S}+\omega_{1I}-\omega_{1S}\right) + J_0\left(\omega_{0I}-\omega_{0S}-\omega_{1I}+\omega_{1S}\right)\Big]$$
$$-\frac{1}{64}\Big[J_0\left(\omega_{0I}-\omega_{0S}+\omega_{1I}\right) + J_0\left(\omega_{0I}-\omega_{0S}-\omega_{1I}\right)\Big]$$
$$+\frac{1}{8}\Big[J_1\left(\omega_{0I}+\omega_{1I}+\omega_{1S}\right) + J_1\left(\omega_{0I}-\omega_{1I}-\omega_{1S}\right)$$
$$+ J_1\left(\omega_{0S}+\omega_{1I}+\omega_{1S}\right) + J_1\left(\omega_{0S}-\omega_{1I}-\omega_{1S}\right)$$
$$+ J_1\left(\omega_{0I}+\omega_{1I}-\omega_{1S}\right) + J_1\left(\omega_{0I}-\omega_{1I}+\omega_{1S}\right)$$
$$+ J_1\left(\omega_{0S}+\omega_{1I}-\omega_{1S}\right) + J_1\left(\omega_{0S}-\omega_{1I}+\omega_{1S}\right)\Big]$$
$$+\frac{1}{2}\Big[J_2\left(\omega_{0I}+\omega_{0S}+\omega_{1I}+\omega_{1S}\right) + J_2\left(\omega_{0I}+\omega_{0S}-\omega_{1I}-\omega_{1S}\right)\Big]$$
$$\left. +\frac{3}{4}\Big[J_2\left(\omega_{0I}+\omega_{0S}+\omega_{1I}-\omega_{1S}\right) + J_2\left(\omega_{0I}+\omega_{0S}-\omega_{1I}+\omega_{1S}\right)\Big]\right\} \qquad \text{(S51)}$$

and

$$\frac{1}{T_{1\rho IS}} = C_{IS}^2 \left\{ \frac{1}{8}\left\{ J_0\left(\omega_{1I}+\omega_{1S}\right) - J_0\left(\omega_{1I}-\omega_{1S}\right)\right\} \right.$$
$$+\frac{1}{128}\Big[J_0\left(\omega_{0I}-\omega_{0S}+\omega_{1I}+\omega_{1S}\right) + J_0\left(\omega_{0I}-\omega_{0S}-\omega_{1I}-\omega_{1S}\right)$$
$$- J_0\left(\omega_{0I}-\omega_{0S}+\omega_{1I}-\omega_{1S}\right) - J_0\left(\omega_{0I}-\omega_{0S}-\omega_{1I}+\omega_{1S}\right)\Big]$$
$$+\frac{1}{8}\Big[J_1\left(\omega_{0I}+\omega_{1I}+\omega_{1S}\right) + J_1\left(\omega_{0I}-\omega_{1I}-\omega_{1S}\right)$$
$$+ J_1\left(\omega_{0S}+\omega_{1I}+\omega_{1S}\right) + J_1\left(\omega_{0S}-\omega_{1I}-\omega_{1S}\right)\Big] \qquad \text{(S52)}$$
$$- J_1\left(\omega_{0I}+\omega_{1I}-\omega_{1S}\right) - J_1\left(\omega_{0I}-\omega_{1I}+\omega_{1S}\right)$$
$$- J_1\left(\omega_{0S}+\omega_{1I}-\omega_{1S}\right) - J_1\left(\omega_{0S}-\omega_{1I}+\omega_{1S}\right)\Big]$$
$$+\frac{1}{2}\Big[J_2\left(\omega_{0I}+\omega_{0S}+\omega_{1I}+\omega_{1S}\right) + J_2\left(\omega_{0I}+\omega_{0S}-\omega_{1I}-\omega_{1S}\right)$$
$$- J_2\left(\omega_{0I}+\omega_{0S}+\omega_{1I}-\omega_{1S}\right) - J_2\left(\omega_{0I}+\omega_{0S}-\omega_{1I}+\omega_{1S}\right)\Big]$$

Equations for the two remaining relaxation times can be obtained by exchanging indexes "*I*" and "*S*".

5.10.3 Approximation: Large Larmor frequency

Usually, $\omega_{0I}, \omega_{0S} >> \omega_{1I}, \omega_{1S}, \omega_{\text{rot}}$:

Spectral densities with static Larmor frequencies in the argument are usually small compared to the first terms, therefore they can be omitted. Only terms without both ω_{1I} and ω_{1S} remain:

$$\boxed{\begin{aligned}\frac{1}{T_{1\rho II}} &= \frac{1}{8}C_{IS}^2\left[J_0\left(\omega_{1I}+\omega_{1S}\right)+J_0\left(\omega_{1I}-\omega_{1S}\right)\right]\\ \frac{1}{T_{1\rho IS}} &= \frac{1}{8}C_{IS}^2\left[J_0\left(\omega_{1I}+\omega_{1S}\right)-J_0\left(\omega_{1I}-\omega_{1S}\right)\right]\end{aligned}} \qquad \text{(S53)}$$

5.10.4 MAS

Replacing J_0 in Eq. (S53) by Eq. (S48) and using $K_0(0) = 4/5$ (Eq. (S27)) we finally get

$$\boxed{\begin{aligned}\frac{1}{T_{1\rho II}} = \frac{1}{60}C_{IS}^2\Big[&2J_0\left(\omega_{1I}+\omega_{1S}+\omega_{\text{rot}}\right)+2J_0\left(\omega_{1I}+\omega_{1S}-\omega_{\text{rot}}\right)\\ &+J_0\left(\omega_{1I}+\omega_{1S}+2\omega_{\text{rot}}\right)+J_0\left(\omega_{1I}+\omega_{1S}-2\omega_{\text{rot}}\right)\\ &+2J_0\left(\omega_{1I}-\omega_{1S}+\omega_{\text{rot}}\right)+2J_0\left(\omega_{1I}-\omega_{1S}-\omega_{\text{rot}}\right)\\ &+J_0\left(\omega_{1I}-\omega_{1S}+2\omega_{\text{rot}}\right)+J_0\left(\omega_{1I}-\omega_{1S}-2\omega_{\text{rot}}\right)\Big]\\ \frac{1}{T_{1\rho IS}} = \frac{1}{60}C_{IS}^2\Big[&2J_0\left(\omega_{1I}+\omega_{1S}+\omega_{\text{rot}}\right)+2J_0\left(\omega_{1I}+\omega_{1S}-\omega_{\text{rot}}\right)\\ &+J_0\left(\omega_{1I}+\omega_{1S}+2\omega_{\text{rot}}\right)+J_0\left(\omega_{1I}+\omega_{1S}-2\omega_{\text{rot}}\right)\\ &-2J_0\left(\omega_{1I}-\omega_{1S}+\omega_{\text{rot}}\right)-2J_0\left(\omega_{1I}-\omega_{1S}-\omega_{\text{rot}}\right)\\ &-J_0\left(\omega_{1I}-\omega_{1S}+2\omega_{\text{rot}}\right)-J_0\left(\omega_{1I}-\omega_{1S}-2\omega_{\text{rot}}\right)\Big]\end{aligned}} \qquad \text{(S54)}$$

6 SPINACH CODE FOR $R_{1\rho}$ SIMULATIONS

File "R1rho_Hdec_input.m"

```
% sample input for R1rho_1Hdec.m pulse sequence
% two site chemical exchange jump model
% jurackhannes@gmail.com

% written in MATLAB R2021a Update 2
% and Spinach version 2.6.5608 (June 2021)

% needed User inputs marked with !!
% explanations can be found in the used pulse sequence "R1rho_1Hdec.m"

% following simulation parameters are automatically varried:
% SL freq., 1H dec freq., jumprate, 1H dec pulseduration
% for each you define a start, end and number of points
% an aequidistant list is created for each (jumprate in logarithmic scale)

% after setting all inputs, make sure that both scripts are in cur work dir
% then simply run "R1rho_1Hdec_input" without paranthesis in matlab console

% outputs in folder output/R1r1Hdec_timestamp:
% (i)
% one decay file per jumprate as e.g. decays_1_00e02 .txt and .mat
% in each the three other paramters are varried
% 1st collumn: timedomain.
% decays in following order: for each pulseduration-step, dec power step
% and irr power step
% permutation example:
% for each pulseduration(123), dec power(45) and irr power(78)
% file column structure: time 147 247 347 157 257 357 148 248 348 ...
% (ii)
% parameters are saved as log file in parameters.txt
% (iii)
% if you specified that the decays shall be fitted and plotted:
% exp_fit results: first column prefactor; second column Relaxation rate
% plots of decays and fits as .pdf and .fig files

% possible improvements:
% apply nonselective damping to all states -> thermal equilibrium
% CSA tensors
% three-site jumps
% multi-spin system
% grumble function improvements
% add a GUI for inputs
% skip redundant simulations (e.g. just 1Hdec changed without decoupling)
% fit errors


function R1rho_1Hdec_input()
%output directory and current time
time = datestr(now,'ddmmyyyy_HHMM');
dirname = sprintf('output/R1r1Hdec_%s', time);
if not(isfolder(dirname))
    mkdir(dirname);
end


% START OF USER INPUTS
```

```
fits_plots = 0; % shall the decays be fitted and plotted? 1=yes    !!
fitting_skipped_rotors = 0; % inital rotor periods to skip for fits    !!

% generate symmetric two site jump spin pair system    !!
jumprate_list_user = []; % leave as "[]" if you want automatic list (below)
jumprate_min = 4.0; % minimum exchange process jump rate EXPONENT (2->100Hz)
jumprate_max = 4.0; % max exchange process jump rate EXPONENT
jumprate_steps = 1; % number of jumprates to simulate (logarithmic space)
% NOTE: automatic aequidistant list creation: e.g. jumprate_steps = 2 only
%       uses min and max rates. jumprate_steps = 3 uses min, max, and mid

if size(jumprate_list_user,2)>0
   jumprate_list = jumprate_list_user;
   jumprate_steps = size(jumprate_list_user,2);
else
   jumprate_list = logspace(jumprate_min,jumprate_max,jumprate_steps);
end
%disp(jumprate_list);
% return;

jumpangle = 10; % two site jump angle in deg  !!
inter_dist = 1.02; % internuclear distance in Angstrom    !!

% build spin positions
pos_1 = inter_dist*sin(jumpangle*pi/180);
pos_2 = inter_dist*cos(jumpangle*pi/180);

sys.magnet=14.1; % static magn field (T), 14.1 T corresponds to 600 MHz  !!
sys.isotopes={'15N','1H','15N','1H'}; % if you want to observe 15N    !!
% sys.isotopes={'1H','13C','1H','13C'}; % if you want to observe 13C    !!

% zeeman interaction !!
%inter.zeeman.eigs={[0 0 160],[0 0 0],[0 0 160],[0 0 0]};
%inter.zeeman.eigs={[0 0 0],[0 0 0],[0 0 0],[0 0 0]};
%inter.zeeman.euler={[0 0 0],[0 0 0],[0 -jumpangle*pi/180 0],[0 0 0]};
% euler angles [+-a b 0]

% Turn off dipolar coupling completely? -> Set cutoff below nuclei distance
% sys.tols.prox_cutoff = 0.1; % distance, in Angstroms
% beyond which dipolar interactions are ignored (default is 100)


% dipolar couplings are calculated from the coordinates
% spin positions  [x,y,z] !!
inter.coordinates={[0.0 0.0 0.0]
                   [0.0 0.0 inter_dist]
                   [0.0 0.0 0.0]
                   [0.0 -pos_1 pos_2]};

inter.chem.parts={[1 2],[3 4]}; % define parts of species
inter.chem.concs=[1.0 1.0]; % start populations    !!

% Basis sets
bas.formalism='sphten-liouv';
bas.approximation='none';

% Algorithmic options
sys.disable={'trajlevel'};
sys.enable={'caching'}; % store propagators for later (repetitive pseq)
% 'gpu' if you have a CUDA ready GPU - first test with RTX 3090:
% considerable slowdown. Maybe parallel slowdown or floating point
```

```
% precision of the GPU is capped (better use TITAN RTX)

% loop over jumprate list
for jumprate_it=1:jumprate_steps
    % exchange rate matrix    !!
   inter.chem.rates=jumprate_list(jumprate_it)*[-1,1;1,-1];

   % Spinach housekeeping
   spin_system=create(sys,inter);
   spin_system=basis(spin_system,bas);

   % PARAMETERS for pulse sequence R1rho.m

   % MAS    !!
   parameters.rate=18000.0;
   parameters.axis=[1 1 1];
   parameters.max_rank=5; % 4 should be max needed for R1r     !!
   % with rank 2 the relaxation RATE didnt change (further testing needed)
   % relevant spins

   parameters.spins={'1H','15N'}; % if you want to obs 15N   !!
   % parameters.spins={'1H','13C'}; % if you want to obs 13C   !!

   % rotating frame transformations
   parameters.rframes={{'1H',2},{'15N',2}};  % if you want to obs 15N !!
   % parameters.rframes={{'1H',2},{'13C',2}};  % if you want to obs 13C !!

   % powder average - 2 angles, 100 points, full sphere, repulsion   !!
   parameters.grid='rep_2ang_200pts_sph';
   parameters.sum_up=1;

   % initial states
   parameters.rho0=state(spin_system,'Lz','15N','exact');
   % parameters.rho0=state(spin_system,'Lz','13C','exact');

   % observed states
   parameters.coil=state(spin_system,'L+','15N','exact');
   % parameters.coil=state(spin_system,'L+','13C','exact');
   %
   % Relevant operators
   Hp=operator(spin_system,'L+','1H');
   Np=operator(spin_system,'L+','15N');
   % Cp=operator(spin_system,'L+','13C');
   Hx=(Hp+Hp')/2; Hy=(Hp-Hp')/2i;
   Nx=(Np+Np')/2; Ny=(Np-Np')/2i;
   % Cx=(Cp+Cp')/2; Cy=(Cp-Cp')/2i;

   % excitation pulse operator (infinitely hard applied)   !!
   parameters.exc_oper=Ny;
   % parameters.exc_oper=Cy;

   % cw spin lock pulse operator and power (soft)    !!
   parameters.irr_oper=Nx;
   % parameters.irr_oper=Cx;
   parameters.irr_power_list_user = []; % leave as "[]" if you want automatic list (below)
   parameters.irr_power_min=4000;
   parameters.irr_power_max=16000;
   parameters.irr_power_npoints=7; % if ...npoints=1, only max is used

   % simultaneous 1H rotorsynched decoupling operator, power (soft)  !!
   parameters.dec_oper=Hx;
```

```
parameters.dec_power_list_user = []; % leave as "[]" if you want automatic list (below)
parameters.dec_power_min=100000;
parameters.dec_power_max=100000;
parameters.dec_power_npoints=1; % if ...npoints=1, only max is used

% number of 1H dec pulses per rotor
parameters.pulses_per_rotor=1;

% simulation precision, datapoints
% dont do more than 100K simulation steps per rotor cycle (RAM)
% Tested good timestep for 18kHz MAS over 10 rotor periods:
% convergence for 100 pulse- & 100 tsteps per pulsestep
% => should be smaller or equal: tstep<=5e-9
% 5e-10 didnt improve evolution!
% faster dynamics -> smaller timesteps needed. (us-ms motions are fine)

% ROTORSYNCHRONIZATION
% NUMBER OF ROTORPERIODS you want to simulate    !!
parameters.rotorperiods=400;

% number of simulation pulse-steps per rotor period    !!
parameters.pulsesteps_per_rotor=128;

% number of timesteps per pulsestep    !!
parameters.tsteps_per_pulsestep=1; % can be set to 1 if you wish

% simulation timestep (calculated to be rotorsynched)
parameters.timestep=1/parameters.rate/parameters.pulsesteps_per_rotor/...
   parameters.tsteps_per_pulsestep; % also being output to console
fprintf('using simulation timestep: %.12f seconds\n',parameters.timestep);

% resulting number of timesteps per rotor
parameters.stepamount=parameters.pulsesteps_per_rotor*...
   parameters.tsteps_per_pulsestep;

% resulting time domain
parameters.timesum=parameters.timestep*parameters.rotorperiods*...
   parameters.pulsesteps_per_rotor*parameters.tsteps_per_pulsestep;

% 1H dec pulsedurations of interest:    !!
% in number of simulation pulsesteps
parameters.pulsedur_list_user = []; % leave as "[]" if you want automatic list (below)
parameters.pulsedur_min=0; % 0: without decoupling
parameters.pulsedur_max=(parameters.pulsesteps_per_rotor/...
   parameters.pulses_per_rotor); % this correponds to CW
parameters.n_pulsedurations=1; % take care, that every entry is integer
% n_pulsedurations=1 means only max is used (e.g. CW); =2 only min and max

% saved data points per MAS rotor period:   !!
parameters.aq_per_rotor=8;

% misc
parameters.verbose=0; % avoid excessive console output

% END OF USER INPUTS


% translate SL power list
if size(parameters.irr_power_list_user,2)>0
   parameters.irr_power_list = parameters.irr_power_list_user;
   parameters.irr_power_npoints = size(parameters.irr_power_list_user,2);
```

```
    else
        parameters.irr_power_list = linspace(parameters.irr_power_min,...
            parameters.irr_power_max,...
            parameters.irr_power_npoints);
    end

    % translate 1H decoupling power list
    if size(parameters.dec_power_list_user,2)>0
        parameters.dec_power_list = parameters.dec_power_list_user;
        parameters.dec_power_npoints = size(parameters.dec_power_list_user,2);
    else
        parameters.dec_power_list = linspace(parameters.dec_power_min,...
        parameters.dec_power_max,...
        parameters.dec_power_npoints);
    end

    % translate 1H decoupling pulse duration list, delays en passant
    if size(parameters.pulsedur_list_user,2)>0
        parameters.pulsedur_list = parameters.pulsedur_list_user;
        parameters.n_pulsedurations = size(parameters.pulsedur_list_user,2);
    else
        parameters.pulsedur_list = linspace(parameters.pulsedur_min,...
            parameters.pulsedur_max,...
            parameters.n_pulsedurations);
    end


    % PULSE SEQUENCE CALL
    % use Fokker-Planck single angle spinning

    % Liouvillian generation and passing to pulse sequence
    decay_mat_fin=singlerot(spin_system,@R1rho_1Hdec,parameters,'labframe');

    % output with time
    time_list=linspace(0,parameters.timesum,parameters.rotorperiods*parameters.aq_per_rotor);


    % SAVE DATA TO DISK
    % reform OUTPUT
    output = [time_list.' re-
shape(real(decay_mat_fin),[],parameters.n_pulsedurations*parameters.dec_power_npoints*parameters.irr_powe
r_npoints)];

    % format current jumprate as string for filenames
    % if you have very small jumprate steps: change '%0.2e' to e.g. '%0.5e'
    str_jumprate = sprintf('%0.2e',jumprate_list(jumprate_it));
    str_jumprate = strrep(str_jumprate,'.','_'); % _ is new comma
    str_jumprate = strrep(str_jumprate,'+',''); % if exponent positv: ommit

    % SAVE decays as matlab and txt outputs
    filename1 = sprintf('output/R1r1Hdec_%s/decays_%s.mat', time, str_jumprate);
    filename2 = sprintf('output/R1r1Hdec_%s/decays_%s.txt', time, str_jumprate);
    save(filename1,'output');
    save(filename2,'output','-ascii');

    fitting_skipped_points = fitting_skipped_rotors*parameters.aq_per_rotor+1;

    if fits_plots==1
        % single exponentional fits
        exp_func = @(paras,xdata)paras(1)*exp(-paras(2)*xdata); %fit function
        paras0 = [0.5,10]; % initial fitting parameters [amplitude,Relax. rate]
```

```
        paras_result = zeros(size(output,2)-1,2); % preallocation of mat size

        % fitting loop: least-squares method (Optimization Toolbox) + plots
        fig1 = figure('visible','off');
        hold on; % all decay plots in same figure (one file per jumprate)
        for output_it=1:(size(output,2)-1)
            paras_result(output_it,:) =
lsqcurvefit(exp_func,paras0,output(fitting_skipped_points:end,1),output(fitting_skipped_points:end,output_it+1)
);
            plot(output(:,1),output(:,output_it+1));

plot(output(fitting_skipped_points:end,1),exp_func(paras_result(output_it,:),output(fitting_skipped_points:end,1
)));
        end
        hold off;

        filename5 = sprintf('output/R1r1Hdec_%s/decays_plt_%s', time, str_jumprate);
        filename6 = sprintf('output/R1r1Hdec_%s/decays_fig_%s.pdf', time, str_jumprate);
        set(fig1, 'visible', 'on');
        saveas(fig1, filename5, 'fig');
        exportgraphics(fig1,filename6,'ContentType','vector')
        close(fig1)

        % save fitting parameters
        filename4 = sprintf('output/R1r1Hdec_%s/exp_fits_%s.txt', time, str_jumprate);
        save(filename4,'paras_result','-ascii');
    end
end

%write parameters to disk:
filename3 = sprintf('output/R1r1Hdec_%s/parameters.txt', time);
fileID = fopen(filename3, 'wt'); % Open and create file in text writing mode.
fprintf(fileID, ' ==========================================\r\n');
fprintf(fileID, '||      PARAMETERS AND LOG              ||\r\n');
fprintf(fileID, '||note: if you created explicit user lists,||\r\n');
fprintf(fileID, '||they will not be displayed here        ||\r\n');
fprintf(fileID, ' ==========================================\r\n\r\n');
fprintf(fileID, 'minimum used jumprate[Hz] =\t%f\r\n', jumprate_min);
fprintf(fileID, 'max jumprate[Hz] =\t%f\r\n', jumprate_max);
fprintf(fileID, 'number of jumprate steps =\t%f\r\n', jumprate_steps);
fprintf(fileID, 'jumpangle[deg] =\t%f\r\n', jumpangle);
fprintf(fileID, 'inter_dist[angstrom] =\t%f\r\n', inter_dist);
if isfield(inter,'coordinates') == 1
    fprintf(fileID, 'distance matrix [Angstrom] =\t%4f %4f %4f\r\n', cell2mat(inter.coordinates).');
end
if isfield(inter.zeeman,'eigs') == 1
    fprintf(fileID, 'CSA eigenvalues [ppm] =\t%4f %4f %4f\r\n', cell2mat(inter.zeeman.eigs).');
end
if isfield(inter.zeeman,'euler') == 1
    fprintf(fileID, 'CSA euler angles [radians] =\t%4f %4f %4f\r\n', cell2mat(inter.zeeman.euler).');
end
fprintf(fileID, 'max harmonic rank =\t%f\r\n', parameters.max_rank);
fprintf(fileID, 'MAS rate[Hz] =\t%f\r\n', parameters.rate);
fprintf(fileID, 'min SL[Hz] =\t%f\r\n', parameters.irr_power_min);
fprintf(fileID, 'max SL[Hz] =\t%f\r\n', parameters.irr_power_max);
fprintf(fileID, 'SL steps =\t%f\r\n', parameters.irr_power_npoints);
fprintf(fileID, 'min 1Hdec[Hz] =\t%f\r\n', parameters.dec_power_min);
fprintf(fileID, 'max 1Hdec[Hz] =\t%f\r\n', parameters.dec_power_max);
fprintf(fileID, '1Hdec steps =\t%f\r\n', parameters.dec_power_npoints);
fprintf(fileID, '1Hdec pulses per rotor =\t%f\r\n', parameters.pulses_per_rotor);
```

```
fprintf(fileID, 'minimum 1Hdec pulseduration (in pulsesteps per rotorperiod) =\t%f\r\n', parameters.pulsedur_min);
fprintf(fileID, 'maximum 1Hdec pulseduration (in pulsesteps per rotorperiod) =\t%f\r\n', parameters.pulsedur_max);
fprintf(fileID, 'number of 1Hdec pulse durations to simulate !use this wisely: each entry needs to be and integer!!! =\t%f\r\n', parameters.n_pulsedurations);
fprintf(fileID, 'number of rotorperiods =\t%f\r\n', parameters.rotorperiods);
fprintf(fileID, 'pulse simulation steps PER ROTORPERIOD =\t%f\r\n', parameters.pulsesteps_per_rotor);
fprintf(fileID, 'additional timesteps per pulsestep (above) =\t%f\r\n', parameters.tsteps_per_pulsestep);
fprintf(fileID, 'resulting simulation timestep[s] =\t%.12f\r\n', parameters.timestep);
fprintf(fileID, 'number of timesteps per rotorperiod =\t%f\r\n', parameters.stepamount);
fprintf(fileID, 'timedomain[s] =\t%.9f\r\n', parameters.timesum);
fprintf(fileID, 'aquisition points per rotor period =\t%f\r\n', parameters.aq_per_rotor);
fprintf(fileID, 'number of skipped inital rotor periods for fitting =\t%f\r\n\r\n', fitting_skipped_rotors);
% fprintf(fileID, '[] =\t%f\r\n', );
fclose(fileID);
%
end
```

File "R1rho_Hdec.m"

```
% Solid-State MAS NMR 15N/13C Relaxation
% in presence of an alternating magnetic cw field 15N/13C
% with simultaneous rotorsynchronized 1H irradiation
% Calculation time: input dependent (minutes up to days)
% jurackhannes@gmail.com
%
% written in MATLAB R2021a Update 2
% and Spinach version 2.6.5608 (June 2021)
%
% Reinforcement Learning Toolbox is not needed
% this is just the NMR pulse sequence
% specify following commands in input file:
%
% pulse sequence call Syntax:
% R1rho_...(spin_system,parameters,H,R,K)
%
% should be called using singlerot.m (MAS) context to obtain  H, R, and K
%    H     - Hamiltonian matrix, received from context function
%    R     - relaxation superoperator, received from context function
%    K     - kinetics superoperator, received from context function
%
% SYSTEM and INTERACTION specifications:
% sys.magnet - primary magnet field must be specified in units of Tesla
% sys.isotopes - Specify the spin system composition by giving a list of isotope names
% sys.labels - Optionally, specify a label for each spin by giving a list of strings
%
% chemical shifts:
% inter.zeeman.eigs
% inter.zeeman.euler
% OR:
% inter.zeeman.matrix  - Full chemical shift tensors (in ppm for nuclei) as matrices.
%
% dipolar interactions:
% inter.coordinates -
% OR: inter.coupling.eigs & inter.coupling.euler
% for dipolar interactions supplied as eigenvalues and Euler angles
% (make sure the eigenvalues sum up to zero)
% inter.pbc - Periodic boundary conditions may also be needed
%
% spinach accuracy:
% sys.disable={'trajlevel'}; - the internal heuristics in Spinach is optimised for very large spin systems. If your system has fewer than five spins, there is no point running sophisticated trajectory-level state space analysis. In that case, some performance may be gained by disabling it
% sys.tols.prox_cutoff  - the distance beyond which dipolar couplings are ignored, default is 100 Angstrom
% sys.tols.inter_cutoff - the amplitude below which spin-spin interactions are ignored, default is 1e-10 rad/s
% sys.disable={'krylov'}; - To force the faster matrix exponential path
% sys.enable={'gpu','caching'}; - speed improvements on SSD or GPU
%
% sys.output='filename'; - output location
% sys.output='hush'; - output disabled
% fclose('all'); - at the end to close all files
%
% BASIS SET specifications: Spinach input must contain a specification of the formalism to be used (Hilbert or Liouville space), a specification of the type of basis operators to be used (Pauli matrices or irreducible spherical tensors), and a choice of which quantum states should be taken into consideration during the simulation process
% spin_system=create(sys,inter);
% spin_system=basis(spin_system,bas); - updates the spin_system data structure with formalism and basis set information
% bas.formalism='sphten-liouv'; - Liouville space formalism
```

```
% bas.approximation - up to 6-12 spins to 'none' to request a complete basis set
% bas.connectivity='full_tensors'; - solid state NMR, DNP and many ESR systems a coupling is essentially ani-
sotropic and the interaction network must also count traceless interactions because they manifest directly.
% bas.projections=[-1 0 1] or bas.longitudinals={'15N','13C'}; or bas.zero_quantum={'15N','13C'}; -  screen the
basis sets according to coherence order and only keep the user-specified coherences
%
% PARAMETERS:
% parameters.npoints - Number of points in each dimension of the output data array
% parameters.verbose -  inside powder.m, singlerot.m and doublerot.m, where the console output is suppressed
unless the user specifically indicates that it shouldn't be by setting
% parameters.sweep - Sweep width in each dimension of the spectrum (Hz)
% parameters.dead_time - the system will be evolved for this time (seconds) before the signal acquisition begins
(look into aquire.m)
%  -
% parameters.spins  - Spins that are to be assigned to each instrument channel during the simulation
% parameters.rframes - numerical rotating frame transformation specification
%  -
% parameters.rate - spinning rate for single rotation solid state NMR experiments
% parameters.axis - rotor axis for single rotation solid state NMR experiments
% parameters.max_rank - maximum harmonic rank to retain (15-85)
% parameters.grid - powder averaging grid
% parameters.ref_frame='rotor'; - Floquet approach
%  -
% parameters.pulse_dur - duration of each pulse, a two-element vector, seconds
% parameters.pulse_amp - amplitude of each pulse, a two-element vector, rad/s
%  -
% parameters.rho0      - initial condition, usually Lz
% parameters.coil      - detection state, usually L+
% parameters.homodec_oper - operator to add to the Liouvillian at the detection stage (look into aquire.m)
% parameters.homodec_pwr - power coefficient for the operator, Hz (look into aquire.m)
%
% additional singlerot.m options:
% parameters.sum_up
% assumptions
%
% MOTIONAL MODEL EXAMPLE (two sites)
% sys.isotopes={'1H','15N','1H','15N'};
% inter.zeeman.scalar={0.0 0.0 0.0 0.0};
% inter.coordinates={[0 0 0]; [0 0 2];
%                    [0 0 0]; [0 2 0]};
% Chemical exchange
% inter.chem.parts={[1 2],[3 4]};
% inter.chem.rates=[-5000 +5000;
%                   +5000 -5000];
% inter.chem.concs=[1 1];
%
% Outputs:
% Relaxation decay matrix - decay as seen by the state parameters.coil
%                           for different 1H dec pulselengths, 1H dec power
%                           and SL powers
%
%
function decay_mat=R1rho_1Hdec(spin_system,parameters,H,R,K)
% preallocate decay_mat size to save time
% it holds all output decays
decay_mat=zeros(parameters.rotorperiods*parameters.aq_per_rotor,parameters.n_pulsedurations,...
   parameters.dec_power_npoints,parameters.irr_power_npoints);
% Check consistency
grumble(spin_system,parameters,H,R,K);
% Project the operators, to match matrix dimensions
parameters.exc_oper=kron(speye(parameters.spc_dim),parameters.exc_oper);
```

```
parameters.irr_oper=kron(speye(parameters.spc_dim),parameters.irr_oper);
parameters.dec_oper=kron(speye(parameters.spc_dim),parameters.dec_oper);
% Compose Liouvillian
L=H+1i*R+1i*K;
% Excitation pulse
rho=step(spin_system,parameters.exc_oper,parameters.rho0,pi/2);
%
% itteration over different parameters:
for irr_power_it=1:parameters.irr_power_npoints
   % add up L and SL irradiation
   L_irr=L+2*pi*parameters.irr_power_list(irr_power_it)*parameters.irr_oper;
   for dec_power_it=1:parameters.dec_power_npoints
      % add up L_irr and 1H decoupling
      L_dec=L_irr+2*pi*parameters.dec_power_list(dec_power_it)*parameters.dec_oper;
      %
      % needed simulation precision estimation; controlled via
      % pulsesteps_per_rotor and timesteps_per_pulsestep
      % disp(1/norm(L_dec));
      % disp(1/normest(L_dec));
      %
      for pulsedur_it=1:parameters.n_pulsedurations
         % find number of timesteps for the two evolution periods
         % CW or non-dec won't work at the moment!
         n_steps_dec=parameters.pulsedur_list(pulsedur_it)*parameters.tsteps_per_pulsestep; % correct???
         % integer size cannot exeed 16K here!
         n_steps_irr=uint32(((parameters.pulsesteps_per_rotor/parameters.pulses_per_rotor)-
parameters.pulsedur_list(pulsedur_it))*parameters.tsteps_per_pulsestep);
         %RUN WITHOUT 15N/13C decoupling:
         if n_steps_dec==0
            % preallocate rho_traj matrizes size to save time
            rho_traj_mat_irr=zeros(size(rho,1),n_steps_irr);
            % overwrite last trajectory vector (new sim start point)
            rho_traj_mat_irr(:,end) = rho;
            % PULSE SEQUENCE:
            for rotorperiods_it=1:parameters.rotorperiods
               coil_vec = zeros(1,n_steps_irr);
               for pulses_it=1:parameters.pulses_per_rotor
                  % basic simulation CONCEPT
                  % overwrite density matrix trajectory, saved matrix
                  % start from last trajectroy vector rho_traj_mat_
                  %
                  %
                  % Run pure 15N/13C SPIN LOCK
                  rho_traj_mat_irr = evolu-
tion(spin_system,L_irr,[],rho_traj_mat_irr(:,end),parameters.timestep,n_steps_irr-1,'trajectory');
                  % obtain obervable as "seen" from coil state
                  % coil_vec will hold evolution of one rotor
                  coil_vec(1,pulses_it*(n_steps_dec+n_steps_irr)+1-
(n_steps_dec+n_steps_irr):pulses_it*(n_steps_dec+n_steps_irr)) = parameters.coil'*rho_traj_mat_irr;
               end
               % AQUISITION
               % only save parameters.aq_per_rotor points every rotor
               decay_mat(rotorperiods_it*parameters.aq_per_rotor-
parame-
ters.aq_per_rotor+1:rotorperiods_it*parameters.aq_per_rotor,pulsedur_it,dec_power_it,irr_power_it)=coil_vec(1
,1:parameters.stepamount/parameters.aq_per_rotor:end);
            end
         %RUN in CW 15N/13C decoupling:
         elseif n_steps_irr==0
            % preallocate rho_traj matrizes size to save time
            rho_traj_mat_dec=zeros(size(rho,1),n_steps_dec);
```

```
            % overwrite last trajectory vector (new sim start point)
            rho_traj_mat_dec(:,end) = rho;
            % PULSE SEQUENCE:
            for rotorperiods_it=1:parameters.rotorperiods
               coil_vec = zeros(1,n_steps_dec);
               for pulses_it=1:parameters.pulses_per_rotor
                  % Run 15N/13C SPIN LOCK + 1H cw decoupling
                  rho_traj_mat_dec = evolu-
tion(spin_system,L_dec,[],rho_traj_mat_dec(:,end),parameters.timestep,n_steps_dec-1,'trajectory');
                  % obtain obervable as "seen" from coil state
                  % coil_vec will hold evolution of one rotor
                  coil_vec(1,pulses_it*(n_steps_dec+n_steps_irr)+1-
(n_steps_dec+n_steps_irr):pulses_it*(n_steps_dec+n_steps_irr)) = parameters.coil'*rho_traj_mat_dec;
               end
               % AQUISITION
               % only save parameters.aq_per_rotor points every rotor
               decay_mat(rotorperiods_it*parameters.aq_per_rotor-
parame-
ters.aq_per_rotor+1:rotorperiods_it*parameters.aq_per_rotor,pulsedur_it,dec_power_it,irr_power_it)=coil_vec(1
,1:parameters.stepamount/parameters.aq_per_rotor:end);
            end
         %RUN in rotorsynch decoupling mode:
         else
            % preallocate rho_traj matrizes size to save time
            rho_traj_mat_dec=zeros(size(rho,1),n_steps_dec);
            rho_traj_mat_irr=zeros(size(rho,1),n_steps_irr);
            % overwrite last trajectory vector (new sim start point)
            % called in alternating mode
            rho_traj_mat_irr(:,end) = rho;
            % PULSE SEQUENCE:
            for rotorperiods_it=1:parameters.rotorperiods
               coil_vec = zeros(1,n_steps_dec+n_steps_irr);
               for pulses_it=1:parameters.pulses_per_rotor
                  % Run 15N/13C SPIN LOCK + 1H decoupling with delays
                  rho_traj_mat_dec = evolu-
tion(spin_system,L_dec,[],rho_traj_mat_irr(:,end),parameters.timestep,n_steps_dec-1,'trajectory');
                  % Run pure 15N/13C SPIN LOCK
                  rho_traj_mat_irr = evolu-
tion(spin_system,L_irr,[],rho_traj_mat_dec(:,end),parameters.timestep,n_steps_irr-1,'trajectory');
                  % obtain obervable as "seen" from coil state
                  % coil_vec will hold evolution of one rotor
                  coil_vec(1,pulses_it*(n_steps_dec+n_steps_irr)+1-
(n_steps_dec+n_steps_irr):pulses_it*(n_steps_dec+n_steps_irr)) = [parame-
ters.coil'*rho_traj_mat_dec,parameters.coil'*rho_traj_mat_irr];
               end
               % AQUISITION
               % only save parameters.aq_per_rotor points every rotor
               decay_mat(rotorperiods_it*parameters.aq_per_rotor-
parame-
ters.aq_per_rotor+1:rotorperiods_it*parameters.aq_per_rotor,pulsedur_it,dec_power_it,irr_power_it)=coil_vec(1
,1:parameters.stepamount/parameters.aq_per_rotor:end);
            end
         end
      end
   end
end
end
% Consistency enforcement
function grumble(spin_system,parameters,H,R,K)
if (~isnumeric(H))||(~isnumeric(R))||(~isnumeric(K))||...
  (~ismatrix(H))||(~ismatrix(R))||(~ismatrix(K))
```

```
    error('H, R and K arguments must be matrices.');
end
if (~all(size(H)==size(R)))||(~all(size(R)==size(K)))
    error('H, R and K matrices must have the same dimension.');
end
if ~isfield(parameters,'spins')
    error('working spins should be specified in parameters.spins variable.');
end
if ~isfield(parameters,'rho0')
    error('initial state must be specified in parameters.rho0 variable.');
end
if ~isfield(parameters,'coil')
    error('detection state must be specified in parameters.coil variable.');
end
if (all(parameters.pulsedur_list==round(parameters.pulsedur_list)))~=1
    disp(parameters.pulsedur_list);
    error('all entries of parameters.pulsedur_list need to be natural numbers. It depends on parameters.pulses_per_rotor, parameters.pulsesteps_per_rotor, parameters.pulsedur_min, parameters.pulsedur_max, parameters.n_pulsedurations. This is needed for rotorsynchronization.');
end
end
```