Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Magnetic Resonance 176 (2005) 7–14
JMR Journal of Magnetic Resonance
www.elsevier.com/locate/jmr

MathNMR: Spin and spatial tensor manipulations in Mathematica

Alexej Jerschow

Chemistry Department, New York University, New York, NY 10003, USA

Received 13 December 2004; revised 4 May 2005
Available online 8 June 2005

Abstract

Spin and spatial tensor manipulations are frequently required to describe the theory of NMR experiments. A Mathematica package is presented here, which provides some of the most common functions for these calculations. Examples are the calculation of matrix representations of operators, commutators, projections, rotations, Redfield matrix elements, matrix decomposition into basis operators, change of basis, coherence filtering, and the manipulation of Hamiltonians. The calculations can be performed for any spin system, containing spins 1/2 and quadrupolar spins alike, subject to computational limitations. The package will be available from http://www.nyu.edu/projects/jerschow/ upon acceptance of the article.

Keywords: Spin tensors; Product operators; Redfield coefficients; Cross-correlated relaxation; Basis decomposition

1. Introduction

Mathematica is a widely used program for performing symbolic calculations [1]. A popular package written for Mathematica is POMA, which can be used for applying product operator transformation rules [2,3]. Its area of application is, however, limited to weakly coupled spin 1/2 systems. The purpose of the package described here is to provide more general applicability, and to offer functions that can be used with any spin system, for both liquid- and solid-state NMR tasks.

In contrast to numerical packages, such as Gamma [4], and SIMPSON [5], the emphasis here is on symbolic computations. While the former allow one to simulate the outcome of NMR experiments, the latter approach is useful in the design of such experiments, and in the derivation of the underlying theory. Frequently, theoretical models have to be developed to analyze the data obtained from NMR experiments. A good example for such a situation is the analysis of relaxation data, in which the Redfield coefficients have to be determined as a function of the nature of the spin system and depending on the interactions that are responsible for relaxation.

2. Products of tensor operators

In NMR we frequently decompose the density matrix in terms of basis operators Q_i as [6]

$$\rho = \sum_i v_i Q_i \tag{1}$$

with the convenient property of orthogonality

$$\langle Q_j | Q_i \rangle = \mathrm{Tr}\{Q_j^\dagger Q_i\} = a_i \delta_{ij}, \tag{2}$$

where δ_{ij} is the Kronecker delta. If $a_i = 1$ the operators are said to be normalized, but this is not necessarily always a good choice of basis. The operators are chosen to be orthogonal, so that one may determine how much each basis operator is contributing to a given density matrix, via the inverse of Eq. (1)

$$v_i = \langle Q_i | \rho \rangle / a_i. \tag{3}$$

In a multi-spin system composed of the spins $I_1, I_2, \ldots, I_n$ a convenient way of choosing the basis operators is by employing the outer product $\otimes$ to link the

E-mail address: alexej.jerschow@nyu.edu

1090-7807/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmr.2005.05.005

individual subspaces of the spins. A basis operator is then composed of a direct product of the basis operators of the subspaces

$$Q_i = Q_i^{(1)} \otimes Q_i^{(2)} \otimes \cdots Q_i^{(n)}. \tag{4}$$

Often, one omits writing the symbol $\otimes$ in these expressions. One also normally omits writing the identity operators in such products.

The most convenient way of spanning the subspaces is to use either spherical tensors

$$Q_i^{(j)} = T_{L_i,M_i}^{(j)} \tag{5}$$

or cartesian tensors

$$Q_i^{(j)} \in \{E, I_x, I_y, I_z\}, \tag{6}$$

where E is the identity operator.

Cartesian operators are less convenient for quadrupolar nuclei, since they do not span the whole space without considering products and sums of these operators.

The nonzero matrix elements for the cartesian operators are defined as [7,8]

$$\begin{aligned} \langle m+1|I_+|m\rangle &= \sqrt{I(I+1)-m(m+1)} \\ \langle m+1|I_-|m\rangle &= \sqrt{I(I+1)-m(m-1)} \\ \langle m|I_z|m\rangle &= m \end{aligned} \tag{7}$$

and $I_x = (I_+ + I_-)/2$ and $I_y = i(I_- - I_+)/2$.

The package provides spherical tensor operators using two common conventions [9], $T_{L,M}^{(a)}$ and $T_{L,M}^{(b)}$. The nonzero matrix elements of these are defined as

$$\langle m+M|T_{L,M}^{(a)}|m\rangle = \langle Im, LM|I(m+M)\rangle \sqrt{\frac{2L+1}{2I+1}} \tag{8}$$

and

$$\langle m+M|T_{L,M}^{(b)}|m\rangle = \langle Im, LM|I(m+M)\rangle L! \times \sqrt{\frac{(2I+L+1)!}{2^L(2L)!(2I-L)!(2I+1)}}, \tag{9}$$

which differ in their normalizations.

The use of $T_{L,M}^{(a)}$ is convenient for creating tensor operators which are automatically normalized, according to $\mathrm{Tr}(T_{L,M}^{(a)\dagger} T_{L,M}^{(a)}) = 1$ (including $T_{00}^{(a)} = E/\sqrt{2I+1}$, where E is the identity matrix).

The use of $T_{L,M}^{(b)}$ is convenient as it more properly corresponds to the operators used in NMR, i.e., $I_z = T_{1,0}^{(b)}$, and $E = T_{0,0}^{(b)}$. It also provides the spin-independent relationship of $I_\pm = \mp\sqrt{2} T_{1,\pm}^{(b)}$. Using this normalization, one may also couple tensors acting on the same space easily according to

$$T_{L,M}^{(b)} = \sum_{m_1,m_2} \langle l_1 m_1, l_2 m_2 | LM\rangle T_{l_1,m_1}^{(b)} T_{l_2,m_2}^{(b)} \tag{10}$$

without the need for additional Wigner coefficients.

The `mrep` function produces the matrix representations of these operators based on notation b, which corresponds to the regular NMR tensors. In the following the superscript b will be dropped. Individual tensors using the alternate normalization can be produced using the function `sphtens`.

3. Operator representation and setting up the spin system

The operators can be represented in terms of products of individual spin operators (using the dot product '.'), or in terms of a single operator. For example, `ispinT[1, 2, -1].ispinT[3, 1, -1]` describes a product operator between a spherical tensor $T_{2,-1}$ of spin one and a tensor $T_{1,-1}$ of spin 3. This can alternatively be represented by `spinT[2, -1, 0, 0, 1, -1]` in a three-spin system (the second spin having rank zero as well as order zero, which is equivalent to the identity operator in the normalization used in the package). Likewise, the cartesian tensor expressions `ispinC[1, x].ispinC[3, z]` and `spinC[x, e, z]` represent the same operator, namely $I_{1x}I_{3z}$. In addition, the raising and lowering operators are defined as `ispinC[i, p]`=I_+, and `ispinC[i, m]`=I_-. Even though they are not cartesian tensors, it seemed appropriate, for the sake of simplicity, to implement them through the `ispinC` command.

To facilitate the examination of the Mathematica calculations, use is made of output formatting options, such that the operator `ispinC[i, x]` is represented as I_{ix}, as is customary, `ispinC[i, p]` is shown as I_{i+}, `ispinC[i, m]` is shown as I_{i-}. An operator `ispinT[i, L, M]` is represented in the output as $T_{L,M}^{(i)}$.

To be able to calculate the matrix representations of the operators it is necessary to define the spin system. This is done by the command `setSpinSys[I1, I2, ...]`, where `I1`, `I2`, etc., are the spin values of the nuclei in the spin system.

Probably the most useful command is `mrep[expr]`, which calculates the matrix representations of an expression `expr` containing the operators in the forms of `spinT`, `spinC`, `ispinT`, or `ispinC` or products thereof, so for example the statements

```
<<MathNMR.m
setSpinSys[1, 1/2]
mrep[ispinT[1, 2, 1].ispinC[2, x]]
```

produce the matrix

$$\begin{pmatrix} 0 & 0 & 0 & \frac{1}{2\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{2\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-1}{2\sqrt{2}} \\ 0 & 0 & 0 & 0 & \frac{-1}{2\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{11}$$

for the spin system consisting of a spin 1 and a spin 1/2 nucleus. The command <<MathNMR.m reads in the package. Single-transition operators are provided through the function ispinST[i, ml, m], which represent transitions between the levels ml and m for a particular spin.

4. Commutators, basis decomposition, change of basis, projection operators, and coherence order filtering

A commonly used task is to change bases, or to determine the basis decomposition of an operator expression, or a matrix representation. This is achieved by the commands spinCDecomposition and spinTDecomposition, which gives the basis decomposition in terms of cartesian or spherical tensor operators, respectively. The following example calculates the decomposition of the operator $T^{(1)}_{2,1}.I_{2x}$ in terms of spherical tensor operators.

```
setSpinSys[1, 1/2]
out=mrep[ispinT[1, 2, 1].ispinC[2, x]]
decomp=spinTDecomposition[out]
```

produces

$$\frac{1}{\sqrt{2}}\left(T^{(1)}_{2,1}.T^{(2)}_{1,-1} - T^{(1)}_{2,1}.T^{(2)}_{1,1}\right). \tag{12}$$

The command commutator[A, B] produces the commutator between the matrices A and B. For convenience, if the operator decomposition is required, the commands spinCCommutator and spinTCommutator are provided.

The statements

```
setSpinSys[{1, 1/2}]
spinTCommutator[ispinT[1, 2, 1]
  .ispinC[2, x],
  ispinT[1, 2, 0]. ispinC[2, y]]
```

produce

$$\left[T^{(1)}_{2,1}.I_{2x}, T^{(1)}_{2,0}.I_{2y}\right] = \frac{i}{2\sqrt{6}}T^{(1)}_{2,1}.T^{(2)}_{1,0}. \tag{13}$$

Frequently, one requires a representation using the $I_{\pm}$ operators instead of $T_{1,\pm1}$. The command convertTtopm replaces the operators $T_{1,0}$, and $T_{1,\pm1}$ by their counterparts of I_z, $I_{\pm}$ with the appropriate scaling factors.

The command

```
convertTtoPM[ispinT[1, 1, 1].
  ispinT[2, 1, -1]]
```

produces

$$-\tfrac{1}{2}I_{1+}.I_{2-}. \tag{14}$$

Another example illustrates the representation of the pseudo-pure state [10] in this spin 1–spin 1/2 system. One of these states can be obtained by writing

```
a=mrep[ispinST[1, -1, -1].ispinST[2,
  -1/2, -1/2]]
```

which gives

$$a = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{15}$$

spinTDecomposition[a] then produces

$$\tfrac{1}{12}[2E + 6T^{(1)}_{1,0}.T^{(2)}_{1,0} - 2\sqrt{6}T^{(1)}_{2,0}.T^{(2)}_{1,0} - 3T^{(1)}_{1,0} + \sqrt{6}T^{(1)}_{2,0} - 4T^{(2)}_{1,0}]. \tag{16}$$

One may project operator expressions and matrix representations onto subspaces of the Hilbert space by using the functions projectionOp and oneMinusProjectionOp. The commands

```
a = ispinST[1, -1, 0].ispinC[2, x]
    +(1/2)ispinST[1, 0, 1].ispinC[2, y]
a = mrep[a]
b = projectionOp[{ispinT[1, 2, 2]
     .ispinT[2, 1, 1],
     ispinT[1, 1, 1].ispinT[2, 1, 1]}, a]
out = spinTDecomposition[b]
```

produce

$$\frac{2-\mathrm{i}}{4\sqrt{2}}\left[T^{(1)}_{1,1}.T^{(2)}_{1,1}\right], \tag{17}$$

which represents the projection of operator a containing the single transitions as above onto the subspace spanned by $\{T^{(1)}_{2,2}.T^{(2)}_{1,1}, T^{(1)}_{1,1}.T^{(2)}_{1,1}\}$.

We can obtain the projection onto the space outside of this subspace by using

```
b = oneMinusProjectionOp[{ispinT[1, 2, 2]
   .ispinT[2, 1, 1],
   ispinT[1, 1, 1].ispinT[2, 1, 1]}, a]
```

in which case the result for out will be

$$\tfrac{1}{8}\left[-(2+i)\sqrt{2}T^{(1)}_{1,1}.T^{(2)}_{1,-1} + (4-2i)T^{(1)}_{2,1}.T^{(2)}_{1,-1} - (4+2i)T^{(1)}_{2,1}.T^{(2)}_{1,1}\right]. \tag{18}$$

The function filterCoherence[p1, p2, ..., expr] filters the operator expression expr for terms

that have coherence orders $p1$, $p2$, etc. for the individual spins. The statements

```
a=ispinT[1, 1, 1].ispinT[2, 1, 1]
    +ispinT[1, 2, 2].ispinT[2, 1, 1]
    +ispinT[1, 2, 2].ispinT[2, 1, 0]
filterCoherence[{2, 1}, a]
```

produces $T^{(1)}_{2,2}.T^{(2)}_{1,1}$. One can also supply the keyword `any` which allows any coherence order for a particular spin. For example

```
filterCoherence[{2, any}, a]
```

produces $T^{(1)}_{2,2}.T^{(2)}_{1,1} + T^{(1)}_{2,2}.T^{(2)}_{1,0}$ instead. This function is useful, for example, for obtaining secular components of Hamiltonians (by setting the coherence orders to zero). If coherence filtering is required for a matrix `mtx` rather than operators, the expression `cohfltM[p1, p2, ...]*mtx` can be used.

5. Spatial tensors

Spatial tensors are implemented in Mathematica as functions. For example, the tensor for the dipolar coupling between spins i and j is input as `wD[i, j, m]`, and each of the m-components can be assigned a value. To facilitate display, an output form is defined to show the component as $\omega^{(i,j)}_{D,m}$. Similarly, the CSA tensors are input as `wcsa[i, m]` and output as $\omega^{(i)}_{CSA,m}$, and the quadrupolar tensors as `wQ[i, m]` and $\omega^{(i)}_{Q,m}$, respectively. The J-coupling constants are input as `J[i, j]` and output as $J^{(i,j)}$, and the isotropic chemical shift is set via `wiso[i]` and output as $\omega^{(i)}_{iso}$.

While an object-oriented implementation of these tensors would seem most practical (as, for example, in GAMMA [4]), in particular with respect to assignment, copying, and rotating tensors, Mathematica does not support objects very well, at least not in a convenient manner. It was therefore decided to implement these tensors as functions. The function `copyTensor[a, b]` can be used to copy tensor a to tensor b. The function `makeSpatialD[wD, i, j, d, α, β, γ]` produces a spatial dipolar coupling tensor in `wD[i, j]` with the coupling constant d, rotated from its principal axis frame by the Euler angles α, β, γ (vide infra). Similarly, `makeSpatialCSA` and `makeSpatialQ` can be used for the CSA and the quadrupolar interactions.

6. Hamiltonians

`makeHDD[i, j]` creates a Hamiltonian for the dipolar coupling between spins i and j, according to

$$H^{(i,j)}_D = \sum_{m=-2}^{2} (-1)^m \omega^{(i,j)}_{D,-m} \sqrt{6} T^{(i,j)}_{2,m}, \tag{19}$$

where

$$\begin{aligned} T^{(i,j)}_{2,0} &= \frac{1}{\sqrt{6}}\left(3I_{i,z}I_{j,z} - I_i \cdot I_j\right), \\ T^{(i,j)}_{2,\pm 1} &= \mp\tfrac{1}{2}\left(I_{i,\pm}I_{j,z} + I_{i,z}I_{j,\pm}\right), \\ T^{(i,j)}_{2,\pm 2} &= \tfrac{1}{2} I_{i,\pm} I_{j,\pm}. \end{aligned} \tag{20}$$

The spatial tensor components $\omega^{(i,j)}_{D,m}$ are obtained via a rotation from the principal axis system (PAS), in which the only nonzero component is $\omega^{(i,j),PAS}_{D,0} = -\frac{\mu_0}{4\pi}\frac{\gamma_i\gamma_j\hbar}{r^3}$ [11] in angular frequency units. If the tensor is created via `makeSpatialD` the correct values are set automatically.

`makeHDD[i, j, m]` creates the m-order component, so for example the command

```
makeHDD[1, 2, 0]
```

produces

$$\omega^{(1,2)}_{D,0}\left[T^{(1)}_{1,-1}.T^{(2)}_{1,1} + 2T^{(1)}_{1,0}.T^{(2)}_{1,0} + T^{(1)}_{1,1}.T^{(2)}_{1,-1}\right]. \tag{21}$$

Using

```
convertTtoPM[makeHDD[1, 2, 0]]
```

one obtains the more familiar form

$$\tfrac{1}{2}\omega^{(1,2)}_{D,0}[4I_{1z}.I_{2z} - I_{1+}.I_{2-} - I_{1-}.I_{2+}]. \tag{22}$$

`makeHCSiso[i]` produces the isotropic shift Hamiltonian and `makeHCSA[i]` produces the second-rank chemical shift anisotropy (CSA) Hamiltonian for spin i according to

$$H_{CSA} = \omega_{CSA,0} I_z + \sqrt{\frac{3}{8}}\omega_{CSA,-1} I_+ - \sqrt{\frac{3}{8}}\omega_{CSA,1} I_-, \tag{23}$$

where, in the PAS system,

$$\begin{aligned} \omega_{CSA,0} &= \delta_{CSA}, \\ \omega_{CSA,\pm 1} &= 0, \\ \omega_{CSA,\pm 2} &= \frac{\delta_{CSA}\eta}{\sqrt{6}}, \end{aligned} \tag{24}$$

and δ_{CSA} is the principal component of the tensor in the PAS given in radians, and η is the anisotropy factor [11]. The components are automatically set correctly if `makeSpatialCSA` is used.

`makeHJ[i, j]` makes the J-coupling Hamiltonian according to

$$H^{i,j}_J = 2\pi J^{(i,j)} I^i \cdot I^j. \tag{25}$$

`makeHQ[i]` produces the quadrupolar Hamiltonian for spin i according to

$$H^{(i)}_Q = \sum_{m=-2}^{m} (-1)^m \omega^{(i)}_{Q,-m} T^{(i)}_{2,m}, \tag{26}$$

where, using the common definition of $\omega_Q = \frac{2\pi C_Q}{2I(2I-1)}$ [12] the tensor components in the PAS are

$$\omega_{Q,0} = \sqrt{\frac{3}{2}}\omega_Q,$$
$$\omega_{Q,\pm 1} = 0,$$
$$\omega_{Q,\pm 2} = \omega_Q \eta/2. \tag{27}$$

The components are automatically set correctly if `makeSpatialQ` is used.

7. Rotations and pulses

7.1. Rotation matrices

The Wigner rotation elements (command `wignerD`) are defined as

$$D^l_{m',m}(\alpha,\beta,\gamma) = \exp(-\mathrm{i}\alpha m' - \mathrm{i}\gamma m) d^l_{m',m}(\beta) \tag{28}$$

and the reduced rotation elements (command `reducedD`) are defined as

$$d^l_{m',m}(\beta) = \sqrt{(l+m')!(l-m')!(l+m)!(l-m)!} \times \sum_s \frac{(-1)^{m'-m+s}\left(\cos\frac{\beta}{2}\right)^{2l+m-m'-2s}\left(\sin\frac{\beta}{2}\right)^{m'-m+2s}}{(l+m-s)!s!(m'-m+s)!(l-m'-s)!}. \tag{29}$$

The summation over s includes only terms for which the arguments of the factorials are non-negative [7,8].

`wignerRotMtx[l, α, β, γ]` produces the full rotation matrix of rank l.

Another rotation parameterization is also provided in the function `dER`: The Euler–Rodriguez parameterization is useful if the axis of rotation and the rotation angle are specified (as opposed to the Euler angles) [13,14]. The rotation element is calculated by

$$D^{l,ER}_{m',m}(\theta,\mathbf{n}) = \sqrt{(l+m')!(l-m')!(l+m)!(l-m)!} \times \sum_s \frac{a^{j-m'-s}(a^*)^{j+m-s}(-b)^{m'-m+s}(b^*)^s}{(l+m-s)!s!(m'-m+s)!(l-m'-s)!}. \tag{30}$$

where θ is the rotation angle around the axis described by the normalized vector $\boldsymbol{n} = (n_1, n_2, n_3)$, and

$$a = \cos(\theta/2) + \mathrm{i}n_3 \sin(\theta/2),$$
$$b = n_2 \sin(\theta/2) + \mathrm{i}n_1 \sin(\theta/2). \tag{31}$$

Note that $D^{l,ER}_{m',m}[\beta,(0,1,0)] = d^l_{m',m}(\beta)$.

The relationship between the Euler and the Euler–Rodriguez angles is given as:[13]

$$\cos(\theta/2) = \cos(\beta/2)\cos\frac{\gamma+\alpha}{2},$$
$$n_1 \sin(\theta/2) = \sin(\beta/2)\sin\frac{\gamma-\alpha}{2},$$
$$n_2 \sin(\theta/2) = \sin(\beta/2)\cos\frac{\gamma-\alpha}{2},$$
$$n_3 \sin(\theta/2) = \cos(\beta/2)\sin\frac{\gamma+\alpha}{2}. \tag{32}$$

7.2. Rotation of spatial and spin tensors

Spatial tensors are rotated using the function `rotateSpatial[ω, i, α, β, γ]`, where i specifies the spin index or indices, and α, β, γ the Euler angles. As discussed above, rotations from the principal-axis system can be performed using the `makeSpatialQ`, `makeSpatialCSA`, and `makeSpatialD` functions. So, for example, `makeSpatialD[wD, 1, 2, d, 0, β, 0]` sets up the dipolar coupling tensor between spins one and two, rotated from the principal-axis system by an angle β. The subsequent command `makeHDD[1, 2, 0]` creates the secular part of the dipolar coupling as follows:

$$\tfrac{1}{4}(1+3\cos 2\beta) d\left[T^{(1)}_{1,-1}.T^{(2)}_{1,1} + 2T^{(1)}_{1,0}.T^{(2)}_{1,0} + T^{(1)}_{1,1}.T^{(2)}_{1,-1}\right]. \tag{33}$$

Rotation matrices for the product spin space are obtained using the function `spinRotMtx[i, α, β, γ]`. The following commands will produce an Euler rotation of spin 1 in the operator $T^{(1)}_{2,2}.I_{2z}$ by $(\alpha,\beta,\gamma) = (\pi, \pi/2, \pi/2)$:

```
setSpinSys[1, 1/2]
R=spinRotMtx[1, Pi, Pi/2, Pi/2]
out1=mrep[ispinT[1, 2, 2].ispinC[2, z]]
out2=R.out1.dagger[R]
out3=spinTDecomposition[out2]
```

where `dagger` returns the Hermitian conjugate matrix. The result of this calculation is

$$\tfrac{1}{4}\Big[-T^{(1)}_{2,-2}.T^{(2)}_{1,0} + 2T^{(1)}_{2,-1}.T^{(2)}_{1,0} - \sqrt{6}T^{(1)}_{2,0}.T^{(2)}_{1,0} + 2T^{(1)}_{2,1}.T^{(2)}_{1,0} - T^{(1)}_{2,2}.T^{(2)}_{1,0}\Big]. \tag{34}$$

7.3. Rf pulses

For convenience, rf-pulses can be applied to matrices via the function `pulse[mtx, i, α, φ, ψ]`. This will perform a pulse on the i-spin subspace of flip angle α, phase ϕ, where the effective field is tilted by the angle ψ from the transverse plane. If $\psi = 0$ this represents an on-resonance pulse. For example

```
setSpinSys[3/2, 3/2]
a=mrep[ispinC[1, z].ispinC[2, z]];
b=pulse[a, 1, Pi/4, 0, 0]
spinTDecomposition[b]
```

gives

$$\tfrac{1}{2}\left[iT^{(1)}_{1,-1}.T^{(2)}_{1,0} + \sqrt{2}T^{(1)}_{1,0}.T^{(2)}_{1,0} + iT^{(1)}_{1,1}.T^{(2)}_{1,0}\right]. \tag{35}$$

8. Redfield relaxation matrix elements

Using the expansion of the density matrix into mutually orthogonal operators Q_i of the form

$$\rho(t) = \sum_i v_i(t) Q_i, \tag{36}$$

the relaxation dynamics can be written in the form

$$v_i(t) = -\sum_j \Gamma_{i,j} v_j(t). \tag{37}$$

The interactions responsible for relaxation are written as

$$H_1 = \sum_r \sum_m (-1)^m F^r_m(t) A^r_m \tag{38}$$

where r is an index that runs through the individual mechanisms (dipolar, CSA, quadrupolar, etc.). The operators A^r_m can be expanded in terms of a basis $A^r_{m,p}$

$$A^r_m = \sum_m A^r_{m,p} \tag{39}$$

such that

$$[H_0, A^r_{m,p}] = \omega_{m,p} A^r_{m,p}. \tag{40}$$

It also follows that $\omega_{m,p} = -\omega_{-m,p}$.

Given these definitions, $\Gamma_{i,j}$ can be evaluated by [15–18]

$$\Gamma_{i,j} = \frac{1}{2} \sum_{r,r'} \sum_{p,m} J^{r'}_r(\omega_{m,p}) \frac{\mathrm{Tr}\left\{ [A^r_{m,p}, Q_i]^\dagger [A^{r'}_{m,p}, Q_j] \right\}}{\mathrm{Tr}\{Q_i^\dagger Q_i\}}. \tag{41}$$

The function `RedfieldCoeff` calculates these coefficients for two given operators Q_i and Q_j.

In order for this to work properly one needs to indicate which interactions should be considered for the relaxation mechanisms, and whether homo- or heteronuclear cases are involved. This is done by using the function `setSpinSys[Il, I2, ..., wl, w2, ..., hamiltonianlist]` to set up the spin system. $w1$, $w2$, etc. are the Larmor frequencies, which may be just symbols without an assigned value. If the same variable is used for different spins, then automatically a homonuclear case is assumed [15]. `hamiltonianlist` contains a list defining the types of Hamiltonians to be considered as relaxation mechanisms.

As an example, we calculate here the TROSY [19] relationships for the two transitions $I_{1-}.I_{2\alpha}$ and $I_{1-}.I_{2\beta}$. These correspond to the transitions labeled S^-_{12} and S^-_{34} in [19]. The dipolar coupling and the two CSA interactions are considered as relaxation mechanisms.

```
setSpinSys[{1/2, 1/2}, {w1, w2},
  {"DD", 1, 2, "CS", 1, "CS", 2}];
Sl2=ispinC[1, m].ispinST[2, -1/2, -1/2];
S34=ispinC[1, m].ispinST[2, 1/2, 1/2];
Sl2rate=RedfieldCoeff[Sl2, Sl2];
S34rate=RedfieldCoeff[S34, S34];
```

The results are

$$\begin{aligned} S12\mathrm{rate} = \frac{1}{8}\Big[& 4J^{CS,1}_{CS,1}(0) + 3J^{CS,1}_{CS,1}(\omega_1) + 3J^{CS,2}_{CS,2}(\omega_2) \\ & + 8J^{CS,1}_{DD,1,2}(0) - 6J^{CS,1}_{DD,1,2}(\omega_1) + 4J^{DD,1,2}_{DD,1,2}(0) \\ & + 3J^{DD,1,2}_{DD,1,2}(\omega_1) + J^{DD,1,2}_{DD,1,2}(\omega_1 - \omega_2) \\ & + 3J^{DD,1,2}_{DD,1,2}(\omega_2) + 6J^{DD,1,2}_{DD,1,2}(\omega_1 + \omega_2)\Big] \end{aligned} \tag{42}$$

and

$$\begin{aligned} S34\mathrm{rate} = \frac{1}{8}\Big[& 4J^{CS,1}_{CS,1}(0) + 3J^{CS,1}_{CS,1}(\omega_1) + 3J^{CS,2}_{CS,2}(\omega_2) \\ & + 8J^{CS,1}_{DD,1,2}(0) + 6J^{CS,1}_{DD,1,2}(\omega_1) + 4J^{DD,1,2}_{DD,1,2}(0) \\ & + 3J^{DD,1,2}_{DD,1,2}(\omega_1) + J^{DD,1,2}_{DD,1,2}(\omega_1 - \omega_2) + 3J^{DD,1,2}_{DD,1,2}(\omega_2) \\ & + 6J^{DD,1,2}_{DD,1,2}(\omega_1 + \omega_2)\Big]. \end{aligned} \tag{43}$$

The cross-correlation spectral density functions, displayed as $J^r_{r'}(\omega)$, or rather as $J^{\mathrm{int2}}_{\mathrm{int1}}(\omega)$ can be set via the function `sdfJ[intl, int2, w]`, where `intl` and `int2` specify the interactions considered. For isotropic tumbling with a correlation time τ_c and no internal motion it is given by [17,18]

$$J^r_{r'}(\omega) = c_r c_{r'} P_2(\cos\theta_{r,r'}) \frac{2}{5} \frac{\tau_c}{1 + (\omega\tau_c)^2}, \tag{44}$$

where $\theta_{r,r'}$ is the angle between the largest principal-axis components of the two interactions, and the constants c_r are defined (in accordance with the definitions of the Hamiltonians above) as

$$c_r = \begin{cases} -\frac{\mu_0}{4\pi}\frac{\gamma_i\gamma_j\hbar}{r^3} & \text{if } r = \text{dipolar} \\ \frac{2}{3}\Delta\sigma & \text{if } r = \text{CSA} \\ \sqrt{\frac{3}{2}}\frac{e^2qQ}{\hbar 2I(2I-1)} & \text{if } r = \text{quadrupolar} \end{cases}. \tag{45}$$

Another, perhaps more elaborate example shows the calculation of several cross- and auto-relaxation rates for a heteronuclear spin system containing two 1/2 spins and one 1 spin. In this case, we consider all dipolar, all CSA, and the quadrupolar interactions for the relaxation processes.

```
setSpinSys[{1/2, 1, 1/2}, {w1, w2, w3},
  {"DD", 1, 2, "DD", 1, 3, "DD", 2, 3,
  "CS", 1, "CS", 2, "CS", 3, "Q", 2}]
out1=RedfieldCoeff[ispinC[1, x],
  ispinC[1, x].ispinC[2, z]]
out2=RedfieldCoeff[ispinC[2, x],
  ispinC[2, x].ispinC[1, z]]
out3=RedfieldCoeff[ispinC[2, x],
  ispinC[2, x]]
```

The results are

$$\begin{aligned} out1 &= \tfrac{4}{3}J^{CS,1}_{DD,1,2}(0) + J^{CS,1}_{DD,1,2}(\omega_1), \\ out2 &= \tfrac{1}{8}\Big[J^{CS,1}_{DD,1,2}(0) + 3J^{CS,1}_{DD,1,2}(\omega_2)\Big], \\ out3 &= \tfrac{1}{8}\Big[4J^{CS,2}_{CS,2}(0) + 3J^{CS,2}_{CS,2}(\omega_2) + 6J^{Q,2}_{Q,2}(0) + 10J^{Q,2}_{Q,2}(\omega_2) \\ &\quad + 4J^{Q,2}_{Q,2}(2\omega_2) + 4J^{DD,1,2}_{DD,1,2}(0) + 6J^{DD,1,2}_{DD,1,2}(\omega_1) \\ &\quad + J^{DD,1,2}_{DD,1,2}(\omega_1 - \omega_2) + 3J^{DD,1,2}_{DD,1,2}(\omega_2) \\ &\quad + 6J^{DD,1,2}_{DD,1,2}(\omega_1 + \omega_2) + 4J^{DD,2,3}_{DD,2,3}(0) + 3J^{DD,2,3}_{DD,2,3}(\omega_2) \\ &\quad + J^{DD,2,3}_{DD,2,3}(\omega_2 - \omega_3) + 6J^{DD,2,3}_{DD,2,3}(\omega_3) \\ &\quad + 6J^{DD,2,3}_{DD,2,3}(\omega_2 + \omega_3)\Big]. \end{aligned} \tag{46}$$

9. Evolution and calculation of spectra

Although it is not one of the major objectives of this package to perform numerical calculations, some functions are added to help in calculating spectra or to perform time propagation of operators. For example, `evolve[rho, H, t]` calculates the time evolution of `rho` under the action of the Hamiltonian `H` for a time `t`. Similarly, the function `acquire[rho, H, dt, A, N]` assembles a series of `N` data points in a vector, where the time evolution of `rho` is governed by `H`, the sampling interval is `dt`, and the detection operator is `A`.

Numerical calculations are usually not as rapid in Mathematica as they are with dedicated numerical packages, and the calculation of time evolution may be slow, especially for larger spin systems. A function was therefore added, which calculates the spectra directly from the combination of Hamiltonian, initial density operator, and detection operator. This is performed by diagonalizing the Hamiltonian first, then putting both the initial density operator and the detection operator into the Hamiltonian basis. The transition frequencies between the states i and j are then given by $(H_{ii} - H_{jj})/2\pi$, and the amplitudes by $A_{ij}\rho_{ji}$, where ρ is the initial density operator, and A the detection operator. This is performed via the `getspec[rho, H, A, N, frange]` statement, where `N` is the number of points to be used in the spectrum, and `frange` specifies the frequency range to be used.

The following example calculates a proton spectrum of benzene in a liquid crystal, with the benzene ring arranged perpendicularly with respect to the magnetic field. Here, we assume an order parameter of 0.01 and bond lengths of 1.39 (C–C) and 1.09 Å (C–H). The code below sets up the spin system and calculates the dipolar coupling constants.

```
setSpinSys[{1/2, 1/2, 1/2, 1/2, 1/2, 1/2}];
coord=2.48*10^(-10)Table[Cos[a],
Sin[a], {a, 0, 2Pi-2Pi/n, 2Pi/n}];
orderpar=0.01;
mu0pi=10^(-7);
hbar=1.05457*10^(-34);
```

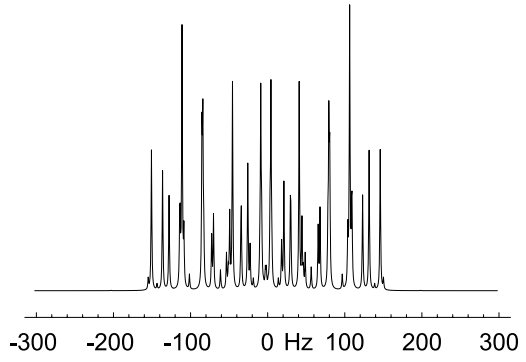

Fig. 1. Calculated proton spectrum of benzene in a liquid crystal.

```
gammaH=2.67522*^8;
prefactor=-orderpar*mu0pi*gammaH^2*hbar;

calcDD[n_]:=
  Module[{distance},
    For[i=1, i<=(n+1), i++,
      For[j=i+1, j<=n, j++,
        distance=
          Sqrt[(coord[[i, 1]]-
             coord[[j, 1]])^2
            +(coord[[i, 2]]-
             coord[[j, 2]])^2];
          wD[i, j, 0]=0.5*prefactor/
              distance^3;
          (*factor 0.5 for perpendicular
             arrangement*)
        ]
      ]
    ];
calcDD[6];
```

The code below calculates the Hamiltonian containing all dipolar couplings and calculates and displays the spectrum.

```
TotalHDD[n_]:=Sum[Sum[makeHDD[i, j, 0],
    {i, j+1, n}], {j, 1, n}];
H0=TotalHDD[6]//mrep;
{scale, spec} =
  getspec[mrep[ispinC[1, X]], H0,
       mrep[ispinC[1, m]], 1000,
       {-300, 300}];
spec = speclb[spec, 5];
plotspec[scale, spec]
```

The `speclb` command is used to add some line-broadening to the spectrum. The resulting spectrum is shown in Fig. 1.

10. Mathematica details

10.1. Computational limits

The package itself does not limit the numbers of spins in a spin system, but clearly computational and memory resources impose an upper bound on the size of a manageable calculation. As an indication of performance of this package we give here the memory and CPU time usage for the calculation of the Redfield coefficients shown in the example of the previous section. The memory indications are recorded via Mathematica's `MemoryInUse[]` statement. These numbers do not reflect temporary or peak memory usage. The CPU time is recorded from Mathematica's `Timing` statement.

The calculations were performed on a PC with Intel Pentium III, 850 MHz, 128 MB RAM, running Linux 2.4.20 and Mathematica 4.2. Reading in the package with `<<MathNMR.m` uses 167 kb of memory. The `setSpinSys` statement uses 1.5 s CPU time and 728 kb of memory. This command is used to set up the spin system, to create all Hamiltonian matrix representations for later use, and sort these according to the transition frequencies that they induce. The calculations of *out*1, *out*2, and *out*3 use each approximately 20 s CPU time, and 238, 92, and 44 kb of memory, respectively.

10.2. Compatibility and syntax

The package was tested with Mathematica version 4.2. It appears to run on version 3.0, however, the output formatting does not always produce the desired results. This can probably be fixed by adapting the `Format` statements in the package to version 3.0.

The product between the spin operators is denoted by the regular dot product. This choice makes it easier to calculate the matrix representations. It is essential to not confuse this product with the regular product (asterisk or blank in Mathematica), since then the `mrep` function will not work properly. In this case, Mathematica would try to multiply the matrices element-wise.

11. Conclusions

A Mathematica package is described which performs symbolic calculations of many spin and spatial tensor manipulations that are commonly used to describe NMR experiments. Such tasks as basis change, rotations, and the calculation of commutators, operator decompositions, and Redfield matrix elements can be performed very conveniently for different spin systems. Major future developments of the package may include the calculation of operator expressions using subspaces to decrease the computational limits. Minor additions may include the calculation of Redfield elements in tilted frames and the calculation of Average Hamiltonian expansions. The package will be available for download, including the documentation of the individual functions and examples of the calculations.

References

[1] Wolfram Research, Mathematica, http://www.wolfram.com/ (2004)
[2] P. Güntert, N. Schaefer, G. Otting, K. Wüthrich, POMA: A complete mathematica implementation of the NMR product-operator formalism, J. Magn. Reson. Ser. A 101 (1993) 103–105.
[3] P. Güntert, N. Schaefer, G. Otting, K. Wüthrich, Erratum: "POMA: A complete Mathematica implementation of the NMR product-operator formalism" by P. Güntert, N. Schaefer, G. Otting, K. Wüthrich, Ser. A, vol. 101, No. 1 (1993), pp. 103–105, J. Magn. Reson. Ser. A 103 (1993) 328.
[4] S.A. Smith, T.O. Levante, B.H. Meier, R.R. Ernst, Computer simulations in magnetic resonance. An object oriented programming approach, J. Magn. Reson. Ser. A 106 (1994) 75–105.
[5] M. Bak, J.T. Rasmussen, N.C. Nielsen, SIMPSON: A general simulation program for solid-state NMR spectroscopy, J. Magn. Reson. 147 (2000) 296–330.
[6] R.R. Ernst, G. Bodenhausen, A. Wokaun, Principles of Nuclear Magnetic Resonance in One and Two Dimensions, Clarendon Press, Oxford, 1987.
[7] R.N. Zare, Angular Momentun: Understanding Spatial Aspects in Chemistry and Physics, Wiley, New York, 1987.
[8] M.E. Rose, Elementary Theory of Angular Momentum, Wiley, New York, 1966.
[9] G.J. Bowden, W.D. Hutchinson, Tensor operator formalism for multiple-quantum NMR. 1. Spin-1 nuclei, J. Magn. Reson. 67 (1986) 403–414.
[10] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, 2000.
[11] M. Mehring, Principles of High-resolution NMR in Solids, second ed., Springer, Berlin, 1983.
[12] A. Jerschow, From nuclear structure to the quadrupolar NMR hamiltonian and high-resolution spectroscopy, Prog. Nucl. Magn. Reson. Spectrosc. 46 (2005) 63–78.
[13] L.C. Biedenharn, J.D. Louck, Angular Momentum in Quantum Physics, Addison Wesley, London, 1981.
[14] A. Jerschow, Nonideal rotations in NMR: Estimation of coherence transfer leakage, J. Chem. Phys. 113 (2000) 979–986.
[15] A. Abragam, Principles of Nuclear Magnetism, Oxford Science Publications, Oxford, 1961.
[16] D. Canet, Construction, evolution and detection of magnetization modes designed for treating longitudinal relaxation of weakly coupled spin 1/2 systems with magnetic equivalence, Prog. Nucl. Magn. Reson. Spectrosc. 21 (1989) 237–291.
[17] R. Brüschweiler, Connections between NMR relaxation measurements and theoretical models of structural dynamics of biopolymers in solution, in: R. Tycko (Ed.), Nuclear Magnetic Resonance Probes of Molecular Dynamics, Kluwer, Amsterdam, 1994, pp. 301–334.
[18] D. Früh, Internal motions in proteins and interference effects in nuclear magnetic resonance, Prog. Nucl. Magn. Reson. Spectrosc. 41 (2002) 305–324.
[19] K. Pervushin, R. Riek, G. Wider, K. Wüthrich, Attenuated T_2 relaxation by mutual cancellation of dipole-dipole coupling and chemical shift anisotropy indicates an avenue to NMR structures of very large biological macromolecules in solution, Proc. Natl. Acad. Sci. 94 (1997) 12366–12371.