Journal of Magnetic Resonance 207 (2010) 95–113

Contents lists available at ScienceDirect

Journal of Magnetic Resonance

journal homepage: www.elsevier.com/locate/jmr

SD-CAS: Spin Dynamics by Computer Algebra System

Xenia Filip, Claudiu Filip *

National Institute for R&D of Isotopic and Molecular Technologies, P.O. Box 700, 400293 Cluj, Romania

ARTICLE INFO

Article history:
Received 23 June 2010
Revised 4 August 2010
Available online 24 August 2010

Keywords:
Spin dynamics
Symbolic computations
NMR
Computer Algebra Systems

ABSTRACT

A computer algebra tool for describing the Liouville-space quantum evolution of nuclear 1/2-spins is introduced and implemented within a computational framework named Spin Dynamics by Computer Algebra System (SD-CAS). A distinctive feature compared with numerical and previous computer algebra approaches to solving spin dynamics problems results from the fact that no matrix representation for spin operators is used in SD-CAS, which determines a *full symbolic character* to the performed computations. Spin correlations are stored in SD-CAS as four-entry nested lists of which size increases linearly with the number of spins into the system and are easily mapped into analytical expressions in terms of spin operator products. For the so defined SD-CAS spin correlations a set of specialized functions and procedures is introduced that are essential for implementing basic spin algebra operations, such as the spin operator products, commutators, and scalar products. They provide results in an abstract algebraic form: specific procedures to quantitatively evaluate such symbolic expressions with respect to the involved spin interaction parameters and experimental conditions are also discussed. Although the main focus in the present work is on laying the foundation for spin dynamics symbolic computation in NMR based on a non-matrix formalism, practical aspects are also considered throughout the theoretical development process. In particular, specific SD-CAS routines have been implemented using the YACAS computer algebra package (http://yacas.sourceforge.net), and their functionality was demonstrated on a few illustrative examples.

1. Introduction

The spectacular evolution of NMR spectroscopy from its early experimental demonstrations to the modern applications in a wide variety of fields, such as molecular biology [1] or quantum information processing [2], has largely relied on the development of theoretical tools capable to solve dynamical problems of increasing complexity with respect to both, size of the relevant coupled spin system, and the applied experimental methodology. Historically, analytical treatments were the first introduced to account for a wealth of NMR phenomena and processes as well as for accompanying spectral features. With respect to the employed mathematical apparatus, and the nature of the problem being addressed, a broad range of analytical models have been developed: from classical theories describing the motion of nuclear magnetization under specific magnetic fields and relaxation, to full quantum mechanical treatments of the evolution under generally time-dependent anisotropic spin-spin interactions, and their coherent manipulation by sophisticated multiple pulse techniques. Depending on the particular features looked for being emphasized, these quantum treatments may also involve a number of distinct analytical formalisms, which differ among themselves through the considered evolution (sub)space, basis set, representation of the spin operators, and the employed approximation method. All these theoretical tools are well documented in many NMR textbooks [3–12].

A particularly high level of difficulty is encountered in solid-state applications, where the complexity of the associated spin interaction patterns often restricts the analytical calculations to small systems and/or low orders of approximation. Although the resulting simplified models do in many cases reproduce basic features of the underlying spin dynamics fairly well, thus offering important physical insights into the investigated phenomena, they cannot also provide detailed quantitative information, or a correct characterization of perturbing multi-spin effects. As such, more and more theoretical efforts have been oriented towards the development of alternative numerical approaches. Starting with small routines designed for very specific purposes [13–15], this field has witnessed a rapid evolution up to the present days when fully featured programs [16,17], or dedicated numerical libraries [18,19], are available for the exact simulation of virtually any kind of NMR experiment on systems that can be considered sufficiently large (12+ spins) [17] as to capture most of the relevant physical properties. The impressive progress in the field has been made possible through the development of many innovative concepts addressing both, fundamental [20–23] and practical aspects

* Corresponding author. Fax: +40 264 420042.
E-mail address: cfilip@itim-cj.ro (C. Filip).

1090-7807/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmr.2010.08.014

[24,25]: their major effect was the increase of the computational speed and efficiency to such an extent that numerical simulations are now considered almost indispensable in any work aimed at further refining the NMR methodology.

Computational speed is definitely one of the biggest advantages of numerical simulations in comparison with analytical treatments. The possibility to at least partially fill this gap, and bring the two distinct methods on more equal grounds, is studied in the present work by introducing symbolic computations as a viable alternative to the lengthy procedures commonly employed for analytical calculations. In particular, a symbolic computational tool called Spin Dynamics by Computer Algebra System (SD-CAS) is developed and illustrated on ensembles of homonuclear 1/2-spins by implementing a set of appropriate spin algebra operations. Compared with previous computer algebra applications to NMR [26–34], SD-CAS has an important distinctive feature, namely, the computations are performed at the level of abstract spin operators, which means that no matrix representation is used to perform the required calculations (spin operator products, commutators, and scalar products). As such, the SD-CAS output consists of general symbolic expressions that are valid to arbitrary systems: for particular applications, these expressions have to be simply evaluated by specifying the number of spins and the corresponding interaction parameters. The results reported here could contribute to renewing the interest in the analytical investigation of NMR spin dynamics, which, despite the approximate character of the derived models, may still offer useful information, e.g., for the physical interpretation of the results obtained by numerical simulations.

2. Liouville space spin dynamics

This section gives a brief overview of the basic concepts underlying most of the computer implementations of nuclear spin dynamics: the emphasis will be on the distinctive features of symbolic computations in comparison with the existing NMR numerical simulation programs. For simplicity, the discussion is confined to pure quantum evolution in ensembles of homonuclear 1/2-spins under high magnetic field truncated anisotropic spin interactions. Extension to higher spin quantum numbers, $I > 1/2$, heteronuclear systems, and presence of relaxation, is also possible but, however, this is beyond the purpose of the present work.

Generally, any spin dynamics computation is centered around the evaluation of the so-called time-propagator, $U(t_2,t_1)$,

$$U(t_2,t_1) = Te^{-i\int_{t_1}^{t_2} H(t')dt'} \tag{1}$$

because this provides the density operator ρ at any time t_2, from its known expression at an earlier time, t_1, as it results by solving the Liouville-von Neumann equation,

$$\frac{d}{dt}\rho(t) = -i\widehat{H}(t)\rho(t) = -i[H(t),\rho(t)] \tag{2}$$

that is,

$$\rho(t_2) = \widehat{U}(t_2,t_1)\rho(t_1) = U(t_2,t_1)\rho(t_1)U^{\dagger}(t_2,t_1) \tag{3}$$

Here, a Liouville-space description is employed where superoperators are denoted by the same symbols as their equivalent Hilbert space operators, but with an appended caret. As there are multiple choices in defining the way a superoperator acts upon a Liouville space vector (an operator), this must be clearly specified for each particular case. For example, the Liouville-space quantum evolution described by Eqs. (2) and (3) involves the action of the commutator superoperator, and the exponential superoperator, respectively.

An analytical evaluation in a closed form of the time-propagator (1) is possible only in two special situations, namely, for an inhomogeneous Hamiltonian [37], when a direct integration of the exponent is allowed because the Hamiltonian at different moments of time commutes with itself,

$$U(t_2,t_1) = e^{-i\int_{t_1}^{t_2} H(t')dt'} \tag{4}$$

and for a time-independent Hamiltonian, when Eq. (4) further simplifies to

$$U(t_2,t_1) = e^{-iH(t_2-t_1)} \tag{5}$$

All the other situations, correspond to homogeneous time-dependent Hamiltonians: in such cases the Dyson time-ordering operator T specifies that a proper sequencing of time has to be considered within any approximation aimed at evaluating the underlying time-propagator (1). For example, in a power expansion approach

$$\widehat{U}(t_2,t_1) \sim 1 - i\int_{t_1}^{t_2}\widehat{H}(\tau_1)d\tau_1 + (-i)^2\int_{t_1}^{t_2}\widehat{H}(\tau_2)d\tau_2 \times \int_{t_1}^{\tau_2}\widehat{H}(\tau_1)d\tau_1 + \cdots \tag{6}$$

the time ordering is equivalent with imposing the condition $t_2 > \tau_n > \cdots > \tau_2 > \tau_1 > t_1$ for the intermediate integration times τ_j in the expression of the general nth order term,

$$\widehat{U}^{(n)}(t_2,t_1) = (-i)^n\int_{t_1}^{t_2}\widehat{H}(\tau_n)d\tau_n \times \int_{t_1}^{\tau_n}\widehat{H}(\tau_{n-1})d\tau_{n-1}\cdots\int_{t_1}^{\tau_2}\widehat{H}(\tau_1)d\tau_1 \tag{7}$$

whereas in an approximation based on dividing the finite evolution period t_2–t_1 in a large number n of infinitely small time-steps, τ, the requirement is that, the individual static-like propagators in the product defining $U(t_2,t_1)$

$$U(t_2,t_1) \sim \underbrace{e^{-iH_n\tau}\cdots e^{-iH_j\tau}\cdots e^{-iH_2\tau}e^{-iH_1\tau}}_{n} \tag{8}$$

are ordered from right to left with increasing the point in time $j\tau$ when the corresponding Hamiltonian $H_j = H(j\tau)$ is evaluated.

In a matrix representation of the involved operators, the evaluation of Eq. (8) can be conveniently handled by numerical methods and therefore constitutes the core task of any NMR simulation program. For a detailed methodological description of the modern numerical approaches used to date in NMR, e.g., for computing the expectation value of an NMR observable M,

$$\langle M(t_2)\rangle = \langle M|\rho(t_2)\rangle = Tr\{M^{\dagger}\widehat{U}(t_2,t_1)\rho(t_1)\} \tag{9}$$

based on approximation (8), the reader is referred to the specialized literature [20,24]. However, despite the substantial progress in this field, numerical simulations are still limited to moderately sized spin systems, especially due to the exponential increase of the required memory and CPU time with the number of spins.

Unlike matrix calculations, most of the analytical treatments of spin dynamics rely on an abstract operator description, and therefore are not suitable for numerical implementation. Alternatively, spin operators algebra can be implemented within the so-called Computer Algebra Systems (CAS). Here, we illustrate this concept with the particular example of the quantum evolution in a Liouville space defined in terms of products of Cartesian spin operators [35, 36, and references therein]. For an ensemble of N 1/2-spins, this space is spanned by the 2^{2N} base vectors

$$Q_j \in \frac{1}{\langle Q_j|Q_j\rangle}\left\{1, \underbrace{I_{1\alpha},\ldots,I_{N\alpha}}_{3C_N^1}, \underbrace{I_{1\alpha}I_{2\beta},\ldots,I_{1\alpha}I_{N\beta},\ldots,I_{N-1,\alpha}I_{N\beta}}_{3^2C_N^2},\ldots, \underbrace{I_{1\alpha}I_{2\beta}\cdots I_{N\gamma}}_{3^NC_N^N}\right\} \tag{10}$$

which are constructed by taking the all possible products of spin operators: each α (β,γ) label above stands for x, y and z, C_N^k

represents the specified binomial coefficient, and $\langle Q_j|Q_j\rangle$ is the normalization factor. In the NMR literature, such a product of n spin operators is often referred to as the n-spin correlation. Using this concept has many advantages: for instance, it enables to associate correlation peaks observed in multi-dimensional NMR spectra with the occurrence of a corresponding spin-correlation term in the expression of the density operator, or to monitor the spread of an initial spin polarization state into multi-spin correlations under different (effective) spin-spin interactions.

Within this framework, density operators (or any other dynamical variable) and spin interaction Hamiltonians should be expressed in terms of the base vectors (10). For instance, considering the typical high-field truncated NMR spin interactions (e.g., chemical shielding, Zeeman type interactions, dipolar, and indirect spin-spin interactions), their general expressions

$$H = \sum_{i_1=1}^{N} C(i_1) I_{i_1\alpha}; \quad H = \sum_{i_1,i_2=1}^{N} {}'D(i_1,i_2) f(I_{i_1\alpha}, I_{i_2\beta}) \tag{11}$$

contain single- or two-spin coupling coefficients, C (D), and characteristic functions f on Cartesian spin operators. If the initial density operator is also put in the form $\rho(t_1) = \sum_{j\leqslant K_1} a_j(t_1) Q_j$ the evaluation of Eq. (3) directly provides the decomposition of the density operator $\rho(t_2)$ in terms of the spin operator products Q_j, i.e.,

$$\rho(t_2) = \sum_{j\leqslant K_2} a_j(t_1,t_2) Q_j; \tag{12}$$

where K_1 and K_2 represent the dimension of the corresponding Liouville subspaces (typically much smaller than 2^{2N}) within which these density operators evolve.

In its symbolic form, the relationship (12) provides qualitative information with respect to the spin correlations Q_j generated from an initial state during its evolution, and also, the functional dependencies of the weighting coefficients a_j on spin interaction parameters. There is a large variety of methods and formalisms that have been introduced over the time for deriving such relationships, each of which being adapted to solve a particular type of spin dynamics problem. For instance, approximations based on the power expansion of the time-propagator like in Eq. (6) are often employed when dealing with large systems of dipolar coupled spins, whereas for simple systems and time-independent Hamiltonians exact solutions can be determined in principle by solving the system of linear differential equations for the underlying dynamical variables $Q_j(t_2) = \widehat{U}(t_2,t_1) Q_j$,

$$\frac{d}{dt} Q_j(t) = -i\widehat{H} Q_j(t) = \sum_{k\leqslant K_2} a_k Q_k(t) \tag{13}$$

In between these two extreme cases there are numerous examples of methods that exploit characteristic features of the investigated dynamical system, for instance the evolution under periodically time-dependent Hamiltonians (continuous or discrete), which is commonly encountered in solid-state NMR spectroscopy. However, no matter the particular method, all of them will make extensive use of commutators between arbitrarily complex spin-correlation terms. Hence, this can be regarded as the most basic operation upon spin variables. A convenient approach for fast evaluation of such commutators, which can be easily translated into algorithms suitable for CAS implementation, is based upon the following properties of the products between two spin-1/2 operators:

$$\begin{aligned} &I_{j\alpha} I_{k\beta} = I_{k\beta} I_{j\alpha}; \quad j\neq k \\ &I_{j\alpha} I_{j\alpha} = \frac{1}{4}; \quad \alpha \equiv x,y,z \\ &I_{j\alpha} I_{j\beta} = \frac{i}{2} \varepsilon_{\alpha\beta\gamma} I_{j\gamma}; \qquad \alpha(\beta,\gamma) \equiv x,y,z \quad \text{and} \quad \alpha\neq\beta\neq\gamma \end{aligned} \tag{14}$$

where $\varepsilon_{\alpha\beta\gamma}$ denotes the Levi–Civita symbol.

Once an approximate (or an exact) expression for the density operator (12) is found, the quantitative evaluation of different observables can be obtained from Eq. (9) by projecting $\rho(t_2)$ onto the observable operator M. The required operations are also easy to implement within specific computer algebra procedures. In particular, the result of the *Trace* operation in (9) can be directly found from a decomposition of the product $M^{\dagger}\rho(t_2)$ in terms of the orthonormal base operators Q_j, because $Tr\{Q_j\} = 0$ except for the case $Tr\{1\} = 2^N$. In the final step, the specified coupling and geometric parameters need to be explicitly replaced within the analytical expressions of the weighting coefficients a_j.

3. SD-CAS implementation of spin dynamics

Following the general description in the preceding section, the CAS implementation of spin dynamics, hereafter referred to as the SD-CAS, will require in principle the development of two distinct types of routines: the former are designed to implement Liouville space operations at the symbolic level, and therefore should be constructed such that a general applicability is guaranteed no matter the particular experimental conditions (e.g., static or rotating samples), whereas the latter are needed to construct particular CAS implementations of various NMR experiments, or to solve specific spin dynamics problems related to such experiments, including the quantitative evaluation of the resulting symbolic expressions. The first type of routines can be thus regarded as belonging to a core engine designed to perform the basic spin-space operations, which are independent of particular experimental details, and to provide in this way the components that commonly enter any symbolic computation code. From this perspective, the construction of such routines constitutes the major objective of the present work (see Section 3.1 below), but however their assembling within a particular application is also discussed in Section 3.2.

3.1. Symbolic computations in the Liouville space

3.1.1. SD-CAS representation of spin correlations

The fundamental difference between pure computer algebra approaches to solve spin dynamics problems and numerical methods is considered here as coming from the fact that no matrix representation of spin operators is employed by the former. In particular, for a general spin-correlation term,

$$\begin{aligned} S = c(i_1,i_2,\ldots,i_p) \times I_{i_1,x} I_{i_2,x} \cdots I_{i_{nx},x} \times I_{i_{nx+1},y} I_{i_{nx+2},y} \cdots I_{i_{nx+ny},y} \\ \times I_{i_{nx+ny+1},z} I_{i_{nx+ny+2},z} \cdots I_{i_{nx+ny+nz},z} \end{aligned} \tag{15}$$

in SD-CAS we introduce a four-entries nested list representation given by

$$\begin{aligned} &\mathtt{S} = \{\mathtt{c(i[1],i[2],\ldots,i[p])},\{\mathtt{i[1],i[2],\ldots,i[nx]}\}, \\ &\quad \{\mathtt{i[nx+1],i[nx+2],\ldots,i[nx+ny]}\}, \\ &\quad \{\mathtt{i[nx+ny+1],i[nx+ny+2],\ldots,i[nx+ny+nz]}\}\} \end{aligned} \tag{16}$$

where the first element is a complex coefficient c, which generally may depend on coupling parameters involving p nuclear spins. The next three elements are sub-lists, each of which containing the spin labels `i[k]` that correspond to the nx (ny, and nz) spin operators I_x, (I_y, and I_z) in the product (15). Like matrices in the case of numerical simulations, lists of the form S, hereafter called *first order SD-CAS lists*, constitute the objects upon which basic SD-CAS operations, e.g., commutators, are applied. They are assigned with the following properties:

i. The size of `S` increases linearly with the number $m = nx+ny+nz$ of the spin operators, and also depends on the number p of the spins ($p \geqslant m$), within the spin-correlation term (15).
ii. The labels `i[1]`, ..., i[nx+ny+nz] are all different from each other: hence, spin correlations of the type `S` are equivalent, up to a weighting coefficient, with operators within the basis set (10), and therefore we call them to be in their *irreducible* form.
iii. Depending on the context, the spin labels `i[k]` can be considered either fixed to some particular values, or as running from 1 up to the maximum number of spins into the system, N. In the first case, the numerical index `k` of an `i` label stands for the corresponding individual k spin within the correlation term, and thus `i[k]` in Eqs. (15) and (16) are not required to be arranged in an ascending order with respect to these indices. In the second case (which is the case illustrated in (15) and (16)), such an ordering is mandatory: here, the relationship (16) provides the general representation for a number $N!/m!$ of distinct individual m-spin correlations, which clearly illustrates the advantages of performing calculations at symbolic level.

In practice, spin correlations do not occur as standing alone terms but rather as sums of different S_j. Examples are the base vectors decomposition of the density operator shown in Eq. (12), or two-spin interaction Hamiltonians, like the dipolar interaction and indirect spin-spin interaction, generically given through

$$H2 = D1(i_1,i_2)I_{i_1,x}I_{i_2,x} + D2(i_1,i_2)I_{i_1,y}I_{i_2,y} + D3(i_1,i_2)I_{i_1,z}I_{i_2,z} \tag{17}$$

All such sums of spin correlations are implemented in SD-CAS as *second order lists* of which individual components S_j are first order SD-CAS lists, that is $\{S_1, S_2, \ldots\}$. For the two-spin interaction Hamiltonian (17), this reads

$$\begin{aligned} \mathtt{H2} = \{&\{\mathtt{D1(i[1],i[2])},\{\mathtt{i[1]},\mathtt{i[2]}\},\{\},\{\}\},\{\mathtt{D2(i[1],i[2])},\{\},\\ &\{\mathtt{i[1]},\mathtt{i[2]}\},\{\}\},\{\mathtt{D3(i[1],i[2])},\{\},\{\},\{\mathtt{i[1]},\mathtt{i[2]}\}\}\} \end{aligned} \tag{18}$$

To conclude, the most general (and complex) object introduced in SD-CAS is a second order list, which provides a convenient representation for a sum of spin correlations. For practical illustration, the spin correlations discussed above have been implemented within the computer algebra package YACAS [38], together with a set of functions and procedures that specify the structure of the associated lists, and also a few representative properties (see the Appendix A). YACAS is an easy to use, general purpose Computer Algebra System, which uses its own programming language designed for symbolic as well as arbitrary-precision numerical computations. The main reason for choosing YACAS to implement the SD-CAS functions and routines is represented by its powerful scripting language, closely related to LISP, which proved enhanced flexibility in performing list operations. However, other CAS's, including commercial programs, may be considered equally well for alternative implementations of SD-CAS. The most important SD-CAS functions in Appendix A are:

`IsSpinCorr(X)` checks if the input list `X` has the structure and defining properties required to represent a spin-correlation term, Eq. (15). It returns *True* if the argument has four components `X[j]`, with `j = 1...4`, fulfilling the condition `X[2]` ∩ X[3] ∩ X[4] = {}, and *False* otherwise.

`IsSumSpinCorr(X)` checks if the input list `X` has the structure required to represent a sum of spin correlations. It returns *True* if, for each of its elements `X[j]`, the function `IsSpinCorr(X[j])` holds *True*, and returns *False* otherwise.

`SymbOrder(X)` does a symbolic reordering of the i_k within a spin-correlation term if they are running labels: as shown below, this operation is required when working with spin correlation sums. For example, suppose that during some calculations the following terms have been obtained in the expression of the density operator:

$$\rho = \cdots + C(i_1,i_2)I_{i_1,x}I_{i_2,y} + D(i_3,i_5,i_4)I_{i_5,x}I_{i_3,y} + \cdots \tag{19}$$

If the involved spin labels are running labels, the two terms in (19) are equivalent, but they will not be treated accordingly by symbolic computation programs. That is, within the associated list

$$\begin{aligned} \mathtt{rho} = \{&\cdots\{\mathtt{C(i[1],i[2])},\{\mathtt{i[1]}\},\{\mathtt{i[2]}\},\{\}\},\{\mathtt{D(i[3],i[5],i[4])},\\ &\{\mathtt{i[5]}\},\{\mathtt{i[3]}\},\{\}\},\cdots\} \end{aligned} \tag{20}$$

they cannot be added together in a proper way by standard CAS routines unless the labels `i[5]` and `i[3]` in the second sub-list are replaced by `i[1]` and `i[2]`. Also, the `i[4]` label has to be also replaced by `i[3]` to account for the fact that such a term originates from the mutual interactions within three-spin subsystems. Applying now the function `SymbOrder` to `rho` in (20), the density operator transforms to

$$\begin{aligned} \mathtt{rho} = \{&\cdots\{\mathtt{C(i[1],i[2])},\{\mathtt{i[1]}\},\{\mathtt{i[2]}\},\{\}\},\{\mathtt{D(i[2],i[1],i[3])},\\ &\{\mathtt{i[1]}\},\{\mathtt{i[2]}\},\{\}\},\cdots\} \end{aligned}$$

The way this reordering task is performed through the function `SymbOrder(X)` is briefly explained in the following by considering the most general case of a spin-correlation term in Eq. (15), where $p > m$, but the labels are not necessarily put in an ascending order. First, an integer pointer is introduced and incremented from 1 up to m such that the label `i[k]` found at the actual pointer position n is replaced by `j[n]` within both, the spin part, and the complex coefficient c of the list. However, when $p > m$ the expression of the coefficient c will still contain old labels that are left unchanged by this transformation: thus, in a second stage, such labels are all identified and replaced by `j[m+1]`, ..., `j[p]`. Finally, the new label identifier `j` is transformed back to `i`, so that in the end the original list is brought to its ordered form as in Eq. (16).

`ReduceSumSpCorr(X)` identifies the spin-correlation terms within a sum that are equal with each other, and then adds them together. In the case of running spin labels this function will be applied only after the components have been symbolically ordered as described above. For this purpose, a standard procedure of comparing the elements of a list is employed: whenever two terms in the list are found to be identical with respect to the incorporated spin operators, they are replaced by a single term of which new coefficient is set as the sum of their individual coefficients. Considering the same example as above, one has:

$$\begin{aligned} \mathtt{ReduceSumSpCorr(rho)} = \{&\cdots\{\mathtt{C(i[1],i[2])} + \mathtt{D(i[2],i[1],i[3])},\\ &\{\mathtt{i[1]}\},\{\mathtt{i[2]}\},\{\}\},\cdots\} \end{aligned}$$

The rest of SD-CAS functions in Appendix A have been introduced for enhanced flexibility in working with lists: as will be seen in Section 3.1.3, many of them constitute useful building blocks required for the implementation of more complex operations, like the product and commutator between general spin-correlation terms. Other functions can be constructed in a similar way, depending on the particular needs in a given symbolic computation.

3.1.2. Spin correlation products and commutators

Elementary operations, such as products of spin correlations and commutators are implemented at the level of first order SD-CAS lists (16), where the spin labels are considered fixed labels. In practice, starting from two input lists `X` = {`X[1]`, X[2], X[3], X[4]} and `Y` = {`Y[1]`, Y[2], Y[3], Y[4]} that represent the spin-correlation terms, X and Y, the problem is to find the list `Z` = {`Z[1]`, Z[2], Z[3], Z[4]} that corresponds to the product $Z = X \cdot Y$. Once this is

calculated, the commutator [X,Y] can be also easily evaluated. The explicit form of the incorporated list elements with respect to the coupling parameters and spin labels is not relevant for discussion below, so that they are generically given through `X(Y,Z)[i]`, with `i = 1,...,4`. The algorithm developed for determining the elements `Z[i]` relies on a systematic use of the relationships (14), and contains the following major steps:

i. First, we introduce the notations: (a) `XiYj = X[i] ∩ Y[j]` with `i(j) = 2,3,4` are lists of spin labels corresponding to the spin operators that are common to *X* and *Y*; (b) d_{ij} is the dimension of the list `XiYj` – this represents the number of elements (spin labels) within the list and can be retrieved by applying a function called `Length(XiYj)`; (c) $l = d_{23} - d_{24} - d_{32} + d_{34} + d_{42} - d_{43}$ and $q = 2(d_{22} + d_{33} + d_{44}) + d_{23} + d_{24} + d_{32} + d_{34} + d_{42} + d_{43}$.
ii. To determine the I_x operators within the spin correlations product *Z*, there are three distinct contributions that have to be taken into account: the I_x operators originally present in the both, *X* and *Y* term, the I_x operators eliminated whenever a products of the type $I_x \cdot I_x$ is encountered, and the additional I_x operators that occur from products of the type $I_y \cdot I_z$, respectively. In the corresponding list representation, this reads: `Z[2]=X[2] ∩ Y[2]-X2Y2+X3Y4+X4Y3`. Similar arguments apply equally well in the case of the I_y and I_z operators, for which one gets `Z[3]=X[3] ∩ Y[3]-X3Y3+X2Y4+X4Y2` and `Z[4]=X[4] ∩ Y[4]-X4Y4+X2Y3+X3Y2`.
iii. The coefficient `Z[1]` is obtained as the product of the coefficients in the original *X* and *Y* spin correlations with an additional factor which results from performing the spin products in Eq. (14), namely: `Z[1]=X[1]Y[1]·i^l·2^-q`.

The SD-CAS implementation of the above algorithm has been done through a function called `SpProd(X,Y)` – see the Appendix B. This function is generally applicable, staring from the simplest case of spin operator products that are easy to evaluate by hand, e.g., $I_{i_1,x} I_{i_1,y} = (i/2) I_{i_1,z}$ up to calculating products between arbitrarily complex spin-correlation terms, such as $X = I_{i_1,x} I_{i_3,y} I_{i_6,x} I_{i_4,z} I_{i_7,y} I_{i_9,x} I_{i_2,z}$ and $Y = I_{i_1,y} I_{i_5,y} I_{i_6,x} I_{i_4,x} I_{i_7,y} I_{i_3,z} I_{i_2,x} I_{i_8,x} I_{i_9,y} I_{i_{10},x}$, where the advantage of using symbolic computations is obvious. For practical illustration, the SD-CAS results obtained for these particular examples are given below:

Example 1

```
X={1, {i[1]}, {}, {}};
Y={1, {}, {i[1]}, {}};
Z=SpProd(X,Y)
Z={Complex(0,1/2), {}, {}, {i[1]}};
```

Example 2

```
X={1, {i[1],i[6],i[9]}, {i[3],i[7]}, {i[4],i[2]}};
Y={1,{i[6],i[4],i[2],i[8],i[10]},  {i[1],i[5],i[7],
   i[9]}, {i[3]}};
Z=SpProd(X,Y)
Z={Complex(0,1/512),{i[8],i[10],i[3]},  {i[5],i[4],
   i[2]}, {i[1],i[9]}};
```

Rewritten in terms of spin operators, the result obtained in the second example reads $Z = X \cdot Y = (i/512) I_{i_8,x} I_{i_{10},x} I_{i_3,x} I_{i_5,y} I_{i_4,y} I_{i_2,y} I_{i_1,z} I_{i_9,z}$. Further, it is to be noted that the function `SpProd(X,Y)` provides the results in the form of first order SD-CAS lists. To account for the fact that in practical applications the results are generally obtained at the level of spin correlation sums, a new function called `SumSpProd(X,Y)` is introduced, of which arguments are second order SD-CAS lists. This function is shown in Appendix B, and the way it operates can be briefly described as follows: given the input lists `X={X1, X2, ...}` and `Y={Y1, Y2, ...}`, the result of `SumSpProd(X,Y)` is calculated as {`SpProd(X1,Y1), SpProd(-X1,Y2), ...`}.

According to the Liouville–von Neumann Eq. (2), the symbolic analysis of spin dynamics essentially relies on the ability to calculate commutators between spin-correlation terms: rigorously, if the definition relationship $[X,Y]=X \cdot Y - Y \cdot X$ is employed, such a task would require to perform two spin correlation products followed by their subtraction. Computationally, however, this is not very efficient: instead, in SD-CAS we introduce an algorithm by which the result of the commutator [X,Y] is simply set to either *0* or $2X \cdot Y$, depending on whether the parameter *l* evaluated for the product $X \cdot Y$ is an odd, or an even number, respectively. As such, the function `SpComm(X,Y)` is basically equivalent with the function `SpProd(X,Y)`, except for containing the additional check point with respect to the value of *l*. Similar to the case of spin products, a function called `SumSpComm(X,Y)` is introduced (see the Appendix B), which retrieves the commutator between two spin correlation sums. To illustrate the SD-CAS symbolic computation of commutators we consider both, the previous examples:

Example 1

```
X={1, {i[1]}, {}, {}};
Y={1, {}, {i[1]}, {}};
Z=SpComm(X,Y)
Z={Complex(0,1), {}, {}, {i[1]}};
```

Example 2

```
X={1, {i[1],i[6],i[9]}, {i[3],i[7]}, {i[4],i[2]}};
Y={1,  {i[6],i[4],i[2],i[8],i[10]},  {i[1],i[5],i[7],
   i[9]}, {i[3]}};
Z=SpComm(X,Y)
Z={Complex(0,1/256),{i[8],i[10],i[3]},{i[5],i[4],i[2]},
   {i[1],i[9]}};
```

as well as another two specific examples, the first showing the case of a zero commutator, e.g., between

$$X = I_{i_1,x} I_{i_3,y} I_{i_6,x} I_{i_4,z} I_{i_7,y} I_{i_9,x} I_{i_2,z} I_{i_5,z} \quad \text{and} \quad Y = I_{i_1,y} I_{i_5,y} I_{i_6,x} I_{i_4,x} I_{i_7,y} I_{i_3,z} I_{i_2,x} I_{i_8,x} I_{i_9,y} I_{i_{10},x},$$

Example 3

```
X={1, {i[1],i[6],i[9]}, {i[3],i[7]}, {i[4],i[2],i[5]}};
Y={1,{i[6],i[4],i[2],i[8],i[10]},{i[1],i[5],i[7] ,i[9]},
   {i[3]}};
Z=SpComm(X,Y)
Z={};
```

whereas the second illustrating the calculation of the commutator between two spin correlation sums,

Example 4

```
X={{1, {i[1],i[3]}, {}, {i[2]}}, {1, {}, {i[2]}, {}}};
Y={{1, {}, {i[1]}, {i[2],i[3]}}, {1, {i[1]}, {}, {}},
{1, {i[2]}, {i[1]}, {}}, {1, {}, {}, {i[1]}}};
```

```
Z=SumSpComm(X,Y)
Z={{Complex(0,-1), {i[3]}, {i[1]}, {i[2]}},
    {Complex(0,1), {i[2]}, {i[1]}, {i[3]}},
    {Complex(0,-1), {}, {i[1]}, {i[2]}}};
```

Analytically, the X and Y terms in Example 4 can be written as

$$X = I_{i_1,x} I_{i_3,x} I_{i_2,z} + I_{i_2,y};$$
$$Y = I_{i_1,y} I_{i_2,z} I_{i_3,z} + I_{i_1,x} + I_{i_2,x} I_{i_1,y} + I_{i_1,z};$$

for which it can be easily verified that

$$Z = [X,Y] = -iI_{i_3,x} I_{i_1,y} I_{i_2,z} + iI_{i_2,x} I_{i_1,y} I_{i_3,z} - iI_{i_1,y} I_{i_2,z};$$

3.1.3. Products and commutators between general spin correlations

The description has so far referred specifically to products and commutators between well defined spin-correlation terms, corresponding to fixed spin labels. To preserve the full symbolic character of the SD-CAS computations, however, these elementary operations have to be incorporated next within more general schemes applicable when spin labels are all running labels. On the one hand, the theoretical construction of such a scheme is relatively straightforward: it consists of decomposing a general spin operator product into a large number (see below) of elementary products between spin correlations having fixed spin labels, which can be thus evaluated by employing the `SpProd` or `SpComm` functions developed previously, and then the result is put in the form of a spin correlations sum. On the other hand, the SD-CAS implementation represents a formidable task, because:

i. The number of elementary products required to multiply two general spin-correlation terms A and B of the form (15) depends on the number of their spin operators, m_a and m_b, according to the formula $\chi = \sum_{k=0}^{\min(m_a,m_b)} P_k C_{m_a}^{m_a-k} C_{m_b}^{k}$, with P_k, the permutations of k: this parameter corresponds to the all distinct situations that may occur with respect to the number of common spin operators, from 0 to min($-m_a, m_b$), between the A and B spin-correlation terms.
ii. At each of this χ computational step the spin product operation must be followed by a symbolic ordering of the spin labels (using the function `SymbOrder`) to account for the fact that they are running labels.
iii. Finally, the calculated χ spin-correlation terms should be added together (using the function `ReduceSumSpCorr`) since, as will be discussed below, many of them are equivalent with respect to their spin part.

The SD-CAS functions implementing the corresponding operations have been named `GenSpProd`, `GenSpComm`, `GenSumSpProd`, and `GenSumSpComm`: they are also explicitly shown in the Appendix B. By analogy with the density matrix diagonalization in the case of numerical simulations, the spin operator product and commutator, in their most general form presented above, play a central role for any symbolic computation of spin dynamics. This is illustrated in the following based on the relatively simple example of evaluating the commutator between the spin correlations terms A and B:

$$A = a(i_1, i_2, i_3) I_{i_1,x} I_{i_2,y} I_{i_3,y}$$
$$B = b(i_1, i_2, i_3, i_4) I_{i_1,x} I_{i_2,x} I_{i_3,y} I_{i_4,z}$$

Assuming that each of the i_k label above corresponds to the fixed spin number k, the commutator $C = [A,B]$ can be immediately evaluated either by conventional analytical calculations (by hand), or by running the SD-CAS function `SpComm(A,B)`. If the A and B are considered instead general spin correlations, i.e., each of the i_k label is running from 1 up to the maximum number of spins into the system, N, the evaluation of their commutator requires $\chi = 49$ basic computational steps, as described at the points (i)–(iii). For instance, referring specifically to the i_1 label in the Acorrelation, this is definitely different from the i_2 and the i_3 labels within the same term, but it may become equal with each of the i_k ($k = 1,\ldots,4$) labels within B, thus giving rise to four distinct situations that have to be treated separately when computing the commutator $[A,B]$. Similar arguments apply also to the i_2 and the i_3 labels, then to the pairs (i_1, i_2), (i_1, i_3), (i_2, i_3), and finally to the triplet (i_1, i_2, i_3), which leads in the end to the above mentioned number of distinct elementary commuators that have to be computed (counting also the situation when there are no common labels between A and B).

The result of running `GenSpComm(A,B)` function, shown in Eq. (21),

$$\begin{aligned}
C = {} & i\begin{bmatrix} a(i_6,i_3,i_4)b(i_1,i_2,i_6,i_5) - a(i_2,i_6,i_4)b(i_6,i_1,i_3,i_5) \\ -a(i_2,i_6,i_4)b(i_1,i_6,i_3,i_5) - a(i_2,i_4,i_6)b(i_6,i_1,i_3,i_5) \\ -a(i_2,i_4,i_6)b(i_1,i_6,i_3,i_5) \end{bmatrix} I_{i_1,x} I_{i_2,x} I_{i_3,y} I_{i_4,y} I_{i_5,z} I_{i_6,z} \\
& - i[a(i_6,i_4,i_5)b(i_1,i_2,i_3,i_6)] I_{i_1,x} I_{i_2,x} I_{i_3,z} I_{i_4,z} I_{i_5,z} I_{i_6,z} \\
& + i[a(i_3,i_4,i_6)b(i_1,i_2,i_5,i_4) + a(i_3,i_6,i_4)b(i_1,i_2,i_5,i_4)] I_{i_1,x} I_{i_2,x} I_{i_3,x} I_{i_4,x} I_{i_5,y} I_{i_6,y} \\
& - \frac{i}{4}\begin{bmatrix} a(i_5,i_4,i_2)b(i_5,i_4,i_1,i_3) + a(i_5,i_2,i_4)b(i_5,i_4,i_1,i_3) \\ +a(i_5,i_4,i_2)b(i_4,i_5,i_1,i_3) + a(i_5,i_2,i_4)b(i_4,i_5,i_1,i_3) \\ -a(i_2,i_3,i_4)b(i_3,i_4,i_1,i_2) - a(i_2,i_3,i_4)b(i_4,i_3,i_1,i_2) \end{bmatrix} I_{i_1,y} I_{i_2,y} I_{i_3,z} I_{i_4,z} \\
& + \frac{i}{4}\begin{bmatrix} a(i_5,i_2,i_4)b(i_5,i_1,i_3,i_2) + a(i_5,i_4,i_2)b(i_5,i_1,i_3,i_2) \\ +a(i_5,i_2,i_4)b(i_1,i_5,i_3,i_2) + a(i_5,i_4,i_2)b(i_1,i_5,i_3,i_2) \\ -a(i_4,i_5,i_3)b(i_1,i_2,i_5,i_4) - a(i_4,i_3,i_5)b(i_1,i_2,i_5,i_4) \end{bmatrix} I_{i_1,x} I_{i_2,x} I_{i_3,y} I_{i_4,y} \\
& - \frac{i}{4}\begin{bmatrix} a(i_2,i_4,i_5)b(i_4,i_1,i_5,i_3) + a(i_2,i_4,i_5)b(i_1,i_4,i_5,i_3) \\ +a(i_2,i_5,i_4)b(i_4,i_1,i_5,i_3) + a(i_2,i_5,i_4)b(i_1,i_4,i_5,i_3) \\ -a(i_3,i_4,i_2)b(i_4,i_1,i_3,i_2) - a(i_3,i_4,i_2)b(i_1,i_4,i_3,i_2) \\ -a(i_3,i_2,i_4)b(i_4,i_1,i_3,i_2) - a(i_3,i_2,i_4)b(i_1,i_4,i_3,i_2) \end{bmatrix} I_{i_1,x} I_{i_2,x} I_{i_3,z} I_{i_4,z} \\
& + \frac{i}{4}[a(i_3,i_5,i_4)b(i_1,i_2,i_5,i_4) + a(i_3,i_4,i_5)b(i_1,i_2,i_5,i_4)] I_{i_1,x} I_{i_2,x} I_{i_3,x} I_{i_4,x} \\
& - \frac{i}{4}[a(i_2,i_3,i_4)b(i_3,i_4,i_1,i_2) + a(i_2,i_3,i_4)b(i_4,i_3,i_2,i_1)] I_{i_1,z} I_{i_2,z} I_{i_3,z} I_{i_4,z} \\
& - \frac{i}{16}\begin{bmatrix} a(i_3,i_2,i_4)b(i_3,i_2,i_4,i_1) + a(i_3,i_4,i_2)b(i_3,i_2,i_4,i_1) \\ +a(i_3,i_2,i_4)b(i_2,i_3,i_4,i_1) + a(i_3,i_4,i_2)b(i_2,i_3,i_4,i_1) \end{bmatrix} I_{i_1,z} I_{i_2,z} \\
& + \frac{i}{16}\begin{bmatrix} a(i_3,i_4,i_2)b(i_3,i_1,i_4,i_2) + a(i_3,i_2,i_4)b(i_3,i_1,i_4,i_2) \\ +a(i_3,i_4,i_2)b(i_1,i_3,i_4,i_2) + a(i_3,i_2,i_4)b(i_1,i_3,i_4,i_2) \end{bmatrix} I_{i_1,x} I_{i_2,x}
\end{aligned} \tag{21}$$

reveals the importance of a symbolic computational tool like SD-CAS in deriving analytic expressions in terms of spin operator products: in particular, the computation of the commutator C=[A,B] took about 0.6 s on a *Pentium 4* PC, which is orders of magnitude less than the time required if the same operation would have been performed by conventional analytical calculations.

The computational efficiency in evaluating the commutator between two general spin-correlation terms has been further investigated by progressively increasing the number of spin operators within B up to $m_b = 32$, when a reasonable upper limit of about one hour for the CPU time was reached. This corresponds to a number $\chi \sim 3 \times 10^4$ of basic computational steps, which confirms the good performances of YACAS in working with lists. However, the greatest benefit of using a list representation for spin correlations in SD-CAS results from the low memory usage compared with the memory requirements in numerical simulation programs based on a matrix representation. Specifically, it was obtained an increase in the memory usage from 8 Mb to about 180 Mb when going from $m_b = 4$ to $m_b = 32$ (which is actually not surprising taking into account the linear increase of the corresponding list size with the number of the incorporated spin operators), whereas the memory required to only store a matrix of size $2^{36} \times 2^{36}$representing the B spin correlation cannot be provided by the existing technology.

3.2. Application: multi-spin correlations in dipolar coupled systems

The SD-CAS functions introduced so far implement a minimal set of operations that are in principle sufficient for obtaining insights into the NMR spin dynamics directly from the computed symbolic expressions of relevant physical quantities. Prominent examples are the derivation of effective Hamiltonians under various periodic perturbations by multiple-pulse sequences, MAS, and the combination of the two, or the evaluation of the density operator in the form of a spin correlations sum, Eq. (12). An example of this latter type will be presented in the following. In particular, the evolution of an initial state of transverse polarization, $\rho_0 = \sum_{i_1}^{N} I_{i_1,x}$, in an N spin system under the action of mutual dipolar interactions is considered within an approximation based on the power expansion of the density operator,

$$\rho(t) = \rho_0 - \frac{it}{1!}[H2,\rho_0] + \frac{(it)^2}{2!}[H2,[H2,\rho_0]] + \ldots + \frac{(it)^n}{2!} \times \underbrace{[H2,\ldots[H2,\rho_0]]}_{n} + \ldots \tag{22}$$

and the result will be then analyzed with respect to the multi-spin correlations generated during the propagation process.

For this purpose, an SD-CAS script called `Dens_Op_Obs` has been written (Appendix C) in which the function `RhoApprox(n)` enables one to compute the nth order term in (22), according to:

$$\rho^{(n)} = \underbrace{[H2,\ldots,[H2,\rho_0]]}_{n} \tag{23}$$

This function is centered on the operation `GenSumSpComm` described above, but it also contains a few auxiliary input/output operations. Namely, the particular form of the input parameters, i.e., the Hamiltonian, *Hint*, the initial density operator, ρ_0, and a possible observable, *Obs*, are red from the file `input.txt` shown in the Appendix C, whereas the results are written in corresponding output files. Also, the special case of the homonuclear dipolar interaction has been considered for *Hint*, i.e., $D1 = D2 = -D$, and $D3 = 2D$, in the general form, Eq. (17), of a two-spin Hamiltonian $H2$.

A theoretical analysis aimed at obtaining physical insights into the spin order propagation process described by Eq. (22) can be performed at different levels of sophistication. In the simplest approach, the explicit dependence on spin labels is omitted from the expression of the dipolar coefficients, D, which renders them all equal with each other. On the one hand, this will remove the connection between the weighting coefficients in Eq. (22) and the structure of the coupled spins network, but on the other hand, the procedure enables their quick evaluation up to relatively high orders at modest computational costs. For instance, the calculation of $\rho^{(n)}$ terms up to the 20th order in our example (performed by running the SD-CAS script `Dens_Op_Obs`, with *Hint* in the input file set to `HInt:={{-D, {i[1], i[2]}, {}, {}}, {-D, {}, {i[1], i[2]}, {}}, {2*D, {}, {}, {i[1], i[2]}}}`) took only ~2.5 h on a conventional *Pentium 4* PC.

Historically, there have been numerous studies performed in the past with the purpose of deriving consistent theoretical models for the spatiotemporal growth of multiple spin/multiple quantum coherences in extended systems of dipolar coupled spins [39–41]. All of the models developed so far are essentially statistical in nature: they describe the spread of the initial spin order through a hopping process between certain points within the accessible Liouville subspace. A disadvantage of this approach is that such states are highly degenerated, as they are labeled according only to the number of spin operators and/or coherence order. Put in this context, the results obtained here even in this lowest approximation reveal the benefits of incorporating a tool like SD-CAS within the analysis. In particular, *the exact distribution* with respect to the number n_x, n_y, and n_z of the spin operators I_x, I_y, and I_z within each such manifold can be easily determined by symbolic computations, which provides a complete picture of the subspace where the evolution process is confined. Using now the compact notation

$$S \sim \underbrace{I_{i_1,x} I_{i_2,x} \ldots I_{i_{nx},x}}_{n_x} \times \underbrace{I_{i_{nx+1},y} I_{i_{nx+2},y} \ldots I_{i_{nx+ny},y}}_{n_y} \times \underbrace{I_{i_{nx+ny+1},z} I_{i_{nx+ny+2},z} \ldots I_{i_{nx+ny+nz},z}}_{n_z} \equiv [n_x\ n_y\ n_z],$$

which is appropriate when the weighting coefficients do not explicitly depend on spin labels, the spin correlations generated by running the script `Dens_Op_Obs` are listed below in Eq. (24) up to the 10th order in the power expansion (22):

$$\begin{aligned}
&\rho^{(0)} \Rightarrow [1\,0\,0];\\
&\rho^{(1)} \Rightarrow [0\,1\,1];\\
&\rho^{(2)} \Rightarrow [1\,0\,0];[1\,0\,2];[1\,2\,0];\\
&\rho^{(3)} \Rightarrow [0\,1\,1];[0\,1\,3];[0\,3\,1];[2\,1\,1];\\
&\rho^{(4)} \Rightarrow [1\,0\,0];[1\,0\,2];[1\,0\,4];[1\,2\,0];[1\,2\,2];[1\,4\,0];[3\,0\,0];[3\,0\,2];[3\,2\,0];\\
&\rho^{(5)} \Rightarrow [0\,1\,1];[0\,1\,3];[0\,1\,5];[0\,3\,1];[0\,3\,3];[0\,5\,1];[2\,1\,1];[2\,1\,3];[2\,3\,1];[4\,1\,1];\\
&\rho^{(6)} \Rightarrow [1\,0\,0];[1\,0\,2];[1\,0\,4];[1\,0\,6];[1\,2\,0];[1\,2\,2];[1\,2\,4];[1\,4\,0];[1\,4\,2];[1\,6\,0];\\
&\qquad [3\,0\,0];[3\,0\,2];[3\,0\,4];[3\,2\,0];[3\,2\,2];[3\,4\,0];[5\,0\,0];[5\,0\,2];[5\,2\,0];\\
&\rho^{(7)} \Rightarrow [0\,1\,1];[0\,1\,3];[0\,1\,5];[0\,1\,7];[0\,3\,1];[0\,3\,3];[0\,3\,5];[0\,5\,1];[0\,5\,3];[0\,7\,1];\\
&\qquad [2\,1\,1];[2\,1\,3];[2\,1\,5];[2\,3\,1];[2\,3\,3];[2\,5\,1];[4\,1\,1];[4\,1\,3];[4\,3\,1];[6\,1\,1];\\
&\rho^{(8)} \Rightarrow [1\,0\,0];[1\,0\,2];[1\,0\,4];[1\,0\,6];[1\,0\,8];[1\,2\,0];[1\,2\,2];[1\,2\,4];[1\,2\,6];[1\,4\,0];\\
&\qquad [1\,4\,2];[1\,4\,4];[1\,6\,0];[1\,6\,2];[1\,8\,0];[3\,0\,0];[3\,0\,2];[3\,0\,4];[3\,0\,6];[3\,2\,0];\\
&\qquad [3\,2\,2];[3\,2\,4];[3\,4\,0];[3\,4\,2];[3\,6\,0];[5\,0\,0];[5\,0\,2];[5\,0\,4];[5\,2\,0];[5\,2\,2];\\
&\qquad [5\,4\,0];[7\,0\,0];[7\,0\,2];[7\,2\,0];\\
&\rho^{(9)} \Rightarrow [0\,1\,1];[0\,1\,3];[0\,1\,5];[0\,1\,7];[0\,1\,9];[0\,3\,1];[0\,3\,3];[0\,3\,5];[0\,3\,7];[0\,5\,1];\\
&\qquad [0\,5\,3];[0\,5\,5];[0\,7\,1];[0\,7\,3];[0\,9\,1];[2\,1\,1];[2\,1\,3];[2\,1\,5];[2\,1\,7];[2\,3\,1];\\
&\qquad [2\,3\,3];[2\,3\,5];[2\,5\,1];[2\,5\,3];[2\,7\,1];[4\,1\,1];[4\,1\,3];[4\,1\,5];[4\,3\,1];[4\,3\,3];\\
&\qquad [4\,5\,1];[6\,1\,1];[6\,1\,3];[6\,3\,1];[8\,1\,1];\\
&\rho^{(10)} \Rightarrow [1\,0\,0];[1\,0\,2];[1\,0\,4];[1\,0\,6];[1\,0\,8];[1\,0\,10];[1\,2\,0];[1\,2\,2];[1\,2\,4];[1\,2\,6];\\
&\qquad [1\,2\,8];[1\,4\,0];[1\,4\,2];[1\,4\,4];[1\,4\,6];[1\,6\,0];[1\,6\,2];[1\,6\,4];[1\,8\,0];[1\,8\,2];\\
&\qquad [1\,10\,0];[3\,0\,0];[3\,0\,2];[3\,0\,4];[3\,0\,6];[3\,0\,8];[3\,2\,0];[3\,2\,2];[3\,2\,4];[3\,2\,6];\\
&\qquad [3\,4\,0];[3\,4\,2];[3\,4\,4];[3\,6\,0];[3\,6\,2];[3\,8\,0];[5\,0\,0];[5\,0\,2];[5\,0\,4];[5\,0\,6];\\
&\qquad [5\,2\,0];[5\,2\,2];[5\,2\,4];[5\,4\,0];[5\,4\,2];[5\,6\,0];[7\,0\,0];[7\,0\,2];[7\,0\,4];[7\,2\,0];\\
&\qquad [7\,2\,2];[7\,4\,0];[9\,0\,0];[9\,0\,2];[9\,2\,0];
\end{aligned} \tag{24}$$

From a brief inspection of the relationships (24) one should notice on one hand that odd and even orders define independent subspaces, which do not share common spin correlations; on the other hand, if taken separately, any given odd (even) order n includes all the spin correlations present in the previous order, $n-2$, in addition to specific terms that correspond to n- and $n+1$-spin correlations. The spin correlations shown in (24) have been determined by effectively computing the required commutators through the systematic use of the `GenSumSpComm` SD-CAS function in the `Dens_Op_Obs` script. However, from a closer analysis of the results it came out clearly that such rigorous calculations can be alternatively replaced by the following rules:

$$\left.\begin{aligned} &n_x = 1,3,\ldots,n-1\\ &n_y, n_z \rightarrow \text{ are even}\\ &0 \leqslant n_y + n_z \leqslant n - n_x + 1 \end{aligned}\right\} \text{ if } n \text{ is even} \tag{25}$$

and

$$\left.\begin{aligned} &n_x = 0,2,\ldots,n-1\\ &n_y, n_z \rightarrow \text{ are odd}\\ &2 \leqslant n_y + n_z \leqslant n - n_x + 1 \end{aligned}\right\} \text{ if } n \text{ is odd} \tag{26}$$

which define a propagation pattern that can be conveniently used for the direct evaluation of the $[n_x\ n_y\ n_z]$ terms up to arbitrarily high orders n in the power expansion of the density operator.

At the other extreme, the full expressions of the terms in Eq. (22) are obtained when the labels of the coupled spins are explicitly incorporated in the corresponding dipolar coefficient, $D(i_j,i_k)$. In practice this is accomplished by running the SD-CAS script `Dens_Op_Obs`, with *Hint* in the input file written in its complete form, `HInt :={{-D(i[1], i[2]), {i[1], i[2]}, {}, {}}, {- D(i[1], i[2]), {}, {i[1], i[2]}, {}}, {2*D(i[1], i[2]), {}, {}, {i[1], i[2]}}}`. The $\rho^{(n)}$ terms computed in this way, and transformed to the conventional analytical representation, are listed in Eq. (27) up to the third order:

$$
\begin{aligned}
\rho^{(1)} &= 3iD(i_1,i_2)I_{i_1,y}I_{i_2,z}\\
\rho^{(2)} &= [6D(i_1,i_2)D(i_1,i_3)+3D(i_1,i_3)D(i_2,i_3)]I_{i_1,x}I_{i_2,z}I_{i_3,z}\\
&\quad +3[D(i_1,i_2)D(i_1,i_3)-D(i_1,i_3)D(i_2,i_3)]I_{i_1,x}I_{i_2,y}I_{i_3,y}\\
&\quad +\frac{9}{4}D(i_1,i_2)^2I_{i_1,x}\\
\rho^{(3)} &= 3i\begin{bmatrix}D(i_2,i_3)D(i_1,i_4)D(i_3,i_4)+4D(i_2,i_3)D(i_1,i_4)D(i_1,i_3)\\-D(i_2,i_3)D(i_1,i_4)D(i_2,i_4)-4D(i_2,i_3)D(i_1,i_4)D(i_1,i_2)\\+D(i_2,i_3)D(i_3,i_4)D(i_1,i_3)+2D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)\\+D(i_2,i_3)D(i_2,i_4)D(i_1,i_2)+D(i_1,i_4)D(i_3,i_4)D(i_2,i_4)\\-2D(i_1,i_4)D(i_1,i_3)D(i_2,i_4)+D(i_3,i_4)D(i_1,i_3)D(i_2,i_4)\\-4D(i_1,i_3)D(i_2,i_4)D(i_1,i_2)\end{bmatrix}I_{i_1,x}I_{i_2,x}I_{i_3,y}I_{i_4,z}\\
&\quad +3i\begin{bmatrix}D(i_2,i_3)D(i_3,i_4)D(i_1,i_4)+2D(i_1,i_4)D(i_3,i_4)D(i_2,i_4)\\+2D(i_1,i_4)D(i_2,i_3)D(i_1,i_3)+4D(i_1,i_4)D(i_1,i_3)D(i_1,i_2)\end{bmatrix}I_{i_1,y}I_{i_2,z}I_{i_3,z}I_{i_4,z}\\
&\quad +3i\begin{bmatrix}D(i_3,i_4)D(i_2,i_4)D(i_1,i_4)-D(i_3,i_4)D(i_2,i_4)D(i_1,i_2)\\-2D(i_3,i_4)D(i_1,i_2)D(i_2,i_3)+2D(i_3,i_4)D(i_2,i_3)D(i_1,i_3)\end{bmatrix}I_{i_1,y}I_{i_2,y}I_{i_3,y}I_{i_4,z}\\
&\quad +\frac{9i}{4}\begin{bmatrix}2D(i_1,i_3)^2D(i_2,i_3)+5D(i_1,i_3)^2D(i_1,i_2)-D(i_1,i_3)D^2(i_2,i_3)\\+2D(i_2,i_3)^2D(i_1,i_2)+D(i_1,i_3)D(i_2,i_3)D(i_1,i_2)+3D(i_1,i_2)^3\end{bmatrix}I_{i_1,y}I_{i_2,z}
\end{aligned}
\tag{27}
$$

They provide the exact formulae of the specified terms in the analytical expression of the density operator with respect to both, spin and coupling variables. For simplifying presentation, the summation over the spin labels i_k has been omitted but, however, this should be implicitly assumed within all these relationships. It is seen that, although they are fairly simple with respect to the involved spin operator products, the approximation method based on a power expansion of the density operator makes the incorporated weighting coefficients to become extremely complex functions of coupling coefficients with going to higher order terms. For this reason, the fourth order term, of which expression extends over three pages, is no longer listed here.

The computed symbolic expression of the density operator can be further fed into the relationship (9) and then used for quantitative investigations. The way that SD-CAS could be employed for such a purpose is briefly illustrated in the reminder of this section. Specifically, the most straightforward approach to calculate the expectation value of a given NMR observable, *M*, based on Eq. (9) requires a set of three successive operations: (a) evaluate the scalar product (*Trace*) of the spin correlations within the product $M\rho(t)$, (b) bring the symbolic expressions of the a_j coefficients in Eq. (12) to an algebraic form by replacing the abstract interaction parameters (like the *C* and *D* couplings in Eq. (11)) with their corresponding algebraic expressions, and (c) bring these expressions to a numerical form by replacing the coupling constants with values corresponding to the particular spin system under study, perform the summations over the all spins and, if necessary, perform the orientational average.

The first step consists of an averaging over spin variables: as such, it naturally fits within the SD-CAS framework developed here, and thus will be the only one explicitly treated in this work. As shown in the Appendix B, for the two different types of spin correlation products introduced in SD-CAS, there are equivalent YACAS functions that implement the corresponding scalar product, namely, `ScalarSpProd(X,Y,N)`, and `ScalarSumSpProd(X,Y,N)`, where `N` is the number of spins into the system. The practical procedure to determine the scalar product between the X and Y spin-correlation terms is fairly simple: the value of this product is assigned to `SpProd(X,Y)[l]`$*2^N$ if the spin parts of `X` and `Y` coincide, and to 0 if otherwise. The function `ScalarSumSpProd` performs the scalar product among the all terms within the spin correlation sums, `X` and `Y`, and then the results are added together.

In the following, we consider the particular case of measuring the spin polarization along the *x* axis, i.e., the observable operator is set to $Obs = I_x = \sum_{i_1}^{N} I_{i_1,x}$. Inserting this in Eq. (22) one obtains

$$
\langle I_x(t)\rangle = 1 - \frac{t^2}{2!}\left\langle I_x^{(2)}(t)\right\rangle + \frac{t^4}{4!}\left\langle I_x^{(4)}(t)\right\rangle - \frac{t^6}{6!}\left\langle I_x^{(6)}(t)\right\rangle + \cdots \tag{28}
$$

where the corresponding contribution to the normalized expectation value of $\langle I_x \rangle$ is:

$$
\left\langle I_x^{(n)}(t)\right\rangle = \frac{(-it)^n}{n!}\frac{\langle I_x|\rho^{(n)}\rangle}{\langle I_x|\rho_0\rangle} \tag{29}
$$

The SD-CAS computation of the *n*th order term can be accomplished by running the function `Observable(N, n, Obs, t)` in the `Dens_Op_Obs` script, where the list representation of the *Obs* operator is red from the input file, `input.txt`. This operation corresponds to the step (a) of the procedure described above and the results obtained up to the 6th order in the power expansion (28) are displayed below (the odd order terms are all zero).

$$
\begin{aligned}
\langle I_x^{(2)}(t)\rangle &= \frac{9}{2^2}D(i_1,i_2)^2\\
\langle I_x^{(4)}(t)\rangle &= \frac{27}{2^4}D(i_1,i_2)\begin{bmatrix}3D(i_1,i_2)^3+D(i_1,i_2)D(i_2,i_3)D(i_1,i_3)\\+3D(i_1,i_2)D(i_2,i_3)^2+4D(i_1,i_2)D(i_1,i_3)^2+D(i_2,i_3)D(i_1,i_3)^2\end{bmatrix}\\
\langle I_x^{(6)}(t)\rangle &= \frac{9}{2^6}D(i_1,i_2)\\
&\quad \times \begin{bmatrix}
81D(i_1,i_2)^5+132D(i_1,i_2)^3D(i_2,i_3)^2+108D(i_1,i_2)^3D(i_2,i_4)^2D(i_1,i_3)\\
+183D(i_1,i_2)^3D(i_1,i_3)^2+93D(i_1,i_2)^3D(i_3,i_4)+27D(i_1,i_2)^3D(i_2,i_4)D(i_1,i_4)\\
+96D(i_1,i_2)^3D(i_1,i_4)^2-18D(i_1,i_2)^2D(i_2,i_3)^3-12D(i_1,i_2)^2D(i_2,i_3)^2D(i_1,i_3)\\
+30D(i_1,i_2)^2D(i_2,i_3)D(i_1,i_3)^2-18D(i_1,i_2)^2D(i_1,i_3)^3+12D(i_1,i_2)^2D(i_2,i_4)^2D(i_1,i_4)\\
+15D(i_1,i_2)^2D(i_2,i_4)D(i_1,i_4)^2+153D(i_1,i_2)D(i_2,i_3)^4+54D(i_1,i_2)D(i_2,i_3)^3D(i_1,i_3)\\
+189D(i_1,i_2)D(i_2,i_3)^2D(i_1,i_3)^2+152D(i_1,i_2)D(i_2,i_3)^2D(i_2,i_4)^2\\
+13D(i_1,i_2)D(i_2,i_3)^2D(i_2,i_4)D(i_1,i_4)+4D(i_1,i_2)D(i_2,i_3)^2D(i_2,i_4)D(i_3,i_4)\\
+63D(i_1,i_2)D(i_2,i_3)^2D(i_1,i_4)^2+2D(i_1,i_2)D(i_2,i_3)^2D(i_1,i_4)D(i_3,i_4)\\
+81D(i_1,i_2)D(i_2,i_3)^2D(i_3,i_4)^2+45D(i_1,i_2)D(i_2,i_3)D(i_1,i_3)^3\\
+55D(i_1,i_2)D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)^2+84D(i_1,i_2)D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)D(i_1,i_4)\\
+13D(i_1,i_2)D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)D(i_3,i_4)+38D(i_1,i_2)D(i_2,i_3)D(i_1,i_3)D(i_1,i_4)^2\\
-4D(i_1,i_2)D(i_2,i_3)D(i_1,i_3)D(i_1,i_4)D(i_3,i_4)+3D(i_1,i_2)D(i_2,i_3)D(i_1,i_3)D(i_3,i_4)^2\\
+14D(i_1,i_2)D(i_2,i_3)D(i_2,i_4)^2D(i_3,i_4)+4D(i_1,i_2)D(i_2,i_3)D(i_2,i_4)D(i_1,i_4)D(i_3,i_4)\\
+4D(i_1,i_2)D(i_2,i_3)D(i_2,i_4)D(i_3,i_4)^2+6D(i_1,i_2)D(i_2,i_3)D(i_1,i_4)^2D(i_3,i_4)\\
-12D(i_1,i_2)D(i_2,i_3)D(i_1,i_4)D(i_3,i_4)^2+216D(i_1,i_2)D(i_1,i_3)^4+63D(i_1,i_2)D(i_1,i_3)^2D(i_2,i_4)^2\\
+20D(i_1,i_2)D(i_1,i_3)^2D(i_2,i_4)D(i_1,i_4)-2D(i_1,i_2)D(i_1,i_3)^2D(i_2,i_4)D(i_3,i_4)\\
+232D(i_1,i_2)D(i_1,i_3)^2D(i_1,i_4)^2+32D(i_1,i_2)D(i_1,i_3)^2D(i_1,i_4)D(i_3,i_4)\\
+96D(i_1,i_2)D(i_1,i_3)^2D(i_3,i_4)^2+12D(i_1,i_2)D(i_1,i_3)D(i_2,i_4)^2D(i_3,i_4)\\
-4D(i_1,i_2)D(i_1,i_3)D(i_2,i_4)D(i_1,i_4)D(i_3,i_4)-12D(i_1,i_2)D(i_1,i_3)D(i_2,i_4)D(i_3,i_4)^2\\
+31D(i_1,i_2)D(i_1,i_3)D(i_1,i_4)^2D(i_3,i_4)-16D(i_1,i_2)D(i_1,i_3)D(i_1,i_4)D(i_3,i_4)^2\\
+18D(i_2,i_3)^4D(i_1,i_3)+63D(i_2,i_3)^3D(i_1,i_3)^2-14D(i_2,i_3)^2D(i_1,i_3)D(i_2,i_4)^2\\
-2D(i_2,i_3)^2D(i_1,i_3)D(i_2,i_4)D(i_1,i_4)+39D(i_2,i_3)^2D(i_1,i_3)D(i_2,i_4)D(i_3,i_4)\\
-26D(i_2,i_3)^2D(i_1,i_3)D(i_1,i_4)^2+3D(i_2,i_3)^2D(i_1,i_3)D(i_1,i_4)D(i_3,i_4)\\
+24D(i_2,i_3)^2D(i_1,i_3)D(i_3,i_4)^2+8D(i_2,i_3)^2D(i_2,i_4)^2D(i_1,i_4)+8D(i_2,i_3)^2D(i_2,i_4)D(i_1,i_4)^2\\
-4D(i_2,i_3)^2D(i_2,i_4)D(i_1,i_4)D(i_3,i_4)-36D(i_2,i_3)^2D(i_1,i_4)^2D(i_3,i_4)\\
+22D(i_2,i_3)^2D(i_1,i_4)D(i_3,i_4)^2+72D(i_2,i_3)D(i_1,i_3)^4+11D(i_2,i_3)D(i_1,i_3)^3D(i_2,i_4)^2\\
+2D(i_2,i_3)D(i_1,i_3)^2D(i_2,i_4)D(i_1,i_4)+12D(i_2,i_3)D(i_1,i_3)^2D(i_2,i_4)D(i_3,i_4)\\
+11D(i_2,i_3)D(i_1,i_3)^2D(i_1,i_4)^2+45D(i_2,i_3)D(i_1,i_3)^2D(i_1,i_4)D(i_3,i_4)\\
+84D(i_2,i_3)D(i_1,i_3)^2D(i_3,i_4)^2+8D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)^2D(i_1,i_4)\\
-20D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)^2D(i_3,i_4)+4D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)D(i_1,i_4)^2\\
-24D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)D(i_1,i_4)D(i_3,i_4)+10D(i_2,i_3)D(i_1,i_3)D(i_2,i_4)D(i_3,i_4)^2\\
-16D(i_2,i_3)D(i_1,i_3)D(i_1,i_4)^2D(i_3,i_4)+8D(i_2,i_3)D(i_1,i_3)D(i_1,i_4)D(i_3,i_4)^2\\
+35D(i_2,i_3)D(i_2,i_4)^2D(i_1,i_4)D(i_3,i_4)-D(i_2,i_3)D(i_2,i_4)D(i_1,i_4)^2D(i_3,i_4)\\
-4D(i_2,i_3)D(i_2,i_4)D(i_1,i_4)D(i_3,i_4)^2-20D(i_1,i_3)^2D(i_2,i_4)^2D(i_1,i_4)\\
-36D(i_1,i_3)^2D(i_2,i_4)^2D(i_3,i_4)+28D(i_1,i_3)^2D(i_2,i_4)D(i_1,i_4)^2\\
-8D(i_1,i_3)^2D(i_2,i_4)D(i_1,i_4)D(i_3,i_4)+32D(i_1,i_3)^2D(i_2,i_4)D(i_3,i_4)^2\\
-2D(i_1,i_3)D(i_2,i_4)^2D(i_1,i_4)D(i_3,i_4)+49D(i_1,i_3)D(i_2,i_4)D(i_1,i_4)^2D(i_3,i_4)\\
+13D(i_1,i_3)D(i_2,i_4)D(i_1,i_4)D(i_3,i_4)^2
\end{bmatrix}
\end{aligned}
$$

The result obtained above is very illustrative for the complexity level that can be attained in the analytical treatment of spin dynamics if symbolic computations are used instead of classical calculations. However, in practice, it is exactly this high complexity the main reason why the power expansion approach (22) is generally not

considered an appropriate method for studying the spin order propagation in strongly dipolar coupled multi-spin systems, and therefore replaced with various simplifying models [42].

Expressions like those derived above are very general because they are valid for any multi-spin system of which dynamics is driven by mutual homonuclear dipolar couplings. In applications to particular systems (with respect to the number, and the spatial positions of the incorporated spins), such expressions will enter as input parameters for the next two steps, (b) and (c), of the protocol outlined above. For instance, considering the simplest case of a two spin system, all the terms containing spin labels i_k, with $k > 2$, in the above relationships are set to zero, and Eq. (28) transforms to

$$\begin{aligned}\langle I_x(t)\rangle &= 1 - \frac{t^2}{2!}\left[\frac{3}{2}D(i_1,i_2)\right]^2 + \frac{t^4}{4!}\left[\frac{3}{2}D(i_1,i_2)\right]^4 - \frac{t^6}{6!}\left[\frac{3}{2}D(i_1,i_2)\right]^6 + \cdots \\ &= \cos\left[\frac{3}{2}D(i_1,i_2)t\right]\end{aligned}$$

Obviously, this represents the simplest case when an exact analytical expression can be easily derived from the power expansion of the normalized x spin-polarization. Inserting now the operations specified at the points (b) and (c), the symbolic expression of $\langle I_x(t)\rangle$ finally transforms to

$$\begin{aligned}\langle I_x(t)\rangle &\rightarrow \cos\left[\frac{3}{2}\sum_{i_k=1}^{2}{}'D(i_1,i_2)t\right] \\ &\rightarrow \frac{1}{2\pi}\int_0^{2\pi}d\varphi\int_0^{\pi}\cos\left[3\omega_{D0}\frac{(3\cos^2\theta_{12}-1)}{2r_{12}^3}t\right]\cos(\theta_{12})d\theta_{12}\end{aligned}$$

where, ω_{D0} is the dipolar coupling between two spins separated by 1 Å, r_{12} is the internuclear distance (in Å), and θ_{12} is the angle of the internuclear vector $\boldsymbol{r}_{12}$ with respect to the static magnetic field. In principle, also such operations can be performed by means of specific computer algebra procedures: however, as they will rely on completely different principles compared with the SD-CAS functions introduced here for operating in the spin space, further investigations along these lines are no longer considered in the present work.

The last relationship constitutes a good starting point for extending the discussion also to time-dependent Hamiltonians, e.g., to the case of MAS. For this purpose the same `Dens_Op_Obs` script can be employed, but after performing the following changes: (i) the dipolar couplings $D(i_1,i_2)$ will have to be replaced with their time dependent expressions under MAS,

$$D(i_1,i_2,t) = \omega_{D0}\sum_{m=-2}^{2}{}'b_m(\theta_{12})e^{im\phi_{12}}e^{im\omega_R t},$$

and (ii) the general power expansion of the time-propagator, Eq. (6), will have to be considered when evaluating the density operator instead of its static form incorporated in Eq. (22). Obviously, this will give rise to more complex weighting coefficients in Eq. (27), but otherwise the two cases remain equivalent with respect to the generated spin correlations. The result clearly shows that the SD-CAS functions developed here to operate in the spin space are generally applicable to both, static and rotating samples. As such, they are well suited to be included within a core library of functions and procedures which provide the basic building blocks for CAS implementation of particular NMR experiments.

To conclude, the examples discussed so far demonstrate that symbolic computations may provide new practical solutions to more effectively cope with the inherent complexity of the dynamics in extended spin systems, which is characteristic to many modern NMR techniques and applications.

4. Conclusions and outlook

The possibility of automating analytical calculations of quantum spin evolution was explored in the present work. Symbolic computations integrated within a CAS framework called *Spin Dynamics by Computer Algebra System* (SD-CAS) have been shown to represent a promising new approach for replacing the traditional manner in which such calculations are carried out (by hand) with more powerful tools. This concept is illustrated here with the SD-CAS implementation of the 1/2-spin operator algebra within a Liouville space of witch base vectors are given in terms of products of Cartesian spin operators. In practice, specific objects and operations have been coded as a set of specialized function libraries within a Computer Algebra System named YACAS. As a most prominent feature in SD-CAS, one should remark the use of a nested list representation for spin operators instead of the matrix representation that is commonly employed in numerical simulation programs. As such, the size of spin operators scales linearly with the number of spins into the system, and not exponentially as in the case of the matrix representation, thus largely reducing memory requirements in multi-spin computations. For the so defined spin operators, functions that are essential for the SD-CAS implementation of the basic spin algebra operations, namely, the spin operator product, commutator, and scalar product have been written for two distinct cases: when the spin labels are either fixed to certain values, or considered running labels, respectively.

The practical examples provided throughout this work demonstrate the potential of SD-CAS for increasing the efficiency of analytical calculations, both, in terms of speed, and complexity level that can be attained in various applications. However, difficulties have also been encountered during implementation, mostly coming from specific features of the employed CAS. For instance, the YACAS built in function `Simplify(x)` incorporated within the script `Dens_Op_Obs` was found responsible for slowing down the computation of the $\rho^{(n)}$ terms in (27), which calls for further optimization work. Also, it is important to emphasize that the functions introduced in this work cover only a small part of possible applications of symbolic computations in investigating spin dynamics. In reality, there are many other theoretical methods, formalisms, and approximations commonly employed for this purpose and suitable for SD-CAS implementation. For example, one could mention the implementation of additional basis sets in terms of the rising and lowering operators, I_+ and I_-, fictitious spin-1/2 operators, spherical tensor operators (which are sometimes preferable to the Cartesian spin operators considered in this work), extension to heteronuclear systems, and to spin quantum numbers larger than 1/2. In conclusion, the present study not only that introduces SD-CAS as a promising tool for analytical treatment of quantum evolution in complex spin systems, but, at the same time it opens up new areas of investigation. In particular, the next steps in the development process should be focused on bringing SD-CAS from the present stage of setting basic principles and concepts, towards making it a fully functional symbolic computation platform.

Acknowledgments

This work was supported by CNCSIS-UEFISCSU, Project no. PN2-IDEI 872/2008. The collaboration with Prof. Steven P. Brown through a Royal Society joint project is also gratefully acknowledged.

Appendix A

```
/* IsSpCorr checks if a product of spin operators (spin correlation) is put in
 * its irreducible form; this is defined as a 4-entry list {c{i[1], i[2],
 * ...},{i[j],i[j+1],...},{i[k],i[k+1],...}}
 * which contains a numerical (generally complex) corefficient, and integer
 * indices for the x, y, and z, spin operators, for example
 * c*I_{i[1]x}*I_{i[2]x}*...*I_{i[j]y}*I_{i[j+2]}*...*I_{i[k]z}*I_{i[k+1z}*
 * ..., with the condition that none of the coefficients are allowed to be equal
 *   with each other
 */
IsSpCorr(_X) <-- And (Length(X) = 4, IsList(X[2]), IsList(X[3]), IsList(X[4]),
Intersection(X[2], x[3]) = {}, Intersection(X[2], X[4]) = {}, Intersection(X[3], X[4])
= {});

/* IsSumSpCorr checks if X is a sum of spin correlations .
 * The internal representation of a general sum of spin correlations
 * is defined as a  list of spin correlations, {Q1, Q2, ..., Qn}, where
 * Qn is a spin correlation as defined above
 */
IsSumSpCorr(_X) <-- And (Length(X) > 1, ForEach(i, X) IsSpCorr(i));


/* SymbOrder orders the labeles i[j] inside a spin correlation term
 * according to the value of their index j
 */
Function("SymbOrder", {X}) [
Local(k,p);
p:=1;
If(Not(Equals(Length(X[2]),0)), [
For (k:=1, k<=Length(X[2]), k++) [
X[1]:= Subst(X[2][k], j[k]) X[1];
DestructiveReplace(X[2],k,i[k]);
];
]
);

If(Not(Equals(Length(X[3]),0)), [
For (k:=1, k<=Length(X[3]), k++) [
X[1]:=Subst(X[3][k], j[k+Length(X[2])]) X[1];
DestructiveReplace(X[3],k,i[k+Length(X[2])]);
];
]
);

If(Not(Equals(Length(X[4]),0)), [
For (k:=1, k<=Length(X[4]), k++) [
X[1]:= Subst(X[4][k], j[k+Length(X[2])+Length(X[3])]) X[1];
DestructiveReplace(X[4],k,i[k+Length(X[2])+Length(X[3])]);
];
]
);

k:=0;
While(HasExpr(X[1],i)) [
k++;
If(HasExpr(X[1],i[k]), [
X[1]:= Subst(i[k],j[Length(X[2])+Length(X[3])+Length(X[4])+p]) X[1];
p++;
]
);
];

X[1]:= Subst(j, i) X[1];
X;
];

/* The function ReduceSumSpCorr brings a sum of spin correlations X to
* its simplest form where all the spin correlation terms are different
* from each other
*/
Function("ReduceSumSpCorr", {X}) [
Local(i, j);
```

```
For (i:=1, i<=Length(X), i++) [
        For (j:=i+1, j<=Length(X), j++) [
        If(Equals(Tail(X[j]), Tail(X[i])), [
        DestructiveReplace(X[i], 1, (X[i][1]+X[j][1]));
        DestructiveDelete(X, j);
        j--;
        ]
        );
        ];
];
X;
];

/* Index extracts the value of the index k within the label i[k]
 */
Function("Index", {X}) [
Local(n,i);
If(Equals(Type(X),"Nth"), [
n:=0;
Until(Equals(Nth(i,n),X) Or Equals(Nth(j,n),X)) n++;
n;
]
);
];

/* The function NoSpins retrieves the number of spins involved in defining a
* spin correlation term (this is extracted as the number of the distinct i[k]
* spin labeles in the corresponding expression)
*/
Function("NoSpins", {X}) [
Local(k,n);
k:=0;
n:=0;
While(HasExpr(X,i)) [
k++;
If(HasExpr(X,i[k]), [
X:= Subst(i[k],j[k]) X;
n++;
]
);
];
n;
];

/* The function NoSpOp retrieves the number of spin operators within a spin
* correlation term (this is extracted as the number of the distinct i[k]
* spin labeles in the spin part of the associated list)
*/
Function("NoSpOp", {X}) [
Local(k,y);
k:=0;
y:=Delete(X, 1);
While(HasExpr(y,i)) [
k++;
If(HasExpr(y,i[k]), [
y:= Subst(i[k],j[k]) y;
]
);
];
k;
];

/* The function ExtrSp extracts in the form of a simple list {i[1], i[2], ...,
* i[n]} the spin component of a spin correlation term expressed as a
* first order SD-CAS list. This function is applicable also when j[k] spin
* labels are used in defining the spin correlation term
*/
Function("ExtrSp", {X}) [
Local(result);
Set(result, Concat(X[2],X[3], X[4]));
result;
];

/* The function ExpandSp expands the the spin part of a spin correlation term in
* first order spin correlations nested list, from the {c, {i[1], ..}, {i[p], ..}
* ,{i[q], ...}} form to the (a, {i[1], ... i[p], ..., i[q], ...}} form. This
```

```
* function is applicable also when j[k] spin labels are used in defining the
* spin correlation term
*/
Function("ExpandSp", {X}) [
Local(result);
Set(result, {X[1],Concat(X[2],X[3], X[4])});
result;
];

/* The function InvExpandSp performs the reverse transformation, from the {a,
* {i[1], ... i[p], ..., i[q], ...}} expanded form, to the conventional first
* order spin correlations nested list {c, {i[1], ..}, {i[p], ..},{i[q], ...}}
* This function is applicable also when j[k] spin labels are used in defining
* the spin correlation term
*/
Function("InvExpandSp", {X,n,p,q}) [
Local(result, temp, temp1, temp2, temp3);
Set(result, Append({}, Head(X)));
temp:=X[2];
temp1:={};
temp2:={};
temp3:={};
If(n>0, [
temp1:=Take(temp, n);
]);
If(q>0, [
temp3:=Take(temp, -q);
]);
temp2:=Difference(Difference(temp, temp1), temp3);
Set(result, Append(result, temp1));
Set(result, Append(result, temp2));
Set(result, Append(result, temp3));
result;
];

/* The function PutI  replaces the spin j[k] labels with i[k] when such j spin
* labels are used in defining the spin correlation term
*/
Function("PutI", {X,na,nb}) [
Local(k);
n:=0;
For(k:=1, k<=nb, k++) [
If(HasExpr(X,j[k]), [
n++;
x:= Subst(j[k],i[na+n]) X;
]);
];
X;
];
```

Appendix B

```
/* The function SpProd performs the product between two spin correlations,
* X, and Y, of which spin labeles are considered fixed. The output is put in the
* form of a list of spin correlations, which represents the sum of the
* corresponding spin correlatuions terms
*/
Function("SpProd", {X, Y}) [
Local(X2Y2, X2Y3, X2Y4, X3Y2, X3Y3, X3Y4, X4Y2, X4Y3, X4Y4, l, result, K);
Set(X2Y2, Intersection(X[2], Y[2]));
Set(X2Y3, Intersection(X[2], Y[3]));
Set(X2Y4, Intersection(X[2], Y[4]));
Set(X3Y2, Intersection(X[3], Y[2]));
Set(X3Y3, Intersection(X[3], Y[3]));
Set(X3Y4, Intersection(X[3], Y[4]));
Set(X4Y2, Intersection(X[4], Y[2]));
Set(X4Y3, Intersection(X[4], Y[3]));
Set(X4Y4, Intersection(X[4], Y[4]));
Set(l, Length(X2Y3)-Length(X2Y4)-Length(X3Y2)+Length(X3Y4)+Length(X4Y2)-Length(X4Y3));
Set(result, {X[1]*Y[1]*2^(-2*Length(X2Y2)-Length(X2Y3)-Length(X2Y4)-Length(X3Y2)-
2*Length(X3Y3)-Length(X3Y4)-Length(X4Y2)-Length(X4Y3)- 2*Length(X4Y4))*I^(l),
Difference(Concat(X[2],Y[2],X3Y4,X4Y3),Concat(X2Y2,X2Y2,X2Y3,X2Y4,X3Y2,X4Y2)),
Difference(Concat(X[3],Y[3],X2Y4,X4Y2),Concat(X3Y3,X3Y3,X3Y2,X3Y4,X2Y3,X4Y3)),
Difference(Concat(X[4],Y[4],X2Y3,X3Y2),Concat(X4Y4,X4Y4,X4Y2,X4Y3,X2Y4,X3Y4))});
result;
];
```

```
/* The function SpProd performs the product between two spin correlations,
* X, and Y, of which spin labeles are considered fixed. The output is put in the
* form of a list of spin correlations, which represents the sum of the
* corresponding spin correlatuions terms
*/
Function("SpProd", {X, Y}) [
Local(X2Y2, X2Y3, X2Y4, X3Y2, X3Y3, X3Y4, X4Y2, X4Y3, X4Y4, l, result, K);
Set(X2Y2, Intersection(X[2], Y[2]));
Set(X2Y3, Intersection(X[2], Y[3]));
Set(X2Y4, Intersection(X[2], Y[4]));
Set(X3Y2, Intersection(X[3], Y[2]));
Set(X3Y3, Intersection(X[3], Y[3]));
Set(X3Y4, Intersection(X[3], Y[4]));
Set(X4Y2, Intersection(X[4], Y[2]));
Set(X4Y3, Intersection(X[4], Y[3]));
Set(X4Y4, Intersection(X[4], Y[4]));
Set(l, Length(X2Y3)-Length(X2Y4)-Length(X3Y2)+Length(X3Y4)+Length(X4Y2)-Length(X4Y3));
Set(result, {X[1]*Y[1]*2^(-2*Length(X2Y2)-Length(X2Y3)-Length(X2Y4)-Length(X3Y2)-
2*Length(X3Y3)-Length(X3Y4)-Length(X4Y2)-Length(X4Y3)- 2*Length(X4Y4))*I^(l),
Difference(Concat(X[2],Y[2],X3Y4,X4Y3),Concat(X2Y2,X2Y2,X2Y3,X2Y4,X3Y2,X4Y2)),
Difference(Concat(X[3],Y[3],X2Y4,X4Y2),Concat(X3Y3,X3Y3,X3Y2,X3Y4,X2Y3,X4Y3)),
Difference(Concat(X[4],Y[4],X2Y3,X3Y2),Concat(X4Y4,X4Y4,X4Y2,X4Y3,X2Y4,X3Y4))});
result;
];

/* The function OrdSpProd is similar with SpProd except that an additional
* symbolic ordering of resultant spin correlations is performed at the end
* of the computation
*/
Function("OrdSpProd", {X, Y}) [
Local(X2Y2, X2Y3, X2Y4, X3Y2, X3Y3, X3Y4, X4Y2, X4Y3, X4Y4, l, result, K);
Set(X2Y2, Intersection(X[2], Y[2]));
Set(X2Y3, Intersection(X[2], Y[3]));
Set(X2Y4, Intersection(X[2], Y[4]));
Set(X3Y2, Intersection(X[3], Y[2]));
Set(X3Y3, Intersection(X[3], Y[3]));
Set(X3Y4, Intersection(X[3], Y[4]));
Set(X4Y2, Intersection(X[4], Y[2]));
Set(X4Y3, Intersection(X[4], Y[3]));
Set(X4Y4, Intersection(X[4], Y[4]));
Set(l, Length(X2Y3)-Length(X2Y4)-Length(X3Y2)+Length(X3Y4)+Length(X4Y2)-Length(X4Y3));
Set(result, {X[1]*Y[1]*2^(-2*Length(X2Y2)-Length(X2Y3)-Length(X2Y4)-Length(X3Y2)-
2*Length(X3Y3)-Length(X3Y4)-Length(X4Y2)-Length(X4Y3)- 2*Length(X4Y4))*I^(l),
Difference(Concat(X[2],Y[2],X3Y4,X4Y3),Concat(X2Y2,X2Y2,X2Y3,X2Y4,X3Y2,X4Y2)),
Difference(Concat(X[3],Y[3],X2Y4,X4Y2),Concat(X3Y3,X3Y3,X3Y2,X3Y4,X2Y3,X4Y3)),
Difference(Concat(X[4],Y[4],X2Y3,X3Y2),Concat(X4Y4,X4Y4,X4Y2,X4Y3,X2Y4,X3Y4))});
Set(result, SymbOrder(result));
result;
];

/* The function SpComm performs the commutator between two spin correlations,
* X, and Y, of which spin labeles are considered fixed. The output is put in the
* form of a list of spin correlations, which represents the sum of the
* corresponding spin correlatuions terms
*/
Function("SpComm", {X,Y}) [
Local(X2Y2, X2Y3, X2Y4, X3Y2, X3Y3, X3Y4, X4Y2, X4Y3, X4Y4, l, result, K);
Set(X2Y2, Intersection(X[2], Y[2]));
Set(X2Y3, Intersection(X[2], Y[3]));
Set(X2Y4, Intersection(X[2], Y[4]));
Set(X3Y2, Intersection(X[3], Y[2]));
Set(X3Y3, Intersection(X[3], Y[3]));
Set(X3Y4, Intersection(X[3], Y[4]));
Set(X4Y2, Intersection(X[4], Y[2]));
Set(X4Y3, Intersection(X[4], Y[3]));
Set(X4Y4, Intersection(X[4], Y[4]));
Set(l, Length(X2Y3)-Length(X2Y4)-Length(X3Y2)+Length(X3Y4)+Length(X4Y2)-Length(X4Y3));
If(IsEven(l), Set(result, {}), [
Set(result, {X[1]*Y[1]*2^(-2*Length(X2Y2)-Length(X2Y3)-Length(X2Y4)-Length(X3Y2)-
2*Length(X3Y3)-Length(X3Y4)-Length(X4Y2)-Length(X4Y3)- 2*Length(X4Y4)+1)*I^(l),
Difference(Concat(Y[2],X[2],X3Y4,X4Y3),Concat(X2Y2,X2Y2,X2Y3,X2Y4,X3Y2,X4Y2)),
Difference(Concat(Y[3],X[3],X2Y4,X4Y2),Concat(X3Y3,X3Y3,X3Y2,X3Y4,X2Y3,X4Y3)),
```

```
Difference(Concat(Y[4],X[4],X2Y3,X3Y2),Concat(X4Y4,X4Y4,X4Y2,X4Y3,X2Y4,X3Y4))});
]
);
result;
];

/* The function OrdSpComm is similar with SpComm except that an additional
* symbolic ordering of resultant spin correlations is performed at the end
* of the computation
*/
Function("OrdSpComm", {X,Y}) [
Local(X2Y2, X2Y3, X2Y4, X3Y2, X3Y3, X3Y4, X4Y2, X4Y3, X4Y4, l, result, K);
Set(X2Y2, Intersection(X[2], Y[2]));
Set(X2Y3, Intersection(X[2], Y[3]));
Set(X2Y4, Intersection(X[2], Y[4]));
Set(X3Y2, Intersection(X[3], Y[2]));
Set(X3Y3, Intersection(X[3], Y[3]));
Set(X3Y4, Intersection(X[3], Y[4]));
Set(X4Y2, Intersection(X[4], Y[2]));
Set(X4Y3, Intersection(X[4], Y[3]));
Set(X4Y4, Intersection(X[4], Y[4]));
Set(l, Length(X2Y3)-Length(X2Y4)-Length(X3Y2)+Length(X3Y4)+Length(X4Y2)-Length(X4Y3));
If(IsEven(l), Set(result, {}), [
Set(result, {X[1]*Y[1]*2^(-2*Length(X2Y2)-Length(X2Y3)-Length(X2Y4)-Length(X3Y2)-
2*Length(X3Y3)-Length(X3Y4)-Length(X4Y2)-Length(X4Y3)- 2*Length(X4Y4)+1)*I^(l),
Difference(Concat(Y[2],X[2],X3Y4,X4Y3),Concat(X2Y2,X2Y2,X2Y3,X2Y4,X3Y2,X4Y2)),
Difference(Concat(Y[3],X[3],X2Y4,X4Y2),Concat(X3Y3,X3Y3,X3Y2,X3Y4,X2Y3,X4Y3)),
Difference(Concat(Y[4],X[4],X2Y3,X3Y2),Concat(X4Y4,X4Y4,X4Y2,X4Y3,X2Y4,X3Y4))});
]
);
If(Not(Equals(result,{})), Set(result,SymbOrder(result)) );
result;
];

/* The function SumSpProd performs the product between two sums of
* spin correlations X, and Y. The output is also put in the form of sum of spin
correlations
*/
Function("SumSpProd", {X,Y}) [
Local(i, j, result);
Set(result, {});
ForEach(i, X) [
        ForEach(j, Y) [
        Set(result,DestructiveAppend(result,SpProd(i,j)));
        ];
];
Set(result,ReduceSumSpCorr(result));
result;
];

/* The function OrdSumSpProd is similar with SumSpProd except that an additional
* symbolic ordering of resultant spin correlations is performed at the end
* of the computation
*/
Function("OrdSumSpProd", {X,Y}) [
Local(i, j, result);
Set(result, {});
ForEach(i, X) [
        ForEach(j, Y) [
        Set(result,DestructiveAppend(result,OrdSpProd(i,j)));
        ];
];
Set(result,ReduceSumSpCorr(result));
result;
];

/* The function SumSpComm performs the commutator between two sums of
* spin correlations X, and Y. The output is also put in the form of sum
* of spin correlations
*/
Function("SumSpComm", {X,Y}) [
Local(i, j, temp, result);
Set(result, {});
ForEach(i, X) [
        ForEach(j, Y) [
        Set(temp, SpComm(i,j));
        If(Not(Equals(temp,{})), Set(result,DestructiveAppend(result,temp)) );
```

```
        ];
];
Set(result,ReduceSumSpCorr(result));
result;
];

/* The function OrdSumSpComm is similar with SumSpComm except that an additional
* symbolic ordering of resultant spin correlations is performed at the end
* of the computation
*/
Function("OrdSumSpComm", {X,Y}) [
Local(i, j, temp, result);
Set(result, {});
ForEach(i, X) [
        ForEach(j, Y) [
        Set(temp, OrdSpComm(i,j));
        If(Not(Equals(temp,{})), Set(result,DestructiveAppend(result,temp)) );
        ];
];
Set(result,ReduceSumSpCorr(result));
result;
];

/* The function CommonSpOp determines the all possible common spin operators
* within the lists X and Y: They are represented in the form X = {i[1], i[2],
* ... , i[ma]}, and Y = {Coeff(j[1], j[2], ...j[mb+p]), {j[1], j[2], ... j[mb]}}
* This is an auxiliary function needed in the functions GenSpProd(Comm))
*/
Function("CommonSpOp", {x,y}) [
Local(temp, l, k, ma, z,  result);
Set(result, {});
If(Not(HasExpr(Tail(y),j)) And Length(Tail(y)[1])>1, Set(result, {{{}, y}}));
ma:=Length(x);
z:=x;
For(k:=1, k<=ma, k++) [
temp:=x[k];
z:=Difference(z, {temp});
ForEach(l, Tail(y)[1]) [
         If(HasExpr(l,j), DestructiveAppend(result, Concat({z}, {Subst(l,temp) y })));
        ];
];
result;
];

/* The function CommonCoeff determines the all possible common spin labels,
* j[mb+1], j[mb+2],..., j[mb+p] in the coefficient of Y = {Coeff(j[1], j[2], ...
* j[mb+p]), {j[1], j[2], ... j[mb]}} with the i spin labels within X = {i[1],
* i[2], ... , i[ma]}. This is an auxiliary function needed in the functions
* GenSpProd(Comm))
*/
Function("CommonCoeff", {x,y,mb,p}) [
Local(temp, l, k, u, v, n, L, y1, coeff, coefft, result);
coeff:=0;
temp:=x;
For(l:=1, l<=p, l++) [
Set(temp, Append(temp, j));
];
Set(temp, Permutations(temp));
L:=Length(temp);
For(k:=1, k<=L, k++) [
DestructiveReplace(temp,k, Take(temp[k],p));
];
Set(temp, RemoveDuplicates(temp));
ForEach(v, temp) [
While(Find(v,j)>0) [
DestructiveReplace(v,Find(v,j), j[mb+Find(v,j)]);
];
];
Set(temp, RemoveDuplicates(temp));
ForEach(u, temp) [
coefft:=y[1];
For(n:=1, n<=p, n++) [
Set(coefft, Subst(j[mb+n],u[n])coefft);
];
Set(coeff, coeff+coefft);
];
y1:=Replace(y,1, coeff);
```

```
Set(result, {x, y1} );
result;
];

/* The function GenSpProd performs the product between two general spin
*  correlations, X, and Y, of which spin labeles are running labels. The output
*  is put in the form of a list of spin correlations, which represents the sum
*  of the corresponding spin correlatuions terms
*/
Function("GenSpProd", {x,y}) [
Local(temp, tempi, tempf, tempf1, tempf2, u, k, p, q, l, ma, mb, nx, ny, nz, result1,
result);
nx:=Length(y[2]);
ny:=Length(y[3]);
nz:=Length(y[4]);
Set(mb, NoSpOp(y));
Set(p,NoSpins(y)-mb);
y:=Subst(i,j) y;
Set(tempi, {{ExtrSp(x), ExpandSp(y)}});
Set(ma, Length(x[2])+Length(x[3])+Length(x[4]));
Set(result1,{{ExtrSp(x), ExpandSp(y)}});
Set(result2,{});
For(k:=1, k<=ma, k++) [
Set(tempf,{});
For(q:=1, q<=Length(tempi), q++) [
If(Length(tempi[q][1])>0, Set(tempf, Concat(tempf,
CommonSpOp(tempi[q][1],tempi[q][2]))));
 ];
Set(tempi,tempf);
If(p>0, [
For(u:=1, u<=Length(tempf), u++) [
tempf1:= {CommonCoeff(Difference(ExtrSp(x), MakeI(tempf[u][2][2],ma)), tempf[u][2],
mb, p)};
Set(result1, Concat(result1,tempf1));
];
],
Set(result1, Concat(result1,tempf)));
];
ForEach(l, result1) [
        Set(result2, Append(result2, InvExpandSp(PutI(l[2], ma, mb+p), nx, ny, nz)));
       ];
result:=OrdSumSpProd({x}, result2);
result;
];

/* The function GenSpComm performs the commutator between two general spin
*  correlations, X, and Y, of which spin labeles are running labels. The output
*  is put in the form of a list of spin correlations, which represents the sum
*  of the corresponding spin correlatuions terms
*/
Function("GenSpComm", {x,y}) [
Local(temp, tempi, tempf, tempf1, tempf2, u, k, p, q, l, ma, mb, nx, ny, nz, result1,
result);
nx:=Length(y[2]);
ny:=Length(y[3]);
nz:=Length(y[4]);
Set(mb, NoSpOp(y));
Set(p,NoSpins(y)-mb);
y:=Subst(i,j) y;
Set(tempi, {{ExtrSp(x), ExpandSp(y)}});
Set(ma, Length(x[2])+Length(x[3])+Length(x[4]));
Set(result1,{{ExtrSp(x), ExpandSp(y)}});
Set(result2,{});
For(k:=1, k<=ma, k++) [
Set(tempf,{});
For(q:=1, q<=Length(tempi), q++) [
If(Length(tempi[q][1])>0, Set(tempf, Concat(tempf,
CommonSpOp(tempi[q][1],tempi[q][2]))));
 ];
Set(tempi,tempf);
If(p>0, [
For(u:=1, u<=Length(tempf), u++) [
tempf1:= {CommonCoeff(Difference(ExtrSp(x), MakeI(tempf[u][2][2],ma)), tempf[u][2],
mb, p)};
Set(result1, Concat(result1,tempf1));
];
],
```

```
Set(result1, Concat(result1,tempf)));
];
ForEach(l, result1) [
         Set(result2, Append(result2, InvExpandSp(PutI(l[2], ma, mb+p), nx, ny, nz)));
        ];
result:=OrdSumSpComm({x}, result2);
result;
];

/* The function GenSumSpProd performs the commutator between two sums of
* the general spin correlations X, and Y. The output is also put in the form of
* sum of spin correlations
*/
Function("GenSumSpProd", {x,y}) [
Local(i, j, temp, result);
Set(result, {});
ForEach(i, x) [
        ForEach(j, y) [
        Set(temp, GenSpProd(i,j));
        If(Not(Equals(temp,{})), Set(result,Concat(result,temp)) );
        ];
];
Set(result,ReduceSumSpCorr(result));
result;
];

/* The function GenSumSpComm performs the commutator between two sums of
* the general spin correlations X, and Y. The output is also put in the form of
* sum of spin correlations
*/
Function("GenSumSpComm", {x,y}) [
Local(i, j, temp, result);
Set(result, {});
ForEach(i, x) [
        ForEach(j, y) [
        Set(temp, GenSpComm(i,j));
        If(Not(Equals(temp,{})), Set(result,Concat(result,temp)) );
        ];
];
Set(result,ReduceSumSpCorr(result));
result;
];


/* The function ScalarSpProd performs the scalar product (Trace) of a
* product of spin correlation, X and Y in an N-spin system.
*/
Function("ScalarSpProd", {X,Y,N}) [
Local(result);
If(Equals(Tail(X), Tail(Y)), Set(result, Head(SpProd(X,Y))*2^N), Set(result, 0));
result;
];

/* The function ScalarSumSpProd performs the scalar product (Trace) of a
* product between the spin correlation sums , X and Y, in an N-spin system.
*/
Function("ScalarSumSpProd", {X,Y,N}) [
Local(result, I, j);
Set(result, 0);
ForEach(i, X) [
ForEach(j, Y) [
Set(result, result + ScalarSpProd(i, j, N));
];
];
result;
];
```

Appendix C

```
/* The script "Dens_Op_Obs.ys" computes the approximate symbolic expression of
* the density operator rho0, and the expectation value of the observable, Obs,
* during the evolution under a two-spin interaction Hamiltonian, HInt
*/


/* ReadInp reads the initial density operator, rho0 and the spin interaction
*  Hamiltonian, HInt, from the file "input.txt"
*/
Function("ReadInp", {}) [
Load("input.txt");
];

/* Rules for symmetrizing the interaction coefficinets, D(u,v)=D(v,u)
*/
RuleBase("D1", {u,v});
Rule("D1", 2, 10,  Index(u) > Index(v)) D1(v,u);
RuleBase("D2", {u,v});
Rule("D2", 2, 10,  Index(u) > Index(v)) D2(v,u);
RuleBase("D3", {u,v});
Rule("D3", 2, 10,  Index(u) > Index(v)) D3(v,u);


/* RhoApprox evaluates the nth order (n = 1, ...) term in the power expansion of
*  the density operator, which is stored in the file "rhon.txt"
*/
Function("RhoApprox", {n}) [
Local(c);
ReadInp();
If(Equals(n,1),
[
Set(c,Simplify(GenSumSpComm(HInt,rho0)));
ToFile(ConcatStrings("rho", String(n), ".txt")) Write(c, ;);
],
[
Set(c,FromFile(ConcatStrings("rho", String(n-1), ".txt")) Read());
Set(c,Simplify(GenSumSpComm(HInt,c)));
ToFile(ConcatStrings("rho", String(n), ".txt")) Write(c, ;);
]
);
];


/* Observable calculates the nth order (n = 1, ...) term of the expectation
*  value of the observable operator Obs; its expression is extracted from the
*  file "input.txt", whereas the output is stored in the file "Obs_n.txt"
*/
Function("Observable", {N, n, Obs, t}) [
Local(temp, coeff0, coeff);
Set(temp, FromFile(ConcatStrings("rho", String(n), ".txt")) Read());
Set(coeff0, ScalarSpProd(Obs,rho0[1],N));
Set(coeff, ((I*t)^n/n!)*ScalarSumSpProd(Obs,temp,N));
ToFile(ConcatStrings("Obs",  "_", String(n), ".txt"))
Write(Simplify(Simplify(coeff/(coeff0*(2^n))), ;);
];



/* The input file, "input.txt" contining the SD-CAS expressions of the initial
*  density operator and the spin interaction Hamiltonian
*/
rho0 := {{1, {i[1]}, {}, {}}};
HInt := {{-D(i[1],i[2]), {i[1], i[2]}, {}, {}}, {-D(i[1],i[2]), {}, {i[1], i[2]}, {}},
{2*D(i[1],i[2]), {}, {}, {i[1], i[2]}} };
Obs := {1, {i[1]}, {}, {}};
```

References

[1] K. Wuthrich, NMR studies of structure and function of biological macromolecules (Nobel Lecture), Angew. Chem. Int. Ed. 42 (2003) 3340–3363.
[2] T.F. Havel, D.G. Cory, S. Lloyd, N. Boulant, E.M. Fortunato, M.A. Pravia, G. Teklemariam, Y.S. Weinstein, A. Bhattacharyya, J. Hou, Quantum information processing by nuclear magnetic spectroscopy, Am. J. Phys. 70 (2002) 345–362.
[3] A. Abragam, The Principles of Nuclear Magnetism, Clarendon Press, Oxford, 1961.
[4] C.P. Slichter, Principles of Magnetic Resonance, Harper and Row, New York, 1963.
[5] M. Goldman, Spin Temperature and Nuclear Magnetic Resonances in Solids, Clarendon Press, Oxford, 1970.
[6] U. Haeberlen, High-Resolution NMR in Solids: Selective Averaging, Academic Press, New York, 1976.
[7] M. Mehring, Principles of High-Resolution NMR in Solids, second revised and enlarged ed., Springer Verlag, 1983.
[8] R.R. Ernst, G. Bodenhausen, A. Wokaun, Principles of Nuclear Magnetic Resonance in One and Two Dimensions, Oxford University Press, Oxford, 1987.
[9] M. Munowitz, Coherence and NMR, Wiley-Interscience, 1988.
[10] M. Goldman, Quantum Description of High-Resolution NMR in Liquids, Oxford University Press, Oxford, 1991.
[11] R. Freeman, Spin Choreography: Basic Steps in High-Resolution NMR, Oxford University Press, Oxford, 1998.
[12] M.H. Levitt, Spin Dynamics: Basics of Nuclear Magnetic Resonance, second ed., Wiley, 2008.
[13] R.J. Wittebort, E.T. Olejniczak, R.G. Griffin, Analysis of deuterium nuclear magnetic resonance line shapes in anisotropic media, J. Chem. Phys. 86 (1987) 5411–5420.
[14] J. Herzfeld, A. Berger, Sidebands intensities in NMR spectra of sample spinning at the magic angle, J. Chem. Phys. 73 (1980) 6021–6030.
[15] M.H. Levitt, D.P. Raleigh, F. Creuzet, R.G. Griffin, Theory and simulations of homonuclear spin pair systems in rotating solids, J. Chem. Phys. 92 (1990) 6347–6365.
[16] M. Bak, J.T. Rasmusen, N.C. Nielsen, SIMPSON: a general simulation program for solid-state NMR spectroscopy, J. Magn. Reson. 147 (2000) 296–330.
[17] M. Veshtort, R.G. Griffin, SPINEVOLUTION: a powerful tool for the simulation of solid and liquid state NMR experiments, J. Magn. Reson. 178 (2006) 248–282.
[18] 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. 106A (1994) 75–105.
[19] W.B. Blanton, BlochLib: a fast NMR C++ tool kit, J. Magn. Reson. 162 (2003) 269–283.
[20] P. Hodgkinson, L. Emsley, Numerical simulations of solid-state NMR experiments, Prog. Nucl. Magn. Reson. Spectrosc. 36 (2000) 201–239.
[21] M. Edén, Computer simulations in solid-state NMR. II. Implementations for static and rotating samples, Concepts Magn. Reson. Part A 17A (2003) 117–154.
[22] P. Hodgkinson, D. Sakellarious, L. Emsley, Simulation of extended periodic systems of nuclear spins, Chem. Phys. Lett. 326 (2000) 515–522.
[23] M.C. Butler, J.-N. Dumez, L. Emsley, Dynamics of large nuclear spin-systems from low-order correlations in Liouville space, Chem. Phys. Lett. 477 (2009) 377–381.
[24] M.H. Levitt, M. Edén, Numerical simulation of periodic nuclear magnetic resonance problems: fast calculation of carousel averages, Mol. Phys. 95 (1998) 879–890.
[25] M. Hohwy, H. Bildsoe, H.J. Jakobsen, N.C. Nielsen, Efficient spectral simulations in NMR of rotating solids. The γ-COMPUTE algorithm, J. Magn. Reson. 136 (1999) 6–14.
[26] T.T. Nakashima, R.E.D. McClung, Simulations of two-dimensional NMR spectra using product operators in the spherical basis, J. Magn. Reson. 70 (1986) 187–203.
[27] J.W. Shriver, NMR product operator calculations in Mathematica, J. Magn. Reson. 94 (1991) 612–616.
[28] P. Guntert, N. Schaefer, G. Otting, K. Wutrich, POMA: a complete Mathematica implementation of the NMR product operator formalism, J. Magn. Reson. 101 A (1993) 103–105.
[29] R.P.F. Kanters, B.W. Char, A.W. Addison, A computer algebra application for the description of NMR experiments using the product operator formalism, J. Magn. Reson. 101 A (1993) 23–29.
[30] I. Rodriguez, J. Ruiz-Cabello, Density matrix calculations in Mathematica, Concepts Magn. Reson. 13 (2002) 143–147.
[31] A. Gencten, I. Saka, A complete product operator theory for IS (I = 1/2, S = 1) spin system and application to DEPT-HMQC NMR experiment, Mol. Phys. 104 (2006) 2983–2989.
[32] A. Jerschow, Math NMR: spin and spatial tensor manipulations in Mathematica, J. Magn. Reson. 176 (2005) 7–14.
[33] I. Kuprow, N. Wagner-Rundell, P.J. Hore, Bloch–Redfield–Wangsness theory engine implementation using symbolic processing software, J. Magn. Reson. 184 (2007) 196–206.
[34] C.K. Anand, A.D. Bain, Z. Nie, Simulation of steady-state NMR of coupled systems using Liouville space and computer algebra methods, J. Magn. Reson. 189 (2007) 200–208.
[35] C.N. Banwell, H. Primas, On the analysis of high-resolution nuclear magnetic resonance spectra. I. Methods of calculating NMR spectra, Mol. Phys. 6 (1963) 225–256.
[36] O.W. Sorensen, G.W. Eich, M.H. Levitt, G. Bodenhousen, R.R. Ernst, Product operator formalism for the description of NMR pulse experiments, Prog. Nucl. Magn. Reson. 16 (1983) 163–192.
[37] M. Maricq, J.S. Waugh, NMR in rotating solids, J. Chem. Phys. 70 (1979) 3300–3316.
[38] yacas: <http://yacas.sourceforge.net>.
[39] J. Baum, M. Munowitz, A.N. Garroway, A. Pines, Multiple-quantum dynamics in solid state NMR, J. Chem. Phys. 83 (1985) 2015–2025.
[40] M. Munowitz, A. Pines, Principles and applications of multiple-quantum NMR, Adv. Chem. Phys. 66 (1987) 1–152.
[41] M. Munowitz, M. Mehring, Whither the free induction: an analysis of multiple-spin dynamics in strongly coupled systems, Chem. Phys. 116 (1987) 79–90.
[42] S. Lacelle, On the growth of multiple spin coherences in NMR of solids, Adv. Magn. Opt. Reson. 16 (1991) 173–263.