ARTICLE IN PRESS

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Journal of Magnetic Resonance 178 (2006) 248–282

JMR Journal of Magnetic Resonance

www.elsevier.com/locate/jmr

SPINEVOLUTION: A powerful tool for the simulation of solid and liquid state NMR experiments

Mikhail Veshtort, Robert G. Griffin *

Department of Chemistry, MIT/Harvard Center for Magnetic Resonance, Francis Bitter Magnet Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Received 3 May 2005; revised 22 July 2005

Abstract

Exact numerical simulations of NMR experiments are often required for the development of new techniques and for the extraction of structural and dynamic information from the spectra. Simulations of solid-state magic angle spinning (MAS) experiments can be particularly demanding both computationally and in terms of the programming required to carry them out, even if special simulation software is used. We recently developed a number of approaches that dramatically improve the efficiency and allow a high degree of automation of these computations. In the present paper, we describe SPINEVOLUTION, a highly optimized computer program that implements the new methodology. The algorithms used in the program will be described separately. Although particularly efficient for the simulation of experiments with complex pulse sequences and multi-spin systems in solids, SPINEVOLUTION is a versatile and easy to use tool for the simulation and optimization of virtually any NMR experiment. The performance of SPINEVOLUTION was compared with that of another recently developed NMR simulation package, SIMPSON. Benchmarked on a series of examples, SPINEVOLUTION was consistently found to be orders of magnitude faster. At the time of publication, the program is available *gratis* for non-commercial use.

Keywords: NMR simulations; Virtual NMR spectrometer; Software; Parameter optimization; Data fitting

1. Introduction

With the continued increase in the complexity of NMR techniques and applications, numerical simulations have become an indispensable part of modern day research. They are now commonly required for the extraction of structural parameters from the data, design of new experiments, and theoretical research. This is especially true for solid-state NMR, where most applications employ magic angle spinning (MAS) and ^{1}H-decoupling to obtain high resolution spectra, while various recoupling methods are used to restore the averaged dipolar and chemical shift anisotropy (CSA) interactions [1,2]. These techniques normally create a homogeneous time-dependent Hamiltonian [3], requiring the use of numerical methods whenever the experiments have to be simulated exactly. Typically, such experiments are designed and interpreted in terms of the first few orders of the average Hamiltonian theory (AHT) [4], which can often be used to obtain approximate solutions. However, while being a powerful analytical tool, AHT is not always applicable or sufficiently accurate. Furthermore, even in the experiments where this formalism provides a reasonably accurate analytic description of the two-spin systems, it remains necessary to invoke numerical methods for most cases involving multiple spins. In solution-state NMR, analytic descriptions of experiments are much more common. However, numerical simulations are still important, for example, when studying non-ideal pulses (including shaped pulses [5]), various

* Corresponding author.
E-mail address: rgg@mit.edu (R.G. Griffin).

1090-7807/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmr.2005.07.018

effects due to strong coupling and relaxation, and even single-scan 2D experiments [6,7].

Driven by demands of the experimental NMR, numerous computer programs were developed over the past four decades for the simulation of NMR experiments [8–41]. Most of this software is easily assigned to one of the following categories: *analytic NMR tools* (typically implemented in Mathematica), *specialized application programs* (e.g., a NOESY spectra simulator), *application programming interfaces* (*API*'s) (providing building blocks for the development of application programs), *general solution-state NMR simulation programs*, and *general NMR simulation programs*. The two later types are called general in the sense that they can be used for the simulation of experiments with a wide variety of pulse sequences and spin systems, and thus have the functionality of a virtual NMR spectrometer (in addition to any other functionalities they may have). Additionally, as opposed to an API, a simulation program must have an abstract, high-level interface, which should not require any programming, in general.

In spite of all these developments, only a few instances of the currently available software are suitable for the simulation of general NMR experiments. As a consequence, until a few years ago, the design of new techniques, particularly in the solid-state NMR, had to be often combined with the development of new problem-specific software [23,25,42–45]. For the solid-state NMR, this had been more than an inconvenience: it seems that the development of the whole field had been held back due to the lack of an appropriate general simulation tool. This rather bold statement is evidenced by the recent popularity enjoyed by SIMPSON [39], and by the vast interest expressed in such a tool in the community.

To the best of our knowledge, the first reasonably general NMR simulation software was ANTIOPE [27]. The program is still maintained and distributed by its author (J.S. Waugh) and has recently acquired a particularly user-friendly interface and undergone other significant changes since its initial publication. An API-type package GAMMA [31], which appeared soon after ANTIOPE, is designed as a library of C++ classes and methods intended as building blocks for the construction of various problem-specific simulation programs. Although the development of these programs is much facilitated by GAMMA, a significant amount of programming is often required to perform a simulation. The package has been extensively used over the years for the simulation of both NMR and EPR experiments. A similar C++ API (BlochLib [40]) has been developed recently that relies on efficient numerical libraries for its core calculations and is superior to GAMMA in terms of efficiency. Currently the most popular in the solid-state NMR community simulation package is SIMPSON [39]. It is designed as a high-level API, where the simulation is driven by an input file written in the scripting language Tcl. SIMPSON requires no compilation and much less programming to perform a simulation. The major application of the program is the simulation of MAS experiments. Overall, it provides a reasonable compromise between efficiency, convenience, and versatility.

The aforementioned three qualities are probably the most important characteristics desired from a general NMR simulation program. Developing software that is not deficient in any of these areas has been a great challenge so far. Indeed, versatility can be achieved in the most straightforward manner by taking the API approach of GAMMA, BlochLib, and, to a lesser degree, SIMPSON. However, an interface that requires writing C++ code for a sophisticated API platform is unlikely to be convenient in most cases. Furthermore, to efficiently propagate a density matrix through a complex MAS experiment with multiple pulse sequences, one has to compute and recycle an elaborate collection of propagators. There is very little support provided to this end in GAMMA, BlochLib, or SIMPSON. As a result, using the most efficient algorithm for such a simulation would be either impossible due to restrictions of the API, or highly challenging due to the substantial amount of expertise and programming required. In addition, the efficiency of any simulation depends also on the efficiency of the computational techniques and algorithms that the simulation package relies upon internally, and which are beyond the user's control. If these techniques comprise only a rather basic set, creation of a simple interface to such a package is relatively straightforward, as exemplified by some older solution-state programs. However, if these techniques comprise an extensive library of highly efficient and often problem-specific algorithms, then their integration into a unified general NMR simulation program with a user-friendly interface presents a significant challenge, which is in addition to the challenge of developing these methods and creating such a library.

In an attempt to solve this problem, we developed SPINEVOLUTION, a general NMR simulation program that implements a number of novel computational techniques and methodological approaches to this end. Although specifically tailored for the simulation of solid-state MAS experiments, SPINEVOLUTION can also be used in the context of solution state NMR, particularly to solve various optimization problems. The program features a friendly, text file based interface, where the pulse sequences are specified in terms of the "canonical representation." The representation is a natural, non-algorithmic description of NMR experiments that captures their periodic structure in a form easily explored for the construction of efficient propagation schemes.

The computational technique largely responsible for the high efficiency of multi-spin computations in SPINEVOLUTION relies on the Chebyshev expansion for the computation of the propagators and thus is able to exploit the sparsity of the Hamiltonian. This algorithm behaves asymptotically as $O(N^2 4^N)$, where N is the number of spins in the system, while the traditional algorithm, which relies on the diagonalization of the Hamiltonian, behaves as $O(8^N)$. Another technique, the g-COMPUTE algorithm, which generalizes the previously known COMPUTE [46] and γ-COMPUTE [47–49] algorithms, combines efficient propagation in the frequency domain with the powder averaging over the γ angle. It extends the applicability of the γ-COMPUTE algorithm to multidimensional experiments, unlimited spectral widths, and much less restrictive conditions for the rotor-synchronization of the pulse sequence. The details of these and other algorithms employed in the program will be discussed elsewhere.

In the present publication, we discuss the methodology, functionality, interface, and performance of SPINEVOLUTION. At the time of publication, the current version of the program is 2.8. In this version, some of the program features described below are not functional yet. However, the work is under way to implement these, as well as a number of other features. Currently, SPINEVOLUTION is available *gratis* for non-commercial purposes. The program compiled for various platforms (Linux, OS X, Irix, Windows, and Solaris) and the examples described in this paper, as well as other examples and materials can be downloaded at the following URL http://web.mit.edu/fbml/cmr/griffin-group/SPINEVOLUTION/.

2. NMR Hamiltonian

Although the NMR literature abounds with excellent discussions of the spin Hamiltonian [50–56], no single treatment contains all the details required for our purposes. Therefore, we feel that it is important to provide the reader with a coherent reference on all relevant theoretical and experimental aspects of the NMR Hamiltonian, spelling out the conventions and definitions employed to this end in SPINEVOLUTION. As mentioned above, we are presently concerned only with *what* is computed by the program, as opposed to *how* these computations are performed internally.

NMR interactions are best defined in the form they enter into the full laboratory frame (Schrödinger representation) spin Hamiltonian

$$H^{\mathrm{L}} = H_{\mathrm{Z}} + H^{\mathrm{L}}_{\mathrm{RF}} + H^{\mathrm{L}}_{\mathrm{D}} + H^{\mathrm{L}}_{\mathrm{J}} + H^{\mathrm{L}}_{\mathrm{Q}}. \tag{1}$$

The interaction of the spins with the static magnetic field is given by the Zeeman term,

$$H_{\mathrm{Z}} = -\sum_i \gamma_{[i]}\, \mathbf{B}^i \cdot \mathbf{I}_i = \sum_i \mathbf{n}_{B_0} \cdot \omega^{[i]}_{\mathrm{ref}} (1 + \boldsymbol{\delta}^i) \cdot \mathbf{I}_i, \tag{2}$$

where the summation is over all nuclei in the system, and $\mathbf{I}_i$, $\gamma_{[i]}$, and $\boldsymbol{\delta}^i$ are, respectively, the angular momentum operator, the gyromagnetic ratio, and the chemical shift tensor of the nucleus i; $\omega^{[i]}_{\mathrm{ref}}$ is the frequency of the reference compound, $\mathbf{B}^i$ is the static magnetic field at the site of the nucleus, and $\mathbf{n}_{B_0}$ is the unit vector in the direction of the static magnetic field $\mathbf{B}_0$. The symbol $[i]$ means here "the species of the nucleus i", so that γ_k and ω^k_{ref} denote the gyromagnetic ratio and the frequency of the reference for the nuclear species k.

For most purposes other than the tabulation of the chemical shifts, it is usually more convenient to use a reference frequency that is not tied to any chemical compound or spectrometer hardware, but can be set instead to any desired value (usually near the Larmor frequency). This frequency will not enter into the final expression for the Hamiltonian and thus cannot affect the results of any computations. However, it will serve as a flexible common reference point for the offsets of all other relevant frequencies. We will denote this frequency as ω^k_*. Assuming that the frequency of the reference compound is given by the offset $\Delta\omega^k_{\mathrm{ref}}$,

$$\omega^k_{\mathrm{ref}} = \omega^k_* + \Delta\omega^k_{\mathrm{ref}} \tag{3}$$

the Zeeman term can be conveniently divided into two parts:

$$H_{\mathrm{Z}} = H'_{\mathrm{Z}} + H^{\mathrm{L}}_{\mathrm{CS}}, \tag{4}$$

where:

$$H'_{\mathrm{Z}} = \sum_k \omega^k_* \mathbf{F}_k \cdot \mathbf{n}_{B_0}, \tag{5}$$

$$H^{\mathrm{L}}_{\mathrm{CS}} = \sum_i \mathbf{n}_{B_0} \cdot \boldsymbol{\Omega}^i \cdot \mathbf{I}_i, \tag{6}$$

and

$$\boldsymbol{\Omega}^i = \omega^{[i]}_*\, \boldsymbol{\delta}^i + \Delta\omega^{[i]}_{\mathrm{ref}}. \tag{7}$$

The summation in Eq. (5) is over all nuclear species and $\mathbf{F}_k$ is the total angular momentum for all spins of the species k. Note that all angular frequencies are *signed* quantities in our notation. Thus, for example, for γ-positive nuclei, $\boldsymbol{\Omega}^i$ and $\boldsymbol{\delta}^i$ have opposite signs (if $\Delta\omega^k_{\mathrm{ref}} = 0$) since ω^k_* is negative. If pulsed field gradients (PFG's) are used in the experiment, $\Delta\omega^k_{\mathrm{ref}}$ becomes a function of time and position in the sample. For a linear gradient along $\mathbf{z}$, for example, the function is

$$\Delta\omega^k_{\mathrm{ref}}(z,t) = \Delta\omega^k_{\mathrm{ref},0} + z\gamma_k G_k(t). \tag{8}$$

The radio-frequency (RF) field term can be usually represented as

$$H^{\mathrm{L}}_{\mathrm{RF}} = -\sum_k \gamma_k B^k_1(t) \cos\left(\left|\omega^k_*\right| t + \int_0^t \tilde{\omega}^k_{\mathrm{off}}(t')\,\mathrm{d}t' + \tilde{\phi}_k(t)\right) \mathbf{F}_k \cdot \mathbf{n}_{B_1} \tag{9}$$

where the summation is over all spectrometer channels (one per each nuclear species), $B_1^k(t) > 0$ is the instantaneous $\mathbf{B}_1$ amplitude, $\tilde{\omega}_{\text{off}}^k(t)$ is the frequency offset, $\tilde{\phi}_k(t)$ is the phase of the RF field, and $\mathbf{n}_{B_1}$ is the unit vector in the direction of $\mathbf{B}_1$ in the RF coil. By projecting $\mathbf{B}_1$ onto the plane perpendicular to the main magnetic field and representing the resulting projection as the sum of two circularly polarized RF components for each channel, one can separate the resonant part of H_{RF}^{L} from the nonresonant. Assuming that the vector $\mathbf{n}_{B_1}$ lies in the **xz** plane at the angle θ_{B_1} with the **x** axis (and **z** is in the direction of $\mathbf{B}_0$), the following expression is obtained:

$$H_{\text{RF}}^{\text{L}} = \sum_k \left|\omega_{\text{RF}}^k(t)\right| e^{-i\int_0^t dt' \left(\omega_*^k + \omega_{\text{off}}^k(t')\right) F_{kz}} \{F_{kx}\cos\phi_k(t) + F_{ky}\sin\phi_k(t)\} e^{i\int_0^t dt' \left(\omega_*^k + \omega_{\text{off}}^k(t')\right) F_{kz}} + H_{\text{RF,nonres}}^{\text{L}}, \quad (10)$$

where

$$\omega_{\text{RF}}^k(t) = -\frac{1}{2}\gamma_k B_1^k(t)\cos\theta_{B_1} \quad (11)$$

and

$$\omega_{\text{off}}^k(t) = -(\text{sign }\gamma_k)\tilde{\omega}_{\text{off}}^k(t) \quad (12a)$$

$$\phi_k(t) = -(\text{sign }\gamma_k)\tilde{\phi}_k(t) \quad (12b)$$

The dependence of the nutation frequency on the sign of γ_k was removed in Eq. (10) by using a coordinate system, which is rotated by 180° about the **z** axis when $\gamma_k > 0$. The nonresonant terms, $H_{\text{RF,nonres}}^{\text{L}}$, will be neglected since their effects (mainly the Bloch-Siegert shift) are usually insignificant. Note that while the spectrometer hardware generates the frequency offsets $\tilde{\omega}_{\text{off}}^k(t)$ and phases $\tilde{\phi}_k(t)$, it is the *sign-corrected* frequency offsets $\omega_{\text{off}}^k(t)$ and phases $\phi_k(t)$ that are "seen" by the nuclei.

The dipolar and the *J*-couplings are given by

$$H_{\text{D}}^{\text{L}} = \sum_{i<j} \mathbf{I}_i \cdot \mathbf{D}^{ij} \cdot \mathbf{I}_j \quad (13)$$

and

$$H_{\text{J}}^{\text{L}} = 2\pi \sum_{i<j} \mathbf{I}_i \cdot \mathbf{J}^{ij} \cdot \mathbf{I}_j, \quad (14)$$

where $\mathbf{D}^{ij}$ and $\mathbf{J}^{ij}$ are the dipolar and *J*-coupling tensors. The principal components of the dipolar coupling tensor are $D_{XX}^{ij} = D_{YY}^{ij} = -b_{ij}$ and $D_{ZZ}^{ij} = 2b_{ij}$, where b_{ij} is the dipolar coupling constant,

$$b_{ij} = -\frac{\mu_0\hbar}{4\pi}\frac{\gamma_{[i]}\gamma_{[j]}}{r_{ij}^3}. \quad (15)$$

The quadrupolar coupling[1] is given by

$$H_{\text{Q}}^{\text{L}} = \sum_i \frac{eQ_i}{2I_i(2I_i-1)\hbar}\mathbf{I}_i \cdot \mathbf{V}^i \cdot \mathbf{I}_i = 2\pi \sum_i \frac{\chi_i}{2I_i(2I_i-1)}\mathbf{I}_i \cdot \tilde{\mathbf{V}}^i \cdot \mathbf{I}_i, \quad (16)$$

where Q_i and I_i are the quadrupole moment and the nuclear spin quantum number of nucleus i, and $\mathbf{V}^i$ is the electric field gradient tensor at the site of the nucleus, $V_{\alpha\beta}^i = \left.\frac{\partial^2 V^i}{\partial r_\alpha \partial r_\beta}\right|_{r=0}$. In the second form of this equation, $\chi_i = e^2 Q_i q_i/h$ is the quadrupolar coupling constant, and $\tilde{\mathbf{V}}^i = \mathbf{V}^i/eq_i$ is the reduced electric field gradient tensor ($eq_i = V_{ZZ}^i$ is the largest principal component of $\mathbf{V}^i$).

The interactions of the internal Hamiltonian depend on the symmetric[2] second-rank tensors $\mathbf{\Omega}^i$, $\mathbf{D}^{ij}$, $\mathbf{J}^{ij}$, and $\mathbf{V}^i$. The following convention is commonly used to characterize any such tensor **A**. The principal axes of **A** are labeled to satisfy the following conditions in these axes: $|A_{ZZ} - A_{\text{iso}}| \geqslant |A_{XX} - A_{\text{iso}}| \geqslant |A_{YY} - A_{\text{iso}}|$, where A_{iso} is the *isotropic value* of **A**,

$$A_{\text{iso}} = \frac{1}{3}\text{Tr}\,\mathbf{A}. \quad (17)$$

This leads to either $A_{XX} \geqslant A_{YY} \geqslant A_{ZZ}$ or $A_{XX} \leqslant A_{YY} \leqslant A_{ZZ}$ order of the principal components. Then, *anisotropy*[3] and the *asymmetry* parameters of **A** are defined, respectively, as

$$A_{\text{aniso}} = A_{ZZ} - A_{\text{iso}} \quad (18)$$

and

$$\eta_A = \frac{A_{YY} - A_{XX}}{A_{\text{aniso}}}. \quad (19)$$

The orientation of the principal axes of **A** is given by the set of Euler angles (Fig. 1), which specify the rotation from the principal to the "crystallite" axes. The later are used as the common reference frame for all tensors in the spin system under consideration. In general, the rotation from any coordinate frame A to any coordinate frame B will be specified by the set of Euler angles $\Omega_{\text{AB}} = (\alpha_{\text{AB}}, \beta_{\text{AB}}, \gamma_{\text{AB}})$.

All interactions in the Hamiltonian above are defined through invariant expressions that involve first- and second-rank Cartesian tensors ($\mathbf{I}_i$, $\mathbf{n}_{B_0}$, $\mathbf{\Omega}^i$, etc.) and have one of two forms, $\mathbf{x}\cdot\mathbf{A}\cdot\mathbf{y}$ or $\mathbf{x}\cdot\mathbf{y}$. Such expressions can be also formulated in terms of the irreducible spherical components of the same tensors [50–52,57–59]:

$$\mathbf{x}\cdot\mathbf{A}\cdot\mathbf{y} = A_{0,0}B_{0,0} + \sum_{q=-1}^{1}(-1)^q A_{1,-q}B_{1,q} + \sum_{q=-2}^{2}(-1)^q A_{2,-q}B_{2,q}, \quad (20)$$

[1] In the current version of the program, this term is not allowed.

[2] Although the chemical shift and the *J*-coupling tensors may possess antisymmetric components, the effects of these components on the NMR spectra are negligible.

[3] A_{aniso} is sometimes denoted as δ_A, leading to an unfortunate confusion with the symbol for the chemical shift. The term "anisotropy of tensor **A**" is also used for the quantity ΔA, which is defined as $\Delta A = A_{ZZ} - \frac{1}{2}(A_{XX} + A_{YY})$ and related to A_{aniso} as $\Delta A = \frac{3}{2}A_{\text{aniso}}$.

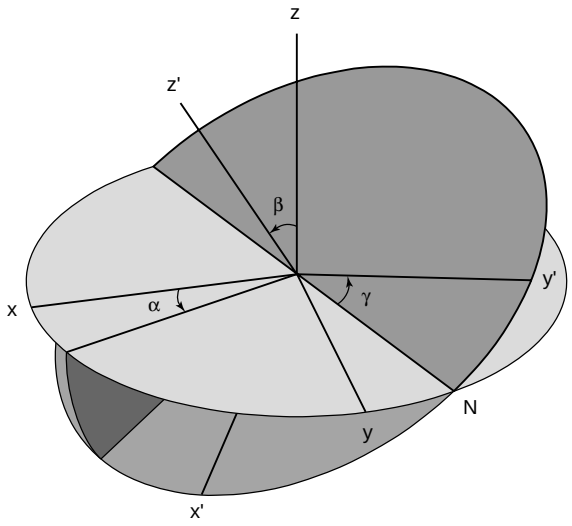

Fig. 1. The definition of the Euler angles.

$$\mathbf{x}\cdot\mathbf{y}=\sum_{q=-1}^{1}(-1)^q x_{1,-q}y_{1,q}, \tag{21}$$

where A_{lq}, B_{lq}, $x_{1,q}$, and $y_{1,q}$ are the irreducible spherical components of tensors $A_{\alpha\beta}$, $x_\alpha y_\beta$, x_α, and y_α, respectively. In general, we will use the symbols $\Omega^i_{2,q}$, $D^{ij}_{2,q}$, etc., to denote the second-rank spherical components of tensors $\mathbf{\Omega}^i$, $\mathbf{D}^{ij}$, etc. The first-rank (antisymmetric) components of these tensors can be neglected in the secular approximation.

The spherical components are particularly convenient for expressing the Hamiltonian in the *rotating frame* and obtaining its high-field, or secular, approximation, which is the form currently assumed for the Hamiltonian in SPINEVOLUTION. The rotating frame is an interaction representation defined in the program by the interaction term

$$H_0=\sum_k\left(\omega^k_*+\omega^k_{\rm off}(t)\right)F_{kz}, \tag{22}$$

where the **z**-axis is chosen in the direction of the static magnetic field $\mathbf{B}_0$. For the spins of species k, the frame rotates *continuously* about the direction of $\mathbf{B}_0$ with the angular frequency $\omega^k_*+\omega^k_{\rm off}(t)$, which is the instantaneous angular frequency of the on-resonance circularly polarized RF component (Eq. (10)). The continuity of rotation implies that the frame moves without jumps at all times. Note that, to avoid ambiguity, the frequency offsets must be defined even in the absence of an RF field.

In the high-field approximation, the rotating frame Hamiltonian is given by the sum of the following terms:

$$H=H_{\rm RF}+H_{\rm CS}+H_{\rm D}+H_{\rm J}+H_{\rm Q}, \tag{23}$$

where:

$$H_{\rm RF}=\sum_k\left|\omega^k_{\rm RF}(t)\right|\left(F_{kx}\cos\phi_k(t)+F_{ky}\sin\phi_k(t)\right), \tag{24}$$

$$H_{\rm CS}=\sum_i\left\{\Omega^i_{\rm iso}+\frac{2}{\sqrt{6}}\Omega^i_{2,0}(t)-\omega^{[i]}_{\rm off}(t)\right\}I_{iz}, \tag{25}$$

$$H_{\rm J}=\sum_{i<j}^{\rm homo}2\pi J^{ij}_{\rm iso}\mathbf{I}_i\cdot\mathbf{I}_j+\sum_{i<j}^{\rm homo}2\pi J^{ij}_{2,0}(t)\frac{1}{\sqrt{6}}(3I_{iz}I_{jz}-\mathbf{I}_i\cdot\mathbf{I}_j)+\sum_{i<j}^{\rm hetero}2\pi J^{ij}_{\rm iso}I_{iz}I_{jz}+\sum_{i<j}^{\rm hetero}2\pi J^{ij}_{2,0}(t)\frac{2}{\sqrt{6}}I_{iz}I_{jz}, \tag{26}$$

$$H_{\rm D}=\sum_{i<j}^{\rm homo}D^{ij}_{2,0}(t)\frac{1}{\sqrt{6}}\left(3I_{iz}I_{jz}-\mathbf{I}_i\cdot\mathbf{I}_j\right)+\sum_{i<j}^{\rm hetero}D^{ij}_{2,0}(t)\frac{2}{\sqrt{6}}I_{iz}I_{jz}, \tag{27}$$

$$H_{\rm Q}=\sum_i\frac{2\pi\chi_i}{2I_i(2I_i-1)}\tilde{V}^i_{2,0}(t)\frac{1}{\sqrt{6}}(3I^2_{iz}-\mathbf{I}^2_i)+\frac{1}{2\omega^i_0}\left(\frac{2\pi\chi_i}{2I_i(2I_i-1)}\right)^2\left\{\left|\tilde{V}^i_{2,2}(t)\right|^2(2\mathbf{I}^2_i-2I^2_{iz}-\mathbf{1})I_{iz}-\left|\tilde{V}^i_{2,1}(t)\right|^2(4\mathbf{I}^2_i-8I^2_{iz}-\mathbf{1})I_{iz}\right\}. \tag{28}$$

All tensor quantities in these equations must be given in the laboratory reference frame with the **z**-axis parallel to the static magnetic field. The anisotropic interactions ($\mathbf{\Omega}^i$, $\mathbf{D}^{ij}$, $\mathbf{J}^{ij}$, $\mathbf{V}^i$) are known in the crystallite (molecular) frame and thus must be properly transformed. In MAS experiments, where the sample is spinning about an axis that makes angle $\theta_{\rm M}=\arccos\sqrt{3}^{-1}\approx 54.74^\circ$ with the static magnetic field, the following sequence of transformations is used: $\mathrm{P}\to\mathrm{C}\to\mathrm{R}\to\mathrm{L}$, where P, C, R, and L stand for the principal, crystallite, rotor, and laboratory frames, respectively. SPINEVOLUTION assumes that the sample rotation proceeds in the *positive* direction about the **z**-axis of the rotor-fixed frame, and that $\Omega_{\rm RL}=(0,\ \beta_{\rm RL},\ 0)$ at the time $t=0$. This leads to the following expression for the laboratory frame components of the tensors:

$$A_{2,q}(t)=\sum_{k=-2}^{2}A^{(k)}_{2,q}\mathrm{e}^{\mathrm{i}k\omega_{\rm R}t}, \tag{29}$$

$$A^{(k)}_{2,q}=\frac{3}{\sqrt{6}}A_{\rm aniso}\left\{D^{(2)}_{0k}(\Omega_{\rm PR})-\frac{\eta_A}{\sqrt{6}}\left[D^{(2)}_{2k}(\Omega_{\rm PR})+D^{(2)}_{-2k}(\Omega_{\rm PR})\right]\right\}\times d^{(2)}_{kq}(\beta_{\rm RL}), \tag{30}$$

$$D^{(2)}_{km}(\Omega_{\rm PR})=\sum_{q=-2}^{2}D^{(2)}_{kq}(\Omega_{\rm PC})D^{(2)}_{qm}(\Omega_{\rm CR}), \tag{31}$$

where $D^{(2)}_{km}(\Omega)$ and $d^{(2)}_{km}(\beta)$ are the Wigner and the reduced Wigner rotation matrices.

Once the Hamiltonian is defined as given by Eqs. (23)–(28), SPINEVOLUTION solves either the Liouville–von Neumann equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\rho(t)=-\mathrm{i}[H(t),\rho(t)] \tag{32}$$

or the master equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\rho(t) = -\mathrm{i}[H(t), \rho(t)] - \Gamma(\rho(t) - \rho_{\mathrm{eq}}) \tag{33}$$

for a specified initial condition $\rho(0) = \rho_0$. In the current version of the program, the relaxation supermatrix Γ must be provided explicitly. The final goal of the simulation is to compute the ensemble averages of one or more observables Q_μ (i.e., the signal)

$$S_\mu(t) = \mathrm{Tr}\big(\rho(t)Q_\mu\big) \tag{34}$$

or, equivalently, their Fourier transforms.

In most solid-state NMR experiments, the results must be averaged over all possible crystallite orientations in the sample. If the sample is an isotropic powder, then the average is given by the integral

$$\bar{S}_\mu(t) = \frac{1}{8\pi^2}\int_0^{2\pi}\mathrm{d}\alpha\int_0^{\pi}\mathrm{d}\beta\sin(\beta)\int_0^{2\pi}\mathrm{d}\gamma S_\mu(\alpha,\beta,\gamma;t), \tag{35}$$

where the Euler angles (α, β, γ) specify the $\mathrm{C}\to\mathrm{R}$ rotation in MAS experiments, and $\mathrm{C}\to\mathrm{L}$ rotation in static experiments. The integral is readily approximated by a variety of finite "powder averaging schemes" [60–65]. If the sample is partially oriented, the signal under the integral must be multiplied by a proper weighting function, and the powder averaging scheme must be appropriately modified. If pulsed field gradients are used in the experiment, the signal is additionally averaged over a number of "gradient slices" to approximate the integral

$$\bar{\bar{S}}_\mu(t) = \frac{1}{L}\int_{-L/2}^{+L/2}\bar{S}_\mu(z,t)\,\mathrm{d}z, \tag{36}$$

where L is the sample length along the gradient axis.

3. Canonical representation of NMR experiments

In an NMR experiment, each data point is acquired by observing the magnetization of the sample after it has been subjected to a certain sequence of RF pulses and delays. In other words, each point is obtained at the end of a certain *RF path*. The pulse sequence of the experiment can then be thought of as an ordered collection of such RF paths. In experiments, a pulse program is used to instruct the spectrometer to generate these paths. In a simulation, one usually has to write a program that describes (algorithmically) which propagators have to be computed and how they need to be manipulated in order to emulate the desired pulse sequence (as is the situation, for example, with GAMMA and SIMPSON).

In order to avoid programming, the RF paths comprising the pulse sequence of the experiment can be (most naively) treated as a set of independent, uniform sequences of time events—pulses and delays. Although most general and non-algorithmic, such representation has virtually no structure and is very difficult to use for the construction of efficient simulations that must take advantage of various time symmetries of the Hamiltonian during the experiment. Fortunately, NMR pulse sequences typically follow certain established patterns, which can be readily described by means of a simple, well-structured *set of variables* that does encapsulate the periodicity of the experiment and can be readily explored for the construction of efficient propagation schemes. Such description is the primary purpose of the canonical representation formulated below. In the present paper, the representation is used mostly as the formalism behind the SPINEVOLUTION interface. However, it will be also heavily relied on for the *analysis* of MAS NMR experiments when the algorithms used in the program are described at a future date.

The basic structural unit of the pulse sequence of an NMR experiment in the canonical representation is an *elementary pulse sequence*, or simply *pulse sequence*. It is defined as a contiguous group of one or more RF pulses (the *RF cycle* of the sequence) that is characterized by a certain *sampling pattern*. Each pulse in the group has a fixed duration, power, phase and frequency (delays are described as pulses given with zero RF power). A pulse sequence is *executed* by the periodic repetition of its RF cycle for a certain amount of time (an example is shown in Fig. 2A). The sampling pattern of the sequence specifies its *sampling rate*, *sampling dimension*, and *sampling direction*, all of which are explained below. When more than one RF channel is active in the experiment, the pulse sequence is represented by an individual RF cycle on every channel. The duration of the RF cycle and the sampling pattern are the same for all channels, but the break-up into individual pulses within the RF cycle is independent on each channel. Pulsed field gradients, if present, add yet another channel to the pulse sequence, and are described using exactly the same formalism, except that the pulses constituting the PFG cycle are given by their durations and field gradients.

The *pulse sequence of the experiment* is described as a linear collection of elementary pulse sequences, meaning that each RF path of the experiment is generated by executing these sequences one after another. The order in which the elementary pulse sequences are executed is always the same in a given experiment, and the sequences are numbered 1 to S according to that order. Each data point is acquired by taking a measurement after the *last* pulse sequence was executed.

The execution time of every pulse sequence s on the RF path to any data point is always a multiple of a period $\tau^{(s)}$ called the *dwell time* of the sequence. (Thus, the pulse sequences are sampled periodically.) Let $t^{(s)}_{\mathrm{seq}}$ denote the length of the RF cycle of the pulse sequence s. Then, the ratio $g^{(s)} = t^{(s)}_{\mathrm{seq}}/\tau^{(s)}$ will be called the *sampling rate* of the sequence. To keep the sampling of the sequence and its RF field synchronized, we require that

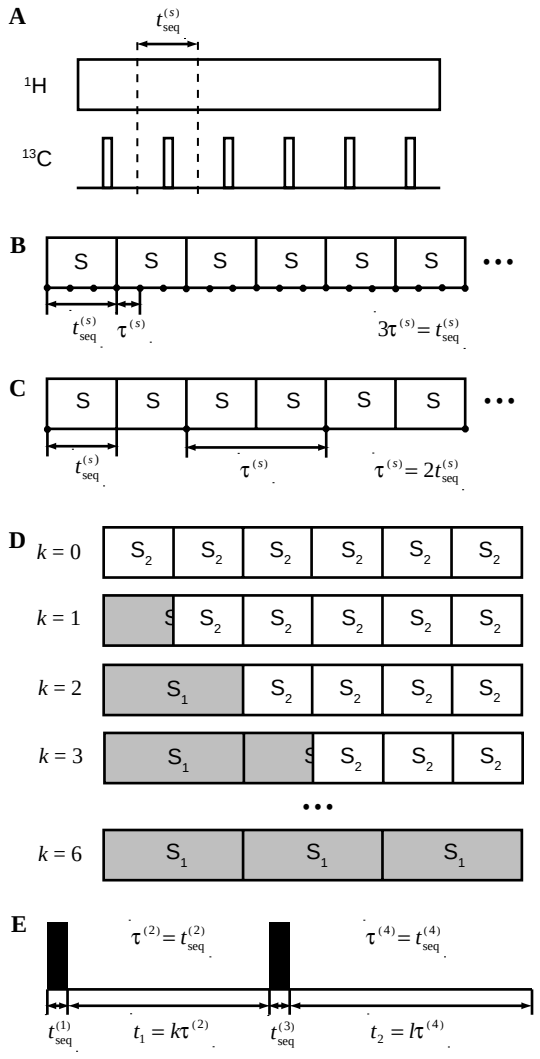

Fig. 2. (A) Execution of six RF cycles of an elementary pulse sequence; the RF cycle is indicated with the dashed lines. (B) Pulse sequence sampling. An elementary pulse sequence is always executed from the start of its RF cycle to one of the sampling points (marked with the thick dots). In the figure, the RF cycle is symbolically represented as the block S; the dwell time $\tau^{(s)}$ is one-third of the RF cycle, implying that the sampling rate $g^{(s)} = 3$. (C) The same sequence is sampled with the group size $G^{(s)} = 2$ (i.e., $g^{(s)} = 1/2$). (D) The RF paths of a constant-time, one-dimensional pulse sequence composed of two elementary pulse sequences. Both sequences are sampled in the same dimension; the first sequence is incremented, with two sampling points per RF cycle ($g^{(1)} = 2$); the second is decremented; $\tau^{(1)} = \tau^{(2)}$. The pulse sequence generates 7 data points (k is the point number). (E) A two-dimensional experiment (COSY [66]) composed of four sequences: 1 and 3 are D0 sequences, 2 is a D1 sequence, and 4 is a D2 sequence. The RF path shown leads to the data point (k, l).

either $g^{(s)}$ or $1/g^{(s)}$ is an integer.[4] In the former case, $g^{(s)}$ is the number of sampling points per one RF cycle of the sequence (Fig. 2B). In the latter case, the integer $G^{(s)} = 1/g^{(s)}$ will be called the *group size*, and, since $\tau^{(s)} = t_{seq}^{(s)} G^{(s)}$, we will say that the pulse sequence is *repeated in groups* of size $G^{(s)}$ (Fig. 2C). Most commonly, however, $g^{(s)} = G^{(s)} = 1$, and the pulse sequence is sampled at the end of every RF cycle.

Any segment of the pulse sequence between two adjacent *sampling points* (potential endpoints of the sequence execution, Figs. 2B and C) will be referred to as a *sampling segment*. Each data point is obtained by executing $N^{(1)}$ sampling segments of the first pulse sequence, $N^{(2)}$ segments of the second, etc., up to the $N^{(S)}$ segments of the last sequence. The set of S numbers $N^{(1)} \ldots N^{(S)}$, which will be called *repeat numbers*, fully identifies the exact RF path leading to this data point.

The complete set of the data points of the experiment is generated by changing the repeat numbers according to a certain pattern. To specify this pattern, each pulse sequence is assigned to a *dimension* of the experiment. A sequence assigned to the dimension i will be called a Di sequence (e.g., D0, D1, or D2 sequence). All sequences assigned to the same dimension change their repeat numbers *simultaneously*. The pulse sequence is *incremented* in the dimension i if its repeat number is changed starting from zero and going up to $N_i - 1$ along the dimension. The sequence is *decremented* in the dimension i if its repeat number starts from $N_i - 1$ and goes down to zero along the dimension. The process is illustrated in Fig. 2D. Sampling the dimension in this manner generates a one-dimensional set of N_i data points. The pulse sequences that are executed with the same repeat numbers for every data point are assigned to the *zeroth* dimension. Each dimension is sampled independently (except for the zeroth dimension, of course, which is not sampled at all), so that an n-dimensional pulse sequence generates an n-dimensional array of data (Fig. 2E).

What we have just described is an *n-dimensional pulse sequence*. However, individual dimensions of NMR experiments can sometimes be constructed in a way that does not fit into the description above, which necessitates the notion of a *parameter scan dimension*. Such a dimension is constructed by declaring a *scan parameter* and allowing other variables of the experiment to become functions of this parameter. Without loss of generality, one may assume that the scan parameter is just the index of that dimension, l_i, which is scanned from 0 to $N_i - 1$. The variables that depend on this parameter can then be as simple as the sample spinning frequency, or as complicated as the set of powers that define a shaped pulse, making the shape change in this dimension. The parameter scan dimensions are also sampled independently from each other and from the dimensions of the pulse sequence. Altogether, the parameter scan dimensions and the pulse sequence dimensions form *the dimensions of the experiment*.

As clear from the description above, non-D0 pulse sequences are periodic by definition. In MAS experiments, however, the pulse sequences must also be *rotor-synchronized* so that the efficient periodic algorithms

[4] More generally, the sampling rate can be a rational number, but this generalization seems to have little practical value.

could be used for their propagation. We will say that the pulse sequence s is rotor-synchronized if the following fundamental condition is satisfied:

$$n^{(s)} \cdot t_{\mathrm{seq}}^{(s)} = m^{(s)} \cdot \tau_{\mathrm{R}}, \tag{37}$$

where $n^{(s)}$ and $m^{(s)}$ are some small positive integers, $t_{\mathrm{seq}}^{(s)}$ is the duration of the pulse sequence cycle, and τ_{R} is the rotor period. The symmetry numbers $n^{(s)}$ and $m^{(s)}$ are unambiguously fixed by the additional requirement that they are relatively prime, i.e., have no common divisors except ± 1. The Hamiltonian of such a sequence has a period extending over $n^{(s)}$ cycles of the sequence or, equivalently, over $m^{(s)}$ rotor cycles. Eq. (37) essentially requires only that $t_{\mathrm{seq}}^{(s)}/\tau_{\mathrm{R}}$ is a rational number. The specific meaning of the ''smallness'' of the integers $n^{(s)}$ and $m^{(s)}$ may depend on the context.

Although most of the terms defined above are very familiar, their precise meanings in the context of the canonical representation may differ from the broader semantic range assumed by these terms in NMR literature. Thus, we would like to emphasize some of these unconventional aspects of the representation. Perhaps, the most important one is the central notion of the elementary pulse sequence as the basic structural unit of the pulse sequence of the experiment. The latter is represented simply as a dynamic sequence of these units, each characterized by its own periodicity and sampling parameters. This is quite different from the conventional viewpoint, where the constituent pulse sequences of an NMR experiment are regarded as modules that accomplish a certain task, such as evolution or transfer of coherences. Furthermore, the manner or the order in which the data points are obtained has no relevance in the canonical representation, while it is very relevant experimentally (one-dimensional data sets are usually obtained in a single transient). The canonical representation also introduces the notion of the zeroth dimension, makes no distinction between pulses and delays, and does make a distinction between the dimensions generated by the pulse sequence and those generated by a general parameter scan.

4. SPINEVOLUTION methodology

SPINEVOLUTION is designed as an application program rather than as an API, thereby relieving the user from any programming, including the pulse programming. In part, this is accomplished by using the canonical representation to represent the pulse sequences. The program relies on this representation both externally—as a part of the interface, and internally—as the means by which the simulation is constructed. The user may, and sometimes has to direct the program to employ one or another computational technique for some aspect of the simulation, but the details of the computation are always left to the program. For the most part, the program itself is able to choose the most efficient way to perform the simulation. Such methodology enables simultaneously a very simple interface and an almost unlimited internal flexibility to analyze the problem, organize the data, and choose the algorithms for the computation in the most efficient manner. Currently, some of these choices are predefined as functions of the pulse sequence and of the spin system, while others are made on the fly, contingent on the combination of various factors, such as the size of the Hamiltonian, or certain characteristics of the hardware (CPU and memory) on which the program is running. The user is not required to understand the details of the computational methods employed in the simulation. However, a general understanding of the basics is still desirable. In particular, one has to choose the most appropriate description of the pulse sequence, specify the powder averaging scheme, and decide if relaxation must be included in the model. These choices will affect the efficiency and, sometimes, even the correctness of the simulation. If the choices are blatantly inappropriate, SPINEVOLUTION will usually advise the user about this. The interface permits extensive checking for inconsistencies on the input, which facilitates detection of these ''user errors,'' adding significantly to the convenience and reliability of the program.

The code of the program is written in C and highly optimized. In addition, extensive uses are made of various numerical libraries and source code packages [67–77]; the latter are coded mostly in FORTRAN.

The main input to the program is provided by the main input file, a plain text file edited by the user from a template. This file may be either self-contained, or may refer to other files with information about the spin system, pulse sequences, various variables and expressions, powder averaging angle sets, and other details of the simulation. In this way, the same pulse sequence and parameter files can be used to construct new simulations. The interface is also well suited for use with the system shell, Pearl, MATLAB, and other scripts.

The pulse sequences are specified through the pulse lengths, powers, phases and frequencies of each pulse and the repeat pattern of the sequence. The dipolar interactions are automatically computed from the atomic coordinates, which may be specified either explicitly or through internal coordinates (internuclear distances, bond angles, torsion angles, and molecular fragment rotations and translations). Any part of the dipolar, chemical shift or RF term can be turned on or off during any part of the experiment. Groups of atoms can be specified to undergo infinitely fast exchange. Variables in SPINEVOLUTION provide a way to calculate the spin system and pulse sequence parameters from expressions provided by the user, perform parameter scans, data fitting, and other tasks. The default course of the computation can be modified by means of the command line options.

The output of the program is normally saved as a table of data points into a simple text file or files. No tools are provided with the program to visualize these data since a number of excellent software packages, both commercial and non-commercial, are widely available to suit everyone's personal preferences. We use Grace (URL: http://plasma-gate.weizmann.ac.il/Grace/) and MATLAB (MathWorks, Natick, MA) for these purposes.

The following introductory examples illustrate the description of pulse sequences in SPINEVOLUTION. First, consider a $\pi/2$ pulse followed by acquisition. In the canonical representation, this is a one-dimensional experiment composed of two pulse sequences: a $\pi/2$ pulse (a D0 sequence) and a delay (a D1 sequence). If, for example, the pulse takes 5 μs, the dwell time during acquisition is 20 μs, and 1024 points are to be collected, the sequence is described in the main input file by the following four self-explanatory lines:

```
timing(usec)   5   (20)1024
power(kHz)     50   0
phase(deg)     0    0
freq_offs(kHz) 0    0
```

A much more sophisticated experiment—a double quantum filter using the SPC-5 recoupling sequence [78] in solids—still looks almost as simple:

```
timing(usec)  (spc5.pp)41 (spc5.pp)41 5
power(kHz)     *           *          50
phase(deg)     *           *          90
freq_offs(kHz) *           *           0
phase_cycling  *          1234         * 3131(RCV)
```

Here, `spc5.pp` is the name of a file that describes the repeated element (the RF cycle) of the SPC-5 pulse sequence; the file itself is just a table of the durations, powers, phases, and frequencies of its 30 constituent pulses. The asterisks instruct the program to load the values from the file. The experiment goes through a four-step phase cycle (each pulse sequence is phase-cycled as a whole). No phase cycling is requested for the first and the third sequences (indicated by the asterisks), while the second sequence and the receiver are cycled as (x, y, $-x$, $-y$) and ($-x$, x, $-x$, x), respectively. The timing line indicates that this is a one-dimensional experiment composed of three pulse sequences, with the first two repeated from 0 to 40 times (incremented simultaneously), yielding 41 data point in the dimension.

The simplest example of a two-dimensional experiment is the basic COSY sequence [66], $\pi/2$ – (t_1-evolution) – $\pi/2$ – (t_2-evolution). Assuming 5 μs $\pi/2$ pulses and 20 μs dwell times in both dimensions, the timing line that specifies the four pulse sequences of this experiment is

```
timing(usec) 5 (20)1024D1 5 (20)1024D2
```

It may seem unusual at first, but zero-dimensional experiments (those composed of D0 pulse sequences only) are very common in simulations. Indeed, the acquisition dimension of the actual experiment is often used only to observe the intensity of the single-quantum coherences for each spin resulting from the prior evolution of the spin system. In simulations, however, an FID does not have to be recorded to acquire this information. Instead, one can simply compute the traces of the density matrix with the appropriate observables. Thus, 1D experiments can often be simulated as zero-dimensional, while 2D experiments—as one-dimensional (as in the SPC-5 double quantum filter example above).

The dimensionality of a simulation (the total number of dimensions in the data output) can be larger than the dimensionality of the pulse sequence. Additional dimensions can come from two other sources: a parameter scan (as explained earlier) and a "trajectory observation." In the latter case, measurements are made after every pulse of the zero-dimensional pulse sequence, thus registering the whole trajectory of the observable rather than just the last point. Currently, SPINEVOLUTION allows up to two-dimensional simulations. However, if the spins are observed individually rather than as a channel, this effectively gives an equivalent of yet another dimension.

A number of computational techniques exploit the periodicity and the time-γ-translational symmetry [25,47,49] of the Hamiltonian to make NMR simulations more efficient. Obviously, these algorithms can be used only for the parts of the experiment that exhibit such symmetries. In SPINEVOLUTION, all calculations within the pulse sequence dimensions are performed by means of such algorithms, while the calculations within dimensions produced by a parameter scan are performed completely independently from one another. Let us once more consider the example of a $\pi/2$ pulse followed by FID acquisition. It is, of course, possible to formulate this experiment as a zero-dimensional pulse sequence with a parameter scan. Namely, one can say that it consists of the $\pi/2$ pulse followed by the $n\tau$-long delay, where n is a scan parameter, which is scanned from 0 to 1023. In this case, however, the relationship between the propagators necessary to compute the FID is obscured. SPINEVOLUTION will recover some of the symmetry from such a pulse sequence by breaking up the delay into small periodic segments (assuming that this is a MAS experiment), using the periodic algorithms to compute the total propagator, and repeating this procedure for each n. However, the computation will not be nearly as efficient as it could have been if the experiment were formulated as one-dimensional. One should also keep in mind that a non-periodic problem can usually be replaced with a very similar, experimentally indistinguishable periodic problem. Taking advantage of this simple fact is completely at the user's discretion.

5. Performance of SPINEVOLUTION

Two examples were chosen to illustrate the performance of SPINEVOLUTION. In both cases, the same simulation problems were solved with our program and with SIMPSON [39], and the results compared.[5] The first example we chose is the simulation of the line shape of a single ^{13}C nucleus coupled to 1–9 protons (Fig. 3) observed using the TPPM decoupling [79]. The TPPM decoupling sequence is a windowless train of constant-power, approximately 180° pulses with phases alternating between 0 and ϕ, where ϕ is usually in the range of 12–18°. The details of the simulation are given in the footnotes to Table 1. Both SPINEVOLUTION and SIMPSON used the same top-level algorithm (γ-COMPUTE [47–49]) to perform these simulations. The CPU time taken by the simulation is indicative of the performance of the many components the algorithm relies on: integration of the equation of motion routines, matrix diagonalization routine, matrix multiplication, fast Fourier transform, etc. However, most of the CPU time is spent in this case on the integration of the equation of motion, which is performed by the Chebyshev expansion algorithm in SPINEVOLUTION, and via matrix diagonalization in SIMPSON.

The spectra produced by SIMPSON and SPINEVOLUTION were identical and exhibited a single narrow line for all spin systems. However, the efficiency of the calculation turned out to be dramatically different for the two programs, as shown in Table 1. For the small spin systems, SPINEVOLUTION was about two orders of magnitude faster than SIMPSON. The difference in the efficiencies quickly increased with the system size, reaching a factor of 3300 for the 7-spin system. A graphical representation of these results is shown on the logarithmic scale in Fig. 4.

It is remarkable that in the case of SPINEVOLUTION, the logarithm of the calculation time is practically linear in the number of spins. The slope corresponds to a factor of 6.3 increase in the computation time for each additional spin. These observations can be qualitatively understood through the following rudimentary considerations.

All numerical routines can be classified according to the manner in which the number of elementary arithmetic operations in the algorithm scales with the dimensionality of the problem. For example, matrix–matrix multiplication and matrix diagonalization are $O(n^3)$ processes, meaning that the number of arithmetic operations performed by these routines is proportional to the third power of the matrix dimensionality n. The most computationally intensive operations from this viewpoint, they usually take most of the CPU time in NMR simulations. Since the space dimensionality of an N-spin-1/2 system is 2^N, the entire simulation is an $O(2^{3N}) = O(8^N)$ process if it is dominated by these algorithms. The diagonalization of a symmetric matrix takes approximately 3–15 times longer to complete than a complex matrix–matrix multi-

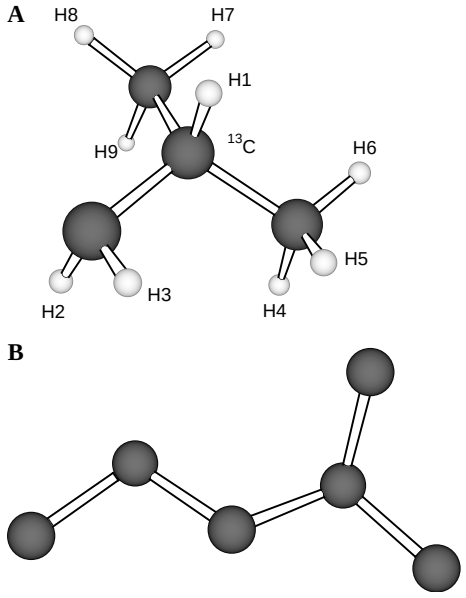

Fig. 3. (A) An iso-butyl group, the configuration used in the TPPM simulations. The ^{13}C nucleus is located at C2. ^{1}H nuclei were added to the spin system as numbered in the figure. (B) The carbon skeleton of Leucine, the spin system used in the 2D correlation spectrum simulation.

Table 1
CPU time (seconds) spent on the calculation of TPPM-decoupled ^{13}C spectra in the systems of 2 to 10 nuclei[a,b]

Spin system[a,c]	SPINEVOLUTION	SIMPSON
CH_1	0.13	8.8
CH_2	0.33	19.1
CH_3	1.27	73.4
CH_4	6.16	638
CH_5	35.0	13300
CH_6	246	822000
CH_7	1600	—
CH_8	9460	—
CH_9	60500	—

[a] Simulation parameters: spinning frequency $\omega_R/2\pi = 15.625$ kHz; RF field strength at the ^{1}H channel $\omega_{RF}/2\pi = 125$ kHz; TPPM pulses were 4 μs long, alternating between 0 and 15° phases; the spins were placed in the configuration of an iso-butyl group, with the ^{13}C nucleus positioned at the C2 location (Fig. 3A).

[b] Simulation methods (for both programs): stepwise Hamiltonian integration in 1 μs steps; powder averaging over 168 (α, β) pairs from the REPULSION set, and eight γ angles, implicitly by the γ-COMPUTE algorithm. Calculations were performed using a Linux PC with a 1.2 GHz Athlon CPU. The versions of the programs used were SIMPSON 1.1.0 and SPINEVOLUTION 2.4.

[c] ^{1}H nuclei were added to the system in order shown in Fig. 3A.

[5] The input files used in these computations are included with the other examples into the current distribution of the program. SPINEVOLUTION was also briefly compared with BlochLib/Solid-2.0 by simulating Rotational Resonance spectra of spin systems with 2 to 9 coupled spins. For these examples, SPINEVOLUTION was found 10–90 times faster than BlochLib.

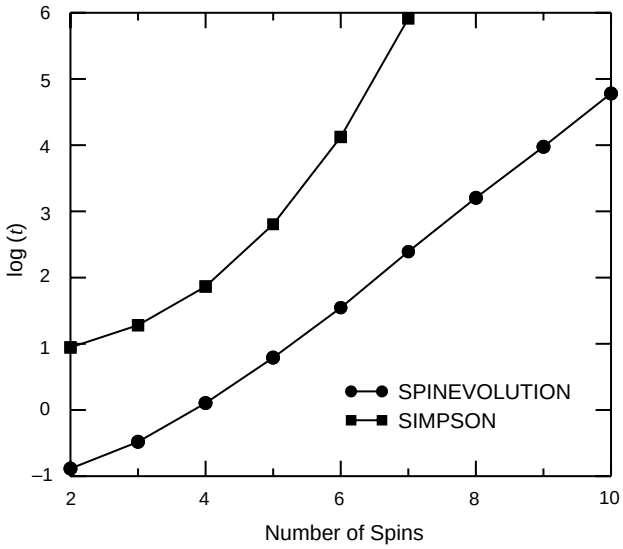

Fig. 4. Decimal logarithm of the CPU time taken by the simulations of TPPM-decoupled ^{13}C spectra (see Table 1).

plication. Hence, the larger the proportion of the diagonalization operations, the slower the calculation. Furthermore, if these routines are implemented inefficiently, for large matrices, execution time will be determined by the speed of the data transfer to and from the main memory of the computer rather then the speed of the processor, leading to significant losses of efficiency. This seems to be the situation with SIMPSON, which is much slower than one would expect from the $O(8^N)$ rule (Fig. 4). The computation time is apparently dominated here by the diagonalizations of the Hamiltonian employed to integrate the equations of motion. SPINEVOLUTION, on the other hand, never diagonalizes the Hamiltonian. Instead, the equation of motion is integrated with a sparse matrix based algorithm that behaves asymptotically as $O(N^2 4^N)$. It should be noted, however, that reaching the theoretical asymptotic performance of sparse matrix algorithms is practically impossible in actual computations. In addition, for most of the values of N in the example, a significant fraction of the computation time is still given to matrix-matrix multiplications. The resulting total CPU time then depends on a combination of $O(N^2 4^N)$, $O(8^N)$, and perhaps other terms, apparently leading to the observed 6.3-fold increase per spin in the simulation time observed for the larger systems. For $N = 2$ and $N = 3$ cases, the increase in the CPU time is not as steep since the contribution from $O(4^N)$ processes, like matrix–matrix addition, is significant for such small matrices.

In the second example, we simulated a 2D MAS ^{13}C–^{13}C correlation spectrum of the Leu fragment of the uniformly (^{15}N, ^{13}C)-labeled N-Ac-Val-Leu peptide molecule. Perfect decoupling from the ^{1}H nuclei was assumed, making it a six-spin computation, with the spin system shown in Fig. 3B. The pulse sequence is given later, in Example 6; other details of the simulation are provided in Fig. 5.

The calculation required 10 h of CPU time on a 1 GHz Pentium 3 computer. The entire simulated spectrum is presented in Fig. 5A. The (β,γ,δ)-region is shown in detail as a surface plot in Fig. 5B. The simulation was performed using regular matrix multiplication for propagation in the first dimension, while the g-COMPUTE algorithm was employed in the second. Such a combination of methods makes the calculation time proportional to the number of points taken in the first dimension, but

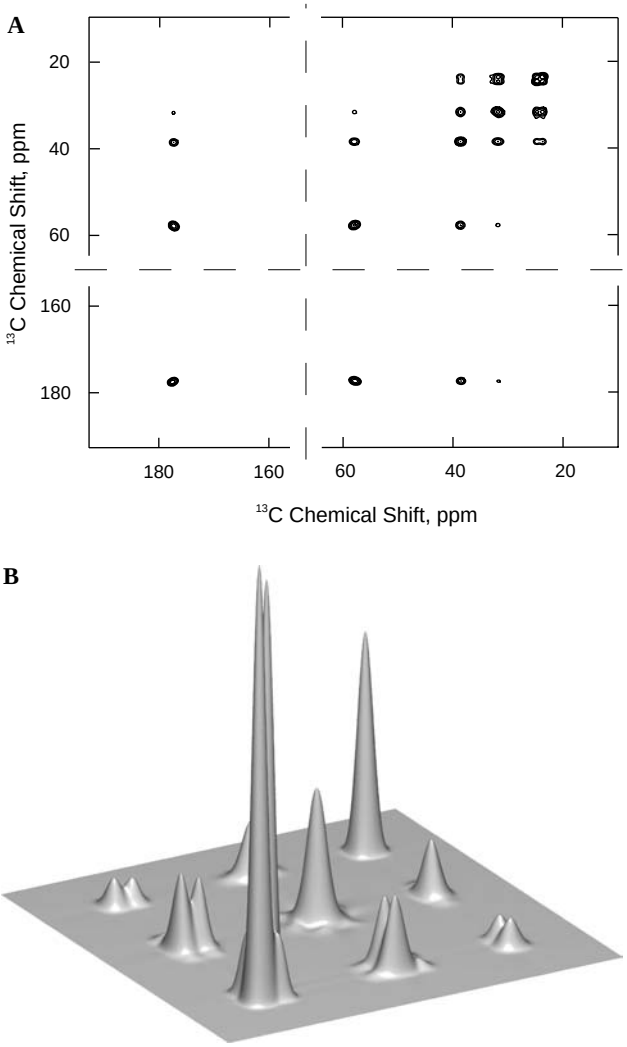

Fig. 5. Simulated 400 MHz ^{13}C chemical shift 2D correlation spectrum of the 6-spin Leu fragment of U-[^{15}N,^{13}C]-N-Ac-Val-Leu (Fig. 3B): contour plot (A) and (β,γ,δ)-region detail (B). The pulse sequence is shown in Example 6 below (Fig. 11). Perfect heteronuclear decoupling and ideal $\pi/2$ pulses were assumed. 2048 points were collected in the indirect dimension with 48 kHz spectral width; cosine transform was applied to obtain a pure phase spectrum. 1024 points were collected with 24 kHz spectral width in the direct dimension. Other simulation parameters: chemical shifts as seen in the spectrum, 40 Hz isotropic J-coupling for directly bonded nuclei, $\omega_R/2\pi = 8.0$ kHz, $\tau_{mix} = 2$ ms, 60 Hz Gaussian line broadening, powder averaging over 100 (α,β) pairs and 12 γ angles. The surface shown in (B) was obtained by a 2D-spline interpolation of the simulated spectrum.

independent of the size of the second. The advantage of this scheme can be illustrated by attempting to simulate this experiment with SIMPSON, which cannot use its γ-COMPUTE for such problems. The CPU time such a simulation would require was estimated by performing it for smaller numbers of sampling points n in each dimension and without powder averaging. The time required by the computation must be a second order polynomial of n. In full agreement with this expectation, all data points for n ranging from 2 to 64 were fit perfectly by such a polynomial. Extrapolation to $n = 1024$ and including a factor that accounts for the powder averaging yielded 38 years of CPU time, which is 33,000 times longer than for the simulation performed with SPINEVOLUTION.

6. SPINEVOLUTION interface

6.1. Main input file

The main input file is a text file that has four sections: System, Pulse sequence, Variables, and Options (see the examples below). Each line of the file has a fixed one-word header descriptive of the information to be specified on this line. The information must be provided according to a certain format. The formats are line-specific, but a few rules summarized below apply to all of them.

Spaces or tabs are used to separate different entries on a single line. Parentheses, brackets, colons, and other special symbols may be used only in compliance with the format specifications. Text lines are separated from each other by the new line character (ASCII code 10, also known as \n, LF, or the line feed character), which is the standard accepted by all Unix-based simple text editors (see `-convert` option for help with formats conversion). If the information requested on the line is irrelevant in the context of the experiment, or if the default values should be used, the line should be marked with an *asterisk*. Most lines can be ended with a comment that begins with an asterisk. This can be used, for example, to comment out the content of the line instead of erasing it.

In general, the order of the lines in the main input file cannot be changed, and none of the lines can be omitted from the file. There are a few exceptions from this rule. The `csa_parameters`, `j_coupling`, and `quadrupole` lines can be repeated as many times as necessary to describe all pertinent interactions; however, each of these lines should appear at least once. The number of the CHN subsections in the file varies according to the number of channels in the experiment. The phase cycling lines on each channel are optional. The Variables section has no mandatory lines, but the lines that do appear in it should be arranged in the order desired for the computation of the variables they define.

The following notation is employed to specify the input formats (particularly in the Tables A1–A3). The part of the format printed in **bold** should appear in the input file exactly as it is printed in the text. An *italicized* group of characters shows a variable part of the construct, the information that must be provided by the user. The vertical bar | and the operator OR separate the alternative forms of the construct. Braces around a group of items separated by vertical bars show that one of the items *must* be chosen. Square brackets around a construct show that the construct is optional. Three dots (an ellipsis) mean that the immediately preceding construct may be repeated one or more times. Examples, file names, etc. are printed using a `fixed-width` font. A detailed description of the main input file formats is given in the Table A1. Command line options and internal variables are described in Tables A2 and A3.

6.2. Variables

Variables play a versatile and important role in the SPINEVOLUTION interface. A variable can be assigned a value calculated from an arbitrary expression involving numerical constants, other variables, and functions thereof. A variable can be scanned through a certain list of values, adding a new dimension to the experiment. Alternatively, the results of the scan can be added up to form a weighted average. Variables also provide a flexible framework for data fitting and optimization.

Any name declared in the Variables section of the input file refers to either internal or a user-defined variable. Internal variables (Table A3) affect the system parameters (typically the Hamiltonian) directly, while user-defined variables can affect the results of the simulation only through internal variables that depend on them. Some internal variables (e.g., `spinning_freq` or `pulse_`*`n_s_p`*) provide an alternative way of referencing parameters that are normally specified in the System, the Pulse Sequence, and the Options sections of the input file, while others provide a unique access to certain features of the program (e.g., relaxation parameters or torsion angles). The value assigned to an internal variable overrides the value of the corresponding parameter specified in the System or the Pulse Sequence sections. All internal variables have names containing at least one underscore character to distinguish them from the user-defined variables, while the latter may be given any names composed of letters and numbers that begin with a letter.

Other than being internal or user-defined, each variable is also characterized by its type, which is determined by the way the variable was declared (see Table A1), and determines the way the variable behaves during the simulation. A variable can be of an active type (*fit parameter*, *scan parameter*, and *average-over parameter*), or of a passive type (the subtypes of which are distinguished only internally).

Declaring an active variable directs SPINEVOLUTION to invoke the appropriate control block. In partic-

ular, declaring a fit parameter invokes data fitting, declaring a scan parameter creates a scan parameter dimension in the experiment, and declaring an average-over parameter creates a "virtual" parameter scan dimension—all computed data points are eventually summed (or averaged with some weights) over this dimension. Every scan and average-over parameter is assigned a *list of values* (see Table A1 footnotes) when declared. These are the values sequentially assigned to the parameter during the scan. A fit parameter may also be (optionally) assigned a list of values, which specify in this case a grid of starting points for the optimization. The values of the fit parameters are controlled by the fitting routine.

A passive variable is declared by a line of the form

```
variable   name=expression
```

and is assigned the value obtained by evaluating the expression. Expressions may involve any previously declared passive or active variables, numeric values, the `pi` constant, the elementary algebraic operations `(+ - * /)`, and a number of other functions (Table A4). The algebraic operations are left-associative (meaning that `a/b/c=(a/b)/c`). Alternatively, variables can be declared from the command line by means of the `-var` option (such variables are evaluated prior to the variables declared in the main input file). This option is particularly useful for running batch simulations from a shell script, since it does not require editing of the main input file. If the variable is internal, it may be referenced from any expression without prior declaration. The initial values of the fit parameters can be assigned with the same syntax (the `variable` statement or the `–var` option).

If the expression of the variable is not dependent, directly or indirectly, on any active parameter, then the value of this variable is computed only once (when the variable is initialized) and remains constant for the rest of the simulation. Other passive variables are re-evaluated from their expressions whenever the active variables that they depend on are changed. Variables are always evaluated in the order in which they are declared.

A special "vectorized" notation can be used to create and operate with a number of variables by means of a one-line syntax. For example, the declaration

```
variable x[0:100]=a[0:100]*exp(-k*[0:0.01:1])+1.0
```

actually stands for 101 different declarations:

```
variable x0=a0*exp(-k*0.0)+1.0
…
variable x100=a100*exp(-k*1.0)+1.0
```

The individual declarations are obtained by substituting each bracketed list of values with one of its components taken in order from first to last.

Variables can be matrices as well as scalars. A row or column matrix can be declared by the `rowmatrix/colmatrix` statements (Table A1). A rectangular matrix can be declared by the `matrix` statement (Table A1), or it can be computed, for example, as a matrix product of a column matrix by a row matrix, or by "reshaping" a row, column, or another rectangular matrix. The individual elements of an m-by-n matrix M can be referred to either as $M(i, j)$, with $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$, or as $M(i)$, with $1 \leqslant i \leqslant mn$. Since the elements of a matrix are stored columnwise (so-called FORTRAN convention), the element $M(i,j)$ can be also referenced as $M(k)$ with $k = i + m(j - 1)$. The function `reshape`(M,m,n) returns an m by n matrix, whose elements are taken columnwise from the M's storage array. For example, the statements

```
rowmatrix   M/0:0:10/
variable    A=reshape(M,2,5)
variable    A(1,[1:5])=w*log(x[1:5])+B(1,1)
```

declare a row matrix `M`, reshape it into `A`, and re-compute the `A`'s first row via other, previously defined variables.

All the operations and functions defined for scalars are defined for matrices by element-wise threading of the operation or function. Element-wise multiplication and division of two matrices is denoted as `mul`(A, B) and `div`(A, B). Only two specifically matrix operations are implemented at this point: matrix multiplication, denoted as $A*B$, and the transpose, denoted as `transpose`(A).

6.3. Relaxation, powder averaging, and internal coordinates

When relaxation must be taken into account, SPINEVOLUTION uses the master equation of motion (Eq. (33)) instead of the Liouville-von Neumann equation. In the current version of the program, the relaxation operator matrix elements must be provided to the program explicitly, in the Zeeman basis. Currently, no cross-relaxation is allowed between off-diagonal elements of the density matrix. Assuming a non-degenerate Hamiltonian, the later restriction amounts to the secular approximation [53]. The required relaxation matrix elements are specified through the longitudinal and transverse relaxation times for individual coherences and corresponding transitions (T1 and T2 variables, Table A3), which are defined as generalizations of the T_1 and T_2 constants in the Bloch equations. A general matrix-based interface will be also available for this purpose in the next version of the program.

The user can choose between two relaxation models built into the program (the choice of the model, as well as the relaxation constants may vary from one pulse

sequence to another within a single experiment). In the "simple" model, the density matrix is allowed to relax after each RF cycle for a D0-sequence, or after each sampling point for a D1 or D2-sequence. The relaxation constants to be used with this model are, in general, effective averages over the RF cycle of the sequence or the period of the Hamiltonian. In a more general but also much more intensive computationally "full" model, the density matrix is allowed to relax after every integration step (set by the `-dt` option), leading to the exact integration of the equation of motion provided that the integrations steps are sufficiently small. In both models, the actual computations are often carried out via the Liouville-space propagators. The equilibrium density matrix is always defined as

$$\rho_{\mathrm{eq}} = \sum_{i=1}^{N} \frac{\gamma_{[i]}}{\gamma_{(^1\mathrm{H})}} I_{iz}. \tag{38}$$

The question of whether the chosen model is valid for the experiment that is being simulated is left to the user. In general, the simple model will not be applicable unless the relaxation rates are much smaller than the size of the effective Hamiltonian. In most cases, the simulation of relaxation significantly slows down the calculation.

Powder averaging is an essential part of most solid-state experiments. In the program, it can be performed over (α,β,γ), (β,γ), (α,β), and (β) angle sets, with the optional additional averaging over the angles that are left out from the set. The two- and three-angle sets are provided as external text files, while the (β) sets are automatically generated. Single-crystal calculations are also possible. A comprehensive compilation of various angle sets from the literature [60–62,64,65] collected by M. Edén and M. Levitt is provided with the program. We suggest consulting the URL http://www.soton.ac.uk/~mhlgroup/ for an overview of this compilation, and [65]—for a discussion of the symmetry implications on the choice of the most appropriate set. At this point, it is the user's responsibility to make sure that the number of crystallites in the powder averaging scheme chosen is sufficient for the given problem. The convergence is usually established by trying angle sets of different sizes. A much more advanced, fully automated approach to powder averaging, which should supersede all of the above, is also being developed.

As a complement or an alternative to the Cartesian coordinates representation, SPINEVOLUTION provides a general way of specifying the structure of the spin system through internal coordinates—distances, bond angles, torsion angles, and rotations and translations of individual molecular fragments. In principle, any structure can be completely defined solely in terms of these coordinates in SPINEVOLUTION. The exact meaning of each internal coordinate is defined in the System section of the main input file (see Table A1 for details), while its value is set in the Variables section. For example, the configuration of a four-spin chain, can be completely parametrized through the following lines in the System section:

```
bond_len_nuclei       (1 2) (2 3) (3 4)
bond_ang_nuclei       (3 2 1) (2 3 4)
tors_ang_nuclei       (1 2 3 4)
```

which define the bond lengths r_{12}, r_{23}, and r_{34}, the bond angles θ_{321} and θ_{234}, and the torsion angle φ_{1234}. Then, for example, the internal coordinates r_{12}, r_{23}, and θ_{321} can be referred to in the Variables section as `r_1`, `r_2`, and `theta_1`, respectively.

A definition of an internal coordinate turns the Cartesian atomic coordinates of certain nuclei (as well as the CSA and quadrupole tensors fixed on these nuclei) into functions that depend on the value of the corresponding internal variable. The internal coordinates are "set" into the spin system in the exact order they appear in the System section of the main input file (this order may be essential). This happens after all variables declared in the Variables section, including the internal coordinates themselves, have been calculated. Thus, any atomic coordinates and interaction tensors can be set into the desired "initial state" through their direct variables prior to the application of the internal coordinates. If the internal variable corresponding to a given internal coordinate is not declared, its value is left unchanged by the program; however, this value is still measured and can be inspected with the `-s` option.

6.4. The output and file naming conventions

The output of the program is produced in a text format with the real, imaginary, and/or longitudinal parts of the magnetization saved in separate files. The names of the output files are formed by appending `_re.dat`, `_im.dat`, or `_z.dat` to the name of the main input file. On the output of two-dimensional simulations, where the spins are observed individually (rather than as a channel), the data for each observed spin are saved in separate files, with the spin number appearing between the name of the input file and the automatically added suffix, e.g., *name*1`_re.dat`. Each output file is a zero-, one-, or two-dimensional data matrix, depending on the experiment. The first column of the file labels the rows of the matrix and is one of the following: time, in milliseconds, frequency, in kilohertz, chemical shift, in ppm, or the value of the scanned parameter. The output produced by the `-pwr` option is saved in the *name*.`pwr` file.

Several command line options exist that alter the default behavior: `-n` sets a different base name for the output files, `-to` transposes the output data matrices with respect to their default arrangement, `-x0` disables label-

ing the output data matrix, `-dw` changes the way the time label is calculated. See Table A2 for more details.

The user-supplied input files referenced from within the main input file can be given any names that do not contain white space characters. The references to these files are understood as pathnames in the main input file, i.e., they are allowed to contain a relative or the absolute path to the file. We suggest the following (optional) guidelines for naming the input files. The main input file name should have no extension and be descriptive of the nature of the experiment. The coordinates, isotropic chemical shifts, CSA, *j*-coupling, and quadrupolar interaction files should have extensions. `.cor`, `.cs`, `.csa`, `.j`, and `.q`, respectively. It is usually helpful to end each line of these files with some comment that labels the nucleus described at this line. Any non-numerical character in these files is considered by the program as the start of a comment that extends to the end of the line. Lines that do not contain any parameters and consist of comments only are also allowed in these files. The names of the files that describe pulse sequences should have extensions `.tm`, `.pwr`, `.phs`, or `.frq` if the file specifies, respectively, *only* the timing, the powers, the phases, or the frequency offsets of the sequence, while the `.pp` extension should be used if the pulse program specifies all four of them. The files that assign values or mathematical expressions to variables should have the extension `.par`. The only exception from this free form naming rule are the input files used for data fitting—these are expected to have certain predefined names.

The following conventions are used for the files used in the data fitting and optimization problems. If the name of the main input file that specifies the problem is *name*, then the to-be-fit data files are expected to have the names *name*`_re.fit`, *name*`_im.fit`, and/or *name*`_z.fit`, while the files with weights to be used for each data point should be named as *name*`_re.wht`, *name*`_im.wht`, and/or *name*`_z.wht`. The best-fit parameters are saved on the completion of the fit in the file *name*`.par`. See also the `-n` and `-fn` options (Table A2), which may be used to provide alternative names for these files. The covariance matrix of the fit parameters and the confidence intervals are saved in the *name*`.cov` and *name*`.cls` files if requested by the `-covmat` and `-confint` options. The grid search journal is written to the *name*`.jnl` file. Assuming that x_1, x_2, ... x_n are the fit parameters, each line of this file has the following format:

$$RSS \quad x_1^0\ x_2^0 \ldots x_n^0 \quad x_1^{\min}\ x_2^{\min} \ldots x_n^{\min},$$

where *RSS* is the residual sum of squares at the local minimum point $\left(x_1^{\min}, x_2^{\min} \ldots x_n^{\min}\right)$, which was obtained by starting the minimization from the $\left(x_1^0, x_2^0 \ldots x_n^0\right)$ point on the grid.

6.5. Running SPINEVOLUTION

The program is started by issuing

`spinev` *inputfile* [*options*]

at the system prompt. For best performance, SPINEVOLUTION should be configured for the machine it will be run on (see `-autoconfig` option).

Before running an actual simulation with a newly edited input file, it is advisable to perform a dummy run with the option `-s`. This option causes SPINEVOLUTION to load all necessary files, prepare for the simulation, and, instead of performing the calculation, print the loaded information to the terminal as it was interpreted and processed by the program.

Unless the full pathname to the angle set for powder averaging is given in the input file, the file will be looked for in the directory specified by the environment variable `ANGLES`. In the Linux `bash` shell, this variable can be set by issuing

```
export ANGLES=/usr/local/NMR/spinev/angles
```

assuming that the directory containing the Euler angle sets is `/usr/local/NMR/spinev/angles`. In `tcsh` shell, `setenv` command should be used instead:

```
setenv ANGLES /usr/local/NMR/spinev/angles
```

The location of the configuration file `spinev.cfg` and of the on-line help manual should also be specified to the program by setting (in the same way as above) the environment variable `SPINEV` to the SPINEVOLUTION home directory, where these files are contained. This directory should be also added to the system path. It is convenient to put all these definitions into a shell script automatically executed at login time (e.g., `.bashrc` or `.cshrc`).

Simulations that involve powder averaging can be parallelized on multi-processor machines or computer clusters configured to support automatic process migration (see option `-split`).

7. Examples

Examples in this section are provided, in part, to illustrate the relevance of the program for contemporary NMR research, but mostly—to demonstrate the proper use of various features of the program. If one desires to quickly become acquainted with most of these features, it is recommended to look through all of these examples, even if their NMR content is of little interest to the reader. Tables A1–A3 should be consulted throughout this section to clarify the details.

7.1. CSA powder pattern

The simulation of a CSA powder pattern is a very simple calculation, with the spin system consisting of only one nucleus, the CSA as the only interaction present in the Hamiltonian, and FID acquisition as the pulse sequence. The full main input file for this simulation is shown below. In the rest of the examples, the lines containing only asterisks will be omitted from the input files.

```
****** The System **********************
spectrometer(MHz)       500
spinning_freq(kHz)      0.0
channels                C13
nuclei                  C13
atomic_coords           *
cs_isotropic            *
csa_parameters          1 1 0.5 0 0 0
j_coupling              *
quadrupole              *
dip_switchboard         *
csa_switchboard         *
exchange_nuclei         *
bond_len_nuclei         *
bond_ang_nuclei         *
tors_ang_nuclei         *
groups_nuclei           *
******** Pulse Sequence *****************
CHN 1
timing(usec)          (200)512
power(kHz)              0
phase(deg)              0
freq_offs(kHz)          0
******* Variables **********************
******* Options ************************
rho0                    1 Ix
observed_spins          1 Ix
EulerAngles             ^400
n_gamma                 100
line_broaden(Hz)        *
zerofill                *
FFT_dimensions          1
options                 -oct
```

Essentially, the only point of concern in this simulation is to correctly specify the powder averaging scheme. An asymmetric CSA tensor generates a powder pattern that requires averaging over thousands of crystallites to be accurately reproduced in a simulation. As in all static experiments, the Euler angles for powder averaging define the transformation from the crystallite frame to the laboratory frame. The **z**-axis of the latter is fixed in the direction of the main magnetic field. Since the Hamiltonian is invariant with respect to the rotations about the field, powder averaging needs to be performed over α and β angles only (in a symmetric tensor, averaging over the angle α would be unnecessary either). The circumflex symbol (^), prefixing the number of β-averaging steps in the `EulerAngles` line, directs the program to interpret the number 100 given at the `n_gamma` line as the number of averaging steps in the angle α used. The option `-oct` restrains the sampling to one octant only. The resulting spectrum (Fig. 6) is obtained with $400 \cdot 100 = 40{,}000$ crystallites. The angle sets specifically designed for two-dimensional powder averaging are usually more efficient than independent sampling in each angle. However, SPINEVOLUTION currently does not automatically generate such sets, and the largest two-angle octant set included with the program (`zcwoct987`) is still far too small to faithfully reproduce the powder pattern in this example.

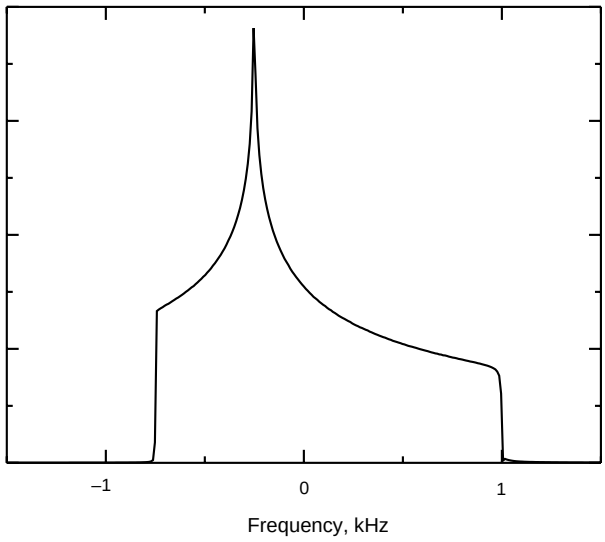

Fig. 6. CSA powder pattern ($\Omega_{aniso} = 1$ kHz and $\eta_{\Omega} = 0.5$).

7.2. REDOR

This is an example of a simple MAS experiment known as REDOR [80]. Its pulse sequence is shown in Fig. 7A. The portions that are grayed out are assumed to behave ideally and are not included into the simulation explicitly. During the REDOR pulse sequence, the transverse magnetization created on the ^{13}C nuclei by the cross-polarization with protons is dephased by the ^{13}C–^{15}N dipolar coupling reintroduced into the effective Hamiltonian by means of a train of π pulses on the ^{15}N channel. The resulting dephasing curve (Fig. 7B) can be computed with the help of the following input file:

```
(the lines containing only asterisks are not shown)
****** The System ********
spectrometer(MHz)  500
spinning_freq(kHz) 10.0
channels           C13 N15
nuclei             C13 N15
atomic_coords      1.367
cs_isotropic       2 0
******* Pulse Sequence ***************************
CHN 1
timing(usec)   (100)61    95  5    100      (100)61
power(kHz)       0          0 100    0          0
phase(deg)       0          0  0     0          0
freq_offs(kHz)   0          0  0     0          0
CHN 2
timing(usec) (redor.pp) 45 5 50 redor.pp (redor.pp)
power(kHz)       *       0 100 0    *          *
phase(deg)       *       0  0  0    *          *
freq_offs(kHz)   *       0  0  0    *          *
******* Variables ********
******* Options **********
rho0               1 0 Ix
observed_spins     1 Ix
EulerAngles        ^zcw987.dat
options            -dw123
```

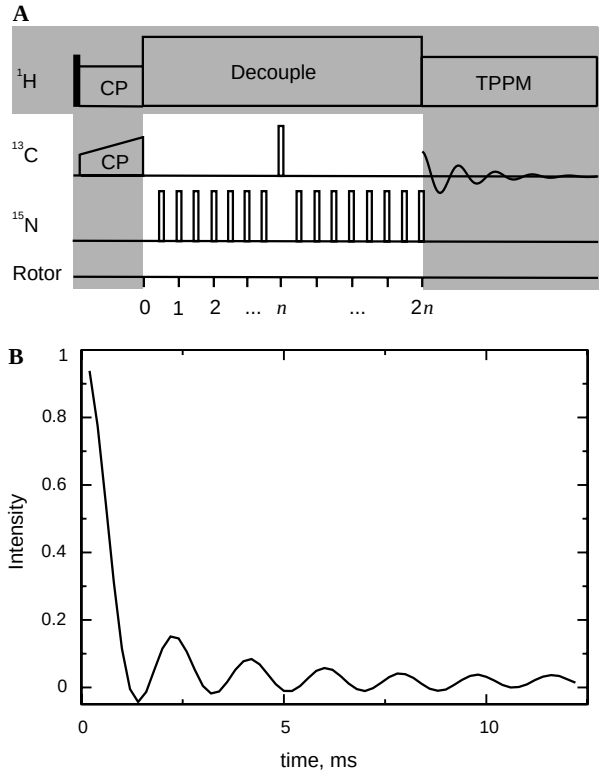

Fig. 7. The REDOR pulse sequence (A) and a simulated dephasing curve (B).

As seen in the timing line for the ^{15}N channel, the RF cycle of the REDOR sequence is provided in a separate file `redor.pp`. The file contains the following pulse program:

```
45     0     0     0
5     100    0     0
45     0     0     0
5     100   90     0
```

the four columns of which represent, respectively, the duration, the power, the phase, and the frequency offset of the four pulses it describes.

The pulse sequence of the experiment is composed of three elementary pulse sequences. The first and the third are incremented simultaneously, "growing" on both sides of the D0-sequence. Note that the duration of each pulse sequence is the same on both channels. The `-dw123` option directs the program to include all three sequences in the calculation of the mixing time that labels the data on the output (but has no effect on the simulation itself). Thus, the first point appears at 0.2 ms rather than at zero time.

The configuration of the two-spin system is specified in this example as the internuclear distance in the `atomic_coords` line. Hence, by default, the spins are aligned along the **z**-axis of the crystallite frame. Since the only anisotropic interaction present in the system is the dipolar coupling of the two nuclei (a symmetric tensor), the Hamiltonian is invariant with respect to rotations about the axis connecting the spins, which are generated by the α_{CR} angle. Therefore, powder averaging has to be performed only over the angles β and γ. As mentioned in the previous example, two-angle sets are usually the most efficient way of accomplishing that. Furthermore, the symmetry of the Hamiltonian is such that averaging only over a hemisphere is required [65]. However, the available two-angle sets restricted to the hemisphere do not seem to produce more accurate results than the full spherical sets with the same number of crystallites in this example. The `zcw987` set used in the example is sufficiently large to reproduce the time evolution up to about 10 ms (for the given coupling of 1.2 kHz, which can be examined by running the example with the `-s` option). For higher precision, averaging has to be done with more crystallites. The largest two-angle set included with the program is `shrewdstephemi5151` [65]. Results of even higher quality of the powder averaging, can be obtained by using automatically generated angle sets, for example:

```
EulerAngles      400
n_gamma          50
options       -dw123   -hemi
```

Note the `-hemi` option and the absence of the circumflex character. Although the powder averaging here is carried over 20,000 crystallites, the computation is only about twice as long as with the 5151 shrewd-step set because averaging over γ in this case is performed by an algorithm that takes advantage of the time-γ-translational symmetry of the Hamiltonian.

7.3. Heteronuclear decoupling: TPPM

Heteronuclear decoupling using the TPPM sequence [79] is an essential part of most MAS experiments performed with ^{1}H-containing samples. The sequence is a windowless train of constant-power, approximately 180° pulses with phases alternating between 0 and ϕ. When the sequence is applied experimentally, it is very important to tune it to obtain the narrowest lines possible, while being limited by the maximum RF power that can be tolerated by the probe or the sample. The main input file below simulates the effect of the decoupling field strength on the resulting line shape. This is accomplished by the explicit simulation of the FID:

```
(the lines containing only asterisks are not shown)

****** The System ***************************
spectrometer(MHz)  500
spinning_freq(kHz) 15.625
channels           C13  H1
nuclei             C13  H1  H1  H1  H1
atomic_coords      ch4.cor
cs_isotropic       0.0 -0.3 0.5 0.5 0.2
csa_parameters     ch4.csa
******* Pulse Sequence **********************
CHN 1
timing(usec)      (8)256x200
power(kHz)         0
phase(deg)         0
freq_offs(kHz)     0
CHN 2
timing(usec)      (4   4)
power(kHz)        125 125
phase(deg)         0   15
freq_offs(kHz)     0   0
****** Variables ****************************
variable           spinning_freq=15.625
scan_par           R/4.5:0.25:5.25/
variable           tc=(1000/spinning_freq)/R
variable           pulse_1_1_1=tc
variable           pulse_2_1_[1:2]=0.5*tc
variable           power_2_1_[1:2]=1000/tc
variable           DW=40*(1000/spinning_freq)
variable           gsize_1=round(DW/tc)
******* Options ******************************
rho0               1 0 0 0 0 Ix
observed_spins     1 Fp
EulerAngles        rep168.dat
n_gamma            16
line_broaden(Hz)   6
FFT_dimensions     1 Hz
```

The Pulse Sequence section shows that the sequence has a group size 200, so it is sampled once every 200 TPPM RF cycles, producing the desired spectral width; 256 points are collected in this way. For the computations to be efficient, the pulse sequence has to be rotor-synchronized (Eq. (37)). The n/m ratio is the number of RF cycles of the pulse sequence per one rotor period, and thus, for a fixed spinning frequency, uniquely determines the length of the RF cycle (the `tc` variable). In this example, the ratio is defined as the variable `R` (a scan parameter), which is made to vary from 4.5 to 5.25 in steps of 0.25. The pulse lengths and the RF power on the ^{1}H channel are recalculated for each of these values. To keep the dwell time (DW) constant, the sequence group size has to be adjusted as well, so the value specified for the group size in the Pulse Sequence section is actually disregarded. The powder averaging in this example is accomplished with a two-angle (α, β) set in combination with independent averaging over 16 γ values.

The line shapes simulated with this input file (Fig. 8) illustrate a very important phenomenon, the effect of which is often neglected: the decoupling becomes inefficient when the RF field coincides with a multiple of the spinning frequency. Furthermore, the peaks obtained with 70.3 and 74.2 kHz of decoupling power have nearly identical intensities.

7.4. Experiments with composite dimensions: HCCN dipolar correlation

The HCCN dipolar correlation experiment [81] simulated in this example was recently proposed to measure the ψ angle in α-helical polypeptides. This experiment has a relatively complex pulse sequence (Fig. 9A). As before, the parts of the experimental pulse sequence that are grayed out are assumed to behave ideally and are not simulated directly. The initial state is prepared by the cross-polarization transfer followed by the SELDOM filter [82]. The latter is intended to suppress the polarization of all nuclei that are not on-resonance with the applied frequency. This part of the experiment is assumed to result in the initial state given at the `rho0` line of the main input file. The I_x polarization of the first ^{13}C nucleus is then partially dephased by the REDOR sequence, transferred to the other ^{13}C nucleus with the HORROR recoupling sequence [83], and, finally, evolved under the Lee-Goldburg cross-polarization (LGCP) sequence [84], which reintroduces the CH dipolar coupling into the Hamiltonian. The HORROR period is kept constant, while the REDOR and LGCP periods are incremented simultaneously.

In the main input file below, the pulse sequence is composed of five elementary pulse sequences:

```
(the lines containing only asterisks are not shown)

****** The System ***************
spectrometer(MHz)  500
spinning_freq(kHz) 10
channels           C13 H1 N15
nuclei             H1 C13 C13 N15
atomic_coords      hccn.cor
tors_ang_nuclei    (1 2 3 4)
******* Pulse Sequence *********************************
CHN 1
timing(usec) (100)21 100 1 100 (100)21 (100)x7 (12.5)21
power(kHz)     0      0 500 0    0      5.0     70.0
phase(deg)     0      0  0  0    0       0        0
freq_offs(kHz) 0      0  0  0    0       0        0
CHN 2
timing(usec) (100)      200    (100)    100    (12.5)
power(kHz)    100       100     100     100     65.574
phase(deg)     0         0       0       0       0
freq_offs(kHz) 0         0       0       0      45.826
CHN 3
timing(usec) (redor1.pp) 200 (redor2.pp) 100    (12.5)
power(kHz)     *         0       *       0       0
phase(deg)     *         0       *       0       0
freq_offs(kHz) *         0       *       0       0
******* Variables ***********************************
scan_par        phi_1/-130:-10:-180/
variable        spinning_freq=12.9
variable        tsf_[1:5]=10/spinning_freq
variable        psf_1_[1:4]=spinning_freq/10
variable        psf_3_[1:4]=spinning_freq/10
variable        power_1_5_1=80-spinning_freq
******* Options **************************************
rho0            0 0 -1 0 Ix
observed_spins  2 Ix
EulerAngles     rep168.dat
n_gamma         16
options         -dw13
```

As follows from their timing format specifications, the pulse sequences 1, 3, and 5 are sampled in the same dimension (D1). The pulse sequences 2 and 4 are D0-se-

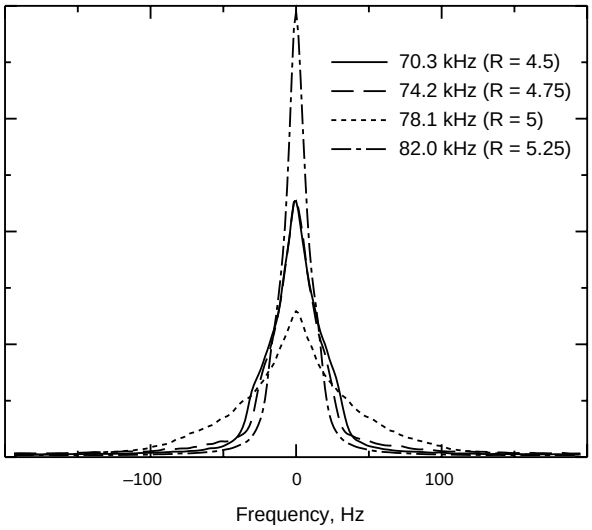

Fig. 8. ^{13}C line shape observed using TPPM decoupling at different RF field strengths. The broadest peak at $\omega_{RF}/2\pi = 78.1$ kHz corresponds to exactly five TPPM cycles per rotor period. TPPM conditions used: 180° pulses with 15° phase alternation.

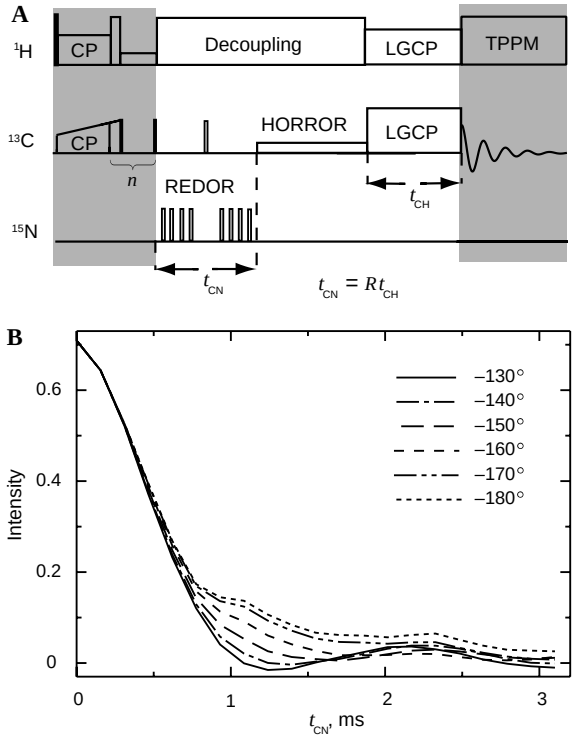

Fig. 9. The HCCN dipolar correlation experiment pulse sequence (A) and the set of dephasing curves (B) obtained for H–C_α–C′–N peptide fragments having different values of the H–C–C–N torsion angle φ_1. The peptide angle $\psi = \varphi_1 + 120°$.

quences; the latter is executed 7 times for every data point, as specified by its group size. The REDOR sequences (at the ^{15}N channel) are supplied as separate files:

```
redor1.pp:
50   0   0   0
1   500  0   0
50   0   0   0
1   500 90   0
```

```
redor2.pp:
1   500  0   0
50   0   0   0
1   500 90   0
50   0   0   0
```

Note that the π pulses in these sequences are simulated as ideal since RF power is greater than 499 kHz. All pulse lengths and powers in the Pulse Sequence section are given for the spinning frequency of 10 kHz. In the Variables section, these parameters are rescaled according to the spinning frequency used for the actual computation. The ^{1}H channel RF power during the LGCP sequence is independent of the spinning frequency and is set (in the Pulse Sequence section) to yield the effective RF field of 80 kHz. The corresponding power at the ^{13}C channel is set one spinning frequency below this value. The option `-dw13` directs the program to count only the first and the third sequences in the calculation of the evolution time that labels the data on the output of the calculation. The HCCN torsion angle is varied throughout the region of sensitivity of the experiment (which covers the conformations of the H-C_α-C′-N group in α-helical peptides) by means of the scan parameter `phi_1`. The resulting dephasing curves are shown in Fig. 9B.

7.5. Double-quantum filtering: SPC-5 and MELODRAMA

The effects of CSA are often neglected when dipolar recoupling sequences are designed. These effects may be very significant, however. The following example compares the efficiency of the double-quantum filters based on two dipolar recoupling sequences, i-MELO (also known as MELODRAMA-4.5), which is an unpublished version of the original MELODRAMA sequence [85] suggested by B. Sun, and SPC-5 [78], in the absence and in the presence of strong CSA interactions. From the viewpoint of the SPINEVOLUTION interface, this example illustrates the use of phase cycling and of the explicit coherence pathway selection as its alternative.

The homonuclear recoupling pulse sequences i-MELO and SPC-5 are shown in Fig. 10. The full RF cycle of i-MELO spans 16 rotor periods, while the RF cycle of SPC-5 spans only four rotor periods. The RF field power is constant during both sequences: it is $4.5\omega_R$ in the case of i-MELO, and $5\omega_R$ in the case of SPC-5. The shortest pulse is $\tau_R/9$ in i-MELO (the π pulse), and $\tau_R/20$ in SPC-5 (the $\pi/2$ pulse). All other pulses in both sequences are multiples of these two periods. It is important to keep these symmetry numbers (9 and 20) in mind when setting the number of averaging steps to take in the angle γ (given at the `n_gamma` line). For the simulation to be as efficient as possible, this number should be a multiple of 9 for the i-MELO, and a multiple of 20, for SPC-5. The input file for the i-MELO-based double-quantum filter is shown below:

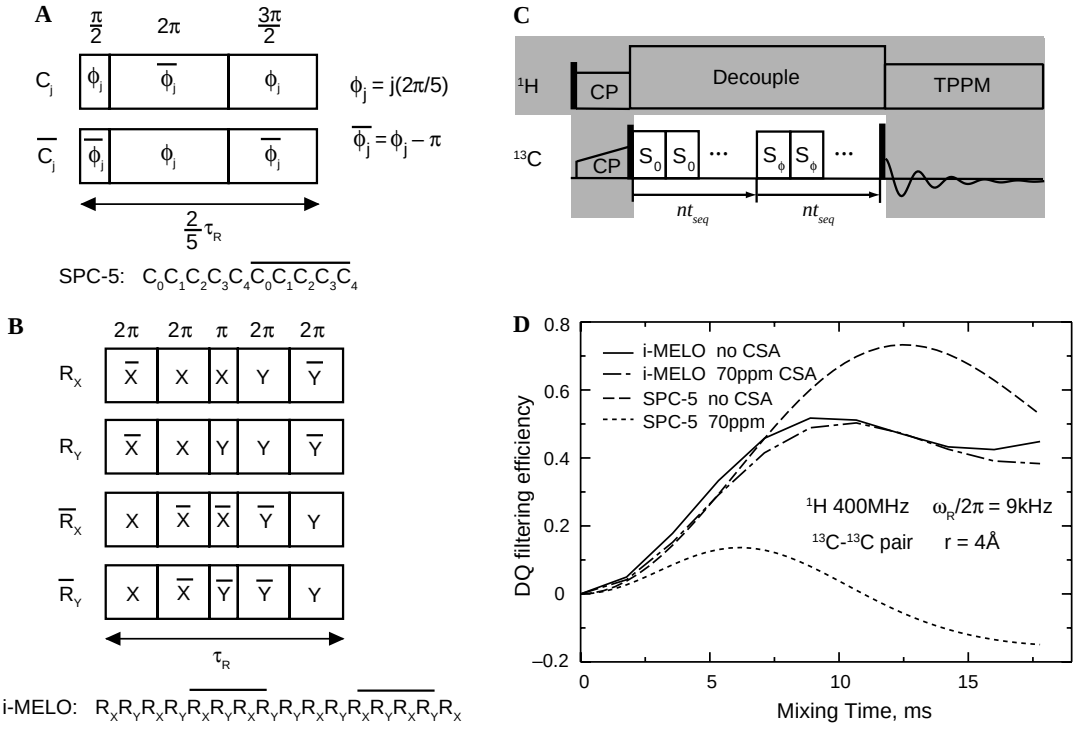

Fig. 10. The RF cycles of the homonuclear recoupling sequences SPC-5 (A) and i-MELO (B). The layout of a double-quantum filter sequence (C); S_ϕ is the RF cycle of the sequence (i-MELO or SPC-5); the double-quantum coherence pathways are selected by phase cycling ϕ (the phase of the sequence during the reconversion period), the phase of the final $\pi/2$ pulse, and the receiver phase. (D) The efficiencies of the double-quantum filters based on i-MELO and SPC-5 computed for a weakly coupled spin pair.

```
(the lines containing only asterisks are not shown)
****** The System ******************************
spectrometer(MHz)  400
spinning_freq(kHz) 10.0
channels           C13
nuclei             C13 C13
atomic_coords      4
csa_parameters     1 70 0.8  0 60 0 ppm
csa_parameters     2 70 0.8 30 90 0 ppm
*** i-MELO = R1R2R1R2 R3R4R3R4 R2R1R2R1 R4R3R4R3
CHN 1
timing(usec) (imelo.pp)11 (imelo.pp)11 0.5
power(kHz)      *            *         500
phase(deg)      *            *         90
freq_offs(kHz)  *            *         0
phase_cycling   *          1234        * 3131(RCV)
******* Variables ******************************
variable           spinning_freq=9
variable           psf_1_[1 2]=0.09*spinning_freq
variable           tsf_[1 2]=1/psf_1_1
******* Options ********************************
rho0               1 1 Iz
observed_spins     1 Ix
EulerAngles        rep168.dat
n_gamma            9
```

The pulse programs of both sequences are supplied as separate files, which are not shown, but can be reconstructed from Figs. 9A–B. The i-MELO pulse program was composed assuming the τ_R of 90 μs, while the SPC-5 pulse program was composed assuming $\tau_R = 100$ μs, both of which result in the 50 kHz RF field and pulses lasting a whole number of microseconds. These choices are very convenient for composing and inspecting these pulse programs. For the simulations at other spinning frequencies, the pulse programs are rescaled using the power and time scaling factors in the Variables section.

The corresponding file for SPC-5 is obtained by substituting the following lines,

```
timing(usec)    (spc5.pp)41  (spc5.pp)41 0.5
variable        psf_1_[1 2]=0.1*spinning_freq
n_gamma         20
```

in place of the similar lines in the i-MELO file. Note the four-fold increase in the number of sampling points (the RF cycle of SPC-5 is four times shorter than that of i-MELO) and a different scaling.

The results of the simulations are shown in Fig. 10D. As one can see, although performing much better than i-MELO in the absence of CSA, SPC-5 fails in the presence of large CSA interaction, while the performance of i-MELO is practically unaffected. The same input files can be used to also show that for directly bonded nuclei, SPC-5 outperforms i-MELO even in the presence of large CSA.

The phase cycling used in the example is self-explanatory. It should be remembered, however, that performing the phase cycling in this manner mimics the actual experiment and thus requires the explicit simulation of each

step in the phase cycle. For long phase cycles, it may slow down the simulation significantly. To avoid this effect or to verify that the phase cycle chosen works as expected, one can often use the explicit coherence selection instead:

```
CHN 1
timing(usec)       (spc5.pp)41 0 (spc5.pp)41  0.5
power(kHz)          *          0  *           500
phase(deg)          *          0  *           -90
freq_offs(kHz)      *          0  *             0
******* Variables ********************************
variable            spinning_freq=9
variable            psf_1_[1 3]=0.1*spinning_freq
variable            tsf_[1 3]=1/psf_1_1
rowmatrix           select_2/+2 -2/
```

Note the additional zero-length pulse sequence inserted between the two halves of the experiment. The ± 2 coherences are selected during this sequence, the rest are destroyed directly in the density matrix during the propagation.

7.6. Two-dimensional MAS experiments: chemical shift correlation

This example was described earlier, in the section dealing with the performance of the program (Fig. 5). The pulse sequence of the experiment (Fig. 11) is a typical example of the preparation-(t_1-evolution)-mixing-(t_2-evolution) layout. Coherence pathway selection is accomplished by means of a z-filter, i.e., by annihilation of all off-diagonal elements of the density matrix. The main input file for this simulation is shown below.

```
(the lines containing only asterisks are not shown)
****** The System ***********************************
spectrometer(MHz)   400
spinning_freq(kHz)  8.0
channels            C13
nuclei              C13 C13 C13 C13 C13 C13
atomic_coords       leu.cor
cs_isotropic        leu.cs
j_coupling          leu.j
******* Pulse Sequence *********************************
CHN 1
timing(usec) (0)2048D1 0.5 (0) (rfdr8.pp)x2 (0) 0.5 (0)1024D2
power(kHz)     0       500  0    *           0  500  0
phase(deg)     0       270  0    *           0  90   0
freq_offs(kHz) 0       0    0    *           0  0    0
******* Variables ************************************
variable            spinning_freq=8
variable            taur=1000/spinning_freq
variable            pulse_1_1_1=taur/6
variable            pulse_1_7_1=taur/3
variable            tsf_4=taur/64
variable            psf_1_4=1/tsf_4
variable            zfilter_[3 5]=1
variable            frs_1_[1 4 7]=[-16 -4 -16]
******* Options **************************************
rho0                Ix
observed_spins      1 Fx
EulerAngles         rep100.dat
n_gamma             12
line_broaden(Hz)    0 0 60 60
FFT_dimensions      1c 2
```

The RF cycle of the RFDR sequence used for mixing is given is a separate file `rfdr8.pp`:

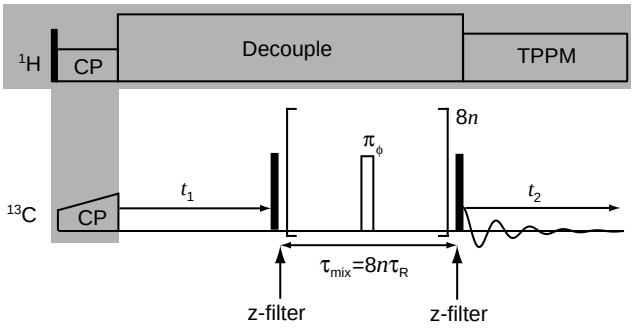

Fig. 11. The pulse sequence of the 2D chemical shift correlation experiment. RFDR recoupling sequence [86] (with XY-8 [87] phase alternation for ϕ) is employed for mixing. The 2D spectrum obtained in the simulation of this experiment is given in Fig. 5 together with the details of the simulation.

```
30   0    0    0
4    125  0    0
60   0    0    0
4    125  90   0
60   0    0    0
4    125  0    0
60   0    0    0
4    125  90   0
60   0    0    0
```

```
4    125  90   0
60   0    0    0
4    125  0    0
60   0    0    0
4    125  90   0
60   0    0    0
4    125  0    0
30   0    0    0
```

To obtain a pure-phase spectrum, the cosine transform is used in the indirect dimension instead of the Fourier transform. Since the cosine transform cannot distinguish between positive and negative frequencies, the spectrum has to be recorded in such a way that all its frequencies have the same sign. In the present example, this is accomplished with the help of the frequency shift variables. The -16 kHz RF offset in both dimensions moves the entire spectrum (specified in the `leu.cs`) into the positive frequencies quadrant, while the -4 kHz offset places the RF into the middle of the spectrum during RFDR (see Eq. (25)).

7.7. Non-periodic problems, gradients, and relaxation: CW NMR

From the experimental viewpoint, continuous wave (CW) NMR is very different from all other examples we have considered so far. From the computational viewpoint, however, the difference is not so vast and consists mostly in the absence of any kind of periodicity in this problem. A CW scan can be done by varying either the static magnetic field or the irradiation frequency. To scan the field in SPINEVOLUTION, one can use the PFG slice simulation feature. In the input file below, the bracketed notation in the Pulse Sequence section is a macro (see `timing(usec)`, Table A1) that expands into a sequence of 5001 identical pulses, 100 ms each:

```
(the lines containing only asterisks are not shown)
****** The System ********************************
channels            C13
nuclei              C13
******* Pulse Sequence ***************************
CHN 1
timing(usec)        500100000
power(kHz)          0.0001
phase(deg)          90
freq_offs(kHz)      0
CHN G
timing(usec)        [100000]5001
gradient(Gs/cm)     [0]5001
******* Variables ********************************
variable T1SQ_1_1=2000
variable T2SQ_1_1=2000
variable grad_offs=1
variable grad_1_[1:5001]=0.025*([-1:0.0004:1])/1070.8
******* Options **********************************
rho0                0.25 Iz
observed_spins      1 Ip
options             -oes -dt10000
```

Declaration of the variable `grad_offs` with the value 1 directs the program to perform the computations for only one slice of the sample, 1 cm away from the center of the gradients. Hence, the pulses at the PFG channel are simply shifting the field in the observed part of the sample. In our example, this shift is ramped from −25 Hz for the first pulse, to +25 Hz for the last. The variable `grad_s_p` sets the gradient in Gs/cm, so the factor 1070.8 (kHz/Gs) is required to convert the values expressed in kHz. Note that no variables are actually scanned in this example. Rather, the option `-oes` (*o*bserve *e*ach *s*tep) tells the program to take a data point after each pulse. This setup leads to the explicit integration of the Bloch equations, with the result shown in Fig. 12.

Both T_1 and T_2 are set to 2 s; the RF power is set to approximately maximize the signal at these relaxation rates. Since the longitudinal relaxation is present, it is important to set the initial density matrix equal to the equilibrium density matrix (Eq. (38)), hence the 0.25 factor in the `rho0` line. The field is swept at the rate of 0.1 Hz/s, leading to the characteristic "ringing" the signal (Fig. 12) with the relaxation parameters used.

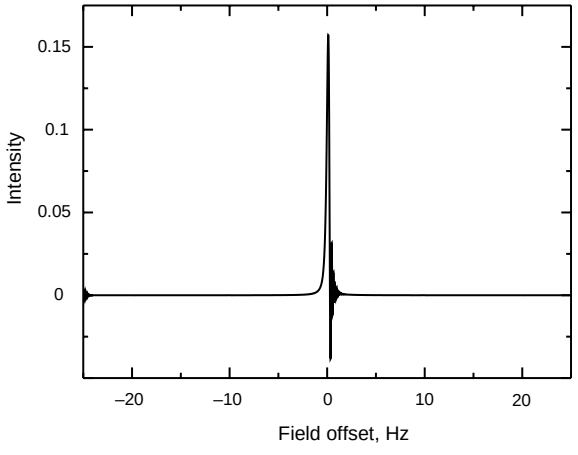

Fig. 12. Continuous-wave NMR scan of a single spin.

The option `-dt10000` limits the integration step to 10 ms (for static problems, the default is the full length of the pulse, 100 ms in this case). This step should be sufficiently small so that relaxation could be neglected on the time scale of a single step.

Exactly the same results could be obtained by sweeping the irradiation frequency instead of the field. The Pulse Sequence and the Variables section of the corresponding input file are shown below:

```
******* Pulse Sequence ************************
CHN 1
timing(usec)     [100000]5001
power(kHz)       [0.0001]5001
phase(deg)       [90]5001
freq_offs(kHz)   [0]5001
******* Variables *****************************
variable T1SQ_1_1=2000
variable T2SQ_1_1=2000
variable freq_1_1_[1:5001]=0.025*([-1:0.0004:1])
```

7.8. Averaging over a distribution: Carr–Purcell echo train

The main point of this example is to illustrate the use of the average-over parameters. Although the pulse sequence of the Carr–Purcell–Meiboom–Gill train [88,89] is periodic, the observation of the echoes requires sampling more than once per sequence RF cycle. In the current version of SPINEVOLUTION, this feature is not functional yet. Hence, the pulse sequence must be treated as a general non-periodic sequence. The input file for the simulation is shown below.

```
(the lines containing only asterisks are not shown)
****** The System ****
channels            C13
nuclei              C13
******* Pulse Sequence **********
CHN 1
timing(usec)        0.5 [1000]2900
power(kHz)          500 [0]2900
phase(deg)           90 [0]2900
freq_offs(kHz)        0 [0]2900
******* Variables *****************************
variable            T2SQ_1_1=1000
variable            sigma=0.006
variable            pulse_1_1_[100:200:2900]=1.0
variable            power_1_1_[100:200:2900]=500
ave_par             x/-0.02:0.0025:0.02/
variable            cs_iso_1=0.1+x
variable            A=0.0025/sqrt(2*pi)/sigma
variable            ave_wht=A*exp(-0.5*(x/sigma)^2)
******* Options *******
rho0                1 Iz
observed_spins      1 Ix
options             -oes
```

With the option `-oes`, the magnetization is observed after each pulse. Every 200-th pulse is turned into an ideal π pulse in the Variables section. The pulses refocus evolution due to the chemical shift, which is given a Gaussian distribution with $\sigma = 6$ Hz. Some care must be taken to make the sum of all weights of the distribution equal to 1. The factor 0.0025 in front of the normalized Gaussian

probability distribution comes from the conversion of the integral over the probability density to the sum of discrete weights (the distribution is sampled with $\Delta x = 0.0025$). Averaging the results over this distribution produces the well-known echo train shown in Fig. 13.

In the upcoming versions of the program, one will be able to perform multiple observations per RF cycle, in which case the pulse sequence of the experiment should be described as follows:

```
timing(usec)        0.5 (100000 1 100000)3000g200
power(kHz)          500     0    500    0
phase(deg)           90     0      0    0
freq_offs(kHz)        0     0      0    0
```

7.9. Optimization problems: design of a selective excitation pulse

This example demonstrates the use of SPINEVOLUTION for solving optimization and data fitting problems. It also illustrates the use of matrix variables. Input files very similar to the one in this example were used to design a family of high-performance excitation pulses known as the E-Family [5].

The timing of the pulse sequence in this example is set up with the help of a "timing program" `t100.tm`, which is simply a column consisting of 100 copies of the number 100. This creates a pulse sequence consisting of 100 pulses, 100 μs each.

```
(the lines containing only asterisks are not shown)

****** The System ****************************
channels            C13
nuclei              C13
******* Pulse Sequence ************************
CHN 1
timing(usec)        t100.tm
power(kHz)          0
phase(deg)          90
freq_offs(kHz)      0
******* Variables *****************************
scan_par            cs_iso_1/0:0.025:5/
colmatrix           A/0:0:15/
colmatrix           B/0:0:15/
rowmatrix           n/1:15/
colmatrix           t/50:100:9950/
variable            C=cos((2*pi/10000)*t*n)
variable            S=sin((2*pi/10000)*t*n)
variable            p=0.1*(A0+C*A+S*B)
variable            power_1_1_[1:100]=p([1:100])
variable            RFC_TOL=0.001
variable            RDX_FDJ=0.01
fit_par             A0 A([1:15]) B([1:15])
******* Options etc ****************************
rho0                1 Iz
observed_spins      1 Ip
options             -varselpul0.par
```

The pulse shape $\omega_{\mathrm{RF}}(t)$ is parametrized by the coefficients of the truncated Fourier series:

$$\omega_{\mathrm{RF}}(t)/\omega = A_0 + \sum_{k=1}^{n}\{A_k \cos(k\omega t) + B_k \sin(k\omega t)\}, \qquad (39)$$

where $\omega = 2\pi/\tau$, and τ is the excitation time, i.e., the total duration of the shape; in our case, $\tau = 10$ ms. The pulse shape is computed according to Eq. (39) from the column-matrices `A` and `B` formed by the Fourier coefficients, and the square matrices `C` and `S` formed by the cosines and sines. The starting point for the optimization is given in the file `selpul0.par`

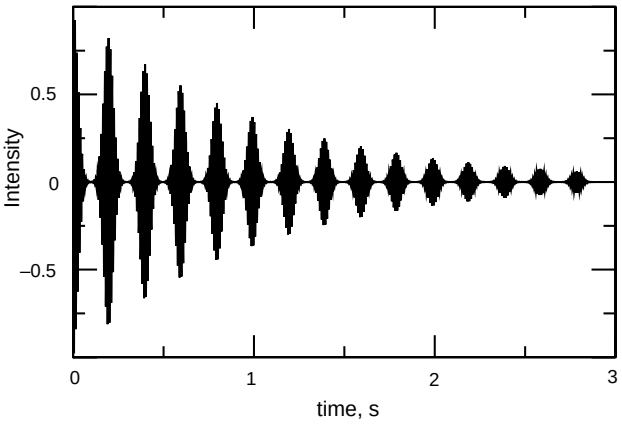

Fig. 13. Carr–Purcell–Meiboom–Gill echo train in an inhomogeneous magnetic field.

```
A0=-0.75
A(1)=-0.5
A(2)=2
A(3)=-1
```

Declaring these variables with the command line option `-varselpul0.par` is often a convenient alternative to the `variable selpul0.par` statement in the Variables section, because different starting point files may be tried in this way without editing the main input file. The fit parameters that have not been initialized explicitly in the `selpul0.par` file are initialized with zeros by default.

The chemical shift of the spin is declared as a scan parameter. Thus, the pulse shape will be executed for all offsets given in the list of values for this parameter, i.e., from 0 to 5 kHz, in steps of 25 Hz. Each execution gives one data point. The collection of all these points gives the excitation profile of the pulse shape. The target excitation profile is given by the files `selpul_re.fit`, `selpul_re.wht`, `selpul_im.fit`, and `selpul_im.wht`, and has a rectangular shape with a zero-weight transition zone extending from 0.05 to 0.2 kHz:

```
selpul_re.fit:     selpul_im.fit:     selpul_re.wht:
                                      selpul_im.wht:
0.000  1.0         0.000  0.0         0.000  1.0
0.025  1.0         0.025  0.0         0.025  1.0
0.050  1.0         0.050  0.0         0.050  1.0
0.075  0.0         0.075  0.0         0.075  0.0
…                  …                   …
                                       0.175  0.0
                                       0.200  1.0
                                       …
5.000  0.0         5.000  0.0         5.000  1.0
```

Once the optimization is started, SPINEVOLUTION finds the best-fit parameters (the E100A member of the E-Family [5]) within a few seconds, and saves them in

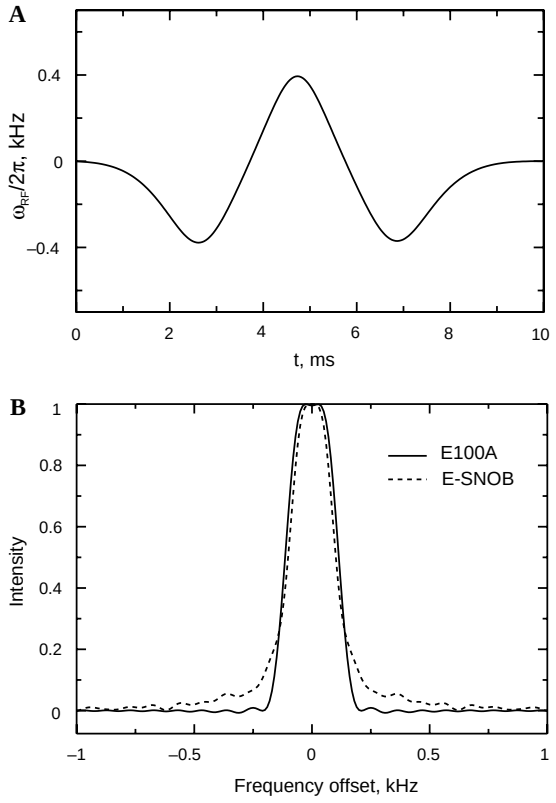

Fig. 14. (A) The selective excitation pulse obtained by the optimization procedure of Example 9. The pulse coincides with the E100A member of the E-Family [5]. (B) The real part of the excitation profiles of the pulse shown in (A) and of the e-SNOB pulse [90].

the file named `selpul.par`. The excitation profile of the resulting pulse can be computed in a separate simulation, where this file is used instead of `selpul0.par` and the main input file is edited to comment out the contents of the `fit_par` line in order to turn off the optimization. A more appropriate value list for the `cs_iso` parameter scan should also be used (e.g., `-2:0.01:2`). The pulse shape of the resulting pulse can be obtained with the help of the `-pwr` option (see Table A3). The real part of the excitation profile and the pulse shape are shown in Fig. 14. The quality of the excitation profile exceeds by far the quality of the excitation profiles of any other analogous pulse known prior to design of the E-Family.

8. Conclusions

SPINEVOLUTION is a state-of-the-art computer program able to meet a wide range of needs arising in the context of exact simulations in NMR. One of the greatest advantages of SPINEVOLUTION is its efficiency, which is particularly remarkable for computations involving large spin systems. In a head-to-head comparison of SPINEVOLUTION with SIMPSON on a series of test examples, the former was consistently found several orders of magnitude faster than the latter, depending on the spin system size and on the type of the experiment. The other advantage of the program is the simplicity of use. In particular, the interface is based on a natural, non-algorithmic description of the NMR pulse sequences (“the canonical representation”), which is also well suited as the framework for the construction of efficient simulations of complex pulse sequences, and can be explored as the language for spectrometer interfaces. For researchers working in solid-state NMR, SPINEVOLUTION should be of great utility as a routine simulation tool for use in conjuction with running actual experiments on the spectrometer, as well as for the design and optimization of new experiments, theoretical research, data fitting, and other purposes. Liquid state spectroscopists will find it particularly useful for solving various optimization problems. With its simplicity of use, SPINEVOLUTION should broaden the community of people who use NMR simulations in their research. With its high performance and ability to deal with large spin systems on a reasonable time scale, it makes possible to use exact numerical simulations in situations not previously amenable to this approach.

Acknowledgments

We thank our colleagues Dr. Vladimir Ladizhansky, Dr. David Ruben, Dr. Jonathan Lansing, and Mr. Vik Bajaj for their stimulating conversations and suggestions during the course of the development of SPINEVOLUTION. This research was supported by grants from the National Institutes of Health (EB-002026, EB-003151, and EB-001960).

Table A1
The main input file formats

System	
spectrometer(MHz)	*freq* ^{1}H frequency of the spectrometer, MHz. The default is 500 MHz.
spinning_freq(kHz)	*freq* Sample spinning frequency, kHz. The default is zero.
channels	Nuc_1 Nuc_2 … Nuc_M
Examples:	`C13 H1` `A(100 -5/2) B(200) F19`

(*continued on next page*)

Table A1 (*continued*)

The sequence of names of the nuclear species participating in the experiment. The order in which the nuclear species appear at this line is used to assign a number (from 1 to M) to every channel. A nucleus that is unknown to the program can be introduced on this line as *Name*(*freq* [*I*]), where *freq* is the (unsigned) frequency of the nucleus, in MHz, and I is its signed spin quantum number (which can be omitted if equal to 1/2).

nuclei $spin_1\ spin_2\ \ldots\ spin_N$

Example:
```
C13 H1 C13 H1 H1
```
An ordered sequence of names that specifies the type of each spin in the system. This line assigns each spin to one of the nuclear species declared at the previous line. All spins in the system are numbered from 1 to N in the order they appear at this line.

atomic_coords *filename* OR r

Examples:
```
ch3.cor
```
```
1.52
```
Atomic coordinates file name. The file[e] should contain the triplets of atomic coordinates (crystallite frame), in Ångströms, one per line, given for each nucleus in order from 1 to N. The coordinates are used by the program to compute the dipolar interactions in the system. For a 2-spin system, the internuclear distance may be given instead of the file name, in which case the two nuclei are placed along the **z**-axis of the crystal, with the first nucleus placed at the origin of coordinates, and the second—at the $(0, 0, r)$ point. If neither *filename* nor r are specified, all dipolar interactions in the system are turned off.

cs_isotropic $\Omega^1_{\rm iso}/2\pi\ \ \Omega^2_{\rm iso}/2\pi\ \ \ldots\ \ \Omega^N_{\rm iso}/2\pi$ [**kHz**|**Hz**] OR
$\delta^1_{\rm iso}\ \ \delta^2_{\rm iso}\ \ \ldots\ \ \delta^N_{\rm iso}$ **ppm** OR
filename [**ppm**|**Hz**|**kHz**]

Examples:
```
0 100
```
```
ch3.cs ppm
```
Isotropic chemical shifts or chemical shift offsets, given for each nucleus in order from 1 to N. The default units for the offsets are kHz. The values can be specified directly on the line, or in a separate file.[e] The values in the file should be arranged one per line. The default values are all zero. Note that for $\gamma > 0$ nuclei, $\Omega^i_{\rm iso}$ and $\delta^i_{\rm iso}$ have opposite signs.

csa_parameters $i\ \ \{\Omega^i_{\rm aniso}/2\pi \mid \delta^i_{\rm aniso}\}\ \ \eta^i_\Omega\ \ \alpha^i_{\Omega,PC}\ \ \beta^i_{\Omega,PC}\ \ \gamma^i_{\Omega,PC}$ [**kHz**|**Hz**|**ppm**] OR
filename [**ppm**|**Hz**|**kHz**]

Examples:
```
1  2.3  0.85  45  90  180
```
```
ch3.csa ppm
```
The CSA tensor parameters[f] of the ith spin. By default, the value of $\Omega^i_{\rm aniso}/2\pi$, kHz, is expected at this line; however, it will be interpreted as $\delta^i_{\rm aniso}$ if the option **ppm** is specified (tensors $\mathbf{\Omega}^i$ and δ^i have the same parameters except for the anisotropy). The values can be specified directly on the line, or in a separate file[e]. In the file, the values should be arranged six per line. The **csa_parameters** line has to be repeated as many times as necessary to specify all CSA interactions present in the system. The default values are all zero.

j_coupling $i\ \ j\ \ J^{ij}_{\rm iso}$ OR
$i\ \ j\ \ J^{ij}_{\rm iso}\ \ J^{ij}_{\rm aniso}\ \ \eta^{ij}_J\ \ \alpha^{ij}_{J,PC}\ \ \beta^{ij}_{J,PC}\ \ \gamma^{ij}_{J,PC}$ OR
filename

J-coupling parameters[f] of spins i and j; $J^{ij}_{\rm iso}$ and $J^{ij}_{\rm aniso}$ should be given in Hz. The values can be specified directly at the line or in a separate file.[e] In the file, the values should be arranged either three or eight in every line. The **j_coupling** line has to be repeated as many times as necessary to specify all j-couplings present in the system. The default values are all zero.

quadrupole $i\ \ \chi^i\ \ \eta^i_V\ \ \alpha^i_{V,PC}\ \ \beta^i_{V,PC}\ \ \gamma^i_{V,PC}$ OR *filename*

Quadrupolar interaction parameters[f] of the spin i; χ^i should be given in kHz. The values can be specified explicitly on the line or in a separate file.[e] In the file, the values should be arranged six per line. The **quadrupole** line has to be repeated as many times as necessary to specify all quadrupolar interactions in the system. The default values are all zero. Quadrupolar interactions are not functional in the current version of the program.

dip_switchboard *filename* OR $filename_1\ filename_2\ \ldots\ filename_S$

Dipolar switchboard. If just one file name is given on the line, it turns on/off the dipolar couplings between the nuclei during the whole experiment. If multiple file names are given, the board switches the couplings during each pulse sequence independently. Each file should be formatted as the lower triangular matrix containing 1 to turn the corresponding coupling on and 0 to turn it off, e.g., the matrix

```
*
1 *
0 1 *
```
turns on the dipolar interaction for the `(1,2)` and `(2,3)`spin pairs and off for the `(1,3)` spin pair. The asterisks are treated as comments. If no file names given on the line, all dipolar interactions are on.

csa_switchboard *filename* OR $filename_1\ filename_2\ \ldots\ filename_S$

The analogous option for the CSA interaction, except that a sequence of 1's and 0's should be specified instead of a matrix in each file.

Table A1 (*continued*)

exchange_nuclei $(i_1^{(1)}\ i_2^{(1)}\ldots)^{g}\ (i_1^{(2)}\ i_2^{(2)}\ldots)^{g}\ldots$

Example: `(2 3 4)`

Each group of spins in the parentheses undergoes fast regime exchange through *cyclic permutation* (e.g. methyl group hopping).

bond_len_nuclei $(j_1^{(1)}\ j_2^{(1)}\ j_3^{(1)}\ldots)^{g}\ (j_1^{(2)}\ j_2^{(2)}\ j_3^{(2)}\ldots)^{g}\ldots$

Example: `(2 1) (2 3 5:7)`

Sequences of nuclei defining the meaning of the internal variables `r_1`, `r_2`, etc. (bond lengths). These variables can be used to set the internuclear distances in the spin system. When the variable **r**_*k* is applied to the structure, all nuclei in the *k*th group, except for the 1st, are translated (as a group) in the 1–2 direction as much as necessary to produce the specified distance between the 1st and the 2nd nuclei.

bond_ang_nuclei $(k_1^{(1)}\ k_2^{(1)}\ k_3^{(1)}\ldots)^{g}\ (k_1^{(2)}\ k_2^{(2)}\ k_3^{(2)}\ldots)^{g}\ldots$

Example: `(3:1) (2:5)`

Sequences of nuclei defining the meaning of the internal variables `theta_1`, `theta_2` etc. (bond angles). These variables can be used to set the bond angles in the spin system. When the variable **theta**_*k* is applied to the structure, all nuclei in the *k*th group, except for the 1st and the 2nd, are rotated (as a group) about the axis passing through the 2nd nucleus perpendicular to the 1–2–3 plane as much as necessary to produce the specified 1–2–3 angle. The CSA and quadrupolar coupling tensors of the 3rd, 4th, etc. nuclei are rotated together with the nuclei.

tors_ang_nuclei $(l_1^{(1)}\ l_2^{(1)}\ l_3^{(1)}\ l_4^{(1)}\ldots)^{g}\ (l_1^{(2)}\ l_2^{(2)}\ l_3^{(2)}\ l_4^{(2)}\ldots)^{g}\ldots$

Example: `(1 2 3 4 5) (2 3 4 5)`

Sequences of nuclei defining the meaning of the internal variables `phi_1`, `phi_2`, etc. (torsion angles). The first four nuclei of each sequence define a torsion (dihedral) angle in the molecule. The rest of the sequence defines the set of nuclei rotated together with the 4th nucleus when the angle is changed. The torsion angle is defined as zero for the *cis*-configuration of the 1–2–3–4 chain and is increased as the 4th, 5th, etc. nuclei are rotated (as a group) around the 2–3 bond counterclockwise looking in the $3 \rightarrow 2$ direction. The CSA and quadrupolar coupling tensors of the 3rd, 4th, etc. nuclei are also rotated as the torsion angle is changed.

groups_nuclei $(m_1^{(1)}\ldots)^{g}\ (m_1^{(2)}\ldots)^{g}\ldots$

Example: `(1) (2:4 8) (5 6)`

Each sequence in parentheses defines a group of nuclei that can be rotated and translated as a whole. When the transformations are applied, the *k*th group is rotated about the origin of coordinates through the Euler angles **group**_*k*_**R**, and then translated by the vector **group**_*k*_**T**. The CSA and quadrupolar coupling tensors are rotated together with the nuclei.

Pulse Sequence

CHN *n* OR **G**

The channel number (same as the nuclear species number). This subsection of the main input file should appear once for every channel. Following the regular channels, a PFG channel may be declared by putting the letter G instead of the channel number.

timing(usec) $(X_1^{(1)}\ X_2^{(1)}\ldots)$[[–]*N*][**x***G*|**g***g*][**D***dim*] … (CHN 1 only)

$(X_1^{(1)}\ X_2^{(1)}\ldots)$ … (other channels)

Examples:

```
(100)61       95  5    100      (100)61
(redor.pp)  45 5 50 redor.pp    (redor.pp)
(100)-20D1  (50 50)20x2D1       (500)1024g16D2
```

The timing and the sampling pattern of each elementary pulse sequence of the experiment. Any $X_i^{(s)}$ can be either a numerical constant specifying the duration of a pulse in the pulse sequence *s*, or a file name. If $X_i^{(s)}$ is a file (a pulse program), its lines should be of the form:

duration power phase frequency

describing each pulse in the group of pulses $X_i^{(s)}$. Alternatively, the file may contain just one column, describing the duration of each pulse in $X_i^{(s)}$.

In general, the number of pulses in a given pulse sequence and their durations are different for every channel, but the total duration of the pulse sequence cycle must be the same for all channels. The sampling patterns for each sequence (which are also the same on all channels) are given at the timing line of the first channel only. In the pattern specification, *dim* is the dimension, in which the pulse sequence is sampled (simultaneously with the other pulse sequences of this dimension). *N* is the number of the sampling points to obtain; it should be the same for all pulse sequences of the dimension. All D1 and D2 pulse sequences must be rotor-synchronized (Eq. (37)).

The sampling pattern of a pulse sequence is given as either **x***G* or **g***g* (or omitted if $g = G = 1$), where *G* is the group size, and *g* is the sampling rate. The sequence is either executed in groups of size *G*, i.e. sampled after every *G* cycles of the sequence, or sampled *g* times per each cycle. The minus sign is placed before *N* for decremented pulse sequences.

(*continued on next page*)

Table A1 (*continued*)

The default value for N, G, and g is 1. If $N = 1$, the default value of *dim* is 0, but if $N > 1$, then the default value of *dim* is 1. For example, the pattern specification may be omitted altogether for a D0-sequence with $G = 1$. The parentheses enclosing the pulse widths may be also omitted in this case if the adjacent pulse sequences are parenthesized. Thus, each of the three examples above describes exactly three elementary sequences.

A pulse may be specified as ideal by putting an apostrophe mark immediately after its duration or by setting its power above the ideal-pulse threshold (see –**id** option). Note that the actual duration of an ideal pulse in the pulse sequence is zero.

The following notation may be used to generate repeated pulse sequence elements:

[*string*] *rep*

This is essentially a macro that will be expanded as the string *string* repeated *rep* times, using spaces as delimiters. For example, `([45 5]100)200` is equivalent to `(45 5 45 5 ...)200` where the sequence "`45 5`" is repeated 100 times in the parentheses.

power(kHz) $W_1^{(1)}\ W_2^{(1)}\ldots$

phase(deg) $P_1^{(1)}\ P_2^{(1)}\ldots$

freq_offs(kHz) $F_1^{(1)}\ F_2^{(1)}\ldots$

Power, phase, and frequency offset of each pulse ($\omega^k_{\mathrm{RF}}(t)/2\pi$, $\phi_k(t)$, and $\omega^k_{\mathrm{off}}(t)/2\pi$, respectively; Eqs. (24) and (25)). Every entry $X_i^{(s)}$ on the **timing(usec)** line of a given channel requires exactly one entry on each of the power/phase/frequency lines of this channel. If a pulse length is given at the timing line, the corresponding entry has to be a numerical constant. However, if $X_i^{(s)}$ is a pulse program, the entry in the power/phase/frequency line can be either of the three: (1) an *asterisk*, which instructs to load the values from the pulse program; (2) a *numerical constant,* which sets constant power, phase or frequency offset during the whole pulse program, possibly overriding the values given in the file; (3) a *file name* of another pulse program, from which the sequence of powers, phases or frequency offsets should be loaded. The latter file may be either in the one-column format containing the powers/phases/frequencis of all pulses, or in the 4-column pulse program format (see **timing(usec)** line). In the later case, only the required column from this file will be loaded. The macro with the brackets described above (see **timing(usec)** line) is allowed in the power/phase/frequency lines as well.

Rather than specifying the desired numerical values describing the RF pulses in the Pulse Sequence section directly, it is often more convenient to compute the values in the Variables section of the file. In this case, any numerical constants (e.g., zeros) can be used at the power/phase/frequency lines for the values that will be recalculated in the Variables section.

If the power given is higher than 499 kHz the pulse is assumed to be an ideal delta-pulse with the same rotation angle it would have if it were a regular pulse (this behavior may be changed with the help of the –**id** command line option). Alternatively, a pulse may be specified as ideal by putting an apostrophe mark immediately after its duration at the **timing(usec)** line.

The phases are interpreted as given in the frame of reference that continuously rotates at the instantaneous RF frequency throughout the duration of the entire RF path.

gradient(Gs/cm) $G_1^{(1)}\ G_2^{(1)}\ldots$

The field gradient of each gradient pulse (PFG channel only). The syntax is the same as with the power/phase/frequency lines above. This line must follow the timing line for the PFG channel.

phase_cycling $PC_1\ PC_2\ldots PC_S\ PC_{RCV}$(**RCV**)

Example: `1234 x2.pc * 4321(RCV)`

The phase cycles of each elementary pulse sequence and of the receiver. An m-step phase cycle PC_s is defined by the set of m phase shifts $i_k^{(s)}\left(1 \leqslant i_k^{(s)} \leqslant n, k = 1\ldots m\right)$. At the step k of the phase cycle, the phases of *all* pulses of the pulse sequence s are shifted from their base values by $\left(i_k^{(s)} - 1\right)\cdot 360^\circ/n$. (The base values are those set at the **phase(deg)** lines or through the variables.) The phase cycle can be specified either explicitly, as a group of digits $i_1^{(s)}\ldots i_m^{(s)}$, or loaded from a file if the file name is given. In the file, the phase shifts must be separated by spaces and/or new line characters. If a pulse sequence is not phase cycled ($i_k^{(s)} = 1$ for all k), an *asterisk* can be used to specify its phase cycle. The default value of n is 4; a different value can be set with the command line option –**pc***n*. For $n > 9$, the phase cycles can be given only through files (since $i_k^{(s)}$ have to be single-digit numbers to be specified explicitly). Note that each pulse sequence is phase-cycled as a whole. The **phase_cycling** line is optional.

Variables

scan_par *name*[a]/*list-of-values*[b]/...

Example: `A/-0.5:0.025:0.5/ B/B.lst/`

A scan parameter. The simulation will be performed for every value of the parameter from the given *list-of-values*. Every scan parameter adds one dimension to the experiment. A maximum of two dimensions, including the dimensions generated by the pulse sequence, is currently allowed.

ave_par *name*[a]/ *list-of-values*[b]/ ...

Example: `x/-0.5:0.025:0.5/`

An average-over parameter. The value lists of all such variables should be of equal size. Let a, b, ... be the average-over parameters, and let a_i, b_i, ..., $i = 1, \ldots, N$ be the components of their value lists. Then the simulation will be performed N times (for $i = 1, \ldots, N$), each time using the ith set of values for the average-over parameters: $a = a_i$, $b = b_i$... The data points obtained in these simulations, $x_k^{(i)}$, will be averaged as

$x_k = \sum_{i=1}^{N} w_i x_k^{(i)}$

Table A1 (*continued*)

	to produce the final data points x_k given at the simulation output. The weights of the distribution, w_i, are given by the variable **ave_wht**. The default value for this variable is $1/N$. If this is the desired distribution, **ave_wht** does not need to be declared.
fit_par	*name*[a,d][/*list-of-values*[b]/] ...

Example:	`Z_2  T2DQ_1_2_2`
	`A[1:10]  B[1:10]  C[1:10]`
	`a b x[1:2]/-1:0.1:1/ y[1:2]/-1:0.1:1/`

	Parameters to be optimized. The values of these variables are changed by the fitting routine during the optimization. Specifying a fit parameter with a *list-of-values*, sets up a grid of the initial points for the minimization. The grid is constructed as the Cartesian (direct) product of all specified *lists-of-values*. If a fit parameter is declared without a *list-of-values*, initial value for this parameter will be zero unless changed by a declaration of this parameter as a **variable**. The only difference in the input files for a parameter fit and a regular simulation of a certain experiment is that in the former, one or more variables are declared as fit parameters. SPINEVOLUTION will look for the files with the data points to be fit and the files containing the weights of these data points. The files should be provided by the user and named accordingly (see the Naming Conventions section); the weights files are optional. The data in these files should be formatted as if produced by the same input file but without fitting. The weights files should also be in this format, except that each data point should be replaced with its weight. The default optimization method used in SPINEVOLUTION is NL2SOL [73,74], which is a Newton-type nonlinear least-squares algorithm. Alternatively, one can choose the MINPACK routine [75,76] (option –**lm**). The initial conditions grid can be used to perform a grid search, in conjunction with one of these methods, or by itself. See also optimization control options (Table A2) and variables (Table A3), and the Output and File Naming Conventions section.
variable	*name*[a,d] **=** *expression*[c,d] OR *filename* OR *names*[a,d]**~***filename*
Example:	`power_1_1_[1:100]=a*[0:99]+b`
	Declaration of a variable. The variable is re-calculated from its expression each time when fit, scan, or average-over parameters that it depends on are changed. If the variable does not depend on any such parameters, it remains constant after it was initialized. The variables are evaluated in the order of their declaration. If a file name is given, declarations in the file should be given one per line as *name*[a,d] **=** *expression* [c,d]. In the third format, the *names* should be given in the vectorized notation, and the file should contain the list of the expressions.
rowmatrix	*name*[a]/*list-of-values*[b]/
Example:	`A/-0.5:0.025:0.5/`
	Declares a row matrix.
colmatrix	*name*[a]/*list-of-values*[b]/
Example:	`B/X.dat/`
	Declares a column matrix.
matrix	*name*[a]/*list-of-values*[b]; *list-of-values*[b]; ... / OR *name*[a]/*filename*/
Examples:	`M/1 2 3; 4 5 6/` `M/M.dat/`
	Declares a rectangular matrix. In the first format, the rows of the matrix are separated by the semicolon; in the second, the matrix is loaded from a file.
penalty	*expression*[c,d]
	The sum of squares of all penalty expressions is added to the weighted sum of the residual squares minimized by the fitting routine.
Options	
rho0	$[c_1 \ldots c_N]$ {**Ix**|**Iz**}
Example:	`1 0 0 0 Ix`
	Contributions to the initial density matrix from each spin in the system, followed by the operator type: $\rho_0 = \sum_{k=1}^{N} c_k I_{kz}$ or $\rho_0 = \sum_{k=1}^{N} c_k I_{kx}$ The default is $c_k = 1$ for all k.
observed_spins	*i j k* ... {**Ix**|**Iy**|**Iz**|**Ip**|**Ia**} OR *n* {**Inx**|**Iny**|**Inz**|**Inp**|**Ina**} OR *n* {**Fx**|**Fy**|**Fp**}
Examples:	`1 4 Iz` `1 Fp`
	In the first version of the format, the list of the spins to be observed is followed with the type of the observable. In the second, all spins on the channel number *n* are observed individually. In the third format, the total transverse polarization on channel *n* is observed. In all three cases, **p** (*p*lus) stands for both **x** and **y**, while **a** (*a*ll) stands for **x**, **y** and **z**. *n*-Quantum observables should also be available in the future versions of the program.

(*continued on next page*)

Table A1 (*continued*)

EulerAngles
Examples:
	[^] *filename* OR [^] *n* OR $\alpha\ \beta\ \gamma$
	`zcw55.dat`
	`^200`
	The file should contain the set of Euler angles (in radians) to be used for powder averaging. The Euler angles define the rotation from the crystallite frame to the rotor frame in MAS experiments, (α_{CR}, β_{CR}, γ_{CR}), or to the laboratory frame in static experiments, (α_{CL}, β_{CL}, γ_{CL}). The **z**-axis of the rotor-fixed frame is aligned with the axis of the rotor, the **z**-axis of the laboratory frame—with B_0. In the file, the angles should be given one orientation per line, ordered as $\alpha\ \beta\ \gamma\ w$ for three-angle sets, and $\alpha\ \beta\ w$—for two-angle sets, with w being the weight of the point. In the case of a two-angle set, additional averaging over the third angle can be requested at the **n_gamma** line. If a single number n is given at the **EulerAngles** line, it is interpreted as number of points in the β-only angle set (which is generated automatically in this case). In this case, no averaging is performed over α, while additional (independent) averaging over γ can still be requested at the next line. When an automatically generated angle set is used for powder averaging in systems with orientational symmetry, one can use the command line options **–oct** and **–hemi** to specify this symmetry.
	Prefixing the *filename* or n with the circumflex character (^) effectively swaps α and γ. The three columns of the two-angle file are understood in this case as $\gamma\ \beta\ w$, while additional averaging, if requested at the **n_gamma** line, is performed over α. Similarly, if ^n is given, γ will be kept constant, while the number of crystallites given at the next line will be interpreted as the number of steps in α to go through.
	For a single crystallite calculation, the orientation of the crystallite is specified as $\alpha\ \beta\ \gamma$ directly on the line (the angles should be given in *degrees* in this case).

n_gamma	n The total number of values of the Euler angle γ to average over (in addition to the averaging requested at the **EulerAngles** line). If $n > 1$, the program may choose a larger number (if this speeds up the calculation). The default value is 1. See also the circumflex switch option for the **EulerAngles** line.
line_broaden(Hz)	Lb_1 [Gb_1] (for 1D experiments) Lb_1 Lb_2 [Gb_1 Gb_2] (for 2D experiments) Line broadening in each dimension. *Lb* — Lorentzian, *Gb* — Gaussian. The default values are all zero.
zerofill	N_1 [N_2] Zero-fill dimension i up to at least N_i points.
FFT_dimension *Example:*	[**1**[**c**]] [**2**] [**ppm**] `1c 2 ppm` Dimensions to be Fourier-transformed on the output. If the option **ppm** is specified, the first (labeling) column of the output data file will be given in ppm units rather than kHz. The option **c** invokes the cosine transform instead of the regular FT in the first dimension, resulting in the pure phase spectra. Since the cosine transform does not distinguish positive and negative frequencies, the spectrum should be recorded in the frame where all observed frequencies have the same sign. Furthermore, since the cosine transform produces a symmetric spectrum, only half of it is saved in the output file(s).
options *Example:*	$opt_1\ opt_2\ \ldots$ `-hemi` Command line options can be given here in the same way as at the command line itself. The options given on this line are set prior to the ones given at the command line itself, so that the latter will override the former in case of a conflict. This line is optional.

[a] *name* Any name composed of alphanumerical characters and beginning with a letter; internal variables (i.e., those with predefined meanings and names) additionally contain at least one underscore character to distinguish them from the user-defined variables.

[b] *list-of-values* is a MATLAB-style notation for a numerical vector. It can be constructed of explicit numerical values, value ranges, or both. The format of a value range is *start_value*:*step*:*end_value*. A unit *step* can be omitted, so that `0:1:100` is equivalent to `0:100`. For example, `2 3.5:-1:1 0` is a list-of-values composed of the following five numbers: `2 3.5 2.5 1.5 0`. The following notation is used to generate n copies of the same value: *value*:0:*n*. The list of values will be loaded from a file if the file name is specified instead of the numerical constants.

[c] *expression* is an algebraic expression composed of numerical constants, variables, the **pi** constant, and common mathematical functions (see Table A4).

[d] Vectorized notation is allowed. The notation is essentially a macro of the form *A*[*list-of-x-values*]*B*[*list-of-y-values*]*C*..., where *A*, *B*, *C* ... are any groups of characters. The macro is expanded as $Ax_1By_1C\ldots$, ... $Ax_NBy_NC\ldots$, where $x_1 \ldots x_N$ and $y_1 \ldots y_N$ are the elements from the *list-of-values*. For example, a single construct `variable x[0:100]=[1:101]`, actually stands for 101 declarations.

[e] Any line in the file may be (optionally) ended with a comment that starts with a non-numeric character, e.g., the chemical name of the atom. An entire line will be considered a comment if it begins with a non-numeric character.

[f] Tensor parameters are defined according to Eqs. (17)–(19). Euler angles (PC) should be given in degrees.

[g] A sequence of nuclei such as `2 3 4 5` can be specified as `2:5`. A reverse order is also possible: the sequence `5 4 3 2` can be specified as `5:2`.

Table A2
Command line options

Output control	
–**h** *keyword*	Search the manual for the information on the *keyword*. The *keyword* can be a part of a main input file line header, an internal variable, an option, or a topic. When this option is given, the input file name may be omitted. The option is currently not functional.
–**autoconfig**	Measures and tunes the performance of various algorithms. The results are saved into `spinev.cfg` file, which should be placed into the SPINEVOLUTION home directory. Usage: `spinev -autoconfig`
–**t**	Print the results of the calculation to the terminal instead of the disk file(s).
–**s**	Show the experiment, but do not perform the simulation. Among other useful diagnostic information, the option displays all interactions present in the system and all the pulse sequences as they were interpreted and processed by the program. We recommend running every new input file with this option prior to performing the actual simulation. The option is also useful for converting atomic coordinates to the dipolar tensor parameters, which could be used, for example, to construct CSA tensors that have known orientation with respect to the chemical bonds in the molecule, or as input parameters to run the same simulation with SIMPSON.
–**n***name*	Use *name* instead of the name of the main input file to construct the names of the output files (*name*_**re.dat**, *name*.**par**, etc).
–**to**	Transpose the output data matrices (with respect to the default format).
–**dw***nm*...	Include the pulse sequences *n*, *m*, ... in the calculation of the time variable that makes up the first column of the output file and labels the data in this dimension. Only the first sequence of the dimension is included by default.
–**pwr**[*n*]	Write the powers of all pulses on channel *n* into *name*.**pwr**. The option can be used only in addition to the –**s** option. Normally, the output produced by –**pwr** has two columns: time at the start of a pulse (measured from the start of the experiment), and the RF power during the pulse. Use –**x0** to suppress the first column and obtain a file suitable for use as a power shape pulse program.
–**s0**	Suppress diagnostic suggestions by the program.
–**x0**	Suppress the first (labeling) column in the output data.
–**v***n*	Set the level of verboseness of the output to the terminal during the calculation. *n* should be an integer between 0 and 4. The default level is set according to the size of the problem.
–**vclk**	Print the total CPU time taken by the simulation.
–**vv**	Verbalize the values of all variables during the simulation.
–**vp**	Verbalize the values of all parameters (active variables) during the simulation.
–**vr**	Verbalize the Fourier coefficients of the dipolar, CSA, and quadrupolar interactions for each crystallite during the simulation. (Eq. (30))
–**ah***s*	Compute and print to the terminal the (exact) average Hamiltonian for one period, $n^{(s)}t_{\mathrm{seq}}^{(s)}$, of the pulse sequence *s*.
–**decimal***n*	Set the number of digits of the floating-point numbers written to the output files.
–**convert**	Converts *any* text file produced by a text editor that does not comply with the Unix convention for the new line notation. Executing `spinev` *filename* `-convert` overwrites the file *filename* by the converted file, where all CR characters have been either removed, or converted to the LF characters.
Input control	
–**pc***n*	Set the basic phase shift for the phase cycling to 360°/*n*. The default value of *n* is 4.
–**id***X*	Set the "maximum power" level to *X* kHz. All pulses given with power larger than *X* will be treated as ideal delta pulses; the default value is 499 kHz.
–**rmin***r*	Set the smallest admissible distance between two nuclei in the spin system, Å; the default is 0.1 Å.
–**oct** –**hemi**	Restrict the powder averaging to an octant or a hemisphere. The options can be used only with automatically generated crystallite sets, i.e., when the format [^]*n* is used at the **EulerAngles** line.
–**euler***file*	The Euler angles file to use for powder averaging. If this option is given, the **EulerAngles** line in the input file is ignored.
–**gamma***n*	The number of γ angles to step through in the powder average. This option overrides the number of γ-steps given at the **n_gamma** line.
–**var***X*	Declare a variable (or a set of variables if *X* is a file name). This is equivalent to adding **variable** *X* line to the Variables section of the main input file. The variables declared at the command line are calculated before the ones declared in the main input file.
Process control	
–**oes**	Observe each step, i.e., calculate the requested observable(s) after each pulse during the experiment. The pulse sequence of the experiment must be zero-dimensional. This option adds another dimension to the experiment.

(*continued on next page*)

Table A2 (*continued*)

–**split***n*	Carry out the simulation via *n* independent processes. This is accomplished by dividing the angle set for the powder averaging into *n* parts and starting a separate process for each part. The option is to be used on a multi-processor computer or a computer cluster (which has to be configured to support automatic process migration). When there is strong disparity between the processors on the cluster, it may be helpful to set *n* much larger than the total number of CPU's in the cluster.
–**savemem**	Use algorithms that economize computer memory usage, possibly sacrificing some efficiency.
–**d0**	Disable propagator diagonalization for the density matrix propagation; use only regular matrix multiplication
–**lv0**	Disable all Liouville space based algorithms
–**m0**	Disable the g-COMPUTE algorithm and use matrix multiplication for propagation
–**t0**	Disable the "turbo" algorithm, which uses the elementary propagators to construct the RF cycle propagators. This is occasionally helpful in situations when accuracy problems are observed.
–**gs***n*	Use *n* slices for PFG averaging (*n* must be odd); the default is 41.
–**sz***n*	When g-COMPUTE method is used, the signal is computed in the frequency domain. While the spectral width in such a calculation is fixed by the dwell time (i.e., by the pulse sequence), the number of the frequency bins can be chosen independently, according to the required precision. This number is set to 2^n by the –**sz***n* option.
–**lp***X*	Set the low-pass filter for the acquisition during g-COMPUTE to *X* kHz. By default, the filter is turned off.
–fft1	By default, the first point of the signal is divided by 2 before its Fourier transform is computed. The option prevents this behavior.
–sp*c*[*i*[,*j*]]	Scale the signal (all data points obtained in the experiment) to make the value of the *c*-component ($c = \mathbf{x},\mathbf{y},\mathbf{z}$) of the data point (*i*, *j*) equal 1; the scaling is applied before FFT. Example: `-spx0,0` (which is the same as `-spx`).
–**dt***X*	Set $dt_{\max}$ to *X* microseconds. $dt_{\max}$ is the maximum allowed step that may be taken during the integration of the equation of motion. Whenever the integration is carried out through the calculation of the cumulative products of $\exp(-iH_k\Delta t_k)$ or $\exp(-iL_k\Delta t_k)$ factors, $dt_{\max}$ sets the upper limit to the step size Δt_k. In MAS experiments, the default value of $dt_{\max}$ is 2 μs. One may want to change this setting, for example, to verify convergence or to speed up the calculation. The steps of 1–5 μs give reasonably accurate results for most MAS experiments. The default $dt_{\max}$ for static experiments is infinity, which sometimes has to be changed to a finite value in the simulations that involve relaxation.
Optimization control	
–**fn***fname*	Specifies the name of the fit data set; *fname* will be used instead of the name of the main input file to construct the names of the input files for data fitting (*fname*_**re.wht**, *fname*_**re.fit**, etc.).
–fb*bname*	Specifies the name of the batch-fit file and turns on the batch fit mode. The file *bname* should contain the list of names of the fit data sets (as in the **–fn** option) that have to be fit. The same names will be used to save the results (the **.par** and **.jnl** files). The initialization of the fit parameters during a batch fit is performed as follows. If a grid of initial conditions is specified for the fit parameters, then, for each fit data set, the residual sum of squares (RSS) is evaluated at all grid points. The point with the lowest RSS found is used as the initial condition for the optimization of this data set. Instead of, or in addition to this initialization from the grid, individual initialization for each fit data set can be used: see **–binit** option. If neither grid, nor **–binit** option are specified, the same starting point is used for all fits.
–binit	Turns on the batch initialization mode. The option can be used only in the batch fit mode (i.e. together with the **–fb** option). In this mode, the file *fname***.par** is loaded for each fit data set *fname* prior to fitting the data from this set. The file may contain initializing expressions for any fit parameters and constants. On the completion of the fit, the file is overwritten with the results.
–**lm**	Use Levenberg–Marquardt (LM) method for data fitting (which is usually inferior to the default method, NL2SOL). The LM implementation used in SPINEVOLUTION is a combination of MINPACK codes developed at Argonne National Laboratory [76].
–**powell**	Use Powell method for data fitting, which is usually much slower than the default method.
–**vf**	Verbalize intermediate results during data fitting.
–**vrss**	Print the residual sum of squares to the terminal whenever this function is evaluated.
–**f0**	Do not perform the optimization but compute the residual sum of squares.
–po	The "penalties only" option, which directs the program to minimize the sum of squares of the penalties only (ignoring the **.fit** files if any).
–**l0**	Suppress checking the correctness of the first (labeling) column in the **.fit** and **.wht** files.
–**pert***x*	When a local minimum is reached, perturb each fit parameter by adding a uniformly distributed random number from the $[-x,x]$ interval and continue the optimization.
–**rpert***x*	When a local minimum is reached, perturb each fit parameter a_i by adding a uniformly distributed random number from the $[-x\lvert a_i\rvert, x\lvert a_i\rvert]$ interval and continue the optimization.

Table A2 (*continued*)

–**shake***n*	Perform *n* "shake-downs" (see the previous two options) before accepting the minimum. The options –**rpert** and –**pert** specify the exact mode in which the parameters are perturbed. The default value of *n* is 0. The default mode is –**rpert** with $x = 0.1$.
–**covmat**	Compute the covariance matrix for the fit parameters at the minimum. The results will be saved in the file *name*.**cov**
–**confint**[*p*]	Compute the *p*-level confidence intervals for each fit parameter. The default *p* value is 95%. The results will be saved in the file *name*.**cls**

Table A3
Internal variables[a]

The system	
H1_freq	Spectrometer proton frequency, MHz
spinning_freq	Spinning frequency, kHz
X_*i* **Y**_*i* **Z**_*i*	Atomic coordinates of the nuclei
cs_iso_*i* **cs_ani**_*i* **cs_asy**_*i* **cs_alpha**_*i* **cs_beta**_*i* **cs_gamma**_*i*	Chemical shift tensor ($\mathbf{\Omega}^i/2\pi$) parameters[b]
j_iso_*i*_*j* **j_ani**_*i*_*j* **j_asy**_*i*_*j* **j_alpha**_*i*_*j* **j_beta**_*i*_*j* **j_gamma**_*i*_*j*	*j*-coupling (J^{ij}) parameters[b]
quad_ani_*i* **quad_asy**_*i* **quad_alpha**_*i* **quad_beta**_*i* **quad_gamma**_*i*	Quadrupolar interaction tensor ($\chi_i \tilde{\mathbf{V}}^i$) parameters[b]
T1ZQ_*i*_*j*_*s* **T1SQ**_*i*_*s* **T1DQ**_*i*_*j*_*s* **T1XQ**_*s* **T2ZQ**_*i*_*j*_*s* **T2SQ**_*i*_*s* **T2DQ**_*i*_*j*_*s* **T2XQ**_*s*	Longitudinal and transverse relaxation times, ms: zero-quantum, single-quantum, double-quantum, and X-quantum (the last term denotes all other coherences). These relaxation times are defined as analogs of the T_1 and T_2 constants in the Bloch equations. Namely, the T1 variables are defined as one-halves of the inverse rate constants for the specified transitions (e.g., flip-flops of spins *i* and *j* in the case of T1ZQ_*i*_*j*_*s*) while the T2 variables are defined as the inverse decay rate constants of the corresponding coherences. Each elementary pulse sequence is assigned its own set of relaxation parameters.
relax_model_*s*	Relaxation model to be used for pulse sequence *s*. Values: 0—no relaxation, 1—the simple model, 2—the full model.
The Pulse Sequence	
pulse_*n*_*s*_*p*	Pulse duration, μs (for PFG pulses, *n* is the number of the channel G)
power_*n*_*s*_*p* **phase**_*n*_*s*_*p* **freq**_*n*_*s*_*p*	RF parameters: power, kHz; phase, degrees; frequency offset, kHz
grad_*s*_*p*	PFG, Gs/cm
grad_offs	Gradient slice offset, cm. Declaration of this variable directs the program to compute only one gradient slice (at the specified offset).
tsf_*s* **psf**_*n*_*s* **fsf**_*n*_*s* **frs**_*n*_*s* **phs**_*n*_*s*	Time, power and frequency offset scaling factors, and phase and frequency offset shifts for a given sequence. These variables modify the RF cycle of the sequence as a whole (rather than pulse by pulse): $t^n_{\text{pulse}}(s,p) = \mathbf{tsf}\_s \cdot \mathbf{pulse}\_n\_s\_p$ $\omega^n_{\text{RF}}(s,p)/2\pi = \mathbf{psf}\_n\_s \cdot \mathbf{power}\_n\_s\_p$ $\phi_n(s,p) = \mathbf{phase}\_n\_s\_p + \mathbf{phs}\_n\_s$ $\omega^n_{\text{off}}(s,p)/2\pi = \mathbf{fsf}\_n\_s \cdot \mathbf{freq}\_n\_s\_p + \mathbf{frs}\_n\_s$ The default value for the scaling factors is 1, and for the shifts is 0. Scaling the power does not affect whether the pulse is interpreted as ideal or not (see the **–id** option).

(*continued on next page*)

Table A3 (*continued*)

RF_swb_*i*_*s*	RF switchboard (1–on, 0–off); the default is 1. This variable may be used to manipulate the spins individually, which may be required, for example, to prepare some special initial state or to produce an ideal selective pulse. Currently, this option *cannot* be used for ideal pulses.
zfilter_*s*	If set to 1, prescribes to annihilate all non-diagonal elements of the density matrix during the zero-length sequence *s*.
select_*s*	A matrix specifying coherence orders to select during the zero-length sequence *s*. All other coherences are destroyed. Each column of the matrix selects a certain coherence order; the elements of the column specify the constituent coherence orders in each spin species. This is an alternative to explicit phase cycling for coherence pathway selection.
gsize_*s*	Group size for the sequence *s*
Other variables	
r_*k*	The *k*th bond length, Å; see the **bond_len_nuclei** line.
theta_*k*	The *k*th bond angle, degrees; see the **bond_ang_nuclei** line.
phi_*k*	The *k*th torsion (dihedral) angle, degrees; see the **tors_ang_nuclei** line.
group_*k*_**R** **group**_*k*_**T**	The Euler rotation angles, degrees, and the translation vector, Å, for the *k*th group; see the **groups_nuclei** line. These variables must be declared as row- or column-matrices.
alpha_CR **beta_CR** **gamma_CR**	The Euler angles (radians) defining the orientation of the (only) crystallite, or molecule, in the R or L frame. These variables can be used, for example, to perform powder averaging according to some analytically defined scheme.
ave_wht	Weights of the distribution (see **ave_par** in Table A1)
ppm_ref_offs_*n*	$\Delta\omega^n_{\mathrm{ref}}/2\pi$, kHz (Eq. (3))
elb_*d* **glb**_*d*	Exponential and Gaussian line broadening in dimension *d*; same variables as given at the **line_broaden**(**Hz**) line
sample_L	Effective sample length along the PFG axis, cm (Eq. (36)); the default is 1 cm.
signal_sf	Scaling factor of the output signal; the default is 1.
signal_ph	Phase correction of the output signal, degrees; the default is 0.
AFC_TOL **RFC_TOL** **XC_TOL** **XF_TOL**	The convergence tolerances for NL2SOL (the default optimization/non-linear regression routine): absolute function convergence, relative function convergence, convergence in the parameter space, false convergence [73,74].
fm_TOL	Levenberg–Marquardt optimization routine convergence tolerance
RDX_FDJ **RDX_FDC**	Relative changes in the fit parameters to use for the forward-difference calculations of the Jacobian matrix during the optimization (**FDJ**), and the covariance matrix computation (**FDC**) [73,74].

[a] In variables names: *i* and *j*—spins, *n*—channel, *s*—pulse sequence, *p*—pulse in a pulse sequence.
[b] The isotropic value of the tensor and its anisotropy are in kHz; the Euler angles (PC) are in degrees.

Table A4
Functions recognized in the Variables section

x^*y*	*x* to the *y*th power
pi	the π constant
f(*x*)	Elementary functions: **sqr, sqrt, sin, cos, tan, asin, acos, atan, cosh, sinh, tanh, exp, ln, log10**
abs(*x*)	Absolute value of *x*
sign(*x*)	The sign of *x*
step(*x*)	The step function
cheb*n*(*x*)	Chebyshev polynomial of order *n*
ceil(*x*)	The ceiling function: the smallest integer greater than, or equal to *x*
floor(*x*)	The floor function: the largest integer less than, or equal to *x*
round(*x*)	Round *x* to the nearest integer
trunc(*x*)	Truncate the fractional part of *x*
arg(*x*,*y*)	The argument of the complex number $x + \mathrm{i}y$
hypot(*x*,*y*)	The Euclidean distance function, $\sqrt{x^2 + y^2}$
remainder(*x*,*y*)	The remainder function, $x - ny$, where *n* is the integer nearest to the exact value of x/y; if $\lvert n - x/y\rvert = 1/2$ then *n* is even
negpen(*x*)	Negative penalty function: returns zero if $x \geqslant 0$, and x^2 if $x < 0$
mul(*X*, *Y*)	Elementwise multiplication of the matrices *X* and *Y*
div(*X*, *Y*)	Elementwise division of *X* by *Y*
transpose(*X*)	Transpose of *X*
reshape(*X*,*M*,*N*)	Reshapes matrix *X* into an *M* by *N* matrix

References

[1] A. Bennett, R.G. Griffin, S. Vega, Recoupling of homo- and heteronuclear dipolar interactions in rotating solids, NMR Basic Principles Progress 33 (1994) 1–77.
[2] S. Dusold, A. Sebald, Dipolar recoupling under magic-angle spinning conditions, Ann. Rep. NMR Spectr. 41 (2000) 185–264.
[3] M.M. Maricq, J.S. Waugh, NMR in rotating solids, J. Chem. Phys. 70 (1979) 3300–3310.
[4] U. Haeberlen, J.S. Waugh, Coherent averaging effects in magnetic resonance, Phys. Rev. 175 (1968) 453–467.
[5] M. Veshtort, R.G. Griffin, High-performance selective excitation pulses for solid- and liquid-state NMR spectroscopy, ChemPhysChem 5 (2004) 834–850.
[6] L. Frydman, A. Lupulescu, T. Scherf, Principles and features of single-scan two-dimensional NMR spectroscopy, J. Am. Chem. Soc. 125 (2003) 9204–9217.
[7] L. Frydman, T. Scherf, A. Lupulescu, The acquisition of multidimensional NMR spectra within a single scan, Proc. Natl. Acad. Sci. USA 99 (2002) 15858.
[8] A.A. Bothner-By, C. Naar-Colin, The proton magnetic resonance spectra of olefins. I. propene, butene-1, and hexene-1, J. Am. Chem. Soc. 83 (1961) 231.
[9] S.M. Castellano, A.A. Bothner-By, Analysis of NMR spectra by least squares, J. Chem. Phys. 41 (1964) 3863–3869.
[10] A.A. Bothner-By, S.M. Castellano, Computer Programs for Chemistry, Benjamin, New York, 1968.
[11] P. Meakin, J.P. Jesson, Computer simulation of multipulse and Fourier transform NMR experiments. I. Simulations using the Bloch equations, J. Magn. Reson. 10 (1973) 290–315.
[12] P. Meakin, J.P. Jesson, Computer simulation of multipulse and Fourier transform NMR experiments. II. Some simulations using the density matrix equation of motion, J. Magn. Reson. 11 (1973) 182–206.
[13] P. Meakin, J.P. Jesson, Computer simulation of multipulse and Fourier transform NMR experiments. III. Density matrix simulations of some repetitively pulsed experiments, J. Magn. Reson. 13 (1974) 354–371.
[14] P. Meakin, J.P. Jesson, Computer simulation of multipulse and Fourier transform NMR experiments. IV. Some simulations including the time development of the off-diagonal density matrix elements, J. Magn. Reson. 18 (1975) 411–426.
[15] G. Bodenhausen, R. Freeman, G.A. Morris, D.L. Turner, Proton-coupled carbon-13 J spectra in the presence of strong coupling. II, J. Magn. Reson. 28 (1977) 17–28.
[16] B.K. John, R.E.D. McClung, Computer simulation of multiple-pulse and two-dimensional FT NMR experiments, J. Magn. Reson. 58 (1984).
[17] D. Nettar, J.J. Villafranca, A Program for EPR Powder Spectrum Simulation, J. Magn. Reson. 64 (1985) 61–65.
[18] L.E. Kay, J.N. Scarsdale, D.R. Hare, J.H. Prestegard, Simulation of two-dimensional cross-relaxation spectra in strongly coupled spin systems, J. Magn. Reson. 68 (1986) 515–525.
[19] T.T. Nakashima, R.E.D. McClung, Simulation of two-dimensional NMR spectra using product operators in the spherical basis, J. Magn. Reson. 70 (1986) 187–203.
[20] H. Widmer, K. Wüthrich, Simulation of two-dimensional NMR experiments using numerical density matrix calculations, J. Magn. Reson. 70 (1986) 270–279.
[21] D. Piveteau, M.A. Delsuc, J.-Y. Lallemand, "BOWMAN": a versatile simulation program for high-resolution NMR multipulse experiments, J. Magn. Reson. 70 (1986) 290–294.
[22] I. Bock, P. Rösch, Simulation of the cross-peak fine structure in 2D NMR spectroscopy by analytical calculation of the density operator in a product operator basis, J. Magn. Reson. 74 (1987) 177–183.
[23] 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.
[24] W. Studer, SMART, a general purpose pulse experiment simulation program using numerical density matrix calculations, J. Magn. Reson. 1988 (1988) 424–438.
[25] 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.
[26] M. Ravikumar, R. Shukla, A.A. Bothner-By, Relaxation and dynamics of coupled spin systems subjected to continuous radio frequency fields, J. Chem. Phys. 95 (1991) 3092–3098.
[27] F.S. Debouregas, J.S. Waugh, ANTIOPE, a program for computer experiments on spin dynamics, J. Magn. Reson. 96 (1992) 280–289.
[28] A.T. Brünger, X-PLOR: Version 3.1: A System for X-ray Crystallography and NMR, Yale University Press, New Haven, Conn., 1992.
[29] P. Güntert, N. Schaefer, G. Otting, K. Wüthrich, POMA: a complete *Mathematica* implementation of the NMR product-operator formalism, J. Magn. Reson. A 101 (1993) 103–105.
[30] R.P. 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. A 101 (1993) 23–29.
[31] 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. A 106 (1994) 75–105.
[32] L. Zhu, B.R. Reid, An improved NOESY simulation program for partially relaxed spectra: BIRDER, J. Magn. Reson. B 106 (1995) 227–235.
[33] T. Allman, A.D. Bain, J.R. Garbow, SIMPLTN, a program for the simulation of pulse NMR spectra, J. Magn. Reson. A 123 (1996) 26–31.
[34] A. Jerschow, N. Müller, Efficient simulation of coherence transfer pathway selection by phase cycling and pulsed field gradients in NMR, J. Magn. Reson. 134 (1998).
[35] A. Görler, W. Gronwald, K.-P. Neidig, H.R. Kalbitzer, Computer assisted assignment of ^{13}C or ^{15}N edited 3D-NOESY-HSQC spectra using back calculated and experimental spectra, J. Magn. Reson. 137 (1999).
[36] G.H. Meresi, M. Cuperlovic, W.E. Palke, J.T. Gerig, Pulsed field gradients in simulations of one- and two-dimensional NMR spectra, J. Magn. Reson. 137 (1999) 186–195.
[37] M. Cuperlovic, G.H. Meresi, W.E. Palke, J.T. Gerig, Spin relaxation and chemical exchange in NMR simulations, J. Magn. Reson. 142 (2000) 11–23.
[38] P. Nicholas, D. Fushman, V. Ruchinsky, D. Cowburn, The virtual NMR spectrometer: a computer program for efficient simulation of NMR experiments involving pulsed field gradients, J. Magn. Reson. 145 (2000) 262–275.
[39] M. Bak, J. Rasmussen, N.C. Nielsen, SIMPSON: a general simulation program for solid-state NMR spectroscopy, J. Magn. Reson. 147 (2000) 296–330.
[40] W.B. Blanton, BlochLib: a fast NMR C++ tool kit, J. Magn. Reson. 162 (2003) 269–283.
[41] M. Helgstrand, P. Allard, QSim, a program for NMR simulations, J. Biomol. NMR 30 (2004) 71–80.
[42] J. Herzfeld, A. Berger, Sideband intensities in NMR spectra of sample spinning at the magic angle, J. Chem. Phys. 73 (1980) 6021–6030.
[43] H. Geen, R. Freeman, Band-selective radiofrequency pulses, J. Magn. Reson. 93 (1991) 93–141.
[44] P. Costa, B. Sun, R.G. Griffin, Rotational resonance tickling: accurate internuclear distance measurement in solids, J. Am. Chem. Soc. 119 (1997) 10821–10830.

[45] S. Ray, V. Ladizhansky, S. Vega, Simulation of CPMAS signals at high spinning speeds, J. Magn. Reson. 135 (1998) 427–434.
[46] M. Edén, Y.K. Lee, M.H. Levitt, Efficient simulation of periodic problems in NMR. Application to decoupling and rotational resonance, J. Magn. Reson. A 120 (1996) 56–71.
[47] M.H. Levitt, M. Edén, Numerical simulation of periodic nuclear magnetic resonance problems: fast calculation of carousel averages, Mol. Phys. 95 (1998) 879–890.
[48] T. Charpentier, C. Fermon, J. Virlet, Numerical and theoretical analysis of multiquantum magic-angle spinning experiments, J. Chem. Phys. 109 (1998) 3116–3130.
[49] M. Hohwy, H. Bildsøe, 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.
[50] U. Haeberlen, High Resolution NMR in Solids: Selective Averaging, Academic Press, New York, 1976.
[51] M. Mehring, Principles of High Resolution NMR in Solids, Springer-Verlag, Berlin, 1983.
[52] H.W. Spiess, Rotation of molecules and nuclear spin relaxation, in: P. Diehl, E. Fluck, R. Kosfeld (Eds.), Dynamic NMR Spectroscopy, Springer-Verlag, Berlin, 1978, pp. 59–213.
[53] R.R. Ernst, G. Bodenhausen, A. Wokaun, Principles of Nuclear Magnetic Resonance in One and Two Dimensions, Clarendon Press, Oxford, 1987.
[54] D.M. Grant, R.K. Harris (Eds.), Encyclopedia of Nuclear Magnetic Resonance, Wiley, New York, 1996.
[55] M.H. Levitt, The signs of frequencies and phases in NMR, J. Magn. Reson. 126 (1997) 164–182.
[56] M. Edén, Computer simulations in solid-state NMR. I. Spin dynamics theory, Conc. Magn. Reson. Part A 17 (2003) 117–154.
[57] A.R. Edmonds, Angular Momentum in Quantum Mechanics, Princeton University Press, Princeton, 1957.
[58] M.E. Rose, Elementary Theory of Angular Momentum, Wiley, New York, 1957.
[59] D.M. Brink, G.R. Satchler, Angular Momentum, Clarendon Press, Oxford, 1968.
[60] S.K. Zaremba, Good lattice points, discrepancy, and numerical integration, Ann. Mat. Pura. Appl. 73 (1966) 293–317.
[61] H. Conroy, Molecular Schroedinger equation. VII. A new method for the evaluation of multidimensional integrals, J. Chem. Phys. 47 (1967) 5307–5318.
[62] V.B. Cheng, H.H. Suzukawa Jr., M. Wolfsberg, Investigations of a nonrandom numerical method for multidimensional integration, J. Chem. Phys. 59 (1973) 3992–3999.
[63] D.W. Alderman, M.S. Solum, D.M. Grant, Methods for analyzing spectroscopic line shapes. NMR solid powder patterns, J. Chem. Phys. 84 (1986) 3717–3725.
[64] M. Bak, N.C. Nielsen, REPULSION, a novel approach to efficient powder averaging in solid-state NMR, J. Magn. Reson. 125 (1997) 132–139.
[65] M. Edén, M.H. Levitt, Computation of orientational averages in solid state NMR by Gaussian spherical quadrature, J. Magn. Reson. 132 (1998) 220–239.
[66] W.P. Aue, E. Bartholdi, R.R. Ernst, Two-dimensional spectroscopy. Application to nuclear magnetic resonance, J. Chem. Phys. 64 (1976) 2229–2246.
[67] W.B. Gragg, The QR algorithm for unitary Heisenberg matrices, J. Comput. Appl. Math. 16 (1986) 1–8.
[68] W.B. Gragg, L. Reichel, A divide and conquer method for unitary and orthogonal eigenproblems, Numer. Math. 57 (1990) 695–718.
[69] D.C. Sorensen, Implicit application of polynomial filters in a K-Step Arnoldi method, SIAM J. Matrix Anal. Appl. 13 (1992) 357–385.
[70] R.B. Lehoucq, D.C. Sorensen, C. Yang, ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods, SIAM, Philadelphia, PA, 1998.
[71] R.C. Whaley, A. Petitet, J. Dongarra, Automated empirical optimizations of software and the ATLAS project, Par. Comput. 27 (2001) 3–35.
[72] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S.J. Hammarling, A. McKenney, S. Ostrouchov, D.C. Sorensen, LAPACK Users' Guide, SIAM, Philadelphia, PA, 1995.
[73] J.E. Dennis Jr., D.M. Gay, R.E. Welsch, An adaptive nonlinear least-squares algorithm, ACM Trans. Math. Soft. 7 (1981) 348–368.
[74] J.E. Dennis Jr., D.M. Gay, R.E. Welsch, ALGORITHM 573 NL2SOL—An adaptive nonlinear least-squares algorithm, ACM Trans. Math. Soft. 7 (1981) 369–383.
[75] J.J. Moré, The Levenberg–Marquardt algorithm implementation and theory, in: G.A. Watson (Ed.), Numerical Analysis, Springer-Verlag, Berlin, 1977.
[76] J.J. Moré, D.C. Sorensen, K.E. Hillstrom, B.S. Garbow, The MINPACK Project, in: W.J. Cowell (Ed.), Sources and Development of Mathematical Software, Prentice-Hall, Englewoods, Cliffs, NJ, 1984.
[77] P.N. Swarztrauber, Vectorising the FFTs, in: G. Rodrigue (Ed.), Parallel Computations, Academic Press, New York, 1982, pp. 51–83.
[78] M. Hohwy, C.M. Rienstra, C.P. Jaroniec, R.G. Griffin, Fivefold symmetric homonuclear dipolar recoupling in rotating solids: application to double quantum spectroscopy, J. Chem. Phys. 110 (1999) 7983–7992.
[79] A.E. Bennett, C.M. Rienstra, M. Auger, K.V. Lakshmi, R.G. Griffin, Heteronuclear decoupling in rotating solids, J. Chem. Phys. 103 (1995) 6951–6958.
[80] T. Gullion, J. Schaefer, Rotational echo double resonance NMR, J. Magn. Reson. 81 (1989) 196–200.
[81] V. Ladizhansky, M. Veshtort, R.G. Griffin, NMR determination of the torsion angle ψ in α-helical peptides and proteins: the HCCN dipolar correlation experiment, J. Magn. Reson. 154 (2002) 317–324.
[82] P. Tekely, J. Brondeau, K. Elbayed, A. Retournard, D. Canet, A simple pulse train, using 90° hard pulses, for selective excitation in high-resolution solid-state NMR, J. Magn. Reson. 80 (1988) 509–516.
[83] N.C. Nielsen, H. Bildsoe, H.J. Jakobsen, M.H. Levitt, Double-quantum homonuclear rotary resonance: efficient dipolar recovery in magic-angle spinning nuclear magnetic resonance, J. Chem. Phys. 101 (1994) 1806–1812.
[84] B.J. van Rossum, C.P. de Groot, V. Ladizhansky, S. Vega, H.J.M. de Groot, A method for measuring heteronuclear (^{1}H–^{13}C) distances in high speed MAS NMR, J. Am. Chem. Soc. 122 (2000) 3465–3472.
[85] B. Sun, P. Costa, D. Kosciko, P.T. Lansbury Jr., R.G. Griffin, Internuclear distance measurements in solid state nuclear magnetic resonance: dipolar recoupling via rotor sychronized spin locking, J. Chem. Phys. 102 (1995) 702–707.
[86] A.E. Bennett, J. Ok, R.G. Griffin, S. Vega, Chemical shift correlation spectroscopy in rotating solids: longitudinaal exchange and RF driven recoupling, J. Chem. Phys. 96 (1992) 8624–8627.
[87] T. Gullion, D.B. Baker, M.S. Conradi, New, compensated Carr–Prcell sequences, J. Magn. Res. 89 (1990) 479–484.
[88] H.Y. Carr, E.M. Purcell, Effects of diffusion on free precession in nuclear magnetic resonance experiments, Phys. Rev. 94 (1954) 630–638.
[89] S. Meiboom, G. D, Modified spin-echo method for measuring nuclear relaxation times, Rev. Sci. Instr. 29 (1958) 688–691.
[90] E. Kupce, J. Boyd, I.D. Campbell, Short selective pulses for biochemical applications, J. Magn. Reson. B 106 (1995) 300–303.